конспект программиста

В.А. Будилов

Практические занятия по PHP4

КРАТКИЙ КУРС

Наука и Техника
ИЗДАТЕЛЬСТВО

В книге изложены основы применения препроцессора языка PHP на практических примерах создания Web-страниц. Материал изложен с нарастанием сложности. Приведен справочник по командам PHP. Прилагается CD с примерами и программами.

В.А. Будилов

Практические занятия по PHP4

Наука и Техника
Санкт-Петербург
2001

Практические занятия по PHP4. Будилов В.А. — СПб: Наука и Техника, 2001. — 352 стр. с ил. Прилагается CD.

ISBN 5-94387-017-2

Под редакцией С.Л. Корякина-Черняка, члена Международной академии информационных процессов и технологий

Серия «Конспект программиста»

В книге на примерах рассмотрены основы использования языка PHP — одного из наиболее популярных языков программирования, используемого при написании серверных программ-сценариев. Это означает, что PHP-программы способны инициировать запросы, отправлять их по сети Интернет другим программам, называемым серверами, ожидать, получать и обрабатывать ответы, полученные от программ-серверов. С помощью PHP можно легко создавать интерактивные динамические HTML-страницы.

Вы научитесь устанавливать и конфигурировать Web-сервер и PHP. Подробно, с примерами рассматриваются все основные элементы языка, основные приемы программирования на PHP версии 4. Приведен краткий справочник функций языка PHP.

К книге прилагается CD-диск, на котором собраны примеры программ, рассмотренных в книге, PHP4 для Windows и UNIX, некомпилированный дистрибутив PHP, несколько программ Web-серверов и другие полезные программы.

Книга совмещает в себе справочник по PHP, рассчитанный на профессионалов, и материалы для начинающих. Шаг за шагом читатель входит в увлекательный мир PHP-программирования для WEB.

(812) 567-70-25, (044) 559-27-40

ISBN 5-94387-017-2

ООО «Наука и техника».
Лицензия № 000350 от 23 декабря 1999 года.
198215, г. Санкт-Петербург, ул. Подводника Кузьмина, д. 46.
Подписано в печать 07.05.01. Формат 70×100/16.
Бумага газетная. Печать офсетная. Объем 22 п. л.
Тираж 4000 экз. Заказ № 216

Отпечатано с готовых диапозитивов
в ФГУП ордена Трудового Красного Знамени «Техническая книга»
Министерства Российской Федерации по делам печати,
телерадиовещания и средств массовых коммуникаций
198005, Санкт-Петербург, Измайловский пр., 29

ВВЕДЕНИЕ

Когда мы впервые пускаемся в путешествие по сети Интернет, нас удивляют и поражают новые ранее невиданные возможности, открывающиеся перед нами прямо на экране домашнего компьютера. Чтобы пуститься в это путешествие, нам необходим всего лишь компьютер и телефонная линия. Компьютер оснащен необходимым для его работы программным обеспечением и модемом, с помощью которого он может общаться с удаленными «братьями по разуму», т.е. с подобными себе компьютерами. А чтобы общение компьютера с другими компьютерами было понятно для человека, разработаны специальные интерфейсы, облегчающие общение человека с компьютером и компьютера с человеком.

Самым распространенным средством для работы в Интернете в настоящее время представляются программы-броузеры, т.е. программы, позволяющие пользователю осуществлять непринужденное «брожение», «плавание» по сети Интернет, незаметно для него самого «перескакивая» из одного места («сайта») сети в другое, причем физически такие сайты могут быть отделены друг от друга расстояниями в десятки тысяч километров. Большие расстояния незаметны пользователю, на компьютере которого установлен броузер.

Существует несколько широкоизвестных программ-броузеров, кроме того, существует большое количество малоизвестных программ-броузеров и огромное количество программ, использующих стандартные возможности программ-броузеров. Общим для них всех является то, что эти программы устанавливаются на компьютере пользователя. Эти программы являются клиентскими программами. Это означает, что они способны инициировать запросы, отправлять их по сети Интернет другим программам, называемым серверами, ожидать, получать и обрабатывать ответы, полученные от программ-серверов.

Партнерами клиентских программ являются серверные программы, программы-сереверы. Программа-сервер «прослушивает» сеть (это может быть необязательно сеть, так как клиент и сервер могут быть установлены и на одной машине, и более того, одна и та же программа может быть одновременно как клиентом, так и сервером) в ожидании запроса клиента. После поступления такого запроса, сервер принимает его, обрабатывает и посылает ответ клиенту.

Примерами клиентских программ могут служить программы-броузеры. Однако, это не единственный вариант реализации клиентских приложений. Клиентскими программами являются, например, почтовые программы, такие как, например, Microsoft Outlook или Outlook Express и другие. Существует большое количество клиентов-игр и соответствующих им игровых серверов, немало встречается клиентов интернет-телефонии, работающих с бесплатными (временно) серверами, помогающими осуществлять телефонные звонки с домашнего компьютера в любую точку мира по номеру телефона. Существует большое число

программ для прослушивания радио- и видеоинформации в Интернет в реальном времени, использующих потоковые файлы, передаваемые сервером клиенту в режиме, напоминающем теле- или радиовещание.

Однако, более всего наше внимание привлекают программы-броузеры. Это можно объяснить тем, что информация, представляемая в броузере, главным образом является графической и текстовой информацией, и именно такая информация наиболее доступна для большого числа пользователей. Любой человек, даже немного знакомый с компьютером, может набрать текст, сохранить картинку, создать документ, составить отчет, сделать Web-сайт своими руками так, как ему это будет удобно. При этом он имеет дело главным образом с текстом, графикой, их сочетаниями, и, немного, с аудиоинформацией. И именно такой вид информации наиболее полно может быть представлен пользователю Интернет, если пользователь воспользуется именно программой-броузером.

Современные броузеры предоставляют пользователю довольно большие возможности. Основным типом документов, с которыми работает броузер, являются HTML-документы, т.е. документы, использующие язык разметки гипертекста, язык HTML. Основной функцией броузера, как программы-клиента, осуществляющей связь с сервером, является формирование запроса HTML-файла и получение этого файла с сервера. Наиболее распространенным (но не единственным) при передаче файлов сервером броузеру является протокол передачи гипертекста, протокол http. В заголовках передаваемых с применением этого протокола файлов содержится информация для броузера, согласно которой броузер может определить способ обработки получаемого файла.

После того (и в процессе того) как броузер получил запрашиваемый файл, он приступает к его обработке в соответствии с инструкциями (ярлыками) HTML, которые встроены непосредственно в HTML-файл. Такой файл главным образом состоит из текста, графических элементов (в виде встроенных страниц и назначенных цветов для элементов), причем как текст, так и графические элементы (картинки) могут быть отформатированы, а исходные данные (текст, графический файл) могут находиться как внутри полученного HTML-файла, так и храниться в виде самостоятельного файла в любом месте глобальной сети. В последнем случае HTML-файл будет содержать лишь ссылку на них. Броузер интерпретирует всю информацию, содержащуюся в полученном HTML-файле, и выводит на экран компьютера результат, который понятен конечному пользователю.

Основными элементами языка HTML являются ярлыки, которые представляют собой стнадартные слова, помещенные в скобки «<...>». Такие ярлыки используются для осуществления «статического» форматирования тексто-графической информации. Броузеры позволяют осуществлять динамическое изменение внешнего вида (и других компонентов) представляемой информации. Способы такого изменения можно описать с использованием языков сценариев. Сценарии могут быть написаны, например, на языке JavaScript, а текст сценария помещен непосредственно в тот же HTML-файл, что и исходная тексто-графическая информация. JavaScript позволяет создавать весьма сложные программы. Эти программы выполняются программой-броузером в режиме интерпретации. Этим обуславливается медлительность в их выполнении. Броузеры имеют возможности

выполнять и более совершенные программные модули, которые загружаются в виде Java-классов, элементов ActiveX, т.е. в виде готовых оттранслированных бинарных файлов программ. Однако, выполнение этих программ возможно лишь после того, как будет осуществлена установка этих программ в систему. Установка внешних программ на свой компьютер сопряжена с определенным риском, так как такая программа (в отличие от программы-сценария в чистом виде), потенциально может осуществить несанкционированный доступ к ресурсам системы. Такие элементы, как ActiveX и классы Java, являются платформенно независимыми в определенной степени модулями. Но и эта независимость оказывается зависимой от конкретной реализации программы-клиента, а также от претензий основных производителей программного обеспечения подобного рода, т.е. от того, в коей мере внешние продукты согласуются с подходами, принятыми в компаниях, например, Netscape или Microsoft.

HTML-файл хранится в памяти (например, на жестком диске) машины в каталоге, доступном web-серверу, и по запросу клиента посылается ему в исходном виде, т.е. в том виде, в каком он был первоначально создан и сохранен, без внесения предварительных (перед отсылкой) изменений. Этот файл может содержать инструкции для динамической обработки его содержания на клиенте, но сервер не будет иметь возможности изменить содержание отсылаемого HTML-файла.

Существуют возможности внести коррективы в отсылаемый клиенту HTML-файл. Для этого используются серверные скрипты. Существует множество вариантов создания серверных скриптов, для этого используются различные языки, в том числе для таких целей можно использовать любые стандартные языки, например, язык Java или C. Однако, есть возможность воспользоваться наиболее компактными языками, разработанными специально для написания программ серверных сценариев, и одним из таких языков является язык PHP, который становится все более популярным.

1

ЧТО ТАКОЕ PHP

Что такое PHP?

PHP — это один из наиболее популярных языков программирования, используемый при написании серверных программ-сценариев. Этот язык является весьма компактным (в смысле написания на нем программ, некоторые задачи (например, простые счетчики посещений и т.п.) могут быть решены средствами PHP буквально в две строчки. Особенностью этого языка является то, что коды PHP могут быть вставлены непосредственно в HTML-файл, т.е. конструкции языка PHP вставляются в исходную HTML-разметку подобно тому, как клиентские сценарии вставляются в ту же HTML-страницу. Это предоставляет дополнительные удобства при написании PHP сценариев.

Перед тем, как HTML-файл будет отправлен сервером по запросу клиента, необходимо выполнить PHP инструкции, вставленные в HTML-файл. Чтобы эти инструкции могли быть выполнены, необходимо на Web-сервере установить тот или иной вариант PHP. Поскольку после запроса клиента простой HTML-файл будет отправлен без изменений и предварительного анализа, необходимо заранее позаботиться о том, чтобы файл с функциями и инструкциями языка PHP перед отправкой клиенту был обработан на сервере. Для этого серверу необходимо знать, какой тип файлов перед отправкой клиенту следует подвергнуть обработке PHP интерпретатора. Для этого при установке и конфигурировании модуля PHP на сервере указывается список расширений файлов, обрабатываемых интерпретатором PHP. Этого мы еще коснемся в дальнейшем.

Изначально язык PHP появился в виде персонального проекта, разработанного Расмусом Лердорфом в первой половине девяностых годов. Благодаря быстрорастущей популярности язык неоднократно усовершенствовался, чтобы удовлетворять постоянно растущие потребности программистов и был почти полностью создан заново уже группой разработчиков. Дистрибутив распространяется бесплатно, имеются реализации языка для различных платформ. Основным преимуществом этого языка является то, что в нем использованы наиболее популярные при программировании серверных сценариев возможности нескольких широкоизвестных языков, в том числе таких, как Java, C, Perl. Структура программ весьма проста, и если новичок имел пусть небольшой опыт использования языков программирования, язык PHP не будет представлять для него большой трудности. Язык содержит в себе множество встроенных заранее определенных функций, выполняющих наиболее часто встречающиеся при программировании серверных задач процедур. Большое количество функций связано с работой с различными базами данных (MySQL, Oracle, Sybase, mSQL, Generic ODBC, PostgreSQL), организацией передачи данных по сетям, работой с файловой системой, работой с идентификаторами пользователя и т.п. Язык позволяет производить сложные математические вычисления, работать с почтой, регулярными выражениями, обрабатывать пользовательские формы и осуществлять множество других действий. Одним из факторов, делающим этот язык столь мощным средством программирования, является его ориентированность на решение конкретных задач, возникающих на серверной стороне.

Резюмируем сказанное еще раз. PHP — это серверный язык сценариев. Для того, чтобы программа-скрипт, написанная на PHP могла работать, необходимо установить и сконфигурировать PHP-интерпретатор на Web-сервере. При конфигурировании указываются каталоги сервера, где могут быть расположены PHP-файлы, а также расширения этих файлов. Файлы с указанными расширениями после обращения клиента о передаче ему таких файлов, будут предварительно обработаны PHP-интерпретатором. Чтобы иметь возможность создавать PHP-файлы необходимо удостовериться, обратившись к интернет-провайдеру, в том, что каталог, в котором расположены ваши HTML и PHP странички на Web-сервере провайдера, «обслуживается» интерпретатором PHP, т.е. что к вашему каталогу существует доступ для установленного на сервере провайдера PHP-интерпретатора, а также в том, что такой интерпретатор вообще установлен на сервере провайдера. Другой вариант решения вопроса о том, где можно интерпретировать серверные скрипты, состоит в том, чтобы самому стать администратором своего собственного web-сервера и установить на нем PHP-интерпретатор. Стандартными расширениями PHP-файлов являются расширения PHP, PHP3, PHP4, PHTML.

Наш первый PHP-файл

Инструкции языка PHP могут быть вставлены непосредственно в HTML-код стандартной web-страницы. Такой метод позволят с легкостью осуществлять возможность динамического изменения отсылаемой клиенту web-страницы перед тем, как измененный в соответствии с PHP инструкциями HTML-код будет отправлен клиенту (броузеру).

В качестве первого примера мы рассмотрим самый примитивный PHP файл (first.php):

```
<HTML>
<HEAD>
<TITLE>Наш первый PHP сценарий</TITLE>
</HEAD>
<BODY>
<CENTER>Эта страница создана с использованием PHP</CENTER>
<?
/* Знак "<?" обозначает начало PHP сценария */
echo ("Эта строка выведена с помощью PHP сценария");
/* Знак "?>" обозначает конец сценария */
?>
 </body>
</html>
```

Прежде чем продолжить работу с эти файлом, напомним, что HTML-ярлык <HTML> сообщает нам о том, что данный файл является (в основном) HTML-файлом, ярлык <head> содержит заголовок файла с его названием, указанным в ярлыке <title>, ярлык <body> содержит основную информацию, основные данные HTML-файла. PHP-код помещен между знаками <? и ?>. В нашем примере PHP помещены между /* и */. Функция ***echo*** используется для вывода содержимого аргумента.

Понятно, что такое использование PHP вряд ли можно назвать эффективным, так как вывод стационарного текста гораздо удобнее осуществлять средствами HTML, не прибегая к помощи PHP. Немного усложним файл, вставим в него возможность посылать пользователю текущую системную дату машины, на которой установлен сервер, для этого используем функцию ***date()***. Отметим, что идентификаторы переменных в PHP начинаются с символа $ (Примета гласит: «Приснился PHP — к деньгам»). Сохраним файл как first1.php.

Файл first1.php

```
<HTML>
<HEAD>
<TITLE>Наш первый PHP сценарий</TITLE>
</HEAD>
<BODY>
<CENTER>Эта страница создана с использованием PHP</CENTER>
<?
/* Знак "<?" обозначает начало PHP сценария */
echo ("Эта строка выведена с помощью PHP сценария");
  $today = date("Y-m-d");
  echo ("<CENTER>Сегодня $today.</CENTER>");
/* Знак "?>" обозначает конец сценария */
?>
</body>
</html>
```

Мы написали наш первый простой PHP сценарий. Давайте попробуем выполнить его.

Сохраним файл first1.php в произвольном месте на диске компьютера, откроем броузер, например, Microsoft Internet Explorer, и впишем в адресной строке путь и имя этого файла, как показано на рис. 1.1. И нажимаем клавишу Enter.

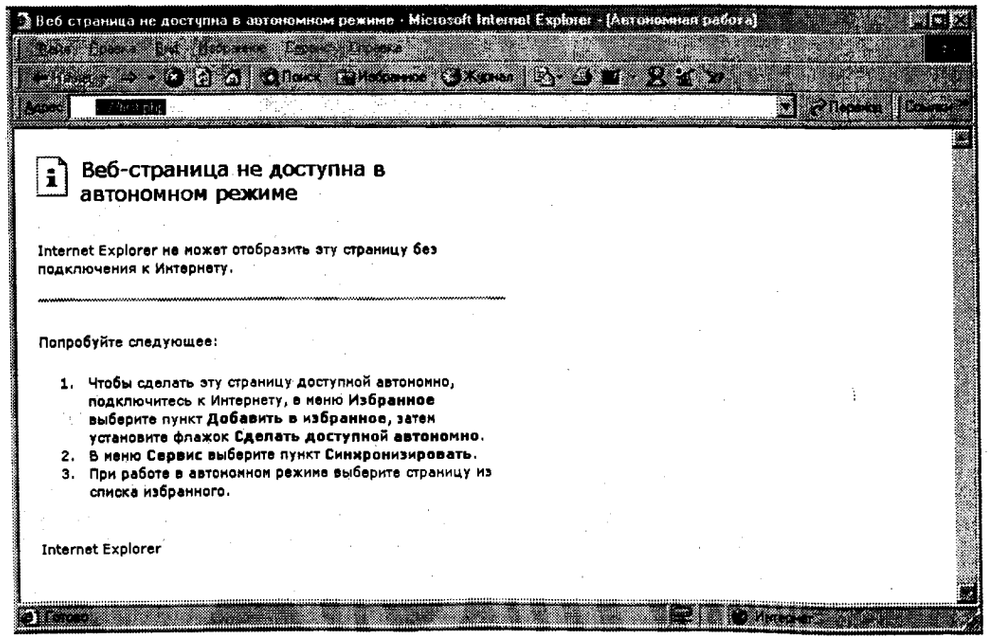


Рис. 1.1. *Готовимся к загрузке первого PHP-файла в броузер*

Но что-то не так... окно броузера остается пустым, а в заголовке в панели броузера содержится надпись «Сервер не найден» (рис. 1.2). Это и не удивительно, потому что для того, чтобы мы смогли получить полноценный HTML-файл, к тому же еще обработанный PHP-интерпретатором, исходный PHP-файл должен находиться в том каталоге, который отведен для PHP-файла каким-либо web-сервером. А мы «положили» наш первый PHP-файл first.php в такой каталог, который недоступен ни одному web-серверу.

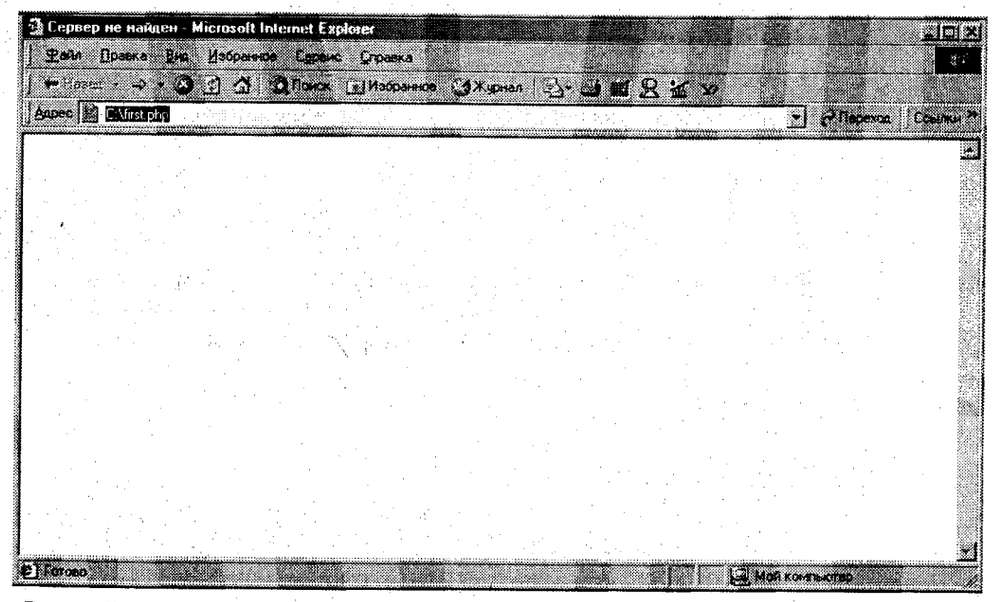


Рис. 1.2.

Как быть? Пусть мы не располагаем доступом к web-серверу, но мы все же можем интерпретировать наш файл first1.php. Для этого можно воспользоваться программой PHP.exe, работающей в DOS. Для этого запускаем сеанс DOS, а для удобства (это, естественно, совершенно необязательно) вызываем какую-либо оболочку для работы с файлами, например, Norton Commander. Находим каталог, в котором располагается программа php.exe, копируем в него наш файл first1.php и пишем команду DOS в виде

```
php first1.php
```

Нажимаем Enter (см. рис. 1.3).

Ваш компьютер слегка задумается, немного пошуршав винчестером, и выдаст что-то вроде того, что изображено на рис. 1.4. Это HTML-код, который мог бы быть отправлен клиенту (броузеру), если бы файл first1.php находился на сервере. «Абракадабра» появилась ввиду того, что на машине, которая использовалась при создании

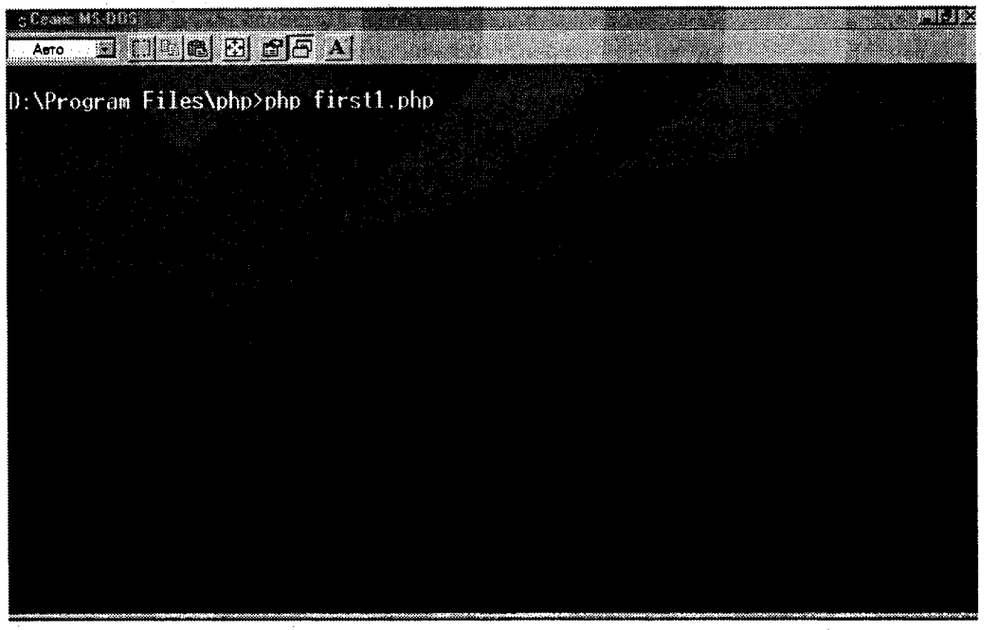


Рис. 1.3. *Запускаем PHP с файлом first1.php*

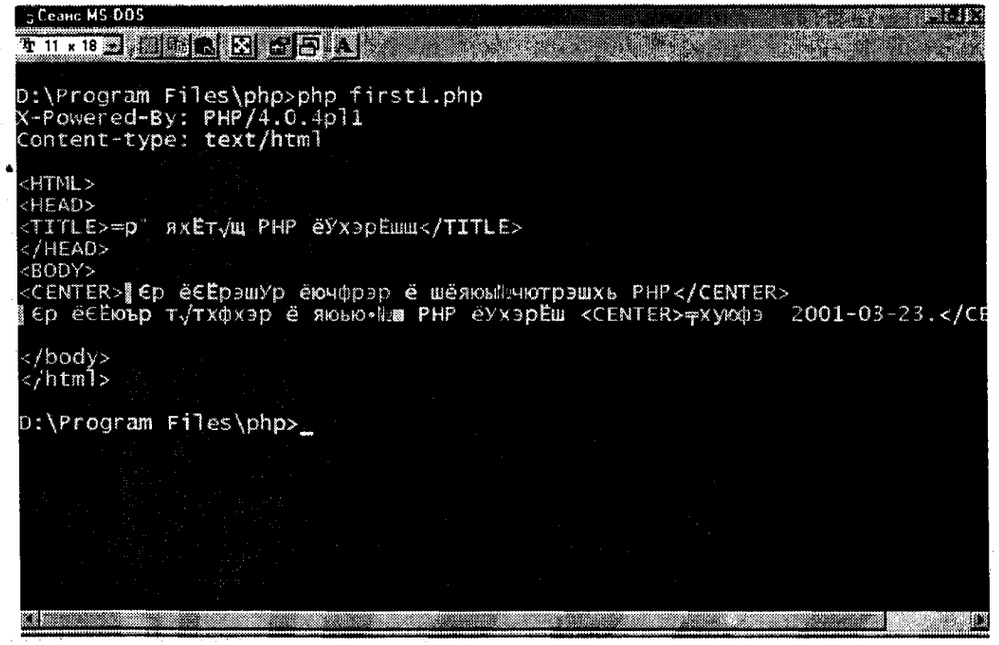


Рис. 1.4. *Файл обработан интерпретатором PHP*

рисунка, DOS не осуществляет перекодировку русских символов. При пересылке файла броузеру и его дальнейшей обработке этот недостаток устраняется.

Это наиболее простой способ проверки PHP-скриптов. Если скрипт содержит ошибки, то появятся сообщения об ошибках. Эти ошибки легко могут быть найдены и устранены (рис. 1.5).

Использование модуля php.exe — это простейший путь «применения» интерпретатора языка PHP для учебных целей. Однако этот способ не решает главной цели, т.е. созданные PHP-коды не будут отправлены программе-броузеру. Чтобы сгенерированный при помощи PHP код был отправлен броузеру, нам необходимо иметь web-сервер с установленным на нем PHP. В следующей главе мы расскажем о том, как можно установить персональный Web-сервер, установить на нем PHP и сконфигурировать его.

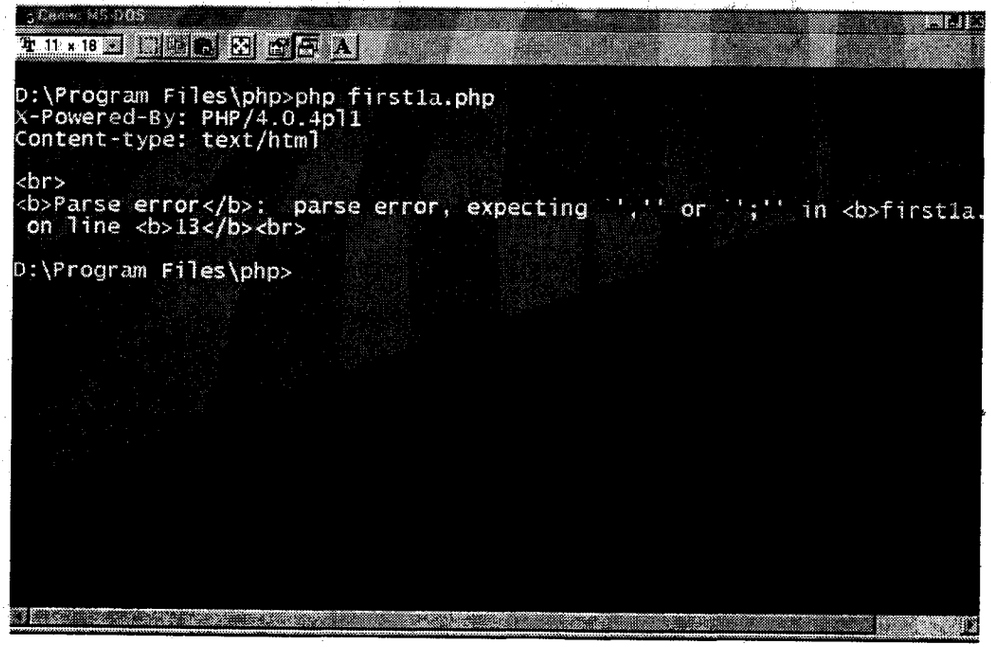


Рис. 1.5. *Сообщение об ошибке*

2

УСТАНОВКА WEB-СЕРВЕРА И PHP

- ***Что такое web-сервер***
- ***Становимся администратором персонального сервера***
- ***Устанавливаем персональный web-сервер***
- ***Настраиваем сервер***
- ***Устанавливаем PHP***
- ***Регистрируем PHP в windows***
- ***Проверяем работу PHP***

Установка web-сервера и PHP

Чтобы облегчить себе задачу по изучению языка PHP и начать создавать собственные PHP сценарии, установим на свой компьютер web-сервер. Персональный web-сервер входит в комплект Windows, поэтому нам далеко ходить не надо. Не будем останавливаться на вопросах установки web-сервера. Каждый может выбрать тот сервер какой ему больше понравится, какой наиболее всего удовлетворяет его требованиям и подходит к той операционной системе, которая используется на его компьютере. После установки сервера следует запомнить те каталоги, к которым открыт доступ серверу.

Существуют различные реализации интерпретатора PHP для разных платформ. Дистрибутивы распространяются бесплатно, их можно найти в Интернете по адресу http://www.php.net/downloads.php. Для того, чтобы установить PHP на компьютер, необходимо получить дистрибутив (например, из Интернета), распаковать его в пользователем выбранный директорий. Пусть устанавливается версия PHP для Windows, в этом случае важно иметь в виду, что она будет работать с такими серверами, как Personal Web Server, Internet Information Server, Apache 1.3.x Omni

HTTPd 2.0b1. После того, как файлы распакованы, находим файл php3-dist.ini и копируем его в тот каталог, где расположен Windows, предположим, в каталог Windows, при этом меняем имя файла на php.ini. После этого необходимо внести изменения в сам файл php.ini. Главное, что здесь будет необходимо сделать — это указать путь к файлам PHP. Остальные установки следует сделать такими, чтобы они максимально соответствовали вашим требованиям. Более подробно директивы файла php.ini мы рассмотрим далее в этой главе.

Затем переходим к ответственной части редактирования регистра Windows. Сделаем это с помощью системной утилиты Regedit.

Запускаем Regedit (Пуск → Выполнить → вводим «regedit») и находим HKEY_LOCAL_MACHINE (см. рис. 2.1).

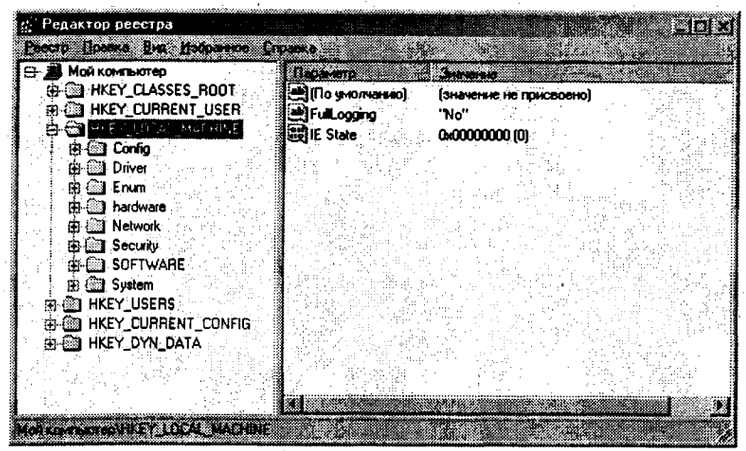


Рис. 2.1. *Вызываем редактор реестра Windows*

В правой панели редактора реестра переходим к /System/CurrentControlSet/Services/W3Svc /Parameters/ScriptMap.

Находим пункт Правка → Создать → Строковой параметр (New → (строковое значение) Value). Здесь мы задаем расширение, которое будет использовано для файлов, содержащих сценарии PHP, например, расширение .php или .phtm (рис. 2.2).

После того, как тип файлов задан, щелкаем двойным щелчком мыши на вновь созданном параметре .php, появляется диалоговое окно свойств этого параметра, в котором мы указываем полный путь к библиотеке для обработки файлов данного типа. Иными словами, мы указываем полный путь к месту расположения рабочих файлов PHP в нашей системе (рис. 2.3).

Определите расширение, которое вы хотите использовать для ваших сценариев PHP (например, .php3).

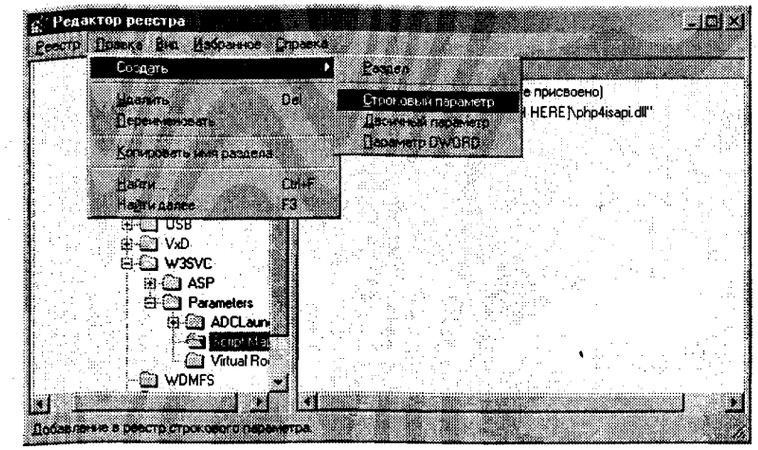


Рис. 2.2. *Указываем расширение файлов со скриптами PHP*

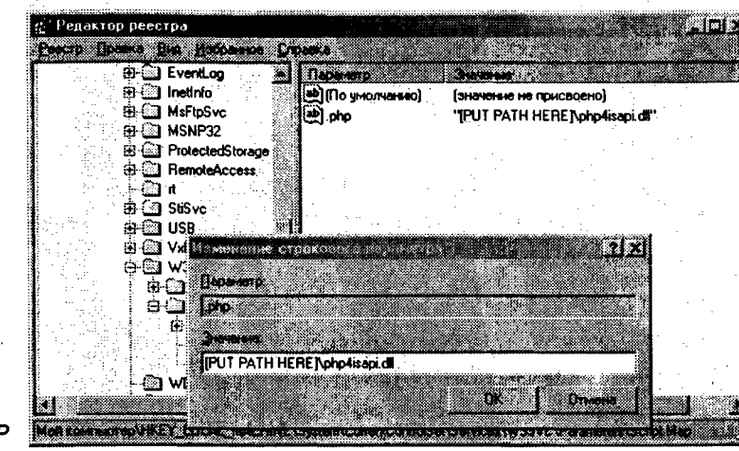


Рис. 2.3. *Указываем путь к файлам PHP*

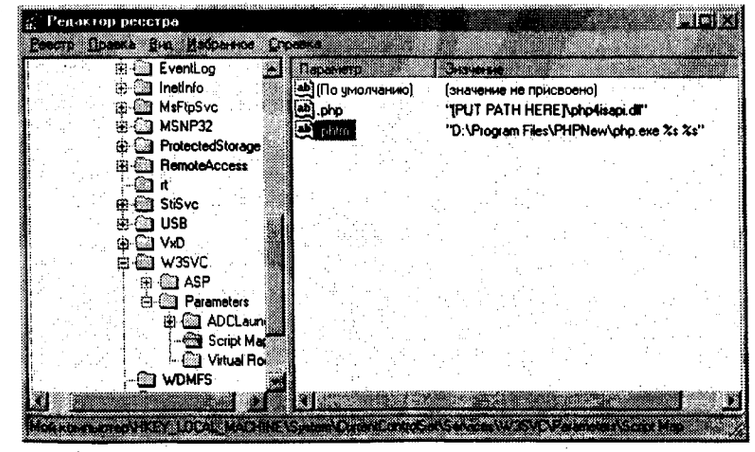


Рис. 2.4. *Указываем значения для всех расширений*

Если мы используем несколько различных расширений для PHP-файлов, то путь к файлам, используемым для обработки PHP-сценариев, необходимо указать для каждого расширения (рис 2.4).

Следующим шагом находим HKEY_CLASSES_ROOT, выбираем пункт меню Правка → Создать → Раздел (New → Key). Определим ключ для заданного выше расширения (например, .phtm) в виде .phtm. В правой панели окна дважды щелкнем «параметр по умолчанию» и введем значение в виде «phpfile» (рис. 2.5).

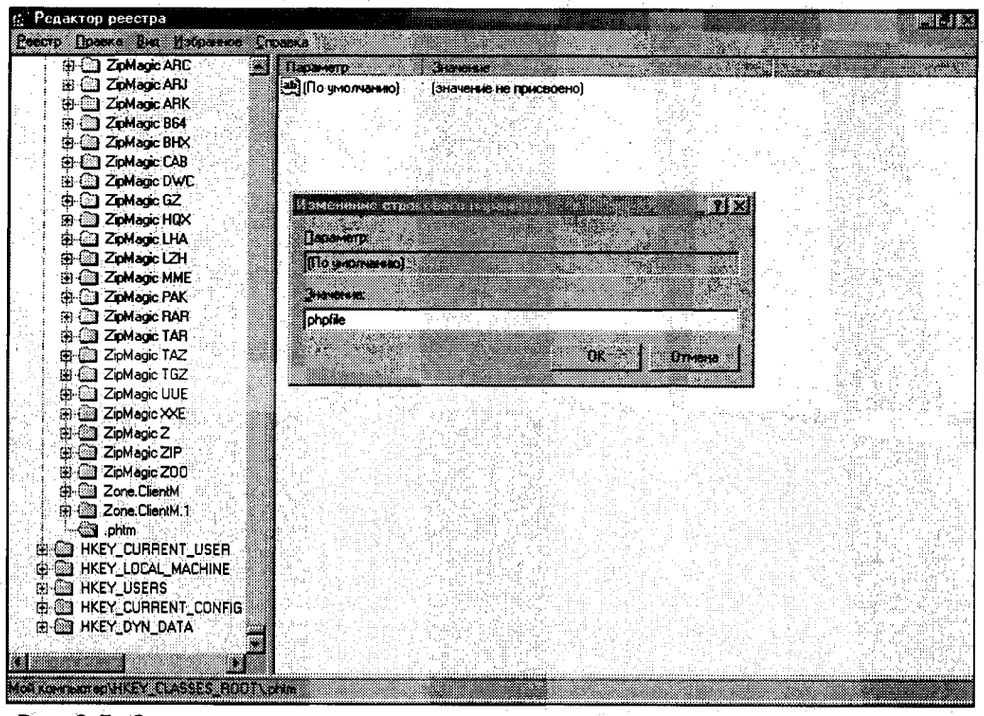


Рис. 2.5. *Создаем ключ*

Такие действия следует произвести для каждого установленного расширения.

Затем создаем новый раздел в HKEY_CLASSES_ROOT — раздел phpfile. Выбираем вновь созданный раздел phpfile и в правой части диалогового окна дважды щелкаем на «по умолчанию» («default value»), в появившемся окне вводим значение PHP Script (рис. 2.6).

В левой панели окна вызовем контекстное меню при выбранном разделе phpfile, для этого щелкнем правой кнопкой мыши на phpfile. В появившемся меню выберем Создать → Раздел. Назовем раздел Shell. Повторим операцию, выберем раздел Shell, щелкнем правой кнопкой мыши, в контекстном меню выберем Создать → Раздел, зададим имя

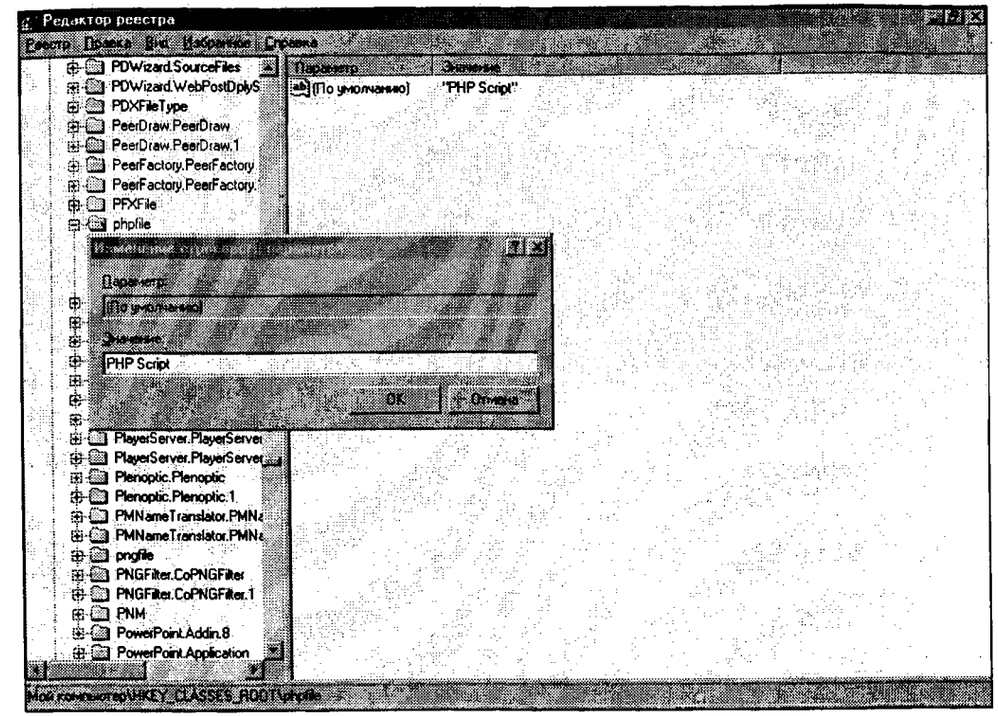


Рис. 2.6. *Раздел phpfile*

open. И еще раз повторим операцию, вызовем контекстное меню правым щелчком мыши при выбранном разделе open, выберем пункт Создать → Раздел, назовем раздел command.

Выделяем раздел command, в правой части окна дважды щелкаем на «значение по умолчанию», в появившемся окне вводим путь к файлу php.exe. Например: D:\Program Files\php\php.exe -q %1 (рис. 2.7).

Сейчас мы можем покинуть программу Regedit.

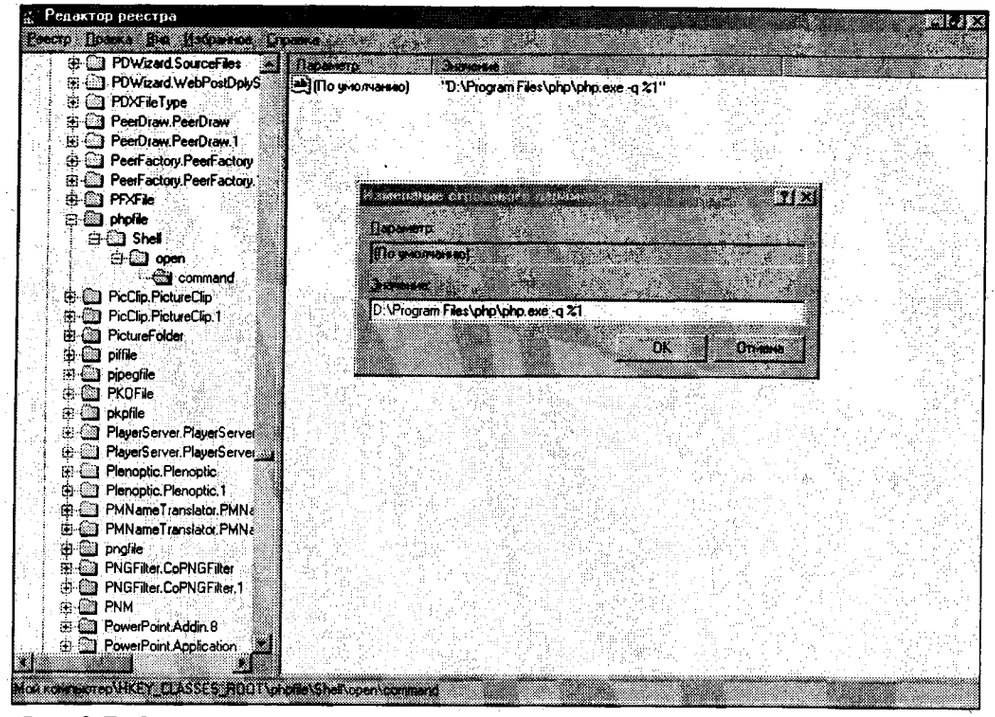


Рис. 2.7. *Завершающие шаги редактирования реестра*

Конфигурирование PHP

Важная информация о конфигурации PHP хранится в файле php.ini (или php3.ini для версии php3), о котором мы уже упоминали выше. Обращение к файлу php.ini происходит один раз при запуске сервера. В варианте CGI обращение к этому файлу будет осуществляться всякий раз при обращении к модулю php.exe. Для правильной работы PHP важно не пожалеть времени и усилий для того, чтобы наиболее оптимальным образом сконфигурировать файл php.ini. В этом файле содержится большое количество команд-директив, от них зависит способ работы php, а соответственно, и получаемый результат. Рассмотрим назначение отдельных команд файла php.ini подробнее.

Основные команды в файле php.ini

asp_tags (логическое значение)

Позволяет использование обрамляющих PHP-скрипт символов в виде <% %>, т.е. символов ASP-скриптов в дополнение к основным обозначениям начала и конца PHP-скрипта (<? и ?>).

auto_append_file — строковое значение

Определяет имя файла, который автоматически добавляется к основному файлу с осуществлением его проверки. Файл включается так же, как если бы была вызвана функция ***include()***, при этом используется значение include_path.

Специальное значение none запрещает автодобавление.

Примечание.

Если сценарий обрывается функцией ***exit()****, вставка файла не произойдет.*

auto_prepend_file — строковое значение

Определяет имя файла, который автоматически добавляется к основному файлу и проверяется, вставка происходит перед основным файлом. Файл включается так же, как если бы была вызвана функция ***include()***, с использованием include_path.

Специальное значение none запрещает автодобавление.

display_errors (логическое значение)

Этот параметр определяет, будут ли сообщения об ошибках выводиться как часть HTML-кода.

doc_root (строковое значение)

Основной каталог («root directory») PHP на сервере. Используется только когда указанный каталог не пуст. Если PHP сконфигурирован при помощи safe mode, то никакие другие файлы за пределами этого каталога не будут обрабатываться с применением PHP.

error_log (строковое значение)

Имя файла, в котором хранится журнал log file. Если используется специальное значение syslog, то содержимое отправляется в системный журнал system logger.

error_reporting (целое значение)

Флаг уровня сообщений об ошибках. Значения флагов приведены в табл. 2.1. Параметр устанавливает уровень сообщений об ошибках. Чтобы установить уровни сообщений об ошибках, выберите те сооб-

Таблица 2.1.
Флаги сообщений об ошибках

Значение	Типы ошибок
1	Обычные ошибки
2	Обычные предупреждения
3	Ошибки синтаксического анализатора
4	Некритические предупреждения

щения, которые вам необходимы, сложите соответствующие им значения, результат укажите в поле error_reporting.

Значение по умолчанию — 7 (показываются обычные ошибки, обычные предупреждения и синтаксические ошибки).

open_basedir (строковое значение)

Задает директорий, в котором могут находиться PHP-файлы, доступные для обработки. Когда сценарий пытается открыть файл с помощью, например, fopen или gzopen, то осуществляется проверка наличия такого файла. Если файл находится за пределами указанного директория, то PHP откажется открыть его. Имена директориев в списке можно отделять друг от друга точками с запятой (в системе Windows).

Специальное значение . показывает, что каталог, в котором находится сценарий, используется в качестве основного каталога.

По умолчанию открыт доступ ко всем файлам.

gpc_order (строковое значение)

Устанавливает порядок обработки методов GET/POST/COOKIE. По умолчанию устанавливается «GPC». Если установить «GP», то PHP будет полностью игнорировать cookies и к тому же перезапишет все переменные метода GET переменными метода POST с теми же именами.

ignore_user_abort (строковое значение)

По умолчанию «On». Если указано «Off», то выполнение скриптов будет прекращено, если вывод информации скриптами был осуществлен после того, как клиент прекратил связь.

include_path (строковое значение)

Определяет список каталогов где будут располагаться файлы для функций ***require()***, ***include()*** и ***fopen_with_path()***. В этих каталогах будет осуществлен поиск запрашиваемых файлов. Например,

```
include_path=".;c:\www\phplib".
```

Значением по умолчанию является «.» (текущий каталог).

log_errors (логическое значение)

Сообщает, что независимые сообщения об ошибке сценария должны регистрироваться в журнале ошибок сервера.

magic_quotes_gpc (логическое значение)

Устанавливает magic_quotes состояние для GPC (Get/Post/Cookie) операций. Когда magic_quotes включено (on), все одиночные кавычки ', двойные кавычки «, обратные косые черты \ и нулевые значения (NUL's) записываются с обратной косой чертой автоматически. Если также включено magic_quotes_sybase, то одиночная кавычка записывается с добавлением дополнительной одиночной кавычкой вместо обратной косой черты.

magic_quotes_runtime (логическое значение)

Если magic_quotes_runtime разрешено, то большинство функций, которые возвращают данные внешнего источника, включая базы данных и текстовые файлы, будут иметь кавычки, записанные в сочетании с обратной косой чертой. Если magic_quotes_sybase включены, то одиночная кавычка записывается с добавлением дополнительной одиночной кавычкой вместо обратной косой черты.

magic_quotes_sybase (логическое значение)

Если включен magic_quotes_sybase, то одиночная кавычка записывается с добавлением дополнительной одиночной кавычкой вместо обратной косой черты, однако в том случае, если включены magic_quotes_gpc или magic_quotes_runtime.

max_execution_time (целое значение)

Эта установка определяет максимальное время в секундах, которое отводится для выполнения сценария, прежде чем выполнение будет прекращено анализатором. Это помогает осуществить защиту от некорректно написанных сценариев.

memory_limit (целое значение)

Эта установка определяет максимальный размер памяти в байтах, отводимый для сценария. Параметр помогает запретить некорректно написанным сценариям использовать всю доступную память сервера.

short_open_tag (логическое значение)

Задает допустимость использования короткой формы (<? ?> ярлыков PHP. Если вы хотите использовать PHP совместно с XML, то эту опцию необходимо отключить. Если опция отключена, следует использовать длинную форму тегов (<?php ?>).

track_errors (логическое значение)

Если опция включена, то последнее сообщение об ошибке всегда будет представлено в содержимом глобальной переменной $php_errormsg.

track_vars (логическое значение)

Если эта опция включена, то входящая информация GET, POST и cookie может быть найдена в глобальных массивах $HTTP_GET_VARS, $HTTP_POST_VARS и $HTTP_COOKIE_VARS, соответственно.

upload_tmp_dir (строковое значение)

Временный каталог, используемый для хранения файлов при их загрузке на сервер. Запись в этот каталог должна быть разрешена всем, независимо от того, какой пользователь обращается к PHP.

user_dir (строковое значение)

Основное имя каталога, используемого в домашнем каталоге пользователей для файлов PHP, например, public_html.

Конфигурирование почты

SMTP (строковое значение)

Имя DNS или IP-адрес почтового сервера SMTP, который должен использоваться PHP (в системе Windows) для отправки почтового сообщения функцией ***mail()***.

sendmail_from (строковое значение)

Определяет значение поля «From:» — почтовый адрес, который указывается в сообщении, отправляемом PHP (в Windows).

sendmail_path (строковое значение)

Задает местоположение программы не компьютере sendmail, как правило, это /usr/sbin/sendmail или /usr/lib/sendmail.

Конфигурирование безопасной настройки (Safe Mode)

safe_mode (логическое значение)

Устанавливает режим (включен, выключен) безопасной работы PHP (safe mode).

safe_mode_exec_dir (строковое значение)

Если PHP работает в безопасном режиме (в режиме safe mode), то ***system()*** и другие функции обращения к системным программам не будут работать с теми программам, которые находятся в каталоге, отличающемся от указанного в качестве значения параметра safe_mode_exec_dir.

Конфигурирование отладчика

debugger.host (строковое значение)

Имя DNS или IP-адрес хоста, используемого отладчиком.

debugger.port (строковое значение)

Номер порта, используемого отладчиком.

debugger.enabled (логическое значение)

Задает возможность (отладчик включен,выключен) использования отладчика.

Конфигурирование загрузки расширений (Extension Loading)

enable_dl (логическое значение)

Эта опция полезна только в модуле PHP для сервера Apache. Опция может разрешать/запрещать динамическую загрузку расширений PHP функцией ***dl()*** для каталогов и/или для виртуальных серверов.

Отключение динамической загрузки может быть использовано в целях безопасности. При включенной динамической загрузке можно обойти ограничения, установленные при помощи safe_mode и open_basedir.

По умолчанию динамическая загрузка разрешена за исключением случаев, когда включен безопасный режим safe-mode. При включенном безопасном режиме safe-mode использование ***dl()*** недопустимо.

extension_dir (строковое значение)

Задает каталог, в котором находятся динамически загружаемые расширения.

extension (строковое значение)

Строка, задающая имена динамически загружаемых расширенных при запуске PHP-расширений.

Конфигурирование MySQL

mysql.allow_persistent (логическое значение)

Разрешает/запрещает постоянные MySQL-соединения.

mysql.max_persistent (целое значение)

Максимальное число постоянных MySQL-соединений в одном процессе.

mysql.max_links (целое значение)

Общее максимальное число MySQL-соединений в одном процессе, включая постоянные соединения.

Конфигурирование mSQL

msql.allow_persistent (логическое значение)

Разрешить/запретить постоянные mSQL-соединения.

msql.max_persistent (целое значение)

Максимальное число постоянных соединений mSQL в одном процессе.

msql.max_links (целое значение)

Общее максимальное число mSQL-соединений в одном процессе, включая постоянные соединения.

Конфигурирование Postgres

pgsql.allow_persistent (логическое значение)

Разрешить/запретить постоянные соединения Postgres.

pgsql.max_persistent (целое значение)

Максимальное число постоянных соединений Postgres в одном процессе.

pgsql.max_links (целое значение)

Общее максимальное число Postgres-соединений в одном процессе, включая постоянные соединения.

Конфигурирование Sybase

sybase.allow_persistent (логическое значение)

Запретить/разрешить постоянные Sybase-соединения.

sybase.max_persistent (целое значение)

Максимальное число постоянных Sybase-соединений в одном процессе.

sybase.max_links (целое значение)

Общее максимальное число Sybase-соединений в одном процессе, включая постоянные соединения.

Конфигурирование Sybase-CT

sybct.allow_persistent (логическое значение)

Разрешить/запретить постоянные Sybase-CT-соединения. По умолчанию разрешено.

sybct.max_persistent (целое значение)

Максимальное число постоянных Sybase-CT-соединений в одном процессе. По умолчанию значение −1, означающее неограниченное число постоянных соединений.

sybct.max_links (целое значение)

Общее максимальное число Sybase-CT-соединений в одном процессе, включая постоянные соединения. По умолчанию значение −1, означающее неограниченное число соединений.

sybct.login_timeout (целое значение)

Максимальное время (в секундах), отведенное для ожидания повторной попытки соединения, после чего будет произведен отказ. По умолчанию — одна минута.

sybct.timeout (целое значение)

Максимальное время (в секундах), отведенное для ожидания окончания выполнения select_db или выполнения запроса. По умолчанию — не ограничено.

sybct.hostname (строковое значение)

Имя хоста, с которым осуеществляется соединение (для показа sp_who). По умолчанию — none (отсутствует).

Конфигурирование BC Math

bcmath.scale (целое значение)

Количество десятичных цифр, используемых во всех функциях bcmath.

Конфигурирование возможностей броузера

browscap (строковое значение)

Имя файла, содержащего установки броузера.

Конфигурирование ODBC

uodbc.default_db (строковое значение)

База данных ODBC, использующая по умолчанию в функциях *odbc_connect()* и *odbc_pconnect()*.

uodbc.default_user (строковое значение)

Имя пользователя, используемое по умолчанию в функциях, если в *odbc_connect()* и *odbc_pconnect()*.

uodbc.default_pw (строковое значение)

Пароль, используемый по умолчанию в функциях *odbc_connect()* и *odbc_pconnect()*.

uodbc.allow_persistent (логическое значение)

Разрешить/запретить постоянные ODBC-соединения.

uodbc.max_persistent (целое значение)

Максимальное число постоянных ODBC-соединений в одном процессе.

uodbc.max_links (целое значение)

Общее максимальное число ODBC-соединений в одном процессе, включая постоянные соединения.

Проверяем работоспособность сервера

Прежде чем написать нашу первую PHP-программу, способную работать в качестве средства обработки клиентских запросов, проверим работоспособность персонального web-сервера. Как мы уже говорили ранее, допускается использование и других типов серверов, но волей случая мы остановились именно на этом сервере. Программа-сервер используется для обработки клиентских запросов, т.е. запросов, поступающих от программы-клиента, как правило, программой-клиентом является брограмма-броузер, например, Netscape Navigator, Opera, Internet Explorer. Клиент отправляет запрос серверу, в запросе указывается имя файла, требуемого клиентом. Сервер находит запрашиваемый файл. В нашем случае персональный web-сервер после получения запроса и нахождения соответствующего файла распознает по расширению этого файла тип его содержимого. В качестве статических HTML-файлов будут использованы файлы с расширениями .html и .htm. Эти файлы будут переданы клиенту с использованием протокола http, причем файлы будут переданы в том виде, в каком они хранятся на диске компьютера без внесения каких-либо изменений перед отправкой.

При установке PHP на компьютер с программой-сервером мы вносили изменения в системный регистр. Мы указали типы файлов, которые будут использоваться в качестве источников, содержащих PHP-коды. С такими PHP-файлами мы связали расширения .php и .phtml. Можно задать и другие типы файлов в качестве файлов, содержание которых будет проверено и обработано интерпретатором PHP перед отправкой клиенту. В нашем случае все файлы, с расширениями .phtml и .php, если их запрашивает программа-клиент при обращении к программе-серверу, в отличие от HTML-файлов (расширения .htm и

.html), будут найдены сервером на диске. Затем, если запрошенный файл существует, то он будет проверен интерпретатором PHP, в случае отсутствия ошибок (все PHP-программы следует проверить и отладить перед тем, как они будут размещены на сервере) PHP-программа будет выполнена и отправлен html-ответ клиенту в соответствии с содержимым запрошенного клиентом PHP-файла. При выполнении PHP-программы сервер и интерпретатор PHP могут не только генерировать ответный HTML-код, используя ресурс сервера, но выполнять дополнительные действия. Что именно позволяет делать PHP нам станет известно в подробностях после того, как мы поближе познакомимся с возможностями этого языка. А сейчас мы просто напишем простейшую PHP-программу, проверим ее, исправим ошибки, разместим на сервере. После этого мы запросим файл с написанной программой из программы-броузера (она может располагаться на том же компьютере, что и сервер), и убедимся в том, что наша программа вполне работоспособна и полученный с ее помощью простейший HTML-код будет представлен в обычном для броузера виде.

Первый шаг — убеждаемся в том, что сервер запущен. В меню Пуск → Программы находим пункт Microsoft Personal Web Server находим Личный диспетчер Web и вызываем его. Перед нами открывается диалоговое окно (рис. 2.8).

Необходимо убедиться в том, что сервер запущен. Если сервер запущен, то в главном окне Личный диспетчер Web присутствует кнопка «Стоп». Если сервер находится в выключенном состоянии, то в окне будет показана кнопка «Пуск» (рис. 2.9). В этом случае необходимо нажать эту кнопку для того, чтобы каталоги сервера стали доступны для общего доступа.

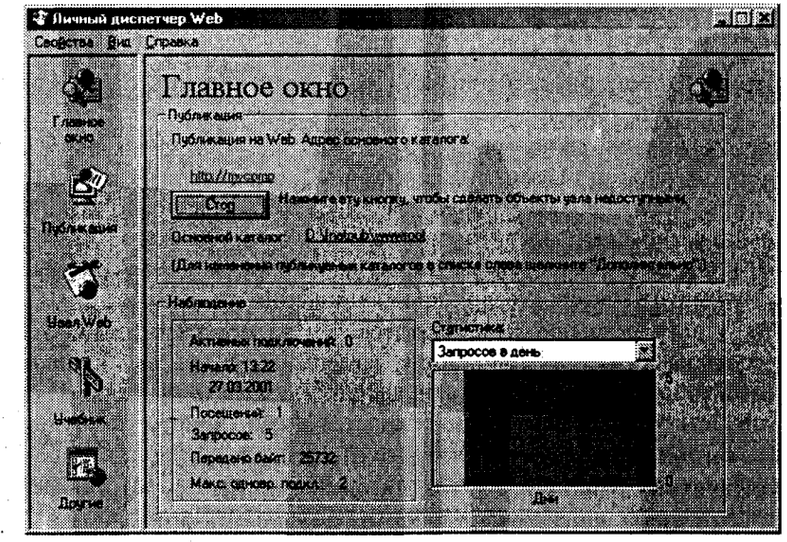


Рис. 2.8. *Управляем Web-сервером*

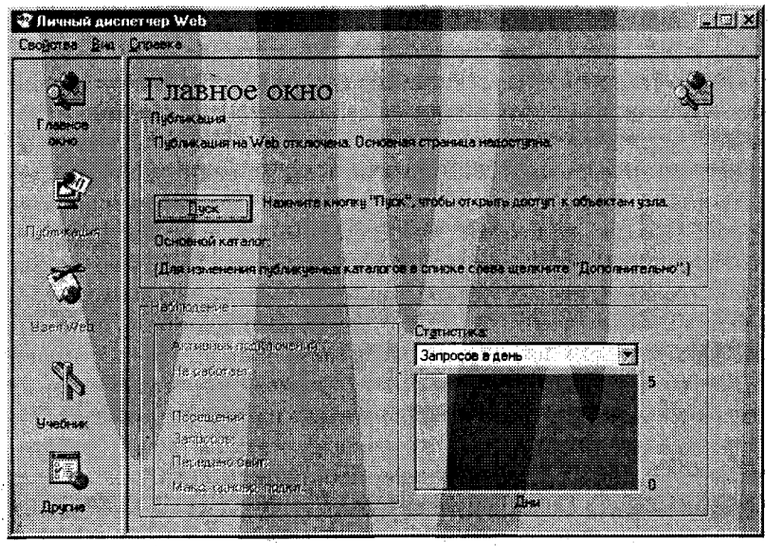


Рис. 2.9. *Сервер находится в остановленном состоянии, для его запуска необходимо нажать кнопку «Пуск»*

Следующим нашим шагом будет определение имен каталогов, в которых будут расположены файлы нашего сайта на том компьютере, где расположена программа-сервер. Выберем в меню, которое расположено в левой части окна Личный диспетчер Web, последнюю опцию, озаглавленную словом «Другие», щелкнем ее, и перед нами появится окно, в котором будут отображены дополнительные параметры (рис. 2.10).

В этом окне мы можем указать имена документов, которые будут открываться по умолчанию, а также разрешить просмотр каталогов сервера и установить режим сохранения журнала активности узла. Но больше всего нас интересуют сами каталоги, в которых мы будем размещать PHP-программы, а точнее, .php-файлы, в которых используются PHP-скрипты.

В представленном окне (рис. 2.10) перед нами показано дерево виртуальных каталогов сервера. Имена папок здесь являются псевдонимами реальных физических папок, расположенных на жестком диске компьютера. Виртуальные имена используются для того, чтобы скрыть от клиента реальное местоположение файлов на диске сервера. Это увеличивает безопасность, степень защищенности сервера.

Чтобы узнать реальное имя, скрытое за псевдонимом, выберем интересующую нас виртуальную папку, выделим ее и нажмем кнопку «Изменить свойства». Если мы выбрали папку cgi-bin, и нажали «Изменить свойства», то перед нами откроется окно, в котором будет содержаться информация о реальном имени каталога, соответствующего виртуальному имени каталога сервера cgi-bin, а именно, мы узнаем, что этому каталогу в действительности соответствует каталог

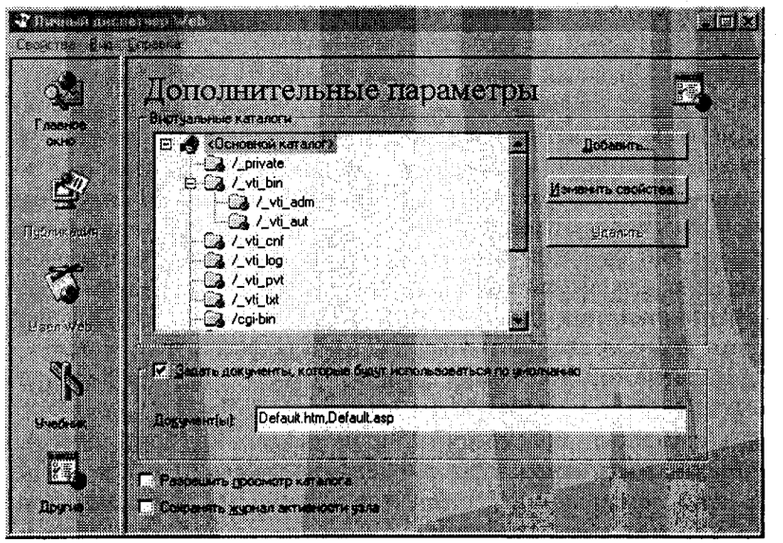


Рис. 2.10. *Окно редактирования дополнительных параметров сервера*

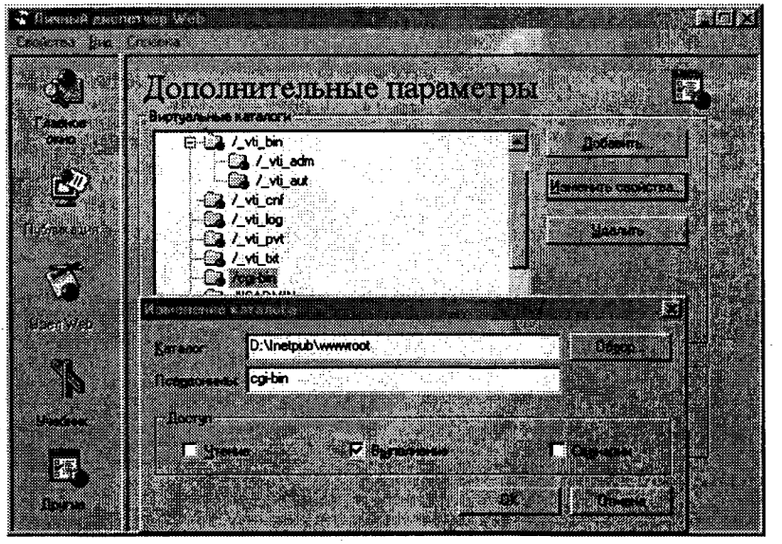


Рис 2.11. *Изменяем свойства виртуального каталога cgi-bin*

D:\Inetpub\wwwroot нашего компьютера (рис. 2.11). Здесь мы имеем возможность не только получить информацию о том, какой реальный каталог соответствует указанному виртуальному каталогу сервера, но и изменить значение имени реального каталога, соответствующего выбранному псевдониму. Мы также можем задать уровень доступа к каталогу, поставив галочки в полях чтение, выполнение, сценарии.

Из главного окна управления персональным сервером (рис. 2.8) мы знаем также, что адрес нашего сервера выглядит в локальной сети (в случае отдельной рабочей станции такой сетью можно считать «тривиальную» сеть, состоящую из этого отдельно стоящего, носимого или перевозимого в случае palmtop и/или notebook компьютера) как http://mycomp. Если мы соединяемся с Интернет посредством модема с помощью удаленного доступа к сети, то запущенный персональный web-сервер будет доступен для любого пользователя Интернет, поскольку в момент соединения с сетью Интернет провайдера автоматически выделяет нам индивидуальный IP-адрес. Каждый раз при соединении с Интернет при помощи модема мы получаем такой адрес, причем этот адрес может изменяться раз от раза. Некоторые провайдеры услуг Интернет предоставляют возможность закрепления фиксированного IP-адреса за пользователем.

Если выделенный при соединении с Интернетом наш IP-адрес будет известен другим пользователям сети Интернет, то они смогут посетить наш домашний персональный web-сервер. Как узнать то, какой именно IP-адрес был выделен для нас на время текущего сеанса? Для этого существует утилита winipcfg, вызвать которую можно, например, из DOS, для чего в командной строке DOS следует набрать

```
Winipcfg
```

Перед нами появится окно, в котором содержится вся интересующая нас информация (рис 2.12). Если соединение с Интернет не установлено, то эта информация не будет слишком полезной.

При наличии соединения с сетью Интернет мы сможем получить интересующую нас информацию (рис. 2.13).

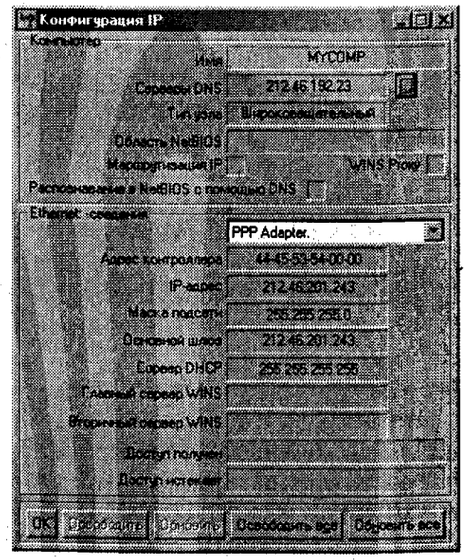


Рис. 2.12. *IP-конфигурация в отсутствии подключения к Интернет*

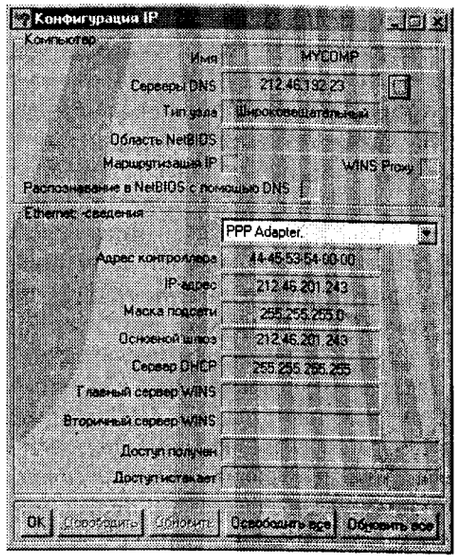


Рис. 2.13. *Сейчас при наличии подключения к Интернет, мы знаем свой IP-адрес*

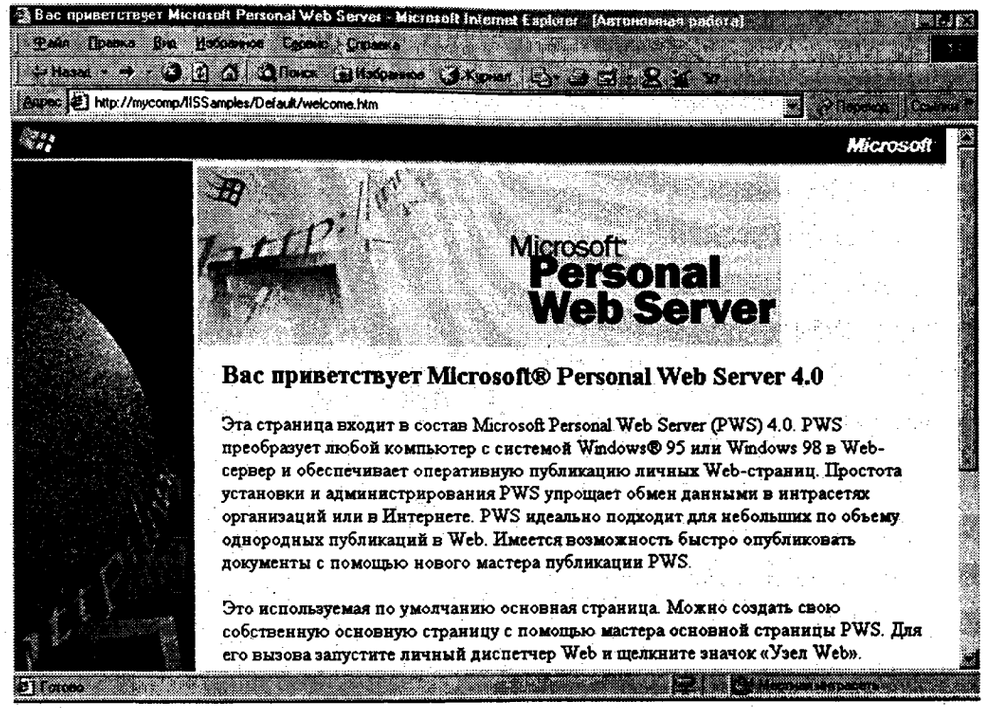


Рис. 2.14. *Приветственная страница персонального сервера. Ее появление говорит о том, что сервер настроен и работает правильно*

Сейчас доступ к нашему серверу открыт для любого пользователя Интернет. Для этого необходимо лишь указать адрес и протокол в броузере пользователя в виде http://212.46.201.243. При этом броузер обратится к нашему серверу и загрузит ту страницу, которая установлена для загрузки по умолчанию. В случае отсутствия соединения с Интернет мы имеем возможность проверить работоспособность установленного и настроенного сервера, если укажем в броузере адрес http://mycomp. Если при этом появится приветственная страница персонального сервера (рис. 2.14), то сервер настроен и работает правильно.

