File()函数：获取文件类型

实例： $fn2="/wamp";

echo '文件类型：'.filetype($fn2)."<br>";

file\_exists()函数：检查文件或目录是否存在。返回值为bool。

file\_get\_contents — 将整个文件读入一个字符串

file\_put\_contents — 将一个字符串写入文件

获取文件属性的函数

|  |  |
| --- | --- |
| Int filesize | 获取文件大小 |
| Int filectime | 获取文件的创建时间 |
| Int filemtime | 获取文件的修改时间 |
| Int fileatime | 获取文件的上次访问时间 |
| Bool is\_readable | 判断给定文件是否可读 |
| Bool is\_writable | 判断给定文件是否可写 |
| Bool is\_executable | 判断给定文件是否可执行 |
| Bool is\_file | 判断给定文件是否为一个正常的文件 |
| Bool is\_dir | 判断给定文件是否是一个目录 |
| Array stat | 给出文件的信息 |

$fn="../web/1.php"; //绝对路径C:\wamp\www\web/1.php

if(file\_exists($fn)&is\_file($fn))

{

echo '文件大小：'.filesize($fn).'字节<br>';

echo '文件的创建时间为'.date("Y-m-d H:i:s",filectime($fn))."<br>";

echo is\_readable($fn)?'该文件可读<br>':"该文件不可读<br>";

}

else

echo "not";

$fn="../web/1.php"; //绝对路径C:\wamp\www\web/1.php

if(file\_exists($fn)&is\_file($fn))

{

echo "<pre>";

print\_r(stat($fn));

echo "</pre>";

}

else

echo "not";

fopen()函数：打开文件

resource fopen( string $filename, string $mode[ ,bool $use\_include\_path=false [,resource $context] ] )

$filename表示指定打开的文件，$mode表示打开的模式，可选参数$use\_include\_path表示是否需要在include\_path中搜寻文件，可选参数$context表示上下文，用来设置一些其他的协议。

文件打开模式

|  |  |
| --- | --- |
| r | 读写方式打开，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0，如果文件不存在尝试创建之。 |
| 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0，如果文件不存在尝试创建之。 |
| 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，如果文件不存在尝试创建之。 |

fclose()函数：关闭文件

Bool fclose( resource $handle)

fread()函数：读取指定长度的字符串

string fread( resource $handle , int $length)

$fn="../1.txt"; //绝对路径C:\wamp\www\web/1.php

$handle1=fopen($fn,"r");

$content1=fread($handle1,9);

echo $content1."<br>";

fclose($handle1);

$handle2=fopen($fn,"r");

$content2=fread($handle2,filesize($fn));

echo $content2;

fclose($handle2);

feof：测试文件指针是否到达文件结束的位置

bool feof( resource $handle)

<?php

$fn=fopen("../1.txt","r");

while(!feof($fn)) //文件指针不指到末尾不停止循环

{

echo fgets($fn)."<br>";

}

fclose($fn);

?>

fgetc()函数：用于在打开的文件中读取一个字符。

string fgetc( resource $handle)

fgets()函数：用于在打开的文件中读取一行。

string fgetc( int $handle [, int $length])

$length指定了一行读取的字节数，碰到换行符、EOF或者已经读取了length-1字节后停止，如果没有指定$length，则默认为1024字节。

file\_get\_contents()函数：将文件的全部内容读取到一个字符串中。

string file\_get\_contents (string $filename[, bool $use\_include\_path[, resource $context[, int $offset[, int $maxlen ]]]])

$fn="../1.txt";

$handle1=file\_get\_contents($fn);

echo $handle1;

此函数是用来将文件内容读取到一个字符串中的首选方法。

file()函数：将整个文件读入到数组中，如果该函数执行成功，则返回一个数组，数组中的每个元素都是文件中的一行，包括换行符，失败返回false。

array file(string $filename[, int $use\_include\_path[, resource $context]])

