文件处理系统

网址：www.lampbrother.net
电话：400 700 1307

无兄弟 不编程

文件处理系统

1. 文件系统概述
2. 目录的基本操作
3. 文件的基本操作
4. 文件的上传与下载
5. 文件系统项目实战

一、文件系统概述

- 1.1 文件类型
- 1.2 文件的属性

1.1 文件类型

- 在程序运行时，程序本身和数据一般都存在内存中，当程序运行结束后，存放在内存中的数据被释放。
- 如果需要长期保存程序运行所需的原始数据，或程序运行产生的结果，就必须以文件形式存储到外部存储介质上。
- 文件一般指存储在外部介质上具有名字（文件名）的一组相关数据集合。用文件可长期保存数据，并实现数据共享。
- PHP是以UNIX的文件系统为模型的。因此在Windows系统中我们只能获得”file”、”dir”或者“unknown”三种文件类型。而在UNIX系统中，我们可以获得block、char、dir、fifo、file、link和unknown七种类型。
- 可以使用函数filetype()获取文件的具体类型。
  - 语法: string filetype (string filename)

Liunx系统中7种文件类型说明

文件类型	描述
block	块设备文件，如某个磁盘分区，软驱，光驱CD-ROM等
char	字符设备是指在I/O传输过程中以字符为单位进行传输的设备，如键盘、打印机等
dir	目录类型，目录也是文件的一种
fifo	命名管道，常用于将信息从一个进程传递到另一个进程
file	普通文件类型，如文本文件或可执行文件等。
link	符号链接，是指向文件指针的指针。类似Windows中的快捷方式
unknown	未知类型。

- is_dir( ) -- 判断给定文件名是否是一个目录
  语法结构: bool is_dir（名称）
  返回类型: 如果文件名存在并且是一个目录则返回 true，否则返回 false。
- is_executable( ) -- 判断给定文件名是否可执行
  语法结构: bool is_executable（名称）
  返回类型: 如果文件存在且可执行则返回 true ，否则返回 false 。
- is_file( ) -- 判断给定文件名是否为一个正常的文件
  语法结构: bool is_file(名称)
  返回类型: 如果文件存在且为正常的文件则返回 true 。
- is_link( ) -- 判断给定文件名是否为一个符号连接
  语法结构: bool is_link(名称)
  返回类型: 如果文件存在并且是一个符号连接则返回 true。
- is_readable( ) -- 判断给定文件名是否可读
  语法结构: bool is_readable（文件名称）
  返回类型: 如果文件存在并且可读则返回 true 。
- is_writable( ) -- 判断给定的文件名是否可写
  语法结构: bool is_writable(文件名称)
  返回类型: 如果文件存在并且可写则返回 true 。

1.2 文件的属性

函数名	作用	参数	返回值
file_exists( )	检查文件或目录是否存在	文件名	存在：true，不存在：false
filesize()	取得文件大小	文件名	返回大小字节数，出错：false
is_readable()	判断文件是否可读	文件名	文件可读返回true
is_writable()	判断文件是否可写	文件名	文件可写返回true
is_executable()	判断文件是否可执行	文件名	文件可执行返回true
filectime()	获取文件的创建时间	文件名	返回UNIX时间戳格式
filemtime()	获取文件的修改时间	文件名	返回UNIX时间戳格式
fileatime()	获取文件的访问时间	文件名	返回UNIX时间戳格式
stat()	获取文件大部分属性	文件名	返回给定属性信息的数组

二、目录的基本操作

- 2.1 解析目录路径
- 2.2 遍历目录
- 2.3 统计目录大小
- 2.4 建立与删除目录
- 2.5 复制与移动目录

2.1 解析目录路径

- 使用PHP脚本可以方便对目录进行操作，如创建目录、遍历目录、复值目录与删除目录等操作。
- 常用的文件目录路径格式:
  - `$unixPath="/var/www/html/index.php";`
    `//在UNIX系统中的绝对路径，必须使用"/"分隔`
  - `$winPath="C:\\Appserv\\www\\index.php";`
    `//在Windows系统的绝对路径，默认使用"\"分隔`
  - `$winPath2="C:/Appserv/www/index.php";`
    `//在Windows系统中也可使用“/”分隔。`
- 注意使用绝对路径与相对路径。