Если сервер не включен, неправильно настроен или отсутствует, то появится страница, содержащая предупреждение о том, что страница недоступна (рис. 2.15). При правильно настроенном и работающем сервере может появиться и другая страница, отличающаяся от той, что приведена на рис. 2.14. Приведенная на этом рисунке страница выводится по умолчанию при установке персонального сервера, но она может быть изменена, заменена, или сервер может быть сконфигурирован для показа иной странички. Все зависит от администратора сервера, т.е. от нас, ибо мы являемся администраторами своего персонального сервера. В случае только что установленного сервера приведенная выше страница направляется программе-клиенту вследствие работы скрипта default.asp.

Итак, мы убедились в том, что сервер функционирует нормально.

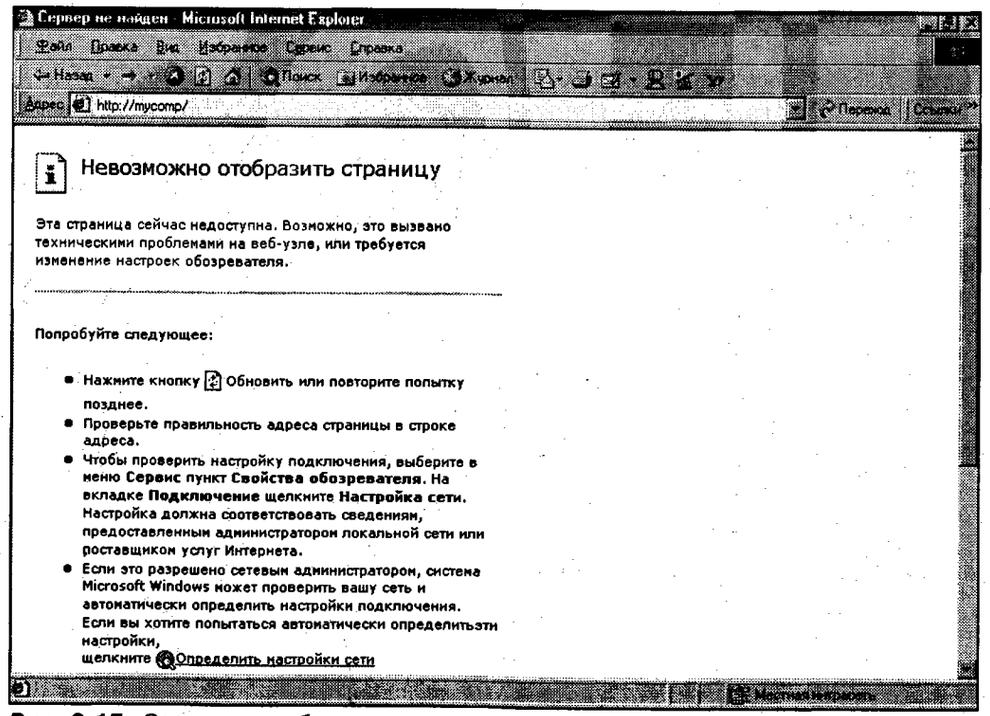


Рис. 2.15. *Сервер не работает*

Создаем простейший PHP-сценарий и размещаем его на сервере

Как мы уже знаем, PHP-сценарий помещается между знаками <?php и ?> и вставляется в обычную HTML-страницу. Существенным является то, что обычная HTML-страница может быть расположена где угодно, в то время как страница, содержащая PHP-сценарий должна располагаться в каталоге, доступном серверу (или, для краткости, на сервере) и иметь расширение, по которому система узнает, что данный файл долен быть обработан интерпретатором PHP.

Создаем файл first.phtm, текст файла таков:

```
<html>
<head>
<title>Первый простейший PHP-сценарий. </title>
</head>
<body>
<P align="center">
В этом файле содержится PHP-сценарий, выполняемый на сервере.
<p>
<?php
echo ("<P>Этот текст выведен при помощи <BR><font size=+1>
<b>PHP.</b></font><p>Удачная попытка!");
?>
<body>
</html>
```

Для создания этого файла мы используем простейший редактор текстов, такой, как например, блокнот в системе Windows, или edit.exe в DOS. Необходимо позаботиться о том, чтобы сохраняемый текст не содержал никакой дополнительной информации, в том числе информации о форматировании текста. Файл frirst.phtm следует запомнить в основном каталоге персонального сервера, в нашем случае, в каталоге d:\inetpub\wwwroot. Из программы-броузера следует запросить файл http://mycomp/first.phtm. В ответ мы получим страницу, показанную на рис. 2.16.

Полученный броузером от сервера HTML-код этой страницы выглядит следующим образом:

```
<html>
<head>
<title>Первый простейший PHP-сценарий. </title>
</head>
```

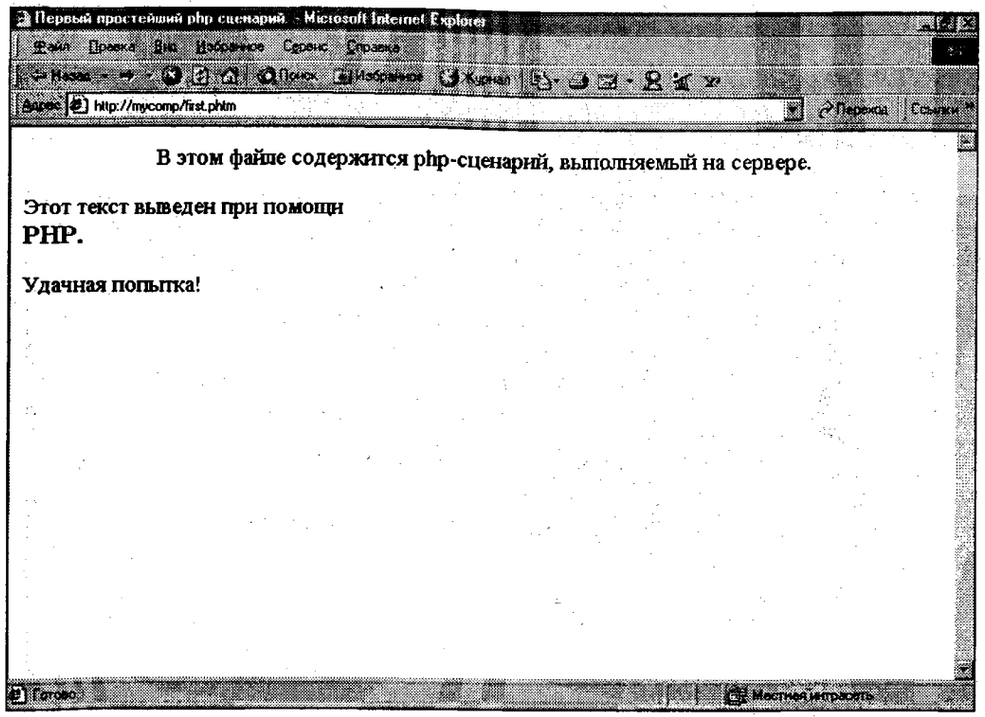


Рис. 2.16. *Страница создана с использованием первого простейшего PHP-сценария*

```
<body>
<P align="center">В этом файле содержится PHP-сценарий, выполняемый
на сервере.
<p>
<P>Этот текст выведен при помощи <BR><font size=+1><b>PHP.</b></
font><p>Удачная попытка!<body>
</html>
```

Если поместить файл first.phtm в другой каталог, который не обслуживается сервером, то броузер воспримет файл first.phtm как неизвестный ему тип файлов и предложит сохранить его (рис. 2.17).

В этой главе мы научились создавать простейшие PHP-сценарии, размещать страницы, содержащие такие сценарии, на сервере, и проверять их работоспособность. В последующих главах мы рассмотрим возможности, предоставляемые PHP, более подробно.

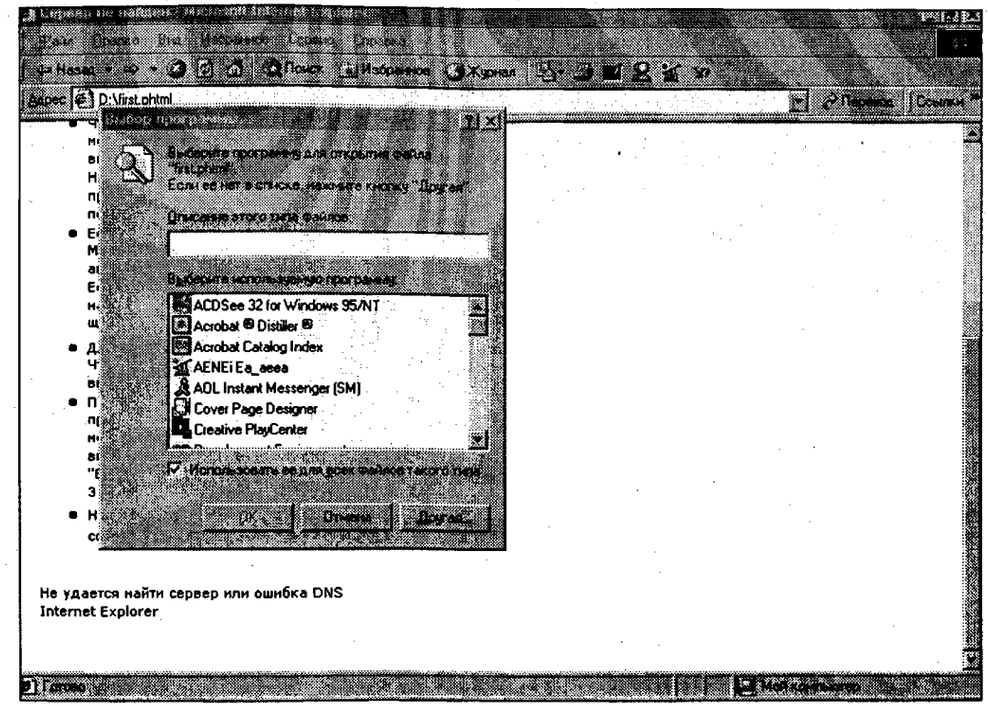


Рис. 2.17. *Броузер предлагает сохранить неизвестный тип файла*

3

ПЕРВЫЕ ПРОГРАММЫ НА PHP

- ***Отладка программ***
- ***Формы***
- ***Имена и значения***
- ***Передача пар имя/значение из формы программе PHP***
- ***Обработка значений***
- ***Программа PHP-почта***

Первые программы на PHP

Сейчас мы уже можем писать свои первые программы на языке PHP, не отвлекаясь на то, как и где они будут работать. Этим вопросам были посвящены предыдущие главы. Мы знаем, как в целом должна выглядеть программа на PHP, как она может быть вставлена в HTML-код исходного phtm-файла.

Начнем с простого примера, в котором язык PHP используется для того, чтобы в отсылаемый броузеру код вставить заранее подготовленный файл целиком. При этом в нашем phtm-файле появится лишь одна совсем маленькая инструкция include(«footer.txt»).

Этого достаточно. Файл footer.txt будет вставлен в том месте HTML-кода, где находится функция include(«footer.txt») (рис. 3.1).

Удобство состоит в том, что вызываемый файл будет содержать лишь незначительно больший объем информации, в то время как размер вставляемого файла может быть сколь угодно большим.

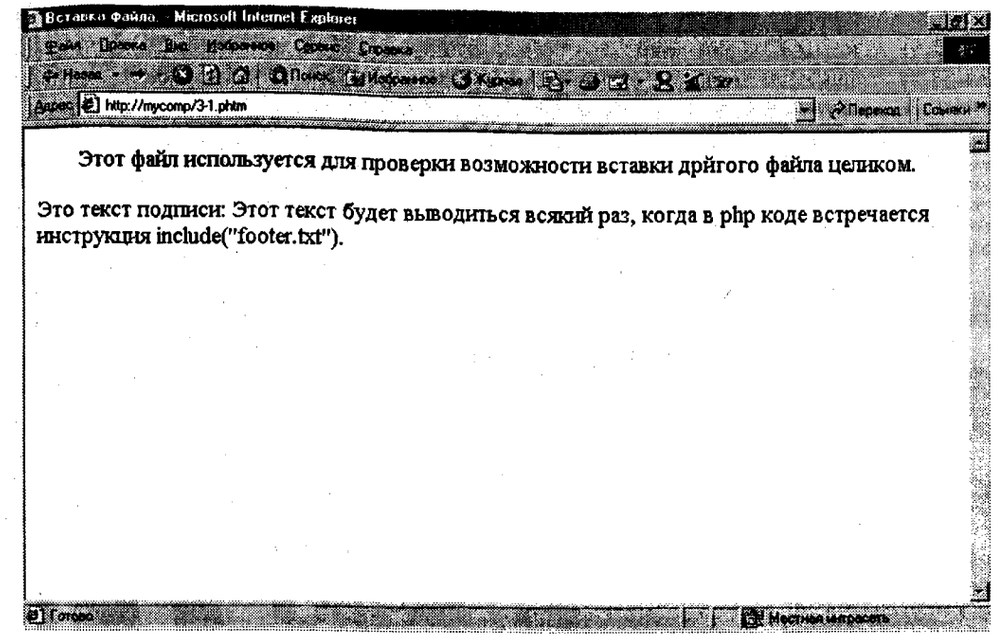


Рис. 3.1. *Вставка файла при помощи одной функции*

Вот текст файла со вставкой (3-1.phtm):

```
<html>
<head>
<title>Вставка файла. </title>
</head>
<body>
<P align="center">Этот файл используется для проверки возможности
вставки другого файла целиком.<p>
<?php include("footer.txt");
?>
<body>
</html>
```

При получении броузером этот файл будет изменен. То, что получит броузер, не будет содержать фрагмента, обрамленного символами <?php и ?>, вместо этого фрагмента будет помещено содержимое файла footer.txt.

Функция ***include()***, с помощью которой осуществляется вставка файлов в исходный HTML-код, может оказаться полезной в том случае, когда существует необходимость скрыть от пользователя структуру сайта. Иными словами, используя функцию ***include()***, можно скрыть от

пользователя то, в каком виртуальном каталоге сервера находится загружаемый файл, так как имя этого файла указано в виде аргумента функции ***include()***. Когда броузер запросит файл, в котором содержится ***include()***, то он в ответ получит только содержимое файла, указанного в качестве аргумента ***include()***, при этом пользователь, управляющий броузером, не будет знать не только имя файла, являющегося аргументом ***include()***, но он даже ничего не будет знать о том, как сформировано содержимое полученное его броузером, он не узнает, что фрагмент его HTML-кода был создан с использованием функции ***include()***.

Вывод запрашиваемого файла

Поскольку PHP является языком программирования, предоставляющим весьма широкие возможности, то можно по-разному организовать отправку клиенту различных файлов в зависимости от того или иного параметра. Так, вместо явно указанного имени вставляемого файла в функции ***include()*** можно указать переменную тип строки, которая будет содержать значение, равное имени файла. При этом значение это можно изменить в программе в зависимости от любых условий. Иными словами, запрашиваемая пользователем страница будет выглядеть по-разному в зависимости от этих условий, так как пользователю будут посылаться различные фрагменты HTML-кода.

Чтобы проиллюстрировать это на простом примере, создадим файл 3-2.phtm в следующем виде:

```
<html>
<head>
<title>Вставка файла в зависимости от условий.</title>
</head>
<body>
<P align="center"> Последущий фрагмент будет вставлен в зависимости от условий.<HR><p>
<?php
$a="3-1.txt";
if(1>0)$a="3-2.txt";
include($a);
?>
<body>
</html>
```

Здесь аргументом функции ***include()*** является переменная. Значение переменной может быть изменено. Для работы программы могут понадобиться два файла: 3-1.txt и 3-2.txt. Создадим их, пусть они будут весьма простыми. Файл 3-1.txt:


```
<font color="red" size=+2>
Этот фрагмент вставлен из файла 3-1.txt
</font>
Файл 3-2.txt:
<font color="green" size=+2>
```

Этот фрагмент вставлен из файла 3-2.txt

```
</font>
```

После запроса файла 3-2.phtm броузер получит следующий HTML-текст:

```
<html>
<head>
<title>Вставка файла в зависимости от условий.</title>
</head>
<body>
<P align="center"> Последующий фрагмент будет вставлен в зависимости от условий.<HR><p>
 <font color="green" size=+2>
```

Этот фрагмент вставлен из файла 3-2.txt

```
</font>
<body>
</html>
```

Как видим, здесь нет никакого намека на то, как был построен этот файл. Ссылка на функцию ***include()*** отсутствует вовсе. Вид окна броузера показан на рис. 3.2.

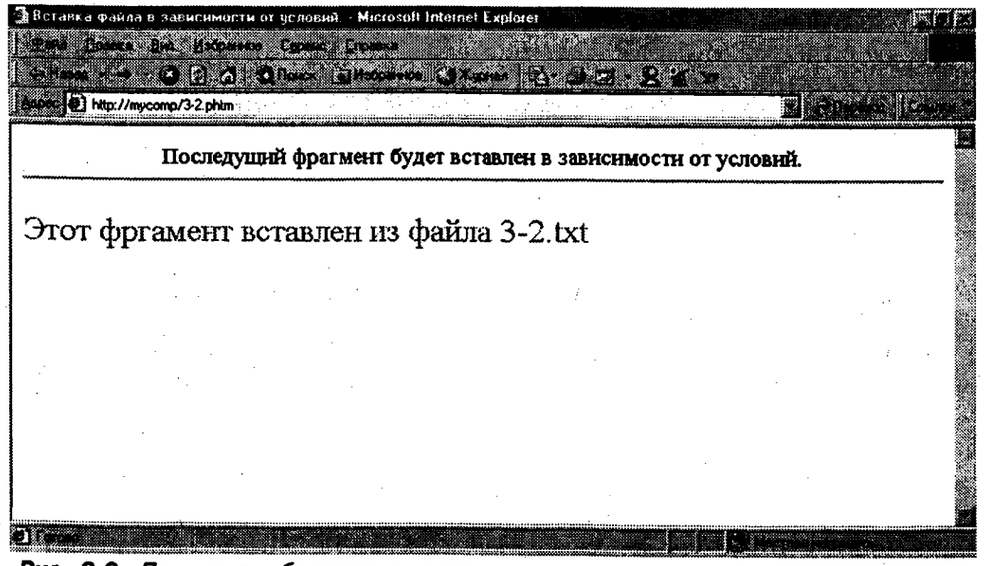


Рис. 3.2. *Полученное броузером содержимое web-страницы меняется в зависимости от условий*

Понятно, что если воспользоваться возможностями обработки полученной на сервере пользовательской информации, т.е. информации, полученной от программы-клиента, то можно организовать вывод файлов в зависимости от этой информации. Но как получить информацию от пользователя?

Существует очень удобный механизм передачи информации от броузера серверу. Специально для того, чтобы передача такой информации могла быть осуществлена с максимальным удобством, были изобретены формы.

Формы и передача информации клиентом серверу

Формы могут быть созданы исключительно средствами HTML. Для создания формы используется ярлык <form>. Внутри формы могут быть расположены элементы, причем количество элементов может быть велико. По своему названию и сути формы, как правило, содержат в себе элементы управления (текстовые поля, поля для галочек, радиокнопки, кнопки), которые предоставляют пользователю осуществлять тот или иной выбор. Конечно, не только стандартные средства сбора пользовательской информации могут быть использованы при работе с формами. Используя языки программирования, например, JavaScript, пользовательская информация может быть получена от пользователя и без ведома пользователя. Однако, сейчас мы не будем останавливаться на этом. Нас будет интересовать вопрос того, как информация от пользователя может быть передана серверу с использованием стандартных возможностей форм.

Рассмотрим пример, содержащий форму. Напишем простой HTML-файл и сохраним его на сервере с именем 3-3.htm:

```
<html>
<head>
<title>
```

Пример формы. Передаем пользовательскую информацию серверу.

```
</title>
<body>
<form name="first" method=post action="3-3.phtm">
<P>Введите текстовую информацию: <input type="text" name="text1">
<P><input type="submit" value="Кнопка отправки формы">
</form>
</body>
</html>
```


Здесь мы используем элемент формы, который описан при помощи ярлыка <form>. Мы присвоили форме имя, указав атрибут name и задав его значение (строка first). Важными атрибутами являются атрибуты method (значение post) и action. Значением атрибута action является имя файла, содержащего программу, которая будет обрабатывать полученную информацию. В нашем случае — это файл 3-3.php. Его созданием мы займемся минутой позже.

Наша форма состоит из двух стандартных элементов управления: текстового поля и кнопки для отправки формы. В HTML все стандартные элементы управления для работы с формами описываются при помощи ярлыка <input>, а все многообразие (порядка десяти) элементов управления обязано существованию различных стандартных типов элементов input. Чтобы создать нужный элемент управления, необходимо указать соответствующий тип элемента управления, присвоив необходимое значение атрибуту type. Существуют следующие стандартные типы элементов управления, используемые в формах:

- Button
- Checkbox
- file
- hidden
- Image
- password
- Radio
- reset
- submit
- Text

Перечислим применение типов элементов управления:

<INPUT type=button> создает элемент управления — кнопку.

Когда задан и включен элемент <INPUT type=checkbox>, то форма FORM посылает пару имя/значение (name/value). По умолчанию элемент INPUT type=checkbox включен.

<INPUT type=file> создает объект, отправляющий серверу файл, в документе показывается окно для ввода текста и кнопка Browse.

INPUT type=file должен быть использован в элементе FORM.

INPUT type=file необходимо указать значение NAME. Значение атрибута METHOD в элементе FORM должно быть указано как post. Значение атрибута ENCTYPE в элементе FORM должно быть указано как multipart/form-data. Для того, чтобы файл мог быть получен серверной стороной, необходимо, чтобы процесс на серверной стороне поддерживал посылаемые ему отправления а форме multipart/form-data. Например, Microsoft Posting Acceptor позволяет серверу Microsoft Internet Information Server (IIS) получать файлы указанным выше способом.

<INPUT type=hidden>. Этот тип не предоставляет пользователю никаких элементов управления, а посылает значение свойства value вместе с посылаемой формой.

<INPUT type=image> создает элемент управления — рисунок, при щелчке на котором происходит немедленная отправка формы серверу.

<INPUT type=password> создает поля для ввода однострочного текста, вводимый текст не показывается пользователю.

<INPUT type=radio> создает элемент управления — радиокнопку. Радиокнопки используются для ограничения пользовательского выбора, пользователь имеет возможность выбрать лишь один вариант из нескольких. Для этого несколько кнопок объединяются в группу, все кнопки в одной группе имеют одно и то же имя name. При посылке формы выбранная кнопка создает пару name/value.

Для отметки выбранной кнопки по умолчанию используется свойство checked, которому присваивается значение true.

<INPUT type=reset> создает кнопку, которая, будучи нажата, восстанавливает значения элементов управления формы и возвращает их к своим первоначальным значениям. Свойство value содержит метку, показываемую на кнопке Reset, точно также, как это происходит в случае с обычной кнопкой INPUT.

<INPUT type=submit> создает кнопку, которая при нажатии производит отправку формы серверу. Используется атрибут VALUE для создания кнопки с надписью-меткой, заданной в этом атрибуте. По умолчанию используется надпись Submit Query. Если пользователь нажимает кнопку Submit и посылает форму, а кнопка имеет имя name a, то с посылаемыми данными отправляется пара имя/значение (name/value).

<INPUT type=text> создает элемент управления для ввода однострочного текста. Атрибут SIZE задает количество видимых текстовых знаков в поле для ввода в элементе INPUT type=text. Атрибут MAXLENGTH задает максимальное количество знаков, которые могут быть введены в поле для ввода текста.

Помимо типа элемента управления, элемент input может иметь и другие атрибуты. Наиболее важными для нас являются два атрибута. Первый — это имя элемента, значение атрибута name, а второй — значение элемента управления. Значением элемента управления является значение атрибута value. В случае текстового поля значение элемента управления соответствует содержимому текстового поля, т.е. равно строчному значению, совпадающему с текстом, введенным в текстовый элемент управления. В соответствии с общепринятыми правилами, клиент после нажатия кнопки отправки формы (элемент управления <input type=submit>) пересылает указанной в атрибуте action текущего элемента form пары имя|значение, name|value. Существует два метода отправки этих значений: метод post и метод det. При использовании метода get пары name|value отправляются в виде пар, присоединенных к запрашиваемому URL-адресу, располагая их в конце строки после знака вопроса.

Одним из преимуществ языка PHP в сравнении с другими средствами обработки форм на серверной стороне, является то, что значения name/value могут быть получены очень просто. Для этого мы укажем в качестве метода отправки формы метод post, а в PHP-программе, расположенной в том файле, что указан в качестве значения атрибута action, будем использовать переменную с именем name. Идентификатор имени в программе PHP будет совпадать с именем, указанным в элементе управления формы. Значение этой переменной будет совпадать со значением формы. В нашем случае имя элемента формы было указано в виде name1, соответствующая переменная в PHP (файл 3-3.phtm) будет иметь вид $hame1. (Помните, что в PHP переменные начинаются со знака доллара $).

Сейчас мы уже вполне готовы к тому, чтобы написать текст файла 3-3.phtm. Вот он:

```
<html>
<head>
<title>
Пример обработки формы.
</title>
<body>
<P> Ваша форма обработана.
<P> От вас получен следующий текст.
<P><i>Текст, введенный в текстовом поле формы, таков:</i>
<?php
echo ("<p><font size=+2 color='red'>$text1");
?>
</form>
</body>
</html>
```

Чтобы полностью убедиться в том, что форма отсылается серверу и обрабатывается в соответствии с нашими инструкциями, загрузим страницу 3-3.htm в броузер и введем в текстовое поле какой-нибудь текст, например, «Зима и солнце — прелесть дня!», так, как это сделано на рис. 3.3.

Сейчас форма готова к отправке на сервер. Чтобы отправить ее, нужно всего лишь нажать кнопку «Кнопка отправки формы». Нажимаем ее, и через несколько секунд в коне броузера мы можем видеть ответ сервера (рис. 3.4). Как видим, форма была успешно обработана.

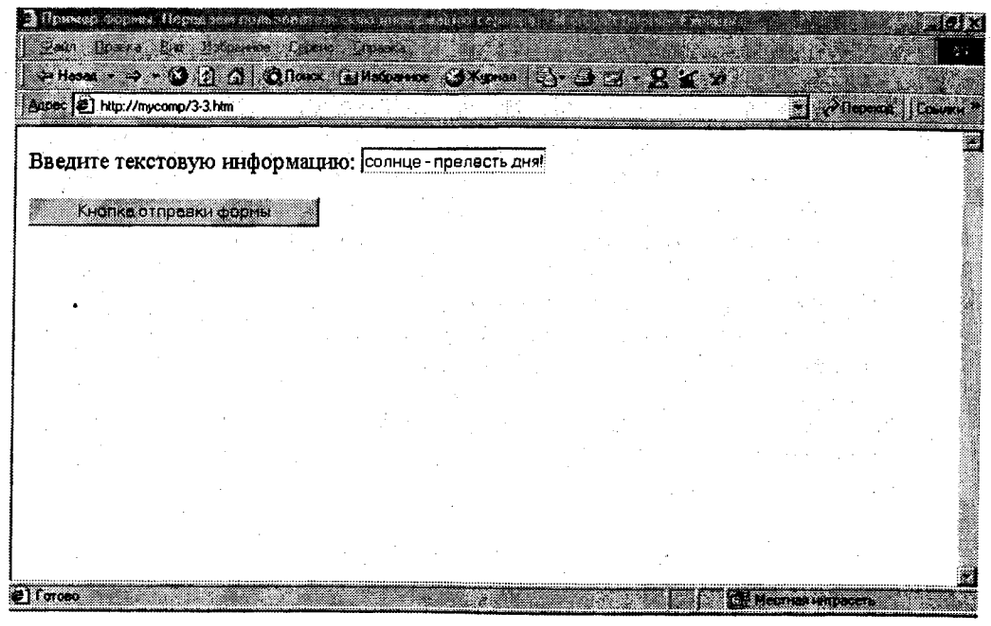


Рис. 3.3. *Форма заполнена и готова к отправке серверу*

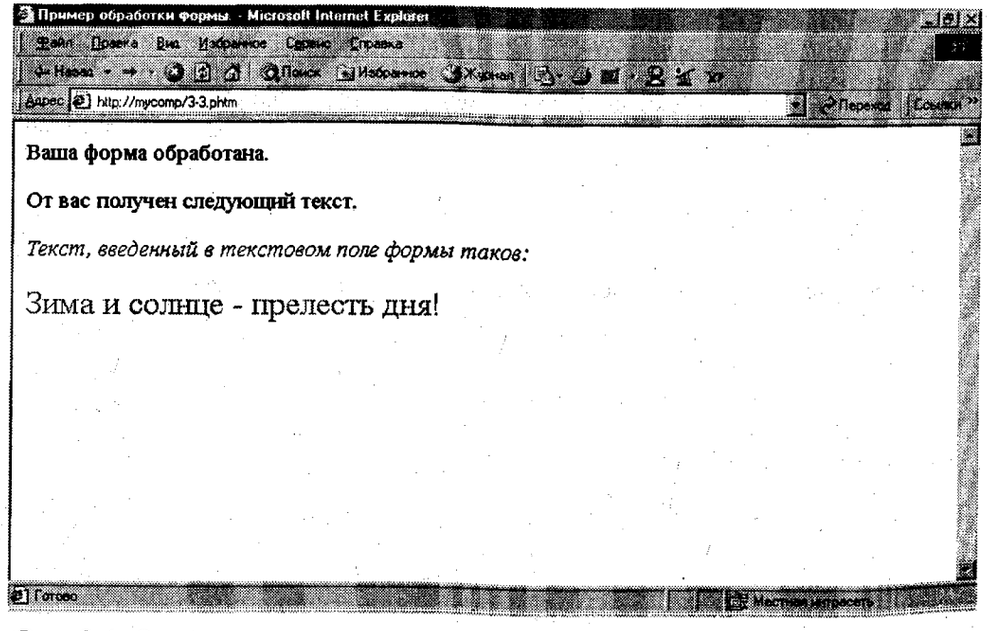


Рис. 3.4. *Ответ сервера получен*

Вывод файла в зависимости от информации о клиенте

Чтобы завершить наш рассказ о том, как можно вывести тот или иной файл в зависимости от того, что получено от пользователя, вернемся к нашему первому в этой главе примеру (файл 3-2.phtm) и изменим его следующим образом (теперь это будет файл 3-4.phtm):

```
<html>
<head>
<title>Вставка файла в зависимости от условий.</title>
</head>
<body>
<P align="center"> Последующий фрагмент будет вставлен в зависи-
мости от условий.<HR><p>
<?php
if($b=="1")$a="3-1.txt";
if($b=="2")$a="3-2.txt";
include($a);
echo ($b);
?>
<body>
</html>
```

В этом файле PHP-скрипт осуществляет вставку одного из двух файлов в зависимости от того, каким будет значение переменной $b. Для того, чтобы определить значение этой переменной, мы создадим форму с двумя радиокнопками и одним и тем же именем b в файле 3-4.htm, а обработку формы поручим файлу 3-4.phtm. Вот файл 3-4.htm:

```
<html>
<head>
<title>
```

Пример формы. Передаем пользовательскую информацию серверу.

```
</title>
<body>
<form name="first" method=post action="3-4.phtm">
<P>Сделайте ваш выбор:
<P>Выбор 1:<input type="radio" name="b" value="1">
<P>Выбор 2:<input type="radio" name="b" value="2">
<P><input type="submit" value="Кнопка отправки формы">
</form>
</body>
</html>
```

В окне броузера этот файл будет показан так же, как на рис 3.5.

Радиокнопки используются для того, чтобы пользователь мог осуществить альтернативный выбор. Лишь один вариант, заданный с использованием радиокнопок, может быть выбран пользователем. Группа радиокнопок состоит из нескольких элементов <input type=radio> с одним и тем же значением атрибута name. Значения атрибута value могут быть различными для разных кнопок. В нашем случае все кнопки имеют имя b, но принимают значение либо 1, либо 2.

Если выбрать первую радиокнопку и нажать кнопку отправки формы, то ответ сервера будет такой, как показано на рис. 3.6.

Если пользователь выберет вторую радиокнопку, то ответ сервера будет иным (рис. 3.7).

В заключение этого раздела заметим, что иногда в процессе отладки программ, мы обнаружили такую деталь. Формы и созданные для их обработки скрипты иногда не воспринимают имена элементов формы, если эти имена состоят из одной буквы, например так, как в приведенном выше примере, где использовано имя b и соответствующая ему переменная $b. Если изменить имя, указав его в виде, например, bbb, и использовать соответствующую ему переменную $bbb, то ошибок не возникает. При создании любых программ требуется их отладка. Если возникают те или иные ошибки, то попробуйте что-нибудь изменить в программе, сравнить неработающую деталь программы с аналогичным фрагментом из другой работающей программы. Это поможет исправить возникающие ошибки.

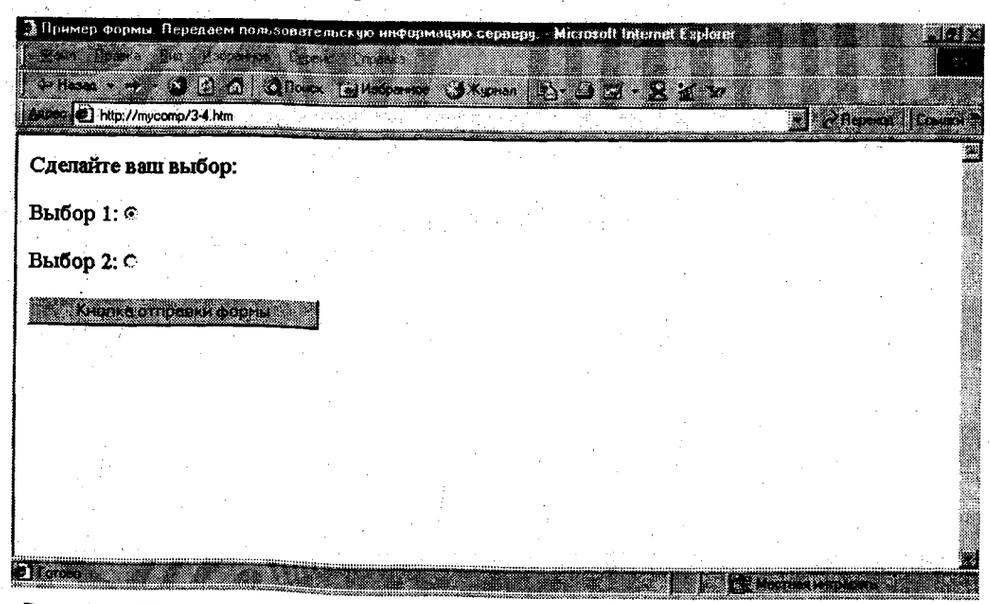


Рис. 3.5. *Пользователь может выбрать одну из радиокнопок*

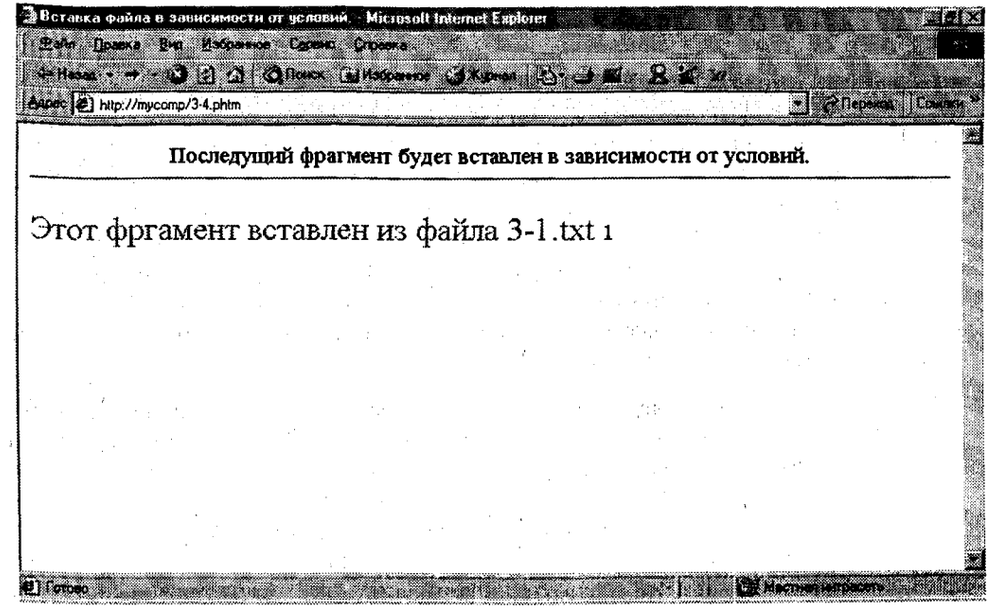


Рис. 3.6. *Пользователь выбрал первую радиокнопку*

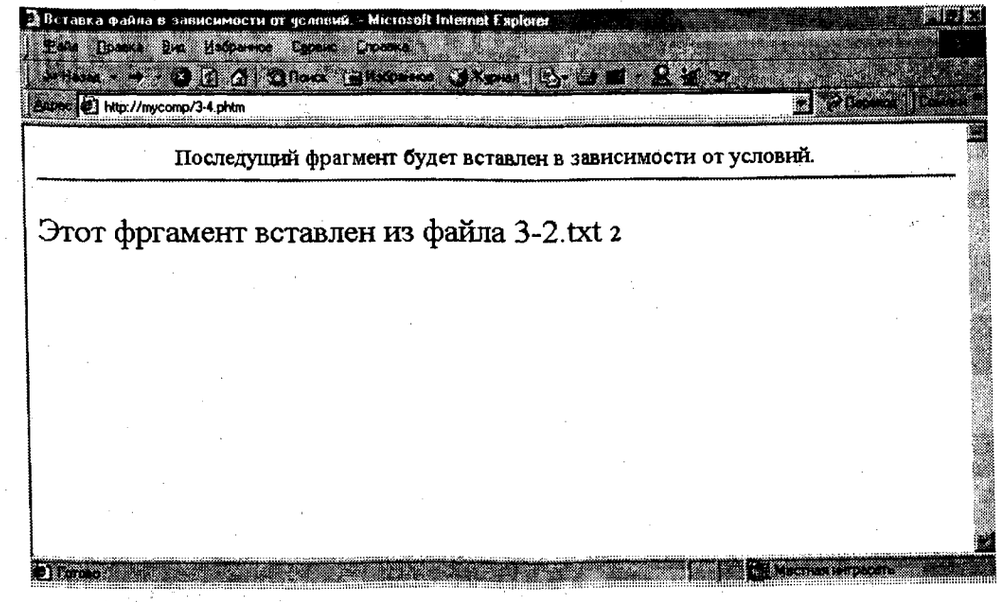


Рис. 3.7. *Пользователь выбрал вторую радиокнопку*

Отправка почты средствами сервера

Можно создать программу, которая будет отправлять почту указанному адресату. Адресат может быть как одним и тем же для всех почтовых отправлений, так и значением строки, содержащей адрес получателя, которое может быть изменено как сервером, так и отправителем формы.

Создадим статический HTML-файл, содержащий форму. В этом файле мы создадим несколько элементов управления для ввода текстовой информации. Первый элемент — текстовое поле, в которое пользователю будет предложено вписать свое имя. Второе поле отведено для ввода адреса электронной почты. Предполагается, что значение этого элемента формы будет обработано на сервере программой, которая отошлет по указанному адресу определенный текст. Содержание текста будет меняться в зависимости от того, какая информация была введена пользователем в форму перед ее отправкой серверу. Наконец, последнее текстовое поле предоставляет пользователю произвести выбор его любимого времени года. Эта информация будет использована сервером при формировании почтового отправления по тому адресу, которое пользователь ввел в поле для ввода электронного адреса. Информация в поле для ввода любимого времени года заносится из списка выбора. Список выбора мы формируем при помощи списка, который создается при помощи ярлыка <select>, которому мы присваиваем имя s. После того, как форма будет отправлена на сервер, ее обработает PHP-скрипт, расположенный в файле mail.php, так как именно этот файл указан в качестве значения в атрибуте action ярлыка <form>.

```
<HTML>
<HEAD>
<TITLE>Заполняем форму</TITLE>
<BODY>

<B>Здесь вы можете указать свой адрес и получить дополнительную
информацию, послав нам форму.</B>
<P>

<TABLE WIDTH = 400><TR><TD align="left">

<FORM ACTION="mail.php" METHOD="POST">
Ваше имя:<BR>
```



```
<INPUT TYPE="text" NAME="name" SIZE="20" MAXLENGTH="30"><P>
Ваш email:<BR>
<INPUT TYPE="text" NAME="email" SIZE="20" MAXLENGTH="30">
<P>Ваше любимое время года:
<SELECT NAME="s">
<OPTION value = zima>Зима
<OPTION value = vesna>Весна
<OPTION value = leto>Лето
<OPTION value = osen>Осень
<OPTION value = zemphira>Земфира
</SELECT><P>
<INPUT TYPE="submit" VALUE="отослать форму">
</FORM>
</TD></TR></TABLE></CENTER>

</BODY>
</HTML>
```

Загруженный в броузер файл 3-5.htm выглядит так, как показано на рис. 3.8.

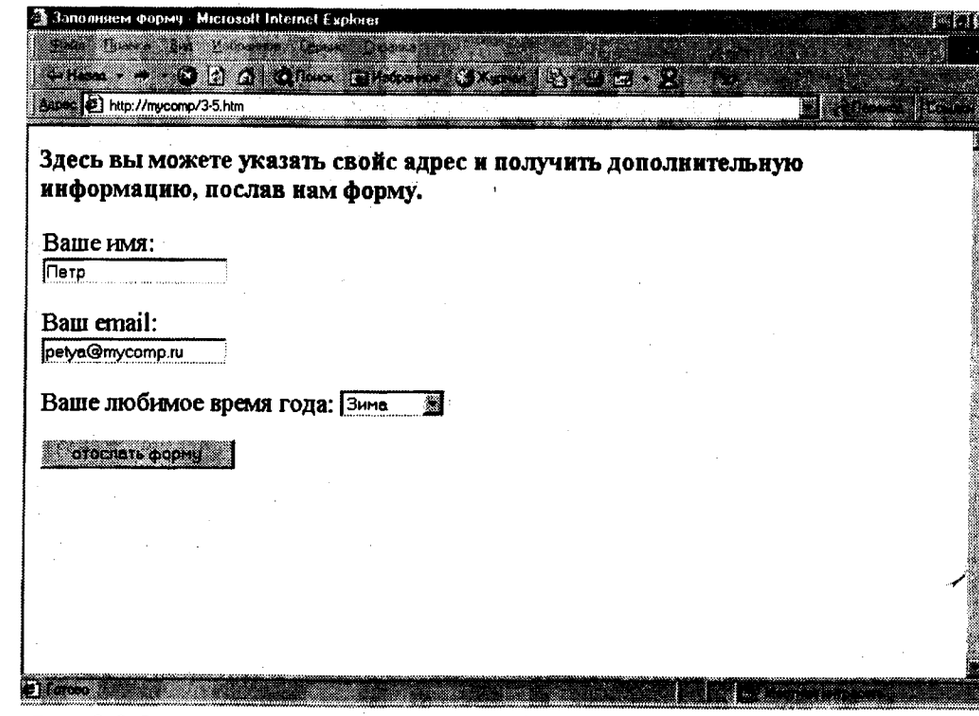


***Рис. 3.8** Заполняем форму, которая будет обработана для отправки почты*

Итак, мы создали три элемента формы, каждый элемент имеет свое индивидуальное имя, которому в программе PHP будет соответствовать своя переменная, эта переменная отличается от имени элемента управления одним символом. В PHP-программе переменные отличаются от имен элементов переданной ей форме наличием символа доллара ($). Вот эти три элемента (табл. 3.1).

Таблица 3.1
Элементы управления формы в файле 3-5.htm и значения переменных, переданные PHP-программе в файле mail.php

Форма	Имя формы	Соответствующая этому имени переменная php	Значение переменной
<INPUT TYPE="text" NAME="name" >	Name	$name	Значение, введенное пользователем в текстовое поле.
<INPUT TYPE="text" NAME="email" >	Email	$email	Значение, введенное пользователем в текстовое поле.
<SELECT NAME="s">	S	$s	В зависимости от выбора пользователя: Зима Весна Лето Осень Земфира

Перечисленные в табл. 3.1 переменные ($name, $email, $s) будут доступны PHP-программе в файле mail.php. Создадим этот файл в следующем виде:

```
<?php
/* Скрипт для обработки переменных, посланных в форме, описанной в
файле mail.htm  */
echo ("<B>");
echo ("Привет,<h1><center> $name!</center></h1>");
echo ("<HR></b>");
echo ("<center>Спасибо за участие!<BR><BR>");
echo ("Мы посылаем информацию по указанному Вами адресу: $email
<BR><BR>
Ваше любимое время года $s.");
echo ("<hr>");

mail($email, "Ваш запрос", "$name
Спасибо за отклик!n
Мы делаем много полезного и забавного!
Приходите к нам развлечься и заработать, мы тоже любим
$s !");
```



```
mail("your@address.ru",
"Запрос информации от пользователя.",
"$name requested for information.n
Адрес пользователя $email. n
Любимое время года пользователя $s.");
?>
```

Здесь используется функция ***mail()***, первый аргумент функции — адрес, по которому отправляется почтовое сообщение, второй аргумент — тема сообщения, третий аргумент — текст сообщения. Программа посылает два сообщения. Первое сообщение отправляется по адресу, указанному пользователем в форме, а второе сообщение посылается по указанному в PHP-программе адресу, в этом сообщении по заданному адресу пересылается сообщение о том, что пользователь произвел отправку формы, а также в сообщении указывается выбранное пользователем время года.

После обработки формы сервер возвращает броузеру информацию, которая будет представлена примерно в таком виде, как показано на рис. 3.9.

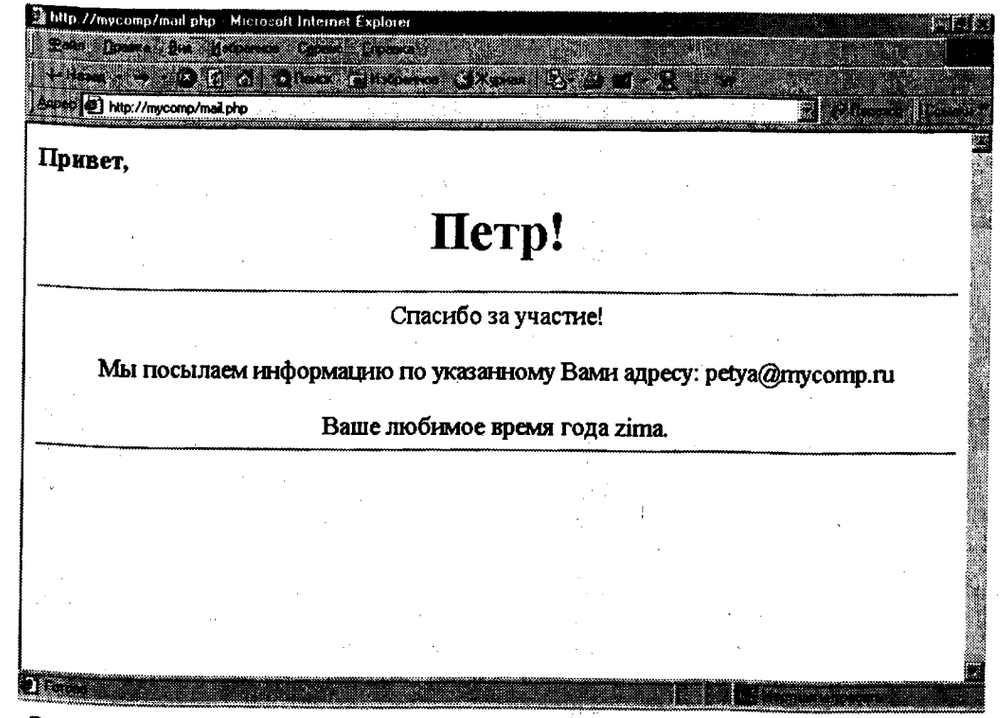


Рис. 3.9. *Форма обработана*

После обработки формы «Пете» будет послано сообщение следующего содержания:

Петр requested for information.
Адрес пользователя petya@mycomp.ru.
Любимое время года пользователя zima.

Для того, чтобы почта работала правильно, необходимо правильно сконфигурировать PHP. Для этого в файле php.ini необходимо указать имя почтового сервера, т.е. указать директиву SMTP и задать имя отправителя в директиве sendmail_from. Соответствующий фрагмент файла php.ini может выглядеть примерно так:

```
[mail function]
SMTP= your.mailserver.ru ; for Win32 only
sendmail_from= me@localhost.com ; for Win32 only
;sendmail_path=;for unix only, may supply arguments as well (default is
'sendmail -t -i')
```

4

РАБОТА С ФАЙЛАМИ

- ***Обработка содержимого файлов***
- ***Пересылка файлов по почте и отправка файлов серверу***
- ***Счетчики посещений и работа с сессиями***

Рассмотрим пример PHP-программы, которая анализирует полученный файл, находит на нем все ссылки, описанные при помощи ярлыка < a href=...> и посылает броузеру список найденных ссылок.

Вначале мы создадим простой HTML-файл, в котором будет предусмотрена форма, содержащая текстовое окно для ввода файла или http адреса страницы, ссылки которой нас интересуют. Файл этот мы назовем cutref.htm.

Файл cutref.htm

```
<html>
<head>
<title>
URL-страницы для получения всех ссылок.
</title>
</head>
<body>
<H1>Форма для ввода адреса страницы.</H1>
<p align=center>Эта форма используется для того,
чтобы указать адрес страницы
(файл или в формате http://host.name.xx/name.file),
```

```
все ссылки которой будут
выведены на экран броузера.<p>
<p>Сама страница, адрес которой был указан в поле для ввода текста,
выведена не будет.
<form action="cutref.phtm" method="post">
Введите имя файла или http адрес: <input type=text name="filename">
<p><input type=submit value="Укажите адрес файла в текстовом поле и
нажмите кнопку">
</form>
</body>
</html>
```

Сохраним этот файл на сервере. После загрузки в броузер мы увидим страничку, показанную на рис. 4.1.

Следующий шаг — создание программы, которая бы смогла проанализировать полученный файл. Эта программа будет получать имя файла через переменную, посылаемую ей приведенной выше формой. Для этого в форме в текстовом поле мы укажем имя файла, затем нажмем кнопку.

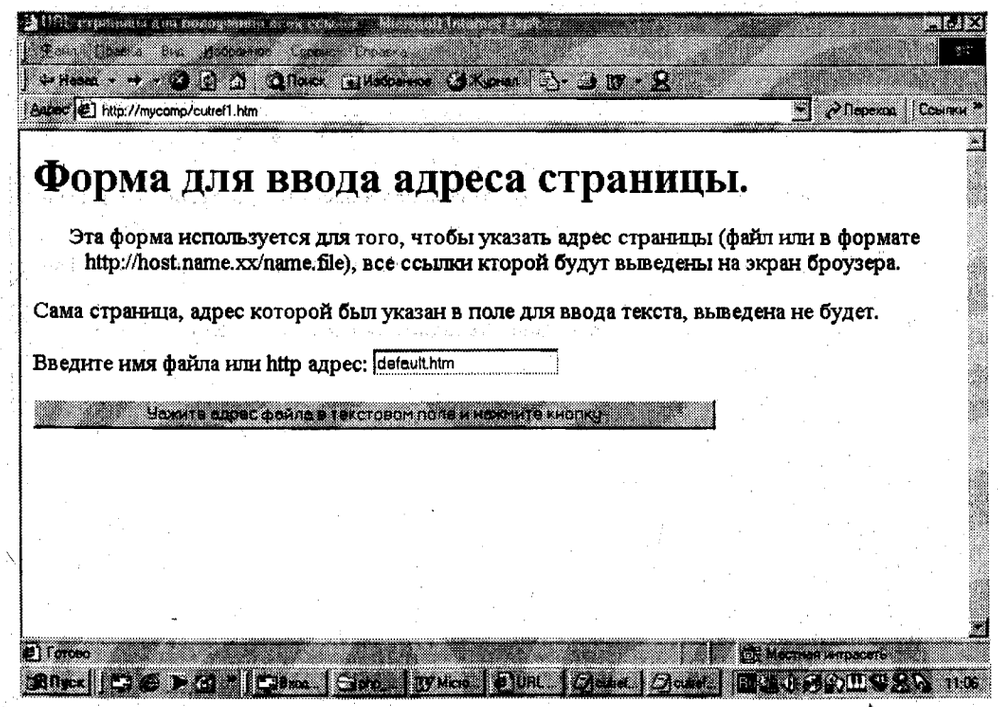


Рис. 4.1. *Форма, используемая для задания имени анализируемого файла*

В тексте файла cutref.htm мы указали адрес файла с PHP-программой (значение атрибута action в ярлыке form) в виде cutref.phtm. Под таким именем мы и сохраним файл с анализирующим ссылки PHP-кодом.

Файл cutref.phtm

```
<?
 echo "<pre>";
 // качаем страницу в переменную $buf
 $buf=implode("",file($filename));
 // получаем ссылки в массив
   preg_match_all("/<[Aa][  \r\n\t]{1}[^>]*[Hh][Rr][Ee][Ff][^=]*=[
'\"\n\r\t]*([^ \"'>\r\n\t#]+)[^>]*>/",$buf,$url);
 // выводим массив на экран
 while($i<count($url[1])) { echo $url[1][$i++]."\n"; }
 echo "</pre>";
?>
```

Сейчас, когда обрабатывающая программа готова и сохранена в файле с требуемым именем, можно отправить форму, для чего нажимаем кнопку формы. Через пару секунд в окне броузера появляется список ссылок (см. рис. 4.2).

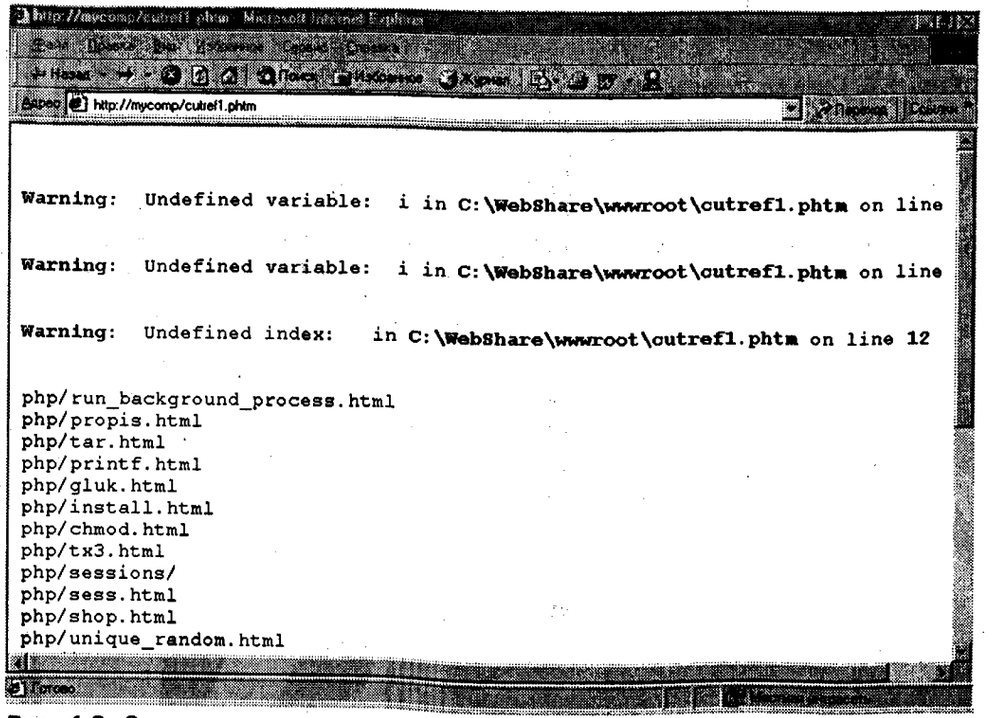


Рис. 4.2. *Список ссылок*

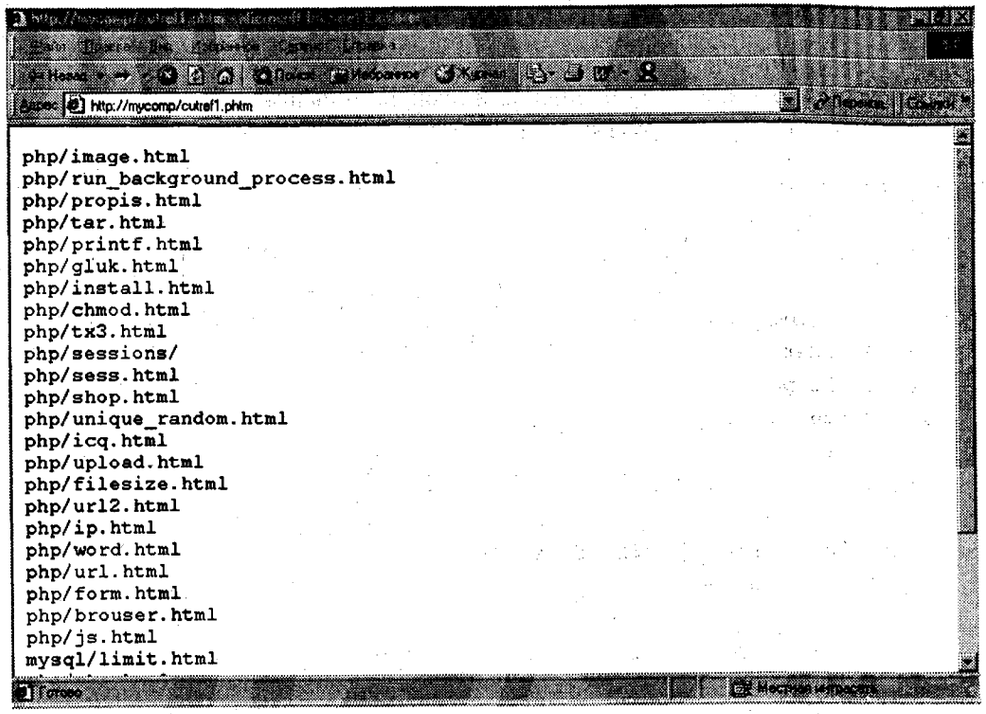


Рис. 4.3. *Ошибки исправлены*

Однако в тексте программы мы забыли явно инициализировать переменную $i, поэтому получили сразу три предупреждения о наличии нефатальных ошибок. Чтобы избежать вывода предупреждений на экран броузера, можно изменить режим вывода сообщений об ошибках. Мы же пойдем другим путем и исправим код программы. Вставим инструкцию $i=0 после комментария //выводим массивы на экран. Сейчас ошибок нет, и предупреждения не появятся (рис. 4.3).

Отправка файла в качестве почтового приложения

Пример отправки почты без использования почтового клиента с приложением HTML-письма и двоичного файла.

Текст PHP-кода помещен в файл mailatt.phtm.

Файл mailatt.phtm

```
<?
// Функции.

class html_mime_mail {
  var $headers;
  var $multipart;
  var $mime;
  var $html;
  var $parts = array();

function html_mime_mail($headers="") {
     $this->headers=$headers;
}

function add_html($html="") {
     $this->html.=$html;
}

function build_html($orig_boundary,$kod) {
     $this->multipart.="—$orig_boundary\n";
   if ($kod=='w' || $kod=='win' || $kod=='windows-1251') $kod='windows-
1251';
     else $kod='koi8-r';
     $this->multipart.="Content-Type: text/html; charset=$kod\n";
     $this->multipart.="BCC: del@ipo.spb.ru\n";
     $this->multipart.="Content-Transfer-Encoding: Quot-Printed\n\n";
     $this->multipart.="$this->html\n\n";
}

function add_attachment($path="", $name="", $c_type="application/octet-
stream") {
     if (!file_exists($path.$name)) {
       print "File $path.$name dosn't exist.";
       return;
     }
     $fp=fopen($path.$name,"r");
```

```
    if (!$fp) {
      print "File $path.$name coudn't be read.";
      return;
    }
    $file=fread($fp, filesize($path.$name));
    fclose($fp);
       $this->parts[]=array("body"=>$file, "name"=>$name,"c_type"=>$c_type);
}

function build_part($i) {
    $message_part="";
    $message_part.="Content-Type: ".$this->parts[$i]["c_type"];
    if ($this->parts[$i]["name"]!="")
        $message_part.="; name = \"".$this->parts[$i]["name"]."\"\n";
    else
        $message_part.="\n";
    $message_part.="Content-Transfer-Encoding: base64\n";
    $message_part.="Content-Disposition: attachment; filename = \"".
        $this->parts[$i]["name"]."\"\n\n";
   $message_part.=chunk_split(base64_encode($this->parts[$i]["body"])."\n";
    return $message_part;
}

function build_message($kod) {
    $boundary="=_".md5(uniqid(time()));
    $this->headers.="MIME-Version: 1.0\n";
   $this->headers.="Content-Type: multipart/mixed; boundary=\"$boundary\"\n";
    $this->multipart="";
    $this->multipart.="This is a MIME encoded message.\n\n";
    $this->build_html($boundary,$kod);
    for ($i=(count($this->parts)-1); $i>=0; $i--)
      $this->multipart.="--$boundary\n".$this->build_part($i);
    $this->mime = "$this->multipart--$boundary--\n";
}

function send($server, $to, $from, $subject="", $headers="") {
     $headers="To: $to\nFrom: $from\nSubject: $subject\nX-Mailer: The
Mouse!\n$headers";
    $fp = fsockopen($server, 25, &$errno, &$errstr, 30);
    if (!$fp)
        die("Server $server. Connection failed: $errno, $errstr");
```



```
    fputs($fp,"HELO $server\n");
    fputs($fp,"MAIL FROM: $from\n");
    fputs($fp,"RCPT TO: $to\n");
    fputs($fp,"DATA\n");
    fputs($fp,$this->headers);
    if (strlen($headers))
      fputs($fp,"$headers\n");
    fputs($fp,$this->mime);
    fputs($fp,"\n.\nQUIT\n2);
    while(!feof($fp))
      $resp.=fgets($fp,1024);
    fclose($fp);
  }
}
// ***************************************************************************
//
//    В качестве приложения присоединяем html-письмо
//    (открывается автоматически).
//    Второе приложение - локальный файл.
//
// ***************************************************************************

  $mail=new html_mime_mail();
  $mail->add_html("<html><body><center><h2>Привет!<br><br>".
                  "<br>Посылаю двоичный файл [/bin/ls] ...".
                  "</h2></center></body></html>");
  $mail->add_attachment("c:\vc.com");
  $mail->build_message('win'); // если не "win", то кодировка koi8
  $mail->send('mail.server.name',
              'address@to.name',
              'address@from.name',
              'посылаю файл');
//здесь mail.server.name - имя почтового сервера
//address@to.name  - адресат, которому посылаем письмо
//address@from.name - адресата, от которого письмо отправлено
// посылаю файл - тема письма
?>
```

Загружаем файл mailatt.phtm в броузер. Броузер запрашивает соединение с Интернетом, осуществляет поиск указанного почтового сервера. Если сервер найден, происходит отправка файлов. Через несколько минут адресат получит отосланное письмо (рис. 4.4).

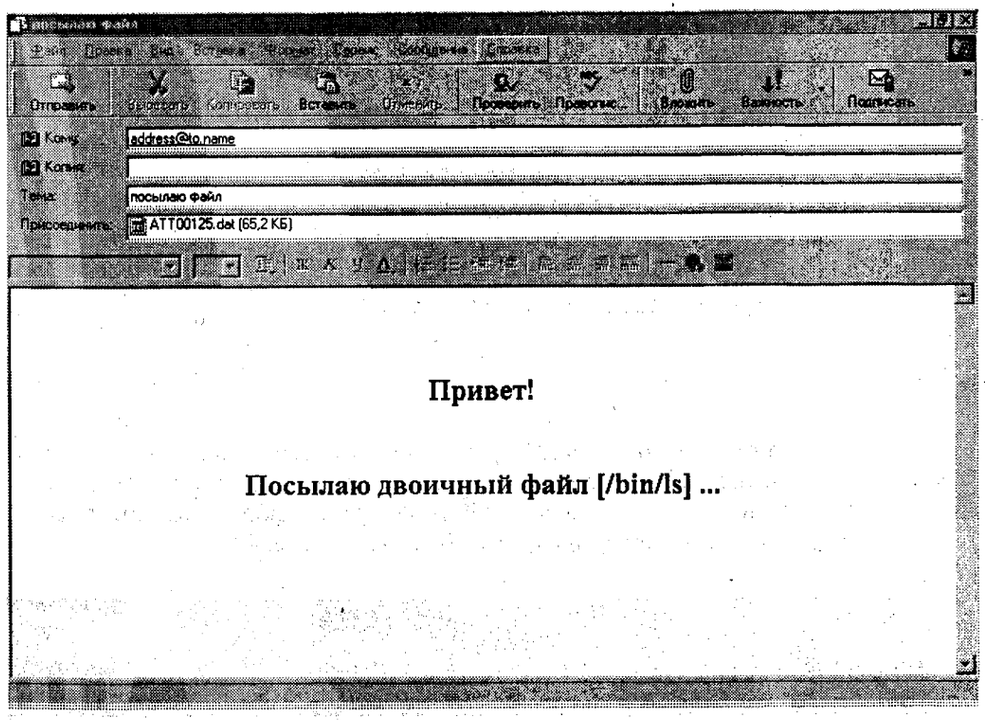


Рис. 4.4. *Письмо с приложением дошло до адресата*

Отправка файла серверу

Для отправки файла серверу удобно использовать несложный PHP-скрипт. Вначале мы создадим HTML-страницу, в которой будут содержаться насколько форм, с помощью которых пользователь будет иметь возможность произвести выбор отправляемых серверу файлов. Отправляемые файлы должны находиться на локальных дисках пользователя. Для выбора файлов используется элемент управления формы <input type=«file»>. Этот элемент позволяет воспользоваться стандартными окнами системы windows, которые позволяют производить передвижение по локальной файловой системе и выбирать файлы.

Создадим файл upload.htm

Файл uplod.htm

```
<html>
<head>
<title>PHP's FileUPLOAD</title>
</head>
<body>
      <form method="post" action="upload.phtm" enctype="multipart/
form-data">
     <input name="userfile[]" type="file">
     <input name="userfile[]" type="file">
     <input name="userfile[]" type="file">
     <input name="userfile[]" type="file">
     <input type="submit" value="Upload!!!" >
     </form>
</body>
</html>
```

С помощью этого HTML-файла мы выбираем файл на локальном диске и направляем его для обработки скрипту, расположенному программе в файле upload.phtm. Имя элемента формы задаем в виде userfile[]. Выбираем файлы, как показано на рис. 4.5.

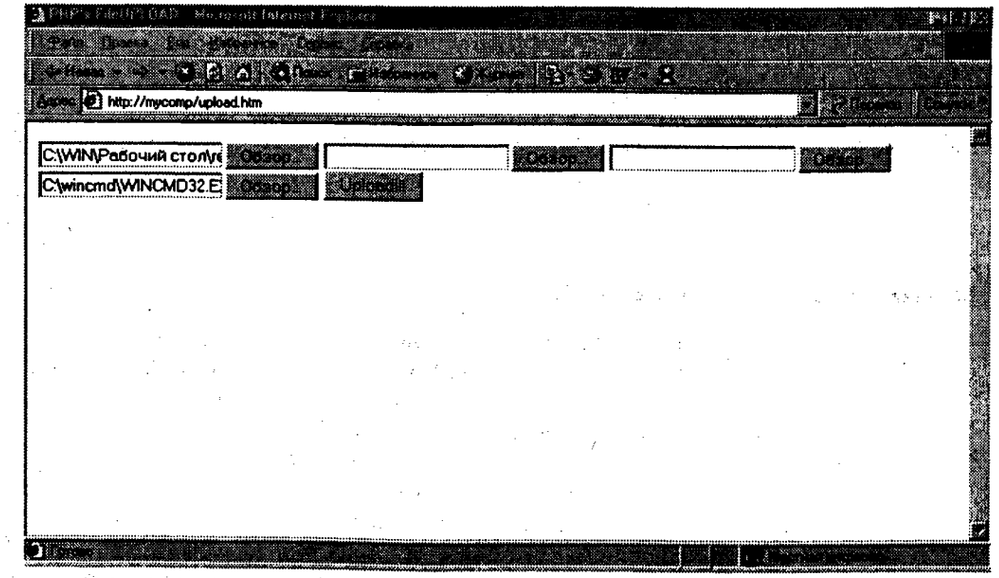


Рис. 4.5. *Выбираем файл для отправки серверу*

Теперь создаем файл uipload.phtm.

Файл upload.phtm.

```
<?
    for($i=0;$i<sizeof($userfile);$i++)
    {
    if(!$userfile_size[$i])
    continue;

    $UPLOAD = fopen( $userfile[$i], "r" );
    $contents = fread( $UPLOAD,$userfile_size[$i]);
    fclose( $UPLOAD );
    $SAVEFILE = fopen( "upload/".$userfile_name[$i], "wb" );
    fwrite( $SAVEFILE, $contents,$userfile_size[$i] );
    fclose( $SAVEFILE );
    }
    echo "Сервер получил файлы!";
    ?>
```

Результат, выводимый после завершения работы скрипта, показан на рис. 4.6.

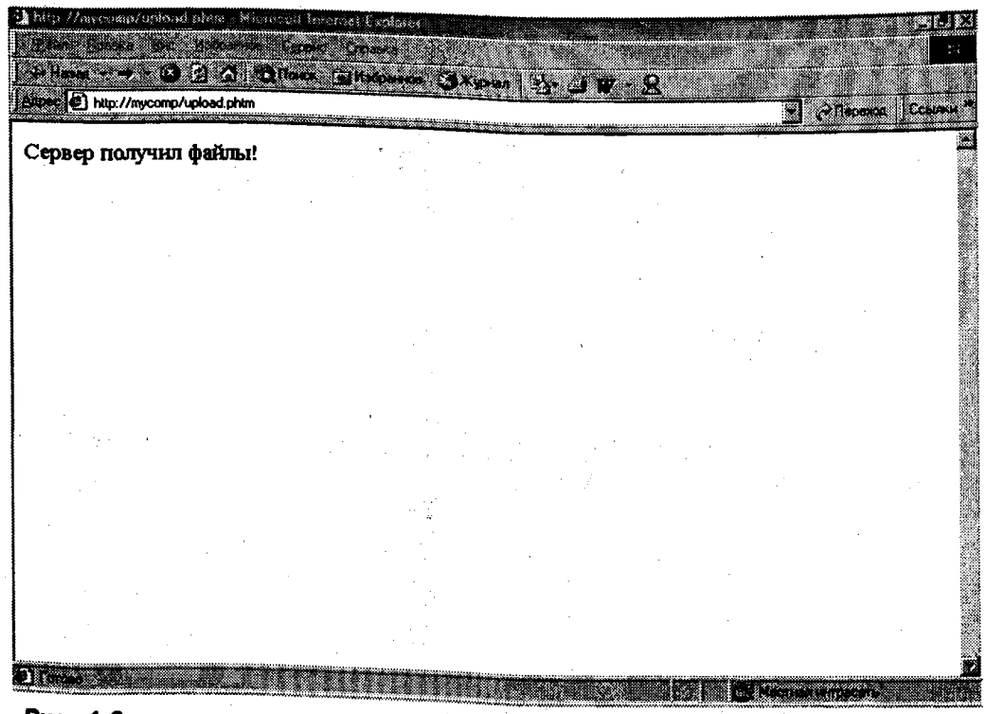


Рис. 4.6.

Рассмотрим этот код подробнее. В самом начале мы организуем цикл, с помощью которого проверяем все элементы массива userfile (каждый элемент содержит имя посылаемого серверу файла). Затем открываем очередной пользовательский файл для чтения с помощью ***fopen()***, затем читаем его с помощью ***fread()*** и закрываем ресурс с применением ***fclose()***. Затем мы открываем файл (ресурс) для записи при помощи ***fopen()***, при этом необходимо убедиться, что в текущей директории есть папка upload, если такой папки нет, то файл не может быть открыт. Затем мы осуществляем запись в этот файл с помощью ***fwrite()*** и затем закрываем файл с помощью ***fclose()***.

После завершения работы скрипта в папке upload сервера мы найдем новые отправленные серверу файлы (см. рис. 4.7).

