第四篇 PHP的高级技术

第19章 与文件系统和服务器的交互

在第2章中，我们已经了解了如何在Web服务器上读数据，以及如何将数据写入一个文件中。本章将介绍允许与Web服务器文件系统进行交互的PHP其他函数。

在本章中，我们将主要介绍以下内容：

- 使用PHP上传文件
- 使用目录函数
- 与服务器上的文件进行交互
- 在服务器上运行程序
- 使用服务器环境变量

为了讨论这些函数的使用，我们首先来了解一个例子。考虑到我们可能会希望客户能够更新一些网站内容，例如，关于他们公司的最新消息（或者我们可能想为自己设计一个比FTP更友好的用户界面）。解决此类问题的一种方法是让客户以普通文本的方式上传文件内容。通过使用由PHP设计的模板，这些文件是可以在网站中访问到的，正如我们在第6章中所介绍的一样。

在开始研究文件系统函数之前，先让我们简单地了解一下文件是如何上传的。

19.1 文件上传

PHP的一个非常有用的功能是它支持文件上传。不是使用HTTP将服务器上的文件传递到客户端的浏览器，而是以相反的方向执行——使用HTTP将文件从客户端浏览器传递到服务器。通常，可以使用HTML表单的界面实现它。图19-1所示的就是在这个例子中我们将使用的HTML表单界面。

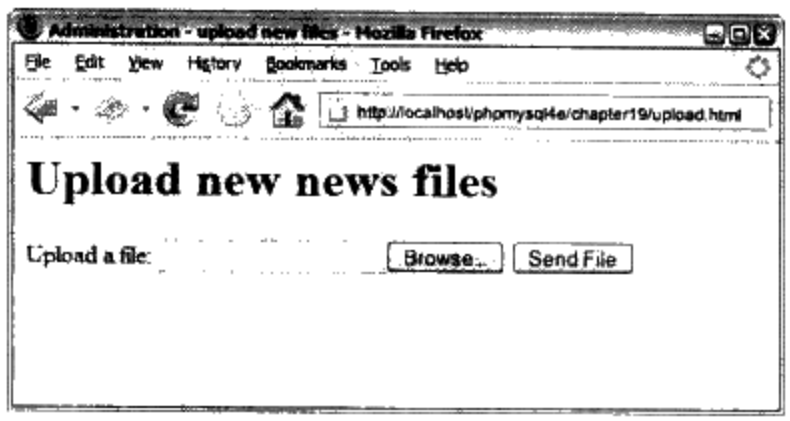

图19-1 用于文件上载的HTML表单包含了与普通HTML表单不同的域和域类型

正如你可以看到的，以上表单包含一个文本输入框，在这个输入框中，用户可以输入文件名，或者点击“Browse”按钮来浏览或寻找本地的有效文件。稍后我们将详细介绍如何实现它。

在输入文件名后，用户可以点击“Send File”按钮将文件上传到服务器，而服务器上的PHP脚本正等着处理它。

在进入文件上传的例子之前，需要注意的是，php.ini文件具有五个能够控制PHP如何处理文件上传的指令。这些指令和默认值及其相关描述如表19-1所示。

表19-1　php.ini中关于文件上传的设置指令

指　令	描　述	默 认 值
file_uploads	控制是否允许HTTP方式的文件上传。允许值为On或Off	ON（启用）
upload_tmp_dir	指定上传的文件在被处理之前的临时保存目录。如果没有设置该选项，将使用系统默认值	NULL
upload_max_filesize	控制允许上传的文件最大大小。如果文件大小大于该值，PHP将创建一个文件大小为0的占位符文件	2M
post_max_size	控制PHP可以接受的，通过POST方法上传数据的最大值。该值必须大于upload_max_filesize设置值，因为它是所有POST数据的大小，包括了任何上传的文件	8M

19.1.1　文件上传的HTML代码

为了实现文件上传功能，需要用到一些专门用于上传文件的HTML语法。上例表单的HTML源代码如程序清单19-1所示。

程序清单19-1　upload.html——上传文件的HTML表单

```
<html>
<head>
  <title>Administration - upload new files</title>
</head>
<body>
<h1>Upload new news files</h1>
<form action="upload.php" method="post" enctype="multipart/form-data"/>
  <div>
    <input type="hidden" name="MAX_FILE_SIZE" value="1000000" />
    <label for="userfile">Upload a file:</label>
    <input type="file" name="userfile" id="userfile"/>
    <input type="submit" value="Send File"/>
  </div>
</form>
</body>
</html>
```

请注意，该表单使用了POST方法。文件上传也可以使用为Netscape Composer和Amaya所支持的PUT方法。但是，这需要对该脚本进行修改，因为这两种浏览器不支持GET方法。

这个表单的其他特性包括：

- 在<form>标记中，必须设置属性enctype="multipart/form-data"，这样，服务器就可以知道上传的文件带有常规的表单信息。
- 必须有一个可以设置上传文件最大长度的表单域。这是一个隐藏的域，如下所示：

```
<input type= "hidden' name= "MAX_FILE_SIZE" value=" 1000000">
```