PHP文件路径相关函数

- basename -- 返回路径中的文件名部分
  - 语法: string basename (string path [, string suffix])
  - 给出一个包含有指向一个文件的全路径的字符串，本函数返回基本的文件名。如果文件名是以 *suffix* 结束的，那这一部分也会被去掉。
- dirname -- 返回路径中的目录部分
  - 语法: string dirname (string path)
  - 给出一个包含有指向一个文件的全路径的字符串，本函数返回去掉文件名后的目录名。

```php
<?php
  $path = "/home/httpd/html/index.php";
  $file = basename($path);           // $file值: "index.php"
  $file = basename($path, ".php "); // $file值: "index "
  $file = dirname($path);     // $file值: "/home/httpd/html"
?>
```

- pathinfo -- 返回文件路径的信息
  - 语法: array pathinfo (string path [, int options])
  - pathinfo() 返回一个联合数组包含有 *path* 的信息。包括以下的数组单元: *dirname*, *basename* 和 *extension*。

```php
<?php
    $path_parts = pathinfo("/www/htdocs/index.html");
    echo $path_parts["dirname"] . "\n";     // /www/htdocs
    echo $path_parts["basename"] . "\n";    // index.html
    echo $path_parts["extension"] . "\n";   // html
?>
```

- realpath -- 返回规范化的绝对路径名
  - 语法: string realpath (string path)
  - realpath() 扩展所有的符号连接并且处理输入的 *path* 中的 '/./', '/../' 以及多余的 '/' 并返回规范化后的绝对路径名。返回的路径中没有符号连接, '/./' 或 '/../' 成分。

2.2 遍历目录

- opendir -- 打开目录句柄
  - 语法: resource opendir (string path [, resource context])
  - 打开一个目录句柄，可用于之后的 closedir()，readdir() 和 rewinddir() 调用中。
- readdir -- 从目录句柄中读取条目
  - 语法: string readdir (resource dir_handle)
  - 返回当前目录指针位置的文件名，没有返回false，并将指针向下移动一位。文件名以在文件系统中的排序返回。
- closedir -- 关闭目录句柄
  - 语法: void closedir (resource dir_handle)
  - 关闭由 *dir_handle* 指定的目录流。流必须之前被 opendir() 所打开。
- rewinddir -- 倒回目录句柄
  - 语法: void rewinddir (resource dir_handle)
  - 将 *dir_handle* 指定的目录流重置到目录的开头。

2.3 统计目录大小

- disk_free_space -- 返回目录中的可用空间
  - 语法: float disk_free_space (string directory)
  - 给出一个包含有一个目录的字符串，本函数将根据相应的文件系统或磁盘分区返回可用的字节数。
- disk_total_space -- 返回一个目录的磁盘总大小
  - 语法: float disk_total_space (string directory)
  - 给出一个包含有一个目录的字符串，本函数将根据相应的文件系统或磁盘分区返回所有的字节数。

```php
<?php   //自定义一个函数dirSize(), 统计传入参数的目录大小
   function dirSize($directory) {
      $dir_size=0;//初值为0, 用来累加各文件大小从而计算目录大小
      if($dir_handle=@opendir($directory)){  //打开目录并判断成功打开
        while($filename=readdir($dir_handle)) {  //循环遍历目录
           if($filename!="." && $filename!="..") { //排除特殊的目录
             $subFile=$directory."/".$filename;    //将文件和目录相连
             if(is_dir($subFile))                  //如果为目录
                $dir_size+=dirSize($subFile);    //求子目录的大小
             if(is_file($subFile))                 //如果是文件
                $dir_size+=filesize($subFile);   //求出文件的大小并累加
           }
        }
        closedir($dir_handle);                    //关闭文件资源
        return $dir_size;                         //返回计算后的目录大小
        }
   }
   $dir_size=dirSize("phpMyAdmin"); //调函数计算目录大小, 返回目录大小
   echo round($dir_size/pow(1024,1),2)."KB";//将目录字节换为“KB”单位
?>
```