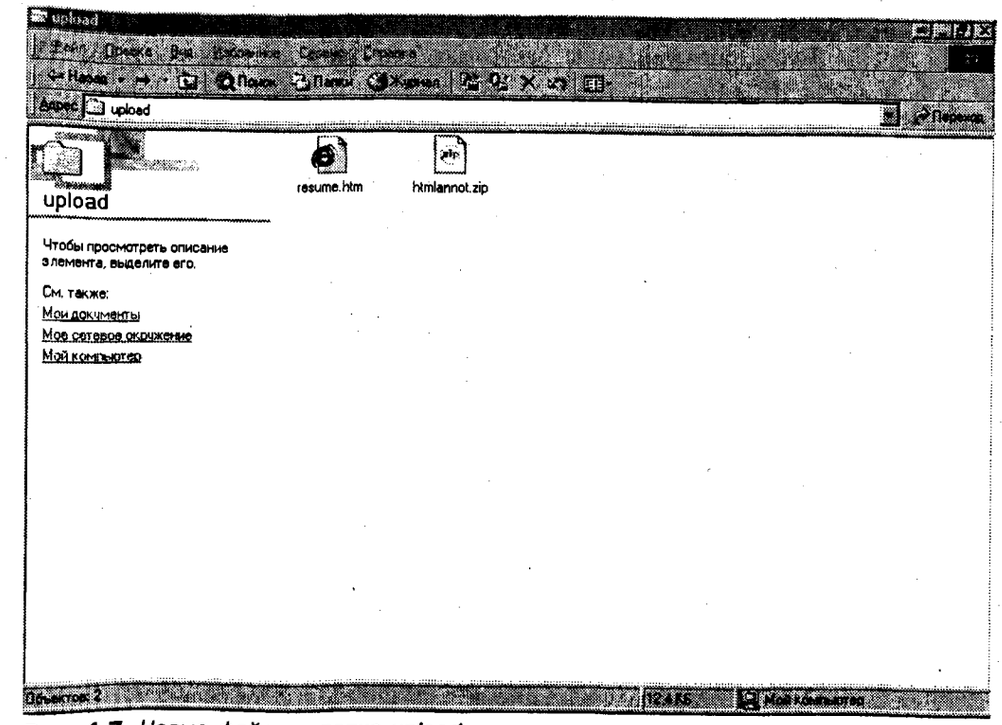


Рис. 4.7. *Новые файлы в папке upload*

Простой счетчик посещений

С помощью языка PHP очень просто написать программу, которая будет осуществлять счет посещений той или иной странички. Причем используя PHP мы имеем в своем распоряжении богатый арсенал средств, позволяющий нам проделывать кроме пересчета посещений любые довольно сложные действия. Остановимся для начала на простом скрипте, пересчитывающим обращения к странице.

Скрипт, который мы сохраним в файле schetchik1.phtm, будет использовать переменную, пусть она называется $content, которую мы будем увеличивать на единицу всякий раз, как произойдет обращение к скрипту. После увеличения значения этой переменной, которая собственно и представляет собой счетчик, мы ее сохраним в файле с именем, например, C:\counter.

Таким образом, наша программа должна будет проделать следующие действия.

1. Открыть файл C:\counter для чтения.
2. Прочитать содержимое файла C:\counter в переменную $content.
3. Закрыть указатель на файл C:\counter.
4. Увеличить на единицу значение переменной $counter.
5. Открыть для записи файл C:\counter.
6. Записать содержимое переменной $counter в файл c:\counter.
7. Закрыть указатель на файл C:\counter.
8. Вывести текущее значение переменой $counter.

Простейший вариант скрипта, выполняющего такие действия, может выглядеть так.

Файл schetchik1.phtm

```
<?
$get = fopen("c:\counter", "r");
$content = fread ($get, 500000);
fclose ($get);
$content++;
$fput = fopen("c:\counter", "w");
fwrite ($fput, $content, 500000);
fclose ($fput);
echo "Страницу посетили <B> $content </B> раз."
?>
```

Сейчас загрузим страницу schetchik1.phtm в броузер и несколько раз нажмем кнопку обновления страницы в главной панели инструментов броузера. При каждом нажатии на эту кнопку счетчик будет показывать увеличение количества посещений на единицу (рис. 4.8).

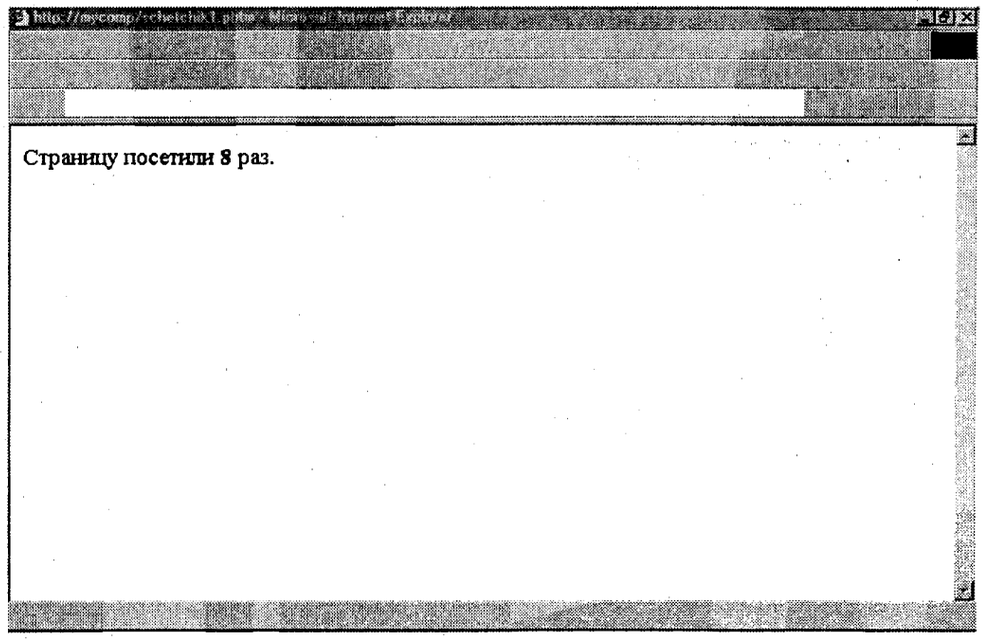


Рис. 4.8. *Простейший счетчик посещений*

Сейчас встает вопрос о том, как можно воспользоваться таким счетчиком на своей странице. Это сделать очень просто. Один из возможных вариантов выглядит так (файл schetchik_1.phtm).

```
<html>
<head>
<title>
Страница, количество посещений которой нужно просчитать.
</title>
<body>
<H1>При посещении этой страницы активизируется счетчик.</h1>
<P>Мы используем общий счетчик. Как только происходит обращение
к счетчику, его значение увеличивается на единицу вне зависимсоти
от того, откуда было произведено обращение к этому счетчику.
</p>
<?
include ("schetchik1.phtm");
?>
</body>
</html>
```

На экране компьютера мы будем видеть такую картинку (рис. 4.9).

Существует множество вариантов организации счетчиков посещений. Даже в том случае, если счетчик посещений служит для выполнения одной основной задачи — пересчету количества обращений к странице. Существует множество возможностей организации программного кода.

Рассмотрим еще один вариант. В этом случае счетчик посещений реализован с помощью средств работы с сессиями. Дело в том, что в PHP поддерживается возможность работы с сессиями. Переменные можно объявить глобальными. Глобальные переменные хранятся на диске в одном файле, которому соответствует уникальный идентификатор пользователя. При вхождении на web-сервер пользователь получает такой идентификатор автоматически. В языке PHP идентификатор может быть установлен в явном виде при помощи функции ***session_id()***. Если в счетчике задан один для всех общий идентификатор сессии, устанавливаемый всякий раз при обращении к счетчику, то уникальный пользовательский идентификатор для текущей сессии будет переписан в соответствии с указанным в ***session_id()*** в качестве аргумента. Именно такой возможностью мы собираемся воспользоваться.

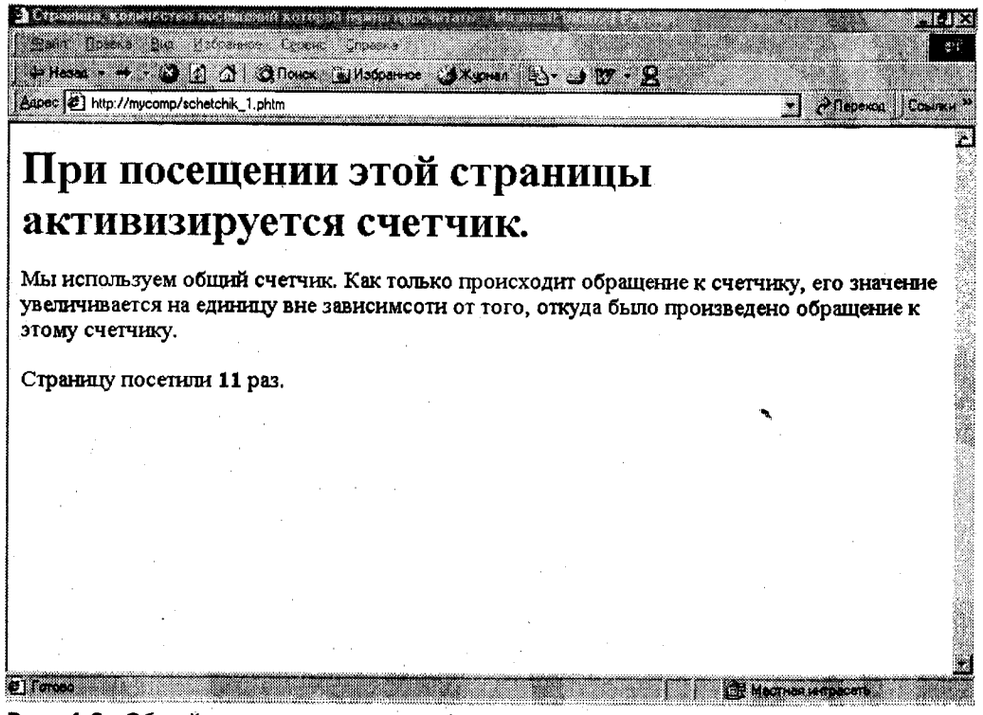


Рис. 4.9. *Общий счетчик встроен в web-страничку*

Чтобы переменные стали глобальными, и их значение было записано на диск, достаточно обратиться к функции ***session_register()***, в качестве аргументов указав имена этих переменных. Итак, после всего сказанного, можно записать скрипт счетчика посещений, использующего работы с сессиями, в нашем случае мы устанавливаем идентификатор сессии в виде «global». Код скрипта восхитительно краток (файл counter.phtm).

Файл counter.phtm

```
<?php
session_id ("global");
session_register ("count");
$count++;
?>
Добрый день! Вы  <? echo $count; ?> -ый посетитель этой странички.<p>
```

На экране мы увидим обычное сообщение (рис. 4.10).

Мы привели пример работы с сессиями. Заметим, что если бы мы не указали в явном виде идентификатор сессии, то счетчик стал бы пересчитывать посещения только этого конкретного пользователя!

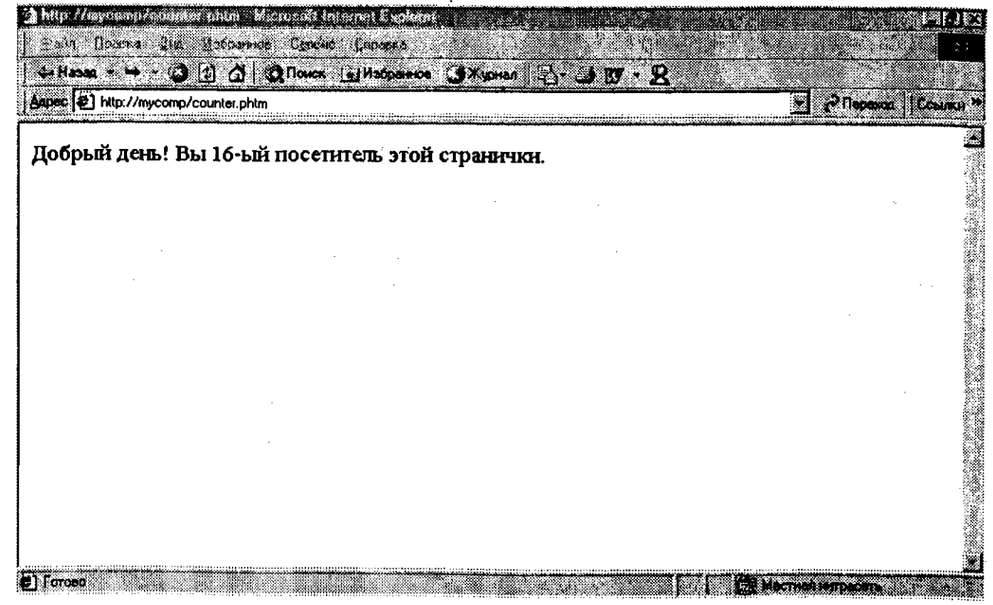


Рис. 4.10. *Пример скрипта с обработкой сессий*

5

РИСУНКИ. ИНФОРМАЦИЯ О PHP

- ***Вывод рисунков***
- ***Создаем рисунок***
- ***Скрытый вывод рисунков***
- ***Внешние рисунки***
- ***Информация, полученная с помощью phpinfo()***
- ***Текст в рисунках***

Вывод рисунков

PHP обладает рядом функций, позволяющих создавать рисунки, манипулировать ими, осуществлять ввод и вывод рисунков. Для того, чтобы функции для работы с рисунками можно было использовать, необходимо убедиться в том, что библиотека для работы с рисунками у вас имеется (она входит в стандартный дистрибутив в виде динамической библиотеки php_gd.dll), а также в том, что файл php.ini инициирован должным образом, т.е. что в нем содержится строка extension=php_gd.dll.

В разделе, посвященном рисункам, мы рассмотрим несколько базовых вопросов, ограничиваясь такими задачами: как создать рисунок средствами PHP, как вывести текстовую информацию в рисунке, как вывести рисунок броузеру и сохранить его в файле. Подробную информацию о возможностях работы с рисунками вы можете узнать из краткого справочника, приведенного в конце книги.

Создаем рисунок

Прежде чем мы сможем начать работать с рисунком, необходимо создать рисунок при помощи функции ImageCreate(x,x). Аргументами этой функции являются размеры картинки в пикселях. После того, как

каркас рисунка создан, можно приступить к рисованию. PHP поддерживает целый ряд функций, позволяющих создавать рисунок программными средствами.

Другой важной особенностью процесса создания рисунков является то, что все используемые в процессе создания рисунка цвета должны быть явно определены и каждому цвету необходимо дать идентификатор. Это осуществляется с применением функции ***ImageColorAllocate(image, R, G, B)***. Здесь в качестве аргументов используются указатель (идентификатор) рисунка image, который определяет, с каким рисунком будет связан данный цвет. Этот параметр возвращается функцией ***ImageCreate()*** во время создания каркаса рисунка. Параетры RGB задают интенсивности красной, зеленой и синей компоненты цвета соответственно в пределах от 0 до 255, например:

```
$gray = ImageColorAllocate($image,204,204,204);
```

Необходимо обратиться к этой функции столько раз, сколько различных цветов будет использовано в рисунке при его рисовании.

После того, как все цвета определены, можно приступить с собственно рисованию. Для этого существуют различные функции рисования линий, фигур, окружностей, эллипсов, прямоугольников, многоугольников, функции заливки и т.п. Подробно эти функции описаны в приложении. Так, например, для того, чтобы нарисовать прямую линию, используется функция ***ImageLine()***.

Нарисуем простую картинку, состоящую из синей линии, проведенной на сером фоне. Для этого напишем такой текст скрипта:

```
<?php
$image = ImageCreate(500,400);
$gray = ImageColorAllocate($image,204,204,204);
$blue = ImageColorAllocate($image,0,0,255);
ImageLine($image,50,60,450,350,$blue);
?>
```

Аргументами функции являются идентификатор рисунка, координаты начальной точки, координаты конечной точки, идентификатор цвета.

Если теперь мы запомним этот текст в файле и загрузим его в броузер, предварительно обработав на сервере с помощью PHP, то мы ничего не увидим. Это произойдет потому, что мы не предусмотрели никакого вывода для созданного рисунка. Вывод созданного файла можно осуществить, обратившись к функции ***ImageJPEG()***. В этом случае созданный рисунок будет сохранен в виде файла в формате jpeg. Первым аргументом функции ***imageJPEG()*** является указатель im созданного рисунка, второй аргумент — имя файла. После того, как файл создан, его можно посмотреть, для этого можно направить его броузеру. Окончательный вариант скрипта (файл image1.php) выглядит следующим образом:

```
<?php
$image = ImageCreate(500,400);
$gray = ImageColorAllocate($image,204,204,204);
```

```
$blue = ImageColorAllocate($image,0,0,255);
ImageLine($image,50,60,450,350,$blue);
ImageJPEG($image,"file.jpg");
?>
<html>
<head>
<title>
Создаем рисунок средствами PHP
</title>
</head>
<body>
<h3>Это созданный нами файл рисунка:</H3>
<P>
<img src="file.jpg">
</P>
</body>
</html>
```

В окне броузера мы увидим такую картинку (рис. 5.1).

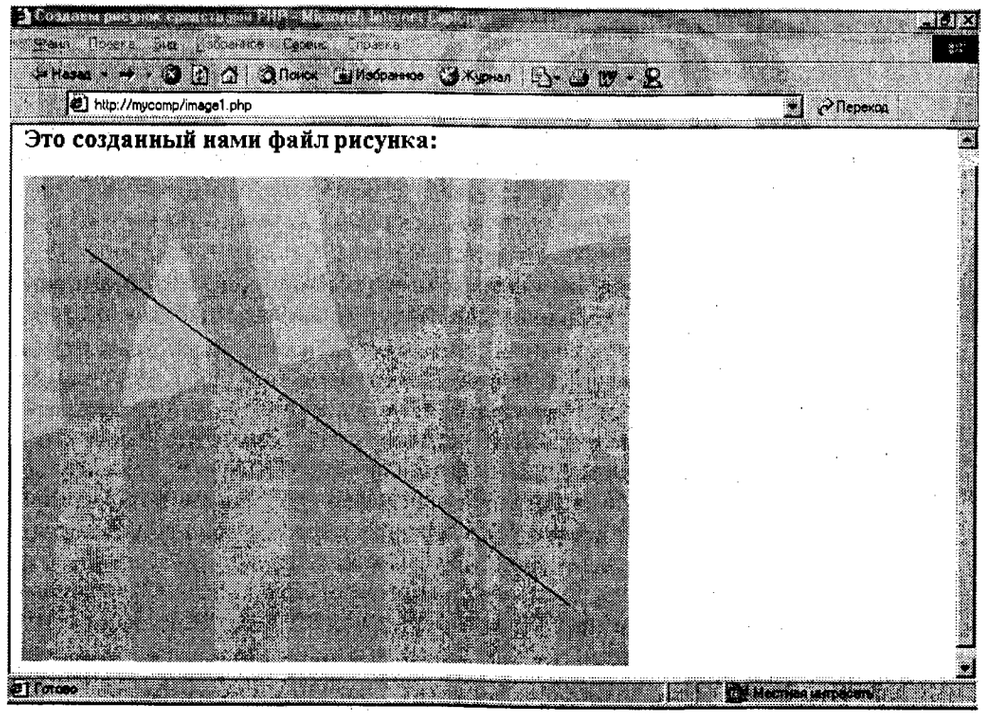


Рис. 5.1. *Создаем картинку средствами PHP*

Скрытый вывод картинок

Остановим внимание на том, что возвращаемый в предыдущем примере HTML-код содержит информацию о рисунке, его имя, местоположение. Если мы нажмем кнопку броузера «Вид» и просмотрим полученное содержимое в текстовом редакторе «Блокнот», то сможем прочитать весь код в том виде, как он получен броузером. Он будет выглядеть так:

```
<html>
<head>
<title>
Создаем рисунок средствами PHP
</title>
</head>
<body>
<h3>Это созданный нами файл рисунка:</H3>
<P>
<img src="file.jpg">
</P>
</body>
</html>
```

Мы знаем, что броузер получил рисунок file.jpg, расположенный в том же каталогое, в котором находится исходный файл image1.php.

PHP предоставляет замечательную возможность осуществления отправки броузеру файлов изображений таким образом, что исходный файл и его местоположение будут скрыты от пользователя. Более того, файл на диске локальной файловой системы сервера вообще может не быть создан.

Для того, чтобы пояснить как такое может быть сделано, рассмотрим пример. Файл image.php — это видоизмененный текст предыдущего примера. Сейчас сохраним пример в виде файла image.php.

Файл image.php

```
<?php
Header("Content-type: image/jpeg");
$image = ImageCreate(500,450);
$gray = ImageColorAllocate($image,204,204,204);
$blue = ImageColorAllocate($image,0,0,255);
ImageLine($image,50,60,450,300,$blue);
ImageJPEG($image);
ImageDestroy($image);
?>
```

В этом примере мы внесли несколько изменений в предыдущий вариант (в файл image1.php). Здесь завершает пример функция ImageDestroy, ее аргумент — указатель на рисунок. Эта функция уничтожает данный рисунок и освобождает память. Для нас более важным является инструкция header(«content-type: image/jpeg»). Эта инструкция посылает броузеру заголовок, который содержит информацию о типе посылаемых данных. В данном случае данные будут представлять собой содержимое картинки в формате jpeg. Другим важным моментом является тот факт, что в функции ***ImageJPEG()*** мы указали единственный аргумент — идентификатор рисунка, а файл для сохранения рисунка не указан. Именно этот факт послужил индикатором для того, чтобы рисунок был направлен прямо броузеру, обратившемуся к файлу image.php.

Теперь в окне броузера мы увидим ту же картинку, что и раньше, но если мы попытаемся рассмотреть полученный HTML-код, то броузер нам ничего полезного не предоставит, соответствующая опция будет выключена (рис. 5.2).

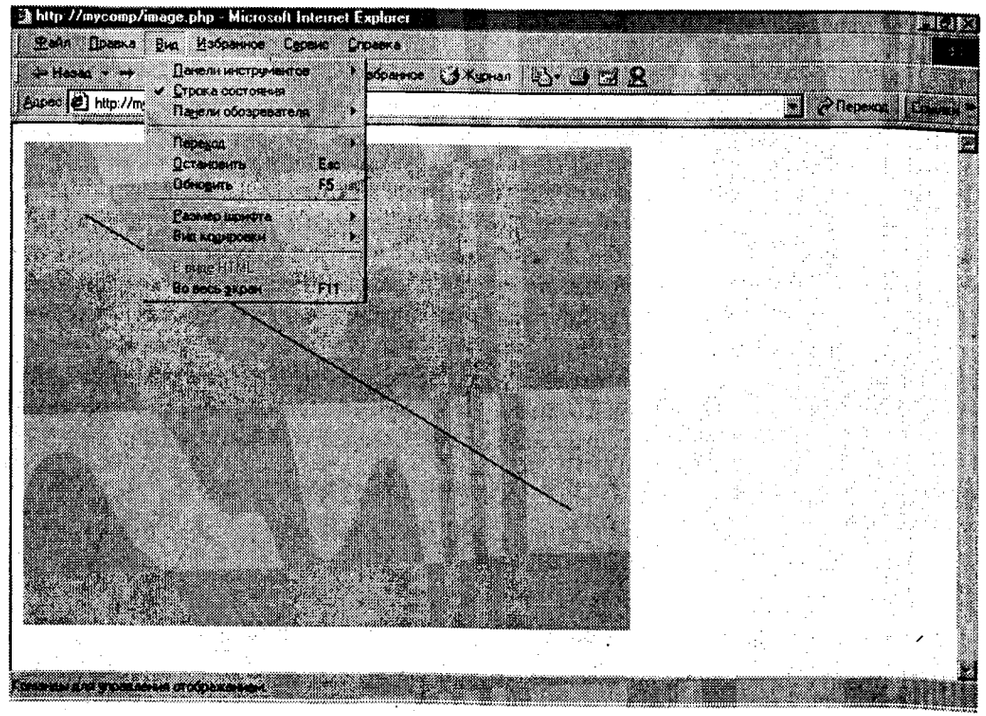


Рис. 5.2. *Рисунок получен броузером, но посмотреть HTML-код невозможно*

Внешние рисунки

В предыдущем примере мы направили броузеру рисунок, созданный средствами PHP. При этом пользователь не имеет возможности определить местонахождение источника рисунка.

Аналогичную задачу можно поставить при необходимости отправить броузеру рисунок, существующий в виде файла на жестком диске с тем условием, чтобы пользователь не имел возможности определить место хранения этого рисунка. Средствами PHP такая задача решается очень просто. Рассмотрим файл image1.phtm — это и есть решение поставленной задачи.

Файл image1.phtm

```
<?php
header ("Content-type: image/jpeg");
$im = ImageCreateFromJPEG ("001.jpg");
imagejpeg ($im);
ImageDestroy($im);
?>
```

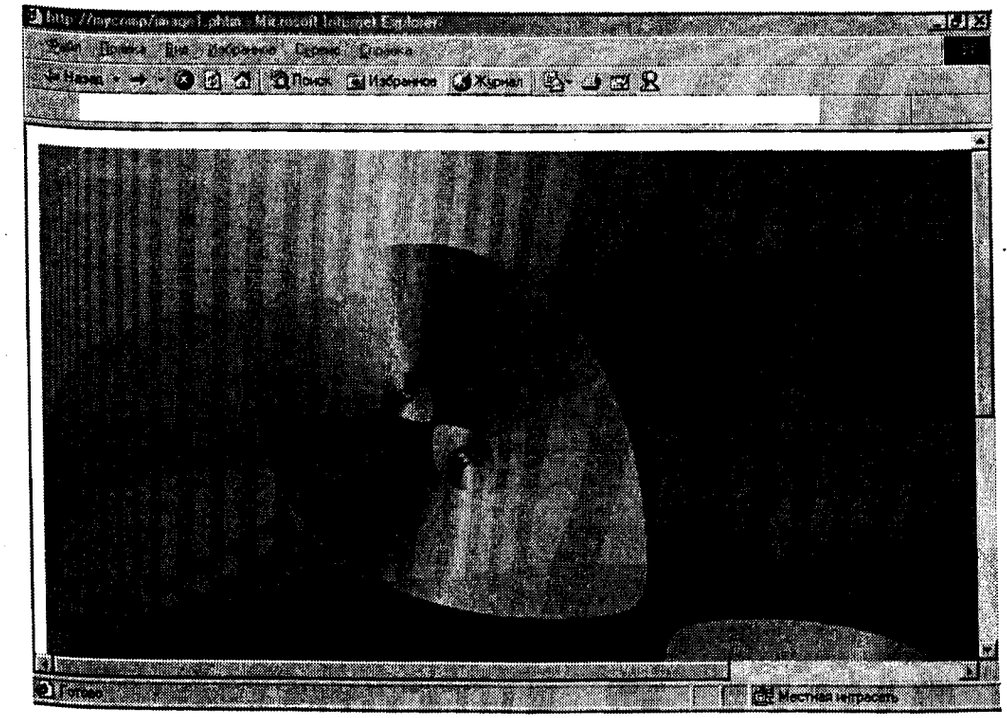
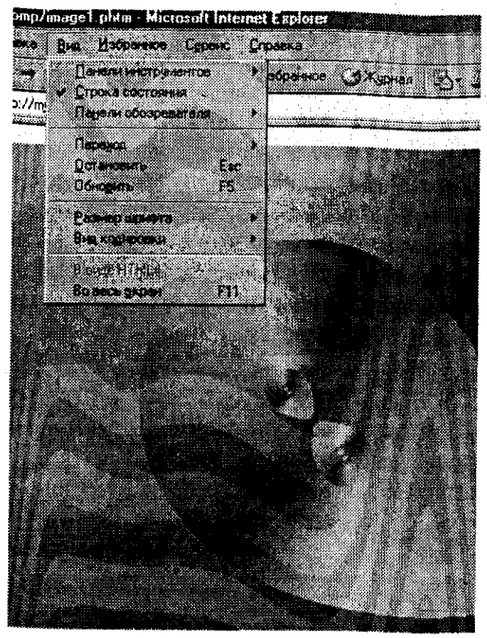


Рис. 5.3. *Картинка отправлена броузеру, но пользователь не имеет возможности определить место хранения файла рисунка*



С помощью этого кода мы создаем рисунок $im из файла 001.jpg, направляем его броузеру, предварительно отправив ему заголовок с указанием типа данных. Затем уничтожаем рисунок. В окне броузера видим картинку (рис. 5.3).

Пользователь, получивший эту картинку, не имеет возможности определить место хранения файла этого рисунка, он не может посмотреть HTML-код (рис. 5.4).

Есдли же пользователь попытается сохранить файл, выбрав в меню Файл пункт сохранить, то ему будет предложено сохранить его в формате HTML (рис. 5.5).

Рис. 5.4. *Пользователь не увидит HTML-код страницы*

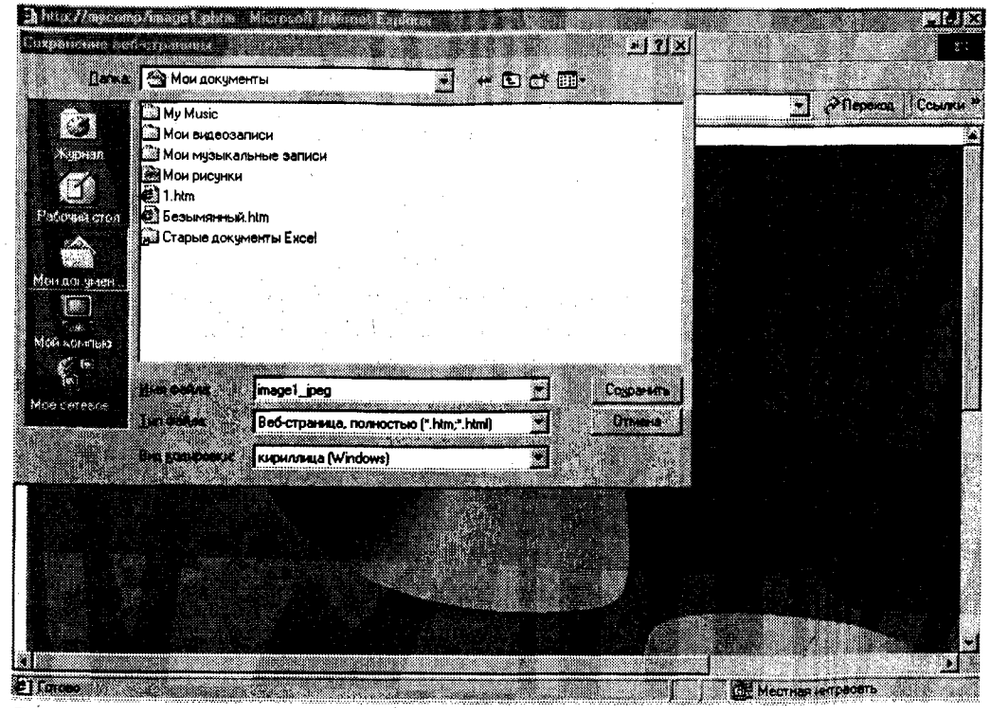


Рис. 5.5. *Сохраняем полученный рисунок*

Сохраним файл image1_jpeg.htm, а заетм посмотрим, что в нем содержится с помощью «Блокнота». Вот что сохранилось в файле image1_jpeg.htm:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!- saved from url=(0025)http://mycomp/image1.phtm ->
<HTML><HEAD>
<META http-equiv=Content-Type content="text/html; charset=windows-1251">
<META content="MSHTML 5.50.4611.1300" name=GENERATOR></HEAD>
<BODY><IMG src="image1_jpeg.files/image1.jpeg"></BODY></HTML>
```

Как и следовало предположить, никакой информации о местоположении источника в этом файле не содержится. Здесь есть ссылка на файл, который инициализировал загрузку рисунка — это файл image1.phtm. Но содержимое этого файла недоступно пользователю. Есть также ссылка на локальный файл с изображением. Этот файл был сохранен тогда, когда мы выбрали опцию сохранения текущей страницы, открытой в броузере.

Информация о PHP

В PHP существует очень полезная функция, с помощью которой можно получить информацию о многих свойствах и возможностях PHP, установленного на данном конкретном сервере. Это функция ***phpinfo()***. Создадим файл phpinfo.php, в котором будет содержаться всего одна строчка.

Файл phpinfo.php.

```
<?php phpinfo()?>
```

Загрузим файл в броузер с сервера, на экране в окне броузера мы увидим информацию о текущей версии PHP (рис. 5.6).

В появившемся на экране броузера файле содержится много полезной информации.

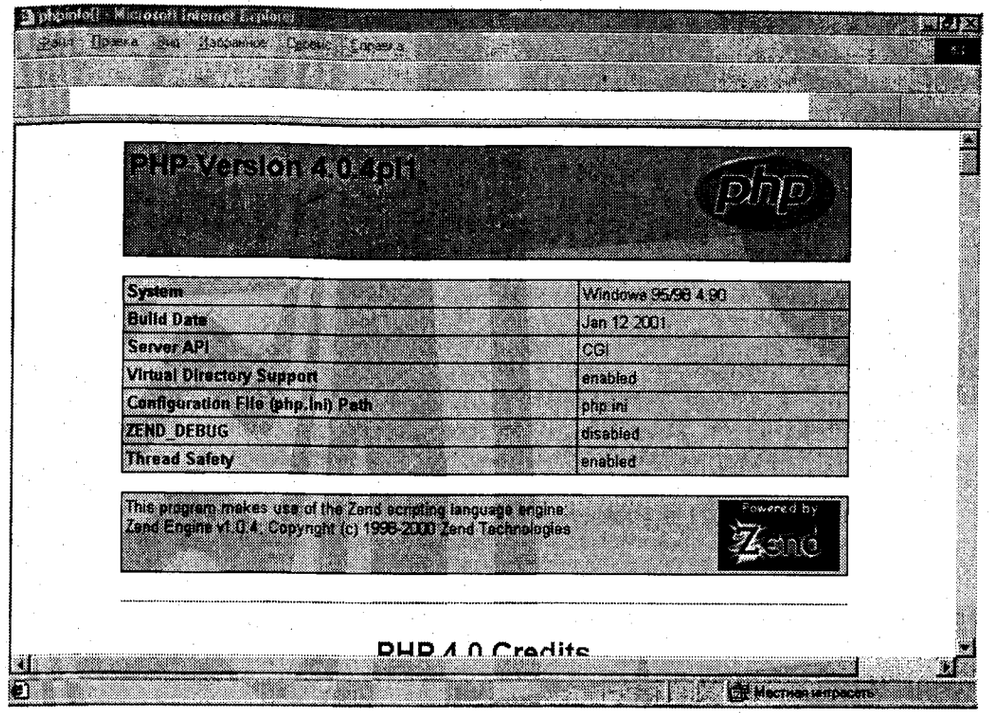


Рис. 5.6. *Текущая информация о PHP*

Информация, полученная с помощью phpinfo()

PHP Version 4.0.4pl1

Здесь содержатся общие сведения о PHP.

SYSTEM	*WINDOWS 95/98 4.90*
Build Date	*Jan 12 2001*
Server API	*CGI*
Virtual Directory Support	*enabled*
Configuration File (php.ini) Path	*php.ini*
ZEND_DEBUG	*disabled*
Thread Safety	*enabled*

Configuration

PHP Core

Здесь указаны основные параметры конфигурации PHP

Директива	Локальное значение	Основное значение
allow_call_time_pass_reference	*On*	*On*
allow_url_fopen	*1*	*1*
arg_separator	*&*	*&*
asp_tags	*Off*	*Off*
auto_append_file	*no value*	*no value*
auto_prepend_file	*no value*	*no value*
Browscap	*no value*	*no value*
default_charset	*no value*	*no value*
default_mimetype	*text/html*	*text/html*
define_syslog_variables	*Off*	*Off*
disable_functions	*no value*	*no value*
display_errors	*On*	*On*
display_startup_errors	*Off*	*Off*
doc_root	*no value*	*no value*
enable_dl	*On*	*On*
error_append_string	*Off*	*Off*
error_log	*no value*	*no value*
error_prepend_string	*Off*	*Off*
error_reporting	*2047*	*2047*
expose_php	*On*	*On*
extension_dir	*./*	*./*
file_uploads	*1*	*1*
gpc_order	*GPC*	*GPC*
highlight.bg	*#FFFFFF*	*#FFFFFF*
highlight.comment	*#FF8000*	*#FF8000*
highlight.default	*#0000BB*	*#0000BB*
highlight.html	*#000000*	*#000000*
highlight.keyword	*#007700*	*#007700*
highlight.string	*#DD0000*	*#DD0000*
html_errors	*On*	*On*
ignore_user_abort	*Off*	*Off*
implicit_flush	*Off*	*Off*

Продолжение таблицы

Директива	Локальное значение	Основное значение
include_path	no value	no value
log_errors	Off	Off
magic_quotes_gpc	On	On
magic_quotes_runtime	Off	Off
magic_quotes_sybase	Off	Off
max_execution_time	300	300
open_basedir	no value	no value
output_buffering	Off	Off
output_handler	no value	no value
post_max_size	8M	8M
Precision	14	14
register_argc_argv	On	On
register_globals	On	On
safe_mode	Off	Off
safe_mode_exec_dir	no value	no value
sendmail_from	me@localhost.com	me@localhost.com
sendmail_path	no value	no value
short_open_tag	On	On
SMTP	Your.mail.server.ru	your.mail.server.ru
sql.safe_mode	Off	Off
track_errors	Off	Off
upload_max_filesize	2M	2M
upload_tmp_dir	no value	no value
user_dir	no value	no value
Variables_order	EGPCS	EGPCS
y2k_compliance	Off	Off

Domxml

Здесь указаны параметры конфигурации для работы с DOM/XML.

DOM/XML, Xpath, XPointer Support	enabled
libmxl Version	2.2.11

Gd

Здесь перечислены параметры конфигурации для работы с рисунками.

GD Support

GD Support	enabled
GD Version	1.6.2 or higher
FreeType Support	enabled
FreeType Linkage	with TTF library
JPG Support	enabled
PNG Support	enabled
WBMP Support	enabled

Standard

Параметры для работы с внешними библиотеками

Regex Library	Bundled library enabled
Dynamic Library Support	enabled
Internal Sendmail Support for Windows 4	enabled

Стандартные директивы

Директива	Локальное значение	Стандартное значение
assert.active	1	1
assert.bail	0	0
assert.callback	no value	no value
assert.quiet_eval	0	0
assert.warning	1	1
safe_mode_allowed_env_vars	PHP_	PHP_
safe_mode_protected_env_vars	LD_LIBRARY_PATH	LD_LIBRARY_PATH
session.use_trans_sid	1	1
url_rewriter.tags	a=href, area=href, frame=src, input=src, form=fakeentry	a=href, area=href, frame=src, input=src, form=fakeentry

Bcmath

Поддержка математических функций bcmath

BCMath support	enabled

Calendar

Функции для работы с календарем

Calendar support	enabled

Com

Работа с com

Директива	Локальное значение	Стандартное значение
allow_dcom	Off	Off
typelib_file	no value	no value

ftp

Поддержка ftp

FTP support	enabled

Mysql

Поддержка mysql

MySQL Support	enabled
Active Persistent Links	0
Active Links	0
Client API version	3.23.22-beta

Директивы, связанные с mysql

Директива	Локальное значение	Стандартное значение
mysql.allow_persistent	On	On
mysql.default_host	no value	no value
mysql.default_password	no value	no value
mysql.default_port	no value	no value
mysql.default_socket	no value	no value
mysql.default_user	no value	no value
mysql.max_links	Unlimited	Unlimited
mysql.max_persistent	Unlimited	Unlimited

Odbc

Установки для работы с базами данных

ODBC Support	enabled
Active Persistent Links	0
Active Links	0
ODBC library	Win32

Директивы для работы с базами данных

Директива	Локальное значение	Стандартное значение
odbc.allow_persistent	On	On
odbc.check_persistent	On	On
odbc.default_db	no value	no value
odbc.default_pw		
odbc.default_user	no value	no value
odbc.defaultbinmode	return as is	return as is
odbc.defaultlrl	return up to 4096 bytes	return up to 4096 bytes
odbc.max_links	Unlimited	Unlimited
odbc.max_persistent	Unlimited	Unlimited

Pcre

Конфигурация PCRE

PCRE (Perl Compatible Regular Expressions) Support	enabled
PCRE Library Version	3.1 09-Feb-2000

Session

Конфигруация поддержки работы с сессиями

Session Support	enabled

Директивы

Директива	Локальное значение	Основное значение
session.auto_start	Off	Off
session.cache_expire	180	180
session.cache_limiter	nocache	nocache
session.cookie_domain	no value	no value
session.cookie_lifetime	0	0
session.cookie_path	/	/
session.cookie_secure	Off	Off
session.entropy_file	no value	no value
session.entropy_length	0	0
session.gc_maxlifetime	1440	1440
session.gc_probability	1	1
session.name	PHPSESSID	PHPSESSID
session.referer_check	no value	no value
session.save_handler	files	files
session.save_path	D:\Program Files\phpnew\sessiondata	D:\Program Files\phpnew\sessiondata
session.serialize_handler	php	php
session.use_cookies	On	On

9Xml

Поддержка XML

XML Support	active

wddx

WDDX Support	enabled

Environment

Системные переменные окружения

Переменная	Значение
BLASTER	A220 I7 D1 H5 P330 T6
CLASSPATH	C:\JAVA\JAVA\LIB\CLASSES.ZIP
COMSPEC	C:\WIN\COMMAND.COM
CONTENT_LENGTH	0
CTSYN	C:\WIN
GATEWAY_INTERFACE	CGI/1.1
HTTP_ACCEPT	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,8application/vnd.ms-excel, application/msword, application/vnd.ms-powerpoint, */*
HTTP_ACCEPT_LANGUAGE	ru
HTTP_CONNECTION	Keep-Alive
HTTP_HOST	mycomp
HTTP_REFERER	http://mycomp/
HTTP_ACCEPT_ENCODING	gzip, deflate
HTTP_USER_AGENT	Mozilla/4.0 (compatible; MSIE 5.5; Windows 98; Win 9 x 4.90)
PATH	C:\WIN;C:\WIN\COMMAND;C:\WINDOWS;C:\WINDOWS\COMMAND;C:\WIN1;C:\WIN1\COMMAND;C:\WINDOWS.000;C:\WINDOWS.000\COMMAND;C:\VC;C:\DOS;C:\ARC;C:\TOOLS;C:\JAVA\JAVA\BIN;C:\ProgramFiles\Mts;C:\Program Files\Mts
PATH_INFO	/phpinfo.php
PATH_TRANSLATED	C:\WebShare\wwwroot\phpinfo.php
PROMPT	pg
REMOTE_ADDR	127.0.0.1
REMOTE_HOST	127.0.0.1
REQUEST_METHOD	GET
SCRIPT_NAME	/phpinfo.php
SERVER_NAME	mycomp
SERVER_PORT	80
SERVER_PORT_SECURE	0
SERVER_PROTOCOL	HTTP/1.1
SERVER_SOFTWARE	Microsoft-PWS-95/2.0
TEMP	C:\WINDOWS.000\TEMP
TMP	C:\WINDOWS.000\TEMP
Winbootdir	C:\WIN
Windir	C:\WIN

PHP Variables

Переменные PHP

Переменная	Значение
PHP_SELF	/phpinfo.php
HTTP_SERVER_VARS["BLASTER"]	A220 I7 D1 H5 P330 T6
HTTP_SERVER_VARS["CLASSPATH"]	C:\\JAVA\\JAVA\\LIB\\CLASSES.ZIP
HTTP_SERVER_VARS["COMSPEC"]	C:\\WIN\\COMMAND.COM
HTTP_SERVER_VARS["CONTENT_LENGTH"]	0
HTTP_SERVER_VARS["CTSYN"]	C:\\WIN
HTTP_SERVER_VARS["GATEWAY_INTERFACE"]	CGI/1.1
HTTP_SERVER_VARS["HTTP_ACCEPT"]	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/msword, application/vnd.ms-powerpoint, */*
HTTP_SERVER_VARS["HTTP_ACCEPT_LANGUAGE"]	ru
HTTP_SERVER_VARS["HTTP_CONNECTION"]	Keep-Alive
HTTP_SERVER_VARS["HTTP_HOST"]	mycomp
HTTP_SERVER_VARS["HTTP_REFERER"]	http://mycomp/
HTTP_SERVER_VARS["HTTP_ACCEPT_ENCODING"]	gzip, deflate
HTTP_SERVER_VARS["HTTP_USER_AGENT"]	Mozilla/4.0 (compatible; MSIE 5.5; Windows 98; Win 9x 4.90)
HTTP_SERVER_VARS["PATH"]	C:\\WIN;C:\\WIN\\COMMAND;C:\\WINDOWS; C:\\WINDOWS\\COMMAND;C:\\WIN1; C:\\WIN1\\COMMAND;C:\\WINDOWS.000; C:\\WINDOWS.000\\COMMAND;C:\\VC; C:\\DOS;C:\\ARC;C:\\TOOLS; C:\\JAVA\\JAVA\\BIN;C:\\ProgramFiles\\Mts; C:\\Program Files\\Mts
HTTP_SERVER_VARS["PATH_INFO"]	/phpinfo.php
HTTP_SERVER_VARS["PATH_TRANSLATED"]	C:\\WebShare\\wwwroot\\phpinfo.php
HTTP_SERVER_VARS["PROMPT"]	pg
HTTP_SERVER_VARS["REMOTE_ADDR"]	127.0.0.1
HTTP_SERVER_VARS["REMOTE_HOST"]	127.0.0.1
HTTP_SERVER_VARS["REQUEST_METHOD"]	GET
HTTP_SERVER_VARS["SCRIPT_NAME"]	/phpinfo.php
HTTP_SERVER_VARS["SERVER_NAME"]	mycomp
HTTP_SERVER_VARS["SERVER_PORT"]	80
HTTP_SERVER_VARS["SERVER_PORT_SECURE"]	0

Переменные PHP (продолжение)

Переменная	Значение
HTTP_SERVER_VARS["SERVER_PROTOCOL"]	HTTP/1.1
HTTP_SERVER_VARS["SERVER_SOFTWARE"]	Microsoft-PWS-95/2.0
HTTP_SERVER_VARS["TEMP"]	C:\\WINDOWS.000\\TEMP
HTTP_SERVER_VARS["TMP"]	C:\\WINDOWS.000\\TEMP
HTTP_SERVER_VARS["winbootdir"]	C:\\WIN
HTTP_SERVER_VARS["windir"]	C:\\WIN
HTTP_SERVER_VARS["PHP_SELF"]	/phpinfo.php
HTTP_SERVER_VARS["argv"]	Array()
HTTP_SERVER_VARS["argc"]	0
HTTP_ENV_VARS["BLASTER"]	A220 I7 D1 H5 P330 T6
HTTP_ENV_VARS["CLASSPATH"]	C:\\JAVA\\JAVA\\LIB\\CLASSES.ZIP
HTTP_ENV_VARS["COMSPEC"]	C:\\WIN\\COMMAND.COM
HTTP_ENV_VARS["CONTENT_LENGTH"]	0
HTTP_ENV_VARS["CTSYN"]	C:\\WIN
HTTP_ENV_VARS["GATEWAY_INTERFACE"]	CGI/1.1
HTTP_ENV_VARS["HTTP_ACCEPT"]	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/msword, application/vnd.ms-powerpoint, */*
HTTP_ENV_VARS["HTTP_ACCEPT_LANGUAGE"]	ru
HTTP_ENV_VARS["HTTP_CONNECTION"]	Keep-Alive
HTTP_ENV_VARS["HTTP_HOST"]	mycomp
HTTP_ENV_VARS["HTTP_REFERER"]	http://mycomp/
HTTP_ENV_VARS["HTTP_ACCEPT_ENCODING"]	gzip, deflate
HTTP_ENV_VARS["HTTP_USER_AGENT"]	Mozilla/4.0 (compatible; MSIE 5.5; Windows 98; Win 9x 4.90)
HTTP_ENV_VARS["PATH"]	C:\\WIN;C:\\WIN\\COMMAND;C:\\WINDOWS; C:\\WINDOWS\\COMMAND;C:\\WIN1; C:\\WIN1\\COMMAND;C:\\WINDOWS.000; C:\\WINDOWS.000\\COMMAND;C:\\VC; C:\\DOS;C:\\ARC;C:\\TOOLS; C:\\JAVA\\JAVA\\BIN;C:\\ProgramFiles\\Mts; C:\\Program Files\\Mts
HTTP_ENV_VARS["PATH_INFO"]	/phpinfo.php
HTTP_ENV_VARS["PATH_TRANSLATED"]	C:\\WebShare\\wwwroot\\phpinfo.php
HTTP_ENV_VARS["PROMPT"]	pg

Переменные PHP (продолжение)

Переменная	Значение
HTTP_ENV_VARS["REMOTE_ADDR"]	127.0.0.1
HTTP_ENV_VARS["REMOTE_HOST"]	127.0.0.1
HTTP_ENV_VARS["REQUEST_METHOD"]	GET
HTTP_ENV_VARS["SCRIPT_NAME"]	/phpinfo.php
HTTP_ENV_VARS["SERVER_NAME"]	mycomp
HTTP_ENV_VARS["SERVER_PORT"]	80
HTTP_ENV_VARS["SERVER_PORT_SECURE"]	0
HTTP_ENV_VARS["SERVER_PROTOCOL"]	HTTP/1.1
HTTP_ENV_VARS["SERVER_SOFTWARE"]	Microsoft-PWS-95/2.0
HTTP_ENV_VARS["TEMP"]	C:\\WINDOWS.000\\TEMP
HTTP_ENV_VARS["TMP"]	C:\\WINDOWS.000\\TEMP
HTTP_ENV_VARS["winbootdir"]	C:\\WIN

PHP License

Лицензия PHP

This program is free software; you can redistribute it and/or modify it under the terms of the PHP License as published by the PHP Group and included in the distribution in the file: LICENSE

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

If you did not receive a copy of the PHP license, or have any questions about PHP licensing, please contact license@php.net.

Полученная таким образом информация позволит узнать возможности PHP и использовать язык наиболее оптимальным образом. Некоторые переменные могут быть изменены программными средствами. Большую помощь могут оказать другие переменные, информация о которых не выведена функцией ***phpinfo()***, а также постоянные PHP. Информацию о переменных и постоянных можно получить в приложении А (Краткий справочник по функциям PHP).

Текст в рисунках

В PHP существует несколько функций, которые позволяют выводить текст поверх рисунка. Рассмотрим простой пример вывода текста с применением фонта TTF. Для этого вначале откроем рисунок, определим два цвета для текста и для фона, укажем размер шрифта, местоположение файла, содержащего фонт, размер шрифта — и программа готова. Скрипт поместим в файл image3.phtm.

Файл image3.phtm

```
<?php
Header ("Content-type: image/jpeg");
$im = imagecreate (800, 900);
$black = ImageColorAllocate ($im, 0, 0, 0);
$white = ImageColorAllocate ($im, 255, 255, 255);
ImageTTFText ($im, 60, 0, 600, 400, $white, "C:/win/fonts/arial.ttf",
"Writing text...");
ImageJPEG ($im);
ImageDestroy ($im);
?>
```

По сравнению с теми примерами, которые мы рассматривали ранее, здесь имеется новая функция ***ImageTTFText()***. Эта функция используется для рисования текста с помощью фонта TTF. Аргументы функции (по порядку их появления) таковы: указатель рисунка, размер шрифта, угол наклона (отсчитывается против часовой стрелки, указывается в градусах, нулевому значению соответствует положение стрелки на трех часах), начальная точка местоположения текста, цвет шрифта, полный путь к файлу, содержащему фонт, наконец, строка, которая будет выведена в виде текста. Возвращаемые этой функцией значения (массив), а также прочие аргументы, которые использовать не обязательно, описаны в приложении в кратком справочнике по функциям.

Если мы запросим файл image3.phtm, то получим такой ответ (рис. 5.7).

Слегка изменив параметры функции ***ImageTTFText()***, увидим другой вариант текста (файл image3_1.phtm) (рис. 5.8).

В этом примере функция ***imageTTFText()*** выглядит так:

```
ImageTTFText ($im, 60, 165, 600, 400, $white, "C:/win/fonts/
times.ttf",
        "Writing text...");
```

Текст может быть написан и поверх рисунка (файл image4.phtm):

```
<?php
Header ("Content-type: image/jpeg");
$im = ImageCreateFromJPEG ("001.jpg");
```

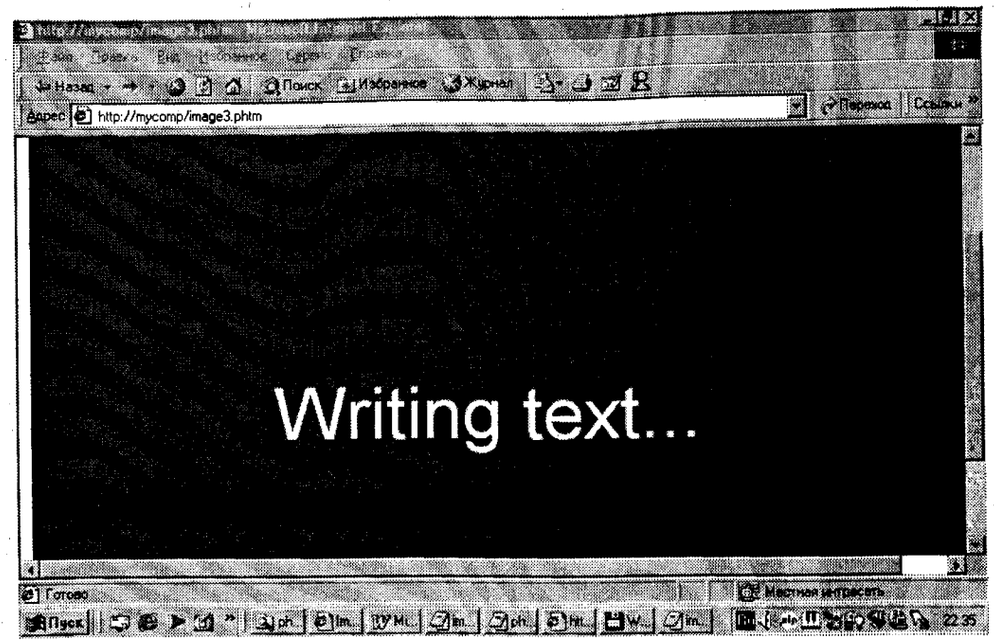


Рис. 5.7. *Пишем текст в рисунке*

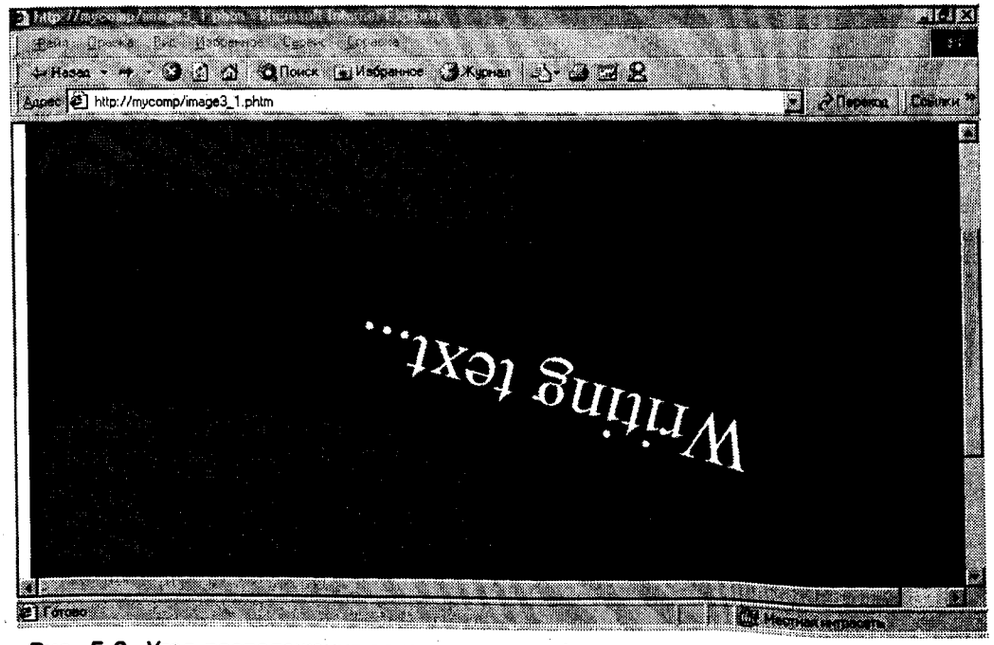


Рис. 5.8. *Угол расположения текста, размер, фонт и другие параметры можно изменять*


```
$black = ImageColorAllocate ($im, 0, 0, 0);
$white = ImageColorAllocate ($im, 255, 255, 255);
ImageTTFText ($im, 60, 135, 600, 400, $black, "C:/win/fonts/
arial.ttf",
     "Writing text...");
ImageJPEG ($im);
ImageDestroy ($im);
?>
```

В окне броузера мы увидим такую картинку (рис. 5.9).

Поскольку параметры выводимого шрифта представляются в виде числовых и строковых аргументов функции, их легко можно менять, например, в зависимости от сколь угодно интересных обстоятельств, скажем от времени суток. Такой вариант приведен в файле image4_1.phtm:

```
<?php
Header ("Content-type: image/jpeg");
$im = ImageCreateFromJPEG ("001.jpg");
$black = ImageColorAllocate ($im, 0, 0, 0);
$white = ImageColorAllocate ($im, 255, 255, 255);
```

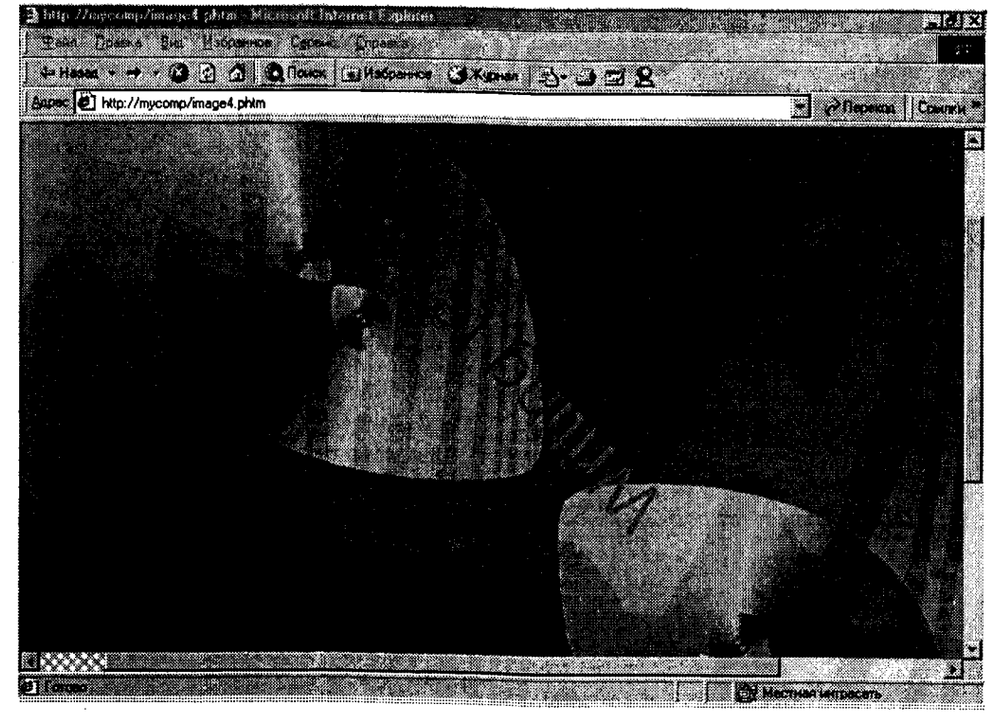


Рис. 5.9. *Пишнем текст поверх рисунка*

```
$time = localtime();
$a = 450 - 30*$time[2];
ImageTTFText ($im, 60, $a, 600, 400, $black, "C:/win/fonts/
arial.ttf",
                "Writing text...");
ImageJPEG ($im);
ImageDestroy ($im);
?>
```

Этот скрипт посылает броузеру рисунок с надписью. Которая располагается в направлении часовой стрелки в соответствии с локальным временем на сервере. Так, если локальное время сервера 23 часа, то мы получим такой рисунок (рис. 5.10).

Рис. 5.10. *Выводи текст под углом, соответствующим локальному времени на сервере*

Иные возможности

PHP предоставляет возможность создавать программы, исполняемые на сервере. Это не единственный язык, разработанный специально для того, чтобы создавать гибкие Интернет-приложения. Существуют другие языки, например, язык Perl, которые позволяют создавать программы, выполняемые на сервере. Стандартные языки сценариев, например, язык JavaScript, также предоставляют богатые возможности управления выводимым содержанием web-страниц на сервере перед отправкой клиенту. Особенно полезен он оказывается при использовании технологии ASP компании Microsoft. Модуль ASP удобно устанавливается на платформе Windows и сразу после установки без необходимости дополнительного конфигурирования системы готов к работе.

При использовании технологии ASP необходимо придерживаться таких правил. Файлы с программами на языке JavaScript (или Jscript или VBScript) должны иметь расширение .asp и располагаться в выполняемом каталоге сервера. Фрагмент скрипта (выполняемый текст программы) отделяется от HTML-текста файла ярлыком <%@ (открывающий) и %> (закрывающий), скрипт должен начинаться с указания используемого языка: Language=«JavaScript». Можно использовать конструкцию <script language=«JavaScript» RUNAT=«Server»>. В последнем случае атрибут RUNAT содержит значение, соответствующее тому, что скрипт предназначен для выполнения на сервере. Однако в настоящем пособии мы не будем останавливаться на вопросах создания ASP файлов.

6

ОСНОВНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА

- Базовые элементы
- Типы
- Переменные
- Константы
- Инструкции

Язык PHP основан на известных языках программирования, таких как C, Java, Perl. Синтаксис PHP во многом заимствован у этих языков.

Базовые элементы

Выделение фрагмента PHP-кода в тексте phtm-файла

Первое, на чем важно остановиться, — это способ перехода из обычного HTML-кода к командам, написанным на PHP. Как осуществляется переход от HTML-разметки и текста к фрагменту, содержащему код PHP? Для этого используются специальные обозначения. Существует несколько способов того, чтобы показать, что в данном фрагменте располагается код PHP.

Способ 1. Для выделения фрагмента PHP-кода используются обрамляющие знаки </ и ?>. Например:

```
<? Echo("Здесь содержится текст, который будет выведен в окне броузера
\n"); ?>
```


Способ 2. Фрагмент PHP кода помещается между знаками <?php и ?>. Например:

```
<?php echo("Что такое здесь сидит? \n Это доктор или кто? \n"); ?>
```

Способ 3. Использование ярлыка <script> с указанием атрибута language, значение которого равно строке, содержащей название языка PHP: <script language=”php”>. Например:

```
<script language="php">
      echo ("Чего бы мы ни написали в этом месте, этот текст будет
видим в окне броузера.");
    </script>;
```

Способ 4. Использование ярлыков в стиле asp, т.е. знаков <% и %>. При этом необходимо соответствующим образом сконфигурировать файл php.ini. Пример:

```
<% echo("Пишем текст"); %>
```

В качестве примера рассмотрим простой файл, в котором используются все указанные способы. Файл 1.phtm:

```
<html>
<head>
<title>Способы выделения PHP кода в тексте .phtm  файла.</title>
<body>
<p align="center"
```

В этом файле мы рассмотрим примеры того, какими способами можно выделить PHP-код в тексте phtm-файла.

```
<p><B>Способ 1.</B> Угловые скобки со знаками вопросов.
 <? Echo("Здесь содержится текст, который будет выведен в окне броузера
\n"); ?>
 <p><B>Способ 2.</B> Угловые скобки со знаками вопросов и словом php в
открывающей скобке.
 <?php echo("Что такое здесь сидит? \n Это доктор или кто? \n"); ?>
 <p><B>Способ 3.</B> Ярлык <i>script</i>.
<script language="php">
      echo ("Чего бы мы ни написали в этом месте, этот текст будет
видим в окне броузера.");
    </script>;
<p><B>Способ 1.</B> Угловые скобки со знаками процентов, как при
использовании asp.

<% echo("Пишем тексты"); %>
</body>
</html>
```

Сохраняем файл в основном каталоге сервера и загружаем в броузер. В результате видим что-то похожее на то, что изображено на рис. 6.1.

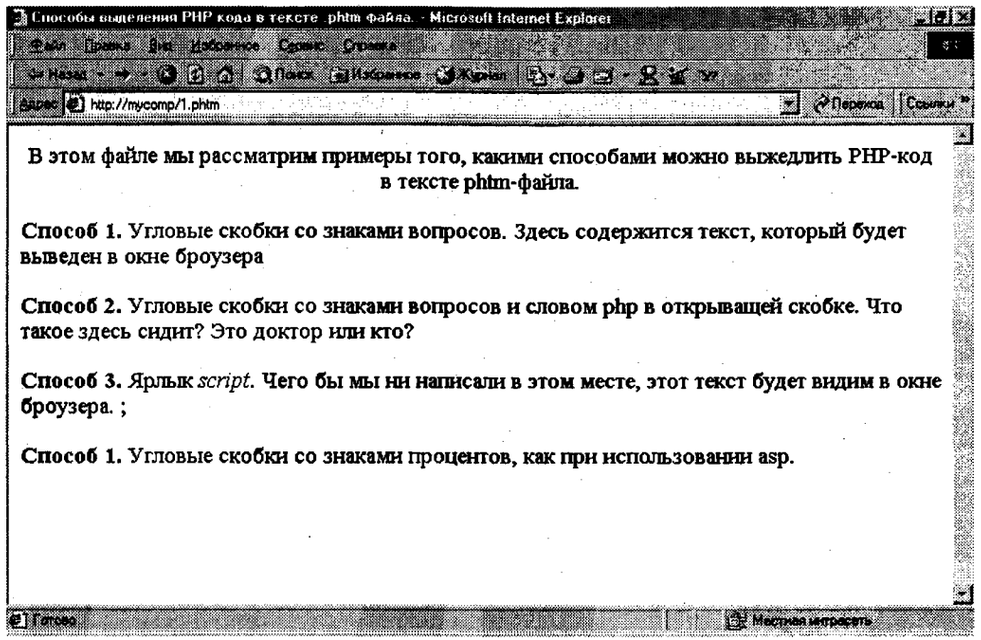


Рис. 6.1. *Способы отделения PHP-кода от остального текста phtm-файла*

Типы

В PHP поддерживаются следующие типы:

- array — массивы;
- floating-point numbers — числа с плавающей точкой («doubles»);
- integer — целые числа;
- object — объекты;
- string — строки.

Как правило, тип не устанавливается разработчиком, вместо этого он определяется интерпретатором PHP во время выполнения программы и зависит от контекста, в котором используется та или иная переменная.

Если возникает необходимость указать тип переменной в явном виде или преобразовать переменную к заданному типу, то для этого можно использовать cast или функцию ***settype()***. При этом следует иметь в виду, что в зависимости от условий различные типы ведут себя по-разному (см. преобразование типов).

Целые числа

Целое число может быть определено с использованием одного из следующих форматов инициализации:

- $a = 1234; # десятичное число;
- $a = -123; # отрицательное число;
- $a = 0123; # восьмиричное число (начинается с 0, равно десятичному 83);
- $a = 0x12; # шестнадцатиричное (равно десятичному 18).

Максимально допустимое значение зависит от платформы, но, как правило, максимальное значение может задаваться в диапазоне 32.

Числа с плавающей точкой

Числа с плавающей точкой («doubles») могут записываться в одном из следующих вариантов:

- $a = 1,234;
- $a = 1,2e3.

Максимальное значение числа с плавающей точкой также зависит от платформы, как правило, оно достигает значения ~1,8e308 при этом допустимая точность составляет 14 десятичных знаков (64 бит).

Предупреждение.

*Следует отметить, что десятичные дроби, например, 0,1 или 0,7, не могут быть превращены в двоичное представление с абсолютной сохранностью точности. Так, например, floor((0,1+0,7)*10) как правило возвращает значение 7, а не предполагаемое значение 8, поскольку выражение в скобках оказывается равным примерно значению 7,9999999999.... Этот факт является следствием того, что невозможно точно представить в виде десятичной записи дробные выражения, например, такие как 1/3.*

Строки

Строки могут быть заданы с использованием одного из двух видов кавычек. При использовании двойных кавычек переменные могут быть раскрыты. Возможно использование специальных наборов символов.

При использовании одиночных кавычек можно использовать только две специальные последовательности символов «\\» и «\'».

Строки можно задавать с использованием синтаксис doc, т.е. при помощи знака «<<<», за которым указывается идентификатор. После этой комбинации следует строка. Строка заканчивается там, где в начале новой строки стоит тот же идентификатор, который открывал строковое значение и был расположен после <<<. Такая строка ведет себя так, как если бы она была помещена между знаками двойных

кавычек. При этом можно пользоваться последовательностями специальных символов (табл. 6.1).

Таблица 6.1
Специальные символы

Символы	Значение
\n	Перевод строки (LF или 0x0A в ASCII)
\r	Возврат каретки (CR или 0x0D в ASCII)
\t	Горизонтальная табуляция (HT или 0x09 в ASCII)
\\	Обратная черта
\$	Символ доллара
\"	Двойная кавычка
\[0-7]{1,3}	Последовательность символов, соответствующая регулярному выражению в восьмиричном представлении
\x[0-9A-Fa-f]{1,2}	Последовательность символов, соответствующих регулярному выражению в шестнадцатиричном представлении

Пример

```
<?php
$str = <<<EOD
Example of string
spanning multiple lines
using heredoc syntax.
EOD;

/* Более сложный пример с переменными. */
class foo {
     var $foo;
     var $bar;

     function foo() {
          $this->foo = 'Foo';
          $this->bar = array('Bar1', 'Bar2', 'Bar3');
     }
}

$foo = new foo();
$name = 'MyName';
echo <<<EOT
```



```
My name is "$name". I am printing some $foo->foo.
Now, I am printing some {$foo->bar[1]}.
This should print a capital 'A': \x41
EOT;
?>
```

Строки могут быть объединены при помощи оператора объединения строк '.' (точка). Отметим, что оператор сложения '+' не будет работать в такой ситуации. Символы внутри строки можно рассматривать в некотором смысле как элементы массива.

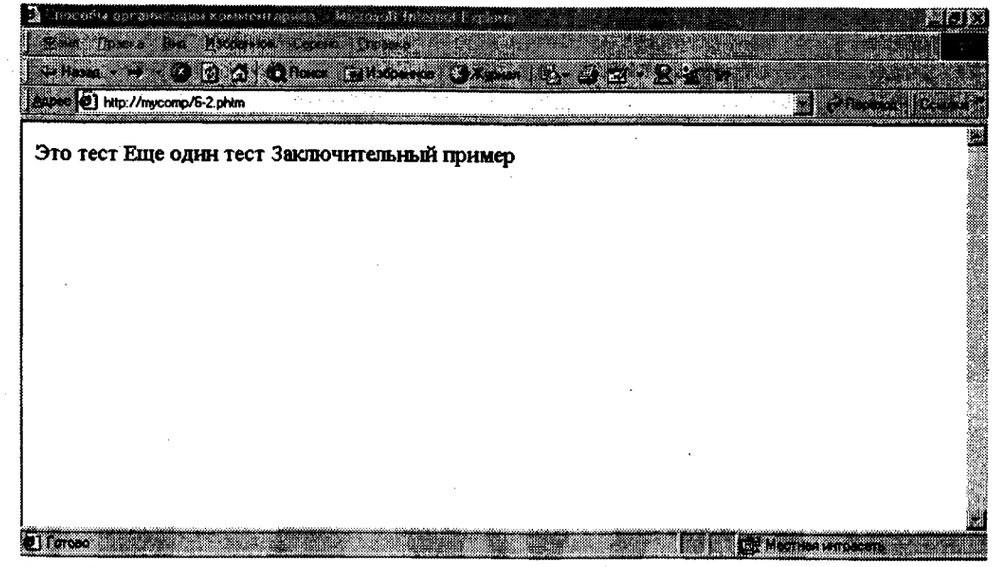


Рис. 6.2.

Пример работы со строками

```
<?php
/* Присваивание строки. */
$str = "This is a string";

/* Присоединение строки. */
$str = $str . "with some more text";

/* Еще один способ присоединения строки. */
$str .= "and a newline at the end.\n";

/* Строка завершится последовательностью '<p>Number: 9</p>' */
$num = 9;
```

```
$str = "<p>Number: $num</p>";

/* В этом примере (одиночные кавычки) перегрузка переменных не будет
произведена, строка будет содержать '<p>Number: $num</p>' */
$num = 9;
$str = '<p>Number: $num</p>';

/* Получаем первый символ строки  */
$str = 'Это тест.';
$first = $str[0];

/* Получаем последний символ строки. */
$str = 'Это по-прежнему тест.';
$last = $str[strlen($str)-1];
?>
```

Преобразование строк

Когда строка преобразуется в численное значение, то результирующий тип и значение определяются по следующим правилам.

Строка преобразуется в тип double, если в ней содержатся символы '.', 'e', 'E'. Если таких символов нет, то строка преобразуется в тип integer. Значение определяется по начальной части строки. Если строка начинается с допустимых числовых символов, то они будут использованы при определении числового значения. Если допустимые числовые символы не были найдены, то значением будет 0 (нуль). Допустимыми числовыми символами являются знаки + и -, за которыми следует одна или несколько цифр, десятичная точка и знак экспоненты в виде символов 'e' или 'E', за которым следует одна или несколько цифр.

Примеры автоматического приведения типов:

```
$foo = 1 + "10.5";                // $foo - double (11.5)
$foo = 1 + "-1.3e3";              // $foo - double (-1299)
$foo = 1 + "bob-1.3e3";           // $foo - integer (1)
$foo = 1 + "bob3";                // $foo - integer (1)
$foo = 1 + "10 Small Pigs";       // $foo - integer (11)
$foo = 1 + "10 Little Piggies";   // $foo - integer (11)
$foo = "10.0 pigs" + 1;           // $foo - integer (11)
$foo = "10.0 pigs" + 1.0;         // $foo - double (11)
```

Более подробную информацию можно узнать, раскрыв страницу руководства UNIXF strtod(3).

Чтобы на практике проверить любой из приведенных выше примеров, скопируйте его в файл *.phtm и после присваивания поместите строку echo "\$foo==$foo; type is ". gettype ($foo) . «
\n";

Массивы

Одномерный массив

PHP поддерживает скалярные и ассоциативные массивы, между которыми не существует различия. Массив может быть создан при помощи функций ***list()*** и ***array()***, массив также можно задать, указав значения его элементов в явном виде:

```
$a[0] = "abc";
$a[1] = "def";
$b["foo"] = 13.
```

К массиву можно добавить элементы, если массив не существовал до того, как в него были добавлены элементы, то он будет создан.