请注意，MAX_FILE_SIZE表单域是可选的，该值也可以在服务器端设置。然而，如果在这个表单中使用，表单域的名称必须是MAX_FILE_SIZE。其值是允许人们上传文件的最大长度值（按字节计算）。在这里，我们将其设置为1 000 000B（几乎就是1MB）。针对你自己的应用程序，可以将该值设置为更大或更小。

- 你需要指定文件类型，如下所示：

```
<input type="file' name='userfile" id-"userfile"/>
```

可以为文件选择喜欢的任何名字，但必须记住，将在PHP接收脚本中使用这个名字来访问文件。

提示　在我们进一步实现该功能之前，必须注意有些PHP版本在文件上传的代码中存在安全漏洞。如果决定在产品服务器上使用文件上传功能，必须确定所使用的PHP是最新版本，而且要时刻注意PHP的补丁发布。

当然，这并不能妨碍使用这样一个有用的技术，但是应该非常小心所编写的代码，考虑限制用户对文件上传功能的使用，例如，只允许站点管理员和内容管理员使用该功能。

19.1.2　编写处理文件的PHP

编写捕获上载文件的PHP代码是相当直观和简单的。

当文件被上传时，该文件将保存在临时目录中，这是通过php.ini文件的upload_tmp_dir指令设置的。正如表19-1所介绍的，如果没有设置该指令，在默认情况下，该目录是Web服务器上的主临时目录。如果在脚本执行完成之前不移动、复制或更改文件名称，该文件将被删除。

在PHP脚本中，需要处理的数据保存在超级全局数组$_FILES中。如果开启了register_globals指令，也可以直接通过变量名称访问这些信息。但是，这可能就是关闭register_globals指令的原因，或者至少认为是关闭了该指令，并且使用超级全局数组而忽略全局变量。

保存$_FILES数组中的元素时，将同时保存HTML表单的<file>标记名称。表单元素名称是userfile，因此该数组将具有如下所示的内容：

- 存储在$_FILES['userfile']['tmp_name']变量中的值就是文件在Web服务器中临时存储的位置。
- 存储在$_FILES['userfile']['name']变量中的值就是用户系统中的文件名称。
- 存储在$_FILES['userfile']['size']变量中的值就是文件的字节大小。
- 存储在$_FILES['userfile']['type']变量中的值就是文件的MIME类型，例如：text/plain或image/gif。

■ 存储在$_FILES['userfile']['error']变量中的值将是任何与文件上传相关的错误代码。这是在PHP 4.2.0中增加的新特性。

如果已经知道上传文件的位置及其名称，现在，就可以将其复制到其他有用的地方。在脚本执行结束前，这个临时文件将被删除。因此，如果要保留上传文件，必须将其重命名或移动。

在我们的例子中，我们打算将上传文件作为最新的新闻文章，因此，我们将删除文件中可能的任何标记，再将它们移动到其他有用的目录，/uploads/目录。请注意，在Web服务器的根目录下，你需要创建一个名为uploads的目录。程序清单19-2给出了能够实现此功能的脚本。

程序清单19-2　upload.php——从HTML表单中获得文件的PHP脚本

```
<html>
<head>
  <title>Uploading...</title>
</head>
<body>
<h1>Uploading file...</h1>
<?php

  if ($_FILES['userfile']['error'] > 0)
  {
    echo Problem: ';
    switch ($_FILES['userfile']['error'])
    {
    case 1: echo'File exceeded upload_max_filesize';
            break;
    case 2: echo'File exceeded max_file_size';
            break;
    case 3: echo'File only partially uploaded';
            break;
    case 4: echo'No file uploaded';
            break;
    case 6: echo 'Cannot upload file: No temp directory specified';
            break;
    case 7: echo 'Upload failed: Cannot write to disk';
            break;
  }
  exit;
}

// Does the file have the right MIME type?
if ($_FILES['userfile']['type'] !='text/plain')
{
  echo'Problem: file is not plain text';
  exit;
}

// put the file where we'd like it
```

```
  $upfile ='/uploads/'.$_FILES['userfile']['name'] ;

  if (is_uploaded_file($_FILES['userfile']['tmp_name']))
  {
    if (!move_uploaded_file($_FILES['userfile']['tmp_name'], $upfile))
    {
       echo'Problem: Could not move file to destination directory';
       exit;
    }
  }
  else
  {
    echo'Problem: Possible file upload attack. Filename:';
    echo $_FILES['userfile']['name'];
    exit;
  }

  echo'File uploaded successfully<br><br>';

  // remove possible HTML and PHP tags from the file's contents
$contents = file_get_contents($upfile);
  $contents = strip_tags($contents);
file_put_contents($_FILES['userfile']['name'], $contents);
  // show what was uploaded
  echo '<p>Preview of uploaded file contents:<br/><hr/>';
  echo nl2br($contents);
  echo'<br/><hr/>';
?>
</body>
</html>
```