2.4 建立与删除目录

- mkdir -- 新建目录
  - 语法: bool mkdir (string pathname [,int mode])
  - 尝试新建一个由 *pathname* 指定的目录。
- rmdir -- 删除目录
  - 语法: bool rmdir (string dirname)
  - 尝试删除 *dirname* 所指定的目录。 该目录必须是空的，而且要有相应的权限。如果成功则返回 TRUE，失败则返回 FALSE。
- unlink -- 删除文件
  - 语法: bool unlink (string filename)
  - 删除 *filename*。和 Unix C 的 unlink() 函数相似。如果成功则返回 TRUE，失败则返回 FALSE。

```php
<?php  //自定义函数递归的删除整个目录
 function delDir($directory) {
   if(file_exists($directory)) {  //判断目录是否存在，如果存在则执行
     if($dir_handle=@opendir($directory)){ //打开返回目录资源，并判断
       while($filename=readdir($dir_handle)){ //遍历目录读出目录中信息
        if($filename!="." && $filename!="..") { //一定要排除两个特殊目录
          $subFile=$directory."/".$filename;//将目录下文件和当前目录相连
          if(is_dir($subFile))                //如果是目录条件则成立
             delDir($subFile);                //递归调用自己删除子目录
          if(is_file($subFile))               //如果是文件条件则成立
             unlink($subFile);                //直接删除这个文件
        }
       }
       closedir($dir_handle);                          //关闭目录资源
       rmdir($directory);                              //删除空目录
     }
   }
 }
 delDir("phpMyAdmin");  //调用函数，将程序所在目录中phpMyAdmin文件夹删除
?>
```

2.5 复制与移动目录

- copy -- 拷贝文件
  - 语法：bool copy (string source, string dest)
  - 将文件从 *source* 拷贝到 *dest*。如果成功则返回 TRUE，失败则返回 FALSE。
- PHP中没有提供复制与移动目录的相关函数。如需要，只要自定义函数了。

三、文件的基本操作

- 3.1 文件的打开与关闭
- 3.2 写入文件
- 3.3 读取文件内容
- 3.4 访问远程文件
- 3.5 移动文件指针
- 3.6 文件的锁定机制
- 3.7 文件的一些基本操作函数

3.1 文件的打开与关闭

- fopen -- 打开文件或者 URL
  - 语法: resource fopen (string filename, string mode [, bool use_include_path [, resource zcontext]])
  - fopen() 将 *filename* 指定的名字资源绑定到一个流上。如果 *filename* 是 "scheme://..." 的格式，则被当成一个 URL，PHP 将搜索协议处理器（也被称为封装协议）来处理此模式。如果该协议尚未注册封装协议，PHP 将发出一条消息来帮助检查脚本中潜在的问题并将 *filename* 当成一个普通的文件名继续执行下去。
  - *mode* 参数指定了所要求到该流的访问类型。
  - 如果也需要在 include_path中搜寻文件的话，可以将可选的第三个参数 *use_include_path* 设为 '1' 或 TRUE。
  - 如果打开失败，本函数返回 FALSE。
- fclose -- 关闭一个已打开的文件指针

表格 1. fopen() 中的 *mode* 的可能值列表

mode	说明
'r'	只读方式打开，将文件指针指向文件头。
'r+'	读写方式打开，将文件指针指向文件头。
'w'	写入方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在则尝试创建之。
'w+'	读写方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在则尝试创建之。
'a'	写入方式打开，将文件指针指向文件末尾。如果文件不存在则尝试创建之。
'a+'	读写方式打开，将文件指针指向文件末尾。如果文件不存在则尝试创建之。
'x'	创建并以写入方式打开，将文件指针指向文件头。如果文件已存在，则 **fopen()** 调用失败并返回 **FALSE**，并生成一条 **E_WARNING** 级别的错误信息。如果文件不存在则尝试创建之。这和给 底层的 *open(2)* 系统调用指定 *O_EXCL|O_CREAT* 标记是等价的。此选项被 PHP 4.3.2 以及以后的版本所支持，仅能用于本地文件。
'x+'	创建并以读写方式打开，将文件指针指向文件头。如果文件已存在，则 **fopen()** 调用失败并返回 **FALSE**，并生成一条 **E_WARNING** 级别的错误信息。如果文件不存在则尝试创建之。这和给 底层的 *open(2)* 系统调用指定 *O_EXCL|O_CREAT* 标记是等价的。此选项被 PHP 4.3.2 以及以后的版本所支持，仅能用于本地文件。

3.2 写入文件

- fwrite -- 写入文件（可安全用于二进制文件）
  - 语法: int fwrite (resource handle, string string [, int length])
  - fwrite() 把 *string* 的内容写入 文件指针 *handle* 处。 如果指定了 *length*，当写入了 *length* 个字节或者写完了 *string* 以后，写入就会停止，视乎先碰到哪种情况。返回写入的字符数，出现错误时则返回 FALSE 。