```
$a[] = "hello"; // $a[2] == "hello"
$a[] = "world"; // $a[3] == "world"
```

Сортировку элементов массива можно произвести при помощи функций ***asort(), arsort(), ksort(), rsort(), sort(), uasort(), usort()*** и ***uksort()***, каждая из них производит сортировку определенного типа. Можно пересчитать количество элементов, содержащихся в массиве, используя функцию ***count()***.

Можно перейти от одного элемента массива к последующему или предыдущему с помощью функций ***next()*** и ***prev()***. Перебрать все элементы массива можно при помощи функции ***each()***.

Многомерный массив

Многомерные массивы создаются с использованием нескольких индексов. Вот ряд примеров создания массивов различной размерности.

```
$a[1]        = $f; # одномерный массив
$a["foo"]    = $f;
$a[1][0]     = $f; # двумерный массив
$a["foo"][2] = $f; # (числовые и ассоциативные индексы можно сочетать)
$a[3]["bar"] = $f;
$a["foo"][4]["bar"][0] = $f;    # четырехмерный массив
```

На элементы массива можно ссылаться непосредственно внутри строк, заключенных в двойные кавычки:

```
$a[3]['bar'] = 'Bob';
echo "This won't work: $a[3][bar]";
```

В результате работы получим строку This won't work: Array[bar] (рис. 6.3).

Чтобы всеже получить сам элемент массива перепишем пример в таком виде:

```
$a[3]['bar'] = 'Bob';
echo "This will work: " . $a[3][bar];
```

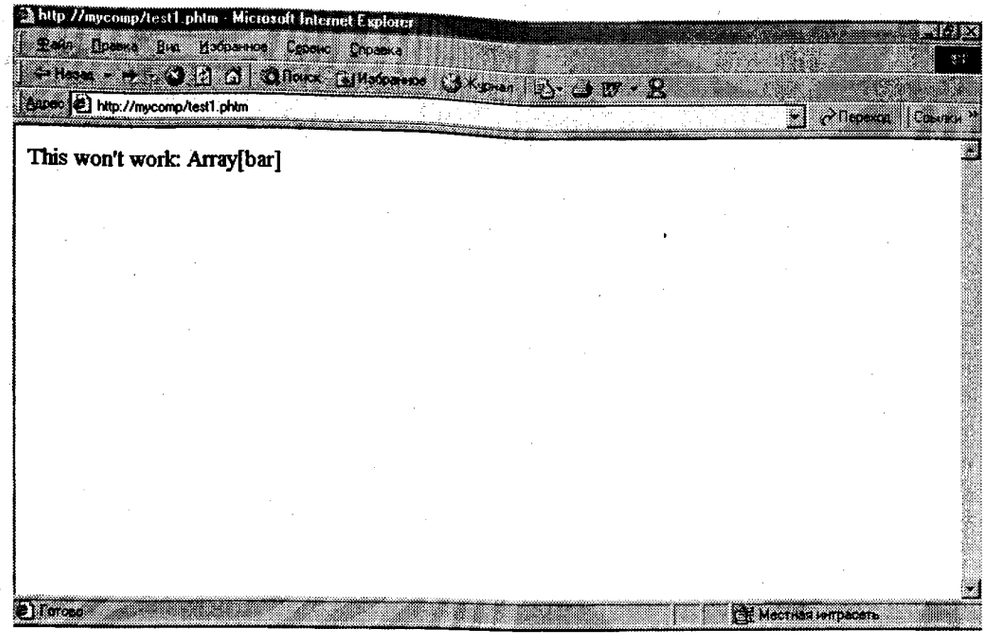


Рис. 6.3. *Вставлена ссылка на массив*

Сейчас все работает как следует (рис. 6.4).

В случае наличия ошибок можно использовать такую конструкцию, в которой индексы помещены в фигурные скобки:

```
$a[3]['bar'] = 'Bob';
echo "This will work: {$a[3][bar]}";
```

Присвоить значения элементам многомерного массива можно различными способами, однако наиболее интересным представляется такой способ, когда используется команда ***array()*** для ассоциативных массивов. Вот примеры.

```
# Пример 1:
$a["color"] = "red";
$a["taste"] = "sweet";
$a["shape"] = "round";
$a["name"]  = "apple";
$a[3]       = 4;
# Пример 2:
$a = array(
      "color" => "red",
      "taste" => "sweet",
```

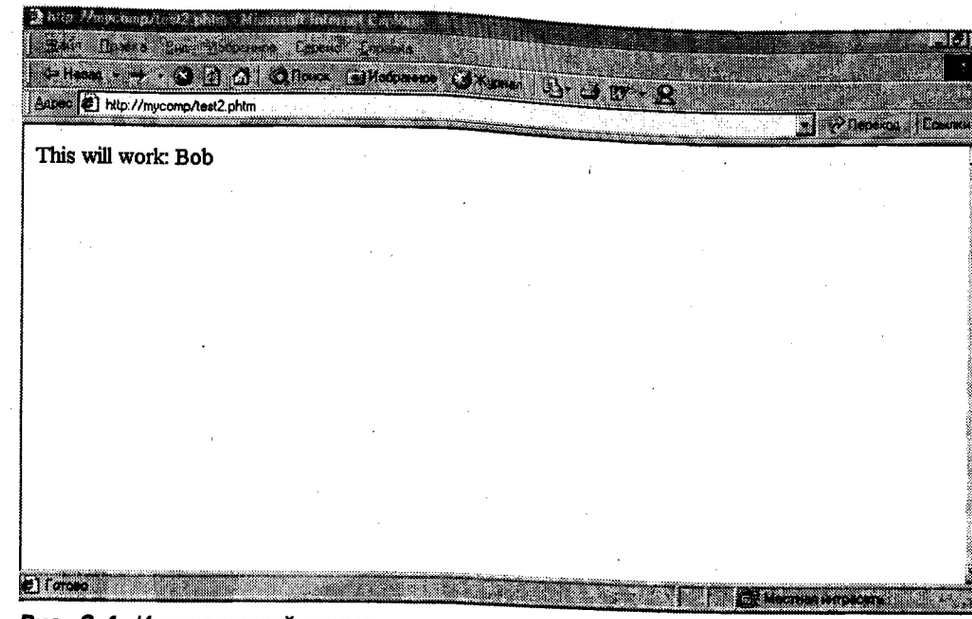



Рис. 6.4. *Исправленный пример*

```
      "shape" => "round",
      "name"  => "apple",
      3       => 4
);
```

Функция ***array()*** может быть использована и с многомерными массивами:

```
<?
$a = array(
      "apple"  => array(
            "color"  => "red",
            "taste"  => "sweet",
            "shape"  => "round"
      ),
      "orange"  => array(
            "color"  => "orange",
            "taste"  => "tart",
            "shape"  => "round2
      ),
      "banana"  => array(
            "color"  => "yellow",
```

```
            "taste"   => "paste-y",
            "shape"   => "banana-shaped"
        )
);

echo $a["apple"]["taste"];      # получим значение "sweet"
?>
```

Объекты

Инициализация объектов

Для инициализации объекта используется слово new и переменной присваивается экземпляр указанного объекта, например:

```
<?php
class foo {
     function do_foo() {
          echo "Doing foo.";
     }
}
$bar = new foo;
$bar->do_foo();
?>
```

Подробное описание см. в разделе *Классы и Объекты*.

Приведение типов

В PHP не поддерживается явное определение типов переменных, тип определяется в зависимости от контекста, в котором появляется переменная. Иными словами, если, например, переменной var присваивается строка, то переменная var становится строковой переменной. Если после этого переменной var будет присвоено целое значение, то переменная будет иметь целый тип.

В качестве примера автоматического преобразования типов в PHP удобно рассмотреть оператор «+». Если хотя бы один из операндов этого оператора имеет тип double, то результат будет иметь тип double. В противном случае операнды будут считаться имеющими тип integer. Выполнение оператора не меняет тип самих операндов. Вот примеры:

```
$foo = "0";  // $foo — string (ASCII 48)
$foo++; // $foo — string "1" (ASCII 49)
$foo += 1; // $foo — integer (2)
$foo = $foo + 1.3;  // $foo — double (3.3)
```



```
$foo = 5 + "10 Little Piggies";     // $foo — integer (15)
$foo = 5 + "10 Small Pigs";         // $foo — integer (15)
```

Если преобразование типов в двух последних строчках выглядит странным, то советуем просмотреть раздел, посвященный преобразованию строк.

Если всеже появляется необходимость преобразовать переменную к определенному типу, то рекомендуем обратиться к разделу Объявление типов. Для задания и изменения типа переменной используется функция *settype()*.

Вышерассмотренные примеры можно дополнить строкой

```
echo "\$foo==$foo; type is " . gettype ($foo) . "<br>\n";
```

Тогда мы сможем проверить автоматическое приведение типов на практике.

Примечание.

Поведение массивов при приведении типов не определено.

Пример.

```
$a = 1;         // $a — integer
$a[0] = "f";    // $a становится массивом,  $a[0] равно "f"
```

Этот пример не представляет никаких трудностей. Однако, рассмотрим другой пример:

```
$a = "1";       // $a is a string
$a[0] = "f";    // What about string offsets? What happens?
```

Сейчас мы сталкиваемся с проблемой, а именно, возникает вопрос о том, должна ли переменная $a содержать массив, в котором первый элемент равен «f», или же «f» будет первым символом в строковой переменной $a?

Именно по причине существования такой проблемы принято считать, что результат приведения типов в таких ситуациях не определен. Решение вопроса находится в стадии проработки.

Объявление типов

Объявление типов в PHP работает, в основном, также, как в языке C: имя типа пишется в скобках перед переменной, тип которой объявляется. Вот пример:

```
$foo = 10;                    // $foo is an integer
$bar = (double) $foo;         // $bar is a double
```

Возможно объявление следующих типов:

- (int), (integer) — целый тип integer;
- (real), (double), (float) — тип double;
- (string) — тип string;
- (array) — тип array;
- (object) — тип object.

Внутри скобок при объявлении типов можно использовать пробелы и знаки табуляции, т.е. обе строки в примере ниже будут эквивалентны:

```
$foo = (int) $bar;
$foo = ( int ) $bar;
```

Не всегда представляется очевидным то, что произойдет при явном приведении различных типов к другим типам. Следует иметь в виду следующее.

При преобразовании скалярных и строковых переменных к массиву, значение переменной становится значением первого элемента этого массива:

```
$var = 'пока';
$arr = (array) $var;
echo $arr[0];  // напишет 'пока'
```

При преобразовании скалярного типа или строки к объекту, значение переменной становится атрибутом объекта, именем такого атрибута является 'scalar':

```
$var = 'ciao';
$obj = (object) $var;
echo $obj->scalar;  // outputs 'ciao'
```

Переменные

Основные понятия

Переменные в языке PHP начинаются со знака доллара, который располагается перед именем переменной. Переменные чувствительны к регистру. Имя переменной должно начинаться с буквы верхнего или нижнего регистра, за которой могут быть расположены буквы, числа, знак подчеркивания.

Примеры:

```
$var = "Bob";
$Var = "Joe";
```



```
echo "$var, $Var";       // напишет "Bob, Joe"

$4site = 'not yet';      // ошибка, первый знак- цифра
$_4site = 'not yet';     // верно, начинается со знака подчеркивания
$тдуte = 'mansikka';     // верно; 'д' - это ASCII 228.
```

Можно использовать русские буквы, например,

```
<?
$a[3]['ящур'] = 'Прыщ';
echo "Это работает: " . $a[3]['ящур'];
?>
```

Результат показан на рис. 6.5.

В PHP переменные всегда имеют значения. Если, например, мы присваиваем переменной выражение, то значение этого выражения будет передано переменной. Если после такого присваивания изменить значения переменных, входящих в выражение, то значение первой переменной (которой присвоено выражение) не будет изменено.

В PHP4 существует другой способ присваивания значений переменным — присваивание по ссылке. Иными словами, новая переменная может служить ссылкой на другую переменную. Изменения, произведенные с новой переменной, сказываются на первой переменной и

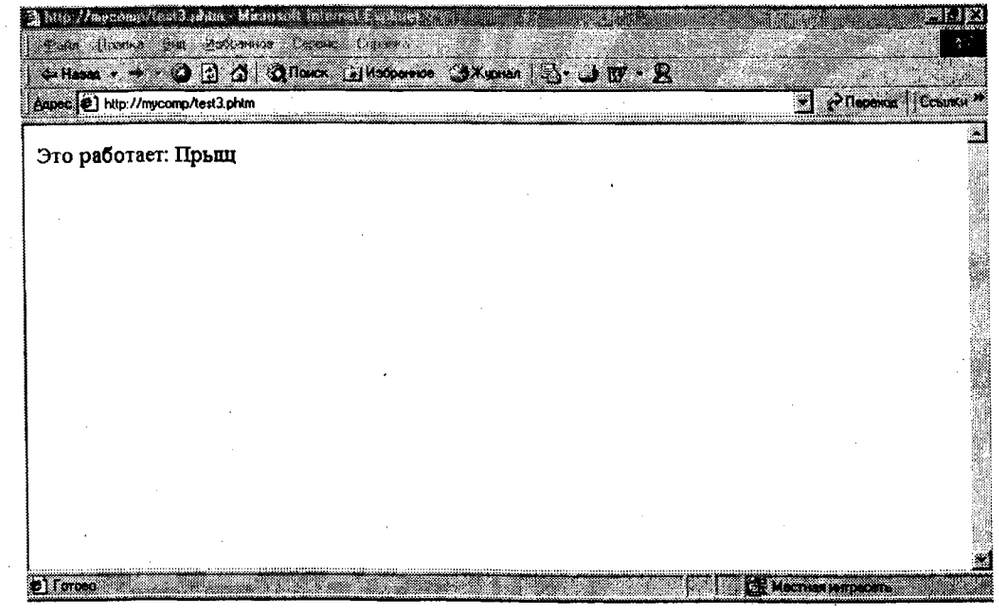


Рис. 6.5. *В качестве идентификаторов были использованы русские слова. Пример корректно работает*

наоборот. Вторая переменная становится псевдонимом первой. При присваивании не происходит копирования значений. Это сказывается на скорости выполнения программ, скорость увеличивается. Однако увеличение скорости может быть заметно лишь при больших массивах или большом количестве присваиваний.

Чтобы создать ссылку достаточно поставить знак & перед переменной, для которой назначается ссылка.

Пример:

```
<?php
$foo = 'Bob';    // Присвоим значение 'Bob' переменной $foo
$bar = &$foo;    // Ссылка на $foo посредством $bar.
$bar = "My name is $bar";  // Изменяем  $bar...
echo $foo;                 // $foo тоже изменилось
echo $bar;
?>
```

Только имена переменных можно использовать, ссылаясь на них. Ссылки на постоянные и литералы недопустимы.

```
<?php
$foo = 25;
$bar = &$foo;       // Создаем ссылку.
$bar = &(24 * 7);   // Ошибка
function test() {
    return 25;
}
$bar = &test();     // Ошибка.
?>
```

Заранее определенные переменные

В PHP существует ряд заранее определенных переменных, которые могут быть использованы во всех скриптах. Многие из таких переменных зависят от того, на каком сервере будет запущен скрипт и от других факторов. Не все переменные доступны в режиме вызова PHP из командной строки. Однако многие переменные доступны во всех конфигурациях.

Чтобы получить список всех заранее определенных переменных полезно обратиться к функции ***phpinfo()***. Эта функция не выводит исчерпывающий список переменных, но она может дать список таких переменных, к которым мы определенно сможем иметь доступ посредством скриптов.

Переменные сервера Apache

Эти переменные создаются при работе сервера Apache. Если используется другой сервер, то, вероятно, не все из перечисленных переменных будут доступны. При этом могут использоваться другие переменные, которые не упомянуты в этом разделе. Большое количество переменных описано в спецификации CGI 1.1, поэтому такие переменные скорее всего будут присутствовать и на других серверах.

GATEWAY_INTERFACE
Какой вариант спецификации CGI используется на сервере, значение 'CGI/1.1'.

SERVER_NAME
Имя сервера, выполняющего текущий скрипт.

SERVER_SOFTWARE
Строка, идентифицирующая сервер.

SERVER_PROTOCOL
Имя используемого протокола, например, 'HTTP/1.0';

REQUEST_METHOD
Какой использовался метод в процессе доступа к странице: 'GET', 'HEAD', 'POST', 'PUT'.

QUERY_STRING
Строка запроса, посредством которой открыта страница.

DOCUMENT_ROOT
Основной каталог, в котором находится документ, в котором исполняется данный скрипт (по тому, как сконфигурирован сервер).

HTTP_ACCEPT
Заголовок текущего запроса.

HTTP_ACCEPT_CHARSET
Заголовок запроса, содержащий кодировку, например, 'iso-8859-1,*,utf-8'.

HTTP_ENCODING
Заголовок, содержащий тип сжатия, например, 'gzip'.

HTTP_ACCEPT_LANGUAGE
Заголовок языка, например, 'en'.

HTTP_CONNECTION
Заголовок соединения Connection, например, 'Keep-Alive'.

HTTP_HOST
Содержимое заголовка Host.

HTTP_REFERER
Адрес документа, сославшегося на текущий документ.

HTTP_USER_AGENT
Тип клиентского приложения, например, Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586). Эта переменная может быть получена с помощью функции ***get_browser()***.

REMOTE_ADDR
Адрес IP, с которого произошло обращение на текущую страницу.

REMOTE_PORT
Порт, с которого было произведено обращение к данной странице.

SCRIPT_FILENAME
Абсолютный путь к выполняемому в данное время скрипту.

SERVER_ADMIN
Значение директивы SERVER_ADMIN.

SERVER_PORT
Порт, на котором работает сервер. По умолчанию используется порт 80.

SERVER_SIGNATURE
Подпись сервера, содержащая версию сервера и имя хоста.

PATH_TRANSLATED
Файловая система по отношению к положению текущего скрипта после преобразования виртуальных каталогов в реальные.

SCRIPT_NAME
Содержит путь к текущему скрипту.

REQUEST_URI
Идентификатор URI, использованный для доступа к текущему скрипту, например, '/index.html'.

Переменные окружения

Эти переменные заимствуются из глобального окружения, в котором функционирует PHP и web-сервер.

Переменные PHP

Эти переменные создаются PHP.

argv
Массив аргументов, переданных скрипту.

argc
Количество параметров в командной строке, переданных скрипту.

PHP_SELF
Имя файла, содержащего текущий скрипт.

HTTP_COOKIE_VARS
Ассоциативный массив переменных, переданных скрипту посредством HTTP cookies. Работает при включенной директиве track_vars или <?php_track_vars?> при активизации внутри скрипта.

HTTP_GET_VARS
Ассоциативный массив переменных, переданных скрипту с помощью метода get. Чтобы переменная была доступна, необходимо включить режим отслеживания переменных при помощи директивы track_vars или с помощью <?php_track_vars?>.

HTTP_POST_VARS
Ассоциативный массив переменных, переданных скрипту при помощи метода POST. Работает при включенной директиве track_vars или <?php_track_vars?>.

Область видимости переменных

Область видимости переменных — это контекст программы, внутри которого определена и может быть использована переменная. Как правило, в PHP все переменные имеют одну и ту же область видимости, которая распространяется на все включенные в скрипт файлы. Например:

```
$a = 1;
include "b.inc";
```

Здесь переменная $a будет доступна для всего скрипта, расположенного в файле b.inc. Однако, важно иметь ввиду, что внутри функций, определенных пользователем, создаются локальные переменные, область видимости которых ограничена такой функцией. Например:

```
$a = 1; /* global scope */
Function Test () {
    echo $a; /* reference to local scope variable */
}
Test ();
```

Этот скрипт не выведет ничего, так как выражение echo использует локальную переменную, а внешняя переменная $a не передается функции в виде фактического параметра (функция вообще не содержит параметров). В этом PHP отличается от C, где глобальные переменные оказываются доступными для пользовательских функций. Для того чтобы глобальная переменная была доступна внутри функции, необходимо внутри функции объявить ее глобальной, например:

```
$a = 1;
$b = 2;
Function Sum () {
     global $a, $b;
     $b = $a + $b;
}
Sum ();
echo $b;
```

Такой скрипт выведет «3». Здесь мы видди, что переменные $a и $b объявлены глобальными внутри функции ***Sum()***. Количество используемых внутри функции глобальных переменных неограничено.

Другой метод осуществить доступ к глобальным переменным — использование заранее определенного массива $GLOBALS. С помощью этого массива предыдущий пример может быть переписан в виде:

```
$a = 1;
$b = 2;
Function Sum () {
     $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}
Sum ();
echo $b;
```

Массив $GLOBALS представляет собой ассоциативный массив имен глобальных переменных, которые используются в качестве ключей-индексов этого массива, значениями элементов массива являются значения глобальных переменных.

Кроме глобальных переменных могут быть использованы статические переменные. Если внутри функции используется обычная переменная, то при повторном возвращении к функции такая переменная может быть потеряна, как, например, это произойдет для следующего фрагмента кода:

```
Function Test () {
     $a = 0;
     echo $a;
     $a++;
}
```


Создавать такую функцию довольно бессмысленно, так как при каждом обращении к ней переменная $a будет устанавливаться равной 0 и увеличение ее на единицу каждый раз будет приводить к значению 1. При этом эта переменная не будет сохранять изменения, произведенные в предыдущем обращении к функции. Чтобы изменить ситуацию, необходимо определить переменную. Чтобы изменить ситуацию, необходимо определить переменную $a как static. Перепишем пример в таком виде:

```
Function Test () {
     static $a = 0;
     echo $a;
     $a++;
}
```

Сейчас переменная $a при повторном обращении к функции ***Test()*** сохраняет значение, присвоенное ей при предыдущем обращении к функции ***Test()***. Повторного присваивания значения 0 не будет произведено.

Статические переменные могут быть использованы в целях определения рекурсивных функций, т.е. таких функций, которые внутри себя производят обращение к самим себе. При создании рекурсивных функций необходимо соблюдать осторожность, так как в такой ситуации легко ошибиться и создать бесконечный цикл. В следующем примере используется рекурсивная функция, число итераций (повторов) которой ограничено числом 10, в которой использована статическая переменная $count.

```
<?
 Function Test () {
     static $count = 0;

     $count++;
     echo $count; echo "<P>";
     if ($count < 10) {
          Test ();
     }
     $count--;
}
 Test();
 ?>
```

Результат работы программы показан на рис 6.6.

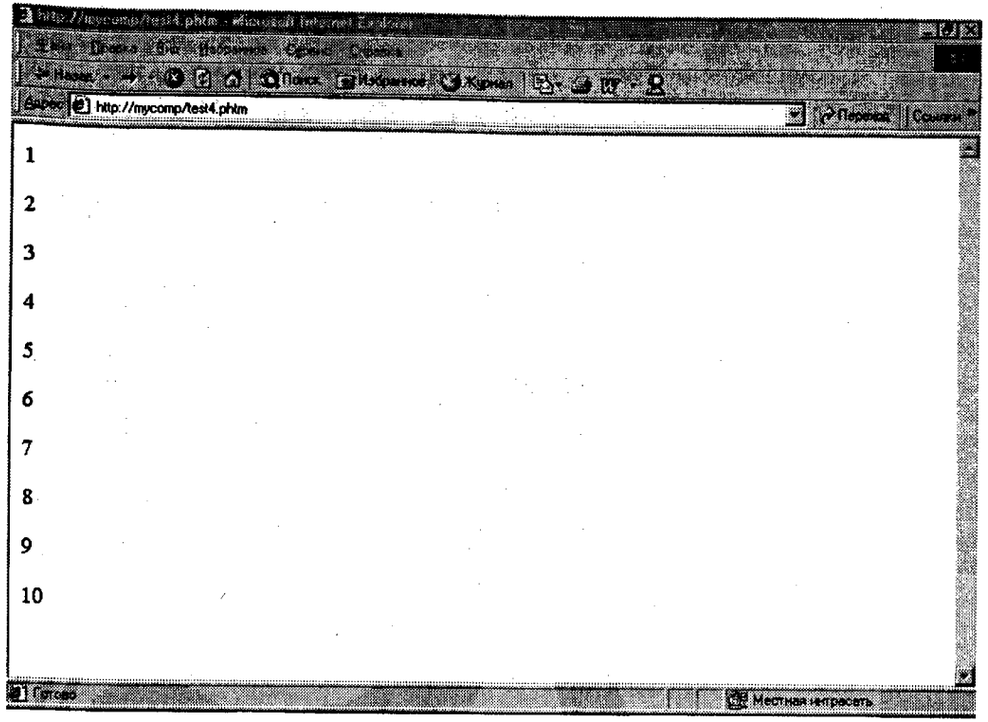


Рис. 6.6. *Результат выполнения рекурсивной функции*

Переменные переменных (имена переменных)

Иногда оказывается полезным использование имен переменных, или переменных для переменных. Переменные переменных могут быть динамически (программными средствами) изменены. Обычные имена могут быть созданы с использованием такого рода инструкций:

```
$a = "hello";
```

Строка 'hello' может использоваться как имя переменной, если мы запишем такое выражение:

```
$$a = "world";
```


Сейчас имена образуют некую древовидную иерархию. Переменная $a содержит 'hello', а переменная $hello содержит 'world'. Приведем пример, который выведет текст HELLO WORLD:

```
echo "$a ${$a}";
```

Эта строка выведет в точности то же самое, что и строка

```
echo "$a $hello";
```

См. рис. 6.7.

При использовании имен имени при работе с массивами важно правильно установить порядок имен, к которым относятся индексы. Для этого удобно использовать скобки. Если мы напишем $$a[1], то анализатору будет необходимо знать, к чему относится индекс, к переменной $a или к переменной $$a. Для того, чтобы не возникало такого вопроса, используются скобки: в первом случае мы напишем ${$a[1]}, а во втором случае — ${$a}[1].

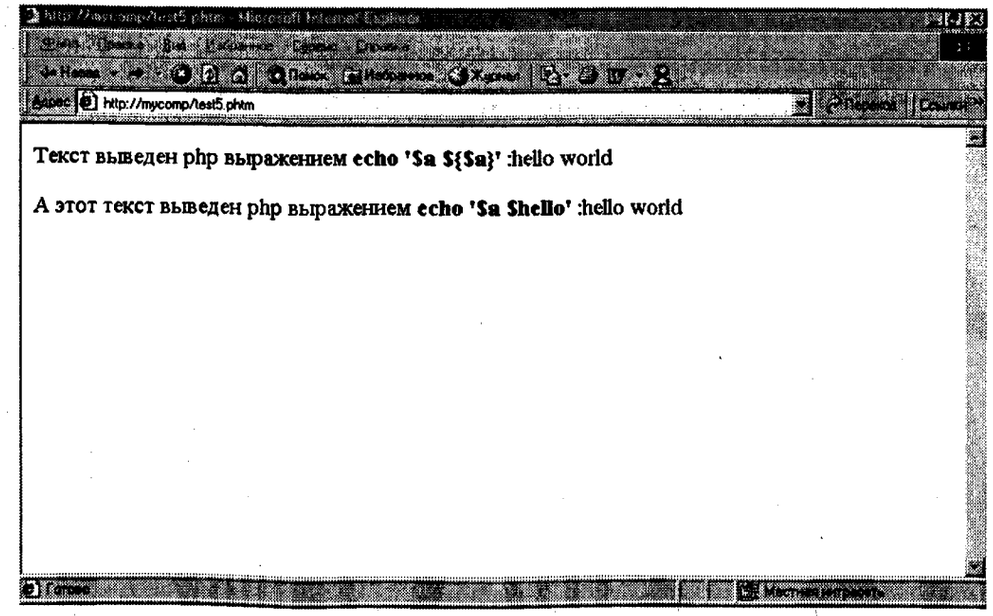


Рис. 6.7. *Разные способы использования имен имени*

Внешние переменные

Формы HTML

Внешние переменные могут быть получены скриптом посредством получения форм. Когда скрипт получает форму, то переменные, содержащиеся в форме, становятся доступны этому скрипту. Рассмотрим пример.

Пример с переменной из формы:

```
<form action="script.phtm" method="post">
     Имя: <input type="text" name="name"><br>
     <input type="submit">
</form>
```

После того, как форма будет отправлена серверу, PHP создаст переменную $name, в которой будет размещено значение, введенное в текстовое поле формы. PHP также имеет возможность распознавать массивы, переданные с помощью форм, однако такие массивы должны быть одномерными.

Более сложный пример:

```
<form action="array.php" method="post">
     Имя: <input type="text" name="personal[name]"><br>
     Email: <input type="text" name="personal[email]"><br>
     Пиво: <br>
     <select multiple name="beer[]">
          <option value="Холстен">Холстен
          <option value="Будвайзер">Будвайзер
          <option value="Балтика">Балтика
          </select>
     <input type="submit">
</form>
```

Если включена директива track_vars (можно использовать <?php_track_vars?>), то полученные переменные независимо от метода POST или GET, могут быть доступны через ассоциативные глобальные массивы $HTTP_POST_VARS и $HTTP_GET_VARS в зависимости от метода получения переменных.

Переменные элемента отправки формы типа IMAGE

При отправке форму можно использовать ярлык <input type=«image»>:

```
<input type=image src="image.gif" name="sub">
```


Если сейчас щелкнуть в любом месте рисунка формы, то эта форма будет отправлена вместе с переменными sub_x и sub_y. Эти переменные содержат координаты положения указателя мыши во время щелчка. В момент получения таких переменных происходит изменение имен, так как броузер посылает переменные, в которых вместо значка подчеркивания используется точка. PHP преобразует точки в знаки подчеркивания.

HTTP Cookies

PHP поддерживает механизм передачи cookies удобным образом. Можно использовать функцию ***SetCookie()*** для задания cookies. Все полученные от броузера в http-заголовке cookies автоматически преобразуются в соответствующие переменные наподобие того, как это происходит при передаче форм с помощью методов get и post.

Если возникает необходимость передать при помощи одного экземпляра cookie нескольких значений, то можно использовать квадратные скобки после имени cookie. Например:

```
SetCookie ("MyCookie[]", "Testing", time()+3600);
```

Определенные таким образом cookies заменят старые, если у них совпадут имена. Чтобы такого не происходило, можно воспользоваться счетчиком.

Пример SetCookie

```
$Count++;
SetCookie ("Count", $Count, time()+3600);
SetCookie ("Cart[$Count]", $item, time()+3600);
```

Переменные окружения

PHP автоматически создает переменные окружения, которые доступны так же, как и обычные переменные.

```
echo $HOME;  /* Показать переменную окружения HOME. */
```

Точки в именах внешних переменных

Как правило, PHP не изменяет имена внешних переменных. Однако, существует одно исключение. Если внешняя переменная содержит точку (а точка не может использоваться в именах PHP), то могут произойти неприятности.

```
$varname.ext;  /* недопустимый символ в имени переменной */
```

При этом анализатор воспримет конструкцию как переменную $varneame, объединенную оператором объединения строк с конструкцией ext. Очевидно, что это не приведет к желаемым последствиям.

Поэтому PHP заменяет точки в получаемых именах переменных знаками подчеркивания.

Определение типов переменных

Поскольку в PHP типы переменных определяются автоматически и зависят от контекста, в котором появляются переменные, то не всегда бывает очевидным то, к какому типу относится та или иная переменная. Для определения типа переменных существует набор функций, среди которых назовем следующие: ***gettype(), is_long(), is_double(), is_string(), is_array(), is_object()***.

Константы

В языке PHP существует механизм создания констант, которые будут существовать наряду с константами, которые всегда существуют при работе PHP. Во многом константы походят на переменные, за тем исключением, что константы не могут быть изменены после того, как они будут определены. Константы определяются при помощи функции ***define()***.

Приведем список заранее определенных констант, существующих всегда, если работает PHP.

__FILE__

Значением этой константы является имя текущего phtm-файла. Если эта константа используется в файле, на который был осуществлен запрос или который был вставлен в родительский файл, то константа равна имени вставленного (но не родительского) файла.

__LINE__

Номер текущей строки в том порядке, как эта строка появляется в текущем файле.

PHP_VERSION

Строка, указывающая номер версии PHP.

PHP_OS

Имя операционной системы, на которой работает PHP.

Пример

```
<?
echo PHP_VERSION."<P>";
echo PHP_OS;
?>
```

На экране мы увидим запрошенную информацию (рис. 6.8).

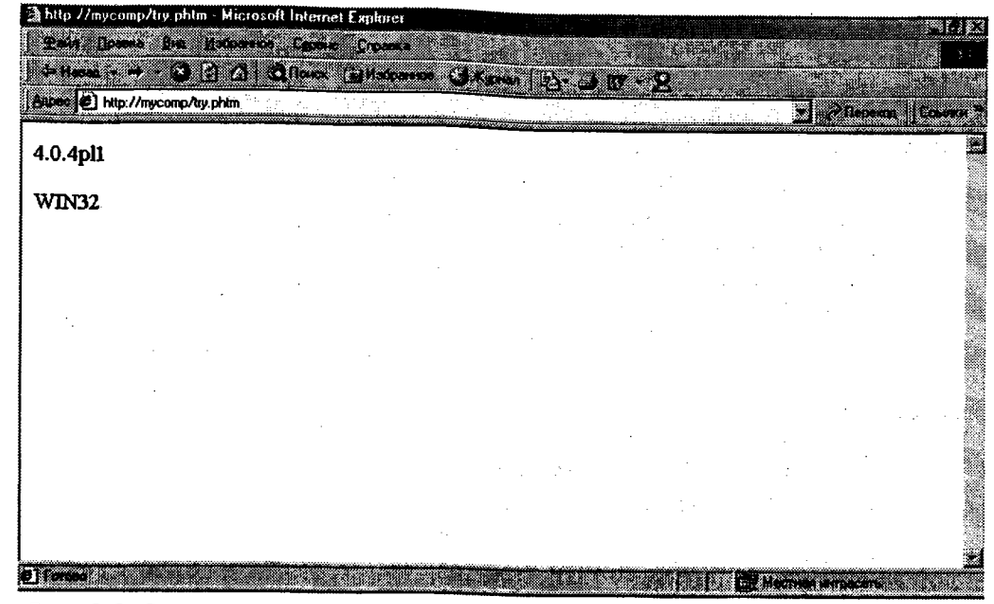


Рис. 6.8. *Версия языка и операционная система*

TRUE

Значение true.

FALSE

Значение false.

E_ERROR

Ошибка, не являющаяся ошибкой, обнаруженной при проверке синтаксиса, но являющаяся фатальной.

E_WARNING

Обозначает ситуацию, когда PHP обнаружил некоторые ошибки, однако, возможно продолжение работы скрипта.

E_PARSE

Анализатор обнаружил фатальную ошибку синтаксиса. Продолжение работы невозможно.

E_NOTICE

Замечание о том, что возникла ситуация, связанная (или не связанная) с возникновением ошибки (не фатальной).

E_ALL

Все перечисленные выше константы E_* объединяются этим именем.

Пример задания констант:

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // выдает "Hello world."
?>
```

Пример использования __FILE__ и __LINE__

```
<?php
function report_error($file, $line, $message) {
     echo "An error occured in $file on line $line: $message.";
}
report_error(__FILE__,__LINE__, "Something went wrong!");
?>
```

Инструкции

Инструкции являются одними из наиболее важных составных блоков программы на PHP. Практически все, что бы ни было записано на PHP, является инструкцией. Простейший, хотя и не самый строгий вариант определения инструкции таков. Инструкцией является все, что имеет значение.

Одними из наиболее основных примеров инструкции являются константы и переменные. Например когда мы записываем выражение «$a=5», то мы присваиваем '5' переменной $a. '5' имеет значение 5. После присваивания $a будет иметь значение 5.

Несколько более сложным примером инструкций (или выражений) является пример с функциями. Например:

```
Function foo () {
     return 5;
}
```

Если вам приходилось программировать раньше, то вы уже знакомы с функциями, поэтому вы понимаете, что для определенной выше функции написать выражение $c = foo() — это почти то же самое, если написать $c = 5, и это действиетально так. Функции представляют собой выражения, значением которых является возвращаемое функцией значение. Поскольку функция *foo()* возвращает 5, то значение выражения 'foo()' будет целое число 5. Как правило, функции возвращают не какое-то фиксированное значение, а для определения возвращаемого значения производят определенные вычисления.

Понятно, что значениями в языке PHP могут быть не только целые числа, гораздо более часто встречаются другие типы. PHP поддерживает три скалярных типа: целые числа, числа с плавающей точкой и строки. Скалярными величинами называются такие величины, которое

не могут быть разделены на более мелкие составные части в отличие, например, от массивов. В PHP также существует два составных типа. Это массивы и объекты. Каждый такой тип может быть присвоен переменной, а также любой из них может быть возвращен в виде значения функции.

В качестве примера выражений полезно рассмотреть условные выражения. Значением условных выражений всегда бывает либо 0, либо 1, это обозначает false и true соответственно. Среди операторов, которые используются в инструкциях сравнения, используются такие операторы, как > (больше), >= (больше или равно), == (равно), != (не равно), < (меньше) и <= (меньше или равно). Такие выражения часто используются внутри условных операторов, в частности при использовании оператора if.

Еще один пример выражения, состоящего из тернарного оператора.

```
$first ? $second : $third
```

Значением такого выражения является значение выражения $second, но только в том случае, если значением выражения $first будет true (или отличное от нуля), если же значением первого выражения будет 0 (или false), то все выражение в целом примет значение, равное значению выражения $third.

В качестве упражнения приведем пример с использованием различных выражений.

```
Function double($i) {
     return $i*2;
}
$b = $a = 5;          /* присваивает значение 5 двум переменным $a и $b */
$c = $a++;            /* постфиксное увеличение и присваивание, присваивает исходное значение переменной $a (5) переменной $c */
$e = $d = ++$b;       /* префиксное увеличение и присваивание, присваивает увеличенное значение переменной $b (6) переменным $d и $e */

/* с этого момента обе переменные $d и $e равны 6 */

$f = double($d++);    /* присваивает двойное значение переменной $d перед тем, как $d будет увеличено на единицу, т.е. 2*6 = 12 присваивается переменной $f */
$g = double(++$e);    /* присваивает двойное значение увеличенной на единицу переменной $e переменной $g, т.е. 2*7 = 14 присваивается $g */
```

```
$h = $g += 10;      /* вначале переменная $g увеличивается на 10,
                    принимая значение 24, а затем значение 24 присва-
                    ивается переменной $h, , т.е. и $h, и $g принимают
                    значение 24. */
```

В начале этого раздела мы предположили, что практически все описываемые нами типы инструкций являются выражениями. Однако, строго говоря, не каждое выражение является инструкцией. Если выражение состоит из нескольких элементов, разделенных одним или несколькими значками точки с запятой, то такое выражение не может считаться инструкцией, однако, составные части такого выражения по-прежнему являются инструкциями. Выражение в целом не является инструкцией, поскольку оно не имеет одного определенного значения. Каждая отдельная инструкция, входящая в такое выражение, имеет значение, и для всего выражения в целом таких значений может быть несколько.

7

УПРАВЛЕНИЕ ХОДОМ ВЫПОЛНЕНИЯ ПРОГРАММЫ

- **Операторы**
  - *Арифметические операторы*
  - *Операторы присваивания*
  - *Побитовые операторы*
  - *Операторы сравнения*
  - *Операторы управления ошибками*
  - *Операторы выполнения системных команд*
  - *Операторы увеличения и уменьшения на единицу*
  - *Логические операторы*
  - *Иерархия операторов*
  - *Операторы для работы со строками*
- **Управление последовательностью выполнения инструкций**
- **Функции**
  - *Функции, определяемые пользователем*
  - *Аргументы функции*
  - *Возвращаемые значения*
  - *Ключевое слово old_function*
  - *Переменные функции*
- **Классы и объекты**
- **Ссылки**
  - *Что такое ссылки*
  - *Для чего используются ссылки*
  - *Чем не являются ссылки*
  - *Возвращение значений при помощи ссылок*

Операторы

Арифметические операторы

Арифметические операторы (табл. 7.1) используются для осуществления обычных арифметических действий, которым нас обучали в школе.

Оператор деления(«/») возвращает целую величину (результат целочисленного деления) в том члучае, когда оба оператора — целые (или строка, преобразованная в целое). Если каждый операнд является величиной с плавающей запятой, то выполнится деление с плавающей запятой.

Таблица 7.1
Арифметические операторы

Пример оператора	Название	Результат
$a + $b	Сложение	Сумма $a и $b.
$a - $b	Вычитание	Вычитает $b из $a.
$a * $b	Умножение	Произведение $a и $b.
$a / $b	Деление	Деление $a на $b.
$a % $b	Модуль	Остаток от деления $a на $b.

Операторы присваивания

Основным оператором присваивания является оператор обычного присваивания «=». Это не знак равенства. Этот оператор присваивает значение, расположенное справа от него левому операнду.

Значением выражения присваивания является присваиваемая величина. Так, выражение «$a = 3» имеет значение 3. Этот факт позволяет использовать довольно тонкие конструкции, например:

```
$a = ($b = 4) + 5; // теперь $a равно 9, а $b стало равным 4.
```

Существуют и другие операторы присваивания, они включют в себя комбинации оператора присваивания с арифметическими операторами. Например:

```
$a = 3; $a += 5; // теперь $a равно 8, как если бы мы написали: $a
= $a + 5;
$b = "Hello ";
$b .= "There!"; // теперь $b равно "Hello There!", как если бы мы
написали $b = $b . "There!";
```


Побитовые операторы

Побитовые операторы позволяют работать с определенными отдельными битами (табл. 7.2).

Таблица 7.2
Побитовые операторы

Пример оператора	Название	Результат
$a & $b	И	Результат равен единице, если оба бита операндов равны единице
$a | $b	ИЛИ	Бит равен единице, если хотя бы в одном из операндов соответствующий бит равен единице
$a ^ $b	Исключающее ИЛИ (XOR)	Результат равен единице в том случае, если только один операнд содержит единицу в данной битовой позиции
~ $a	Отрицание	Битовая позиция меняет значение на противоположное
$a << $b	Сдвиг влево	Сдвигает битовое представление $a на $b влево
$a >> $b	Сдвиг вправо	То же, что и выше, но сдвиг вправо (деление на два)

Операторы сравнения

Операторы сравнения, как и подобает в соответствии с их названием, позволяют сравнивать величины (табл. 7.3).

Таблица 7.3
Операторы сравнения

Пример оператора	Название	Результат
$a == $b	Равно	true, если $a равно $b
$a===$b	Идентично	True, если $a равно $b и оба операнда одного типа
$a != $b	Не равно	True, если $a не равно $b
$a !== $b	Не идентично	True, если или $a не равно $b, или операнды различного типа
$a < $b	Меньше чем	True, если $a меньше чем $b
$a > $b	Больше чем	True, если $a больше $b
$a <= $b	Меньше или равно	True, если $a меньше или равно $b
$a >= $b	Больше или равно	True, если $a больше или равно $b

Операторы обработки ошибок

В PHP определен оператор управления ошибками. Это оператор @. Если этот оператор вставлен перед выражением, то все сообщения об ошибках, генерируемые этим выражением, будут проигнорированы. Если включена опция track_errors, то все сообщения об ошибках будут сохранены в глобальной переменной $php_errormsg. При возникновении очередной ошибки значение этой переменной переписывается, поэтому при необходимости следует своевременно воспользоваться текущим значением этой переменной.

```
<?php
/* Умышленная ошибка */
$res = @mysql_query ("select name, code from 'namelist") or
     die ("Query failed: error was '$php_errormsg'");
?>
```

См. также ***error_reporting().***

Предупреждение.

Оператор @ отключает вывод сообщений о критических ошибках, приводящих к прекращению выполнения скрипта.

Оператор выполнения системных команд

В PHP существует один оператор выполнения системных команд — оператор обратных кавычек. Это не одиночные, а обратные кавычки. PHP предпримет попытку выполнить системную команду, размещенную между обратными кавычками. Результат работы системной команды можно сохранить в виде значения переменной. Например:

```
$output = 'ls -al';
echo "<pre>$output</pre>";
```

См. также system(), passthru(), exec(), popen(), escapeshellcmd().

Операторы увеличения и уменьшения

В PHP поддерживаются операторы увеличения и уменьшения на единицу, подобные тем, что используются в языке C (табл. 7.4).

Таблица 7.4
Операторы увеличения и уменьшения на единицу

Пример оператора	Название	Результат
++$a	*Префиксный оператор увеличения на единицу*	*Увеличивает значение $a на единицу и возвращает $a ПОСЛЕ ПРИСВАИВАНИЯ*
$a++	*Постфиксный оператор присваивания*	*Возвращает $a после чего увеличивает значение этой переменной на единицу*
--$a	*Префиксный оператор уменьшения на единицу*	*Уменьшает переменную $a на единицу и возвращает значение после присваивания*
$a--	*Постфиксный оператор уменьшения на единицу*	*Возвращает $a, а затем присваивает этой переменной значение, меньшее на единицу*

Простой скрипт с примером:

```
<?php
echo "<h3>Постфиксный увеличения</h3>";
$a = 5;
echo "Будет 5: " . $a++ . "<br>\n";
echo "Будет 6: " . $a . "<br>\n";

echo "<h3>Префиксный увеличения</h3>";
$a = 5;
echo "Будет 6: " . ++$a . "<br>\n";
echo "Будет be 6: " . $a . "<br>\n";

echo "<h3>Постфиксный уменьшения</h3>";
$a = 5;
echo "Будет 5: " . $a– . "<br>\n";
echo "Будет 4: " . $a . "<br>\n";

echo "<h3>Префиксный уменьшения</h3>";
$a = 5;
echo "Будет 4: " . –$a . "<br>\n";
echo "Будет 4: " . $a . "<br>\n";
?>
```

На экране мы увидим то, что изображено на рис. 7.1.

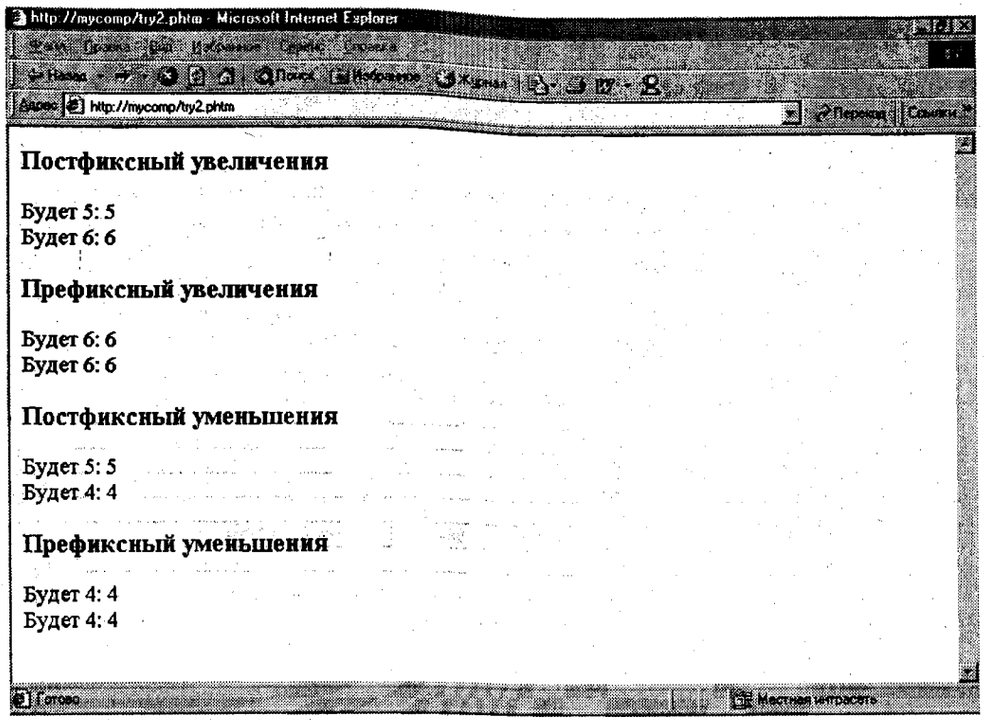


Рис. 7.1. *Операторы уменьшения и увеличения с присваиванием*

Логические операторы

Логические операторы соответствуют обычным классическим логическим операциям (табл. 7.5).

Таблица 7.5
Логические операторы

Пример оператора	Название	Результат
$a and $b	И	Истина, если истинны $a и $b
$a or $b	ИЛИ	Истина, если истинны $a или $b
$a xor $b	ИЛИ	Истина, если истинны $a или $b, но не оба
! $a	НЕ	Истина, если $a ложно
$a && $b	И	Истина, если истинны и $a и $b
$a || $b	ИЛИ	Истина, если истинны $a или $b

Разница в двух операторах «and» и «or» состоит в разнице их приоритетов (см. ниже).

Иерархия операторов

Между операторами устанавливается старшинство, т.е. порядок, в котором эти операторы будут выполняться. Приведенная ниже табл. 7.6 показывает старшинство операторов

Таблица 7.6
Старшинство операторов

Символы	Значение
Левый	,
Левый	Or
Левый	Xor
Левый	And
Правый	Print
Левый	= += -= *= /= .= %= &= |= ^= ~= <<= >>=
Левый	? :
Левый	||
Левый	&&
Левый	|
Левый	^
Левый	&
Не установлено	== != === !==
Не установлено	< <= > >=
Левый	<< >>
Левый	+ - .
Левый	* / %
Правый	! ~ ++ -- (int) (double) (string) (array) (object) @
Правый	[
Не установлено	New</TBODY >

Операторы для работы со строками

Для работы со строками существует два оператора. Основным оператором является оператор объединения строк (.) — точка. Этот оператор возвращает строку, состоящую из строки, хранящейся в левом операнде, объединенную со строкой, хранящейся в правом операнде. Второй оператор — это оператор объединения строк с присваиванием ('.='), который присваивает левому операнду исходное значение левого операнда, объединенное в общую строку со значением правого операнда.

```
$a = "Hello ";
$b = $a . "World!"; // $b содержит "Hello World!"

$a = "Hello";
$a .= "World!";     // сейчас $a содержит "Hello World!"
```

Управление последовательностью выполнения инструкций

В этом разделе мы рассмотрим следующие операторы и функции, используемые для задания последовательности выполнения инструкций PHP.

- if.
- else.
- elseif

Альтернативный синтаксис условных операторов

- while.do..while.for.
- foreach.
- break.
- continue.
- switch.

Специальные конструкции

- require().
- include().
- require_once().
- include_once().

Все программы на языке PHP состоят из блоков выражений, из блоков инструкций. Выражения могут быть выражениями, содержащими присваивания, вызовы функций, циклы, условные выражения и пустые выражения. Каждое выражение, как правило, заканчивается знаком точки с запятой. Набор выражений может быть сгруппирован в отдельный самостоятельный блок. Для этого выражения, образующие такой блок, помещаются между открывающей и закрывающей фигурными скобками. Такой блок выражений представляет собой самостоятельное выражение. В данной главе мы рассматриваем различные типы такого рода выражений.

if

Конструкция, содержащая IF, предоставляет одну из наиболее ценных возможностей многих языков, включая язык PHP. Она позволяет организовать выполнение фрагментов кода с проверкой выполнения того или иного условия. Возможности PHP по использованию выражения IF похожи на возможности, предоставляемые языком C. Синтаксис таков:

```
if (expr) statement
```


Вначале вычисляется значение «expr». Если expr равно TRUE, то PHP выполнит «statement», а если FALSE, то выражение statement не будет выполнено.

Следующий пример выведет фразу 'a is bigger than b', если $a больше $b:

```
if ($a > $b)
    print "a is bigger than b";
```

Часто требуется исполнить больше чем одно выражение с проверкой условия. В этом случае нет необходимости включать каждое выражение в конструкцию IF. Вместо этого можно сгруппировать несколько выражений в блок выражений с использованием фигурных скобок. Следующий код не только выведет фразу, но и присвоит значение $a переменной $b:

```
if ($a > $b) { print "a is bigger than b"; $b = $a; }
```

Выражения IF могут быть вложены друг в друга. Уровень вложенности не ограничивается.

else

Иногда возникает необходимость проверить условие и в случае его выполнения выполнить заданное выражение, а в случае невыполнения условия, выполнить другое выражение. Для организации такого поведения используется конструкция ELSE. ELSE расширяет возможности IF в части возможностей обработки выражений.

Пример, приведенный ниже, выведет фразу 'a is bigger than b' если $a больше $b, и 'a is NOT bigger than b', в противном случае:

```
if ($a > $b) {
    print "a is bigger than b";
} else {
    print "a is NOT bigger than b";
}
```

Выражение, расположенное после ELSE, выполняется только в том случае, если выражение, следующее за IF, равно FALSE, а если есть конструкции ELSEIF, то если и они также равны FALSE (см. ниже).

elseif

Конструкция ELSEIF, как и следует из ее названия, является комбинацией IF и ELSE. ELSEIF, как и ELSE, позволяет выполнить выражение, если значение IF равно FALSE, но в отличие от ELSE оно выполнится только тогда, если выражение ELSEIF равно TRUE. На-

пример, следующий код выведет 'a is bigger than b' если$a>$b, 'a is equal to b' если $a==$b, и 'a is smaller than b' если$a<$b:

```
if ($a > $b) {
    print "a is bigger than b";
} elseif ($a == $b) {
    print "a is equal to b";
} else {
    print "a is smaller than b";
}
```

Внутри одного выражения IF может быть несколько ELSEIF. Первое выражение ELSEIF (если таковое есть), равное TRUE, будет выполнено. В PHP можно написать 'else if' (два слова), что будет значить то же самое, что и 'elseif' (одно слово). Выражение ELSEIF будет выполнено только если выражение IF и все предыдущие ELSEIF, равны FALSE, а данный ELSEIF равен TRUE.

Альтернативный синтаксис операторов

PHP предлагает и иной путь для группирования операторов, содержащих IF, а именно для операторов if, while, for, switch. В таких конструкциях вместо фигурных скобок используются другие обозначения. Вместо открывающей скобки используется двоеточие (:). Закрывающая скобка заменяется одним из следующих слов: endif;, endwhile;, endfor;, endswitch;, (в зависимости от оператора).

Наиболее часто такие конструкции используются при осуществлении вставки блоков HTML внутрь оператора IF, но они могут использоваться в любом месте. Вместо использования фигурных скобок за «IF (statement)» должно следовать двоеточие, одно или несколько выражений и завершающий ENDIF. Вот пример:

```
<?php if ($a==5): ?> A = 5 <?php endif; ?>
```

В этом примере блок «A = 5» вставлен внутрь выражения IF, который представлен с применением двоеточия. Блок HTML будет передан клиенту только тогда, когда $a равно 5.

Такой синтаксис применим как к ELSE, так и к ELSEIF (expr). Вот еще один пример:

```
if ($a == 5):
    print "a РАВНО 5";
    print "...";
elseif ($a == 6):
    print "a РАВНО 6";
    print "!!!";
else:
    print "a НЕ РАВНО НИ 5, НИ 6";
endif;
```

См. также while, for, if.

while

Смысл действий оператора WHILE весьма прост. Этот оператор аналогичен оператору while в языке C.

```
while (expr) statement
```

Он предписывает выполнять вложенный в тело конструкции while оператор (или набор операторов, заключенных в фигурные скобки) до тех пор, пока выражение expr имеет значение TRUE. Значение выражения проверяется каждый раз при очередном начале цикла (при возврате к началу блока операторов), так что если значение выражения изменится внутри цикла, то он не прервется до конца блока. В случае, если значение expr равно FALSE с самого начала, цикл не выполняется ни разу.

Как и в случае с оператором IF, мы можем сгруппировать несколько операторов внутри фигурных скобок или использовать альтернативный синтаксис:

```
WHILE(expr): выражения ... ENDWHILE;
```

Следующие варианты представления операторов эквивалентны (оба выводят числа с 1 по 10):

```
/* example 1 */

$i = 1;
while ($i <= 10) {
    print $i++;  }

/* example 2 */

$i = 1;
while ($i <= 10):
    print $i;
    $i++;
endwhile;
```

do..while

Цикл DO..WHILE очень похож на WHILE за исключением того отличия, что значение логического выражения в случае с оператором do ... while проверяется не до, а после окончания выполнения блока операторов. Основное отличие состоит в том, что DO..WHILE гарантированно выполнится хотя бы один раз, что в случае WHILE не обязательно.

Для циклов DO...WHILE существует только один вид синтаксиса:

```
$i = 0;
do {
     print $i;
} while ($i>0);
```

Этот цикл выполнится один раз, так как после окончания итерации будет проверено значение логического выражения, а оно равно FALSE ($i не больше 0), и выполнение цикла завершится.

Программисты, использующие язык C, могут быть знакомы с иным вариантом употребления DO...WHILE, позволяющем прекратить исполнение блока операторов в середине, не выполняя блок до конца, путем внедрения его в цикл DO...WHILE(0) оператора BREAK. Следующий пример демонстрирует такую возможность:

```
do {
     if ($i < 5) {
          print "i еще не слишком велико";
          break;
     }
     $i *= $factor;
     if ($i < $minimum_limit) {
          break;
     }
     print "i уже достаточно велико";
     ... дальнейший код...
} while(0);
```

for

Циклы FOR — наиболее мощные циклы в PHP. Они работают подобно тому, как работают циклы FOR в C. Синтаксис цикла FOR:

```
FOR (expr1; expr2; expr3) statement
```

Первое выражение (expr1) всегда вычисляется в начале цикла. В начале каждой итерации вычисляется expr2. Если оно равно TRUE, то цикл продолжается и выполняются вложенный оператор (или блок операторов). Если оно равно FALSE, то цикл заканчивается. В конце каждой итерации вычисляется expr3.

Каждое из этих выражений может быть пустым. Если expr2 пусто, то цикл продолжается бесконечно (PHP по умолчанию считает его равным TRUE, как и в C). Такая ситуация оказывается не столь бесполезной, как может показаться, так как зачастую бывает нужно закончить выполнение цикла, используя оператор BREAK в сочетании с логическим условием, вместо использования логического выражения в FOR.

Рассмотрим следующие примеры. Все они выводят числа с 1 по 10:

```
/* пример 1 */

for ($i = 1; $i <= 10; $i++) {
     print $i;
}

/* пример 2 */

for ($i = 1;;$i++) {
     if ($i > 10) {
          break;
     }
     print $i;
}

/* пример 3 */

$i = 1;
for (;;) {
     if ($i > 10) {
          break;
     }
     print $i;
     $i++;
}

/* пример 4 */

for ($i = 1; $i <= 10; print $i, $i++) ;
```

Конечно, первый вариант кажется лучшим (или, может быть, четвертый). Важно то, что возможность использования пустых выражений в цикле FOR зачастую оказывается полезной.

PHP также поддерживает альтернативный синтаксис FOR:

```
FOR (expr1; expr2; expr3): выражение; ...; endfor;
```

Другие языки используют оператор foreach для того, чтобы обрабатывать массивы или списки. PHP использует для этого оператор while и функции ***list()*** и ***each()***. (Смотрите описания этих функций.)

foreach

В PHP4 существует конструкция foreach, напоминающая аналогичные конструкции других языков, например, языка Perl. Эта конструкция предоставляет удобный вариант просмотреть все элементы массива и выполнить определенные действия с ними. Эта конструкция может быть использована с применением двух вариантов синтаксиса. Первый вариант:

```
foreach(array_expression as $value) statement
```

Второй вариант представляется небольшим расширением первого:

```
foreach(array_expression as $key => $value) statement
```

Такая конструкция организует цикл проверки всех элементов массива, имя которого дано в array_expression. При каждом повторе цикла значение текущего элемента массива присваивается переменной $value, а внутренний указатель в массиве перемещается на одну позицию вперед.

Второй вариант синтаксиса позволяет задавать текущее значение для $key, равное значению элемента массива, для каждого цикла.

Примечание.

В начале выполнения цикла внутренний указатель возвращается к первому элементу массива, поэтому вызывать функцию reset() перед циклом нет необходимости. Кроме того, оператор foreach работает не с самим массивом, а с копией этого массива.

Следующие примеры оказываются функционально одинаковыми:

```
reset ($arr);
while (list(, $value) = each ($arr)) {
    echo "Value: $value<br>\n";
}

foreach ($arr as $value) {
    echo "Value: $value<br>\n";
}
```

Еще один пример одинаковых конструкций:

```
reset ($arr);
while (list($key, $value) = each ($arr)) {
    echo "Key: $key; Value: $value<br>\n";
}

foreach ($arr as $key => $value) {
    echo "Key: $key; Value: $value<br>\n";
}
```


Еще несколько примеров использования конструкции foreach:

```
 /* foreach example 1: value only */
$a = array (1, 2, 3, 17);
foreach ($a as $v) {
    print "Current value of \$a: $v.\n";
}
/* foreach example 2: value (with key printed for illustration) */

$a = array (1, 2, 3, 17);
$i = 0; /* for illustrative purposes only */

foreach($a as $v) {
    print "\$a[$i] => $k.\n";
}
/* foreach example 3: key and value */
$a = array (
    "one" => 1,
    "two" => 2,
    "three" => 3,
    "seventeen" => 17
);

foreach($a as $k => $v) {
    print "\$a[$k] => $v.\n";
}
```

break

Оператор break прерывает выполнения циклов for, while и структур switch.

break может (но не обязан) иметь числовой аргумент, который указывает количество вложенных структур, прекращение выполнения которых будет осуществлено этим оператором break.

```
$arr = array ('one', 'two', 'three', 'four', 'stop', 'five');
while (list (, $val) = each ($arr)) {
    if ($val == 'stop') {
        break;    /* You could also write 'break 1;' here. */
    }
    echo "$val<br>\n";
}
/* Using the optional argument. */
```

```
$i = 0;
while (++$i) {
     switch ($i) {
     case 5:
          echo "At 5<br>\n";
          break 1;  /* Exit only the switch. */
     case 10:
          echo "At 10; quitting<br>\n";
          break 2;  /* Exit the switch and the while. */
     default:
          break;
     }
}
```

continue

Оператор continue используется внутри структур циклов для того, чтобы пропустить следующую за оператором часть цикла при выполнении текущей итерации (текущего шага цикла) и перейти к следующему шагу цикла, начав выполнение его с начала. Оператор continue может иметь необязательный числовой аргумент, указывающий количество вложенных циклов, окончание которых будет пропущено.

```
while (list ($key, $value) = each ($arr)) {
     if (!($key % 2)) { // skip odd members
          continue;
     }
     do_something_odd ($value);
}

$i = 0;
while ($i++ < 5) {
     echo "Outer<br>\n";
     while (1) {
          echo "  Middle<br>\n";
          while (1) {
               echo "  Inner<br>\n";
               continue 3;
          }
          echo "This never gets output.<br>\n";
     }
     echo "Neither does this.<br>\n";
}
```


switch

Оператор switch работает также, как последовательность, состоящая из нескольких операторов IF, которые имеют одинаковое выражение. Часто оказывается необходимым осуществлять проверку и сравнение одной и той же переменной (или выражения) и выполнять те или иные действия в зависимости от того, каким будет значение этой переменной (или выражения). Для этого используется оператор switch.

В приведенном ниже примере одна и та же задача решается двумя разными способами (при помощи операторов if и при помощи оператора switch).

```
if ($i == 0) {
     print "i equals 0";
}
if ($i == 1) {
     print "i equals 1";
}
if ($i == 2) {
     print "i equals 2";
}

switch ($i) {
     case 0:
          print "i equals 0";
          break;
     case 1:
          print "i equals 1";
          break;
     case 2:
          print "i equals 2";
          break;
}
```

Чтобы избежать ошибок, полезно понимать механизм работы оператора switch. Выражения оператора выполняются строка за строкой, однако вначале код не будет выполняться. Код будет выполнен лишь в том случае, когда выражение, стоящее после case, совпадет с указанным в начале оператора выражением. Затем будет выполняться код без прерываний до конца блока switch. Чтобы прервать выполнение операторов, необходимо поместить инструкцию break. Вот пример:

```
switch ($i) {
     case 0:
          print "i equals 0";
     case 1:
```

```
            print "i equals 1";
        case 2:
            print "i equals 2";
}
```

В этом примере если $i равняется 0, то будут выполнены все операторы print. Если $i равно 2, то будут выполнены только два оператора print. И лишь при выполнении условия равенства $i двум мы получаем то, что и предполагается, т.е. будет выполнен лишь последний оператор.

В операторе switch проверка условия осуществляется лишь один раз, а затем происходит сравнение этого результата со значениями, указанными в case. В случае с оператором elseif проверка условия происходит на каждом шаге.

Еще один пример:

```
switch ($i) {
    case 0:
    case 1:
    case 2:
        print "i is less than 3 but not negative";
        break;
    case 3:
        print "i is 3";
}
```

В этом операторе может использоваться специальная конструкция default, в которой можно указать выражение, выполняемое в том случае, если явных совпадений не было найдено. Вот пример:

```
switch ($i) {
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
    default:
        print "i is not equal to 0, 1 or 2";
}
```

Выражение, стоящее в операторе case, должно быть приводимо к простому типу, т.е. к целому числу или числу с плавающей точкой, а

также к строке. Объекты и массивы не могут использоваться в качестве выражения, если только они не являются ссылками на простые типы и не могут быть приведены к ним.

Для переключателей switch может быть использован альтернативный синтаксис.

```
switch ($i):
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
    default:
        print "i is not equal to 0, 1 or 2";
endswitch;
```

Некоторые функции, используемые для управления последовательностью выполнения инструкций

require()

Функция ***require()*** производит вставку (в том месте, где используется эта функция) запрашиваемого файла. Функция похожа на препроцессорную директиву #include в языке C.

Если в PHP включена возможность использования URL-адресов («URL fopen wrappers»), то в качестве аргумента функции ***require()*** может быть указан URL-адрес наряду с именами локальных файлов. См. также описание функции ***fopen()***.

При работе с функциями ***include()*** и ***require()***, важно иметь в виду то, что анализ PHP-кода прекращается в месте вставки файла, вставляемый фрагмент анализируется в режиме HTML, начиная с начала вставляемого файла. После того, как будет обнаружен конец вставляемого файла, вновь вступает в силу режим анализатора PHP. По этой причине все содержимое вставляемого файла, если оно должно быть выполнено с использованием PHP, должно быть вставлено внутрь открывающего и закрывающего ярлыков PHP.

Слово ***require()*** по сути своей не является функцией PHP, оно скорее представляет собой специальную конструкцию языка. По этим причинам эту «функцию» нельзя использовать внутри других, содержащих ее выражений, а также осуществлять попытки прочитать возвращаемое ею значение, так как она не возвращает никакого значения.

В отличие от функции ***include()***, функция ***require()*** всегда осуществляет вставку запрошенного файла даже в том случае, когда строка PHP-кода, в которой расположена инструкция ***require()***, не будет никогда выполнена. Если вставка файла должна быть осуществлена в зависимости от выполнения тех или иных условий, необходимо использовать инструкцию ***include()***. В то время как условное выражение не влияет на ***require()***, инструкция ***include()*** не будет выполнена, если встретится во фрагменте условного оператора, условие которого принимает значение false.

По аналогии с условными операторами, на поведение инструкции ***require()*** не влияют операторы циклов. Однако, условия циклов будут приняты во внимание в случае использования инструкции ***include()***. Это означает, что если инструкция ***require()*** будет помещена внутрь цикла, то вставка указанного в качестве аргумента файла будет осуществляться на каждом шаге оператора. Чтобы избежать этого, можно использовать инструкцию ***include()***.

```
require ('header.inc');
```

Вставленный при помощи инструкции ***require()*** код наследует все переменные, которые содержались в той строке исходного файла, в которой находилась инструкция ***require()***. Если же инструкция ***require()*** была размещена внутри функции, то весь код, размещенный в вызываемом файле, будет выполняться также, как если бы он было размещен внутри этой функции.

Если запрошенныйпри помощи ***require()*** файл открывается с использованием протокола HTTP с применением fopen, а сервер, на котором расположен файл, рассматривает запрошенный файл, как файл с PHP-кодом, то переменные могут быть переданы запрашиваемому файлу при помощи обычной строки запроса URL таким способом, как это делается с использованием метода HTTP GET. Такой способ по сути своей отличается от простого включения файла в вызывающий файл с помощью инструкции ***require()***, поскольку в этом случае файл предварительно будет обработан на удаленном сервере, так как все программные инструкции будут уже выполнены, а вместо инструкции ***require()*** будет вставлен результат выполнения PHP-кода на удаленном сервере.

```
/* В этом примере подразумевается, что сервер с именем someserver
сконфигурирован так, что производит обработку файлов .php, но не
обрабатывает файлы .txt files. */
```



```
/* Не работает. t");  /* Работает. */
require ("file.php");  /* Работает. */
```

См. также ***include(), require_once(), include_once(), readfile(), virtual().***

include()

Инструкция ***include()*** осуществляет вставку (в то место, где расположена инструкция) и выполнение указанного в качестве аргумента файла.

Если в PHP включена возможность использования URL-адресов («URL fopen wrappers»), то в качестве аргумента функции ***require()*** может быть указан URL-адрес наряду с именами локальных файлов. См. также описание функции ***fopen()***.

Вставляемый фрагмент рассматривается как HTML-код. Если вставляемый фрагмент должен быть интерпретирован как PHP-код, то такой фрагмент вставляемого файла должен быть расположен внутри открывающего и закрывающего ярлыков PHP.

Пример:

```
$files = array ('first.inc', 'second.inc', 'third.inc');
for ($i = 0; $i < count($files); $i++) {
    include $files[$i];
}
```

Инструкция ***include()*** отличается от ***require()***, она выполняется всякий раз, когда встречается как часть PHP-кода, в то время как инструкция ***require()*** выполняется единожды независимо от того, где она встречается в исходном PHP-коде и будет ли она выполнена по условиям операторов программы, т. е. вне зависимости, что покажет условие, если инструкция расположена внутри условного оператора, например, внутри оператора if.