有趣的是，以上脚本主要进行的操作是错误检测。文件上传存在潜在的安全危险，因此，我们应该尽量避免这些安全风险。我们还需要尽可能地测试文件以确认上传的文件是否适合显示给网站的访问者。

下面，我们来具体了解以上代码。首先，我们要检查在`$_FILES['userfile']['error']`中返回的错误代码。同样，每一个错误代码都有一个相关的错误常量。这些可能的常量和错误代码如下所示：

- `UPLOAD_ERROR_OK`，值为0，表示没有发生任何错误。
- `UPLOAD_ERR_INI_SIZE`，值为1，表示上传文件的大小超出了约定值。文件大小的最大值是在PHP配置文件中指定的，该指令是`upload_max_filesize`。
- `UPLOAD_ERR_FORM_SIZE`，值为2，表示上传文件大小超出了HTML表单的`MAX_FILE_SIZE`元素所指定的最大值。
- `UPLOAD_ERR_PARTIAL`，值为3，表示文件只被部分上传。

- UPLOAD_ERR_NO_FILE，值为4，表示没有上传任何文件。
- UPLOAD_NO_TMP_DIR，值为6，表示在php.ini文件中没有指定临时目录（在PHP 5.0.3版本引入）。
- UPLOAD_ERR_CANT_WRITE，值为7，表示将文件写入磁盘失败（在PHP 5.1.0版本引入）。

如果使用旧版本的PHP，也可以使用PHP手册给出的或本书以前版本给出的示例代码执行“手工版本”的检查操作。

也可以检查MIME类型。在这个例子中，我们只希望上传文本文件。因此通过确认$_FILES['userfile']['type']包含text/plain可以检测MIME类型。这只是错误检查。它不是安全性检查。MIME类型可以从用户浏览器判断文件扩展名获得并传递给服务器。如果希望通过传递一个错误的类型来“蒙混过关”，这对一些恶意的用户是很难的。

然后，我们再检查要打开的文件是否已经真正被上传而且不是一个本地文件（例如，/etc/passwd）。稍后，我们将详细介绍这一点。

如果所有的工作进展正常，我们可以将上传的文件复制到包含目录中。在这个例子中，我们使用/uploads/目录——它不存在于Web文档树结构中，因此是存放文件的适当位置，因为这些文件应该存放于Web文档树之外。

接下来，我们打开这个文件，使用strip_tags()函数清除任何HTML标记或PHP标记，并保存该文件。最后，显示文件的内容。这样，用户就可看到他们的文件被成功上传。

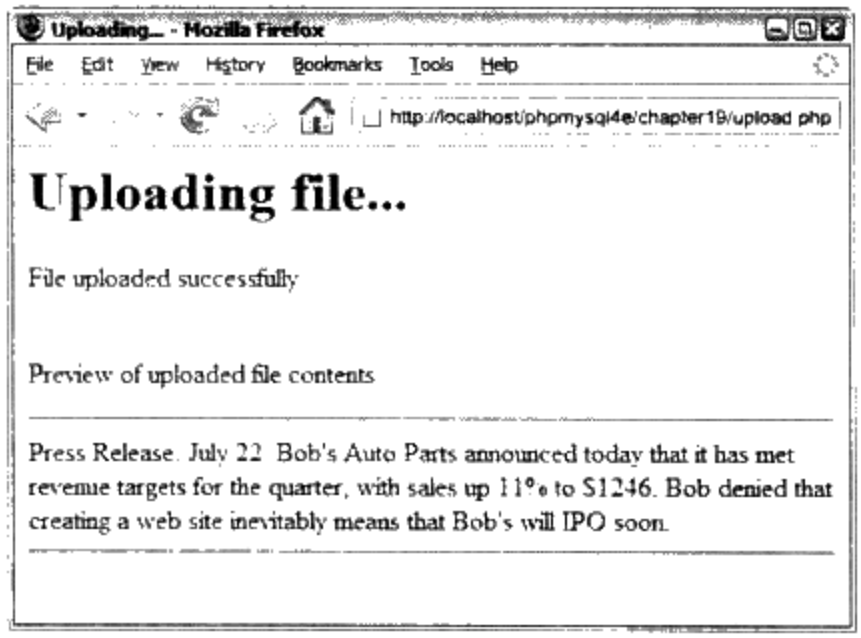

图19-2　在复制和重新格式化文件后，向用户显示成功上载文件的确认信息

图19-2所示的是一个成功运行脚本的结果。

在2000年9月，出现了一种软件，它能够让侵入者修改文件上传脚本，使此脚本可以将本地文件当成上传的文件进行处理。这个软件的文档保存在BUGTRAQ邮件发送清单中。

我们可以在许多BUGTRAQ文档中看到正式的安全咨询，例如：http: //lists.insecure.org/ bugtraq/2000/Sep/0237.html。