3.3 读取文件内容

- fread -- 读取文件（可安全用于二进制文件）
  - string fread (int handle, int length)
  - fread() 从文件指针 *handle* 读取最多 *length* 个字节。 该函数在读取完 *length* 个字节数，或到达 EOF 的时候，或（对于网络流）当一个包可用时就会停止读取文件，视乎先碰到哪种情况。

```
<?php
   $handle = fopen ("http://www.example.com/", "rb");
   $contents = "";
   while (!feof($handle)) {
     $contents .= fread($handle, 8192);
   }
   fclose($handle);
?>
```

- fgets -- 从文件指针中读取一行
  - 语法: string fgets (int handle [,int length])
  - 从 *handle* 指向的文件中读取一行并返回长度最多为 *length* - 1 字节的字符串。碰到换行符（包括在返回值中）、EOF 或者已经读取了 length - 1 字节后停止（看先碰到那一种情况）。如果没有指定 *length*，则默认为 1K，或者说 1024 字节。
- fgetc -- 从文件指针中读取字符
  - 语法: string fgetc (resource handle)
  - 返回一个包含有一个字符的字符串，该字符从 *handle* 指向的文件中得到。碰到 EOF 则返回 FALSE。

- file -- 把整个文件读入一个数组中
  - 语法: array file (string filename [, int use_include_path [, resource context]])
  - 和 readfile() 一样，只除了 file() 将文件作为一个数组返回。数组中的每个单元都是文件中相应的一行，包括换行符在内。如果失败 file() 返回 FALSE。
- readfile -- 输出一个文件
  - 语法: int readfile (string filename [, bool use_include_path [, resource context]])
  - 读入一个文件并写入到输出缓冲。
  - 返回从文件中读入的字节数。如果出错返回 FALSE 并且除非是以 @readfile() 形式调用，否则会显示错误信息。

3.4 访问远程文件

- 如果需要访问远程文件，必须在PHP的配置文件中激活“allow_url_fopen”选项，才能使用fopen()函数打开运程文件。而且还要确定其他服务器中的文件是否访问权限，如果使用PHP协议对远程文件进行链接，只能以“只读”模式打开。如果需要访问的远程FTP服务器中，对所提供的用户开启了“可写”权限，则使用FTP协议链接远程的文件时，就可以使用“只写”或“只读”模式打开文件。但不可以使用“可读可写”的模式。
- 使用PHP访问远程文件就像访问本地文件一样，都是使用相同的读写函数处理。
- $file=fopen(“http://www.lampbrother.com/”,”r”) or die(“打开远程文件失败！！”);
- $file=fopen(“ftp://user:password@ftp.lampbrother.net/path/to/file”,”w”);

3.5 移动文件指针

- ftell -- 返回文件指针读/写的位置
  - 语法: int ftell (resource handle)
  - 返回由 handle 指定的文件指针的位置，也就是文件流中的偏移量。 如果出错，返回 FALSE。文件指针必须是有效的，且必须指向一个通过 fopen()或 popen()成功打开的文件。
- fseek -- 在文件指针中定位
  - 语法: int fseek (resource handle, int offset [, int whence])
  - 在与 *handle* 关联的文件中设定文件指针位置。新位置，从文件头开始以字节数度量，是以 *whence* 指定的位置加上 *offset*。*whence* 的值定义为:
  - SEEK_SET - 设定位置等于 *offset* 字节。
  - SEEK_CUR - 设定位置为当前位置加上 *offset*。
  - SEEK_END - 设定位置为文件尾加上 *offset*。（要移动到文件尾之前的位置，需要给 *offset* 传递一个负值。）
  - 如果 没有指定 *whence*，默认为 SEEK_SET。
  - 成功则返回 0；否则返回 -1。注意移动到 EOF 之后的位置不算错误

- rewind -- 倒回文件指针的位置
  - 语法: bool rewind (resource handle)
  - 将 *handle* 的文件位置指针设为文件流的开头。 如果成功则返回 TRUE，失败则返回 FALSE。 文件指针必须合法，并且指向由 fopen() 成功打开的文件。