Можно использовать такие конструкции, содержащие в себе ***include()***, как, например (учитывая, что ***onclude()*** — это не функция, а специальная конструкция языка):

```
/* Это, однако, ошибочно, и не будет работать так, как предполагается. */
if ($condition)
    include($file);
else
    include($other);
/* Это правильно. */
if ($condition) {
    include($file);
```

```
} else {
     include($other);
}
```

Еще один пример:

```
/* Здесь предполагается наличие файла test.inc, расположенного в
той же директории, что и основной файл. */
<?php
echo "Before the return <br>\n";
if (1) {
     return 27;
}
echo "After the return <br>\n";
?>
```

Предположим, что основной файл main.phtm содержит следующий код:

```
<?php
$retval = include ('test.inc');
echo "File returned: '$retval'<br>\n";
?>
```

При обращении к файлу main.phtm произойдет ошибка во второй строке, тем не менее результат работы должен быть таким (см. рис. 7.2):

```
Before the return
File returned: '27'
```

Сейчас изменим содержимое файла main.html и сделаем его таким:

```
<?php
include ('test.inc');
echo "Back in main.html<br>\n";
?>
```

Сейчас PHP4 возвратит такой ответ (см. рис. 7.3):

```
Before the return
Back in main.html
```

В варианте PHP3 мы увидели бы такой ответ:

```
Before the return
27Back in main.html
Parse error: parse error in
main.html on line 5
```

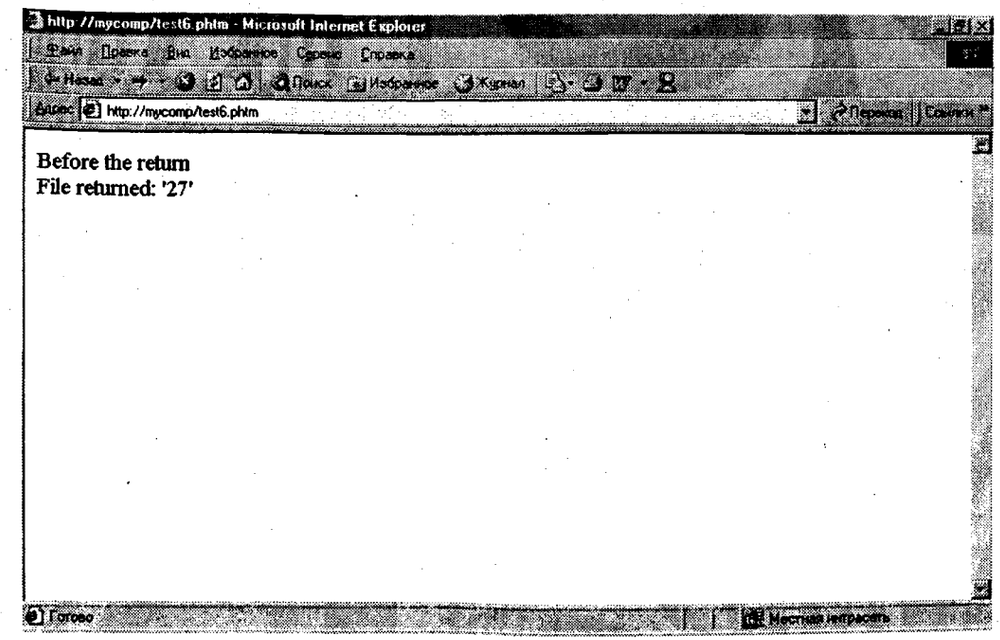



Рис. 7.2. *Вставка файла с помощью include()*

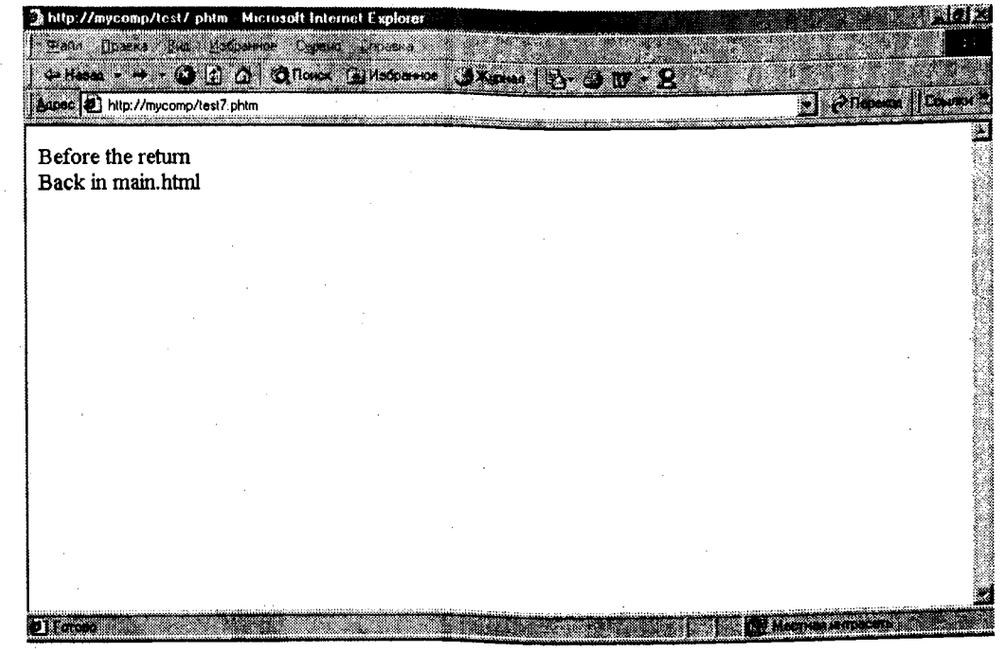


Рис. 7.3. *Вставка не видна*

Ошибка произошла потому, что выражение return было включено за пределами функционального блока в файле test.inc. Если исправить ошибку, то ответ будет таким:

```
Before the return
27Back in main.html)
```

Вставленный при помощи инструкции ***require()*** код наследует все переменные, которые содержались в той строке исходного файла, в которой находилась инструкция ***require()***. Если же инструкция ***require()*** была размещена внутри функции, то весь код, размещенный в вызываемом файле, будет выполняться так же, как если бы он было размещен внутри этой функции.

Если запрошенныйпри помощи ***include()*** файл открывается с использованием протокола HTTP с применением fopen, а сервер, на котором расположен файл, рассматривает запрошенный файл, как файл с PHP-кодом, то переменные могут быть переданы запрашиваемому файлу при помощи обычной строки запроса URL таким способом, как это делается с использованием метода HTTP GET. Такой способ по сути своей отличается от простого включения файла в вызывающий файл с помощью инструкции ***include()***, поскольку в этом случае файл предварительно будет обработан на удаленном сервере, так как все программные инструкции будут уже выполнены, а вместо инструкции ***include()*** будет вставлен результат выполнения PHP-кода на удаленном сервере.

```
/* Сервер someserver выполняет файлы .php, но не выполняет файлы .txt
files. */

/* Не будет работать; file.txt не исполняется сервером someserver. */
include ("http://someserver/file.txt?varone=1&vartwo=2");

/* не будет работать; поиск будет осуществлен в локальной файловой
системе для файла 'file.php?varone=1&vartwo=2'. */
include ("file.php?varone=1&vartwo=2");

/* Работает. */
include ("http://someserver/file.php?varone=1&vartwo=2");

$varone = 1;
$vartwo = 2;
include ("file.txt");   /* Работает. */
include ("file.php");   /* Работает. */
```

См. также require(), require_once(), include_once(), readfile(), virtual().

require_once()

Эта инструкция работает во многом схожим образом с тем, как работает инструкция ***require()***. Единственное отличие состоит в том, что эта инструкция осуществляет вставку указанного файла в исходный текст лишь один раз, что позволяет избежать возникновения ошибок. Вот пример использования ***require_once()***:

```
<?php
define(PHPVERSION, floor(phpversion()));
echo "GLOBALS ARE NICE\n";
function goodTea() {
   return "Oolong tea tastes good!";
}
?>
Файл foolib.inc
<?php
require ("utils.inc");
function showVar($var) {
   if (PHPVERSION == 4) {
      print_r($var);
   } else {
      dump_var($var);
   }
}
// прочие инструкции ...
```

Затем можно создать файл cause_error_require.phtm

Пример файла cause_error_require.phtm

```
<?php
require("foolib.inc");
/* the following will generate an error */
require("utils.inc");
$foo = array("1",array("complex","quaternion"));
echo "this is requiring utils.inc again which is also\n";
echo "required in foolib.inc\n";
echo "Running goodTea: ".goodTea()."\n";
echo "Printing foo: \n";
showVar($foo);
?>
```

Изменения в foolib.inc:

```
...
require_once("utils.inc");
function showVar($var) {
...
```

Файл avoid_error_require_once.phtm

```
...
require_once("foolib.inc");
require_once("utils.inc");
$foo = array("1",array("complex","quaternion"));
...
После выполнения мы получим:
GLOBALS ARE NICE
This is requiring globals.inc again which is also
Required in foolib.inc
Running goodTea: Oolong tea tastes good!
Printing foo:
Array
(
     [0] => 1
     [1] => Array
          (
               [0] => complex
               [1] => quaternion
          )
)
```

Заметим, что данная инструкция работает по аналогии с директивой препроцессора C #include, т.е. в момент работы анализатора и перед тем, как скрипт будет выполнен. Вставляемый фрагмент из файла не должен быть фрагментом, вставляемым динамически во время выполнения скрипта. Для таких целей подойдут инструкции ***include()*** и ***include_once()***.

См. также ***require(), include(), include_once(), get_required_files(), get_included_files(), readfile(), virtual()***.

include_once()

Инструкция ***include_once()*** осуществляет вставку и исполнение кода файла, указанного в качестве аргумента. Поведение этой инструкции схоже с поведением инструкции ***include()*** с той разницей, что вставка кода происходит только один раз.

См. также ***require(), include(), require_once(), get_required_files(), get_included_files(), readfile(), virtual()***.

Функции

В этом разделе мы рассмотрим такие вопросы.

- Функции, определяемые пользователем.
- Аргументы функции.
- Возвращаемые значения.
- Ключевое слово old_function.
- Переменные функции.

Функции, определяемые пользователем

Функции могут быть определены с использованием следующего синтаксиса:

```
function foo ($arg_1, $arg_2, ..., $arg_n) {
     echo "Example function.\n";
     return $retval;
}
```

В теле функции может быть размещен любой допустимый PHP-код, включая описания других функций и классов. В PHP3 функция должна была быть определена до того, как к ней будет произведено обращение. В PHP4 таких ограничений уже нет. PHP не поддерживает перегрузку типов для функций, здесь также нет возможности отменить определение функции или определить функцию заново.

В PHP3 не поддерживается переменное число аргументов функции, хотя поддерживается использование аргументов по умолчанию. В PHP4 поддерживается как переменное число аргументов, так и аргументы по умолчанию, а также ссылки на другие функции. Для получения дополнительной информации см. ***func_num_args(), func_get_arg(), func_get_args()***.

Аргументы функции

Информация может быть передана функции с использованием списка аргументов функции, который представляет собой список переменных и констант, отделенных друг от друга запятыми.

В PHP используется передача переменных с их значениями, передача по ссылке и передача значений по умолчанию. Возможно использования списка аргументов переменной длины и ссылок для функций, см. описание ***func_num_args(), func_get_arg(), func_get_args()***.

Пример:

```
function takes_array($input) {
     echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
```

Аргументы, передаваемые по ссылкам

По умолчанию аргумент функции передаются в виде своих значений, таким образом, если значение переменной будет изменено внутри функции, то значение переменной за пределами функции не будет изменено. Если стоит задача изменения значения переменной и за пределами функции, то аргументы должны быть переданы по ссылке.

Для того, чтобы аргумент был передан функции по ссылке, следует использовать знак &, поместив его перед именем аргумента в описании функции. Вот пример:

```
function add_some_extra(&$string) {
    $string .= 'and something extra.';
}
$str = 'This is a string, ';
add_some_extra($str);
echo $str;     // outputs 'This is a string, and something extra.'
```

Если необходимо передать аргумент по ссылке для функции, которая не принимает ссылки на аргументы по умолчанию, то сделать это можно, поместив знак & перед именем аргумента при обращении к функции. Например:

```
function foo ($bar) {
    $bar .= ' and something extra.';
}
$str = 'This is a string, ';
foo ($str);
echo $str;     // outputs 'This is a string, '
foo (&$str);
echo $str;     // outputs 'This is a string, and something extra.'
```

Значения аргументов по умолчанию

Можно определить значения аргментов по умолчанию внутри самой функции подобно тому, как это делается в C++ для скалярных аргументов:

```
function makecoffee ($type = "cappucino") {
    return "Making a cup of $type.\n";
}
echo makecoffee ();
echo makecoffee ("espresso");
```

Этот код выведет такой ответ (рис. 7.4):

```
Making a cup of cappucino.
Making a cup of espresso.
```

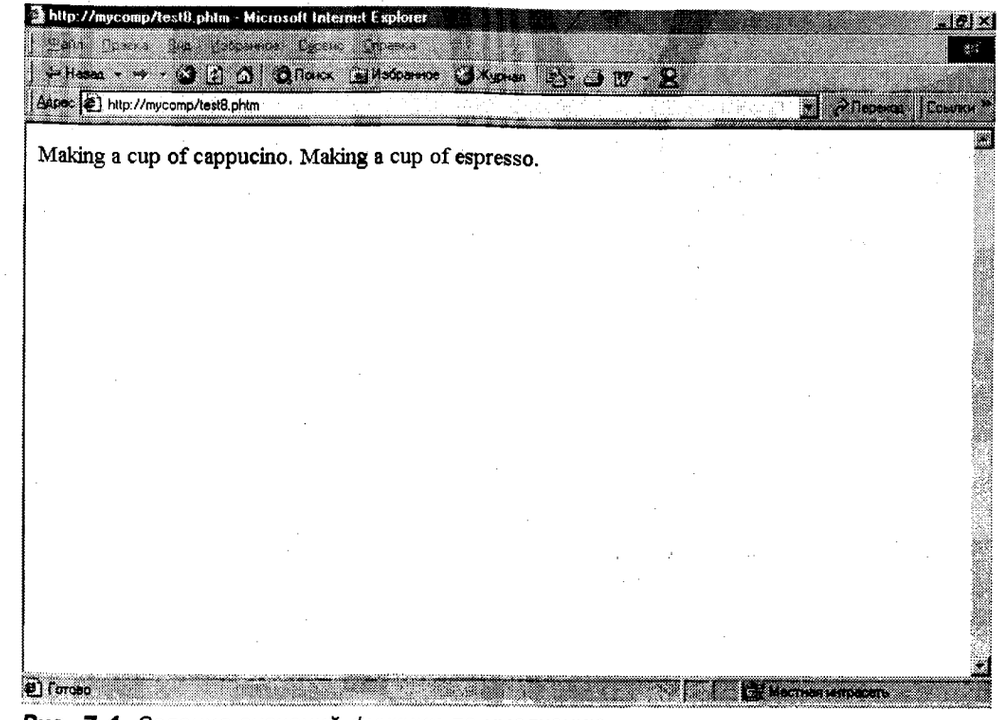



Рис. 7.4. *Задание значений функции по умолчанию*

Значением по умолчанию должна быть константа, а не, например, переменная или экземпляр класса.

При задании аргументов по умолчанию следует помнить, что все значения аргументов, задаваемые по умолчанию, должны быть описаны справа по отношению к тем аргументам, для которых не указываются значения по умолчанию. В противном случае программа не будет работать таким образом, как задумано. Вот пример:

```
function makeyogurt ($type = "acidophilus", $flavour) {
    return "Making a bowl of $type $flavour.\n";
}

echo makeyogurt ("rasberry");
// не будет работать как положено
```

Этот пример выведет примерно такой результат (рис. 7.5):

```
Warning: Missing argument 2 in call to makeyogurt() in
functest.html on line 41
Making a bowl of rasberry .
```

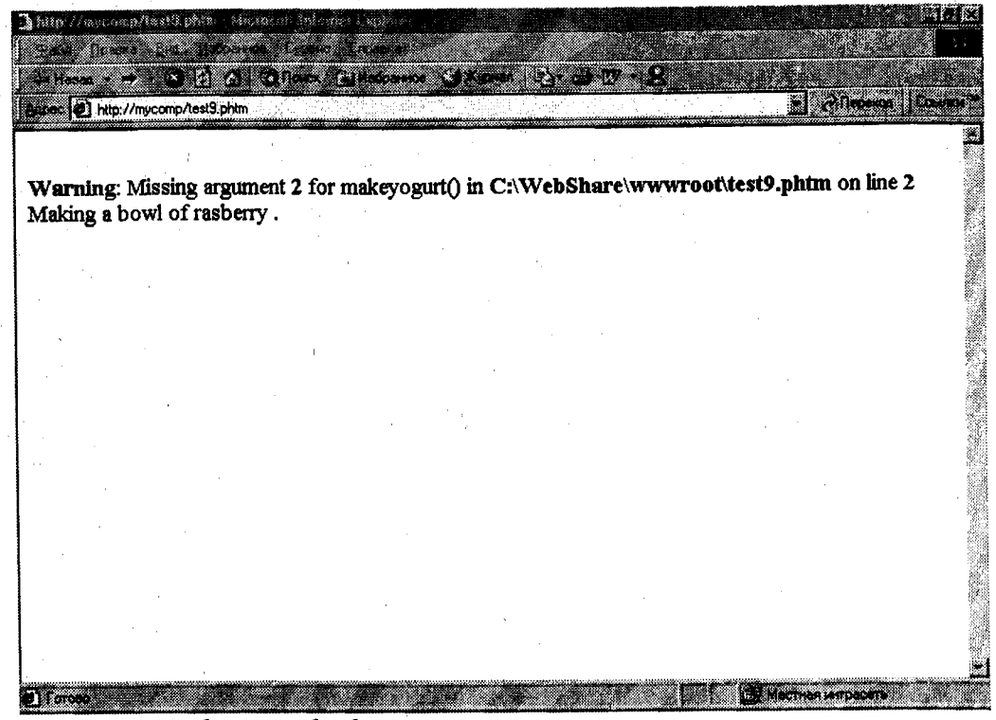


Рис. 7.5. *Ошибка при обработке аргументов*

Сравним предыдущий пример со следующим:

```
function makeyogurt ($flavour, $type = "acidophilus") {
     return "Making a bowl of $type $flavour.\n";
}

echo makeyogurt ("rasberry");
// работает так, как и предоплагалось
```

Результат работы будет выглядеть так (рис. 7.6):

```
Making a bowl of acidophilus raspberry.
```

Список аргументов переменной длины

В PHP4 список аргументов может состоять из переменного числа аргументов, размещенных в списке. Для работы со списками аргументов переменной длины удобно использовать функции ***func_num_args(), func_get_arg(), func_get_args()***. Для создания функций со списком аргументов переменной длины нет специальных правил, функции определяются так же, как и обычно.

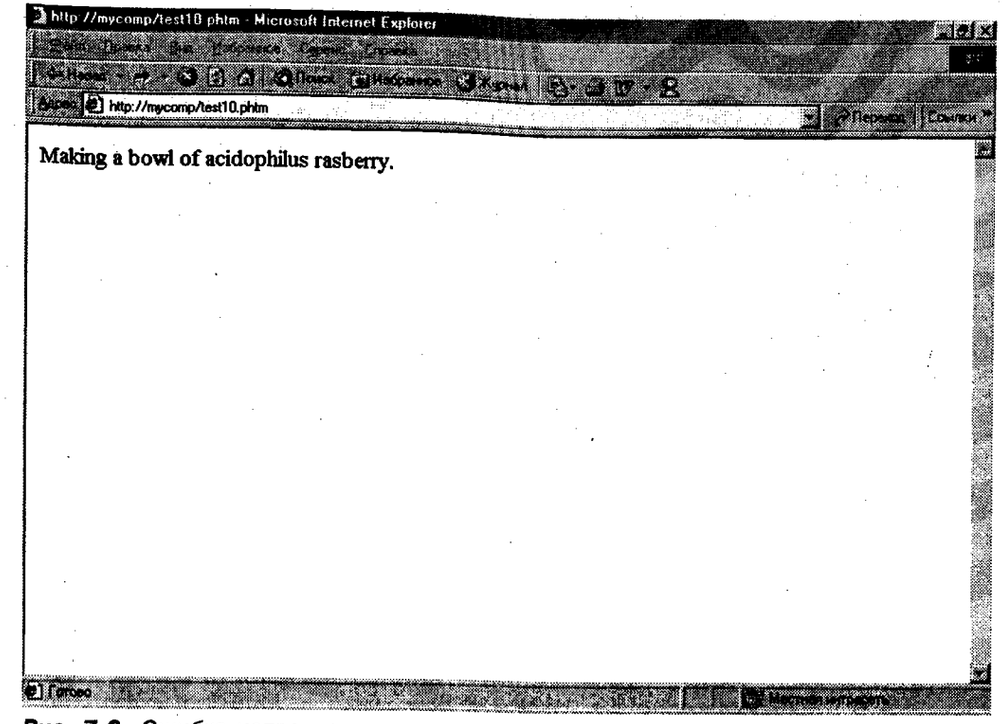


Рис. 7.6. *Ошибка исправлена*

Значения, возвращаемые функцией

Значение может быть возвращено функцией, для этого может быть использован оператор return. Функция может возвращать значения любого типа, включая списки объектов.

```
function square ($num) {
     return $num * $num;
}
echo square (4);    // outputs '16'.
```

Функция не может возвратить несколько значений одновременно, однако, можно возвратить список значений, и это во многом подобно тому, как если бы было возвращено несколько значений сразу.

```
function small_numbers() {
     return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
```

Чтобы функция могла возвратить указатель, расположенный внутри функции, необходимо использовать оператор ссылки & как в опи-

сании функции, так и в том месте, где возвращаемое значение присваивается переменной:

```
function &returns_reference() {
     return &$someref;
}

$newref = &returns_reference();
```

old_function

Выражение old_function позволяет создавать функции с использованием старого синтаксиса PHP/FI2, при этом слово functiuon следует заменить словом old_function.

Переменные функции

В PHP поддерживаются переменные функции. Это значит то, что если в скрипте встречается переменная, за которой следуют скобки, то PHP произведет поиск функции с именем, равным тому, к чему может быть приведено значение упомянутой переменной, после чего будет предпринята попытка выполнить эту функцию. Такое поведение может быть использовано для создания повторных обращений к функции, использования таблиц функций и т.п. Вот пример:

```
<?php
function foo() {
     echo "In foo()<br>\n";
}

function bar( $arg = '' ) {
     echo "In bar(); argument was '$arg'.<br>\n";
}

$func = 'foo';
$func();
$func = 'bar';
$func( 'test' );
?>
```

Результат выполнения файла показан на рис. 7.7.

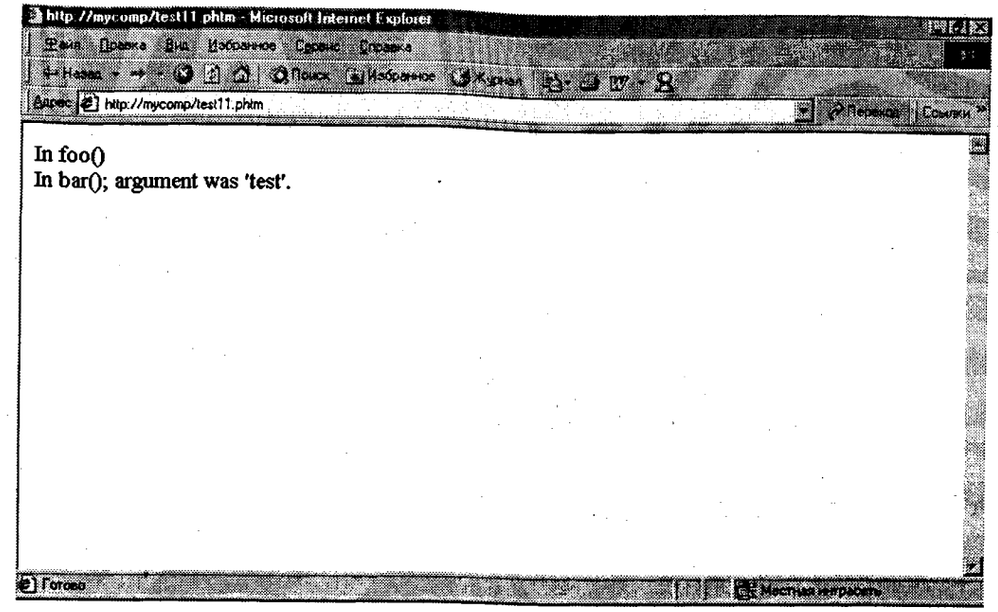


Рис. 7.7. Переменные функции

Классы и объекты

class

Класс представляет собой набор переменных и функций, работающих с этими переменными. Класс определяется с использованием следующего синтаксиса:

```
<?php
class Cart {
     var $items;  // Items in our shopping cart

     // Add $num articles of $artnr to the cart

     function add_item ($artnr, $num) {
          $this->items[$artnr] += $num;
     }

     // Take $num articles of $artnr out of the cart
```

```
    function remove_item ($artnr, $num) {
        if ($this->items[$artnr] > $num) {
            $this->items[$artnr] -= $num;
            return true;
        } else {
            return false;
        }
    }
}
?>
```

В этом примере определен класс с именем Cart, который состоит из ассоциативного массива предметов, расположенных в корзине Cart и двух функций, с помощью которых происходит добавление предметов в корзину и удаление их из корзины.

Классы представляют собой типы, таким образом, классы являются своего рода шаблонами для переменных. Переменная типа того или иного класса может быть создана с применением оператора new.

```
$cart = new Cart;
$cart->add_item("10", 1);
```

В этом примере создается объект (экземпляр класса) $cart класса Cart. Функция ***add_item()*** этого объекта добавляет один предмет в корзину, номер предмета 10. Классы могут представлять собой расширения других классов, являющихся базовыми для данного класса. Такие производные классы содержат в себе все переменные и функции базового класса, а также все то, что было добавлено в класс, но не содержалось в базовом классе. Наследование осуществляется с использованием ключевого слова extends. Множественное наследование не поддерживается.

```
class Named_Cart extends Cart {
    var $owner;

    function set_owner ($name) {
        $this->owner = $name;
    }
}
```

В вышеприведенном примере описан класс Named_Cart, в котором есть все функции и переменные класса Cart, а также дополнительная переменная $owner и функция ***set_owner()***. С использованием этого класса можно создать именованную корзину и задать (или прочитать) владельца этой корзины:

```
$ncart = new Named_Cart;     // создаем именованную корзину
$ncart->set_owner ("kris"); // задаем имя корзины
```



```
print $ncart->owner;          // печатаем имя владельца корзины
$ncart->add_item ("10", 1); // функции, унаследованные из cart
```

Внутри классов переменная $this указывает на текущий объект. Для того, чтобы получить доступ к любой переменной или функции текущего объекта, можно использовать конструкцию $this->something.

Конструкторы внутри классов представляют собой такие функции, которые автоматически вызываются в момент создания нового экземпляра класса. Функция становится конструктором тогда, когда имя функции совпадает с именем класса.

```
class Auto_Cart extends Cart {
    function Auto_Cart () {
        $this->add_item ("10", 1);
    }
}
```

В этом примере определен класс Auto_Cart, который наследуется из класса Cart, но в добавок содержит конструктор, который инициализирует новый объект этого класса, помещая в корзину один предмет с номером «10». Конструктор может иметь аргументы, однако наличие таких аргументов необязательно. Это делает использование конструкторов более гибким.

```
class Constructor_Cart extends Cart {
    function Constructor_Cart ($item = "10", $num = 1) {
        $this->add_item ($item, $num);
    }
}

// Shop the same old boring stuff.

$default_cart    = new Constructor_Cart;

// Shop for real...

$different_cart = new Constructor_Cart ("20", 17);
```

Внимание

Для производных классов, наследующих переменные и функции из родительского класса, при создании экземпляров производных классов не происходит автоматического вызова конструктора родительского класса.

Ссылки

В этом разделе мы рассмотрим следующие вопросы.

- Что такое ссылки.
- Для чего используются ссылки.
- Чем не являются ссылки.
- Возвращение значений при помощи ссылок.
- Удаление ссылок.

Что представляют собой ссылки

Ссылки в PHP представляют собой средство обращения к одним и тем же переменным при помощи разных имен. Такие ссылки отличаются от ссылок, используемых в языке C. В PHP имена ссылок и содержание переменных различаются, иными словами одно и то же содержимое может быть названо несколькими разными именами.

Для чего используются ссылки

Ссылки в PHP позволяют двум различным переменным использовать одно и то же содержание. Вот пример:

```
$a =& $b
```

Это обозначает, что и $a и $b содержат одно и то же значение.

Примечание.

Здесь $a и $b абсолютно равноценны, т.е. неверно было бы считать, что $a указывает на $b или наоборот. Здесь и $a и $b указывают на одно и то же «место».

Второе, для чего могут быть использованы ссылки — это передача значений по ссылкам. Для этого создается ситуация, когда локальная переменная, определенная внутри функции, а также внешняя переменная, к которой обращается данная функция, указывают на одно и то же место, например:

```
function foo (&$var) {
     $var++;
}

$a=5;
foo ($a);
```

Здесь обращение к функции приведет к тому, что $a станет равно 6. Это произойдет потому, что в функции foo переменная $var ссылается на то же самое место, что и переменная $a.

Третья причина использования указателей — возвращение значений функции при помощи ссылок.

Для чего нельзя использовать ссылки

Как уже было сказано выше, ссылки не являются указателями в том смысле, как они понимаются в языке C. По этой причине приведенный ниже пример не будет работать так, как ожидается на первый взгляд.

```
function foo (&$var) {
     $var =& $GLOBALS["baz"];
}
foo($bar);
```

При обращении к функции foo переменная $var, которая используется в качестве формального параметра при описании функции, будет указывать на то же содержимое, что и переменная $bar, указанная в качестве фактического параметра при обращении к функции. Затем она будет связана с переменной $GLOBALS[«baz»]. Не существует другого способа связывания переменной $bar с чем-либо иным, если используется механизм ссылок, поскольку переменная $bar не доступна внутри функции foo. Эта переменная внутри функции представлена другим именем — именем $var.

Ссылки, возвращающие значения в функциях

Использование возвращения значений по ссылкам полезно в тех случаях, когда в функции используется такая переменная, которая может быть связана с другой переменной. Вот пример:

```
function &find_var ($param) {
     ...code...
     return $found_var;
}

$foo =& find_var ($bar);
$foo->x = 2;
```

Примечание.

Символ & необходимо использовать в двух местах — один раз, чтобы показать, что значение возвращается по ссылке, а затем, чтобы показать, как возвращаемое значение связано с другими переменными.

Удаление ссылок

В момент удаления ссылки мы удаляем связь между двумя именами, ссылающимися на одно и то же место. Это, однако, не означает того, что содержимое переменных будет уничтожено, например:

```
$a = 1;
$b =& $a;
unset ($a);
```

Глобальные ссылки

Если переменная определена как глобальная переменная $var, то по сути создается ссылка на глобальную переменную как если бы была выполнена следующая инструкция:

```
$var =& $GLOBALS["var"];
```

При этом, конечно, удаление ссылки для переменной $var не приведет к удалению глобальной переменной.

$this

Ссылка $this всегда указывает на текущий объект (на текущий экземпляр класса).

ФУНКЦИИ В PHP

Функции для работы с сервером Apache

Apache_lookup_uri

Выполняет запрос по указанному URI и возвращает информацию

Apache_note

Получает и устанавливает значения в таблице примечаний запроса Apache

Getallheaders

Выбор всех заголовков HTTP-запросов

Пример GetAllHeaders():

```
$headers = getallheaders();
while (list($header, $value) = each($headers)) {
     echo "$header: $value<br>\n";
}
```

Этот пример покажет все заголовки текущего запроса.

virtual

Выполняет подзапрос Apache

Функции для работы с массивами

Эти функции используются для манипуляций с массивами, в которых хранятся значения наборов переменных. Поддерживаются как простые одномерные массивы, так и многомерные массивы. Массивы могут быть созданы в явном виде или с помощью вызова тех или иных функций, которые возвращают массивы.

array

Создает массив

Следующий пример демонстрирует, как можно создать двумерный массив.

Пример array():

```
$fruits = array(
     "fruits"  => array("a"=>"orange","b"=>"banana","c"=>"apple"),
     "numbers" => array(1, 2, 3, 4, 5, 6)
     "holes"    => array("first", 5 => "second", "third")
);
```

array_count_values

Пересчитывает значения, содержащиеся в массиве

array_walk

Применение функции к каждому члену массива

Описание

```
int array_walk(array arr, string func);
```

Применяет функцию с именем func для каждого элемента массива arr.

Пример array_walk():

```
$fruits = array("d"=>"lemon","a"=>"orange","b"=>"banana","c"=>"apple");
function test_alter( $item1 ) {
    $item1 = 'bogus';
}
function test_print( $item2 ) {
    echo "$item2<br>\n";
}
array_walk( $fruits, 'test_print' );
array_walk( $fruits, 'test_alter' );
array_walk( $fruits, 'test_print' );
```


arsort

Сортировка массива в обратном порядке

Пример arsort():

```
$fruits = array("d"=>"lemon","a"=>"orange","b"=>"banana","c"=>"apple");
arsort($fruits);
for(reset($fruits); $key = key($fruits); next($fruits)) {
     echo "fruits[$key] = ".$fruits[$key]."\n";
}
```

Пример нам покажет:

```
fruits[a] = orange
fruits[d] = lemon
fruits[b] = banana
fruits[c] = apple
```

Фрукты показаны в обратном алфавитном порядке, и поддерживаются связанные с ними индексы.

См. также asort(), rsort(), ksort(), sort().

asort

Сортирует массив и поддерживает связанные индексы

Пример asort():

```
$fruits = array("d"=>"lemon","a"=>"o range","b"=>"banana","c"=>"apple");
asort($fruits);
for(reset($fruits); $key = key($fruits); next($fruits)) {
     echo "fruits[$key] = ".$fruits[$key]."\n";
}
```

Этот пример приведет к следующим значениям:

```
fruits[c] = apple
fruits[b] = banana
fruits[d] = lemon
fruits[a] = orange
```

Фрукты показаны в алфавитном порядке, и поддерживаются индексы связанные с каждым элементом массива.

См. также arsort(), rsort(), ksort(), sort().

count

Возвращает число элементов в var

```
int count(mixed var);
```

Возвращает 0, если переменная не определена.

Возвражает 1, если переменная не является массивом.

*См. также **sizeof(), isset(), is_array().***

current

Возвращает текущий элемент массива

Описание

```
mixed current(array array);
```

Каждая переменная-массив имеет внутренний указатель, который указывает на один из своих элементов. Кроме того, все элементы в массиве связываются двунаправленным списком указателей для дополнительных целей.

Функция ***current()*** возвращает элемент массива, на который в данный момент указывает внутренний указатель. Она не перемещает указатель. Если внутренний указатель указывает на конец списка элементов, ***current()*** возвращает false (ложно).

Внимание: если массив содержит пустые элементы (0 или «», пустую строку), то для каждого такого элемента функция возвратит «false».

*См. также **end(), next(), prev(), reset().***

each

Возвращает следующую пару ключ/значение из массива

Описание

```
array each(array array);
```

Возвращает пару ключ/значение.

Пример each():

```
$foo = array( "bob"", "fred", "jussi", "jouni2 );
$bar = each( $foo );
       $bar теперь содержит следующие пары ключ/значение:
0 => 0
1 => 'bob'
key => 0
value => 'bob'
$foo = array( "Robert" => "Bob", "Seppo" => "Sepi" );
$bar = each( $foo );
```


Значения $bar таковы:

- 0 => 'Robert'
- 1 => 'Bob'
- key => 'Robert'
- value => 'Bob'

Each(), как правило, используется вместе с ***list()*** чтобы просмотреть массив; например, $HTTP_POST_VARS:

Пример. Просмотр $HTTP_POST_VARS с помощью each():

```
echo "Values submitted via POST method:<br>";
while ( list( $key, $val ) = each( $HTTP_POST_VARS ) ) {
    echo "$key => $val<br>";
}
```

*См. также **key(), list(), current(), reset(), next(), prev().***

end

Устанавливает внутренний указатель массива на последнем элементе

Описание

```
end(array array);
```

End() перемещает array's внутренний указатель на последний элемент массива.

*См. также **current(), each(), end(), next(), reset().***

key

Выбирает ключ из ассоциативного массива

Описание

```
mixed key(array array);
```

Key() возвращает индекс элемента в текущей позиции массива

*См. также **current(), next().***

ksort

Сортирует массив по ключам

Описание

```
int ksort(array array);
```

Пример ksort():

```
$fruits = array("d"=>"lemon","a"=>"orange","b"=>"banana","c"=>"apple");
ksort($fruits);
```

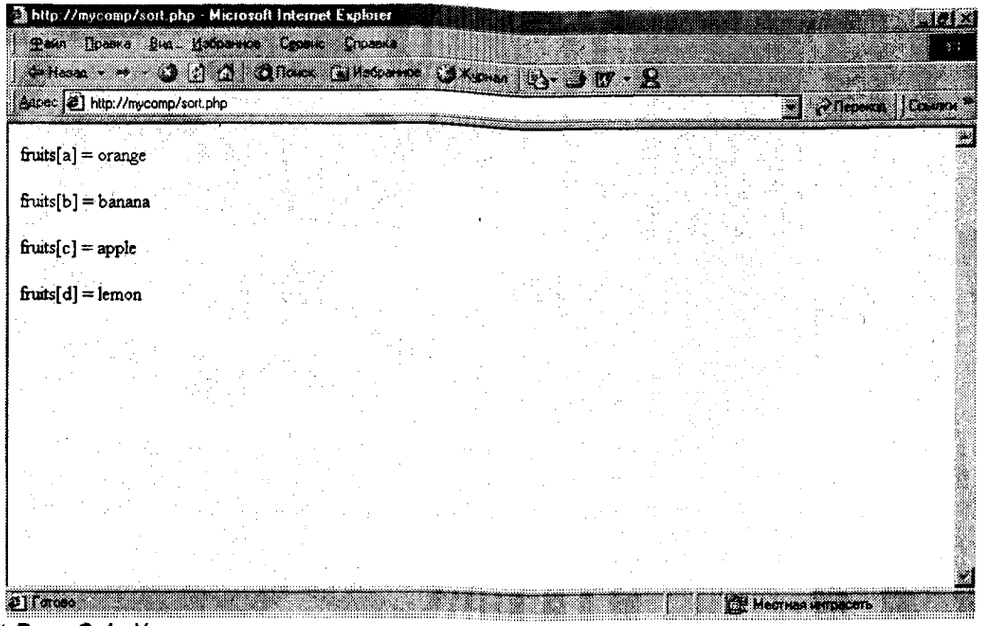


Рис. С.1. *Упорядочение массива*

```
for(reset($fruits); $key = key($fruits); next($fruits)) {
    echo "fruits[$key] = ".$fruits[$key]."\n";
}
```

После выполнения (см. рис. С.1):

```
fruits[a] = orange
fruits[b] = banana
fruits[c] = apple
fruits[d] = lemon
```

См. также ***asort(), arsort(), sort(), rsort().***

list

Связывает переменные, как если бы они были массивом

Пример list():

```
<table>
 <tr>
  <th>Employee name</th>
  <th>Salary</th>
 </tr>
<?php
$result = mysql($conn, "SELECT id, name, salary FROM employees");
while (list($id, $name, $salary) = mysql_fetch_row($result)) {
    print(" <tr>\n2.
          "  <td><a href=\"info.php?id=$id\">$name</a></td>\n".
          "  <td>$salary</td>\n".
          " </tr>\n");
}
?></table>
```

См. также ***each(), array().***

next

Передвигает внутренний указатель массива

Описание

```
mixed next(array array);
```

Возвращает следующий элемент массива, по сравнению с текущей позицией, на которую указывает внутренней указатель, или false, если в массиве больше нет элементов (false также будет возвращен для всех пустых элементов массива). Для просмотра массива, содержащего пустые элементы, рекомендуется использовать функцию ***each()***.

Next() ведет себя подобно ***current()***, с одной лишь разницей. Он передвигает внутренний указатель массива на один элемент вперед прежде, чем возвратить элемент. Это означает, что он возвращает значение следующего элемента и передвигает на него внутренний указатель массива. Если при обращении к следующему элементу обнаружен конец массива, то ***next()*** возвращает «ложь» (false).

См. также ***current(), end(), prev(), reset()***

pos

Возвращает текущий элемент массива

Описание

```
mixed pos(array array);
```

Это псевдоним для ***current()***.

См. также ***end(), next(), prev(), reset().***

prev

Перемещает внутренний указатель массива назад

Описание

```
mixed prev(array array);
```

Возвращает предыдущий элемент массива, или false, если перед текущим нет больше элементов (и для пустых элементоы массива).

Prev() ведет себя подобно ***next()***, за исключением того, что он переводит внутренний указатель массива на одну позицию назад, а не вперед.

См. также ***current(), end() next(), reset().***

reset

Устанавливает внутренний указатель массива на первом элементе

Описание

```
mixed reset(array array);
```

Reset() возвращает внутренний указатель массива в первый элемент.

Reset() возвращает первый элемент массива.

См. также ***current(), each(), next() prev(), reset().***

rsort

Сортирует массив в обратном порядке

Описание

```
void rsort(array array);
```

Сортирует массив в обратном порядке (по убыванию).

Пример rsort():

```
$fruits = array("lemon","orange","banana","apple");
    rsort($fruits);
    for(reset($fruits); ($key,$value) = each($fruits); ) {
        echo "fruits[$key] = ".$value."\n";
  }
```

Результат работы:

```
fruits[0] = orange
fruits[1] = lemon
fruits[2] = banana
fruits[3] = apple
```

Фрукты отсортированы в обратном алфавитном порядке.

См. также ***arsort(), asort(), ksort(), sort(), usort().***

sizeof

Получает размер массива

Описание

```
int sizeof(array array);
```

Возвращает число элементов списка.

См. также ***count().***

sort

Сортирует массив

Описание

```
void sort(array array);
```

Эта функция сортирует массив — все элементы по окончании ее работы будут расположены по возрастанию.

Пример sort():

```
$fruits = array("lemon","orange","banana","apple");
sort($fruits);
for(reset($fruits); $key = key($fruits); next($fruits)) {
    echo "fruits[$key] = ".$fruits[$key]."\n";
}
```

Результат работы:

```
fruits[0] = apple
fruits[1] = banana
fruits[2] = lemon
fruits[3] = orange
```

Фрукты будут отсортированы по возрастанию в алфавитном порядке.

См. также ***arsort(), asort(), ksort(), rsort(), usort().***

uasort

Сортирует массив с использованием функций сравнения, определенных пользователем

Описание

```
void uasort(array array, function cmp_function);
```

Эта функция сортирует массив так, что индексы массива поддерживают их корреляцию с элементами массива.

uksort

Сортирует массив по ключам, с использованием функций сравнения, определенных пользователем

Описание

```
void uksort(array array, function cmp_function);
```

Эта функция отсортирует ключи массива, используя функции сравнения, определенные пользователем.

Пример uksort():

```
function mycompare($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}
$a = array(4 => "four", 3 => "three", 20 => "twenty", 10 => "ten");
uksort($a, mycompare);
while(list($key, $value) = each($a)) {
    echo "$key: $value\n";
}
```

Результат:

```
20: twenty
10: ten
4: four
3: three
```

См. также ***arsort(), asort(), uasort(), ksort(), rsort(), sort().***

usort

Сортирует массив по значениям, используя функции сравнения, определенные пользователем

Пример usort():

```
function cmp($a,$b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}
$a = array(3,2,5,6,1);
usort($a, cmp);
while(list($key,$value) = each($a)) {
    echo "$key: $value\n";
}
```


Математические функции BCMath

Эти функции доступны только в том случае, если включен параметр enable-bcmath.

bcadd

Сложение двух чисел произвольной точности

Описание

```
string bcadd(string левый операнд, string правый операнд, int [масштаб]);
```

Прибавляет левый операнд к правому операнду и возвращает сумму типа string (строковая переменная). Параметр масштаб используется для того, чтобы установить количество разрядов после десятичной отметки в результате.

См. также ***bcsub().***

bccomp

Сравнение двух чисел произвольной точности

Описание

```
int bccomp (string левый операнд, string правый операнд, int [масштаб]);
```

Сравнивает левый операнд с правым операндом и возвращает результат типа integer (целое). Параметр масштаб используется для установки количества цифр после десятичной отметки, используемых при сравнении. При равенстве двух операндов возвращается значение 0. Если левый операнд больше правого операнда возвращается +1, и если левый операнд меньше правого операнда возвращается −1.

bcdiv

Операция деления для двух чисел произвольной точности

Описание

```
string bcdiv(string левый операнд, string правый операнд, int [масштаб]);
```

Делит левый операнд на правый операнд и возвращает результат. Параметр масштаб устанавливает количество цифр после десятичной отметки в результате.

См. также ***bcmul().***

bcmod

Получение модуля числа произвольной точности

Описание

```
string bcmod(string левый операнд, string модуль);
```

Получение модуля (остатка от деления первого параметра на второй) левого операнда, используя операнд модуль.

См. также ***bcdiv().***

bcmul

Операция умножения для двух чисел произвольной точности

Описание

```
string bcmul(string левый операнд, string правый операнд, int [масштаб]);
```

Умножает левый операнд на правый операнд и возвращает результат. Параметр масштаб устанавливает количество цифр после десятичной отметки в результате.

См. также ***bcdiv().***

bcpow

Возведение одного числа произвольной точности в степень другого

Описание

```
string bcpow(string x, string y, int [масштаб]);
```

Возведение x в степень y. Параметр масштаб может использоваться для установки количества цифр после десятичной отметки в результате.

См. также ***bcsqrt().***

bcscale

Устанавливает масштаб по умолчанию для всех математических BC-функций

Описание

```
string bcscale(int масштаб);
```

Эта функция устанавливает заданный по умолчанию параметр масштаба для всех последующих математических BC-функций, в которых не определен параметр масштаба в явном виде.

bcsqrt

Получение квадратного корня числа произвольной точности

Описание

```
string bcsqrt(string операнд, int масштаб);
```

Возвращает квадратный корень операнда. Параметр масштаб устанавливает количество цифр после десятичной отметки в результате.

См. также ***bcpow().***

bcsub

Вычитает одно число произвольной точности из другого

Описание

```
string bcsub(string левый операнд, string правый операнд, int [масштаб]);
```

Вычитает правый операнд из левого операнда и возвращает результат типа string. Параметр масштаб устанавливает количество цифр после десятичной отметки в результате.

См. также ***bcadd().***

Функции для работы с датами

Календарные функции доступны только если установлено календарное расширение возможностей PHP.

Календарное расширение в PHP представляет серию функций, которые упрощают преобразование дат между разными календарными форматами. В качестве посредника преобразования используется Юлианский календарь (пересчет дней по Юлианскому календарю). Для того, чтобы изменить числа между разными календарными системами необходимо сначала преобразовать дату в Юлианский формат, а только затем в любую другую календарную систему.

См. http://genealogy.org/~scottlee/cal-overview.html.

JDToGregorian

Преобразование юлианского счета в грегорианскую дату

Описание

```
string jdtogregorian(int julianday);
```

Преобразование юлианского счета в грегорианскую в формате «месяц/день/год».

GregorianToJD

Преобразовывает грегорианскую дату на юлианский счет

Описание

```
int gregoriantojd(int month, int day, int year);
```

Правильный диапазон для грегорианского календаря 4714 до н.э. до 9999 н.э.

Пример. Календарные функции:

```
<?php
$jd = GregorianToJD(10,11,2001);
echo("$jd\n");
$gregorian = JDToGregorian($jd);
echo("$gregorian\n");
?>
```

Этот пример показан на рис. С.2. Заметим, что посредником во всех преобразованиях даты является количество дней по юлианскому календарю, т.е. юлианский счет.

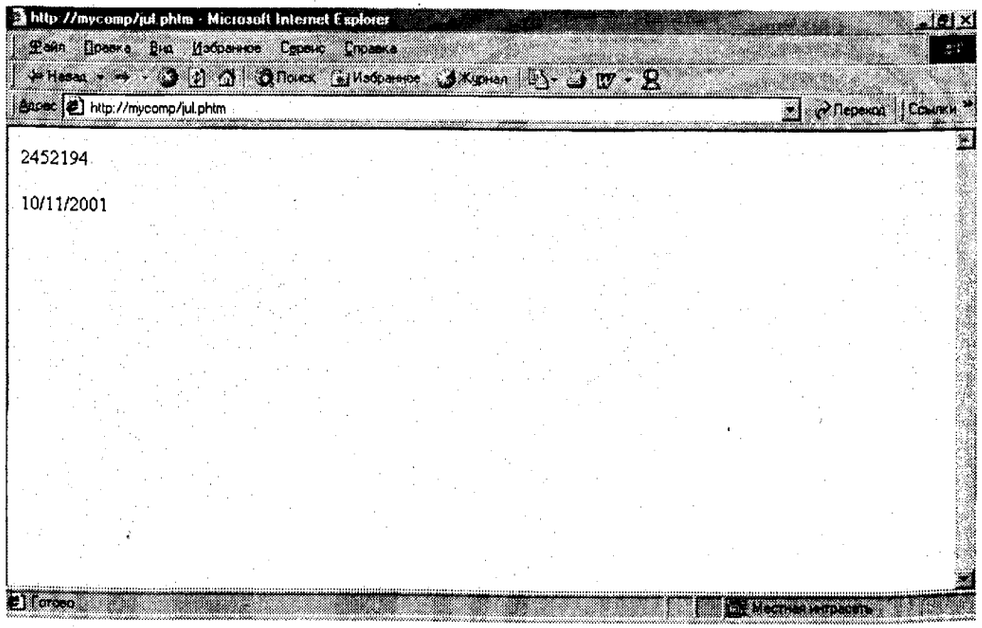


Рис. С.2. *Пример преобразования даты. Верхняя строка — юлианский счет для даты, указанной в нижней строке*

JDToJulian

Преобразовывает дату юлианского календаря на юлианский счет

Описание

```
string jdtojulian(int julianday);
```

Преобразование юлианского счета в строку, содержащую дату юлианского календаря в формате «месяц/день/год».

JulianToJD

Преобразовывает дату юлианского календаря на юлианский счет

Описание

```
int juliantojd(int month, int day, int year);
```

Правильный диапазон для юлианского календаря 4713 до н.э. до 9999 н.э.

JDToJewish

Преобразовывает юлианский счет в еврейский календарь

Описание

```
string jdtojewish(int julianday);
```

Преобразование дневного юлианского счета в еврейский календарь.

JewishToJD

Преобразовывает дату в еврейском календаре на юлианский счет

Описание

```
int jewishtojd(int month, int day, int year);
```

JDToFrench

Преобразовывает юлианский счет во французский республиканский календарь

Описание

```
string jdtofrench(int month, int day, int year);
```

FrenchToJD

Преобразовывает дату французского республиканского календаря в юлианский счет

Описание

```
int frenchtojd(int month, int day, int year);
```

JDMonthName

Возвращает название месяца

Описание

```
string jdmonthname(int julianday, int mode);
```

Значения для mode:

0 — грегорианский (краткий);
1 — грегорианский;
2 — юлианский (краткий);
3 — юлианский;
4 — еврейский;
5 — французский республиканский.

JDDayOfWeek

Возвращает день недели

Описание

```
Mixed jddayofweek(int julianday, int mode);
```

Возвращает день недели. Может возвратить string или int в зависимости от значеия параметра mode.

Значения параметра mode:

0 ..... возвращает порядковый номер дня недели в виде целого значения int (0 = воскресенье, 1 = понедельник, и т.п.);
1 ..... возвращает строку string — название дня недели (английское-грегорианское);
2 ..... возвращает строку, содержащую краткое название дня недели (английский-грегорианский).

easter_date

Получает время формата UNIX для начала Пасхи текущего года

Описание

```
int easter_date (int year)
```

Пример:

```
echo date («M-d-Y», easter_date(1999));        /* «Apr-04-1999» */
echo date («M-d-Y», easter_date(2000));        /* «Apr-23-2000» */
echo date («M-d-Y», easter_date(2001));        /* «Apr-15-2001» */
```

easter_days

Получает количество дней до пасхи после 21 марта для текущего года

Описание

```
int easter_days (int year)
```

Пример (см. рис. С.3):

```
echo easter_days (1999);          /* 14, i.e. April 4    */
echo easter_days (1492);          /* 32, i.e. April 22   */
echo easter_days (1913);          /*  2, i.e. March 23   */
```

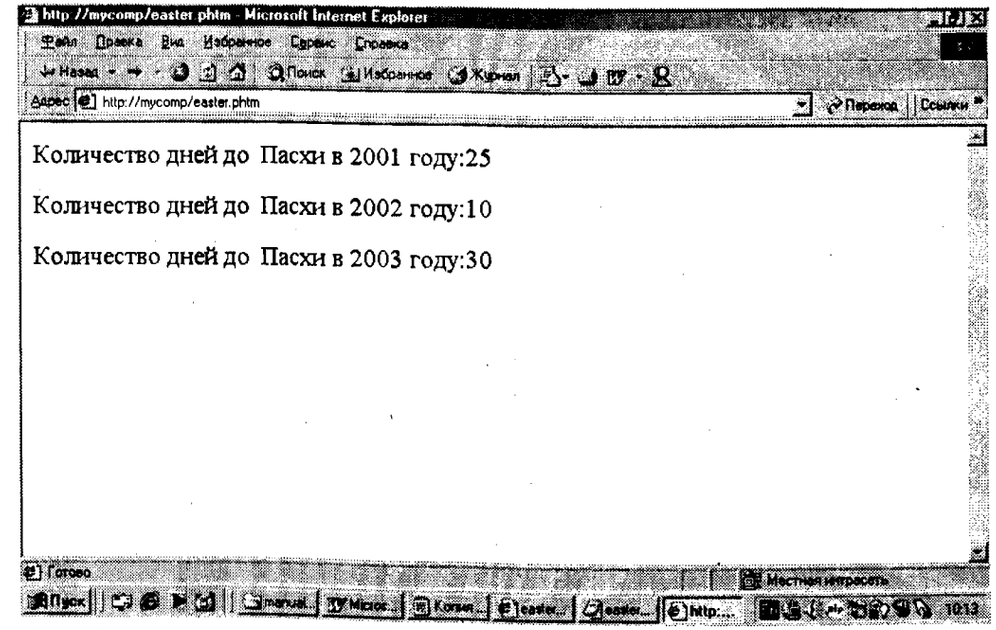


Рис. С.3. *Количество дней до Пасхи от 21 марта текущего года*

unixtojd

Преобразует формат временит UNIX в юлианские дни

Описание

```
int unixtojd ([int timestamp])
```

jdtounix

Преобразует юлианские дни в формат UNIX

Описание

```
int jdtounix (int jday)
```

Функции для работы с классами и объектами

Эти функции позволяют получать информацию, касающуюся классов и объектов, экземпляров классов. С их помозью можно получить имя класса, к которому относится тот или иной объект, а также получить информацию о его свойствах и методах. Функции позволяют также получить информацию о родительских класса.

Пример

В этом примере мы определим базовый класс и его расширение. Базовый класс будет описывать овощи, указывая их свойсва (съедобен или нет и цвет). Подкласс Spinach будет содержать метод с описанием приготовления овоща и другие методы.

Файл classes.inc

```
<?php

// базовый класс, содержащий свойства и методы
class Vegetable {

  var $edible;
  var $color;

  function Vegetable( $edible, $color="green" ) {
    $this->edible = $edible;
    $this->color = $color;
  }
```



```
  function is_edible() {
    return $this->edible;
  }

  function what_color() {
    return $this->color;
  }

} // конец базового класса

// расширение базового класса
class Spinach extends Vegetable {

  var $cooked = false;

  function Spinach() {
    $this->Vegetable(true, "green");
  }

  function cook_it() {
    $this->cooked = true;
  }

  function is_cooked() {
    return $this->cooked;
  }

} // конец класса Spinach

?>
```

Далее нам следует инициализировать два объекта этого класса и вывести информацую о них, включая информацию о наследовании. Мы определим вспомогательные функции, осуществляющие вывод.

Файл veg.php

```
<pre>
<?php
include "classes.inc";

// Вспомогательные функции
function print_vars($obj) {
```

```
  $arr = get_object_vars($obj);
  while (list($prop, $val) = each($arr))
    echo "\t$prop = $val\n";
}

function print_methods($obj) {
  $arr = get_class_methods(get_class($obj));
  foreach ($arr as $method)
    echo "\tfunction $method()\n";
}

function class_parentage($obj, $class) {
  global $$obj;
  if (is_subclass_of($$obj, $class)) {
    echo "Object $obj belongs to class ".get_class($$obj);
    echo " a subclass of $class\n";
  } else {
    echo "Object $obj does not belong to a subclass of $class\n";
  }
}

// инициализация двух объектов

$veggie = new Vegetable(true,"blue");
$leafy = new Spinach();

// ввывод информации об объектах
echo "veggie: CLASS ".get_class($veggie)."\n";
echo "leafy: CLASS ".get_class($leafy);
echo ", PARENT ".get_parent_class($leafy)."\n";

// вывод свойств veggie
echo "\nveggie: Properties\n";
print_vars($veggie);

// методы leafy
echo "\nleafy: Methods\n";
print_methods($leafy);
echo "\nParentage:\n";
class_parentage("leafy", "Spinach");
class_parentage("leafy", "Vegetable");
?>
</pre>
```


В приведенном примере важен тот факт, что объект $leafy является экземпляром класса Spinach, который является подклассом класса Vegetable, поэтому скрипт выведет такой фрагмент (рис. С.4):

```
[...]
Parentage:
Object leafy does not belong to a subclass of Spinach
Object leafy belongs to class spinach a subclass of Vegetable
```

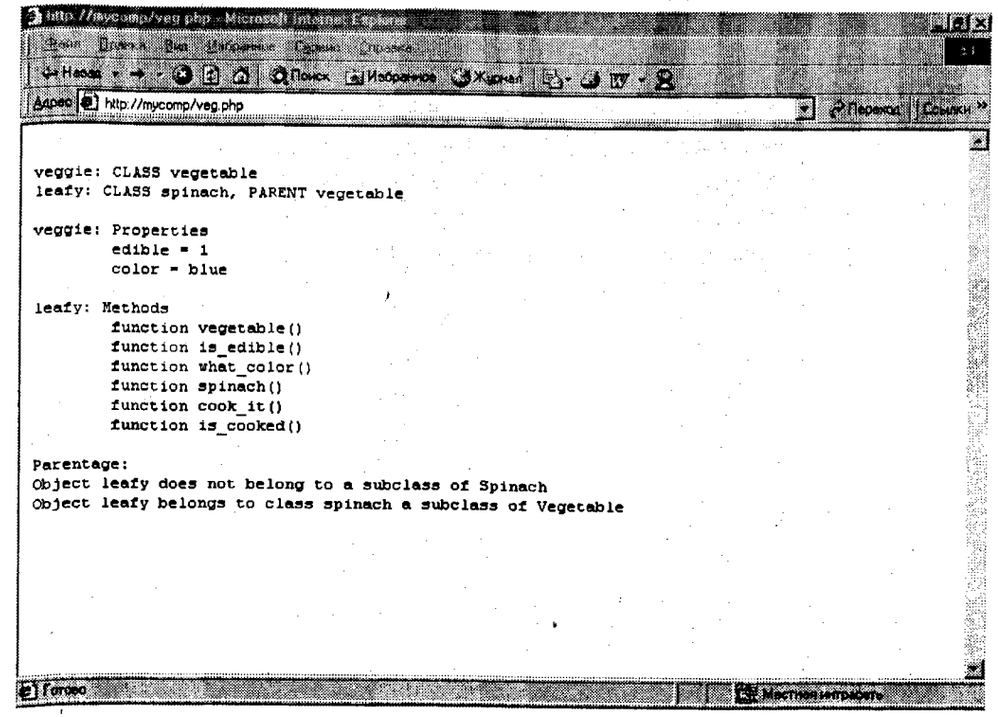


Рис. С.4. *Пример использования функций работы с классами*

ФУНКЦИИ

get_class

Возвращает имя класса объекта

Описание

```
string get_class (object obj)
```

См. также ***get_parent_class(), is_subclass_of().***

get_parent_class

Возвращает имя родительского класса

Описание

```
string get_parent_class (object obj)
```

См. также ***get_class(), is_subclass_of().***

get_class_methods

Возвращает массив имен методов класса

Описание

```
array get_class_methods (string class_name)
```

См. также ***get_class_vars(), get_object_vars().***

get_class_vars

Возвращает массив свойств класса

Описание

```
array get_class_vars (string class_name)
```

См. также ***get_class_methods(), get_object_vars().***

get_object_vars

Возвращает ассоциативный массив свойств объекта

Описание

```
array get_object_vars (object obj)
```

См. также ***get_class_methods(), get_class_vars().***

is_subclass_of

Определяет, является ли объект подклассом указанного класса

Описание

```
bool is_subclass_of (object obj, string superclass)
```

Функция возвращает true в том случае, если объект obj, относится к классу superclass. Если объект не принадлежит указанному классу, то функция возвращает false.

См. также ***get_class(), get_parent_class().***

class_exists

Функция находит, был ли класс определен

Описание

```
bool class_exists (string class_name)
```

Функция возвращает true, если класс class_name был описан, в противном случае возвращает false.

method_exists

Функция проверяет наличие указанного метода

Описание

```
bool method_exists (object object, string method_name)
```

Функция возвращает true, если метод method_name определен в указанном объекте object, если метод не определен, то возвращает значение false.

get_declared_classes

Возвращает массив с именами описанных классов

Описание

```
array get_declared_classes (void)
```

Выводит имя всех классов текущего скрипта. В PHP4 всегда существуют классы stdClass (описан в Zend/zend.c), OverloadedTestClass (описан в ext/standard/basic_functions.c) и Directory (описан in ext/standard/dir.c).

call_user_method

Вызывает метод указанного класса

Описание

```
mixed call_user_method (string method_name, object obj [, mixed
parameter [, mixed...]])
```

Вызываемый метод — это метод method_name в объекте obj.

Пример:

```
<?php
class Country {
  var $NAME;
  var $TLD;

  function Country($name, $tld) {
    $this->NAME = $name;
    $this->TLD = $tld;
  }

  function print_info($prestr="") {
    echo $prestr."Country: ".$this->NAME."\n";
    echo $prestr."Top Level Domain: ".$this->TLD."\n";
  }
}

$cntry = new Country("Peru","pe");

echo "* Calling the object method directly\n»;
$cntry->print_info();

echo "\n* Calling the same method indirectly\n»;
call_user_method ("print_info", $cntry, "\t");
?>
```

В броузере будет показана информация в соответсвии с рис. С.5.

См. также call_user_func().

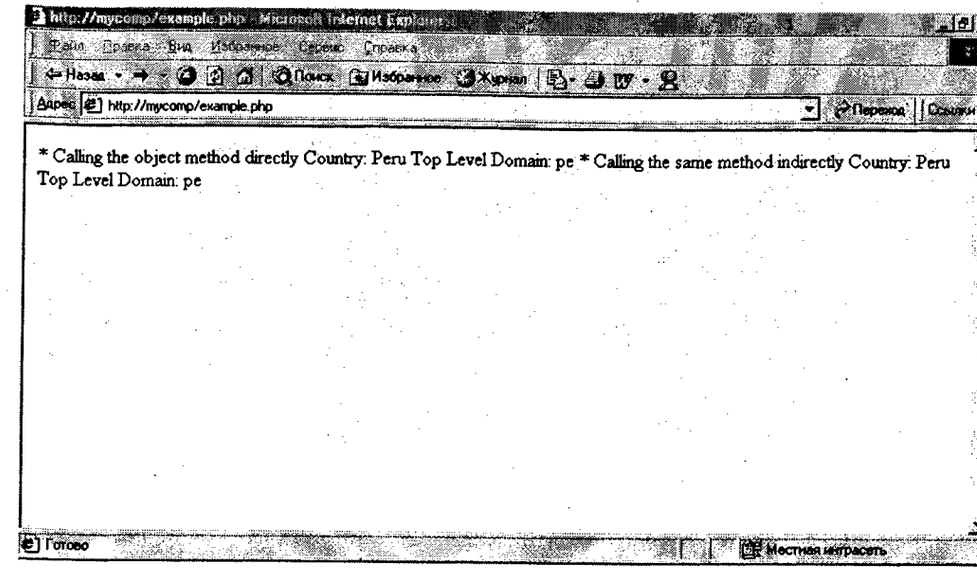


Рис. С.5. Вызов методов объекта

Функции работы с датой и временем

checkdate

Проверяет правильность даты/времени

Описание

```
int checkdate (int month, int day, int year);
```

Возвращает true, если данная дата верна, в противном случае возвращает false. Дата будет верной в том случае, если:

- год находится между 1900 и 32767 (включительно);
- месяц — между 1 и 12 включительно;
- день находится в диапазоне разрешенных дней данного месяца, високосные годы учитываются.

date

Формат локального времени/даты

Описание

```
string date (string format, int timestamp);
```

Возвращает строку, отформатированную в соответствии с форматом.

Форматы:

A..... «AM» или «PM»;
d ..... день месяца, цифровой, 2 цифры (на первом месте ноль);
D .... день недели, текстовый, 3 буквы; т.е. «Fri»;
F ..... месяц, текстовый, длинный; т.е. «January»;
h ..... час, цифровой, 12-часовой формат;
H .... час, цифровой, 24-часовой формат;
i ..... минуты, цифровой;
j ..... день месяца, цифровой, без начальных нулей;
l ..... (строчная 'L') день недели, текстовый, длинный; т.е. «Friday»;
m .... месяц, цифровой;
M .... месяц, текстовый, 3 буквы, т.е. «Jan»;
s ..... секунды, цифровой;
S ..... английский порядковый суффикс, текстовый, 2 символа; т.е. «th», «nd»;
U .... секунды с начала века;
Y ..... год, цифровой, 4 цифры;
w ..... день недели, цифровой, 0 означает воскресенье;
y..... год, цифровой, 2 цифры;
z..... день года, цифровой, т.е. «299».

Нераспознанные символы в форматной строке будут печататься как есть.

Пример date():

```
print(date( "l dS of F Y h:i:s A" ));
print("July 1, 2000 is on a ". date("l", mktime(0,0,0,7,1,2000)));
```

На экран будет выведена информация, показанная на рис. С.6.

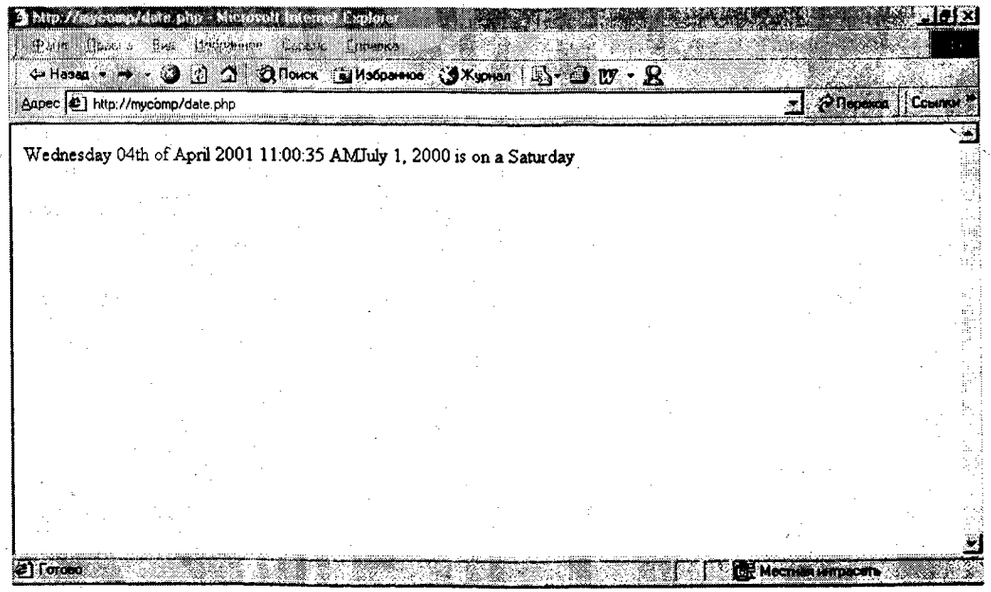


Рис. С.6. *Работа с датой*

Функции ***date()*** и ***mktime()*** возможно использовать совместно для определения даты в будущем или прошлом.

Пример функций date() и mktime():

```
$tomorrow  = mktime(0,0,0,date("m")  ,date("d")+1,date("Y"));
$lastmonth = mktime(0,0,0,date("m")-1,date("d"),date("Y"));
$nextyear  = mktime(0,0,0,date("m"),date("d"),date("Y")+1);
```

Для того, чтобы перевести даты в другие календарные системы, используются функции ***setlocale()*** и ***strftime()***.

См. также ***gmdate(), mktime().***

strftime

Форматирует локальное время

Описание

```
string strftime (string format, int timestamp);
```

Возвращает строку, отформатированную согласно заданному формату:
%a ..... сокращенное название дня недели согласно текущему locale;
%A..... полное название дня недели согласно текущему locale;
%b ..... сокращенное название месяца согласно текущему locale;
%B..... полное название месяца согласно текущему locale;
%c ..... предпочтительное представление даты и времени для текущего locale;
%d ..... день месяца как десятичное число (в диапазоне от 0 до 31);
%H ..... час как десятичное число в 24-часовом формате (в диапазоне от 00 до 23);
%I..... час как десятичное число в 12-часовом формате (в диапазоне от 01 до 12);
%j ..... день года как десятичное число (в диапазоне от 001 до 366);
%m ..... месяц как десятичное число (в диапазоне от 1 до 12);
%M..... минуты как десятичное число;
%p ..... 'am' или 'pm' согласно текущему времени, или соответствующие строки для текущего locale;
%S ..... секунды как десятичное число;
%U ..... номер недели текущего года как десятичное число, начиная с первого воскресенья в качестве первого дня первой недели;
%W..... номер недели текущего года как десятичное число, ночиная с первого понедельника в качестве первого дня первой недели;
%w..... день недели как целое число, воскресенье — 0-й день;
%x ..... предпочитаемое представление даты для текущего locale, не включающее время;

%X..... предпочитаемое представление времени для текущего locale, не включающее дату;
%y ..... год как десятичное число без столетия (в диапазоне от 00 до 99);
%Y..... год как десятичное число, включая столетие;
%Z..... временная зона или название или сокращение;
%% ..... символ '%'.

Пример функции strftime():

```
setlocale ("LC_TIME", "C");
print(strftime("%A in Finnish is "));
setlocale ("LC_TIME", "fi");
print(strftime("%A, in French "));
setlocale ("LC_TIME", "fr");
print(strftime("%A and in German "));
setlocale ("LC_TIME", "de");
print(strftime("%A.\n"));
```

См. также ***setlocale(), mktime().***

getdate

Получает информацию о дате/времени

Описание

```
array getdate (int timestamp);
```

Возвращает ассоциативный массив, содержащий информацию о дате, со следующими элементами:
«seconds»..... секунды;
«minutes» ..... минуты;
«hours» ..... часы;
«mday» ..... день месяца;
«wday» ..... день недели, цифровой;
«mon» ..... месяц, цифровой;
«year» ..... год, цифровой;
«yday» ..... день года, цифровой, т.е. «299»;
«weekday»..... день недели, текстовый, полный, т.е. «Friday»;
«month»..... месяц, текстовый, полный, т.е. «January».

gmdate

Форматирует GMT/CUT время/дату

Описание

```
string gmdate (string format, int timestamp);
```

Аналогична функции ***date()*** за исключением того, что время возвращается в гринвичском формате Greenwich Mean Time (GMT). Например, при запуске в России (GMT +0300), первая строка ниже напечатает «Jan 01 1998 00:00:00», в то время как вторая строка напечатает «Dec 31 1997 22:00:00» (см. рис. С.7).

Пример gmdate():

```
echo date( "M d Y H:i:s",mktime(0,0,0,1,1,1998) );
echo gmdate( "M d Y H:i:s",mktime(0,0,0,1,1,1998) );
```

См. также ***date(), mktime(), gmmktime().***

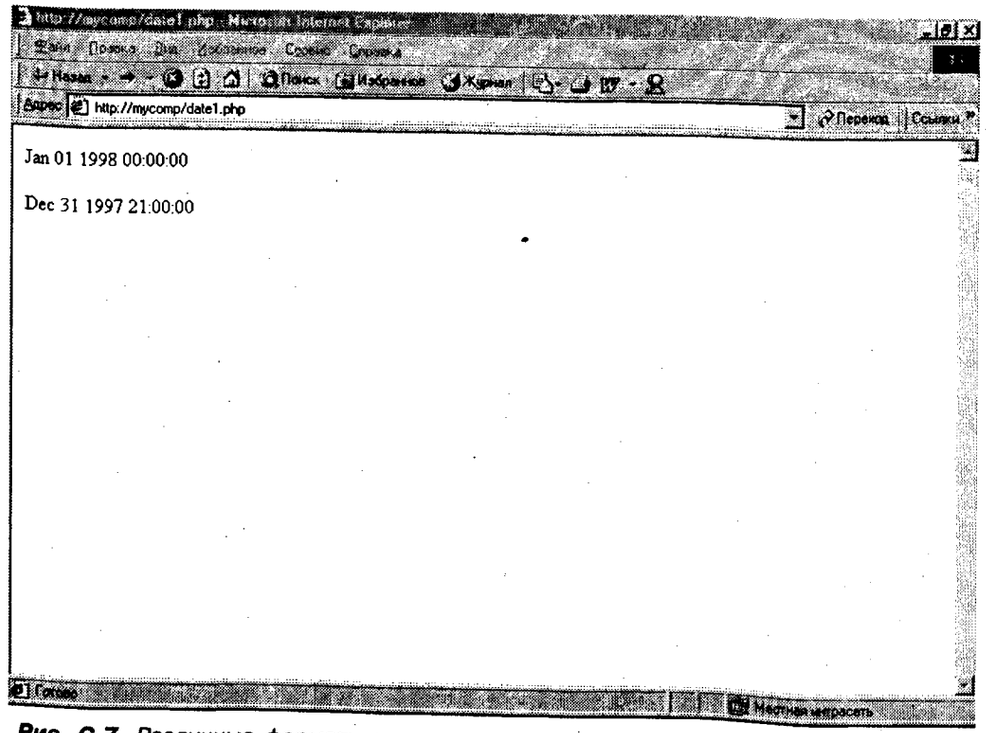


Рис. С.7. *Различные форматы времени*

mktime

Получает временную метку UNIX для даты

Описание

```
int mktime (int hour, int minute, int second, int month, int day,
int year);
```

Пример mktime():

```
echo date( "M-d-Y", mktime(0,0,0,12,32,1997) );
echo date( "M-d-Y", mktime(0,0,0,13,1,1997) );
echo date( "M-d-Y", mktime(0,0,0,1,1,1998) );
```

См. также ***date(), time().***

gmmktime

Получает временную метку UNIX для даты в GMT

Описание

```
int gmmktime (int hour, int minute, int second, int month, int day,
int year);
```

time

Возвращает текущую временную метку UNIX

Описание

```
int time (void);
```

Возвращает текущее время, измеренное в числе секунд с эпохи Unix (1 января 1970 00:00:00 GMT).