要确保自己不易受到攻击，这个脚本使用了is_uploaded_file()函数和move_uploaded_file()函数来确保所处理的文件已经被上传，而且不是一个本地文件，例如/etc/passwd。这个函数在PHP 4.0.3版本及其以后的版本都可以使用。

如果在编写文件上传脚本时考虑的问题不够全面，一些恶意的访问者可能会提供一个他自己的临时文件名称，并且使脚本认为这个文件就是上传的文件。由于许多文件上传脚本都会向用户显示所上传的文件内容，或者将其保存在可以载入该文件的地方，这就可能导致人们能够访问只有Web服务器才能访问的文件。这就可能包括一些敏感文件，例如，/etc/passwd和包括数据库密码的PHP源代码。

19.1.3 避免常见上传问题

在上传文件时，需要注意以下几个问题：

- 前面的例子是假设用户已经在某些地方通过了身份验证。我们不应该允许任何人都可以上传文件到网站。
- 如果允许不信任的或者没有通过身份验证的用户上传文件，就必须对文件的内容作出严格的规定。我们最不希望出现的是上传和运行恶意的脚本。应该非常小心，不仅是文件的类型和内容，还有文件本身的名字。一个好办法是将我们认为是“安全”的上传文件进行重命名。
- 要降低用户“浏览Web服务器目录”的风险，你可以使用basename()函数来修改所接收文件的名称。这个函数将过滤作为文件名称一部分而传入的目录路径，这是在服务器的不同目录下放置文件的常见攻击手段。该函数示例如下所示：

```
<?php
    $path = '/home/httpd/html/index.php';
    $file1    basename($path);
    $file2 = basename($path, ".php");
    print $file1 . "<br />";          // the value of $file1 is 'index.php'
    print $file2 . "<br />";          // the value of $file2 is 'index'
```

- 如果使用的是基于Windows的机器，通常情况下，必须确保在文件路径中用“\\”或“/”替代“\”。
- 就像我们以上脚本所做的，使用用户提供的文件名称可能会导致一系列的问题。最明显的问题就是如果上传的文件名称已经存在，可能会意外地覆盖已有文件。还有一个不是非常明显的风险是不同的操作系统，甚至是本地的不同语言设置将允许文件名称中包含不同的合法字符集。对于一个被上传的文件来说，其文件名称可能包含了对系统来说是非法的字符。
- 如果文件上传脚本的运行出现问题，请检查PHP配置文件。需要将upload_tmp_dir指令设置成指向有访问权限的目录。如果要上传大文件，同时可能还需要调整memory_limit指令；该指令决定能够上传的最大文件字节数。Apache还有一些可配置的超时设置和事务大小限制，如果在上传大文件时遇到问题，可以考虑检查这些设置。

19.2 使用目录函数

在用户上传一些文件之后，能够看到所上传的文件，并可以操作这些文件的内容对他们来说是非常必要的。PHP提供了一系列目录函数与文件系统函数，它们都是实现此功能的非常有用的工具。

19.2.1 从目录读取

首先，我们将实现一个能够浏览所上传文件的目录的脚本。在PHP中，浏览目录实际上是非常简单的。在程序清单19-3中，我们给出了能够实现此功能的示例脚本。

程序清单19-3　browsedir.php—能够列出上传文件的目录

```
<html>
<head>
  <title>Browse Directories</title>
</head>
<body>
<h1>Browsing</h1>
<?php
  $current_dir = '/uploads/';
  $dir = opendir($current_dir);

  echo "<p>Upload directory is $current_dir</p>";
  echo '<p>Directory Listing:</p><ul>';

  while(false !== ($file = readdir($dir)))
  {
    //strip out the two entries of . and ..
      if($file != "." && $file != "..")
      {
        echo  <li>$file</li>';
      }
  }
  echo '</ul> ;
    closedir($dir);
?>
</body>
</html>
```

以上脚本使用了函数opendir()、closedir()和readdir()。

函数opendir()用于打开所浏览的目录。这类似于用函数fopen()打开所读取的文件。只是传递给此函数的参数不是文件名称，而是一个目录名称：

```
$dir = opendir ($current_dir);
```

该函数将返回一个目录句柄，这一点与函数fopen()返回文件句柄是非常类似的。

在目录打开后，可以通过调用函数readdir($dir)从目录中读取文件，正如本例所示。当该目录中没有可读的文件时，此函数将返回false。请注意，当此函数读取一个名为“0”的文件时，也将返回false——为了确保这一点，可以通过测试来确定返回值是否为false。

```
while(false !== ($file = readdir($dir)))
```

当完成从目录中读取文件的步骤后，可以通过调用函数closedir($dir)关闭该目录，这也类似于调用函数fclose()来关闭文件。

图19-3所示的是目录浏览脚本的输出示例。

通常，.（当前目录）和..（上一级）目录也会显示在图19-3所示的清单中。但是，我们使用了如下所示的语句将它们过滤了：

```
if($file != "." && $file != "..")
```