$fielname指定了读取的文件名，也就是说该函数不用fopen()打开文件。

$lines=file("http://www.itcast.cn/");

foreach($lines as $line\_num => $line)

{

echo "Line #<b>{$line\_num}</b>:".htmlspecialchars($line)."<br>\n";

}

fseek() 函数：在打开的文件中定位。

该函数把文件指针从当前位置向前或向后移动到新的位置，新位置从文件头开始以字节数度量。成功则返回 0；否则返回 -1。注意，移动到 EOF 之后的位置不会产生错误。

fseek( file, offset, whence )

|  |  |
| --- | --- |
| file | 必需。规定要在其中定位的文件。 |
| offset | 必需。规定新的位置（从文件头开始以字节数度量）。 |
| whence | 可选。可能的值：   * SEEK\_SET - 设定位置等于 *offset* 字节。默认。 * SEEK\_CUR - 设定位置为当前位置加上 *offset*。 * SEEK\_END - 设定位置为文件末尾加上 *offset* （要移动到文件尾之前的位置，*offset* 必须是一个负值）。 |

$filename="../1.txt";

$content="似乎不关闭文件就不能进行读取?";

$handle=fopen($filename,"w+"); // 写+读，就填w+

fwrite($handle,$content);

fseek($handle, 0); // 文件指针要重新回到0，再读取内容。

$content2=fread($handle,filesize($filename));

echo "succes!!!";

echo $content2;

fwirte()函数：用于写入文件。

int fwrite( resource $handle, string $string[, int $length])

$length如果省略，则写入整个字符串

$filename="../1.txt";

$content="似乎不关闭文件就不能进行读取?";

$handle=fopen($filename,"r");

fwrite($handle,$content);

fclose($handle);

$handle2=fopen($filename,"a+");

$content2=fread($handle2,filesize($filename));

echo "succes!!!";

echo $content2;

似乎不关闭文件就不能进行读取？

这是因为你的指针指向了末尾当然读取不了了。

file\_put\_contents()函数：对文件进行写入操作，且不需要使用fopen()。

int file\_put\_contents( string $filename, mixed $data,[, int $flags[, resource $context ] ] )

知道有这个函数就好。

**文件加锁机制**

并发访问：指多个浏览器在同一时刻访问服务器上的某一个文件，这意味着不同的访问进程会在同一时刻读写同一文件，很有可能造成数据的紊乱或者文件的损坏，为了避免这个问题，php提供了文件加锁机制。

flock()函数：文件加锁

bool flock( resource $handle, int $operation,[, int, $wouldblock ])

$handle参数表示文件系统指针，$operation指定了使用哪种锁类型，

$wouldblock为可选，若设置为1或true，则当进行锁定时阻挡其他进程。

$operation参数

|  |  |
| --- | --- |
| LOCK\_SH: | 取得共享锁定(读文件时使用) |
| LOCK\_EX: | 取得独占锁定(写文件时使用) |
| LOCK\_UN: | 释放锁定(无论共享或独占) |
| LOCK\_NB: | 不希望flock()在锁定时堵塞，则加上 |

copy()函数：实现文件拷贝功能

bool copy (string $source, string $dest)

参数$source表示源文件，参数$dest表示指定目标文件。

rename()函数：实现文件或目录的重命名功能

bool rename(string $oldname, string $newname[, resource $context])

$oldname表示指定源文件的名称，$newname表示指定新的文件名称

注意：如果两个文件在同一目录下，则是重命名操作，在不同目录下，则为移动操作。

rename("zz.txt","dazz.txt"); //重命名操作

echo "success";

rename("lock.txt","1.txt.bak"); //移动操作

echo "rename and move success";

unlink()函数：删除文件

bool unlink ( string $filename )

getcwd：获得当前工作目录

**解析目录**

1.basename()函数：返回路径中的文件名。

string basename( string $path [,string $suffix ])

$suffix：文件名以$suffix结尾的，则返回结果中会被去掉这一部分字符。

<?php

$path="dazz.txt";