См. также ***date().***

microtime

Возвращает текущую временную метку UNIX в микросекундах

Описание

```
string microtime (void);
```

Возвращает строку «msec sec», где sec — текущее время, измеренное в числе секунд с эпохи Unix (0:00:00 1 января, 1970 GMT), а msec — это часть в микросекундах. Эти функции доступны только в операционных системах, поддерживающих системный вызов ***gettimeofday().***

См. также ***time().***

gettimeofday

Получить текущее время

Описание

```
array gettimeofday (void)
```

Возвращает ассоциативный массив:
«sec» ..... секунды;
«usec» ..... микросекунды;
«minuteswest» ..... минуты;
«dsttime» ..... тип корректировки dst.

gmdate

Форматировать дату/время

Описание

```
string gmdate (string format, int timestamp)
```

gmstrftime

Форматировать дату/время GMT/CUT в соответствии с локальными установками

Описание

```
string gmstrftime (string format, int timestamp)
```

localtime

Получить местное время

Описание

```
array localtime ([int timestamp [, bool is_associative]])
```

Возвращает массив (ассоциативный, если is_associative равен 1):
«tm_sec» ..... секунды;
«tm_min» ..... минуты;
«tm_hour» ..... часы;
«tm_mday» ..... дата (день месяца);
«tm_mon» ..... месяц;
«tm_year» ..... год;
«tm_wday» ..... день недели;
«tm_yday» ..... день года;
«tm_isdst» ..... включена ли поддержка дневного времени.

microtime

Текущее время UNIX с микросекундами

Описание

```
string microtime(void);
```

time

Возвращает текущее время UNIX

Описание

```
int time(void);
```

strtotime

Анализирует формат текстового времени и переводит его во время UNIX

Описание

```
string strftime (string format [, int timestamp])
```

%a .....день недели кратко;
%A.....день недели полностью;
%b .....месяц кратко;
%B.....месяц полностью;
%c .....дата и время;
%C .....номер века;
%d .....день месяца;
%D .....эквивалентно %m/%d/%y;
%e .....день месяца (от '1' до '31');
%h .....эквивалентно %b;
%H .....часы (до 24);
%I.....часы (до 12);
%j .....день года (366);
%m .....месяц (от 1 до 12);
%M.....минуты;
%n .....переход на новую строку;
%p .....обозначение времени суток 'am' или 'pm' the current locale;
%r.....время с обозначениями a.m. или p.m.;
%R .....время в обозначении до 24 часов;
%S .....секунды;
%t .....символ табуляции;
%T.....текущее время, эквивалентно %H:%M:%S;
%u .....дни недели в числовом представлении, 1 — понедельник;

%U .....номер недели в году;
%V.....номер недели в году (ISO 8601:1988), первой неделей считается неделя, содержащая по крайней мере 4 дня, пришедшиеся на текущий год. Диапазон от 01 до 53;
%W.....номер недели, первая неделя та, на которую проходится первый понедельник года;
%w.....день недели, воскресенье 0;
%x .....дата без времени;
%X.....время без даты;
%y .....год без века;
%Y.....год (полный, с веком);
%Z.....временная зона;
%% .....символ '%'.

Пример:

```
setlocale ("LC_TIME", "C");
print (strftime ("%A in Finnish is "));
setlocale ("LC_TIME", "fi_FI");
print (strftime ("%A, in French "));
setlocale ("LC_TIME", "fr_CA");
print (strftime ("%A and in German "));
setlocale ("LC_TIME", "de_DE");
print (strftime ("%A.\n"));
```

Функции для работы с каталогами

chdir

Смена каталога

Описание

```
int chdir(string directory);
```

Изменяет текущий PHP-каталог на новый каталог с именем directory. Возвращает FALSE, если текйщий каталог изменить не удалось, TRUE — если изменение каталога произведено.

dir

Читает содержание текущего каталога

Описание

```
new dir(string directory);
```

Пример с Dir():

```
$d = dir("/etc");
echo "Handle: ".$d->handle."<br>\n";
echo "Path: ".$d->path."<br>\n";
while($entry=$d->read()) {
     echo $entry."<br>\n";
}
$d->close();
```

closedir

Закрыть каталог

Описание

```
void closedir(int dir_handle);
```

Закрывает поток каталога, обозначенный как dir_handle. Поток предварительно должен быть открыт финкцией ***opendir()***.

opendir

Открыть каталог

Описание

```
int opendir(string path);
```

Возвращает дескриптор (handle) каталога, который в последующем используется в ***closedir()***, ***readdir()***, и ***rewinddir()*** обращениях.

readdir

Чтение данных из каталога по дескриптору(handle)

Описание

```
string readdir(int dir_handle);
```

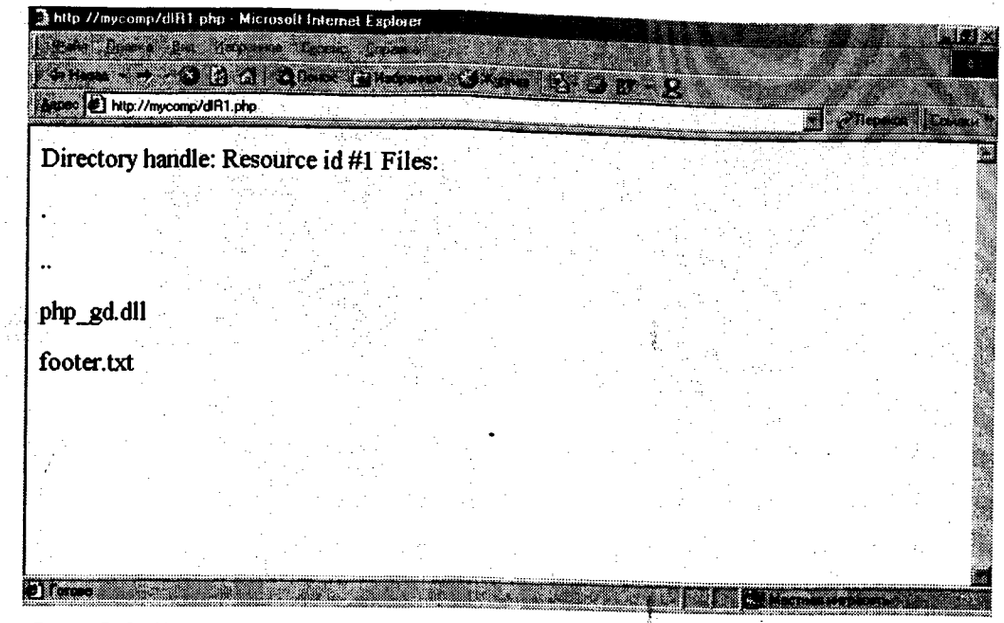



Рис. С.8. *Чтение каталога*

Пример. Вывод всех файлов в текущем каталоге *(см рис. С.8):*

```
<?php
     chdir("dll");
     $handle=opendir('.');
     echo "Directory handle: $handle\n";
     echo "Files:<P>";
     while ($file = readdir($handle)) {
          echo "$file<p>";
     }
     closedir($handle);
?>
```

rewinddir

Возврат к началу данных каталога по дескриптору(handle)

Описание

```
void rewinddir(int dir_handle);
```

Очищает поток каталога, обозначенный как dir_handle, возвращая в начало данных.

Функции XML/DOM

Эти функции доступны только тогда, когда PHP сконфигурирован с поддержкой библиотеки GNOME XML. Для этого необходимо включить в файл pbp.ini строку extension=php_domxml.dll.

В этом модуле определен целый ряд классов. Функции DOM XML производят анализ XML-файла и возвращают дерево его объектов с учетом их иерархии.

Функции

xmldoc

Создает объект DOM в XML-документе

Описание

```
object xmldoc (string str)
```

Эта функция анализирует документ XML, указанный в качестве аргумента str, и возвращает объект класса «Dom document», обладающий свойствами «doc» (resource), «version» (string) и «type» (long).

xmldocfile

Создает объект DOM из XML файла

```
object xmldocfile (string filename)
```

Функция осуществляет анализ XML-файла с именем filename и возвращает объект класса «Dom document», со свойствами «doc» (resource), «version» (string).

xmltree

Строит дерево объектов PHP на основе документа XML

```
object xmltree (string str)
```

Функция анализирует документ XML, указанный в str, и возвращает дерево объектов PHP в том виде, как они описаны в XML-документе.

Функции для работы с файловой системой

basename

Возвращает имя файла по полному пути к файлу

Описание

```
string basename(string path);
```

Получив строку, содержащую путь к файлу, данная функция возвратит базовое имя файла.

В Windows можно использовать оба вида знака черты — нормальную черту (/) и обратную черту (\). Они используются как разделители при задании пути. В других средах можно использовать только нормальную черту (/).

Пример basename():

```
$path = "/home/httpd/html/index.php";
$file = basename($path); // $file устанавливается в "index.php"
```

См. также dirname()

chgrp

Изменить файловую группу, т.е. принадлежность файла

Описание

```
int chgrp(string filename, mixed group);
```

Пытается изменить группу файла filename на group.

Возвращает true при успешном завершении, в противном случае — false.

В Windows при выполнении этой функции ничего не происходит и возвращается true.

См. также chown(), chmod().

chmod

Изменить режим файла

Описание

```
int chmod(string filename, int mode);
```

Пытается изменить режим файла, указанного в filename на новый режим mode.

Заметим, что mode не присваивает автоматически восьмиричное значение, для этого необходимо использовать префикс mode с нулем (0):

```
chmod( "/somedir/somefile", 755 );
// десятичный; возможно появление ошибки
chmod( "/somedir/somefile", 0755 );
// восьмиричный; корректное значение режима
```

Возвращает true при успешном завершении, в противном случае — false.

*См. также **chown(), chgrp().***

chown

Изменяет владельца файла

Описание

```
int chown(string filename, mixed user);
```

Пытается изменить владельца файла filename на пользователя user. Возвращает true при успешном завершении, в противном случае — false. В Windows ничего не выполняется и возвращается true.

*См. также **chown(), chmod().***

clearstatcache

Очистить кеш статистики файла

Описание

```
void clearstatcache(void);
```

copy

Скопировать файл

Описание

```
int copy(string source, string dest);
```

Создает копию файла. Возвращает true при успешном завершении, в противном случае — false.

Пример copy():

```
if (!copy($file, $file.'.bak')) {
    print("failed to copy $file...<br>\n");
}
```

*См. также **rename().***

dirname

Путь к файлу

Описание

```
string dirname(string path);
```

По строке, содержащей путь к файлу возвращает каталог, содержащий этот файл.

Пример dirname():

```
$path = "/etc/passwd";
$file = dirname($path); // $file is set to "/etc"
```

*См. также **basename()***

fclose

Закрывает открытый указатель на файл

Описание

```
int fclose(int fp);
```

Указатель на файл fp закрывается. Возвращает true при удачной операции и false при ошибке. Указатель должен быть открыт и указывать на файл, открыть который можно с помощью ***fopen()*** или ***fsockopen()***.

feof

Проверка на достижение указателем конца файла

Описание

```
int feof(int fp);
```

Возвращает true, если указатель файла равен EOF (или в случае ошибки), в противном случае возвращается false.

Указатель должен быть предварительно открыт с помощью ***fopen(), popen()*** или ***fsockopen()***.

fgetc

Получить символ из файла

Описание

```
string fgetc(int fp);
```

Возвращает строку, содержащую один символ, прочитанный по файловому указателю fp. При EOF возвращается false. Указатель должен быть открыт с помощью ***fopen(), popen(),*** или ***fsockopen()***.

*См. также **fopen(), popen(), fsockopen(), fgets().***

fgets

Получить строку по указателю на файл

Описание

```
string fgets(int fp, int length);
```

Возвращает строку длиной до length, читается по одному байту из файла, указанного в fp. Чтение заканчивается, если прочитано число символов, равное length, чтение прекращается также в случае обнаруженеия символа конца строки, возврата каретки или конца файла EOF. При ошибке возвращается false. Указатель должен быть открыт с помощью *fopen()*, *popen()* или *fsockopen()*.

См. также ***fopen(), popen(), fgetc(), fsockopen()***.

fgetss

Получает строку по указателю файла и убирает из нее ярлыки HTML

Описание

```
string fgetss(int fp, int length);
```

Отличается от *fgetss()* тем, что удаляет ярлыки HTML и PHP-тэги из прочитанного текста.

См. также ***fgets(), fopen(), fsockopen(), popen()***.

file

Читает файл в массив

Описание

```
array file(string filename);
```

Идентична *readfile()*, отличие состоит в том, что *file()* выводит массив. Каждый элемент массива соответствует строке файла (вместе с символом возврата строки).

См. также ***readfile(), fopen(), popen()***.

file_exists

Проверяет существование файла

Описание

```
int file_exists(string filename);
```

Возвращает true, если файл, указанный в filename существует; в противном случае возвращает false.

См. также ***clearstatcache()***.

fileatime

Время последнего обращения к файлу

Описание

```
int fileatime(string filename);
```

Возвращает время последнего обращения к файлу, в случае возникновения ошибки возвращает false.

filectime

Время последнего изменения файла

Описание

```
int filectime(string filename);
```

Возвращает время последего изменения файла, или false в случае возникновения ошибки.

filegroup

Группа файла

Описание

```
int filegroup(string filename);
```

Возвращает идентификатор ID группы владельца файла, в влучае возникновения ошибки возвращает false.

fileinode

fileinode — inode файла

Описание

```
int fileinode(string filename);
```

Возвращает номер inode файла, или false в случае ошибки.

filemtime

Время модификации файла

Описание

```
int filemtime(string filename);
```

Возвращается время последего изменения файла, или false в случае ошибки.

fileowner

Владелец файла

Описание

```
int fileowner(string filename);
```

Возвращает ID владельца файла, или false в случае ошибки.

fileperms

Ограничения использования файла

Описание

```
int fileperms(string filename);
```

Возвращает разрешения, установленные для файла, или false в случае ошибки.

filesize

Размер файла

Описание

```
int filesize(string filename);
```

Возвращает размер файла, или false в случае ошибки.

filetype

Тип файла

Описание

```
string filetype(string filename);
```

Возвращает тип файла.

Возможные значения:

- fifo;
- char;
- dir;
- block;
- link;
- file;
- unknown.

В случае ошибки возвращается false.

fopen

Открыть файл или URL

Описание

```
int fopen(string filename, string mode);
```

Если filename начинается с указания названия протокола «http://» (без учета регистра), открывается соединение HTTP 1.0 с указанным сервером и возвращается указатель файла на начало текста ответа. Поскольку переадресация в HTTP не обрабатывается, то необходимо вставить в указание каталога завершающие слэши.

Если filename начинается с уазания названия протокола «ftp://» (без учета регистра), то откроется ftp-соединение с указанным сервером и возвращается указатель на искомый файл. Если сервер не поддерживает режим пассивного ftp, то данная операция завершится ошибкой. Существует возможность открытия файлов через ftp как для чтения, так и для записи (но не обновременно).

Если filename начинается иначе, то откравается файл файловой системы, и возвращается указатель на открытый файл.

Если при открытии файла происходит ошибка, функция возвращает false.

Mode может принимать такие значения:

"r" ..... открыть только для чтения, помещает указатель на начало файла;

"r+" ... открыть для чтения и для записи, помещает указатель на начало файла;

"w" ..... открыть только для записи, помещает указатель на начало файла и очищает все содержимое файла. Если файл не существует, создается новый файл;

"w+" .. открыть для чтения и для записи, помещает указатель на начало файла и очищает все содержимое файла. Если файл не существует, создается новый файл;

"a" ..... открыть только для записи, помещает указатель на конец файла. Если файл не сществует, создается новый файл;

"a+" ... открыть для чтения и для записи, помещает указатель на конец файла. Если файл не существует, создается новый файл.

Mode может содержать символ 'b' для систем, различающих бинарные и текстовые файлы (не используется в Unix).

Пример fopen():

```
$fp = fopen("/home/rasmus/file.txt", "r");
$fp = fopen("http://www.php.net/", "r");
$fp = fopen("ftp://user:password@example.com/", "w");
```

Если вы испытываете проблемы с чтением и записью в файл, используя PHP как серверный модуль, то необходимо убедиться в том, что файлы и каталоги доступны для серверных процессов.

*См. также **fclose(), fsockopen(), popen().***

fpassthru

Вывод всех данных из указателя файла

Описание

```
int fpassthru(int fp);
```

Осуществляет чтение до конца файла EOF по полученному указателю файла и записывает результат на стандартное устройство вывода. При возникновении ошибки ***fpassthru()*** возвращает false.

Файловый указатель должен быть открыт при помощи ***fopen()*** ***popen()***, или ***fsockopen()***. Для получения вывода файла целиком на stdout можно использовать функцию ***readfile()***, при этом будет сэкономлен один ресурс, так как не потребуется открывать файл с помощью ***fopen()***.

*См. также **readfile(), fopen(), popen(), fsockopen().***

fputs

Запись в файл

Описание

```
int fputs(int fp, string str, int [length]);
```

Fputs() — это псевдоним ***fwrite()***, и обе функции полностью идентичны. Заметим, что параметр length не является обязательным и при его отсутствии записывается вся строка str.

fread

Бинарное чтение файла

Описание

```
string fread(int fp, int length);
```

Fread() читает байты из файла, на который ссылается fp длиной не более length. Чтение заканчивается, когда прочитан length-байт или достигнут конец файла EOF.

```
// получить содержимое файла в строку
$filename = "/usr/local/something.txt";
$fd = fopen( $filename, "r" );
$contents = fread( $fd, filesize( $filename ) );
fclose( $fd );
```

*См. также **fwrite(), fopen(), fsockopen(), popen(), fgets(), fgetss(), file(), fpassthru().***

fseek

Поиск в файле

Описание

```
int fseek(int fp, int offset);
```

Для файла fp будет произведена установка внутреннего указателя с отступом на offset-байт. Эквивалентно вызову fseek(fp, offset, SEEK_SET) в языке C. При удачном выполнении возвращает 0, в противном случае возвращается −1, при этом поиск после EOF не рассматривается как ошибка. Не используется для файловых указателей, возвращенных функцией ***fopen()*** при использовании форматов «http://» или «ftp://».

*См. также **ftell(), rewind().***

ftell

Текущая позиция указателя в файле

Описание

```
int ftell(int fp);
```

Возвращает позицию указателя в файле, на который ссылается fp, т.е. смещение в потоке файла. При возникновении ошибки возвращается false. Файловый указатель должен быть открыт при помощи ***fopen()*** или ***popen()***.

*См. также **fopen(), popen(), fseek(), rewind().***

fwrite

Бинарная запись в файл

Описание

```
int fwrite(int fp, string string, int [length]);
```

Fwrite() записывает содержимое string в файловый поток, указанный fp. Если аргумент length присутствует, запись останавливается после записи байта с номером length, или после записи всей строки string. Заметим, что если есть аргумент length, то конфигурационные опции magic_quotes_runtime игнорируются и никакие знаки из string не удаляются.

*См. также **fread(), fopen(), fsockopen(), popen(), fputs().***

is_dir

Проверка каталога

Описание

```
bool is_dir(string filename);
```

Возвращает true, если filename существует и является каталогом.

См. также ***is_file(), is_link().***

is_executable

Файл относится к классу исполняемых

Описание

```
bool is_executable(string filename);
```

Возвращает true, если filename существует и является исполняемым файлом.

См. также ***is_file(), is_link().***

is_file

Проверка файла

Описание

```
bool is_file(string filename);
```

Возвращает true, если filename существует и является файлом.

См. также ***is_dir(), is_link().***

is_link

Проверка файла ссылок

Описание

```
bool is_link(string filename);
```

Возвращает true, если filename существует и является ссылкой.

См. также ***is_dir(), is_file().***

is_readable

Проверка читаемых классов

Описание

```
bool is_readable(string filename);
```

Возвращает true, если filename существует и является доступным для чтения.

См. также ***is_writeable().***

is_writeable

Проверка записываемых файлов (тех, которые можно изменить)

Описание

```
bool is_readable(string filename);
```

Возвращает true, если файл существует и доступен для записи.

См. также ***is_readable().***

link

Создать ссылку

Описание

```
int link(string target, string link);
```

Link() создает ссылку.

См. также ***symlink(), readlink(), linkinfo().***

linkinfo

Информация о ссылке

Описание

```
int linkinfo(string path);
```

Linkinfo() возвращает поле st_dev из UNIX C-структуры stat, возвращенной системным вызовом lstat. Эта функция используется для проверки того, существует ли ссылка. Возвращает 0 или FALSE в случае ошибки.

См. также ***symlink(), link(), readlink().***

mkdir

Создать каталог

Описание

```
int mkdir(string pathname, int mode);
```

Создает каталог, указанный в pathname.

```
mkdir("/path/to/my/dir", 0700);
```

Возвращает true при успешном выполнении и false при ошибке.

*См. также **rmdir()**.*

pclose

Закрывает файловый указатель

Описание

```
int pclose(int fp);
```

Закрывает файловый указатель, открытый с помощью ***popen()***.

*См. также **popen()**.*

popen

Открыть файловый указатель

Описание

```
int popen(string command, string mode);
```

Возвращает файловый указатель, идентичный возвращаемому функцией ***fopen()***, однако этот указатель может использоваться только для чтения или только для записи и в дальнейшем должен быть закрыт с помощью функции ***pclose()***. Этот указатель можно использовать с ***fgets()***, ***fgetss()*** и ***fputs()***. При возникновении ошибки возвращает false.

```
$fp = popen( "/bin/ls", "r" );
```

*См. также **pclose()**.*

readfile

Вывод файла

Описание

```
int readfile(string filename);
```

Читает файл и записывает его на стандартное устройство вывода. Возвращает количество прочитанных байтов. В случае возникновения ошибки возвращается false.

Если filename начинается с «http://» (без учета регистра), открывается соединение HTTP 1.0 к указанному серверу и текст ответа выводится на стандартное устройство вывода.

Если filename начинается с «ftp://» (без учета регистра), открывается ftp-соединение с указанным сервером и файл ответа выводится на стандартное устройство вывода. Если сервер не поддерживает режим пассивного ftp, этот вызов завершится ошибкой.

Если filename начинается иначе, то будет открыт файл файловой системы и его ссодержимое выведется на стандартное устройство вывода.

*См. также **fpassthru()**, **file()**, **fopen()**, **include()**, **require()**, **virtual()**.*

readlink

Цель символической ссылки

Описание

```
string readlink(string path);
```

Readlink() работает аналогично функции языка C readlink и возвращает содержимое символической ссылки path или 0 в случае ошибки.

*См. также **symlink()**, **readlink()**, **linkinfo()**.*

rename

Переименовать файл

Описание

```
int rename(string oldname, string newname);
```

Пытается переименовать oldname в newname. Возвращает true при успешном выполнениии и false при сбое.

rewind

Обновляет положение внутреннего указателя, возвращая его в начало файла

Описание

```
int rewind(int fp);
```

Позиционирует файловый указатель для fp на начало потока файла. При возникновении ошибки возвращается 0. Файловый указатель должен быть открыт при помощи функции ***fopen()***.

*См. также **fseek()**, **ftell()**.*

rmdir

Удалить каталог

Описание

```
int rmdir(string dirname);
```

Пытается удалить каталог. Каталог должен быть пустым. При возникновении ошибки возвращается 0.

См. также ***mkdir()***.

stat

Информация о файле

Описание

```
array stat(string filename);
```

Возвращает сведения о файле filename. Возвращает массив статистической информации о файле со следующими элементами:

1) устройство;
2) inode;
3) номер ссылки;
4) id пользователя или владельца;
5) id группы владельцев;
6) тип устройства inode *;
7) размер в байтах;
8) время последнего доступа;
9) время последней модификации;
10) время последнего изменения;
11) размер блока для I/O файловой системы *;
12) количество занятых блоков.

* — только для систем, поддерживающих тип st_blksize, в других системах (т.е. Windows) возвращается −1.

lstat

Информация о файле или символической ссылке

Описание

```
array lstat(string filename);
```

Возвращает информацию о файле или символической ссылке filename. Эта функция идентична ***stat()***, но если filename — это символическая ссылка, то возвращается статус символической ссылки, а не статус файла, на который указывает данная ссылка.

Возвращает массив статистической информации со следующими элементами:

1) устройство;
2) inode;
3) число ссылок;
4) id пользователя или владельца;
5) id группы владельцев;
6) тип устройства, если inode — устройство *;
7) размер в байтах;
8) время последнего доступа;
9) время последней модификации;
10) время последнего изменения;
11) размер болка для I/O файловой системы *;
12) количество занятых блоков.

* — только для систем, поддерживающих тип st_blksize, в других системах (т.е. Windows) возвращается −1.

symlink

Создать символическую ссылку

Описание

```
int symlink(string target, string link);
```

Symlink() создает символическую ссылку, указывающую на существующий ресурс target с именем link.

См. также ***link(), readlink(), linkinfo()***.

tempnam

Создать имя файла

Описание

```
string tempnam(string dir, string prefix);
```

Создает уникальное имя файла в указанном каталоге. Если каталог не существует, ***tempnam()*** может сгенерировать имя файла во временном каталоге системы.

Возвращает новое временное имя файла, или нулевую строку при ошибке.

Пример tempnam():

```
$tmpfname = tempnam( "/tmp", "FOO" );
```

touch

Установить время модификации файла

Описание

```
int touch(string filename, int time);
```

Пытается установить время модификации файла filename в виде значения time. Если опция time отсутствует, то используется текущее время. Если файл не существует, то он создается. Возвращает true при успешном выполнении, в обратном случае — false.

umask

Изменить маску

Описание

```
int umask(int mask);
```

Umask() устанавливает текущую umask PHP в mask & 0777 и возвращает старую umask. *Umask()* без аргументов возвращает текущую маску.

unlink

Удалить файл

Описание

```
int unlink(string filename);
```

Удаляет файл filename. Аналогично функции Unix C *unlink()*. Возвращает 0 или FALSE при ошибке.

См. также ***rmdir()*** *для удаления директорий.*

delete

Описание

```
void delete (string file)
```

diskfreespace

Возвращает количество свободного места в директории

Описание

```
float diskfreespace (string directory)
```


Пример:

```
$df = diskfreespace("/"); // $df contains the number of bytes
                          // available on "/"
```

См. также рис. С.9.

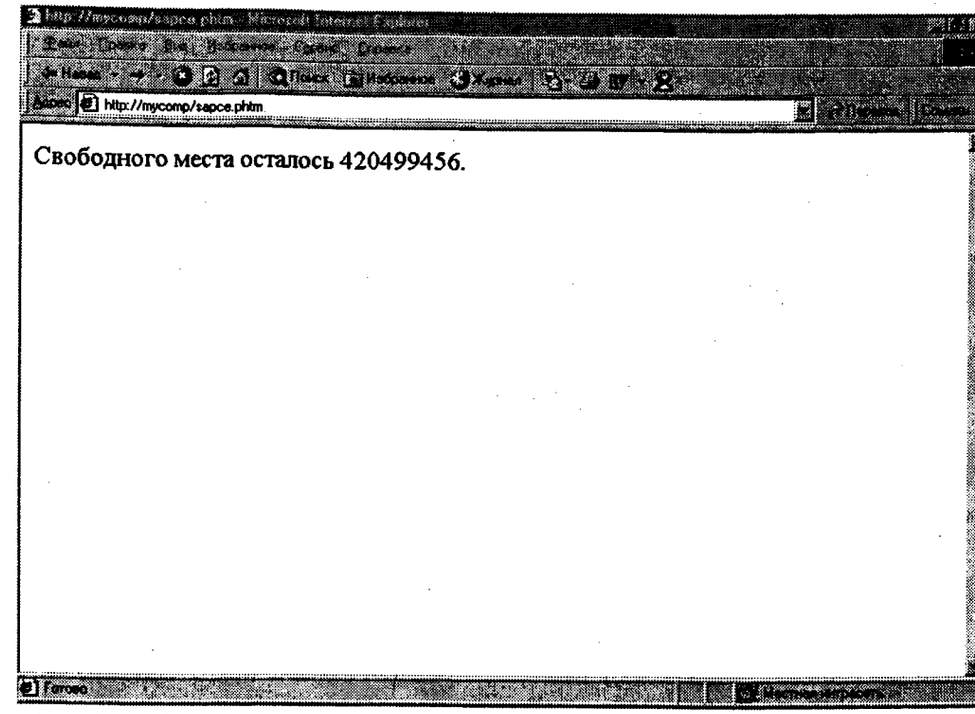


Рис. С.9. *Определяем количество свободного места в каталоге*

fgetcsv

Получает строку из указателя строки

Описание

```
array fgetcsv (int fp, int length [, string delimiter])
```

Пример:

```
$row = 1;.
$fp = fopen ("test.csv","r");
while ($data = fgetcsv ($fp, 1000, ",")) {
     $num = count ($data);
     print "<p> $num fields in line $row: <br>";
     $row++;
```

```
        for ($c=0; $c<$num; $c++) {
            print $data[$c]. "<br>";
        }
    }
    fclose ($fp);
```

flock

Запирает файл

Описание

```
bool flock (int fp, int operation [, int wouldblock])
```

Operation может принимать значения:

- LOCK_SH 1 (чтение);
- LOCK_EX 2 (запись);
- LOCK_UX 3 (освободить);
- LOCK_NB 4 (не блокировать).

fscanf

Анализирует содержимое файла в соответствии с указанным форматом

Описание

```
mixed fscanf (int handle, string format [, string var1...])
```

Функция похожа на *sscanf()*.

См. также fread(), fgets(), fgetss(), sscanf(), printf(), sprintf().

fstat

Получает информацию о файле

Описание

```
array fstat (int fp)
```

Возвращает массив со следующими элементами:

1) устройство device;
2) inode;
3) количество ссылок;
4) идентификатор пользователя id;
5) идентификатор группы;
6) тип устройства;
7) размер в байтах;
8) время последнего доступа;

9) время последней модификации;
10) время последнего изменения;
11) размер блока ввода-вывода;
12) количество блоков.

ftell

Сообщает указателю положение для чтения или записи

Описание

```
int ftell (int fp)
```

ftruncate

Обрезает файл в соответствии с заданной величиной

Описание

```
int ftruncate (int fp, int size)
```

set_file_buffer

Задает размер буфера для данного указателя файла

Описание

```
int set_file_buffer (int fp, int buffer)
```

realpath

Возвращает путь к файлу

Описание

```
string realpath (string path)
```

Пример:

```
$real_path = realpath ("../../index.php");
```

tmpfile

Создает временный файл

Описание

```
int tmpfile (void)
```

Создает временный файл в режиме для записи и возвращает указатель на этот файл. По окончании работы скрипта или вызове функции закрытия файла временный файл будет автоматически уничтожен.

См. также tempnam().

Функции для работы с FTP

При работе с модулем FTP определены константы FTP_ASCII и FTP_BINARY.

ftp_connect

Создает и возвращает поток FTP, при возникновении ошибки возвращает false

Описание

```
int ftp_connect (string host [, int port])
```

Функция *ftp_connect()* открывает FTP-соединение с хостом host. Параметр порта port не является обязательным. По умолчанию (или если задано значение 0) используется порт 21.

ftp_login

Передает имя и пароль

Описание

```
int ftp_login (int ftp_stream, string username, string password)
```

При удачном выполнении возвращает true, в противном случае — false.

ftp_pwd

Возвращает имя текущего директория

Описание

```
int ftp_pwd (int ftp_stream)
```

Возвращает текущий директорий или false при возникновении ошибки.

ftp_cdup

Родительский директорий

Описание

```
int ftp_cdup (int ftp_stream)
```

При удачной смене каталога возвращает true, при ошибке — false. Осуществляет переход в файловой системе на один каталог вверх.

ftp_chdir

Изменяет текущий директорий ftp-сервера

Описание

```
int ftp_chdir (int ftp_stream, string directory)
```

Осуществляет переход к указанному каталогу directory. Возвращает true, при возникновении ошибки возвращает false.

ftp_mkdir

Создает каталог

Описание

```
string ftp_mkdir (int ftp_stream, string directory)
```

Создает новый каталог на ftp-сервере (каталог directory). Возвращает true, при возникновении ошибки — false.

ftp_rmdir

Удаляет директорий

Описание

```
int ftp_rmdir (int ftp_stream, string directory)
```

Удаляет директорий directory. При удачном выполнении возвращает true, при возникновении ошибки возвращает false.

ftp_nlist

Возвращает список файлов текущего каталога

Описание

```
int ftp_nlist (int ftp_stream, string directory)
```

Возвращает массив имен фалов, при возникновении ошибки возвращает false.

ftp_rawlist

Возвращает детальный список файлов указанного каталога

Описание

```
int ftp_rawlist (int ftp_stream, string directory)
```

Функция *ftp_rawlist()* выполняет комантду FTP LIST и возвращает массив, каждый элемент которого представляет собой строку.

ftp_systype

Возвращает идентификатор типа удаленной системы ftp-сервера

Описание

```
int ftp_systype (int ftp_stream)
```

Возвращает тип удаленной системы, в случае возникновения ошибки возвращает false.

ftp_pasv

Включает и выключает пассивный режим

Описание

```
int ftp_pasv (int ftp_stream, int pasv)
```

Функция ***ftp_pasv()*** включает пассивный режим в том случае, если параметр pasv имеет значение true — пассивный метод выключен, если значение pasv равно false. В пассивном режиме связь инициализируется клиентом, а не сервером. При удачном выполнении функция возвращает true, при возникновении ошибки — false.

ftp_get

Загружает файл с ftp-сервера

Описание

```
int ftp_get (int ftp_stream, string local_file, string remote_file,
int mode)
```

Функция ***ftp_get()*** получает файл remote_file с ftp-сервера и сохраняет его в виде локального файла local_file. Режим mode должен иметь значение FTP_ASCII или FTP_BINARY. Возвращает true при благоприятном выполнении, при возникновении ошибки — false.

ftp_fget

Загружает файл с ftp-сервера и сохраняет его в виде открытого файла

Описание

```
int ftp_fget (int ftp_stream, int fp, string remote_file, int mode)
```

Функция ***ftp_fget()*** получает файл remote_file с ftp-сервера и записывает его в указатель fp. Режим устанавливается в виде FTP_ASCII или FTP_BINARY. При успешном выполнении возвращает true, при возникновении ошибки возвращает false.

ftp_put

Передает локальный файл ftp-серверу

Описание

```
int ftp_put (int ftp_stream, string remote_file, string local_file,
int mode)
```

Функция ***ftp_put()*** передает локальный файл local_file ftp-серверу и сохраняет на нем в виде файла с именем remote_file. Режим mode должен быть установлен в виде FTP_ASCII или FTP_BINARY. Возвращает true при удачном выполнении, при возникновении ошибки — false.

ftp_fput

Передает открытый файл ftp-серверу

Описание

```
int ftp_fput (int ftp_stream, string remote_file, int fp, int mode)
```

Функция ***ftp_fput()*** загружает локальный файл из указателяч fp (до достижения конца файла) на ftp-сервер и сохраняет его там под именем remote_file. Режим передачи должен быть установлен FTP_ASCII или FTP_BINARY. Возвращает true при благополучном выполнении, при наличии ошибок — false.

ftp_size

Возвращает размер указанного файла

Описание

```
int ftp_size (int ftp_stream, string remote_file)
```

Возвращает размер файла или −1 в случае ошибки.

ftp_mdtm

Возвращает время последнего изменения указанного файла

Описание

```
int ftp_mdtm (int ftp_stream, string remote_file)
```

Функция ***ftp_mdtm()*** определяет время последнего изменения файла и возвращает это время. В случае возникновения ошибки будет возвращено −1.

ftp_rename

Изменяет имя файла на ftp-сервере

Описание

```
int ftp_rename (int ftp_stream, string from, string to)
```

Функция ***ftp_rename()*** изменяет имя файла from и устанавливает новое имя to. При удачном выполнении возвращает true, при возникновении ошибки — false.

ftp_delete

Удаляет файл с ftp-сервера

Описание

```
int ftp_delete (int ftp_stream, string path)
```

Функция ***ftp_delete()*** удаляет файл path и возвращает true в случае удачного выполнения, при возникновении ошибки — false.

ftp_site

Посылает команду серверу

Описание

```
int ftp_site (int ftp_stream, string cmd)
```

Функция ***ftp_site()*** посылает команду cmd ftp-серверу. Набор допустимых команд зависит от типа сервера. При удачном выполнении возвращает true, при возникновении ошибки — false.

ftp_quit

Закрывает связь с ftp-сервером

Описание

```
int ftp_quit (int ftp_stream)
```


Функции для работы на уровне протокола HTTP

Эти функции позволяют работать с содержимым ответа сервера, посылаемого броузеру, на уровне протокола HTTP.

header

Посылает заголовок HTTP

Описание

```
int header (string string)
```

Функция ***Header()*** используется в начале HTML-файла для создания HTTP-заголовка. Эта функция должна быть вызвана перед тем, как будут отправлены данные, такие как ярлыки и текст HTML-файла. Использование таких функций как ***include()*** приводит к тому, что вывод данных может быть осуществлен перед тем, как будет сформирован заголовок при помощи функции ***header()***.

Существует два специальных случая вызова функции header. Первый вариант — указание места «Location». Такой заголовок не только будет послан броузеру, но и изменит статус сервера Apache (REDIRECT).

```
header ("Location: http://www.php.net");
/* Перенаправлет броузер на сайт PHP  */
exit;
/* Убедитесь в том, что расположенный далее код не будет выполнен до
того, как произойдет перенаправление броузера. */
```

Второй случай заголовков — заголовки, начинающиеся со строки «HTTP/» (без учета регистра). Например, если при ненахождении запрошенного ресурса можно обратиться к функции так:

```
header ("HTTP/1.0 404 Not Found");
```

Заголовок может содержать информации о кешировании страниц. Кеширование можно выключить, использовав, например, такой заголовок (приведено несколько вариантов):

```
header ("Expires: Mon, 16 Apr 2001 05:00:00 GMT");
// Это прошедшая дата
header ("Last-Modified: ". gmdate("D, d M Y H:i:s"). " GMT");
// страница изменена в момент загрузки (всегда изменена)
header ("Cache-Control: no-cache, must-revalidate");
// используется в HTTP/1.1
header ("Pragma: no-cache");
// используется в HTTP/1.0
```

См. также ***headers_sent()***.

header_sent

Возвращает значение true, если заголовок был послан

Описание

```
boolean headers_sent (void)
```

См. также ***header().***

setcookie

Посылает cookie

Описание

```
int setcookie (string name [, string value [, int expire [, string path
[, string domain [, int secure]]]]])
```

Функция должна быть выполнена перед тем, как будет осуществлен вывод данных перед ярлыками <html> и <head>.

Все аргументы, за исключением name, являются необязательными.

Пример setcookie():

```
setcookie ("TestCookie", "Test Value");
setcookie ("TestCookie", $value,time()+3600);  /* истекает через час */
setcookie ("TestCookie", $value,time()+3600, "/~rasmus/", ".utoronto.ca", 1);
```

Пример удаления cookies предыдущего примера.

Еще пример setcookie():

```
setcookie ("TestCookie");
setcookie ("TestCookie", "", time());
setcookie ("TestCookie", "", time(), "/~rasmus/", ".utoronto.ca", 1);
```

Содержимое cookies можно посмотреть таким способом:

```
echo $TestCookie;
echo $HTTP_COOKIE_VARS["TestCookie"];
```

Можно создать массив cookies:

```
setcookie ("cookie[three]", "cookiethree");
setcookie ("cookie[two]", "cookietwo");
setcookie ("cookie[one]", "cookieone");
if (isset ($cookie)) {
     while (list ($name, $value) = each ($cookie)) {
          echo "$name == $value<br>\n";
     }
}
```

Для получения подробной информации о работе с cookies смотрите http://www.netscape.com/newsref/std/cookie_spec.html.

Функции для работы с изображениями

Эти функции доступны при наличии библиотеки GD и позволяют работать с изображениями JPEG GIF, PNG, SWF (см. http://www.boutell.com/gd/).

Формат изображений, с которыми можно работать, зависит от версии библиотеки gd. Версии, старше gd-1.6, поддерживают gif, но не поддерживают png, в то время как версии моложе gd-1.6 поддерживают png, но не поддерживают gif.

GetImageSize

Возвращает размер кортинок в формате GIF, JPEG, PNG, SWF

Описание

```
array getimagesize (string filename [, array imageinfo])
```

Функция определяет размер изображений в формает GIF, JPG, PNG, SWF и возвращает тип файла, размеры по вертикали и горизонтали. Возвращает 4 элемента: элемент 0 — ширина рисунка в пикселях, 1 — высота рисунка, 2 — флаг типа рисунка (1 = GIF, 2 = JPG, 3 = PNG, 4 = SWF), 3 — текстовая строка типа «height=xxx width=xxx», такая строка может быть помещена непосредственно в ярлык <IMG>.

Пример GetImageSize:

```
<?php $size = GetImageSize ("img/flag.jpg"); ?>
<IMG SRC="img/flag.jpg" <?php echo $size[3]; ?>
```

Параметр imageinfo не является обязательным. С его помощью можно получить дополнительную инфомрмацию о рисунке.

Пример:

```
<?php
     $size = GetImageSize ("testimg.jpg",&$info);
     if (isset ($info["APP13"])) {
          $iptc = iptcparse ($info["APP13"]);
          var_dump ($iptc);
     }
?>
```

ImageArc

Рисует фрагмент эллипса

Описание

```
int ImageArc (int im, int cx, int cy, int w, int h, int s, int e, int col)
```

С помощью этой функции можно нарисовать часть эллипса с центром в cx, cy (верхний правый угол имеет координаты 0, 0) в картинке im. W и h — это ширина и высота эллипса соответственно, параметры s и e — это начальная и конечная точки эллипса (части эллипса) в градусах.

ImageChar

Горизонтальное рисование символа

Описание

```
int imagechar (int im, int font, int x, int y, string c, int col)
```

Функция ***ImageChar()*** рисует первый символ строки в рисунке im, расположенную верхним левым углом в точке с координатами x, y цветом, указанным в виде параметра col. Если в качестве параметра font используются числа 1, 2, 3, 4 и 5, то используется встроенный фонт, чем больше число, тем крупнее изображение рисуемого символа.

См. также ***imageloadfont()***.

ImageCharUp

Вертикальное рисование символа

Описание

```
int imagecharup (int im, int font, int x, int y, string c, int col)
```

Функция ***ImageCharUp()*** рисует первый символ из строки c, располагая его вертикально, в рисунке im в положении с коорждинатами x и y с цветом, указанным в col. В качестве font указывается значения от 1 до 5 (целыек числа). Чем больше число, тем крупнее символ.

См. также ***imageloadfont()***.

ImageColorAllocate

Определить (задать) цвет рисунка

Описание

```
int imagecolorallocate (int im, int red, int green, int blue)
```


Функция ***ImageColorAllocate()*** возвращает идентификатор цвета для каждого из цветов, используемых в рисунке. Цвет задается в формате RGB. Значение аргумента im может быть получено при помощи функции ***imagecreate()*** (она возвращает значение именно этого аргумента. Функцию ***ImageColorAllocate()*** необходимо вызывать для того, чтобы создать все цвета, которые будут использованы в данном рисунке im.

```
$white = ImageColorAllocate ($im, 255, 255, 255);
$black = ImageColorAllocate ($im, 0, 0, 0);
```

ImageColorDeAllocate

Убрать цвет рисунка

Описание

```
int imagecolordeallocate (int im, int index)
```

Функция ***ImageColorDeAllocate()*** удаляет цвет, который предварительно был определен при помощи функции ***ImageColorAllocate()***.

```
$white = ImageColorAllocate($im, 255, 255, 255);
ImageColorDeAllocate($im, $white);
```

ImageColorAt

Получить цвет данного пикселя

Описание

```
int imagecolorat (int im, int x, int y)
```

Возвращает цвет пикселя с указанными координаты для изображения im.

См. также ***imagecolorset(), imagecolorsforindex()***.

ImageColorClosest

Получить цвет, ближайший к указанному

Описание

```
int imagecolorclosest (int im, int red, int green, int blue)
```

Возвращает цвет, ближайщий к указанному, в виде параметров функции, из набора цветов, определенных для рисунка im. Расстояние до цвета определяется как расстояние между точками в трехмерном пространстве с координатами RGB.

См. также ***imagecolorexact()***.

ImageColorExact

Возвращает имя цвета

Описание

```
int imagecolorexact (int im, int red, int green, int blue)
```

Возвращает имя цвета по значениям RGB для рисунка im. Если указанный цвет в риснуке не определен, то возвращает −1.

См. также ***imagecolorclosest().***

ImageColorResolve

Получить индекс цвета или индекс ближайшего цвета

Описание

```
int imagecolorresolve (int im, int red, int green, int blue)
```

Эта функция возвращает индекс (название) цвета в рисунке im по указанным значениям RGB. Если указанный цвет не используется в рисунке, то возвращается индекс ближайшего цвета, используемого в рисунке.

См. также ***imagecolorclosest().***

ImageGammaCorrect

Применить коррекцию цвета в рисунке GD

Описание

```
int imagegammacorrect (int im, double inputgamma, double outputgamma)
```

Функция ***ImageGammaCorrect()*** использует коррекцию гаммы в рисунке im на основе параметров inputgamma и outputgamma.

ImageColorSet

Задает цвет

Описание

```
bool imagecolorset (int im, int index, int red, int green, int blue)
```

Задает цвет для индекса цвета в наборе цветов.

См. также ***imagecolorat().***

ImageColorsForIndex

Определяет цвет по индексу цвета

Описание

```
array imagecolorsforindex (int im, int index)
```

Возвращает ассоциативный массив (с элементами red, green, blue) для указанного цветового индекса.

См. также ***imagecolorat(), imagecolorexact().***

ImageColorsTotal

Определяет количество цветов, используемых в рисунке

Описание

```
int imagecolorstotal (int im)
```

Возвращает количество цветов рисунка im.

См. также ***imagecolorat(), imagecolorsforindex().***

ImageColorTransparent

Делает цвет прозрачным

Описание

```
int imagecolortransparent (int im [, int col])
```

Функция ***ImageColorTransparent()*** задает цвет col в рисунке im как прозрачный. Col — это знгачение, возвращаемое функцией ***ImageColorAllocate()***. Функция возвращает идентификатор прозрачного цвета.

ImageCopy

Копировать часть рисунка

Описание

```
int ImageCopy (int dst_im, int src_im, int dst_x, int dst_y,
int src_x, int src_y, int src_w, int src_h)
```

Копировать часть рисунка src_im в рисунок dst_im, фрагмент рисунка начинается в точке с координатами src_x, src_y, ширина фрагмента src_w, высота фрагмента src_h. Копируемый фрагмент будет помещен в новый рисунок в положение, заданное при помощи координат dst_x и dst_y.

ImageCopyResized

Копировать и масштабировать фрагмент рисунка

Описание

```
int imagecopyresized (int dst_im, int src_im, int dstX, int dstY,
int srcX, int srcY, int dstW, int dstH, int srcW, int srcH)
```

Функция *ImageCopyResized()* копирует прямоугольный фрагмент рисунка в другой рисунок. Dst_im — рисунок, в который производится копирование, src_im — рисунок, из которого производится копирование. Если ширина и высота исходного фрагмента отличаются от ширины и высоты конечного фрагмента, по производится необходимое масштабирование. Эта функция может быть испльзована для осуществления копирования фрагмента внутри одного и того же рисунка. Однако, если копируемый фрагмент накладывается на область, которая хотя бы частично пересекается с исходной областью, то результат может быть непредсказуемым.

ImageCreate

Создает новый рисунок

Описание

```
int imagecreate (int x_size, int y_size)
```

Функция *ImageCreate()* возвращает идентификатор рисунка (пустого), размеры которого определяются параметрами x_size и y_size.

Пример:

```
<?php
header ("Content-type: image/png");
$im = @ImageCreate (50, 100)
    or die ("Cannot Initialize new GD image stream");
$background_color = ImageColorAllocate ($im, 255, 255, 255);
$text_color = ImageColorAllocate ($im, 233, 14, 91);
ImageString ($im, 1, 5, 5,  "Простая текстовая строка", $text_color);
ImagePng ($im);
?>
```

ImageCreateFromGIF

Создать новый рисунок из файла или URL

Описание

```
int imagecreatefromgif (string filename)
```

Функция *ImageCreateFromGif()* возвращает идентификатор для вновь созданного рисунка.

ImageCreateFromJPEG

Создает новый рисунок из файла или URL

Описание

```
int imagecreatefromjpeg (string filename)
```

Функция *ImageCreateFromJPEG()* возвращает идентификатор нового рисунка (в случае ошибки — пустую строку).

Пример:

```
function LoadJpeg ($imgname) {
    $im = @ImageCreateFromJPEG ($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
        $im = ImageCreate (150, 30); /* Create a blank image */
        $bgc = ImageColorAllocate ($im, 255, 255, 255);
        $tc  = ImageColorAllocate ($im, 0, 0, 0);
        ImageFilledRectangle ($im, 0, 0, 150, 30, $bgc);
        /* Output an errmsg */
       ImageString ($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
```

ImageCreateFromPNG

Создает новый рисунок из файла или URL

Описание

```
int imagecreatefrompng (string filename)
```

Функция *ImageCreateFromPNG()* возвращает идентификатор рисунка или пустую строку (в случае ошибки).

Пример:

```
function LoadPNG ($imgname) {
    $im = @ImageCreateFromPNG ($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
        $im  = ImageCreate (150, 30); /* Create a blank image */
        $bgc = ImageColorAllocate ($im, 255, 255, 255);
        $tc  = ImageColorAllocate ($im, 0, 0, 0);
        ImageFilledRectangle ($im, 0, 0, 150, 30, $bgc);
        /* Output an errmsg */
       ImageString ($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
```

ImageDashedLine

Рисует пунктирную линию

Описание

```
int imagedashedline (int im, int x1, int y1, int x2, int y2, int col)
```

Функция *ImageDashedLine()* рисует пунктирную линию из точки x1, y1 к точке x2, y2 в рисунке im цветом col.

См. также ImageLine().

ImageDestroy

Уничтожает рисунок

Описание

```
int imagedestroy (int im)
```

Функция *ImageDestroy()* уничтожает рисунок im и освобождает память.

ImageFill

Заливка цветом

Описание

```
int imagefill (int im, int x, int y, int col)
```

Функция *ImageFill()* производит заливку цветом col в рисунке im, заливка начинается в точке с координатами x, y.

ImageFilledPolygon

Рисует закрашенный многоугольник

Описание

```
int imagefilledpolygon (int im, array points, int num_points, int col)
```

Функция *ImageFilledPolygon()* создает закрашенный многоугольник в рисунке im. Массив Points содержит набор координат многоугольника, те. points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1 и т.д. Num_points — общее количество вершин многоугольника.

ImageFilledRectangle

Рисует закрашенный прямоугольник

Описание

```
int imagefilledrectangle (int im, int x1, int y1, int x2, int y2, int col)
```

Функция *ImageFilledRectangle()* создает прямоугольник цвета col в рисунке im с верхней левой координатой x1, y1 и правой нижней координатой x2, y2.

ImageFillToBorder

Заливка заданным цветом

Описание

```
int imagefilltoborder (int im, int x, int y, int border, int col)
```

Функция *ImageFillToBorder()* осуществляет заливку с цветом границы, соответствующим border. Начальная точка заливки определяется координатами x, y, основной цвет заливки — col.

ImageFontHeight

Определяет размер фонта

Описание

```
int imagefontheight (int font)
```

Возвращает высоту символов указанного фонта.

См. также ImageFontWidth(), ImageLoadFont().

ImageFontWidth

Определяет ширину фонта

Описание

```
int imagefontwidth (int font)
```

Возвращает ширину фонта font в пикселях.

См. также ImageFontHeight(), ImageLoadFont().

ImageGIF

Выводит рисунок броузеру или в файл

Описание

```
int imagegif (int im [, string filename])
```

Функция ***ImageGIF()*** создает файл GIF с указанным именем im. Аргумент im — это значение, возвращаемое функцией ***imagecreate()***.

Формат рисунка — GIF87a или, если использована функция для задания прозрачных цветова ***ImageColorTransparent()***, то GIF89a.

Имя файла указывать необязательно. Если имя не указано, то поток данных рисунка направляется броузеру. Для того, чтобы такой поток был правильно обработан броузером, необходимо воспользоваться функцией ***header()*** и указать тип посылаемых данных в виде image/gif. Таким образом можно создать скрипт, который выводит изображение непосредственно.

ImagePNG

Выводит рисунок PNG в файл или передает его броузеру

Описание

```
int imagepng (int im [, string filename])
```

Функция ***ImagePng()*** выводит поток рисунка im в файл. Если имя файла не указано, то поток будет направлен непосредственно броузеру (на стандартное устройство вывода).

```
<?php
$im = ImageCreateFromPng("test.png");
ImagePng($im);
?>
```

ImageJPEG

Выводит рисунок jpeg в файл или передает броузеру

Описание

```
int imagejpeg (int im [, string filename [, int quality]])
```

Функция ***ImageJPEG()*** создает файл с рисунком JPEG для картинки im.

Если имя файла не указано, то поток данных будет направлен непосредственно броузеру. Чтобы броузер смог правильно обработать полученные данные, необходимо предварительно указать тип посылаемых данных (image/jpeg), используя функцию ***header()***.

ImageInterlace

Включает и выключает режим interlace

Описание

```
int imageinterlace (int im [, int interlace])
```

Функция ***ImageInterlace()*** включает или выключает бит interlacer.

ImageLine

Рисует линию

Описание

```
int imageline (int im, int x1, int y1, int x2, int y2, int col)
```

Функция ***ImageLine()*** рисует линию из точки x1, y1 к точке x2, y2 цветом col.

См. также ***ImageCreate(), ImageColorAllocate().***

ImageLoadFont

Загружает новый фонт

Описание

```
int imageloadfont (string file)
```

Функция ***ImageLoadFont()*** загружает пользовательский фонт и возвращает идентификатор фонта (всегда больше 5, поэтому никогда не конфликтует со встроенными фонтами) (табл. С.1).

Таблица С.1
Формат файла пользовательского фонта

Положение байта	Тип данных	Определение
Байты 0...3	int	Количество символов фонта
Байты 4...7	int	Значение первого символа фонта (часто 32 для пробела)
Байты 8...11	int	Ширина в пикселях для символов фонта
Байты 12...15	int	Высота в пикселях для символов фонта
Байты 16...	char	Массив данных символов, один байт на пиксель, всего байт (nchars*width*height)

См. также ***ImageFontWidth() и ImageFontHeight().***

ImagePolygon

Рисует многоугольник

Описание

```
int imagepolygon (int im, array points, int num_points, int col)
```

Функция *ImagePolygon()* создает многоугольник в рисунке im. Массив Points содержит координаты вершин прямоугольника, т.е. points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1. Num_points — это общее количество вершин прямоугольника.

См. также ***imagecreate().***

ImagePSBBox

Создает рамку

Описание

```
array imagepsbbox (string text, int font, int size [, int space [,
int tightness [, float angle]]])
```

Size ..... в пикселях.

Space ..... позволяет изменить размер пробела, указанный по умолчанию. Эта величина прибавляется к исходному значению, может быть отрицательна.

Tightness ..... позволяет контролировать количество пробелов между символами. Может быть отрицательна (прибавляется к стандартному значению).

Angle ..... в градусах.

Функция возвращает массив, содержащий следующие элементы:

0 ..... нижняя левая x-координата;

1 ..... нижняя левая координата y;

2 ..... верхняя правая координата x;

3 ..... верхняя правая координата y.

См. также ***imagepstext().***

ImagePSEncodeFont

Изменяет вектор кодировки символов

Описание

```
int imagepsencodefont (string encodingfile)
```

Функция загружает вектор кодировки из файла в соответствии с которым меняет кодировку символов.

ImagePSFreeFont

Освобождает память, отведенную под фонт PostScript Type 1

Описание

```
void imagepsfreefont (int fontindex)
```

См. также ***ImagePSLoadFont().***

ImagePSLoadFont

Загружает из файла фонт PostScript Type 1

Описание

```
int imagepsloadfont (string filename)
```

Возвращает индекс фонта.

См. также ***ImagePSFreeFont().***

ImagePsExtendFont

Сжимает или расширяет фонт

Описание

```
bool imagepsextendfont (int font_index, double extend)
```

Расширяет или сжимает фонт font_index, в соответствии с масштабом, указанным в виде значения extend.

ImagePsSlantFont

Изменяет фонт в соответствии с параметром

Описание

```
bool imagepsslantfont (int font_index, double slant)
```

Величина изменения задается параметром slant.

ImagePSText

Пишет текст поверх рисунка, используя фонт PostScript Type1

Описание

```
array imagepstext (int image, string text, int font, int size, int
foreground, int background, int x, int y [, int space [, int tightness
[, float angle [, int antialias_steps]]]])
```

Size ..... в пикселях.
Foreground ..... цвет текста.
Background ..... цвет фона, к которому сводится цвет фонта при использовании эффектов, в которые вовлечен фон.
Координаты x, y ..... определяют начало текста (положение первого символа).
Space ..... параметр расстояния между символами. Может быть отрицательным, так как прибавляется к величине, установленной по умолчанию.
Tightness ..... количество пробелов между символами. Может быть отрицательным.
Angle ..... угол в градусах.
Antialias_steps ..... количество цветов от 4 до 16.

Параметры space и tightness выражается в количестве единиц. 1 единица равна 1/1000 от em-square.

Функция фозвращает массив, содержащий следующие значения.
0 ..... нижняя левая x-координата;
1 ..... нижняя левая координата y;
2 ..... верхняя правая координата x;
3 ..... верхняя правая координата y.

См. также ***imagepsbbox().***

ImageRectangle

Рисовать прямоугольник

Описание

```
int imagerectangle (int im, int x1, int y1, int x2, int y2, int col)
```

Функция ***ImageRectangle()*** рисует прямоугольник цветом col, верхняя левая координата x1, y1, правая нижняя координата x2, y2.

ImageSetPixel

Рисует один пиксель

Описание

```
int imagesetpixel (int im, int x, int y, int col)
```

Функция ***ImageSetPixel()*** рисует пиксель с координатами x, y (верхний левый угол соответствует 0, 0) в картинке im цветом col.

См. также ***ImageCreate(), ImageColorAllocate().***

ImageString

Рисует строку, расположенную горизонтально

Описание

```
int imagestring (int im, int font, int x, int y, string s, int col)
```

Функция ***ImageString()*** пишет строку s в рисунке im с координатами x, y цветом col. В случае параметров фонтов из набора 1, 2, 3, 4 и 5, используется встроенный фонт.

См. также ***ImageLoadFont().***

ImageStringUp

Рисует стрку вертикально

Описание

```
int imagestringup (int im, int font, int x, int y, string s, int col)
```

Функция ***ImageStringUp()*** рисует строку s вертикально в рисунке im с координатами x, y и цветом col. Параметр фонта из набора 1, 2, 3, 4 и 5 соответствует встроенным фонтам.

См. также ***ImageLoadFont().***

ImageSX

Получить ширину рисунка

Описание

```
int imagesx (int im)
```

Функция ***ImageSX()*** возвращает ширину рисунка im.

См. также ***ImageCreate(), ImageSY().***

ImageSY

Получить высоту рисунка

Описание

```
int imagesy (int im)
```

Функция ***ImageSY()*** возвращает высоту рисунка im.

См. также ***ImageCreate(), ImageSX().***

ImageTTFBBox

Определяет обрамляющий прямоугольник текста, созданного с использованием фонта TTF

Описание

```
array imagettfbbox (int size, int angle, string fontfile, string text)
```

Функция вычисляет и возвращает параметры обрамляющего прямоугольника для текста, написанного фонтом TTF:

text ..... строка, размер которой определяет функция;
size ..... размер фонта;
fontfile ..... имя файла, в котором хранится фонт TTF (или URL);
angle ..... угол в градусах.

Функция ***ImageTTFBBox()*** возвращает массив, состоящий из 8 элементов, соответствующих четырем вершинам обрамляющего текст прямоугольника:

0 ..... левый нижний угол, x координата;
1 ..... левый нижний угол, y координата;
2 ..... правый нижний угол, x координата;
3 ..... правый нижний угол, y координата;
4 ..... верхний правый угол x координата;
5 ..... верхний правый угол, y координата;
6 ..... верхний левый угол, x координата;
7 ..... верхний левый угол, y координата.

Точки определяется относительно текста и вне зависимости от значения угла.

См. также ***ImageTTFText().***

ImageTTFText

Пишет текст в картинке с использование фонта TTF

Описание

```
array imagettftext (int im, int size, int angle, int x, int y, int
col, string fontfile, string text)
```

Функция ***ImageTTFText()*** пишет текст text в рисунке im, с началом в координатах x, y под углом angle и цветом col с использованием фонта TTF, хранящегося в файле fontfile.

Угол указывается в градусах, нулевой угол отсчитывается от положения стрелки на трех часах, отсчет производится против часовой стрелки.

Fontfile ..... путь и название файла с фонтом TTF.
Text ..... текстовая строка, предназначенная для записи в рисунке, может включать символы в кодировке UTF-8 (например, {).
Col ..... цвет шрифта.

Функция ***ImageTTFText()*** возвращает массив, состоящий из восьми элементов, которыми являются координаты четырех вершин обрамляющего текст прямоугольника.

Пример функции ImageTTFText():

```
<?php
Header ("Content-type: image/gif");
$im = imagecreate (400, 30);
$black = ImageColorAllocate ($im, 0, 0, 0);
$white = ImageColorAllocate ($im, 255, 255, 255);
ImageTTFText ($im, 20, 0, 10, 20, $white, "/path/arial.ttf",
                "Testing... ");
ImageGif ($im);
ImageDestroy ($im);
?>
```

ImageTypes

Возвращает тип изображения, поддерживаемый этой версией PHP

Описание

```
int imagetypes(void);
```

Эта функция возвращает бит-поле, соответствующее типу изображения в PHP, поддерживаемого GD. Могут быть возвращены IMG_GIF| IMG_JPG | IMG_PNG | IMG_WBMP. Приведем пример проверки изображения формата PNG:

Пример функции ImageTypes:

```
<?php
if (ImageTypes() & IMG_PNG) {
     echo "PNG Support is enabled";
}
?>
```

read_exif_data

Читать заголовки EXIF из файла JPEG

Описание

```
array read_exif_data (string filename)
```

Функция ***read_exif_data()*** читает заголовки EXIF из файла JPEG. Функция возвращает ассоциативный массив, индексами которого являются имена заголовков Exif, а значениями являются значения, связанные с этими заголовками.

Пример с функцией read_exif_data():

```
<?php
 $exif = read_exif_data ('p0001807.jpg');
 while(list($k,$v)=each($exif)) {
   echo "$k: $v<br>\n";
 }
?>
```

Результат работы:

```
FileName: p0001807.jpg
FileDateTime: 929353056
FileSize: 378599
CameraMake: Eastman Kodak Company
CameraModel: KODAK DC265 ZOOM DIGITAL CAMERA (V01.00)
DateTime: 1999:06:14 01:37:36
Height: 1024
Width: 1536
IsColor: 1
FlashUsed: 0
FocalLength:  8.0mm
RawFocalLength: 8
ExposureTime:  0.004 s (1/250)
RawExposureTime: 0.0040000001899898
ApertureFNumber: f/ 9.5
RawApertureFNumber: 9.5100002288818
FocusDistance: 16.66m
RawFocusDistance: 16.659999847412
Orientation: 1
ExifVersion: 0200
```

Эта функция работает только в PHP4 и не требует библиотеки GD.

Функции imap для работы с почтой

imap_append

Добавляет текстовое сообщение в указанный почтовый ящик

Описание

```
int imap_append(int imap_stream, string mbox, string message,
stringflags);
```

Возвращает true, в случае возникновения ошибки — false. Функция ***imap_append()*** добавляет текстовое сообщение в указанный почтовый ящик mbox. Если указаны необязательные флаги, то записывает в почтовый ящик и флаги.

При общении с сервером Cyrus IMAP нужно использовать в качестве ограничителей строки «\r\n» вместо «\n», иначе действие не будет выполняться.

imap_base64

Декодирует текст, закодированный с помощью BASE64

Описание

```
string imap_base64(string text);
```

Функция ***imap_base64()*** декодирует текст в формате BASE-64. Декодированное сообщение возвращается как строка.

imap_body

Читает тело сообщения

Описание

```
string imap_body(int imap_stream, int msg_number, int flags);
```

Функция ***imap_body()*** возвращает тело сообщения, имеющего номер msg_number в текущем почтовом ящике. Необязательные флаги — это битовые маски из следующего списка:

FT_UID ..... номер сообщения msgno является уникальным идентификатором сообщения;

FT_PEEK ..... не устанавливать флаг \Seen;

FT_INTERNAL ..... возвращаемая строка записана во внутреннем формате и не может быть приведена к канонической форме с CRLF.

imap_check

Проверяет текущий почтовый ящик

Описание

```
array imap_check(int imap_stream);
```

Возвращает информацию о текущем почтовом ящике. В случае неуспеха возвращает FALSE.

Функция ***imap_check()*** проверяет статус текущего почтового ящика на сервере и возвращает информацию в объекте со следующими свойствами:

- **Date** ..... дата сообщения;
- **Driver** ..... драйвер;
- **Mailbox** ..... название почтового ящика;
- **Nmsgs** ..... количество сообщений;
- **Recent** ..... количество недавно пришедших сообщений.

imap_close

Закрывает поток IMAP

Описание

```
int imap_close(int imap_stream, int flags);
```

Закрывает поток imap. Необязательный флаг CL_EXPUNGE заставляет стереть помеченные на удаление сообщения при закрытии.

imap_createmailbox

Создает новый почтовый ящик

Описание

```
int imap_createmailbox(int imap_stream, string mbox);
```

Imap_createmailbox() создает новый почтовый ящик, указанный в mbox. Возвращает true в случае успеха и false при ошибке.

imap_delete

Помечает сообщение из текущего почтового ящика на удаление

Описание

```
int imap_delete(int imap_stream, int msg_number);
```

Возвращает true. Функция ***imap_delete()*** помечает сообщение, указанное через msg_number на удаление. Настоящее удаление сообщений осуществляется функцией ***imap_expunge()***.

imap_deletemailbox

Удаляет почтовый ящик

Описание

```
int imap_deletemailbox(int imap_stream, string mbox);
```

Imap_deletemailbox() удаляет указанный почтовый ящик. Возвращает tru, при ошибке — false.

imap_expunge

Удаляет все сообщения, помеченные на удаление

Описание

```
int imap_expunge(int imap_stream);
```

Imap_expunge() удаляет все сообщения, помеченные на удаление с помощью ***imap_delete()***. Возвращает true.

imap_fetchbody

Извлекает фрагмент тела сообщения

Описание

```
string imap_fetchbody(int imap_stream, int msg_number, int
part_number, flags flags);
```

Эта функция заставляет извлечь подробную секцию указанного сообщения как текстовую строку. Секция — это строка целых чисел, разделенных точками, которые указывают на части тела сообщения в списке частей согласно спецификации IMAP4. Части тела не декодируются этой функцией. Необязательным параметром к ***imap_fetchbody()*** является битовая маска из:

- **FT_UID** ..... msgono является уникальным идентификатором;
- **FT_PEEK** ..... не устанавливать флаг \Seen;
- **FT_UID** ..... возвращаемая строка записана во внутреннем формате, которая не может быть канонизирована с помощью CRLF.

imap_fetchstructure

Читает структуру простого сообщения

Описание

```
array imap_fetchstructure(int imap_stream, int msg_number);
```

Эта функция заставляет извлечь всю информацию о структуре сообщения с номером msg_number. В случае сообщения из нескольких частей, функция также возвращает массив объектов всех свойств под названием parts[].

imap_header

Читает заголовок сообщения

Описание

```
object imap_header(int imap_stream, int msg_number, int fromlength,
int subjectlength, int defaulthost);
```

Эта функция возвращает объект для различных элементов заголовка: remail, date, Date, subject, Subject, in_reply_to, message_id, newsgroups, followup_to, references.

to[] — возвращает массив объектов из строки To, содержит:

- personal
- adl
- mailbox
- host

fromaddress — полная строка From: строка длиной до 1024 символов;
from[] — возвращает массив объектов из строки From, содержит:

- personal
- adl
- mailbox
- host

ccaddress (полная строка Cc: строка длиной до 1024 символов);
cc[] — возвращает массив объектов из строки Cc, содержит:

- personal
- adl
- mailbox
- host

bccaddress (полная строка Bcc: строка длиной до 1024 символов);
bcc[] — возвращает массив объектов из строки Bcc, содержит:

- personal
- adl
- mailbox
- host

reply_toaddress (полная строка Reply_to: строка длиной до 1024 символов);
reply_to[] — возвращает массив объектов из строки Reply_to, содержит:

- personal
- adl
- mailbox
- host

senderaddress (полная строка Sender: строка длиной до 1024 символов);
sender[] — возвращает массив объектов из строки Sender, содержит:

- personal
- adl
- mailbox
- host

return_path (полная строка Return-path: строка длиной до 1024 символов);
return_path[] — возвращает массив объектов из строки Return_path, содержит:

- personal
- adl
- mailbox
- host

udate (дата сообщения в формате времени unix);

fetchfrom (строка From, отформатированная до fromlength символов);

fetchsubject (строка Subject, отформатированная до subjectlength символов).

imap_headers

Возвращает заголовки всех сообщений в почтовом ящике

Описание

```
array imap_headers(int imap_stream);
```

Возвращает строковый массив из информации по заголокам. Один элемент массива на сообщение.

imap_listmailbox

Читает список почтовых ящиков

Описание

```
array imap_listmailbox(int imap_stream, string ref, string pat);
```

Возвращает массив, содержащий названия почтовых ящиков.

imap_listsubscribed

Перечисляет все подписанные ящики

Описание

```
array imap_listsubscribed(int imap_stream, string ref, string
pattern);
```

Возвращает массив всех почтовых ящиков, на которые существует подписка. Аргументы ref и pattern указывают начальное месторасположение откуда начинать поиск и шаблон, которому должны удовлетворять названия почтовых ящиков.

imap_mail_copy

Копирует указанные сообщения в почтовый ящик

Описание

```
int imap_mail_copy(int imap_stream, string msglist, string mbox, int flags);
```

Возвращает true, при ошибке — false.

Копирует почтовые сообщения, указанные с помощью msglist, в почтовый ящик mbox. Msglist — это диапазон, а не просто номера сообщений.

Флаги — это битовые маски из списка:

- **CP_UID** ..... номера в последовательности содержат уникальные идентификаторы;
- **CP_MOVE** ..... после копирования удалить сообщения из текущего почтового ящика.

imap_mail_move

Переносит указанные сообщения в почтовый ящик

Описание

```
int imap_mail_move(int imap_stream, string msglist, string mbox);
```

Переносит почтовые сообщения, указанные с помощью msglist, в почтовый ящик mbox. Msglist — это диапазон, а не просто номера сообщений.

Возвращает true в случае успеха, иначе — false.

imap_num_msg

Выдает количество сообщений в текущем почтовом ящике

Описание

```
int imap_num_msg(void);
```

Возвращает количество сообщений в текущем почтовом ящике.

imap_num_recent

Возвращает количество недавно пришедших сообщений в текущем почтовом ящике

Описание

```
int imap_num_recent(int imap_stream);
```

Возвращает количество недавно пришедших сообщений в текущем почтовом ящике.

imap_open

Открывает поток IMAP в почтовый ящик

Описание

```
int imap_open(string mailbox, string username, string password, int
flags);
```

В случае успеха возвращает поток IMAPl, иначе — false. Эта функция может быть использована для открытия потоков к POP3- и NNTP-серверам. Для того, чтобы присоединиться к серверу IMAP на 143-й порт на локальной машине, сделайте следующее:

```
$mbox = imap_open("{localhost:143}INBOX","user_id","password");
```

Для того, чтобы подсоединиться к POP3-серверу на 110-й порт на локальном сервере используйте:

```
$mbox = imap_open("{localhost/pop3:110}INBOX","user_id","password");
```

Для того, чтобы подсоединиться к NNTP-серверу на 119-й порт на локальном сервере используйте:

```
$nntp = imap_open("{localhost/nntp:119}comp.test","","");
```

Для того, чтобы подсоединиться к удаленному серверу, замените «localhost» на имя или IP-адрес сервера, к которому вы хотите подсоединиться.

Опции — битовая маска из:

- **OP_READONLY** ..... открыть почтовый ящик в режим «только чтение»;
- **OP_ANONYMOUS** ..... не использовать или не обновлять .newsrc при использовании новостей;
- **OP_HALFOPEN** ..... для IMAP и NNTP устанавливает соединение, но не открывает почтовый ящик;
- **CL_EXPUNGE** ..... автоматически очищать почтовый ящик при закрытии.

imap_ping

Проверяет поток IMAP на работоспособность

Описание

```
int imap_ping(int imap_stream);
```

Возвращает true, если поток еще работоспособен, иначе — false.

Функция ***imap_ping()*** проверяет поток на работоспособность. Он может также проверять новую почту. Это предпочтительный метод для периодической проверки новой почты и «живучести» удаленных серверов.

imap_renamemailbox

Переименовывает старый почтовый ящик в новый

Описание

```
int imap_renamemailbox(int imap_stream, string old_mbox, string
new_mbox);
```

Эта функция переименовывает старый почтовый ящик в новый.
Возвращает true или false.

imap_reopen

Заново открывает поток IMAP на новый почтовый ящик

Описание

```
int imap_reopen(string imap_stream, string mailbox, string [flags]);
```

Возвращает true в случае успеха, иначе — false.

Эта функция заново открывает указанный поток на новый ящик.