如果删除了这行代码，.（当前目录）和..（上一级）目录就会出现在目录清单中。

如果通过这种机制提供浏览目录的功能，那么限制可浏览的目录是很明智的，因为这样用户就不能够浏览特殊的目录列表。

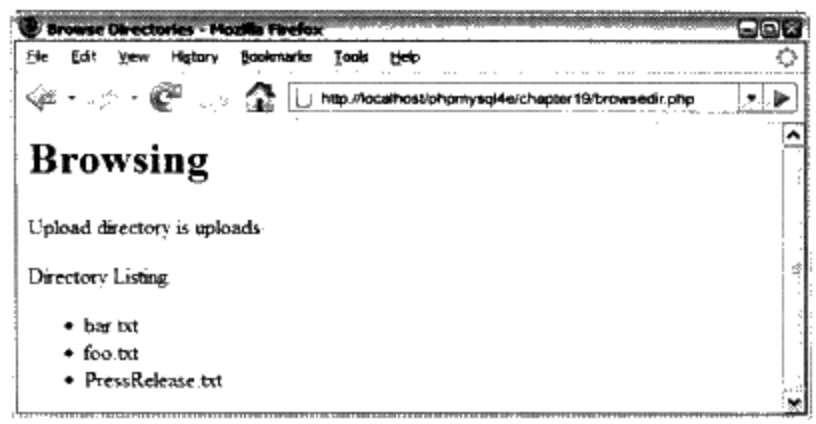

图19-3　目录清单列出了指定目录中的所有文件

在这里，一个相关的并且是非常实用的函数是rewinddir($dir)，此函数将所读的文件恢复到开始的目录。

除了这些函数外，还可以使用PHP提供的dir类，该类具有handle和path属性，以及read()、close()和rewind()方法，它们与以上所介绍的非类函数有着相同的作用。

在程序清单19-4中，我们使用dir类重新编写了以上示例。

程序清单19-4　browsedir2.php——使用dir类显示目录列表

```
<html>
<head>
  <title>Browse Directories</title>
</head>
<body>
<h1>Browsing</h1>
<?php
  $dir = dir("uploads/");

  echo "<p>Handle is $dir->handle</p>";
  echo "<p>Upload directory is $dir->path</p>";
  echo '<p>Directory Listing:</p><ul>';

  while(false !== ($file = $dir->read()))
    //strip out the two entries of . and ..
      if($file != "." && $file != "..")
      {
        echo "<li>$file</li>";
      }
  echo '</ul>';
  $dir->close();
?>
</body>
</html>
```

以上示例中的文件名称并没有以任何顺序保存，因此如果需要一个有序列表，你可以使用scandir()函数，该函数是在PHP 5中引入的。这个函数可以将文件名称保存在一个数组，并且以字母表的顺序排序，升序或降序，如程序清单19-5所示。

程序清单19-5　scardir.php——使用scandri()函数对文件名称进行字母表方式排序

```
<html>
<head>
  <title>Browse Directories</title>
</head>
<body>
<h1>Browsing</h1>
<?php
$dir = '/uploads/' ;
$files1 = scandir($dir);
$files2 = scandir($dir, 1);

echo "<p>Upload directory is $dir</p>";
echo '<p>Directory Listing in alphabetical order, ascending:</p><ul>';


foreach($files1 as $file)
{
        if($file != "." && $file != "..")
    echo "<li>$file</li>";
}

echo  </ul>';
echo '<p>Upload directory is $dir</p>";
echo '<p>Directory Listing in alphabetical, descending:</p><ul>';


foreach($files2 as $file)
{
        if($file != "." && $file != "..")
    echo "<li>$file</li>";
}

echo '</ul>';

?>
</body>
</html>
```

19.2.2　获得当前目录的信息

我们可以获得给定文件路径的额外信息。

dirname(*$path*)函数和basename(*$path*)函数将分别返回路径的目录部分和路径的文件名称部分。对于目录浏览器来说，这可能会非常有用，尤其是根据有意义的目录名称和文

件名称内容创建一个复杂的目录结构时。

使用disk_free_space(*$path*)函数，我们也可以在目录列表中增加一项说明，表示还有多少空间可以用来保存上传文件。如果给这个函数传递了一个目录的路径，它将返回磁盘（在Windows系统下）或该目录所在的文件系统（在UNIX系统下）上的空余空间（以字节为单位）。

19.2.3 创建和删除目录

除了可以被动地读取信息和目录外，还可以用PHP的mkdir()和rmdir()函数来创建与删除目录，当然只能在用户具有访问权限的路径上创建和删除目录。

函数mkdir()的使用比想像中要复杂。它带有两个输入参数：目标目录的路径（包括新的目录名）和希望该目录拥有的访问权限，例如：

```
mkdir("/tmp/testing", 0777);
```

然而，要得到的权限码不一定是上面所需的权限码，当前的umask将与该值相“与”（像做一次减法），这样才能获得真正的权限码。例如，如果umask是022，那么得到的权限码是0755。