3.6 文件的锁定机制

- flock -- 轻便的咨询文件锁定
  - 语法: bool flock (int handle, int operation [, int &wouldblock])
  - PHP 支持以咨询方式（也就是说所有访问程序必须使用同一方式锁定，否则它不会工作）锁定全部文件的一种轻便方法。
- *handle* 必须是一个已经打开的文件指针。
- *operation* 可以是以下值之一:
  - 要取得共享锁定（读取程序），将 *operation* 设为 LOCK_SH。
  - 要取得独占锁定（写入程序），将 *operation* 设为 LOCK_EX。
  - 要释放锁定（无论共享或独占），将 *operation* 设为 LOCK_UN。
  - 如果你不希望 flock() 在锁定时堵塞，则给 *operation* 加上 LOCK_NB。
- 如果成功则返回 TRUE，失败则返回 FALSE。

3.7 文件的一些基本操作函数

- copy -- 拷贝文件
  - 语法: bool copy (string source, string dest)
  - 将文件从 *source* 拷贝到 *dest*。如果成功则返回 TRUE，失败则返回 FALSE。
- unlink -- 删除文件
  - 语法: bool unlink (string filename)
  - 删除 *filename*。和 Unix C 的 unlink() 函数相似。如果成功则返回 TRUE，失败则返回 FALSE。
- ftruncate -- 将文件截断到给定的长度
  - 语法: bool ftruncate (resource handle, int size)
  - 接受文件指针 *handle* 作为参数，并将文件大小截取为 *size*。如果成功则返回 TRUE，失败则返回 FALSE。
- rename -- 重命名一个文件或目录
  - 语法: bool rename (string oldname, string newname [, resource context])
  - 尝试把 *oldname* 重命名为 *newname*。 如果成功则返回 TRUE，失败则返回 FALSE。

四、文件的上传与下载

- 4.1 文件上传
- 4.2 处理多个文件上传
- 4.3 文件下载

4.1 文件上传

- 在B/S程序中文件上传已经成为一个常用功能。其目的是客户可以通过浏览器(Browser)将文件上传到服务器（Server）上的指定目录。
- PHP中文件上传的基础知识
  - 表单
  - 对上传文件的操作

- HTML规范规定上传文件时表单头必须使用

```
<html>
<head><title>文件上传</title></head>
<body>
   <form action="todo.php"  method="post"
        enctype="multipart/form-data">
   <input type="hidden" name="MAX_FILE_SIZE"
   value="100000">
   上传文件: <input type="file" name="userfile">
   提交: <input type="submit" value="提交查询">
   </form>
</body>
</html>
```

- 注意几个特征属性:
- POST方法:
  - 表单最常用的功能,向目标页面传递变量,我们在上传文件的时候，会在表单中设置相应的属性，来完成文件的传递
- enctype="multipart/form-data"
  - 这样服务器就会知道，我们要传递一个文件,这样服务器可以知道上载的文件带有常规的表单信息。
- MAX_FILE_SIZE
  - 此字段必须在文件输入字段之前，控制最大的传递文件的大小(字节) ——真的可以控制吗?
- <input type="file" name="userfile">
  - 设置浏览器文件输入浏览按钮

php.ini文件上传参数设置

我们在服务器端的php.ini中设置对表单传递的数据进一步判断

- file_uploads = On /Off　　是否允许文件上传
- upload_max_filesize = 2M　上传的文件的最大大小
- post_max_size = 8M　　POST数据所允许的最大大小
- upload_tmp_dir　　上传文件放置的临时目录

- 表单传递的数据，文件只是其中的一部分，所以设置时，upload_max_filesize应该小于post_max_size

超级全局数组$_FILES

- PHP程序中，需要处理的上传数据保存在全局数组中$_FILES（超级全局数组）
- 保存$_FILES数组中的元素，将HTML表单的type="file"标记的名称name="userfile" 存放在数组中。
- 1：存储在$_FILES['userfile']['name']中的值是:
  - 客户端文件系统的文件的名称
- 2：存储在$_FILES['userfile']['type']中的值是:
  - 客户端传递的文件的类型

超级全局数组$_FILES

- 3：存储在$_FILES['userfile']['size']中的值是:
  - 文件的字节的大小
- 4：存储在$_FILES['userfile']['tmp_name']中的值
  - 文件被上传后在服务器存储的临时全路径
- 5：存储在$_FILES['userfile']['error']中的值是:
  - 文件上传的错误代码－php 4.2以后增加的功能