Опции — битовая маска из:

- **OP_READONLY** ..... открыть почтовый ящик в режиме только чтение;

- **OP_ANONYMOUS** ..... не использовать или не обновлять .newsrc при работе с новостями;
- **OP_HALFOPEN** ..... для IMAP и NNTP устанавливает связь но не открывает почтовый ящик;
- **CL_EXPUNGE** ..... счищает почтовый ящик при закрытии.

imap_subscribe

Подписывает на почтовый ящик

Описание

```
int imap_subscribe(int imap_stream, string mbox);
```

Подписывает на новый почтовый ящик.
Возвращает true или false.

imap_undelete

Снимает отметку с сообщения, помеченного на удаление

Описание

```
int imap_undelete(int imap_stream, int msg_number);
```

Эта функция снимает отметку с сообщения, помеченного на удаление функцией ***imap_delete()***.

Возвращает true в случае успеха, иначе — false.

imap_unsubscribe

Снимает подписку с почтового ящика

Описание

```
int imap_unsubscribe(int imap_stream, string mbox);
```

Снимает подписку с почтового ящика.
Возвращает true в случае успеха, иначе — false.

imap_qprint

Конвертирует строку формата quoted-printable в 8-битовую строку

Описание

```
string imap_qprint(string string);
```

Конвертирует строку формата quoted-printable в 8-битовую строку.
Возвращает 8-битовую (бинарную) строку.

imap_8bit

Конвертирует 8-битовую строку в формат quoted-printable

Описание

```
string imap_8bit(string string);
```

Конвертирует 8-битовую строку в формат quoted-printable. Возвращает строку в формате quoted-printable.

imap_binary

Конвертирует 8-битную строку в формат base64

Описание

```
string imap_binary(string string);
```

Конвертирует 8-битную строку в формат base64. Возвращает строку в формате base64.

imap_scanmailbox

Читает список почтовых ящиков, проводит поиск в названиях ящиков

Описание

```
array imap_scanmailbox(int imap_stream, string string);
```

Возвращает массив, содержащий имена почтовых ящиков, которые имеют строку string в названии.

imap_mailboxmsginfo

Получает информацию о текущем почтовом ящике

Описание

```
array imap_mailboxmsginfo(int imap_stream);
```

Возвращает информацию о текущем почтовом ящике. False — в случае неудачи.

Функция ***imap_mailboxmsginfo()*** проверяет статус текущего почтового ящика на сервере и возвращает информацию в объекте со следующими свойствами:

- **Date** ..... дата сообщения;
- **Driver** ..... драйвер;
- **Mailbox** ..... название почтового ящика;
- **Nmsgs** ..... количество сообщений;
- **Recent** ..... количество недавно пришедших сообщений;
- **Unread** ..... количество непрочитанных сообщений;
- **Size** ..... размер почтового ящика.

imap_rfc822_write_address

Возвращает правильно отформатированный email-адрес

Описание

```
string imap_rfc822_write_address(string mailbox, string host, string
personal);
```

Возвращает правильно отформатированный email-адрес по данному почтовому ящику, хосту и персональной информации.

imap_rfc822_parse_adrlist

Проводит разбор адресной строки

Описание

```
string imap_rfc822_parse_adrlist(string address, string default_host);
```

Эта функция разбирает адресную строку и для каждого адреса возвращает массив объектов. Есть 4 типа объектов:

- **mailbox** ..... название почтового ящика (имя пользователя);
- **host** ..... название хоста;
- **personal** ..... личное имя;
- **adl** ..... путь к домену-источнику.

imap_setflag_full

Устанавливает флаги на сообщения

Описание

```
string imap_setflag_full(int stream, string sequence, string flag,
string options);
```

Эта функция заставляет добавить указанный флаг к набору флагов сообщения в указанной последовательности.

imap_clearflag_full

Очищает флаги сообщения

Описание

```
string imap_clearflag_full(int stream, string sequence, string flag,
string options);
```

Эта функция заставляет удалить флаги из набора флагов сообщения в указанной последовательности.

imap_sort

Сортирует сообщения в текущем почтовом ящике

Описание

```
string imap_sort(int stream, int criteria, int reverse, int options);
```

Возвращает массив номеров сообщений рассортированных по данному параметру.

Rev должен быть равен 1 в том случае, если нужна сортировка в обратном порядке.

Критерии сортировки (должен быть указан только один):

- **SORTDATE** ..... по дате сообщения;
- **SORTARRIVAL** ..... по дате поступления;
- **SORTFROM** ..... по полю FROM;
- **SORTSUBJECT** ..... по теме сообщения;
- **SORTTO** ..... по полю TO;
- **SORTCC** ..... по полю CC;
- **SORTSIZE** ..... по размеру.

Опции — битовая маска из:

- SE_UID ..... возвратить идентификаторы вместо номеров последовательности;
- SE_NOPREFETCH ..... не извлекать заранее найденные сообщения.

imap_fetchheader

Возвращает заголовок сообщения

Описание

```
stringimap_fetchheader(int imap_stream, int msgno, int flags);
```

Эта функция заставляет извлечь полный, неотфильтрованный заголовок указанного сообщения в формате RFC 822 как текстовую строку.

Опции:

- **FT_UID** ..... msgno является UID'ом;
- **FT_INTERNAL** ..... возвращаемая строка записана во внутреннем формате без попыток канонизировать ее с помощью CRLF;
- **FT_PREFETCHTEXT RFC822.** .... текст должен быть предварительно разобран. Это поможет избежать эстренных задержек если требуется извлечь полный текст сообщения (например, в операции «сохранить в локальном файле»).

imap_uid

Эта функция возвращает UID по данному номеру сообщения в последовательности

Описание

```
String imap_uid(string mailbox, int msgno);
```

Эта функция возвращает UID по данному номеру сообщения в последовательности.

imap_rfc822_parse_headers

Возвращает набор различных заголовков

Описание

```
object imap_rfc822_parse_headers (string headers [, string defaulthost])
```

imap_getmailboxes

Читает список почтовых ящиков и возвращает информацию о них

Описание

```
array imap_getmailboxes (int imap_stream, string ref, string pattern)
```

imap_getsubscribed

Список всех почтовых ящиков, на которые существует подписка

Описание

```
array imap_getsubscribed (int imap_stream, string ref, string pattern)
```

imap_msgno

Номер сообщения по заданному UID

Описание

```
int imap_msgno (int imap_stream, int uid)
```

imap_search

Возвращает массив сообщений, удовлетворяющих заданным критериям

Описание

```
array imap_search (int imap_stream, string criteria, int flags)
```

imap_last_error

Возвращает последнюю ошибку IMAP

Описание

```
string imap_last_error (void)
```

imap_errors

Список всех ошибок IMAP

Описание

```
array imap_errors (void)
```

imap_alerts

Список всех предупреждений IMAP

Описание.

```
array imap_alerts (void)
```

imap_status

Статус почтового ящика (не текущего)

Описание

```
object imap_status (int imap_stream, string mailbox, int options)
```

imap_utf7_decode

Декодирует строку из UTF-7

Описание

```
string imap_utf7_decode (string text)
```


imap_utf7_encode

Преобразует 8-битовые данные в UTF-7

Описание

```
string imap_utf7_encode (string data)
```

imap_utf8

Преобразует текст в UTF8

Описание

```
string imap_utf8 (string text)
```

imap_fetch_overview

Читает информацию заголовков указанного сообщения

Описание

```
array imap_fetch_overview (int imap_stream, string sequence [, int
flags])
```

imap_mime_header_decode

Декодирует заголовки типов MIME

Описание

```
array imap_header_decode (string text)
```

imap_mail_compose

Создает сообщение MIME с использованием тела сообщения при заданном конверте

Описание

```
string imap_mail_compose (array envelope, array body)
```

Пример:

```
<?php

$envelope["from"]="musone@afterfive.com";
$envelope["to"]="musone@darkstar";
$envelope["cc"]="musone@edgeglobal.com";

$part1["type"]=TYPEMULTIPART;
$part1["subtype"]="mixed";
```

```
$filename="/tmp/imap.c.gz";
$fp=fopen($filename,"r");
$contents=fread($fp,filesize($filename));
fclose($fp);

$part2["type"]=TYPEAPPLICATION;
$part2["encoding"]=ENCBINARY;
$part2["subtype"]="octet-stream";
$part2["description"]=basename($filename);
$part2["contents.data"]=$contents;
$part3["type"]=TYPETEXT;
$part3["subtype"]="plain";
$part3["description"]="description3";
$part3["contents.data"]="contents.data3\n\n\n\t";

$body[1]=$part1;
$body[2]=$part2;
$body[3]=$part3;

echo nl2br(imap_mail_compose($envelope,$body));
?>
```

imap_mail

Посылает почтовое сообщение

Описание

```
string imap_mail (string to, string subject, string message
[, string additional_headers [, string cc [, string bcc [, string
```


Функции для работы с почтой

mail

Шлет почту

Описание

```
bool mail(string to, string subject, string message, string
additional_headers);
```

Функция ***mail()*** позволяет отсылать почту. ***Mail()*** автоматически посылает сообщение, содержащееся в message адресату, указанному в поле to. Несколько получателей могут быть указаны в поле to в виде строки с адресами, разделенными пробелами.

Пример 1. Посылка почты:

```
mail("rasmus@lerdorf.on.ca", "Моя тема", "Строка 1\nСтрока 2\nСтрока 3");
```

Если задан четвертый строковый аргумент, он автоматически вставляется в конец заголовка. обычно это используется при добавлении дополнительных полей в заголовок. Несколько дополнительных полей разделяются символом новой строки.

Пример 2. Посылка почты с дополнительными полями заголовка:

```
mail("ssb@guardian.no", "the subject", $message, "From:
webmaster@$SERVER_NAME\nReply-To: webmaster@$SERVER_NAME\nX-Mailer:
PHP/". phpversion());
```

ezmlm_hash

Определяет скрытое значение, которое используется в EZMLM

Описание

```
int ezmlm_hash (string addr)
```

Пример:

```
$user = "putin@koegde.koekak.ru";
$hash = ezmlm_hash ($user);
$query = sprintf ("INSERT INTO sample VALUES (%s, '%s')", $hash,
$user);
$db->query($query);
```

Математические функции

Математические константы

В табл. С.2 представлены математические константы, используемые в PHP.

Таблица С.2
Математические константы

Константа	Значение	Определение
M_PI	3.14159265358979323846	p (Pi)
M_E	2.7182818284590452354	E
M_LOG2E	1.4426950408889634074	log_2 e
M_LOG10E	0.43429448190325182765	log_10 e
M_LN2	0.69314718055994530942	log_e 2
M_LN10	2.30258509299404568402	log_e 10
M_PI_2	1.57079632679489661923	pi/2
M_PI_4	0.78539816339744830962	pi/4
M_1_PI	0.31830988618379067154	1/pi
M_2_PI	0.63661977236758134308	2/pi
M_SQRTPI	1.77245385090551602729	sqrt(pi) [CVS]
M_2_SQRTPI	1.12837916709551257390	2/sqrt(pi)
M_SQRT2	1.41421356237309504880	sqrt(2)
M_SQRT3	1.73205080756887729352	sqrt(3) [CVS]
M_SQRT1_2	0.70710678118654752440	1/sqrt(2)
M_LNPI	1.14472988584940017414	log_e(pi) [CVS]
M_EULER	0.57721566490153286061	Euler constant [CVS]

Функции

Abs

Абсолютная величина

Описание

```
mixed abs(mixed number);
```

Возвращает абсолютную величину (модуль) числа. Если число с плавающей запятой, то также возвращает число с плавающей запятой.

Acos

Арккосинус

Описание

```
float acos(float arg);
```

Возвращает арккосинус аргумента в радианах.

См. также ***asin(), atan().***

Asin

Арксинус

Описание

```
float asin(float arg);
```

Возвращает арксинус аргумента в радианах.

См. также ***acos(), atan().***

Atan

Арктангенс

Описание

```
float atan(float arg);
```

Возвращает арктангенс аргумента в радианах.

См. также ***acos(), atan().***

Atan2

Арктангенс от двух переменных

Описание

```
float atan2(float y, float x);
```

Эта функция вычисляет арктангенс от двух переменных x и y. Аналогично вычислению арктангенса y/x, за исключением того, что знаки обоих аргументов используются для определения сектора результата.

Функция возвращает результат в радианах, находящихся между −PI и PI (включительно).

См. также ***acos(), atan().***

base_convert

Конвертирует число между произвольными основаниями (системами счисления)

Описание

```
strin base_convert(string number, int frombase, int tobase);
```

Возвращает строку, содержащую number, представленного по основанию tobase. Основание, в котором дается число which number указывается в frombase. Основания frombase и tobase должны находиться в диапазоне от 2 до 36 включительно. Цифры в числах с основанием выше, чем 10 будут представлены буквами a...z, со значениями a = 10, b = 11 и z = 36.

Пример base_convert():

```
$binary = base_convert($hexadecimal, 16, 2);
```

BinDec

Двоичное в десятичное

Описание

```
int bindec(string binary_string);
```

Возвращает десятичный эквивалент двоичного числа, представленного аргументом binary_string.

OctDec конвертирует двоичное число в десятичное. Наибольшее число, которое может быть сконвертировано, равно 31 битам или 2147483647 в десятичном виде.

См. также ***decbin().***

Ceil

Округлить дробную часть наверх

Описание

```
int ceil(float number);
```

Возвращает следующее наивысшее целое значение number. Использование ***ceil()*** на целых числах — абсолютная трата времени.

См. также ***floor(), round().***

Cos

Косинус

Описание

```
float cos(float arg);
```

Возвращает косинус аргумента в радианах.

См. также ***sin(), tan().***

DecBin

Десятичное в двоичное

Описание

```
string decbin(int number);
```

Возвращает строку, содержащую двоичное представление аргумента number. Наибольшее число, которое может быть сконвертировано, равно 2147483647 в десятичном виде или 31 бит.

См. также ***bindec().***

DecHex

Десятичное в шестнадцатиричное

Описание

```
string dechex(int number);
```

Возвращает строку, содержащую шестнадцатиричное представление аргумента number. Наибольшее число, которое может быть сконвертировано, равно 2147483647 в десятичном виде или «7fffffff» в шестнадцатиричном.

См. также ***hexdec().***

DecOct

Десятичное в восьмиричное

Описание

```
string decoct(int number);
```

Возвращает строку, содержащую восьмиричное представление аргумента number. Наибольшее число, которое может быть сконвертировано, равно 2147483647 в десятичном виде или «17777777777» — в восьмиричном.

См. также ***octdec()***.

Exp

e в степени... (экспонента)

Описание

```
float exp(float arg);
```

Возвращает число e, возведенное в степень arg.

См. также ***pow()***.

Floor

Округляет дробную часть вниз

Описание

```
int floor(float number);
```

Возвращает следующее нижележащее значение после number. Использование ***floor()*** на целых числах — абсолютная потеря времени.

См. также ***ceil(), round()***.

getrandmax

Показывает наибольшую возможную случайную величину

Описание

```
int getrandmax(void );
```

Возвращает максимальную величину, которая может быть возвращена вызовом функции ***rand()***.

См. также ***rand(), srand() mt_rand(), mt_srand(), mt_getrandmax()***.

HexDec

Шестнадцатиричное в десятичное

Описание

```
int hexdec(string hex_string);
```

Возвращает десятичный эквивалент числа, представленного аргументом hex_string. HexDec конвертирует шестнадцатиричную строку в десятичное число. Наибольшее число, которое может быть сконвертировано, равно 7fffffff в шестнадцатиричном виде или 2147483647 — в десятичном.

См. также ***dechex()***.

Log

Натуральный логарифм

Описание

```
float log(float arg);
```

Возвращает натуральный логарифм от аргумента arg.

Log10

Логарифм по основанию 10

Описание

```
float log10(float arg);
```

Возвращает логарифм по основанию 10 от аргумента arg.

max

Находит максимум

Описание

```
mixed max(mixed arg1, mixed arg2, mixed argn);
```

Max() возвращает наибольшее число из перечисленных в параметрах.

Если первый элемент является массивом, ***max()*** возвращает максимальную величину массива. Если первый параметр — целое, строка или типа double, следует использовать как минимум два параметра, и в этом случае ***max()*** возвращает наибольшее из этих величин. Вы можете сравнивать неограниченное количество значений.

Если одна или более величин типа double, все остальные величины будут обращены double, и, соответственно, возвратится число типа double. Если ни одно из чисел не является double, то все будут обращены в целые и возвратится целое число.

min

Находит минимум

Описание

```
mixed min(mixed arg1, mixed arg2, mixed argn);
```

Min() возвращает наименьшее значение из указанных в аргументах.

Если первый параметр — массив, ***min()*** возвратит наименьшую величину массива. Если первый параметр — целое число, строка или double, следует указать минимум два параметра и ***min()*** возвратит наименьшую из них величину. Вы можете сравнивать неограниченно количество величин.

Если одна или более величин типа double, все остальные величины будут обращены double, и, соответственно, возвратится число типа double. Если ни одно из чисел не является double, то все будут обращены в целые и возвратится целое число.

mt_rand

Генерирует наилучшее случайное число

Описание

```
int mt_rand([int min], [int max]);
```

Множество генераторов случайных чисел, написанных на старой библиотеке libcs, имеют неясные или неизвестные характеристики, и к тому же, медленны. По умолчанию с функцией ***rand()*** PHP использует генератор случайных чисел, написанный на libc. Функция ***mt_rand()*** является его полной заменой. Она использует генератор случайных чисел с известными характеристиками (Mersenne Twister), который производит случайные числа, пригодные для использования в криптографии и работает в четыре раза быстрее, чем средняя скорость, которую обеспечивает libc. Домашнюю страницу Mersenne Twister'a вы можете найти по адресу: http://www.math.keio.ac.jp/~matumoto/emt.html, а оптимизированную версию исходных текстов на http://www.scp.syr.edu/~marc/hawk/twister.html.

Если функция вызывается без необязательных аргументов min и max, ***mt_rand()*** возвращает псевдослучайное число между 0 и RAND_MAX. Если вы хотите получить случайное число между 5 и 15 (включительно), то можно использовать следующий вызов функции: mt_rand(5,15).

Не забудьте инициализировать генератор случайных чисел перед использованием функции ***mt_srand()***.

См. также ***mt_srand(), mt_getrandmax(), srand(), rand(), getrandmax().***

mt_srand

Инициализирует лучший генератор случайных чисел

Описание

```
void mt_srand(int seed);
```

Инициализирует генератор случайных чисел значением seed.

Инициализируется количеством микросекунд, истекших с последней «целой» секунды:

```
mt_srand((double)microtime()*1000000);
$randval = mt_rand();
```

См. также ***mt_rand(), mt_getrandmax(), srand(), rand(), getrandmax().***

mt_getrandmax

Показывает наибольшее возможное случайное число

Описание

```
int mt_getrandmax(void );
```

Возвращает максимальную величину, которая может быть возвращена вызовом функции ***mt_rand()***.

См. также ***mt_rand(), mt_srand() rand(), srand(), getrandmax().***

number_format

Форматирует число с сгруппированными тысячами

Описание

```
string number_format(float number, int decimals, string dec_point,
string thousands_sep);
```

Number_format() возвращает форматированную версию числа number. Эта функция принимает один, два или четыре параметра (не три):

Если дан только один параметр, число number будет отформатировано без десятичных цифр, но с запятой («,») между каждой группой тысяч.

Если дано два параметра, число number will будет отформатировано с десятичным знаком decimals с точкой («.») впереди и запятой («,») между каждой группой тысяч.

Если даны все четыре параметра, то число number будет отформатировано с десятичным знаком decimals, dec_point вместо точки («.») перед десятичным знаком и thousands_sep вместо запятой («,») между каждой группой тысяч.

OctDec

Восьмиричное в десятичное

Описание

```
int octdec(string octal_string);
```

Возвращает десятичный эквивалент восьмиричного числа, представленного аргументом octal_string. OctDec конвертирует восьмиричное число в десятичное. Максимальное число, которое может быть сконвертировано, равно 17777777777 или 2147483647 в десятичном виде.

См. также ***decoct()***.

pi

Величина pi

Описание

```
double pi(void );
```

Возвращает аппроксимированное значение pi.

pow

Степень числа

Описание

```
float pow(float base, float exp);
```

Возвращает base, возведенное в степень exp.

См. также ***exp()***.

rad2deg

Конвертирует число в радианах в число в градусах

rand

Генерирует случайную величину

Описание

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

Если функция вызывается без необязательных параметров min и max, ***rand()*** возвращает псевдослучайную величину между 0 и RAND_MAX. При желании получить случайное число между 5 и 15 (включительно), используйте rand(5,15).

Не следует забывать инициализировать генератор случайных чисел перед использованием ***srand()***.

См. также ***srand(), getrandmax(), mt_rand(), mt_srand(), mt_getrandmax()***.

round

Округляет число с плавающей запятой

Описание

```
double round(double val);
```

Возвращает округленную величину val.

```
$foo = round( 3.4 );    // $foo == 3.0
$foo = round( 3.5 );    // $foo == 4.0
$foo = round( 3.6 );    // $foo == 4.0
```

См. также ***ceil(), floor()***.

Sin

Синус

Описание

```
float sin(float arg);
```

Возвращает синус аргумента.

См. также ***cos(), tan()***.

Sqrt

Квадратный корень

Описание

```
float sqrt(float arg);
```

Возвращает квадратный корень аргумента.

srand

Инициализирует генератор случайных чисел

Описание

```
void srand(int seed);
```

Инициализирует генератор случайных чисел значением seed.

```
//Генератор инициализируется числом микросекунд, истекших с послед-
дней "целой" секунды
srand((double)microtime()*1000000);
$randval = rand();
```

См. также ***rand(), getrandmax(), mt_rand(), mt_srand(), mt_getrandmax().***

Tan

Тангенс

Описание

```
float tan(float arg);
```

Возвращает тангенс аргумента.

См. также ***sin(), cos().***

Функции для работы с сетью

fsockopen

Открывает соединение с узлом в Интернете или Unix-системой через socket

Описание

```
int fsockopen(string hostname, int port, int [errno], string [errstr]);
```

Открывает сокетное соединение с доменом Internet по адресу hostname на порт port и возвращает файловый указатель, который может использоваться функциями ***fgets()***, ***fgetss()***, ***fputs()*** и ***fclose()***. Если вызов завершается неудачей, он возвращает FALSE и если указаны дополнительные аргументы errno и errstr, то они будут использованы, чтобы указать фактическую системную ошибку, которая случилась на системном уровне при вызове ***connect()***. Если возвращенное errno — 0, но функция вернула FALSE, это признак того, что ошибка произошла перед вызовом ***connect()***. Это наиболее вероятно, из-за проблемы инициализации сокета. Имейте в виду, что аргументы errno и errstr необязательно должны упоминаться.

Если port — 0 и ОС поддерживает доменные сокеты Unix (domain sockets), hostname будет использован для подключения в качестве filename доменного сокета Unix.

По умолчанию сокет откроется в режиме blocking mode. Вы можете переключить его в non-blocking mode используя ***set_socket_blocking()***.

Пример:

```
$fp = fsockopen("www.php.net", 80, &$errno, &$errstr);
if(!$fp) {
  echo "$errstr ($errno)<br>\n";
} else {
  fputs($fp,"GET / HTTP/1.0\n\n");
  while(!feof($fp)) {
    echo fgets($fp,128);
  }
  fclose($fp);
}
```

set_socket_blocking

Устанавливает blocking/non-blocking режимы работы сокета

Описание

```
int set_socket_blocking(int socket descriptor, int mode);
```

Если mode отсутствует, данный дескриптор сокета переключится на non-blocking режим, а если присутствует, то переключится на blocking режим. Это влияет на вызовы типа ***fgets()***, который читает из сокета. В non-blocking режиме ***fgets()*** вызов всегда будет возвращаться немедленно, в то время как в blocking режиме он ждет данные, чтобы стать доступным на сокете.

gethostbyaddr

Получает имя хоста Internet, соответствующее данному IP-адресу

Описание

```
string gethostbyaddr(string ip_address);
```

Возвращает имя хоста Internet определенного аргументом ip_address. Если происходит ошибка, возвращается ip_address.

См. также ***gethostbyname().***

gethostbyname

Получает IP-адрес, соответствующий заданному имени хоста Интернет

Описание

```
string gethostbyname(string hostname);
```

Возвращает IP-адрес хоста, указанного аргументом hostname.

См. также ***gethostbyaddr().***

gethostbynamel

Получает список IP-адресов, соответствующих заданному имени хоста Интернет

Описание

```
array gethostbynamel(string hostname);
```

Возвращает список IP-адресов, на которых разрешен хост Internet, указанный аргументом hostname.

См. также ***gethostbyname(), gethostbyaddr(), checkdnsrr(), getmxrr().***

checkdnsrr

Проверяет записи DNS, соответствующие заданному хосту или IP-адресу

Описание

```
int checkdnsrr(string host, string [type]);
```

Ищет в DNS записи типа type, соответствующие аргументу host. Возвращает true, если обнаруживаются какие-либо записи. Возвращает false, если не обнаружены никакие записи или если произошла ошибка.

Type может быть любым из значений: A, MX, NS, SOA, PTR, CNAME, или ANY. По умолчанию — MX.

Host может или быть адресом IP или именем хоста.

См. также ***getmxrr(), gethostbyaddr(), gethostbyname(), gethostbynamel().***

getmxrr

Получает MX-записи, соответствующие заданному имени хоста

Описание

```
int getmxrr(string hostname, array mxhosts, array [weight]);
```


Ищет в DNS MX-записи, соответствующие hostname. Возвращает true если записи найдены, false — если записей не найдено или произошла ошибка.

Список MX-записей может размещаться в массиве mxhosts. Если задано weight массива, то он может быть заполнен собранной информацией.

См. также ***checkdnsrr(), gethostbyname(), gethostbynamel(), gethostbyaddr().***

openlog

Открывает соединение к системным логам (system logger)

Описание

```
int openlog(string ident, int option, int facility);
```

Openlog() открывает для программы соединение с system logger. Строка ident добавляется к каждому сообщению. Значение для option и facility даются в следующем разделе. Использование ***openlog()*** не обязательно. Это может быть автоматически вызвано вызовом ***syslog()***, если необходимо, в этом случае ident по умолчанию будет false.

См. также ***syslog(), closelog().***

syslog

Генерирует системное регистрационное сообщение

Описание

```
int syslog(int priority, string message);
```

Syslog() генерирует регистрационное сообщение, для system logger. Priority — комбинация легкости и уровня, значения для которых даются в следующем разделе. А другой аргумент является отсылаемым сообщением, кроме того, два символа %m заменятся строкой сообщения ошибки (strerror) соответствующей значению errno.

Больше информации о средствах syslog можно обнаружить на man-страницах для syslog в Unix-системах. В WindowsNT, сервис syslog эмулируется использованием Event Log.

closelog

Закрывает соединение с system logger

Описание

```
int closelog(void);
```

Closelog() закрывает дескриптор, используемый для записи в system logger. Использование ***closelog()*** необязательно.

debugger_on

Включает внутренний PHP отладчик

Описание

```
int debugger_on(string address);
```

Включает внутренний PHP-отладчик, соединяя его с address.

debugger_off

Запрещает внутренний PHP отладчик

Описание

```
int debugger_off(void);
```

Выключает внутренний PHP-отладчик.

getprotobyname

Определяет номер протокола по названию протокола

Описание

```
int getprotobyname (string name)
```

См. также ***getprotobynumber().***

getprotobynumber

Определяет название протокола по номеру протокола

Описание

```
string getprotobynumber (int number)
```

См. также ***getprotobyname().***

getservbyname

Определяет номер порта, связанный со службой Интернет и соответствующим протоколом

Описание

```
int getservbyname (string service, string protocol)
```

См. также ***getservbyport().***

getservbyport

Определяет службу Интернет по номеру порта и протоколу

Описание

```
string getservbyport (int port, string protocol)
```

Протокол — либо TCP, либо UDP.

См. также ***getservbyname().***

pfsockopen

Открывает постоянное соединение сокета

Описание

```
int pfsockopen (string hostname, int port [, int errno [, string errstr
[, int timeout]]])
```

Эта функция работает в точности так же, как и функция ***fsockopen()*** с той лишь разницей, что образованная связь не закрывается после завершения выполнения скрипта, а остается открытой.

ip2long

Преобразует Интернет-адрес в формате с точками в адрес типа long

Описание

```
int ip2long (string ip_address)
```

Пример с функцией Ip2long():

```
<?
$ip = gethostbyname("www.php.net");
$out = "Следубщие адреса URL эквивалентны:<br>\n";
$out.= "http://localhost/, http://".$ip."/, and http://
".ip2long($ip)."/<br>\n";
echo $out;
?>
```

См. также ***long2ip(), рис. С.10.***

long2ip

Преобразует Интернет-адрес в стандартный формат с точками

Описание

```
string long2ip (int proper_address)
```

См. также ***ip2long().***

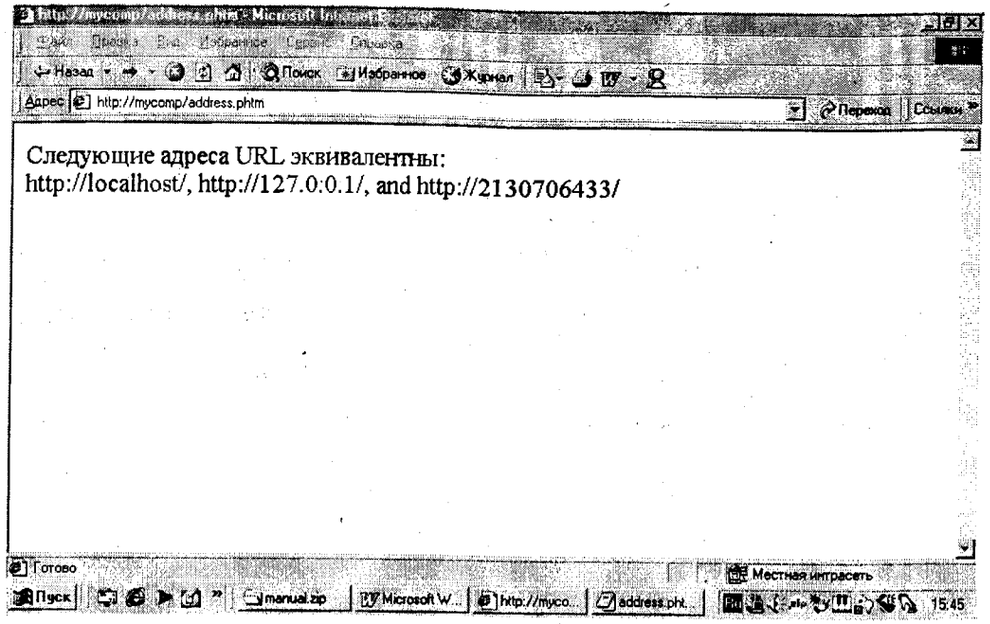


Рис. С.10. *Преобразование ip-адресов в различный формат*

Функции определения и установки параметров интерпретатора PHP

error_log

Отослать сообщение об ошибке

Описание

```
int error_log (string message, int message_type, string [ destination
], string [ extra_headers ]);
```

Посылает сообщение об ошибке на сервер или в файл. Первый параметр, message, определяет сообщение об ошибке, которое должно быть отослано. Второй параметр, message_type, определяет способ передачи сообщения:

Message = 0 ..... посылает в системный лог PHP, используя механизм системной записи операционной системы или в файл;

Message = 1 ..... посылается по электронной почте, адрес определен в destination. Только для этого типа сообщений используется параметр extra_headers. Этот тип использует ту же внутреннюю функцию, что и ***Mail()***;

message = 2 ..... посылается через соединение PHP отладки. Эта опция доступна только тогда, когда включена удаленная отладка ошибок, destination определяет имя машины или IP-адрес и номер порта сокета, получающего отладочную информацию;

message = 3..... добавляется к файлу, определенному в destination.

Пример с функцией error_log():

```
// Послать сообщение в системный лог, если не получено соединение с
базой данных
if (!Ora_Logon($username, $password)) {
     error_log("Oracle database not available!", 0);
}

// Информировать администратора по почте об ошибке выделения FOO
if (!($foo = allocate_new_foo()) {
     error_log("Big trouble, we're all out of FOOs!", 1,
                "operator@mydomain.com");
}

// другие способы вызова error_log():
error_log("You messed up!", 2, "127.0.0.1:7000");
error_log("You messed up!", 2, "loghost");
error_log("You messed up!", 3, "/var/tmp/my-errors.log");
```

Error_reporting

Устанавливает уровень сообщений об ошибках PHP

Описание

```
int error_reporting(int [ level ]);
```

Устанавливает уровень сообщений об ошибках PHP и возвращает старый уровень. Уровень сообщений об ошибках задается битовой маской и использует следующие значения.

Таблица С.3
Битовые значения для функции error_reporting() (параметр level)

Значение	Внутреннее имя
1	E_ERROR
2	E_WARNING
4	E_PARSE
8	E_NOTICE
16	E_CORE_ERROR
32	E_CORE_WARNING

assert

Проверяет выражение, является ли его значение равным false

Описание

```
int assert (string|bool assertion)
```

assert-options

Устанавливает различяные флаги для проверки

Описание

```
mixed assert_options (int what [, mixed value])
```

См. табл. С.4.

Таблица С.4
assert options

Парметр	Ini-параметр	По умолчанию	Описание
ASSERT_ACTIVE	assert.active	1	Учитывать значение assert()
ASSERT_WARNING	assert.warning	1	Разрешить вывод PHP-предупреждения для каждой ошибки
ASSERT_BAIL	assert.bail	0	Прекратить выполнение при ошибке
ASSERT_QUIET_EVAL	assert.quiet_eval	0	Отменить вывод сообщения об ошибке при оценке выражения
ASSERT_CALLBACK	assert_callback	(null)	Пользовательская функция обработки ошибки

extension_loaded

Проверяет, загружено ли расширение

getenv

Возвращает значение внешней переменной

get_cfg_var

Получает значения конфигурации PHP

Описание

```
string get_cfg_var (string varname)
```

get_current_user

Получает владельца текущего скрипта

Описание

```
string get_current_user (void)
```

См. также ***getmyuid(), getmypid(), getmyinode(), getlastmod().***

get_magic_quotes_gpc

Получает текущие параметры magic quotes gpc

Описание

```
long get_magic_quotes_gpc (void)
```

См. также ***get_magic_quotes_runtime(), set_magic_quotes_runtime().***

get_magic_quotes_runtime

Получает текущую конфигурацию magic_quotes_runtime

Описание

```
long get_magic_quotes_runtime (void)
```

См. также ***get_magic_quotes_gpc(), set_magic_quotes_runtime().***

getlastmod

Получает время последней модификации страницы

Описание

```
int getlastmod (void)
```

См. также ***date(), getmyuid(), get_current_user(), getmyinode(), getmypid().***

getmyinode

Получает inode текущего скрипта

Описание

```
int getmyinode (void)
```

См. также ***getmyuid(), get_current_user(), getmypid(), getlastmod().***

getmypid

Получает идентификатор процесса PHP

Описание

```
int getmypid (void)
```

См. также ***getmyuid(), get_current_user(), getmyinode(), getlastmod().***

getmyuid

Получает идентификатор UID

Описание

```
int getmyuid (void)
```

См. также ***getmypid(), get_current_user(), getmyinode(), getlastmod()***

getrusage

Информация об использовании текущего ресурса

Описание

```
array getrusage ([int who])
```

Пример:

```
$dat = getrusage();
echo $dat["ru_nswap"];           # number of swaps
echo $dat["ru_majflt"];          # number of page faults
echo $dat["ru_utime.tv_sec"];  # user time used (seconds)
echo $dat["ru_utime.tv_usec"]; # user time used (microseconds)
```


phpcredits

Информация о PHP

Описание

```
void phpcredits (int flag)
```

См. табл. С.5.

Таблица С.5
Значения флагов

Флаг	Описание
CREDITS_ALL	Эквивалентно CREDITS_DOCS + CREDITS_GENERAL + CREDITS_GROUP + CREDITS_MODULES + CREDITS_FULLPAGE. (см. рис. Credits).
CREDITS_DOCS	Команда документации
CREDITS_FULLPAGE	Создать отдельную HTML страницу
CREDITS_GENERAL	Общая информация
CREDITS_GROUP	Разработчики ядра
CREDITS_MODULES	Авторы модулей расширений
CREDITS_SAPI	

См. также ***phpinfo(), phpversion(), php_logo_guid().***
См. рис. С.11.

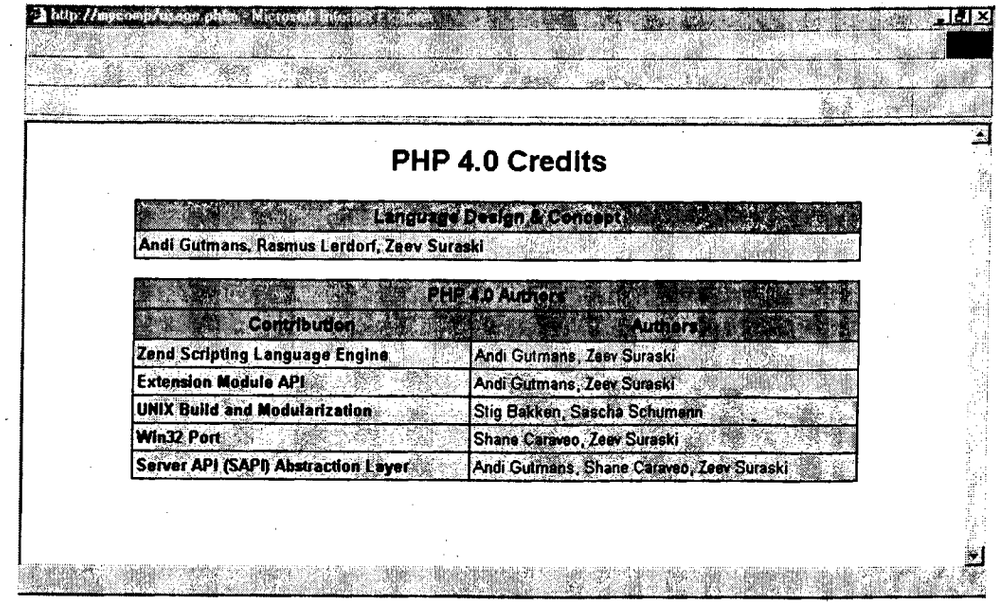


Рис. С.11. *Информация о PHP*

phpinfo

Большая коллекция информации о PHP

Описание

```
int phpinfo (void)
```

См. также ***phpversion(), phpcredits(), php_logo_guid()***

phpversion

Текущая версия PHP

Описание

```
string phpversion (void)
```

Пример:

```
// prints e.g. 'Current PHP version: 3.0rel-dev'
echo "Current PHP version: ".phpversion();
```

См. рис. С.12.

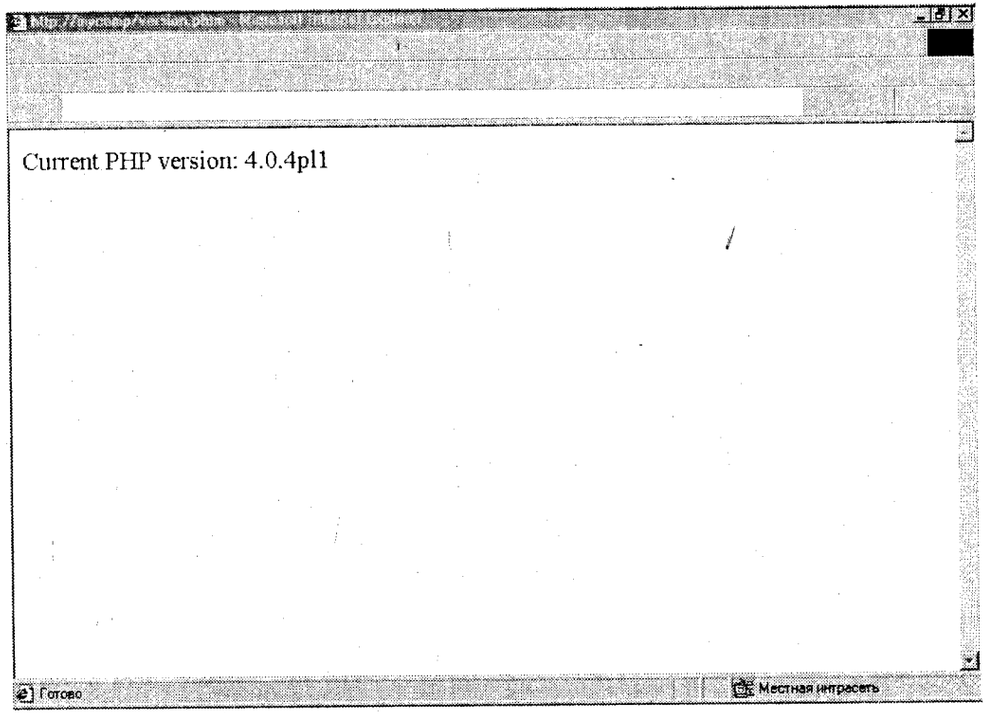


Рис. С.12. *Текущая версия PHP*

php_logo_guid

Получает логотип-идентификатор (см. рис. С.13)

Описание

```
string php_logo_guid (void)
```

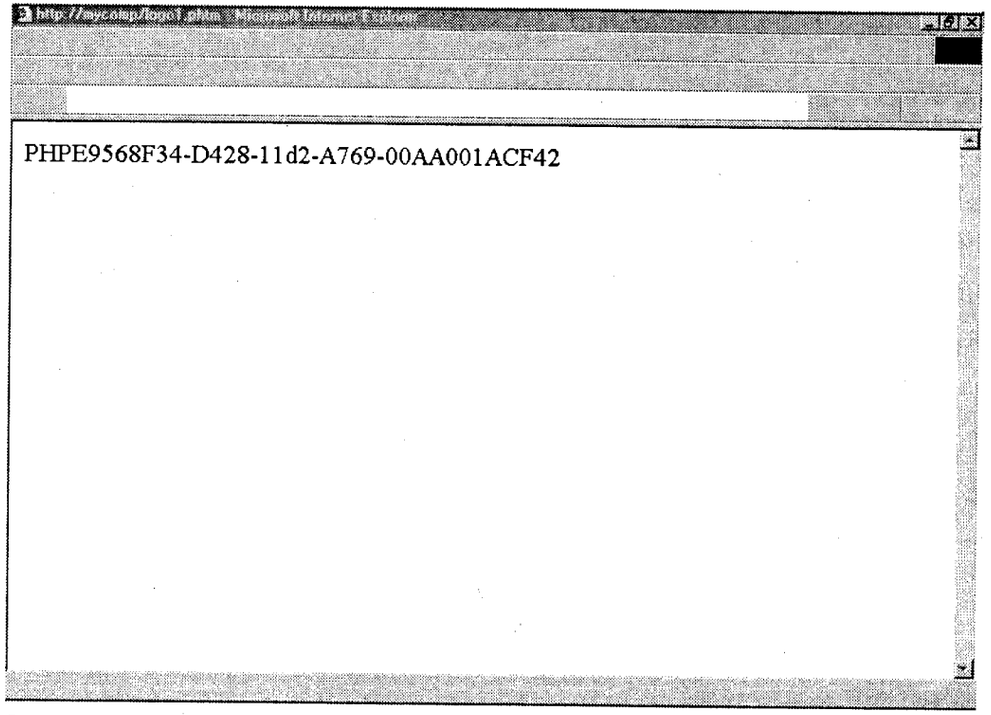


Рис. С.13. *Идентификатор PHP*

php_sapi_name

Получает тип интерфейса (см. рис. С.14)

Описание

```
string php_sapi_name (void)
```

Пример:

```
$inter_type = php_sapi_name();
if ($inter_type == "cgi") print "You are using CGI PHP\n"; else
print "You are not using CGI PHP\n";
```

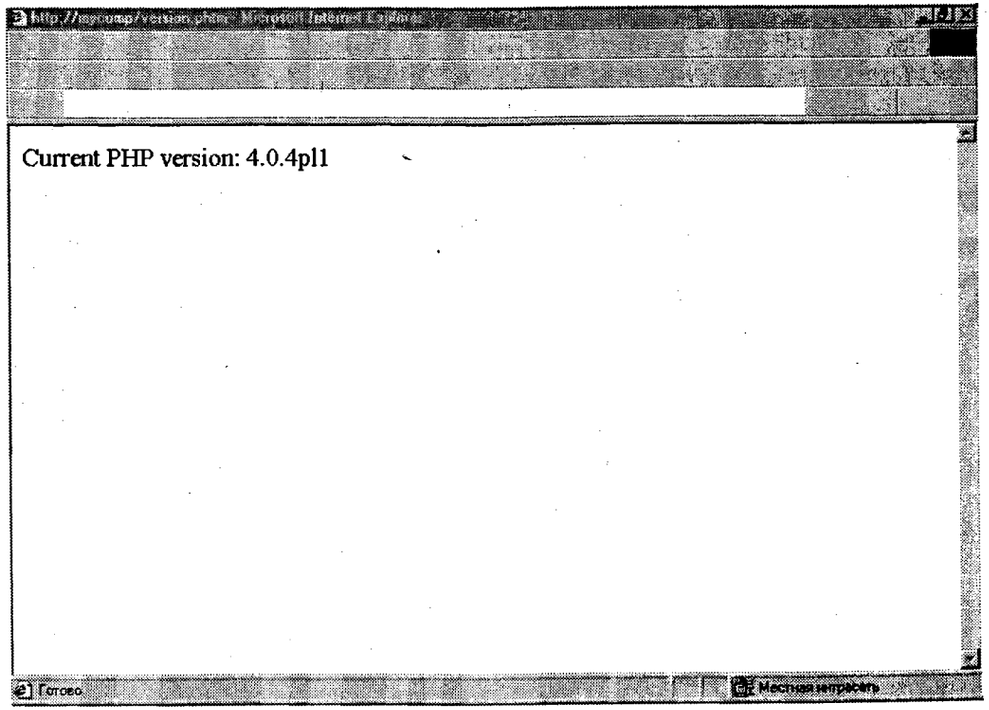


Рис. С.14. *Получаем тип интерфейса*

putenv

Задает значение переменной окружения

Описание

```
void putenv (string setting)
```

Пример:

```
putenv ("UNIQID=$uniqid");
```

set_magic_quotes_runtime

Устанавливает конфигурацию magic_quotes_runtime

Описание

```
long set_magic_quotes_runtime (int new_setting)
```

См. также ***get_magic_quotes_gpc(), get_magic_quotes_runtime().***

set_time_limit

Ограничивает максимальное время выполнения скрипта

Описание

```
void set_time_limit (int seconds)
```

zend_logo_guid

Получить идентификатор zend-logo

Описание

```
string zend_logo_guid (void)
```

См. рис С.15.

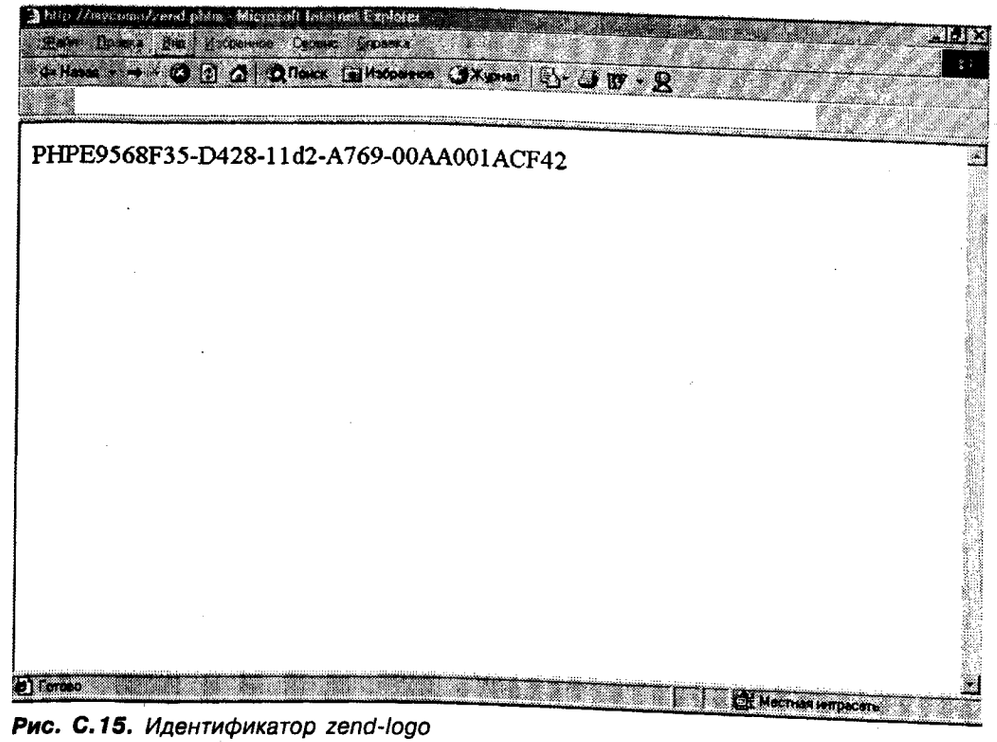


Рис. С.15. *Идентификатор zend-logo*

get_loaded_extensions

Возвращает массив с именами всех загруженных и скомпилированных модулей

Описание

```
array get_loaded_extensions (void)
```

Пример:

```
print_r (get_loaded_extensions());
```

Результат работы будет примерно таким (*см. рис. С.16*):

```
Array
(
    [0] => xml
    [1] => wddx
    [2] => standard
    [3] => session
    [4] => posix
    [5] => pgsql
    [6] => pcre
    [7] => gd
    [8] => ftp
    [9] => db
    [10] => Calendar
    [11] => bcmath
)
```

См. также ***get_extension_funcs().***

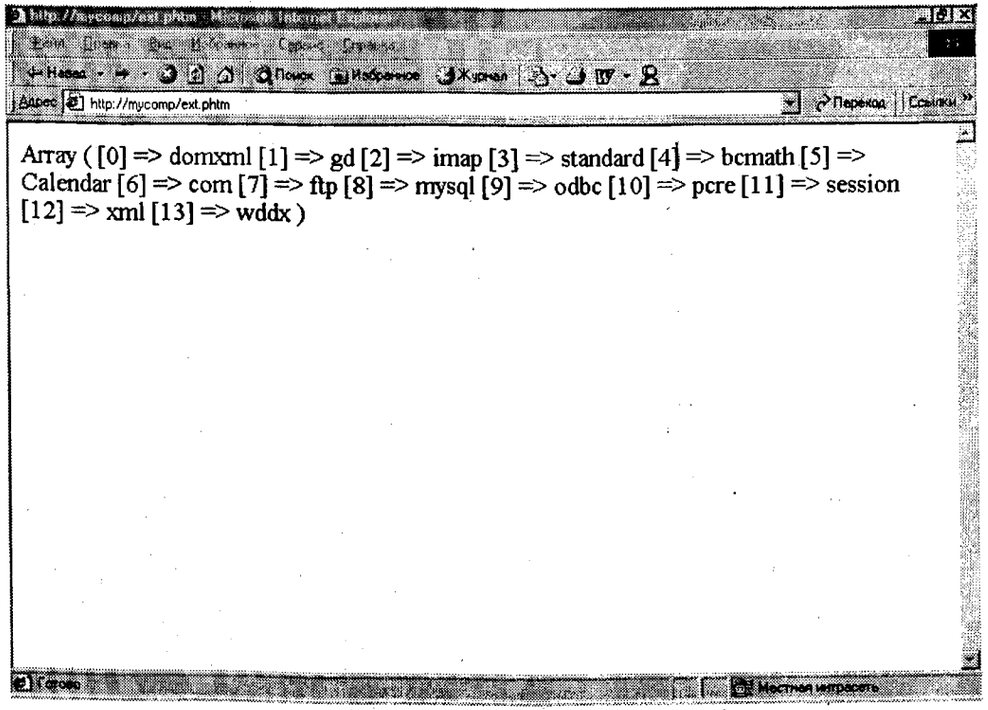


Рис. С.16. *Показаны загруженные модули*

get_extension_funcs

Дозвращает массив с именами функций модуля

Описание

```
array get_extension_funcs (string module_name)
```

Пример:

```
print_r (get_extension_funcs ("xml"));
print_r (get_extension_funcs ("gd"));
```

Результат работы см. на рис. С.17.

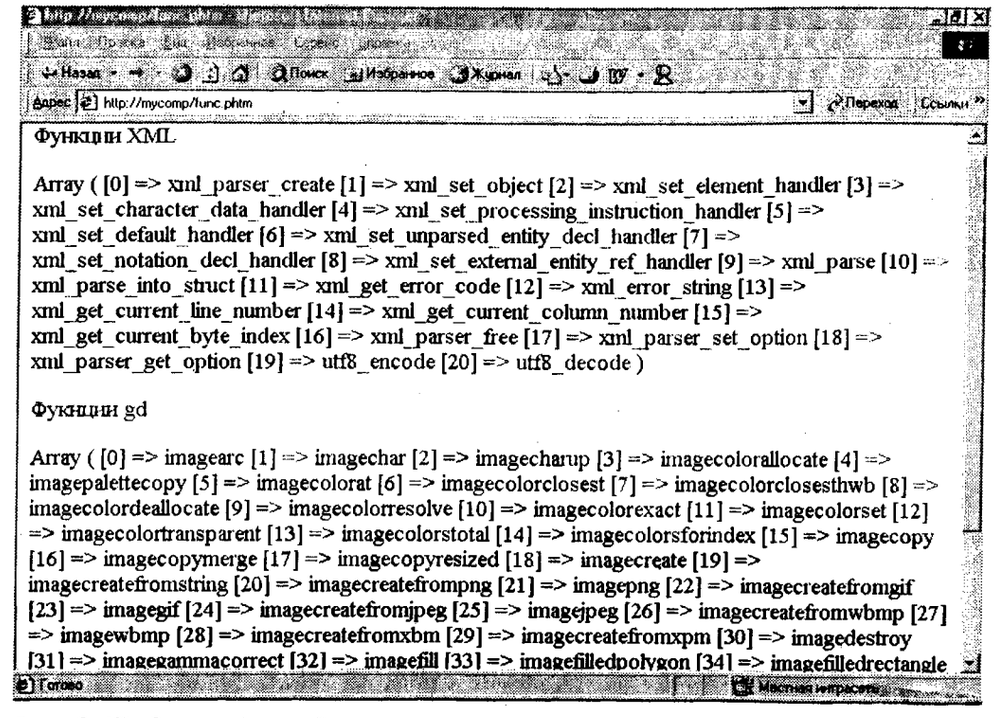


Рис. С.17. *Список функций для указанного расширения*

get_required_files

Возвращает массив, содержащий список имен файлов, требуемых для выполнения скрипта

Описание

```
array get_required_files (void)
```

Пример:

```
<?php
require_once ("local.php");
require_once ("../inc/global.php");
for ($i=1; $i<5; $i++) include "util".$i."php";
echo "Required_once files\n";
print_r (get_required_files());
echo "Included_once files\n";
print_r (get_included_files()); ?>
```

Результат работы будет примерно таким:

```
Required_once files
Array
(
    [local] => local.php
    [../inc/global] => /full/path/to/inc/global.php
)

Included_once files
Array
(
    [util1] => util1.php
    [util2] => util2.php
    [util3] => util3.php
    [util4] => util4.php
)
```

get_included_files

Возвращает список имен (массив) файлов, включенных в скрипт

Описание

```
array get_included_files (void)
```

См. также ***require_once(), include_once(), get_required_files().***

Функции выполнения системных программ

escapeshellcmd

Убирает метасимволы shell

Описание

```
string escapeshellcmd(string command);
```

EscapeShellCmd() убирает любые символы в строке, которые могут быть использованы в командном интерпретаторе как произвольные команды. Эту функцию нужно использовать для того, чтобы убедиться, что все данные введены правильно. Эту функцию лучше всего вставлять в функции ***exec()*** или ***system()***. Стандартное использование этой функции выглядит так:

```
system(EscapeShellCmd($cmd))
```

exec

Запуск внешней программы

Описание

```
string exec(string command, string [ array ], int [ return_var ]);
```

Exec() запускает программу из строки command, весь стандартный вывод отключен. Возвращает последнюю строку результата выполнения программы. Если вы хотите запустить команду и использовать все данные непосредственно без всякого вмешательства других программ, то используйте функцию ***PassThru()***.

Если параметр array установлен, то указанный массив будет заполнен выводом из программы. Помните, если массив уже содержит данные, то ***exec()*** добавляет свои данные. Если вы не хотите, чтобы функция добавляла данные, вызывайте ***unset()*** для массива перед использованием ***exec()***.

Если параметр return_var установлен наряду с параметром array, то в него записывается результат выполнения команды.

Обратите внимание, что если ваша функция будет использовать данные из ввода пользователей, то надо использовать ***EscapeShellCmd()*** для того, чтобы пользователи не смогли запустить произвольные программы.

См. также ***system(), PassThru(), popen(), EscapeShellCmd().***

system

Запуск внешней программы с выводом результата

Описание

```
string system(string command, int [ return_var ]);
```

System() такая же, как и C-версия этой функции для запуска command и вывода результата. Если используется второй параметр, то в него записывается результат выполнения команды.

Обратите внимание, что если ваша функция будет использовать данные из ввода пользователей, то надо использовать ***EscapeShellCmd()*** для того, чтобы пользователи не смогли запустить произвольные программы.

Вызов ***System()*** также пробует автоматически вставить в буфер вывода web-сервера после каждой строки вывода, если PHP запущен как модель сервера.

Если вы хотите запустить команду и использовать все данные непосредственно без всякого вмешательства других программ, то используйте функцию ***PassThru()***.

См. также ***exec(), popen()***.

passthru

Запускает внешнюю программу и выводит данные напрямую

Описание

```
string passthru(string command, int [ return_var ]);
```

Функция ***passthru()*** похожа на функцию ***Exec()*** для запуска command. Если параметр return_var установлен, то результат Unix-команды помещается здесь. Эта функция должна использоваться вместо ***Exec()*** или ***System()*** тогда, когда вывод из Unix-команды является двоичными данными, которые должны быть переданы непосредственно обратно в окно просмотра(browser). Это можно использовать, например, для запуска утилиты pbmplus для вывода непосредственно потока изображения. Установка типа image/gif и вызов программы pbmplus, чтобы вывести gif-рисунок, вы можете создавать PHP-скрипты, которые выводят изображения непосредственно.

См. также ***exec(), fpassthru()***.

Функции для работы с сессиями

Поддержка сессий в PHP включает в себя набор средств, позволяющих работать с набором последовательных обращений к документу (или нескольким документам). Эти средства позволяют организовать работу web-сайта более гибким образом, увеличивая его привлекательность и возможность обработки поступающих запросов.

Пользователь, обращающийся на web-страницу, получает уникальный идентификатор, называемый идентификатором сессии. Этот идентификатор может храниться либо в виде cookie на строке пользователя, он может передаваться серверу в составе URL.

Поддержка сессий в PHP позволяет осуществить регистрацию произвольного числа переменных, объявив их глобальными переменными для текущей сессии. Эти переменные будут сохранены, их значения можно будет использовать при повторном обращении к сайту. Когда посетитель странички входит на наш сайт, PHP автоматически проверяет (но только в том случае, если директива session.auto_start имеет значение 1) был ли вместе с запросом послан идентификатор сессии. Если идентификатор был послан, то PHP восстанавливает окружение, связанное с данным идентификатором. Проверка наличия переданного посетителем может быть осуществлена вручную при помощи функции ***session_register()*** или ***session_start()***.

Все зарегистрированные переменные сессии после завершения запроса будут сохранены. Если зарегистрированная глобальная переменная сессии не определена, то она будет сохранена с пометкой о том, что переменная не определена. Во всех последующих сеансах, имеющих тот же идентификатор сессии, эта переменная будет оставаться неопределенной до тех пор, пока она не станет определена.

Директивы track_vars и register_globals влияют на то, как глобальные переменные сессий будут храниться и восстанавливаться.

Если включена функция track_vars, но выключена функция register_globals, то в качестве глобальных переменных сессии могут быть зарегистрированы элементы ассоциативного массива $HTTP_SESSION_VARS. Восстановленные значения в последующих обращениях с тем же идентификатором сессии могут быть доступны только в виде элементов массива $HTTP_SESSION_VARS.

Пример регистрации глобальных переменных сессии с включенной директивой track_vars.

```
<?php
session_register("count");
$HTTP_SESSION_VARS["count"]++;
?>
```

Если директива register_globals будет включена, то все глобальные переменные скрипта могут быть зарегистрированы как глобальные

переменные сессии, и при повторных обращениях с тем же идентификатором сессии, как и тот, когда переменные были объявлены переменными сессии, эти переменные будут доступны под теми же именами, под которыми они существовали в предыдущем сеансе.

Пример регистрации переменных с включенной директивой register_globals:

```
<?php
session_register("count");
$count++;
?>
```

Если включены обе директивы (и track_vars, и register_globals), то глобальные переменные сессии и элементы массива $HTTP_SESSION_VARS будут ссылаться на одни и те же значения.

Существует два способа передачи идентификатора сессий:

- cookies;
- параметр URL.

Оптимальный вариант — использование cookies. Однако, не каждый клиент сконфигурирован таким образом, что в нем включена поддержка cookies, поэтому при работе с сессиями не cookies не следует полагаться полностью.

Следующий пример показывает, как можно зарегистрировать переменную и как организовать связь с другой страницей с использованием идентификатора сессии SID.

Пример подсчета посещений страницы конкретным пользователем:

```
 <?php
session_register ("count");
$count++;
?>
Добрый день! Вы посетили эту страницу <? echo $count; ?> раз.<p>
<php?
# Далее мы используем инструкцию <?=SID?> на тот случай,
# если пользователь не поддерживает cookies
?>
Чтобы продолжить, перейдите на следующую страницу <A
HREF="nextpage.php?<?=SID?>">по этой ссылке</A>
```

Обратите внимание (см. рис. С.18), как выглядит ссылка на страницу (в строке состояния в нижней чсти окна).

После перехода по ссылке весь идентификатор сессии станет видим в строке адреса (рис. С.19).

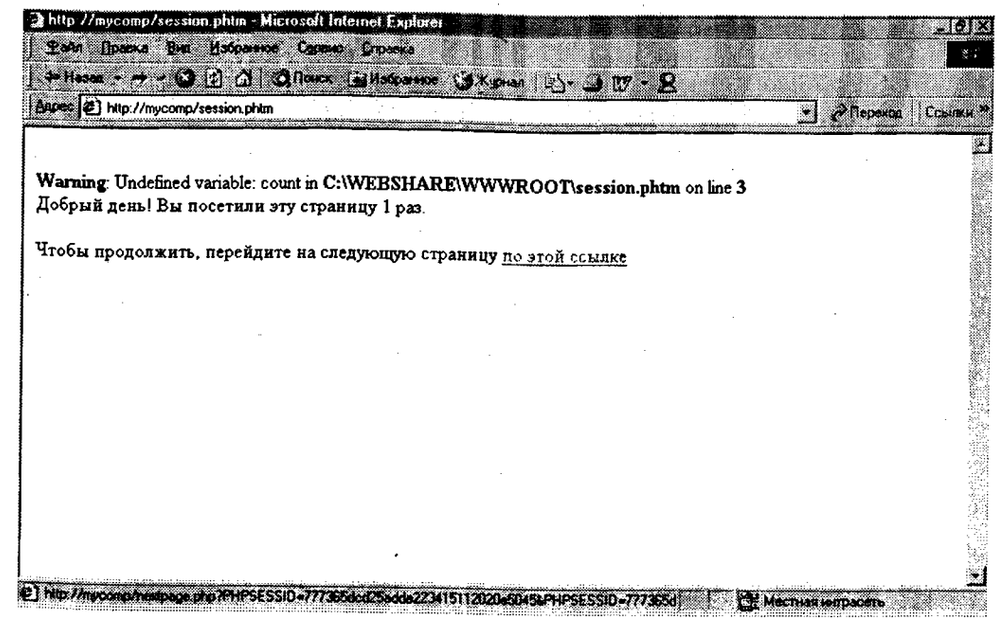


Рис. С.18. *Передаем идентификатор сессии в виде части адреса URL*

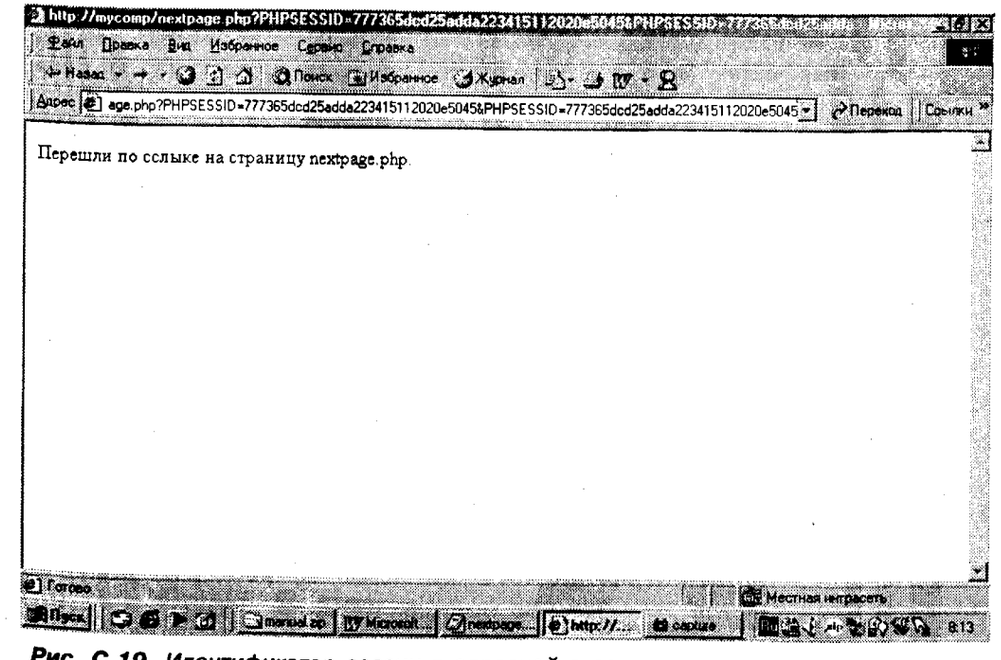


Рис. С.19. *Идентификатор сессии в адресной строке*

Система поддержки сессий предоставляет ряд возможностей для конфигурирования ряда опций. Вот некоторые из них.

- Параметр session.save_handler определяет имя средства обработки, используемого для сохранения и получения данных, связанных с сессией. По умолчанию files.
- Параметр session.save_path определяет аргументы, передаваемые средству сохранения параметров сессии, это может быть путь к файлам, хранящим данные сессий, по умолчанию /tmp.
- Параметр session.name задает имя сессии, используемой как имя cookie. По умолчанию PHPSESSID.
- Параметр session.auto_start указывает на то, будет ли модуль, поддерживающий сессии, включаться автоматически, по умолчанию 0 (не будет).
- Параметр session.cookie_lifetime задает время жизни cookies, посылаемых броузеру, в секундах. По умолчанию 0.
- Параметр session.serialize_handler задает имя средства обработки, осуществляющего сериализацию и извлечение сохраненных данных. Поддерживается php (внутренний формат) и WDDX. По умолчанию php.
- Параметр session.use_cookies задает состояние, определяющее то, будет ли модуль использовать cookie для хранения идентификатора сессии на стороне клиента. По умолчанию 1.

session_start ..... инициализирует работу с данными сессии;
session_destroy ..... разрушает все данные, связанные с сессией;
session_name ..... получает (или задает) имя текущей сессии;
session_module_name ..... получает (иди задает) модуль для текущей сессии;
session_save_path ..... получает (или задает) путь для сохранения данных текущей сессии;
session_id ..... получает (или задает) идентификатор текущей сесии;
session_register ..... регистрирует одну или несколько переменных как переменные сессии;
session_unregister ..... снимает регистрацию переменной сессии
session_unset ..... удаляет все переменные сессии;
session_is_registered ..... определяет, зарегистрирована ли переменная как переменая сессии;
session_get_cookie_params .... получает параметры cookie для сессии;
session_set_cookie_params .... задает параметр cookie для сессии;
session_decode ..... декодирует данные сессии из строки;
session_encode ..... кодирует данные сессии в строку.

Функции для работы со строками

Эти функции используются для манипулирования строками.

AddSlashes

Добавляет к строке символ обратной черты

Описание

```
string addslashes(string str);
```

Возвращает строку с обратной чертой (/) перед символами, которые должны быть выделены в запросах к базам данных и т.п. Эти символы: (’), двойные кавычки («), (\) и NUL (нулевой байт).

См. также ***stripslashes(), htmlspecialchars(), quotemeta().***

Chop

Удаляет повторяющиеся пробелы

Описание

```
string chop(string str);
```

Возвращает строку без повторяющихся пробелов.

Пример с функцией chop():

```
$trimmed = Chop($line);
```

См. также ***trim().***

Chr

Возвращает специальный символ

Описание

```
string chr(int ascii);
```

Возвращает односимвольную строку, содержащую символ, определенный кодом ascii.

Пример chr():

```
$str.= chr(27); /* добавляет символ ESC в конец $str */
$str = sprintf("The string ends in escape: %c", 27);
```

Эта функция дополняет функцию ***ord().***

См. также ***sprintf() с форматирующей строкой %c.***

chunk_split

Разбивает строку на мелкие части

Описание

```
string chunk_split(string string, int [chunklen], string [end] );
```

Может быть использована для разбития строки на части, например, при конвертировании результата функции ***base64_encode*** в формат RFC 2045. Функция вставляет в каждый фрагмент (по умолчанию до 76) символ конца строки (по умолчанию «\r\n»). Функция возвращает новую строку, оставляя исходную без изменений.

Пример chr_replace():

```
# format $data using RFC 2045 semantics
$new_string = chunk_split(base64_encode($data));
```

Эта функция выполняется значительно быстрее, чем ***ereg_replace()***.

convert_cyr_string

Переводит из одной русской кодовой таблицы в другую

Описание

```
string convert_cyr_string(string str, string from, string to);
```

Эта функция переводит указанную строку из одной русской кодовой таблицы в другую. Аргументы from и to определяют исходную и конечную кодовые таблицы1. Поддерживаемые типы кодировок:

k — koi8-r;
w — windows-1251;
i — iso8859-5;
a — x-cp866;
d — x-cp866;
m — x-mac-cyrillic.

crypt

Шифрует строку методом DES

Описание

```
string crypt(string str, string [salt]);
```

Функция ***crypt()*** шифрует строку, используя стандартный метод шифрации UNIX DES. Аргументы являются строками: строка, которую нужно зашифровать, и дополнительная 2-символная строка salt, на которой будет основываться шифрование. См. документацию UNIX

для дополнительной информации. Если аргумент salt отсутствует, то он будет генерирован случайным образом.

Некоторые операционные системы поддерживают больше одного типа шифрования. В действительности, иногда метод шифрования DES заменяется основанными на MD5 алгоритмами. Тип шифрования устанавливается аргументом salt. Во время установки PHP определяет возможности функций шифрации и будет поддерживать аргумент salt для других методов шифрации. Если параметр salt не установлен, то PHP автоматически сгенерирует стандартный 2-символный ключ DES, если же в системе по умолчанию установлен тип шифрации MD5, то будет сгенерирован MD5-совместимый ключ.

echo

Выводит текстовую строку

Описание

```
echo(string arg1, string [argn]...);
```

Выводит все параметры. ***Echo()*** в действительности не является функцией (это языковая конструкция), поэтому не обязательно использовать круглые скобки.

Пример echo:

```
echo "Hello World";
```

См. также ***print() printf() flush().***

explode

Разбивает строку на меньшие строки

Описание

```
array explode(string separator, string string);
```

Возвращает массив строк, содержащий элементы, разделенные строкой separator.

Пример explode():

```
$pizza = "slice1 slice2 slice3 slice4 slice5 slice6";
$pieces = explode(" ", $pizza);
```

См. также ***split(), implode().***

flush

Освобождает буферы вывода

Описание

```
void flush(void);
```

Освобождает буферы вывода PHP и все остальные, используемые PHP (CGI, web-сервер и т.д.). Это эффективная возможность вывести все накопленное в буферах в броузер пользователя.

get_meta_tags

Извлекает все содержимое атрибутов ярлыка meta из файла и возвращает в массиве

Описание

```
array get_meta_tags(string filename, int [use_include_path]);
```

Открывает файл filename и обрабатывает его строка за строкой и извлекает ярлыки <meta>.

Пример ярлыков Meta:

```
<meta name="author" content="name">
<meta name="tags" content="php3 documentation">
</head> <!- parsing stops here ->
```

Значение свойства name становится ключем, значение свойства content становится значением возвращаемого массива, поэтому можно легко использовать стандартные функции для его обработки или доступа к отдельным элементам. Специальные символы в значении свойства заменяются символом '_', остальные переводятся в нижний регистр.

Установка параметра use_include_path в 1 приведет к тому, что PHP3 будет пытаться открыть файл по стандартному include пути.

htmlspecialchars

Переводит специальные символы в коды HTML

Описание

```
string htmlspecialchars(string string);
```

Определенные символы имеют особое значение в HTML и должны быть заменены кодами HTML, если они таковые имеют. Эта функция возвращает строки с произведенными такими изменениями.