在创建具有这个效果的目录之前，我们可能会重新设置umask码。可以输入如下命令：

```
$oldumask = umask(0);
mkdir("/tmp/testing", 0777);
umask($oldumask);
```

以上代码使用了umask()函数，该函数可以用来检查并修改当前的umask码。它可以将当前的umask码修改为任何其他所希望并传入的umask码，并且返回原来的umask码。

或者，如果该函数在被调用时不带有任何其他参数，它就会返回当前umask码。

请注意，umask()函数对Windows系统是没有作用的。

函数rmdir()将删除一个目录，例如：

```
rmdir("/tmp/testing");
```

或者

```
rmdir("c:\\tmp\\testing");
```

请注意，要删除的目录必须为空目录。

19.3 与文件系统的交互

除了能够查看并获得与目录有关的信息外，我们还可以获取Web服务器上文件的信息，并且与这些文件进行交互。前面我们已经了解了如何读文件和写文件。我们还可以使用许多其他的文件函数。

19.3.1 获取文件信息

我们可以对目录浏览器脚本的部分内容进行修改，使其能够读取文件，如下所示：

```
 while(false !== ($file = readdir($dir))) {
echo '<a href="filedetails.php?file='.$file.'">'.$file.'</a><br>';
}
```

这样，我们就可以创建脚本filedetails.php，用来提供与文件相关的详细信息。这个文件的内容如程序清单19-6所示。

使用该脚本的警告：Windows下不支持或者不可靠地支持这里所使用的一些函数，包括posix_getpwuid()函数、fileowner()函数和filegroup()函数。

程序清单19-6　filedetails.php——文件状态函数和它们的运行结果

```
<html>
<head>
  <title>File Details</title>
</head>
<body>
<?php
  $current_dir = '/uploads/';
  $file = basename($file); // strip off directory information for security

  echo '<h1>Details of file: '.$file.'</h1>';

  echo '<h2>File data</h2>';
  echo 'File last accessed: '.date('j F Y H:i', fileatime($file)).'<br>';
  echo 'File last modified: '.date('j F Y H:i', filemtime($file)).'<br>';

  $user = posix_getpwuid(fileowner($file));
  echo 'File owner: '.$user['name'].'<br>';

  $group = posix_getgrgid(filegroup($file));
  echo 'File group: '.$group['name'].'<br>';

  echo 'File permissions: '.decoct(fileperms($file)).'<br>';

  echo 'File type: '.filetype($file).'<br>';

  echo 'File size: '.filesize($file).'bytes<br>';


  echo '<h2>File tests</h2>';

  echo 'is_dir: '.(is_dir($file)? 'true': 'false').'<br>';
  echo 'is_executable: '.(is_executable($file)? 'true': 'false').'<br>';
  echo 'is_file: '.(is_file($file)? 'true': 'false').'<br>';
  echo 'is_link: '.(is_link($file)? 'true': 'false').'<br>';
  echo 'is_readable: '.(is_readable($file)? 'true': 'false').'<br>';
  echo 'is_writable: '.(is_writable($file)? 'true': 'false').'<br>';

?>
</body>
</html>
```

程序清单19-6运行的结果如图19-4所示。

现在，我们讨论一下程序清单19-6中所用到的函数所实现的功能。正如前面所提到的，函数basename()获得的是不带目录的文件名（也可以使用函数dirname()获得不带文件名称的目录名称）。

函数fileatime()和filemtime()将分别返回该文件最近被访问和最近被修改的时间戳。可以通过函数date()来重新格式化时间戳，这样使得它具有更好的可读性。在一些操作系统中，这些函数将返回同样的值（如这个例子中），这取决于系统所存储的信息。

函数fileowner()和filegroup()将分别返回文件的用户标识（uid）和组标识（gid）。通过分别使用函数posix_getpwuid()和posix_getgrgid()，我们将它们转变成容易理解的名字。这些函数将uid或gid作为参数，并返回关于用户或者组的相关数组，包括该脚本用到的用户或组的名字。