存储在$_FILES['userfile']['error']中的值

在$_FILES['userfile']['error']中返回的错误代码是在PHP4.2.0版本中引入的。具体如下:

- 值为0: 表示没有发生任何错误。
- 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。
- 值为2: 表示上传文件大小超出了HTML表单隐藏域属性的MAX_FILE_SIZE元素所指定的最大值。
- 值为3: 表示文件只被部分上传。
- 值为4: 表示没有上传任何文件。

错误值对应的常量

- UPLOAD_ERR_OK ： 对应值0
- UPLOAD_ERR_INI_SIZE ： 对应值1
- UPLOAD_ERR_FORM_SIZE ： 对应值2
- UPLOAD_ERR_PARTIAL ： 对应值3
- UPLOAD_ERR_NO_FILE ： 对应值4

数据格式（MIME）

文件类型	MIME类型
图片文件	image/gif，image/jpg，image/jpeg，image/png，image/x-png
纯文本和HTML	text/txt，text/plain，text/html
二进制文件	application/octet-stream
音频格式	audio/basic
视频格式	video/mpeg

- 文件上传后的临时存放目录

- 上传的文件被放置到服务器端临时目录：/tmp目录里面命名为一个唯一的，随机生成的临时文件名。
  注：该文件在程序执行完后将自动被删除掉。在删除前可以像本地文件一样操作。

- /tmp目录是默认的上传临时文件存放地点，

  如果需要更改这个目录：

  可以编辑/etc/php.ini 文件File Uploads 段的

  upload_tmp_dir 属性值。

上传后的文件处理

- 使用is_uploaded_file( )函数来检查此文件是否是上传文件。
- 应该使用move_uploaded_file(临时路径/临时文件名，目的路径/目的文件名) 函数将存放在临时目录下的上传文件拷贝出来，存放到指定目录的指定文件名，如果目标存在将会被覆盖。
- 当配置文件php.ini的register_globals属性被设置成on的情况
  - <input type=file name=myfilename>
  - 将生成全局变量: $myfilename等

文件上传后的处理页面

```
<html>
<head>  <title>上传文件...</title></head>
<body>  <h1>上传文件...</h1>
<?php
  if ($_FILES['userfile']['error'] > 0){
        echo '上传错误: ';
    switch ($_FILES['userfile']['error']){
      case 1:
        echo '上传文件大小超出了PHP配置文件中的约定值:
        upload_max_filesize';
        break;
      case 2:
        echo '上传文件大小超出了表单中的约定值:  max_file_size';
        break;
      case 3:
        echo '文件只被部分上载';
        break;
      case 4:
        echo '没有上传任何文件';
        break;
    }    exit;
  }
```

```
if ($_FILES['userfile']['type'] != 'text/plain'){
                echo '问题: 文件不是一个文本文件。';
        exit;
}
$upfile = './uploads/'.$_FILES['userfile']['name'];
 if (is_uploaded_file($_FILES['userfile']['tmp_name'])){
   //判断是否为上传文件
     if (!move_uploaded_file($_FILES['userfile']['tmp_name'], $upfile)){
        //移动文件
                echo '问题: 不能将文件移动到指定目录。';
        exit;
   }
 }else {
        echo '问题: 上传文件不是一个合法文件: ';
        echo $_FILES['userfile']['name'];
        exit;
 }
 echo '文件上传成功! <br><br>';
?>
```

4.2 处理多个文件上传

当需要上传多个文件的情况,有两种实现的解决方法:

使用不同的表单元素

```
<input type=file name=file_a>
<input type=file name=file_b>
```

使用数组格式的表单元素

```
<input type=file name=file[1]>
<input type=file name=file[2]>
```

4.3 文件下载

```
<?php
 //文件下载练习
$filename="./upload/aa.png";
$basename=pathinfo($filename);
header("Content-Type: image/png"); //指定下载文件类型的
header("Content-Disposition:attachment;filename=".$basename["basename"]);
//指定下载文件的描述信息
header("Content-Length:".filesize($filename));  //指定文件大小的
readfile($filename);//将内容输出，以便下载。
?>
```

五、文件系统项目实战

- 5.1 功能分析
- 5.2 系统功能设计
- 5.3 实例代码实现

谢谢！