Эта функция полезна для очистки полученного от пользователя текста от разметки HTML (доски сообщений, гостевые книги и т.п.).

В настоящее время осуществляются следующие замены:

"&" (амперсанд) становится "&"

"«" (двойные кавычки) становится """

"<" (знак меньше) становится "<"

">" (знак больше) становится ">"

Следует отметить, что эта функция не заменяет ничего, кроме символов, указанных выше. Для полной обработки см. функцию ***htmlentities()***.

См. также ***htmlentities(), nl2br().***

htmlentities

Переводит все возможные символы в коды HTML

Описание

```
string htmlentities(string string);
```

Эта функция идентична ***htmlspecialchars()***, кроме того, что все символы, которые имеют соответствующий код HTML заменяются на этот HTML-код. Применяется кодовая таблица ISO-8859-1.

См. также ***htmlspecialchars(), nl2br().***

implode

Объединяет массив элементов в строку

Описание

```
string implode(array pieces, string glue);
```

Возвращает строку, содержащую совокупность всех элементов массива, в том же порядке, со строкой glue — между каждым элементом.

Пример implode():

```
$colon_separated = implode($array, ":");
```

См. также ***explode(), join(), split().***

join

Присоединяет элементы массива в строку

Описание

```
string join(array pieces, string glue);
```

Join() является псевдонимом функции ***implode()***, и полностью ей идентична.

ltrim

Удаляет пробелы из начала строки

Описание

```
string ltrim(string str);
```

Эта функция удаляет пробелы из начала строки и возвращает обрезанную строку.

*См. также **chop()**, **trim()**.*

md5

Вычисляет значение md5 для строки

Описание

```
string md5(string str);
```

Вычисляет значение MD5 для строки str, используя алгоритм RSA. Data Security, Inc. MD5 Message-Digest.

nl2br

Переводит символы новой строки в ярлык HTML разрыва строки

Описание

```
string nl2br(string string);
```

Возвращает string с '<BR>' вставляемыми перед каждой новой строкой.

*См. также **htmlspecialchars()**, **htmlentities()**.*

Ord

Возвращает ASCII-значение символа

Описание

```
int ord(string string);
```

Возвращает ASCII-значение первого символа строки string. Эта функция дополняет функцию ***chr()***.

Пример ord():

```
if (ord($str) == 10) {
    echo("The first character of \$str is a line feed.\n");
}
```

*См. также **chr()**.*

parse_str

Анализирует строку

Описание

```
void parse_str(string str);
```

Анализирует строку str, как если бы она была URL-строкой запроса, и устанавливает переменные среды.

Пример parse_str():

```
$str = "first=value&second[]=this+works&second[]=another";
parse_str($str);
echo $first; /* prints "value" */
echo $second[0]; /* prints "this works" */
echo $second[1]; /* prints "another" */
```

print

Выводит строку

Описание

```
print(string arg);
```

Выводит строку arg.

*См. также **echo()** **printf()**, **flush()**.*

printf

Выводит форматированную строку

Описание

```
int printf(string format, mixed [args]...);
```

Осуществляет вывод в соответствии с параметром format, который описан в описании функции ***sprintf()***.

*См. также **print()**, **sprintf()**, **flush()**.*

quoted_printable_decode

Переводит строку в кавычках в 8-битную строку

Описание

```
string quoted_printable_decode(string str);
```

Эта функция возвращает 8-битную строку, соответствующую декодированной строке в кавычках. Эта функция аналогична ***imap_qprint()***, за исключением того, что она не требует IMAP-модуль для работы.

QuoteMeta

Выделяет meta-символы

Описание

int quotemeta(string str);

Возвращает обработанную str с символами (\) перед каждым из следующих символов:

```
. \\ + w ? [ ^ ] ( $ )
```

См. также ***addslashes(), htmlentities(), htmlspecialchars(), nl2br(), stripslashes().***

rawurldecode

Декодирует URL-кодированную строку

Описание

```
string rawurldecode(string str);
```

Возвращает строку, в которой последовательность из символа процента (%) с последующих двух шестнадцатиричных цифр заменяется соответствующим буквенным символом. Например, строка

```
foo%20bar%40baz
```

будет заменена на

```
foo bar@baz
```

См. также ***rawurlencode().***

rawurlencode

URL-кодирует строку в соответствии с RFC1738

Описание

```
string rawurlencode(string str);
```

Возвращает строку, в которой все небуквенно-цифровые символы, кроме -_. заменяются на знак (%) с последующими двумя шестнадцатиричными цифрами. Это кодирование, описанное в RFC1738, применяется для защиты символов от интерпритации их как особых разделителей URL, и для защиты URL от искажения системами передачи данных с переводом символов (как некоторые e-mail системы). НаПример, если вы хотите включить пароль в ftp URL.

Пример rawurlencode():

```
echo '<A HREF="ftp://user:', rawurlencode ('foo @+%/'),
     '@ftp.my.com/x.txt">';
```

Или если требуется передать информацию в качестве части URL.

Пример rawurlencode():

```
echo '<A HREF="http://x.com/department_list_script/',
     rawurlencode ('sales и marketing/Miami'), '">';
```

См. также ***rawurldecode().***

setlocale

Устанавливает локальную информацию

Описание

```
string setlocale(string category, string locale);
```

Category является строкой, определяющей категорию функций, изменяемую строкой locale:

LC_ALL..... для всех нижеследующих строк;
LC_COLLATE ..... для сравнения строк;
LC_CTYPE..... для классификации и перевода символов, например: strtoupper();
LC_MONETARY ..... для localeconv();
LC_NUMERIC ..... для десятичного разделителя;
LC_TIME..... для даты и времени, форматируемых функцией strftime().

Если locale является пустой строкой «», то локальные имена будут установлены из значений переменных окружения с теми же именами, как у вышеописанных категорий, или из «LANG». Если locale равна нулю или «0», то местные установки не изменяются, возвращаются текущие установки.

Setlocale возвращает новое текущее locale, или false, если locale функционально не поддерживается текущей платформой, указанный locale не существует или категории неверны. Неверное имя категории также вызывает предупреждающее сообщение.

soundex

Вычисляет soundex-ключ для строки

Описание

```
string soundex(string str);
```

Вычисляет soundex ключ для str.

Ключ soundex имеет такое свойство, что слова, произносимые одинаково, имеют одинаковый soundex-ключ, и это может быть использовано в поиске в базах данных, когда вы знаете произношение и не знаете написание. Эта функция возвращает строку длиной 4 символа, начинающуюся буквой.

Эта функция описана Дональдом Кнутом в книге «The Art Of Computer Programming, vol. 3: Sorting и Searching», Addison-Wesley (1973), pp. 391-392.

Пример Soundex:

```
soundex("Euler") == soundex("Ellery") == 'E460';
soundex("Gauss") == soundex("Ghosh") == 'G200';
soundex("Knuth") == soundex("Kant") == 'H416';
soundex("Lloyd") == soundex("Ladd") == 'L300';
soundex("Lukasiewicz") == soundex("Lissajous") == 'L222';
```

sprintf

Возвращает форматированную строку

Описание

```
sprintf(string format, mixed [args]...);
```

Возвращает строку, обрабатываемую в соответствии с форматирующей строкой format.

Форматирующая строка, содержащая ноль или более директив: обычные символы (кроме %), которые копируются прямо в результат, и описания изменений, каждое из которых выполняет определенные действия. Это применительно и к ***sprintf()*** и к ***printf()***. Каждое описание изменений состоит из следующих элементов.

Дополнительный описатель заполнения, который говорит, какие символы будут использоваться для заполнения результата до правильного размера строки. Это могут быть пробелы или 0 (символ нуля). По умолчанию заполняется пробелами. Альтернативный символ заполнения может быть определен одинарной кавычкой ('). См. примеры ниже.

Дополнительный описатель выравнивания, который говорит, что результат должен быть выравнен по левому или по правому краю. По умолчанию выравнивание происходит по правому краю, символ — приведет к вырвниванию по левому краю.

Дополнительный описатель ширины, который говорит, с каким количеством символов (минимум) может производиться данная замена.

Дополнительный описатель точности, который говорит, сколько десятичных знаков следует отображать для чисел с плавающей точкой. Этот описатель не действует на остальные типы, кроме double. (Другая полезная функция для форматирования чисел это ***number_format()***.)

Описатель типа, который говорит о том, как тип данных аргумента должен трактоваться. Возможные типы: % — символ процента. Аргумент не требуется.

b..... аргумент трактуется как integer и представляется как двоичное число;

c..... аргумент трактуется как integer и представляется как символ с ASCII-значением;

d..... аргумент трактуется как integer и представляется как десятичное число;

f..... аргумент трактуется как double и представляется как число с плавающей точкой;

o..... аргумент трактуется как integer и представляется как восьмиричное число;

s..... аргумент трактуется и представляется как строка;

x..... аргумент трактуется как integer и представляется как шестнадцатиричное число (с буквами в нижнем регистре);

X..... аргумент трактуется как integer и представляется как шестнадцатиричное число (с буквами в верхнем регистре).

*См. также **printf()**, **number_format()**.*

Пример sprintf() числа с нулями:

```
$isodate = sprintf("%04d-%02d-%02d", $year, $month, $day);
```

Пример sprintf: форматирование денежной единицы:

```
$money1 = 68.75;
$money2 = 54.35;
$money = $money1 + $money2;
// echo $money will output "123.1";
$formatted = sprintf ("%01.2f", $money);
// echo $formatted will output "123.10"
```

strchr

Находит первое появление символа

Описание

```
string strchr(string haystack, string needle);
```

Эта функция является псевдонимом для функции ***strstr()***, и полностью ей идентична.

strcmp

Двоичное сравнение строк

Описание

```
int strcmp(string str1, string str2);
```

Возвращает < 0 если str1 меньше чем str2, > 0 если str1 больше чем str2, и 0 если они равны.

Следует отметить, что это сравнение чувствительно к регистру.

*См. также **ereg(), substr(), strstr()**.*

strcspn

Находит длину начального сегмента, не совпадающего с маской

Описание

```
int strcspn(string str1, string str2);
```

Возвращает длину начального сегмента str1, который не содержит любые символы в str2.

*См. также **strspn()**.*

StripSlashes

Удаляет символы \ из строки

Описание

```
string stripslashes(string str);
```

Возвращает строку с вырезанными символами \. (\' заменяется на ' и так далее). Двойные \\ заменяются на \.

*См. также **addslashes()**.*

strlen

Возвращает длину строки

Описание

```
int strlen(string str);
```

Возвращает длину строки string.

strrpos

Находит позицию последнего появления символа в строке

Описание

```
int strrpos(string haystack, char needle);
```

Возвращает номер позиции последнего появления символа needle в строке haystack. Следует отметить, что needle в этом случае может быть

только единственным символом. Если в качестве параметра needle указывается строка, то только первый символ будет использован.

Если needle не найден, то возвращается false.

Если параметр needle не является строкой, то он переводится в десятичное число и рассматривается как числовое значение символа.

*См. также **strpos(), strrchr(), substr(), strstr()**.*

strpos

Находит позицию первого появления строки

Описание

```
int strpos(string haystack, string needle, int [offset]);
```

Возвращает номер позиции первого появления строки needle в строке haystack. В отличии от ***strrpos()***, эта функция может рассматривать целую строку в качестве параметра needle и целая строка будет использоваться.

Если параметр needle не найден, то возвращается false.

Если параметр needle не является строкой, то он переводится в целое число и рассматривается как числовое значение символа.

Дополнительный параметр offset позволяет вам определять, с какого символа в строке haystack начинать поиск. Позиция возвращается все равно относительно начала строки haystack.

*См. также **strrpos(), strrchr(), substr(), strstr()**.*

strrchr

Находит последнее появление символа в строке

Описание

```
string strrchr(string haystack, string needle);
```

Эта функция возвращает позицию haystack, с которой начинается последнее появление needle и продолжается до конца haystack.

Возвращает false, если needle не найдена.

Если параметр needle содержит более чем один символ, то используется первый символ.

Если параметр needle не является строкой, то он переводится в целое число и рассматривается как числовое значение символа.

Пример strrchr():

```
// получение последней директории в $PATH
$dir = substr( strrchr( $PATH, ":" ), 1 );
// получение всего после последней новой строки
```

```
$text = "Line 1\nLine 2\nLine 3";
$last = substr( strrchr( $text, 10 ), 1 );
```

См. также ***substr(), strstr().***

strrev

Переворачивает строку

Описание

```
string strrev(string string);
```

Возвращает перевернутую строку string.

strspn

Находит длину начального сегмента, отвечающего маске

Описание

```
int strspn(string str1, string str2);
```

Возвращает длину начального сегмента строки str1, который содержит все символы из str2.

См. также ***strcspn().***

strstr

Находит первое появление строки

Описание

```
string strstr(string haystack, string needle);
```

Возвращает все haystack с первого появления строки needle и до конца.

Если параметр needle не найден, то возвращается false.

Если параметр needle не является строкой, то он переводится в целое число и рассматривается как числовое значение символа.

См. также ***strrchr(), substr(), ereg().***

strtok

Разбивает строку

Описание

```
string strtok(string arg1, string arg2);
```


Strtok() используется для разбития строки. Это значит, что если вы имеете строку типа «This is an example string», то вы можете разбить эту строку на отдельные слова, используя пробел в качестве разделителя.

Пример strtok():

```
$string = "This is an example string";
$tok = strtok($string," ");
while($tok) {
     echo "Word=$tok<br>";
     $tok = strtok(" ");
}
```

Следует отметить, что только первый вызов функции strtok использует строковый аргумент. Для каждого последующего вызова функции strtok необходим только разделитель, так как это позволяет контролировать положение в текущей строке. Для начала заново или для разбития новой строки вам необходимо просто вызвать strtok с параметром строки опять для ее инициализации. Вы можете вставлять несколько разделителей в параметр разделителя. Строка будет разделяться при обнаружении любого из указанных символов.

Также будьте внимательны к разделителям равным «0». Это может вызвать ошибку в определенных выражениях.

См. также ***split(), explode().***

strtolower

Переводит строку в нижний регистр

Описание

```
string strtolower(string str);
```

Возвращает сроку string со всеми буквенными символами, переведенными в нижний регистр. Буквенные символы определяются текущими локальными установками.

См. также ***strtoupper(), ucfirst().***

strtoupper

Переводит строку в верхний регистр

Описание

```
string strtoupper(string string);
```

Возвращает строку string со всеми буквенными символами, переведенными в верхний регистр. Следует отметить, что буквенные символы определяются текущими локальными установками.

См. также ***strtolower(), ucfirst().***

str_replace

Заменяет все вхождения строки на указанную строку

Описание

```
string str_replace(string needle, string str, string haystack);
```

Эта функция заменяет все вхождения строки needle в строке haystack на указанную строку str. Если вам не требуются причудливые правила замены, то вам следует всегда использовать эту функцию вместо ***ereg_replace()***.

Пример str_replace():

```
$bodytag = str_replace("%body%", "black", "<body text=%body%>");
```

См. также ***ereg_replace()***.

strtr

Переводит определенные символы

Описание

```
string strtr(string str, string from, string to);
```

Эта функция обрабатывает строку str, заменяя все появления каждого символа из строки from на соответствующие символы в строке to, и возвращает результат.

Если строки from и to имеют различную длину, то дополнительные символы более длинной из строк игнорируются.

Пример strtr():

```
$addr = strtr($addr, "дец", "xyz");
```

См. также ***ereg_replace()***.

substr

Возвращает часть строки

Описание

```
string substr(string string, int start, int [length]);
```

Эта функция возвращает часть строки string, определяемую параметрами start (начало) и length (длина).

Если параметр start положительный, то возвращаемая строка будет начинаться с start символа строки string.

Примеры:

```
$rest = substr("abcdef", 1); // возвращает "bcdef"
$rest = substr("abcdef", 1, 3); // возвращает "bcd"
```


Если параметр start отрицательный, то возвращаемая строка будет начинаться с start символа от конца строкиstring.

Примеры:

```
$rest = substr("abcdef", -1); // возвращает "f"
$rest = substr("abcdef", -2); // возвращает "ef"
$rest = substr("abcdef", -3, 1); // возвращает "d"
```

Если параметр length указан и он положительный, то возвращаемая строка закончится за length символов от начала start. Это приведет к строке с отрицательной длиной (потому что начало будет за концом строки), поэтому возвращаемая строка будет содержать один символ от начала строки start.

Если параметр length указан и имеет отрицательное значение, то возвращаемая строка закончится за length от конца строки string. Это приведет к строке с отрицательной длиной, поэтому возвращаемая строка будет содержать один символ от начала строки start.

Примеры:

```
$rest = substr("abcdef", -1, -1); // вернет "bcde"
```

См. также ***strrchr(), ereg()***.

trim

Обрезает пробелы с начала и с конца строки

Описание

```
string trim(string str);
```

Эта функция обрезает пробеды с начала и с конца строки и возвращает обрезанную строку.

См. также ***chop(), ltrim()***.

ucfirst

Переводит первый символ строки в верхний регистр

Описание

```
string ucfirst(string str);
```

Делает заглавным первый символ строки str, если этот символ буквенный.

Следует напомнить, что буквенные символы определяются текущими настройками.

См. также ***strtoupper(), strtolower()***.

Ucwords

Переводит в верхний регистр первые символы каждого слова в строке

Описание

```
string ucwords(string str);
```

Делает заглавным первый символ каждого слова в строке str, если этот символ буквенный.

*См. также **strtoupper(), strtolower(), ucfirst().***

Новые функции

AddCSlashes ..... строки со знаками цитат в стиле C;
bin2hex ..... преобразует двоичные данные в шестнадцатиричное предлставление;
count_chars ..... возвращает информацию о символах, используемых в строке;
crc32 ..... вычисляет crc32 для строки;
get_html_translation_table .... возвращает таблицу преобразования с использованием ***htmlspecialchars()*** и ***htmlentities()***;
hebrev ..... преобразует логический текст Hebrew в видимый текст;
hebrevc ..... преобразует логический Hebrew в видлимый текст с использованием новой строки;
levenshtein ..... вычисляет расстояние в смысле Levenshtein для двух строк;
Metaphone ..... вычисляет индекс metaphone для строки;
ob_start ..... включает выходной буфер;
ob_get_contents ..... возвращает содержимое выходного буфера;
ob_end_flush ..... посылает выходной буфер и выключает выходной буфер;
ob_end_clean ..... очищает выходной буфер и выключает его;
ob_implicit_flush ..... переключает буферизацию;
rtrim ..... удаляет разделительные пробелы;
sscanf ..... анализирует входную строку в соответствии с заданным форматом;
similar_text ..... проверяет подобие двух строк;
strcasecmp ..... безопасное сравнение строк, чувствительное к регистру;
strip_tags ..... удаление ярлыков HTML и PHP из строки;
stripcslashes ..... удаляет знаки цитат, установленные с помощью ***addcslashes()***;
stristr ..... чувствительный к регистру ***strstr()***;
strlen ..... получить длину строки;

strnatcmp ..... сравнение строк с использованием алгоритма естественного порядка;
strnatcasecmp ..... чувствительное к регистру сравнение строк с использованием алгоритма естественного порядка;
strncmp ..... безопасное сравнение строк с использованием первых n символов;
strpos ..... найти положение первого вхождения символа в строке;
strrchr ..... найти положение последнего вхождения символа в строке;
str_repeat ..... воспроизвести строку;
substr_count ..... сосчитать количество вхождений подстроки в строке;
substr_replace ..... удалить текст внутри части строки.

Для полного описания функций см. документацию на компакт-диске.

Функции для работы с переменными

gettype

Получает тип переменной

Описание

```
string gettype (mixed var);
```

Возвращает тип переменной var.

Возможные значения для возвращаемой строки:

- "integer"
- "double"
- "string"
- "array"
- "object"
- "unknown type"

*См. также **settype().***

intval

Возвращает целочисленное значение переменной

Описание

```
int intval (mixed var, int [base]);
```

Возвращает целочисленное значение переменной var, используя указанное основание для перевода (по умолчанию 10). var может быть скалярного типа. Нельзя использовать функцию ***intval()*** для массивов или объектов.

См. также ***doubleval(), strval(), settype(), приведение типов.***

doubleval

Получает значение переменной в формате double

Описание

```
double doubleval (mixed var);
```

Возвращает double (с плавающей точкой) значение переменной var. Параметр var может быть скалярного типа. Вы не можете использовать ***doubleval()*** на массивах и объектах.

См. также ***intval(), strval(), settype(), приведение типов.***

empty

Определяет, присвоено ли переменной какое-либо значение

Описание

```
int empty (mixed var);
```

Возвращает false, если var существует и имеет не пустое или не нулевое значение, true — в обратном случае.

См. также ***isset(), unset().***

is_array

Определяет, входит ли переменная в массив

Описание

```
int is_array (mixed var);
```

Возвращает true, если var есть в массиве, false — в обратном случае.

См. также ***is_double(), is_float(), is_int(), is_integer(), is_real(), is_string(), is_long(), is_object().***

is_double

Определяет, является ли переменная типа double

Описание

```
int is_double (mixed var);
```

Возвращает true, если var является типа double, false — в обратном случае.

См. также ***is_array(), is_float(), is_int(), is_integer(), is_real(), is_string(), is_long(), is_object().***

is_float

Определяет, является ли переменная типа float

Описание

```
int is_float (mixed var);
```

Эта функция является псевдонимом для функции ***is_double()***.

См. также ***is_double(), is_real(), is_int(), is_integer(), is_string(), is_object(), is_array(), is_long().***

is_int

Определяет, является ли переменная типа integer

Описание

```
int is_int (mixed var);
```

Эта функция является псевдонимом для функции ***is_long()***.

См. также ***is_double(), is_float(), is_integer(), is_string(), is_real(), is_object(), is_array(), is_long().***

is_integer

Определяет, является ли переменная типа integer

Описание

```
int is_integer (mixed var);
```

Эта функция является псевдонимом для функции ***is_long()***.

См. также ***is_double(), is_float(), is_int(), is_string(), is_real(), is_object(), is_array(), is_long().***

is_long

Определяет, является ли переменная переменной типа integer

Описание

```
int is_long (mixed var);
```

Возвращает true, если var является типа integer (long), false — в обратном случае.

См. также ***is_double(), is_float(), is_int(), is_real(), is_string(), is_object(), is_array(), is_integer().***

is_object

Определяет, является ли переменная типа object

Описание

```
int is_object (mixed var);
```

Возвращает true, если var типа object, false — в обратном случае.

См. также ***is_long(), is_int(), is_integer(), is_float(), is_double(), is_real(), is_string(), is_array().***

is_real

Определяет, является ли переменная переменной типа real

Описание

```
int is_real (mixed var);
```

Эта функция является псевдонимом для функции ***is_double()***.

См. также ***is_long(), is_int(), is_integer(), is_float(), is_double(), is_object(), is_string(), is_array().***

is_string

Определяет, является ли переменная переменной типа string

Описание

```
int is_string (mixed var);
```

Возвращает true, если var is a string, false — в обратном случае.

См. также ***is_long(), is_int(), is_integer(), is_float(), is_double(), is_real(), is_object(), is_array().***

isset

Определяет, существует ли указанная переменная

Описание

```
int isset (mixed var);
```

Возвращает true, если var существует, false — в обратном случае.

Если переменная была удалена функцией ***unset()***, она больше не будет определяться функцией ***isset()***.

```
$a = "test";
echo isset($a); // true
unset($a);
echo isset($a); // false
```

См. также ***empty(), unset().***

settype

Устанавливает тип переменной

Описание

```
int settype (string var, string type);
```

Устанавливает тип переменной var как type.

Возможные значения type:

- «integer»
- «double»
- «string»
- «array»
- «object»

Возвращает true, при успехе, false — в обратном случае.

См. также ***gettype().***

strval

Получает строковое значение переменной

Описание

```
string strval (mixed var);
```

Получает строковое значение var.

Var может быть любого скалярного типа. Вы не можете использовать ***strval()*** на массивах или объектах.

См. также ***doubleval(), intval(), settype()*** *и подмена типов.*

unset

Удаляет указанную переменную

Описание

```
int unset (mixed var);
```

Unset() уничтожает указанную переменную и возвращает true.

Пример с unset():

```
unset( $foo );
unset( $bar['quux'] );
```

См. также ***isset(), empty().***

call_user_func

Вызвать пользовательскую функцию

is_bool

Проверить, является ли переменная логической переменной

is_numeric

Является ли переменная числовой или числовой строкой

is_resource

Является ли переменная ресурсом

print_r

Вывести понятную для человека информацию о переменной (см. рис. С.20)

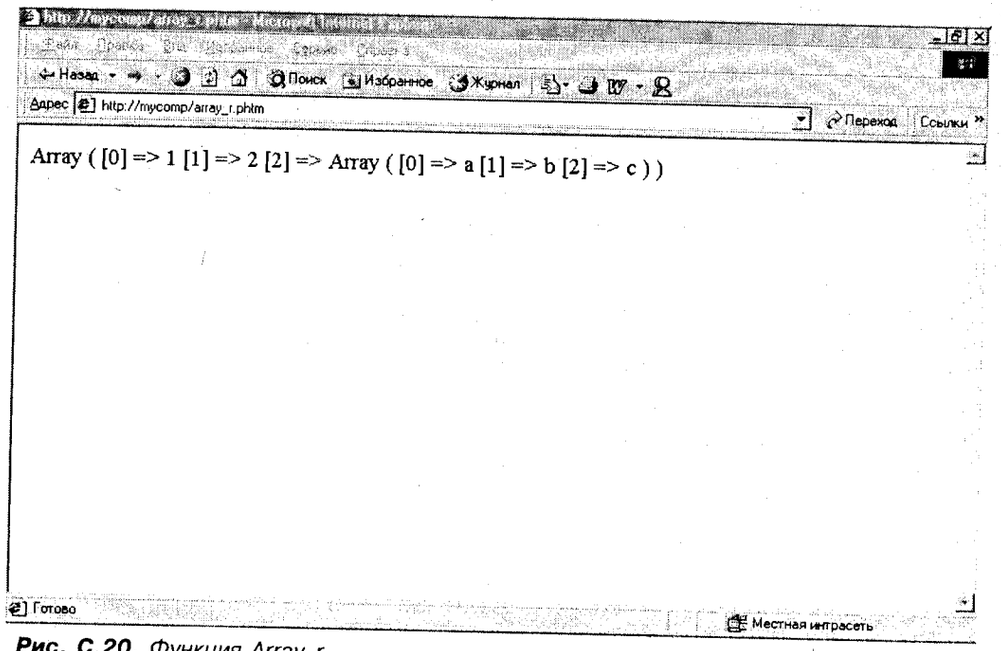


Рис. С.20. *Функция Array_r*

var_dump

Вывести информацию о переменной

Пример:

```
<pre>
<?php
     $a = array (1, 2, array ("a", "b", "c"));
     var_dump ($a);
?>
</pre>
```

Результат работы см. на рис. С.21.

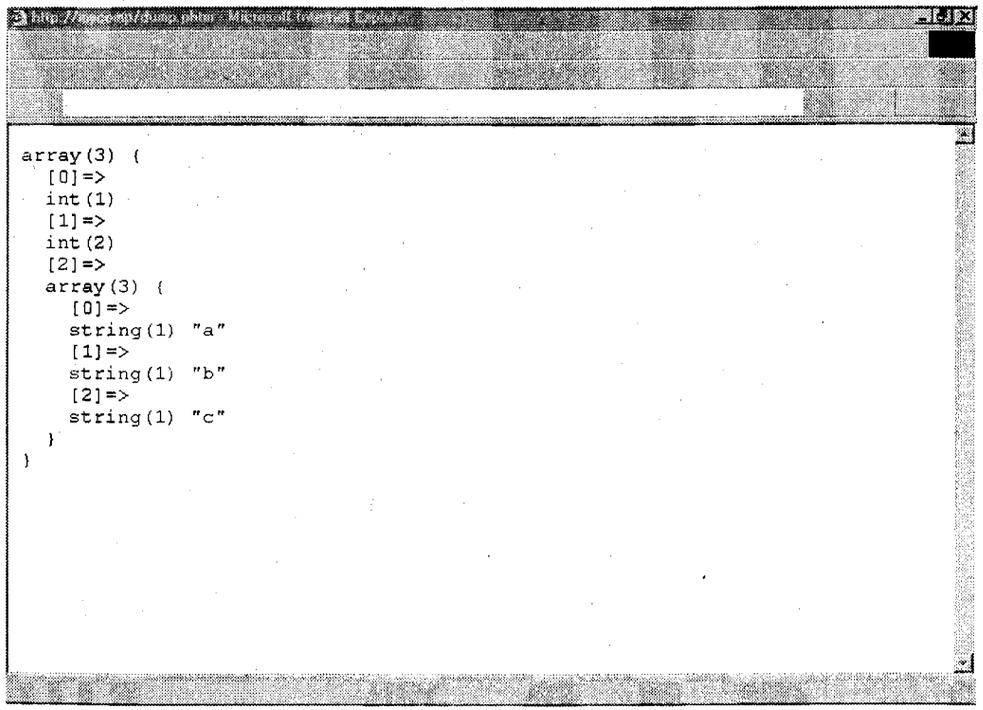


Рис. С.21. *Функция dump*

Функции XML-анализатора

XML (eXtensible Markup Language, расширяемый язык разметки) — это формат, используемый для создания документов в сети Интернет. Информацию о XML можно получить по адресу http://www.w3.org/XML/.

Расширение языка PHP для работы с XML поддерживает три базовых кодировки US-ASCII, ISO-8859-1 и UTF-8, кодировка UTF-16 не поддерживается.

Расширение, позволяюшее работать с XML, предоставляет возможности для создания анализаторов XML, в которых могут быть определены средства обработки событий XML. Для каждого анализатора можно задать несколько настраиваемых параметров.

Определены такие средства обработки событий XML (табл. С.6).

Таблица С.6
Средства обработки событий XML

Функция PHP для задания средства обработки событий	Определение
xml_set_element_handler()	*Событие элемента. Событие наступает всякий раз, когда XML-анализатор встречает открывающий или закрывающий ярлык. Для открывающего и закрывающего ярлыков могут быть определены отдельные независимые функции обработки этих событий*
xml_set_character_data_handler()	*Символьными данными (character data) грубо можно считать все содержимое XML-документа, которое не относится к специальным элементам XML-разметки, включая пробелы между ярлыкми. Эта функция обработки будет применяться ко всем таким текстовым элементам (символьным данным)*
xml_set_processing_instruction_handler()	*Инструкции обработки (processing instructions) определяются с использованием специальных ярлыков. Так, ярлык <?php ?> представляет собой ярлык инструкции обработки, в нем PHP носит название «цели» инструкции обработки ("PI target"), т.е. того способа, который должен быть применен для обработки содержимого элемента. Обработка таких событий зависит от того, что представляет собой программа*
xml_set_default_handler()	*Это средство обработки, используемое по умолчанию. Все что не делают другие средства обработки, направляется средству обработки, используемому по умолчанию.*
xml_set_unparsed_entity_decl_handler()	*Это средство обработки вызывается для задания обработки непроанализированных фрагментов (NDATA)*
xml_set_notation_decl_handler()	*Это средство обработки вызывается для объявления обозначений*
xml_set_external_entity_ref_handler()	*Это средство обработки вызывается тогда, когда анализатор встречается со ссылкой на файл или URL внешнего элемента*

Коды ошибок

Для кодов ошибок XML определены следующие значения (используются в ***xml_parse()***):

- XML_ERROR_NONE
- XML_ERROR_NO_MEMORY
- XML_ERROR_SYNTAX
- XML_ERROR_NO_ELEMENTS

- XML_ERROR_INVALID_TOKEN
- XML_ERROR_UNCLOSED_TOKEN
- XML_ERROR_PARTIAL_CHAR
- XML_ERROR_TAG_MISMATCH
- XML_ERROR_DUPLICATE_ATTRIBUTE
- XML_ERROR_JUNK_AFTER_DOC_ELEMENT
- XML_ERROR_PARAM_ENTITY_REF
- XML_ERROR_UNDEFINED_ENTITY
- XML_ERROR_RECURSIVE_ENTITY_REF
- XML_ERROR_ASYNC_ENTITY
- XML_ERROR_BAD_CHAR_REF
- XML_ERROR_BINARY_ENTITY_REF
- XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF
- XML_ERROR_MISPLACED_XML_PI
- XML_ERROR_UNKNOWN_ENCODING
- XML_ERROR_INCORRECT_ENCODING
- XML_ERROR_UNCLOSED_CDATA_SECTION
- XML_ERROR_EXTERNAL_ENTITY_HANDLING

Кодировка символов

В PHP, в расширении XML поддерживается кодировка Unicode для различных наборов символов. Существует два типа кодов для символов: исходная кодировка и конечная кодировка. В PHP для внутреннего представления всегда используется кодировка UTF-8.

Исходная кодировка определяется в момент анализа XML-документа. При создании XML-анализатора можно указать вид исходной кодировки, т.е. той кодировки, которая используется в анализируемых XML-документах. Эта кодировка не может быть изменена в дальнейшем в процессе работы XML-анализатора. Поддерживаемые кодировки таковы: ISO-8859-1, US-ASCII, UTF-8. По умолчанию используется ISO-8859-1.

Конечная кодировка используется в момент обработки данных функциями обработки XML-анализатора.

Примеры

Здесь размещено несколько примеров скриптов, анализирующий XML-документы.

Структура XML-документа

Этот пример выводит структуру документа.

Пример вывода структуры XML-элементов документа:

```
$file = "data.xml";
$depth = array();

function startElement($parser, $name, $attrs) {
     global $depth;
     for ($i = 0; $i < $depth[$parser]; $i++) {
          print "  ";
     }
     print "$name\n";
     $depth[$parser]++;
}

function endElement($parser, $name) {
     global $depth;
     $depth[$parser]--;
}

$xml_parser = xml_parser_create();
xml_set_element_handler($xml_parser, "startElement", "endElement");
if (!($fp = fopen($file, "r"))) {
     die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
     if (!xml_parse($xml_parser, $data, feof($fp))) {
          die(sprintf("XML error: %s at line %d",
                  xml_error_string(xml_get_error_code($xml_parser)),
                        xml_get_current_line_number($xml_parser)));
     }
}
xml_parser_free($xml_parser);
```

Преобразование XML-ярлыков

Пример преобразования XML в HTML. В этом примере происходит преобразование ярлыков XML в ярлыки HTML. Все ярлыки, не найденные в массиве ярлыков, будут проигнорированы. Пример будет работать только с определенным типом документов XML.

```
$file = "data.xml";
$map_array = array(
     "BOLD"      => "B",
     "EMPHASIS" => "I",
     "LITERAL"   => "TT"
);

function startElement($parser, $name, $attrs) {
     global $map_array;
     if ($htmltag = $map_array[$name]) {
          print "<$htmltag>";
     }
}

function endElement($parser, $name) {
     global $map_array;
     if ($htmltag = $map_array[$name]) {
          print "</$htmltag>";
     }
}

function characterData($parser, $data) {
     print $data;
}

$xml_parser = xml_parser_create();
// use case-folding so we are sure to find the tag in $map_array
xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, true);
xml_set_element_handler($xml_parser, "startElement", "endElement");
xml_set_character_data_handler($xml_parser, "characterData");
if (!($fp = fopen($file, "r"))) {
     die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
     if (!xml_parse($xml_parser, $data, feof($fp))) {
          die(sprintf("XML error: %s at line %d",
                 xml_error_string(xml_get_error_code($xml_parser)),
                       xml_get_current_line_number($xml_parser)));
     }
}
xml_parser_free($xml_parser);
```


Использование внешних фрагментов в XML

В этом примере показан вариант использования средства обработки внешних элементов в XML, а также способы обработки PI.

Пример обработки внешних объектов:

```
$file = "xmltest.xml";

function trustedFile($file) {
     // only trust local files owned by ourselves
     if (!eregi("^([a-z]+)://", $file)
          && fileowner($file) == getmyuid()) {
               return true;
     }
     return false;
}

function startElement($parser, $name, $attribs) {
     print "&lt;<font color=\"#0000cc\">$name</font>";
     if (sizeof($attribs)) {
          while (list($k, $v) = each($attribs)) {
               print " <font color=\"#009900\">$k</font>=\"<font
                         color=\"#990000\">$v</font>\"";
          }
     }
     print "&gt;";
}

function endElement($parser, $name) {
     print "&lt;/<font color=\"#0000cc\">$name</font>&gt;";
}

function characterData($parser, $data) {
     print "<b>$data</b>";
}

function PIHandler($parser, $target, $data) {
     switch (strtolower($target)) {
          case "php":
               global $parser_file;
               // If the parsed document is "trusted", we say it is safe
               // to execute PHP code inside it.  If not, display the code
```

```
            // instead.
            if (trustedFile($parser_file[$parser])) {
                eval($data);
            } else {
                printf("Untrusted PHP code: <i>%s</i>",
                        htmlspecialchars($data));
            }
            break;
    }
}

function defaultHandler($parser, $data) {
    if (substr($data, 0, 1) == "&" && substr($data, -1, 1) == ";") {
        printf('<font color="#aa00aa">%s</font>',
                htmlspecialchars($data));
    } else {
        printf('<font size="-1">%s</font>',
                htmlspecialchars($data));
    }
}

function externalEntityRefHandler($parser, $openEntityNames, $base,
$systemId, $publicId) {
    if ($systemId) {
        if (!list($parser, $fp) = new_xml_parser($systemId)) {
            printf("Could not open entity %s at %s\n",
                   $openEntityNames, $systemId);
            return false;
        }
        while ($data = fread($fp, 4096)) {
          if (!xml_parse($parser, $data, feof($fp))) {
            printf("XML error: %s at line %d while parsing entity %s\n",
                   xml_error_string(xml_get_error_code($parser)),
                   xml_get_current_line_number($parser), $openEntityNames);
                   xml_parser_free($parser);
            return false;
            }
        }
        xml_parser_free($parser);
        return true;
    }
```



```
    return false;
}

function new_xml_parser($file) {
    global $parser_file;

    $xml_parser = xml_parser_create();
    xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, 1);
    xml_set_element_handler($xml_parser, "startElement", "endElement");
    xml_set_character_data_handler($xml_parser, "characterData");
    xml_set_processing_instruction_handler($xml_parser, "PIHandler");
    xml_set_default_handler($xml_parser, "defaultHandler");
    xml_set_external_entity_ref_handler($xml_parser, "externalEntityRefHandler");

    if (!($fp = @fopen($file, "r"))) {
        return false;
    }
    if (!is_array($parser_file)) {
        settype($parser_file, "array");
    }
    $parser_file[$xml_parser] = $file;
    return array($xml_parser, $fp);
}

if (!(list($xml_parser, $fp) = new_xml_parser($file))) {
    die("could not open XML input");
}

print "<pre>";
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d\n",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
print "</pre>";
print "parse complete\n";
xml_parser_free($xml_parser);

>
```

Файл xmltest.xml:

```
<?xml version='1.0'?>
<!DOCTYPE chapter SYSTEM "/just/a/test.dtd" [
<!ENTITY plainEntity "FOO entity">
<!ENTITY systemEntity SYSTEM "xmltest2.xml">
]>
<chapter>
 <TITLE>Title &plainEntity;</TITLE>
 <para>
  <informaltable>
   <tgroup cols="3">
    <tbody>
      <row><entry>a1</entry><entry morerows="1">b1</entry><entry>c1</
entry></row>
     <row><entry>a2</entry><entry>c2</entry></row>
     <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
    </tbody>
   </tgroup>
  </informaltable>
 </para>
 &systemEntity;
 <sect1 id="about">
  <title>About this Document</title>
  <para>
   <!- this is a comment ->
   <?php print 'Hi!  This is PHP version '.phpversion(); ?>
  </para>
 </sect1>
</chapter>
```

Файл xmltest2.xml:

```
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ENTITY testEnt "test entity">
]>
<foo>
   <element attrib="value"/>
   &testEnt;
   <?php print "This is some more PHP code being executed."; ?>
</foo>
```


xml_parser_create

Создать XML-анализатор

Описание

```
int xml_parser_create ([string encoding])
encoding (optional)
```

Возвможные значения:

- ISO-8859-1 (default)
- US-ASCII
- UTF-8

Эта функция создает XML-анализатор и возвращает на него указатель, который может быть использован в других XML-функциях. В случае ошибки возвращает false.

Функции XML-анализатора

xml_set_object ..... использовать анализатор XML в объекте;
xml_set_element_handler ..... задать средства обработки для начала и конца элемента;
xml_set_character_data_handler ..... задать средство обработки символьных данных;
xml_set_processing_instruction_handler ..... задать средства обработки процессорных инструкций PI;
xml_set_default_handler ..... задать средство обработки, используемое по умолчанию;
xml_set_unparsed_entity_decl_handler ..... задать средство обработки неанализируемых элементов;
xml_set_notation_decl_handler ..... задать средство обработки задания обозначений;
xml_set_external_entity_ref_handler ..... задать средство обработки внешних элементов;
xml_parse ..... начать анализ XML-документа;
xml_get_error_code ..... получить код ошибки XML-анализатора;
xml_error_string ..... получить строку ошибки XML-анализатора;
xml_get_current_line_number ..... получить номер текущей строки, обрабатываемой анализатором XML;
xml_get_current_column_number ..... получить номер текущего положения в строке;
xml_get_current_byte_index ..... получить номер текущего байта;
xml_parse_into_struct ..... анализировать данные XML в виде массива;
xml_parser_free ..... освободить анализатор XML;

xml_parser_set_option ..... установить параметры анализатора XML;
xml_parser_get_option ..... получить параметры XML-анализатора;
utf8_decode ..... преобразовать строку символов ISO-8859-1, кодированных с помощью UTF-8, в ISO-8859-1;
utf8_encode ..... кодировать строку символов ISO-8859-1 в UTF-8.

Прочие функции

create_function ..... создает анонимную (lambda-style) функцию;
connection_aborted ..... возвращает true, если клиент отсоединен;
connection_status ..... возвращает статус соединения;
connection_timeout ..... возвращает true, если время выполнения скрипта истекло;
define ..... определяет именованную константу;
defined ..... проверяет наличие указанной именованной константы;
die ..... выводит сообщение и прекращает выполнение скрипта;
eval ..... проверяет строку PHP-кода;
exit ..... прекращает выполнение текущего скрипта;
func_get_arg ..... возвращает элемент из списка аргументов;
func_get_args ..... возвращает массив аргументов функции;
func_num_args ..... возвращает количество аргументов, переданных функции;
function_exists ..... возвращает true, если указанная функция определена;
get_browser ..... сообщает возможности клиентского броузера;
ignore_user_abort ..... устанавливает необходимость прекращения выполнения скрипта в случае, если клиент прекращает соединение;
iptcparse ..... анализирует двоичный блок IPTC (http://www.xe.net/iptc/);
leak ..... утечка памяти;
pack ..... упаковка данных в двоичную строку;
register_shutdown_function ..... регистрация функции, выполняемой при отключении;
serialize ..... создает представление величины для ее хранения;
sleep ..... откладывает выполнение;
uniqid ..... создает уникальный идентификатор (id);

unpack ..... распаковывает данные из двоичной строки;
unserialize ..... создает значение из представляения, созданного с помощью serialize для хранения;
usleep ..... отложить выполнение на указанное количество микросекунд;
highlight_string ..... синтаксис строки;
highlight_file ..... синтаксис файла;
show_source ..... синтаксис файла.

Оригинальная документация по функциям PHP4 содержится на прилагаемом компакт-диске.

Приложение А

ФАЙЛ PHP.INI

Важным составным элементом процесса установки PHP является правильная конфигурация файла PHP.ini. В настоящем приложении содержится описание директив файла php.ini.

Файл PHP.ini состоит из следующих инструкций и комментариев к ним.

```
;Файл [PHP.ini]

;;;;;;;;;;;;;;;;;;;
; Что находится в этом файле ;
;;;;;;;;;;;;;;;;;;;
;С помощью этого файла можно управлять многими аспектами поведения
;интерпретатора PHP.

;Исходный файл php.ini-dist должен быть переименован в файл php.ini и
;скопирован в основной каталог, в котором находится система Winodws.
;Файл может находиться и в другой папке. PHP производит поиск этого
;файла в текущем каталоге, в каталогах, указанных в переменной окруже-
;ния PHPRC и в каталогах, которые могли быть заданы во время компиляции
;кода, поскольку существует возможность получения дистрибутива PHP в
;виде исходного программного кода. Директорий, заданный для поиска
;файла php.ini, может быть изменен, если будет указан аргумент -c в
;режиме задания командной строки.
```



```
;Синтаксис файла очень прост. Пробелы, а также строки, начинающиеся с
;символа точки с запятой, игнорируются, заголовки разделов ([Foo])
;также пропускаются.
;Однако такие заголовки в дальнейшем могут оказаться полезными.
;Директивы указываются с использованием следующего синтаксиса:
directive=value (директива=значение)
;Имена директив чувствительны к регистру, т.е., например, foo=bar – это
;не то же самое, что FOO=bar.
;Значением может быть строка, число или константа PHP, например, E_ALL
;или M_PI, одна из констант INI (например, On, Off, True, False, Yes,
;No, None), выражение (например, E_ALL & ~E_NOTICE или строка, помещен-
;ная в кавычки ("foo").
```

Выражения в файле INI могут быть составлены лишь с использованием побитовых операторов и скобок. В качестве операторов допускается использование

| (побитовый OR);
& (побитовый AND);
~ (побитовый NOT);
! (логический NOT).

Логические значения могут быть установлены в положительное (true) состояние с использованием величин 1, On, True и Yes, отрицательное состояние (false) устанавливается при помощи значений 0, Off, False и No.

Пустая строка может быть заменена простым пропуском. Так, директива foo=; устанавливает значение строки foo в виде пустой строки. При этом важно иметь в виду, что директива foo=none устанавливает пустую строку, а директива foo=«none» устанавливает значение foo в виде строки 'none'.

Все значения, содержащиеся в файле php.ini-dist соответствуют значениям, задаваемым по умолчанию. Если эти значения не будут изменены, то работа PHP будет происходить так же, как если бы файл php.ini не был установлен совсем.

```
;;;;;;;;;;;;;;;;;;;
; Опции языка ;
;;;;;;;;;;;;;;;;;;;
engine=On
;Включает язык PHP при работе с сервером  Apache
```

```
short_open_tag=On
;допускает использование ярлыка <?. Если выключен, то допускается
;использование только ярлыков <?php и <script>.

asp_tags=Off
;допускает использование ярлыков в стиле ASP, т.е. ярлыков <% %>.
precision=14
;количество значащих цифр при отображении чисел с плавающей точкой

output_buffering=Off
;Буферизация позволяет посылать заголовки, в том числе cookies.
;Буферизация может быть активизирована даже после того, как будет
;послано тело файла (что приведет, однако, к некоторому замедлению в
;работе PHP), это может быть осуществлено путем вызова функции буферизации.

output_handler=
;Задав эту директиву, мы имеем возможность перенаправить все выходные
;данные, создаваемые скриптами функции. В качестве такой функции можно
;указать, например, функцию "ob_gzhandler", которая будет упаковывать
;данные.

implicit_flush=Off
;Включение этой опции вызову функции flush() после кажого обращения к
;функции print() или echo(), а также после каждого HTML-блока.
;Включение этой опции рекомендуется только в целях отладки программ.

allow_call_time_pass_reference=On
;Приводит к передаче аргументов при ссылке на функцию.
;Безопасный режим
safe_mode=Off
safe_mode_exec_dir=
safe_mode_allowed_env_vars=PHP_

;По умолчанию. При этом пользователь имеет возможность устанавливать
;только такие переменные окружения, которые начинаются с префикса PHP_
;(например, PHP_FOO=BAR).
;Если эта директива не задана (пустая строка), то пользователь может
;устанавливать любые переменные окружения.

safe_mode_protected_env_vars=LD_LIBRARY_PATH
;Эта директива содержит список переменных окружения, разделенных запятой.
;Эти переменные не могут быть изменены при помощи функции putenv(), они
;будут защищены.
```



```
disable_functions=
;Эта директива позволяет выключить те или иные функции. Выключенные
;функции перечисляются в списке, при этом функции отделяются друг от
;друга запятыми. На эту директиву не влияет то, включен или выключен
;безопасный режим Safe Mode.
;Цвета, которыми будет показан текст того или иного элемента в
синтаксисе html.

highlight.string=#DD0000
;строки

highlight.comment=#FF8000
;комментарии

highlight.keyword=#007700
;ключевые слова

highlight.bg=#FFFFFF
;фон

highlight.default=#0000BB
;цвет по умолчанию

highlight.html=#000000
;цвет для HTML-текста

; Прочие
expose_php=On
;Устанавливет возможность отображения в подписи web-сервера того факта,
;что на сервере установлен PHP. Это позволит определить пользователю
;то, что ваш сервер поддерживает PHP.

;;;;;;;;;;;;;;;;;;;
; Ограничение ресурсов ;
;;;;;;;;;;;;;;;;;;;

max_execution_time=30
; Максимальное время в секундах, отведенное для обработки скрипта
memory_limit=8M
; Максимальный размер памяти, отводимый скрипту для работы
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Обработка ошибок ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Ошибки сообщаются в виде появления определенных значений того или
;иного бита (флаги ошибок). Для задания уровня сообщений об ошибках
;используются следующие значения:
; E_ALL            - все ошибки и предупреждения
; E_ERROR          - фатальные ошибки времени выполнения
; E_WARNING        - не фатальные ошибки (предупреждения) времени исполнения
; E_PARSE          - ошибки проверки (синтаксические ошибки)
; E_NOTICE         - замечания времени исполнения
; E_CORE_ERROR     - фатальные ошибки, произошедшие во время инициализации PHP
; E_CORE_WARNING   - предупреждения, появляющиеся во время инициализации PHP
; E_COMPILE_ERROR - фатальные ошибки компиляции
; E_COMPILE_WARNING  - предупреждения (не фатальнфе ошибки) компиляции
; E_USER_ERROR     - сообщение об ошибке, созданное пользователем
; E_USER_WARNING   - предупреждение, созданное пользователем
; E_USER_NOTICE    - замечание, созданное пользователем
; Примеры:
; error_reporting=E_ALL & ~E_NOTICE
; Выдаются все сообщения об ошибках, за исключением замечаний
; error_reporting=E_COMPILE_ERROR|E_ERROR|E_CORE_ERROR
; Выводятся только сообщения об ошибках (но не предупреждения)

error_reporting= E_ALL
; выводятся все сообщения об ошибках, предупреждения и замечания

display_errors=On
; Сообщения об ошибках выводятся в виде фрагмента текста, выводимого в
;общем содержании. Рекомендуется выключать эту опцию. Это особенно
;важно по соображениям безопасности. Сообщения об ошибках могут
;содержать небезопасную для сервера информацию, которая будет передана
;пользователю.

display_startup_errors=Off
; Задание этого параметра приведет к тому, что даже при включенном
;режиме сообщения об ошибках ошибки, возникающие при запуске PHP
;отображаться не будут. Рекомендуется всегда (за исключением случаев
;отладки) держать эту опцию выключенной.

log_errors=Off
; Если включена эта опция, то сообщения об ошибках выводятся в
;специальный файл
```



```
; Рекомендуется включать эту опцию, при этом следует отключить
;возможность вывода сообщений об ошибках пользователю (см. выше).

track_errors=Off
; Эта опция позволяет сохранять сообщения об ошибке в переменной
;$php_errormsg (boolean)
;error_prepend_string="<font color=ff0000>"
; строка, которая выводится перед сообщением об ошибке
;error_append_string="</font>"
; Строка, которая выводится сразу после сообщения об ошибке
;error_log=filename
; файл, куда выводятся сообщения об ошибках
;error_log=syslog
; сообщения об ошибках выводятся в системный файл ошибок (для WindowsNT)

warn_plus_overloading=Off

;;;;;;;;;;;;;;;;;
; Обработка данных ;
;;;;;;;;;;;;;;;;;
; Опция track_vars всегда включена (в PHP4), последовательность
;переменых устанавливается в виде:
:variables_order="EGPCS"
;Эта переменная сождержит порядок регистрации переменных (GET, POST,
;Cookie, Environment и встроенных переменных, для краткости G, P, C, E
;и S соответственно, что сокращенно обозначается как EGPCS или GPC).

register_globals=On
; задает необходимость регистрации переменных EGPCS в качестве глобальных
; переменныхvariables as global

register_argc_argv=On
;устанавливает режим декларирования переменных argv&argc, содержащих
информацию из метода GET

post_max_size=8M
;максимальный размер данных метода POST, принимаемых интерпретатором PHP

gpc_order="GPC"
; Вместо этой директивы следует использовать variables_order.
```

```
; Magic quotes
magic_quotes_gpc=On
;включает magic quotes для входящих данных из методов GET/POST и для
Cookie

magic_quotes_runtime=Off
; magic quotes для данных, сгенерированных в процесе работы, например,
;для данных, полученных из SQL, exec(), и т.п.
magic_quotes_sybase=Off
;Использование magic quotes в стиле Sybase ( ' в сочетании с ''  вместо \')

; Автоматически вставляет файл в начало или в конец документа PHP:
auto_prepend_file=
auto_append_file=

; PHP 4.0b4 всегда выводит заголовок, в котором указывается кодировка
;Content-type: header. Чтобы выключить эту опцию и не посылать заголовок
;с указанием кодировки, нужно установить эту директиву равной пустой строке.
; По умолчанию используется заголовок

default_mimetype="text/html"
;default_charset="iso-8859-1"

;;;;;;;;;;;;;;;;;;;;;;;;;
; Пути и каталоги;
;;;;;;;;;;;;;;;;;;;;;;;;;

include_path=
; дляUNIX: "/path1:/path2"
; для  Windows: "\path1;\path2"

doc_root=
; корневой директориий для файлов php, используется только в том
;случае, ЕСЛИ ЭТА ДИРЕКТИВА НЕ ПУСТА

user_dir=
;директорий, в котором находятся PHP-файлы скриптов, если в пути указан
;директорий вида /~username, используется только тогда, когда директива
;не пуста

extension_dir=./
```



```
; директорий, в котором расположены библиотеки расширений возможностей
;языка PHP

enable_dl=On
;Включена или выключена возможность использования функции dl().
;Функция dl() не работает должным образом на многопоточных серверах,
;например, на сервере IIS или Zeus, в этих случаях она автоматически
;выключается на этих серверах.

;;;;;;;;;;;;;;;;
; Пересылка файлов серверу ;
;;;;;;;;;;;;;;;;
file_uploads=On
; Включается или выключается возможность посылки файлов серверу
;с помощью HTTP
;upload_tmp_dir=
; временный директорий для размещения файлов, посланных на сервер
;с помощью HTTP

upload_max_filesize=2M
; максимальный размер файлов, посылаемых серверу

;;;;;;;;;;;;;;;;;;
; Параметры для Fopen ;
;;;;;;;;;;;;;;;;;;

allow_url_fopen=On
; Допускается ли использование адресов типа http:..., ftp:...
;в качестве имен файлов

;;;;;;;;;;;;;;;;;;;;;
; Динамические расширения ;
;;;;;;;;;;;;;;;;;;;;;
;Если требуется автоматическая загрузка расширений, то можно использовать
;синтаксис следующего вида: extension=modulename.extension
;для Winfows это может быть сделано так:
;extension=msql.dll
;в UNIX
;extension=msql.so
;Заметим, что здесь должно быть указано лишь имя файла расширения, но
;не весь путь к файлу целиком.
;Местоположение файла указывается в директиве extension_dir.
```

```
;Расширения для Windows
;Расширения для MySQL и ODBC встроены в PHP и дополнительные *.dll
;файлы для работы с ними не требуются.
;
;extension=php_bz2.dll
;extension=php_ctype.dll
;extension=php_cpdf.dll
;extension=php_curl.dll
;extension=php_cybercash.dll
;extension=php_db.dll
;extension=php_dba.dll
;extension=php_dbase.dll
;extension=php_domxml.dll
;extension=php_dotnet.dll
;extension=php_exif.dll
;extension=php_fdf.dll
;extension=php_filepro.dll
;extension=php_gd.dll
;extension=php_gettext.dll
;extension=php_ifx.dll
;extension=php_iisfunc.dll
;extension=php_imap.dll
;extension=php_interbase.dll
;extension=php_java.dll
;extension=php_ldap.dll
;extension=php_mhash.dll
;extension=php_mssql65.dll
;extension=php_mssql70.dll
;extension=php_oci8.dll
;extension=php_openssl.dll
;extension=php_oracle.dll
;extension=php_pdf.dll
;extension=php_pgsql.dll
;extension=php_printer.dll
;extension=php_sablot.dll
;extension=php_snmp.dll
;extension=php_sybase_ct.dll
;extension=php_yaz.dll
;extension=php_zlib.dll
```



```
;;;;;;;;;;;;;;;;;;;
; Конфигурирования модулей ;
;;;;;;;;;;;;;;;;;;;

[Syslog]
define_syslog_variables=Off
; требуется или нет задание различных системных переменных таких как
; $LOG_PID, $LOG_CRON, и т.п. Во время выполнения программы эти
;переменные могут быть заданы с использованием define_syslog_variables().


[mail function]
SMTP= mail.server. name.ru
; только для for Win32, имя почтового сервера
sendmail_from= me@localhost.com
; только для Win32, значение строки заголовка почты From
;sendmail_path=
;только для unix, могут быть использованы аргументы (по умолчанию
;'sendmail -t -i')

[Debugger]
debugger.host=localhost
debugger.port=7869
debugger.enabled=False

[Logging]
;logging.method=db
;logging.directory=/path/to/log/directory

[Java]
;java.class.path=.\php_java.jar
;java.home=c:\jdk
;java.library=c:\jdk\jre\bin\hotspot\jvm.dll
;java.library.path=.\


[SQL]
sql.safe_mode=Off
```

```
[ODBC]
;odbc.default_db=Еще не используется
;odbc.default_user=Еще не используется
;odbc.default_pw=Еще не используется
odbc.allow_persistent=On
; разрешает или нет использование постоянных связей
odbc.check_persistent=On
;проверяет, является ли установленная связь все еще открытой перед
;повторным использованием
odbc.max_persistent=-1
; Максимальное число одновременно существующих постоянных соединений. -1
;обозначает неограниченное количество
; odbc.max_links=-1
; макисчмальное число связей,  -1 — не ограничено
odbc.defaultlrl=4096
; Использование длинных полей. Возвращает количество переменных.
odbc.defaultbinmode=1
; обработка двоичных данных: 0 — пропускается, 1 — оставлять как есть,
;2 — преобразовывать в символы

[MySQL]
mysql.allow_persistent=On
;разрешает или запрещает постоянные соединения
mysql.max_persistent=-1
;максимльное число постоянных соединений, -1 — не ограничено
mysql.max_links=-1
;общее максимальное чило соединений (постояных и не постоянных),
;-1 — без ограничений
mysql.default_port=
;номер порта по умолчанию для mysql_connect(). Если значение не
;установлено, то используется $MYSQL_TCP_PORT или the mysql-tcp из
;/etc/services или определенное во время компиляции значение MYSQL_PORT
;(в перечисленном порядке).  Win32 использует только MYSQL_PORT.
mysql.default_socket=
; имя сокета, используемое по умоляанию для соединений MySQL. Если не
;указано, то используется встроенное имя.Значнеия по умоляанию для
;MySQL
mysql.default_host=
; хост используемый по умолчанию в mysql_connect()
mysql.default_user=
; пользователь, используемый по умолчанию в mysql_connect()
mysql.default_password=
```



```
; пароль, используемый по умолчанию в mysql_connect();
; Необходимо напомнить, что хранение паролей в этом файле — это не
;лучшая идея. Любой пользователь может запустить команду
;'echo cfg_get_var("mysql.default_password")' и узнать хранящиеся в
;этом файле пароли.

[mSQL]
msql.allow_persistent=On
; позволяет или запрещает использование постоянных связей
msql.max_persistent=-1
; максимальное количество постоянных связей, -1 — не ограничено
msql.max_links=-1
; общее максимальное количество связей(постоянных и непостоянных),
;-1 — не ограничено

[PostgresSQL]
pgsql.allow_persistent=On
; разрешает или запрещает использование постоянных связей
pgsql.max_persistent=-1
; максимальное число постоянных связей, -1 — нет ограничений
pgsql.max_links=-1
; максимальное количество связей (постоянных и непостоянных),
; -1 означает отсутствие ограничений

[Sybase]
sybase.allow_persistent=On
; разрешает или запрещает использование постоянных связей
sybase.max_persistent=-1
; максимальное число постоянных связей, — 1 — не ограничено
sybase.max_links=-1
; общее максимальное число связей  -1 — не ограничено
;sybase.interface_file="/usr/sybase/interfaces"
sybase.min_error_severity=10
; минимальное количество показываемых ошибок
sybase.min_message_severity=10
sybase.compatability_mode=Off
; режим совместимости со старыми версиями PHP 3.0.

[Sybase-CT]
sybct.allow_persistent=On
; позволяет или запрещает использование постоянных соединений
sybct.max_persistent=-1
```

```
; макисмальное количество постоянных соединений, -1 — нет ограничений
sybct.max_links=-1
; максимальное количество соединений, -1 — нет ограничений
sybct.min_server_severity=10
sybct.min_client_severity=10

[bcmath]
bcmath.scale=0
;количество десятичных цифр для всех функций
[browscap]
;browscap=extra/browscap.ini

[Informix]
ifx.default_host=
; хост по умолчанию для ifx_connect() (не работает в безопасном режиме)
ifx.default_user=
; пользователь по умолчанию для ifx_connect() (не работает в безопасном
;режиме)
ifx.default_password=
; пароль по умолчанию для ifx_connect() (не работает в безопасном
;режиме)
ifx.allow_persistent=On
; резрешено или нет постоянные соединения
ifx.max_persistent=-1
; максимальное число постоянных соединений, — 1 — нет ограничений
ifx.max_links=-1
; максимальное число соединений (общее число), — 1 — нет ограничений
ifx.textasvarchar=0
ifx.byteasvarchar=0
ifx.charasvarchar=0
ifx.blobinfile=0
ifx.nullformat=0


[Session]
session.save_handler=files
; используется для хранения и получения данных
session.save_path= D:\Program Files\php\sessiondata
; аргумент, передаваемый средству управления сессиями, указываетя путь к
;каталогу, в котором хранятся файлы данных
session.use_cookies=1
; используются или нет cookies
```



```
session.name=PHPSESSID
; имя сессии
; используется как имя cookie
session.auto_start=0
; сессия инициализируется в начале запроса
session.cookie_lifetime=0
; время жизни cookie в секундах
; если указано 0, то время жизни cookie устанавливается действительным
;до момента повторного старта броузера
session.cookie_path=/
; путь, для которого действительны cookie
session.cookie_domain=
; домен, к которому относятся cookie
session.serialize_handler=php
; обработчик, используемый для сериализации данных
; php — стандартный обработчик
session.gc_probability=1
session.gc_maxlifetime=1440
; по истечении указанного количества секунд данные будут
; рассматриваться как мусор и будут удалены
session.referer_check=
; проверка ссылок HTTP
session.entropy_length=0
; количество байтов, читаемых из файла.
session.entropy_file=
; указывается для задания идентификатора сессии id
;  session.entropy_length=16
;  session.entropy_file=/dev/urandom
session.cache_limiter=nocache
; устанавливается равным {nocache,private,public}
; для определения последовательности кеширования
session.cache_expire=180
; время, по истечении которого документ устаревает, в минутах
session.use_trans_sid=1
; используется поддержка transient sid
; by compiling with -enable-trans-sid
url_rewriter.tags="a=href,area=href,frame=src,input=src,form=fakeentry"

[MSSQL]
mssql.allow_persistent=On
; разрешает или запрещает использование постоянных соединений
```

```
mssql.max_persistent=-1
; масимальное количество постоянных соединений, – 1 – обозначает
;отсутствие ограничений
mssql.max_links=-1
; общее максимальное количество соединений, – 1 – нет ограничений
mssql.min_error_severity=10
mssql.min_message_severity=10
mssql.compatability_mode=Off
; режим совместимости с  PHP 3.0.
;mssql.textlimit=4096
; допускается в интервале от 0 до – 21474836470 по умолчанию = 4096
;mssql.textsize=4096
; допускается в интервале от 0 до – 21474836470 по умолчанию = 4096

;mssql.batchsize=0
;ограничение количества записей в одном фрагменте, 0 соответствует
тому, ;что все записи делаются в одном фрагменте.

[Assertion]
;assert.active=On
;assert.warning=On
;assert.bail=Off
;assert.callback=0
;assert.quiet_eval=0

[Ingres II]
ingres.allow_persistent=On
; разрешает или запрещает использование постоянных соединений
ingres.max_persistent=-1
; максимальное число постоянных соединений, – 1 – нет ограничений
ingres.max_links=-1
; общее максимальное число соединений, – 1-нет ограничений
ingres.default_database=
; база данных, используемая по умолчанию в формате  [node_id::]dbname[/
srv_class]
ingres.default_user=
; пользователь по умолчанию
ingres.default_password=
; пароль по умолчанию
[Verisign Payflow Pro]
pfpro.defaulthost="test.signio.com"
```



```
; Signio сервер по умолчанию
pfpro.defaultport=443
; порт по умолчанию, с которым производится соединение
pfpro.defaulttimeout=30
; временная задержка, допустимая по умолчанию
; pfpro.proxyaddress=
; адрес прокси-сервера, используемый по умолчанию
; pfpro.proxyport=
; порт прокси-сервера по умолчанию
; pfpro.proxylogon=
; имя для прокси-сервера по умолчанию
; pfpro.proxypassword=
; пароль для прокси-сервера по умолчанию
[Sockets]
sockets.use_system_read=On
; Использовать системную функцию read() вместо php_read().
; Локальные переменные:
; tab-width: 4
; End:
```

Содержание

Введение ... 3

Глава 1.

Что такое PHP ... 6

Что такое PHP? ... 6

Наш первый PHP-файл ... 8

Глава 2.

Установка Web-сервера и PHP ... 13

Установка Web-сервера и PHP ... 13

Конфигурирование PHP ... 18

Основные команды в файле php.ini ... 19

Конфигурирование почты ... 22

Конфигурирование безопасной настройки (Safe Mode) ... 22

Конфигурирование отладчика ... 23

Конфигурирование загрузки расширений (Extension Loading) ... 23

Конфигурирование MySQL ... 23

Конфигурирование mSQL ... 24

Конфигурирование Postgres ... 24

Конфигурирование Sybase ... 24

Конфигурирование Sybase-CT ... 24

Конфигурирование BC Math ... 25

Конфигурирование возможностей броузера ... 25

Конфигурирование ODBC ... 25

Проверяем работоспособность сервера ... 26

Создаем простейший PHP-сценарий и размещаем его на сервере ... 33

Глава 3.

Первые программы на PHP ... 36

Первые программы на PHP ... 36

Вывод запрашиваемого файла ... 38

Формы и передача информации клиентом серверу ... 40

Вывод файла в зависимости от информации о клиенте ... 45

Отправка почты средствами сервера ... 48

Глава 4.

Работа с файлами ... 53

Отправка файла в качестве почтового приложения ... 57

Отправка файла серверу ... 60

Простой счетчик посещений ... 64

Глава 5.

Рисунки. Информация о PHP ... 68

Вывод рисунков ... 68

Создаем рисунок ... 68

Скрытый вывод картинок ... 71

Внешние рисунки ... 73

Информация о PHP ... 75

Информация, полученная с помощью phpinfo() ... 76

PHP Version 4.0.4pl1 ... 76

Configuration ... 77

Environment ... 83

PHP Variables ... 84

PHP License ... 86

Текст в рисунках ... 87

Иные возможности ... 91

Глава 6.

Основные элементы языка ... 92

Базовые элементы ... 92

Выделение фрагмента PHP-кода в тексте phtm-файла ... 92

Типы ... 94

Целые числа ... 95

Числа с плавающей точкой ... 95
Строки ... 95
Массивы ... 99
Объекты ... 102
Приведение типов ... 102
Объявление типов ... 103
Переменные ... 104
Основные понятия ... 104
Заранее определенные переменные ... 106
Область видимости переменных ... 109
Переменные переменных (имена переменных) ... 112
Внешние переменные ... 114
Переменные окружения ... 115
Константы ... 116
Инструкции ... 118

Глава 7.
Управление ходом выполнения программы ... 121
Операторы ... 121
Арифметические операторы ... 122
Операторы присваивания ... 122
Побитовые операторы ... 123
Операторы сравнения ... 123
Операторы обработки ошибок ... 124
Оператор выполнения системных команд ... 124
Операторы увеличения и уменьшения ... 125
Логические операторы ... 126
Иерархия операторов ... 127
Операторы для работы со строками ... 127
Управление последовательностью выполнения инструкций ... 128
if ... 128
else ... 129
elseif ... 129
Альтернативный синтаксис операторов ... 130
while ... 131
do..while ... 131

for ... 132
foreach ... 134
break ... 135
continue ... 136
switch ... 137
Некоторые функции, используемые для управления последовательностью выполнения инструкций ... 139
Функции ... 147
Функции, определяемые пользователем ... 147
Аргументы функции ... 147
Значения, возвращаемые функцией ... 151
old_function ... 152
Переменные функции ... 152
Классы и объекты ... 153
class ... 153
Ссылки ... 156
Что представляют собой ссылки ... 156
Для чего используются ссылки ... 156
Для чего нельзя использовать ссылки ... 157
Ссылки, возвращающие значения в функциях ... 157
Удаление ссылок ... 158
Глобальные ссылки ... 158
$this ... 158

Краткий справочник.
Функции в PHP ... 159
Функции для работы с сервером Apache ... 159
Функции для работы с массивами ... 160
Математические функции BCMath ... 169
Функции для работы с датами ... 171
Функции для работы с классами и объектами ... 176
Функции работы с датой и временем ... 183
Функции для работы с каталогами ... 191
Функции XML/DOM ... 194
Функции для работы с файловой системой ... 195

Функции для работы с FTP 214
Функции для работы на уровне протокола HTTP 219
Функции для работы с изображениями 221
Функции imap для работы с почтой 239
Функции для работы с почтой 255
Математические функции 256
Функции для работы с сетью 266
Функции определения и установки параметров интерпретатора PHP. 272
Функции выполнения системных программ 285
Функции для работы с сессиями 287
Функции для работы со строками 291
Функции для работы с переменными 309
Функции XML-анализатора 316

Приложение А. *Файл PHP.ini* 328

БЛАНК ЗАКАЗА (принимаются ксерокопии)

Книги по компьютерным технологиям

Автор	Название книги	Цена Россия руб.	Цена Украина грн.	Кол-во в заказе
Николенко	Практические занятия по JavaScript + дискета	65.00	15.90	
Цеховой	Macromedia FLASH 5 + дискета	75.00	18.00	
Николенко	Самоучитель по Visual C++6	95.00	20.10	
Николенко	MIDI - язык богов (+CD)	68.00	15.00	
Цеховой	Web: дизайн и коммерция	58.00	14.04	
Костельцев	Руссификация Linux RedHat 7.0 + CD	75.00	18.00	
Картузов	Программирование на языке Java	66.00	16.00	

Книги по радиоэлектронике

Автор	Название книги	Цена Россия руб.	Цена Украина грн.	Кол-во в заказе
Брускин	Заруб. резидентные радиотелефоны. Изд.2	65.00	13.00	
Брускин	Схемотехника автоответчиков (+14 схем)	80.00	16.00	
Виноградов	Ист. питания (трансформаторные) ВМ и ВП	55.00	11.00	
Виноградов	Сервисные режимы телевизоров	51.00	12.30	
Виноградов	Источники питания видеомагнитофонов	116.00	27.60	
Заболотный	Заруб. транзист. и диоды: A...Z. Справочник	75.00	15.00	
Каменецкий	Радиотелефоны (+ 35 схем)	125.00	25.00	
Корякин-Черняк	Телефонные аппараты от А до Я. Изд. 2	92.00	22.08	
Кульский	КВ-приемник мирового уровня	60.00	12.00	
Куприянов	Тех. обеспеч. цифр. обраб. сигналов. Справ.	125.00	25.00	
Кучеров	Источники питания мониторов	76.00	18.30	
Лаврус	Практика измерений в телевизионной технике	30.00	6.00	
Лукин	ТВ ближнего зарубежья (3УСЦТ, 4УСЦТ, Банга)	42.00	8.40	
Мухин	Энциклопедия мобильной связи. Т.1	76.00	18.30	
Панков	Энц. телемаст. Том 1. Схемотехника отеч. ТВ	118.00	23.60	
Партала	Видеокамеры (+ 13 схем)	80.00	16.00	
Партала	Цифровые КМОП-микросхемы	112.00	23.40	
Пестриков	Энциклопедия радиолюбителя	92.00	18.40	
Рубаник	Усовершенствование телевизоров 3...5УСЦТ	85.00	17.00	
Ульрих	Микроконтроллеры PIC16C7X	69.01	13.80	
Янковский	Видеомагнитофоны серии ВМ. (+ 18 схем)	160.00	32.00	
Янковский	Энц. телемастера. Том 3. Блоки пит. ТВ. Кн. 1.	80.00	19.20	

Внимание!!! В цену книг не включены почтовые расходы, которые в зависимости от страны составляют на Украине до 20%, в России 20...40% от стоимости заказа.