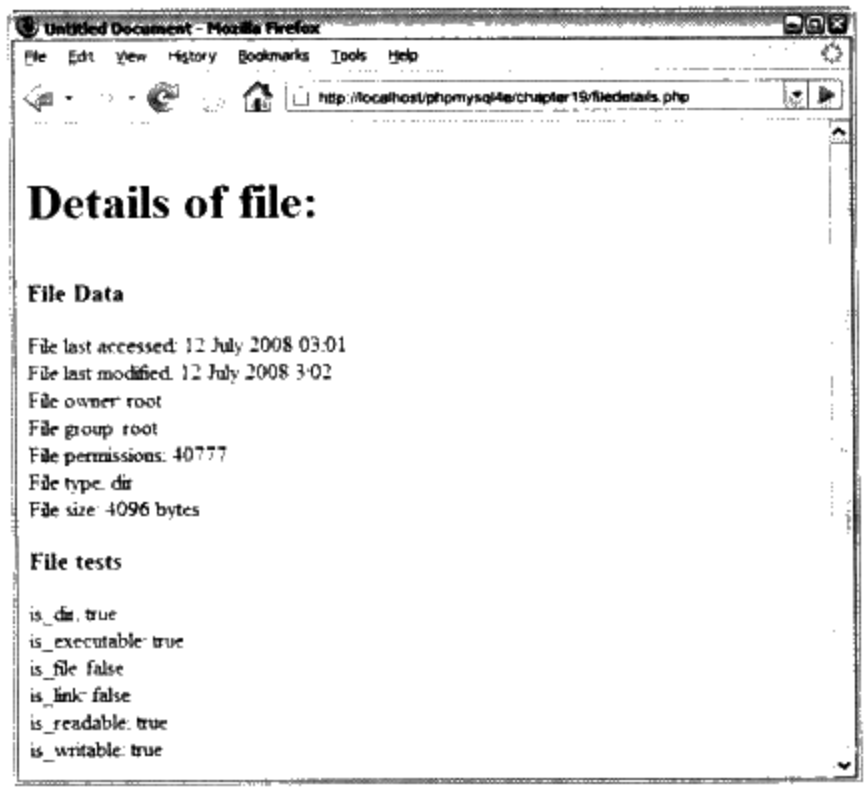

图19-4　文件内容视图显示关于一个文件的系统信息。请注意显示的权限码是用八进制表示的

函数fileperms()返回文件权限码。我们可以使用函数decoct()将权限码重新格式化为八进制数，该格式对于UNIX用户来说非常熟悉。

函数filetype()返回一些关于所检查文件的类型信息。可能的结果有fifo、char、dir、block、link、file和unknown。

函数filesize()返回文件的大小（以字节计算）。

另一组函数——is_dir()、is_executable()、is_file()、is_link()、is_readable()和is_writable()——每一个函数都将检测一个文件的指定属性并返回true或false。

或者，也可以通过函数stat()获得许多同样的信息。当传递给此函数一个文件名时，它将返回与上述函数组所返回的数据类似的数组。函数lstat()也与之类似，但是仅用于符号链接文件。

所有的文件状态函数的运行都很费时间。因此它们的结果将被缓存起来。如果要在变化之前或者变化之后检查文件信息，需要调用函数：

```
clearstatcache();
```

来清除以前的缓存结果。如果希望在改变文件数据之前或者之后使用以前的脚本，应该先调用此函数来更新产生的数据。

19.3.2　更改文件属性

除了能够查看文件的属性外，还可以更改文件属性。

函数chgrp(*file,group*)、chmod(*file,permissions*)、chown(*file,user*)的功能都类似于UNIX中的同名函数。它们都不能在基于Windows的系统中运行，尽管函数chown()也可以执行，但是它将总是返回true。

函数chgrp()用于修改文件的组。但是此函数只能用于将组改成该用户所在的组，root用户除外。

函数chmod()用于修改文件的权限。传递给此函数的权限码是UNIX中常见的chmod格式——如果按八进制方式来显示它们，应该给它们冠以前缀“0”。例如：

```
chmod('somefile.txt', 0777);
```

函数chown()用于修改文件的所有者。只有管理员运行该脚本时才使用本函数，这种情况最好不要发生。除非专门从命令行下运行这个脚本来执行一个管理任务。

19.3.3　创建、删除和移动文件

可以使用文件系统函数创建、移动和删除文件。

首先，可以很简单地调用函数touch()来创建一个文件，或者修改文件上次被修改的时间。其功能类似UNIX的命令touch。该函数具有如下所示的原型：

```
bool touch (string file, [int time [, int atime]])
```

如果要创建的文件已经存在，它的修改时间将会改成当前的时间，或者所给出的第二个时间参数。如果要指定时间，就应该使用时间戳格式。如果文件不存在，将创建此文件。

文件访问时间也将被修改：在默认情况下，将修改为当前的系统时间，或者在可选的*atime*参数中指定的时间戳。

也可以使用函数unlink()来删除一个文件（请注意，这个函数的名称不是delete——PHP中没有delete函数）。可以按如下方式使用此函数：

```
unlink($filename);
```

你可以使用函数copy()和rename()来复制和重命名（移动）文件，如下所示：

```
copy($source_path, $destination_path);

rename($oldfile, $newfile);
```

可以注意到在程序清单19-2中我们已经使用了函数copy()。

因为PHP中并没有提供move()函数，作为一个函数，rename()完成了两项操作，它可以将一个文件从一个位置移动到另一个位置。是否能够将一个文件从一个文件系统移动到另一个系统，以及当使用rename()函数时文件是否会被覆盖，这都取决于操作系统，因此我们需要检查服务器上运行结果。此外，还必须注意文件名的路径。如果有关联的话，路径所指的应该是相对脚本的地址，而不是原文件的地址。

19.4　使用程序执行函数

现在，我们要将视线从文件系统函数转移到能够运行服务器命令的函数。

当要为一个已有的命令行系统提供一个基于Web的前台时，这些函数是非常有用的。例如，我们使用这些命令为邮件清单管理器ezmlm建立了一个前台。当我们在本书后面的部分讨论实例时，将再次使用这些函数。