$file1=basename($path);

echo $file1."<br>";

$file2=basename($path,"txt");

echo $file2;

?>

2.dirname()函数：返回路径中的目录部分。

$path="C:\wamp\www\xiangmu\imgs";

$a=dirname($path);

echo $a; //返回C:\wamp\www\xiangmu

3.pathinfo()函数：以数组的形式返回路径的信息。

mixed pathinfo ( string $path [, int $options ])

$options指定返回哪些项，默认返回全部。

$path="C:\wamp\www\xiangmu\imgs/9.jpg";

echo "<pre>";

print\_r(pathinfo($path));

echo "</pre>";

**遍历目录**

1. opendir()函数：用于打开一个目录句柄

resource opendir( string $path [, resource $context])

1. readdir()函数：用于从目录句柄中读取条目

string readdir ( resource $dir\_handle)

参数$dir\_handle用于接受一个目录句柄的$resource,函数执行成功返回目录中下一个文件的文件名，否则返回false。

1. closedir()函数：关闭目录句柄

void closedir (resource $dir\_handle)

接受一个目录句柄的$resource,没有返回值。

1. rewinddir()函数：用于倒回目录句柄

void rewinddir (resource $dir\_handle)

指定由opendir()函数打开的目录句柄的$resource，执行该函数会将$dir\_handle指定的目录流重置到目录的开头，没有返回值。

遍历目录实例：

$path="C:\wamp\www\web";

$handle=opendir($path);

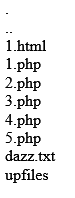
while(false!==($file=readdir($handle)))

{

echo "$file<br>";

}

closedir($handle);



“.”表示当前目录，“..”表示上一级目录。

**创建和删除目录**

1. mkdir()函数：用于新建目录

bool mkdir ( string $pathname [, int $mode [, bool $recursive [, resource $context ] ] ] )

$pathname指定要创建的目录，$mode为可选参数，指定目录的访问权限，默认值为0777。$recursive为可选参数，指定是否递归创建目录，默认值为false。$context为可选参数，指定上下文，通常忽略。

mkdir("path");

if(false!=mkdir("path1/path2",0777,true))

{

echo "sucess";

}

1. rmdir()函数：用于删除目录

bool rmdir ( string $dirname )

**1.文件上传表单**

表单提交方式必须为POST，

还需要添加上传的属性enctype=”multipart/form-data”,

说明浏览器可以提供文件上传功能，服务器端提交的数据中包含文件的数据。

<form enctype="multipart/form-data" action="3.php" method="post">

选择文件：<input type="file" name="userfile" />

<input type="submit" value="上传文件" name="buttom"/>

</form>

<?php

header('Content-Type:text/html;charset=utf-8');

$example=substr($\_FILES["userfile"]["name"],strrpos($\_FILES["userfile"]["name"],".")+1);

$mydir="upfiles/".md5(time()).".".$example;

move\_uploaded\_file($\_FILES["userfile"]["tmp\_name"],$mydir);

?>

统计目录中所有文件的大小

<?php

function getDirSize($dirname)

{

$dirsize=0;

$handle=opendir($dirname);

//从目录句柄中循环读取条目，并按照系统中的顺序返回目录中下一个文件的文件名

while($filename=readdir($handle))

{

if($filename!="." && $filename!="..") //如果不是上级目录和上一级目录

{

$file=$dirname.'/'.$filename;

if(is\_dir($file))

{

//如果是一个目录，则调用函数getDirSize()

$dirsize+=getDirSize($file);

}

else

{

//如果是一个文件，则取得其大小并且累加到变量$dirsize

$dirsize+=filesize($file);

}

}

}

closedir($handle); //关闭目录句柄

return $dirsize; //返回累加的大小

}

$dirname="F:/documents";

echo $dirname."目录中文件的总大小为：".getDirSize($dirname)."B";

?>