可以使用4种主要的技术在Web服务器上执行命令。这些技术都非常相似，但也有一些微小的区别：

■ exec()—exec()函数原型如下所示：

```
string exec (string command [, array &result  , int &return_value]])
```

可以将要执行的命令作为输入参数传递给该函数，例如：

```
exec("ls -la");
```

该函数exec()没有直接的输出。此函数将返回命令结果的最后一行。

如果以result传递一个变量，此函数将返回字符串组数，这些字符串代表输出的每一行。

如果以return_value传递一个变量，将获得返回代码。

■ passthru()—函数passthru()原型如下所示：

```
void passthru (string command [, int return_value ]
```

函数passthru()直接将输出显示到浏览器。（如果输出是二进制，例如一些图像数据，这是非常有用的。）此函数无返回值。

此函数的参数与函数exec()的参数格式一样。

■ system()—函数system()原型如下：

```
string system (string command [, int return_value])
```

此函数将该命令的输出回显到浏览器。它将把每一行的输出向后对齐（假设以服务器模块运行PHP），这就是与passthru()函数的不同之处。此函数返回输出（如果成功运行）的最后一行或false（如果失败）。

函数的参数与其他函数的参数格式一样。

■ 反引号——我们在第1章中简单地提到过它。这些实际上是可执行的操作符。它们没有直接的输出。执行这个命令的结果是以字符串的形式返回的，我们可以决定是否显示这些字符串。

如果有更复杂的需求，也可以使用popen()、proc_open()和proc_close()函数，这些函数可以启动外部进程，并且在这些进程之间传递数据。其中，后两个函数是在PHP 4.3版本中新添加的。

程序清单19-7所示的脚本说明如何使用这些函数来实现相同的功能。

程序清单19-7　progex.php——文件状态函数和它们的输出结果

```
<?php
```

```
    chdir('/uploads/');

/////  exec version
    echo  <pre>';

    // unix
    exec('ls -la', $result);
    // windows
    // exec('dir', $result);
    foreach ($result as $line)
      echo "$line\n";

    echo '</pre>';
    echo '<br><hr><br>';

/////  passthru version
    echo '<pre>';

    // unix
    passthru( 'ls -la') ;
    // windows
    // passthru( 'dir');

    echo '</pre> ;
    echo '<br><hr><br>';

/////  system version

    echo '<pre>';
    // unix
    $result = system( 'ls -la');
    // windows
    // $result = system( 'dir');
    echo  </pre>';
    echo  <br><hr><br>';

/////backticks version

    echo '<pre>';
    // unix
    $result = `ls -al`;
    // windows0
    // $result = `dir`;
    echo $result;
    echo '</pre>';

? >
```

我们可以使用以上方法中的任何一种方法来替代前面所介绍的目录浏览脚本。请注意，以上代码充分说明了使用外部函数的一个缺点——代码不再是可移植的。在这里，我们使用了UNIX命令，因此，这些代码将肯定无法在Windows上运行。

如果打算把用户提交的数据包括在将执行命令的一部分，首先应该通过函数escapeshellcmd()来运行它。此函数将阻止用户在系统上恶意地执行命令。可以按如下方式来调用这个函数：

```
system(escapeshellcmd($command));
```

也应该使用escapeshellarg()函数来整理任何要传递给shell命令的参数。

19.5 与环境变量交互：getenv()和putenv()

在本章结束之前，我们介绍一下如何在PHP中使用环境变量。PHP提供了两个函数来使用环境变量：getenv()和putenv()，其中getenv()函数能够获得环境变量值，而putenv()函数能够设置环境变量值。请注意，这里所说的环境变量是指运行PHP的服务器上的环境变量。

运行phpinfo()函数，可以获得PHP所有环境变量的列表。其中的一些变量比其他变量更有用处，例如：

```
getenv( 'HTTP_REFERER');
```

将返回用户来到当前页之前的上一页面URL。

根据需要，也可以调用函数putenv()来设置环境变量，如下所示：

```
$home = '/home/nobody';
putenv ( "HOME=$home");
```

如果你是一个系统管理员并且希望限制程序员可以设置的环境变量，你可以在PHP的配置文件中对safe_mode_allowed_env_vars指令进行设置。当PHP在安全模式下运行时，用户只能对具有该指令给出的带有前缀的环境变量进行设置。

提示　如果希望了解更多关于环境变量的信息，可以查看CGI说明，请访问如下URL：http: //hoohoo.ncsa.uiuc.edu/cgi/env.html。

19.6 进一步学习

PHP中的大多数文件系统函数都与操作系统函数对应——如果正在使用UNIX，可以使用man手册页获取更多信息。

19.7 下一章

在第20章中，我们将运用PHP的网络函数和协议函数与Web服务器以外的系统进行交互，这将再一次扩展脚本的应用范围。