更多优质资源和原创文章在 西岭老湿 微信公众号

整理者：西岭； [redacted] 微博：@西岭i (http://weibo.com/u/3265712573) [redacted]

数学函数

函数名	描述	实例	输入	输出
abs()	求绝对值	$abs = abs(-4.2); //4.2	数字	绝对值数字
ceil()	进一法取整	echo ceil(9.999); // 10	浮点数	进一取整
floor()	舍去法取整	echo floor(9.999); // 9	浮点数	直接舍去小数部分
fmod()	浮点数取余	$x = 5.7;$y = 1.3; $r = fmod($x, $y); // $r equals 0.5, because 4 * 1.3 + 0.5 = 5.7	两个浮点数,x>y	浮点余数
pow()	返回数的n次方	echo pow(-1, 20); // 1	基础数|n次方	乘方值
round()	浮点数四舍五入	echo round(1.95583, 2);// 1.96	一个数值|保留小数点后多少位,默认为0	舍入后的结果
sqrt()	求平方根	echo sqrt(9); //3	被开方的数	平方根
max()	求最大值	echo max(1, 3, 5, 6, 7); // 7 echo max(array(2, 4, 5)); // 5	多个数字或数组	返回其中的最大值
min()	求最小值		多个数字或数组	返回其中的最小值
mt_rand()	更好的随机数	echo mt_rand(0,9);//n	最小|最大,随机数	随机返回范围内的值
rand()	随机数		最小|最大,随机数	随机返回范围内的值
pi()	获取圆周率值	echo pi(); // 3.1415926535898	无	获取圆周率

字符串函数

函数名	描述	实例	输入	输出
去空格或其他字符:				
trim()	删除字符串两端的空格或其他预定义字符	$str = "\r\nHello World!\r\n"; echo trim($str);	目标字串	清除后的字符串
rtrim()	删除字符串右边的空格或其他预定义字符	$str = "Hello World!\n\n"; echo rtrim($str);		
chop()	rtrim()的别名			
ltrim()	删除字符串左边的空格或其他预定义字符	$str = "\r\nHello World!"; echo ltrim($str);		
dirname()	返回路径中的目录部分	echo dirname("c:/testweb/home.php");	一个包含路径的字符串	返回文件路径的目录部分//c:/testweb
字符串生成与转化:				
str_pad()	把字符串填充为指定的长度	$str = "Hello World"; echo str_pad($str,20,"..");	要填充的字符串|新字符串的长度|供填充使用的字符串，默认是空白	完成后的字符串
str_repeat()	重复使用指定字符串	echo str_repeat(".",13);	要重复的字符串|字符串将被重复的次数	13个点
str_split()	把字符串分割到数组中	print_r(str_split("Hello"));	要分割的字符串|每个数组元素的长度，默认1	拆分后的字符串数组
strrev()	反转字符串	echo strrev("Hello World!");	目标字符串	颠倒顺序后的字符串!dlroW olleH
wordwrap()	按照指定长度对字符串进行折行处理	$str = "An example on a long word is: Supercalifragulistic"; echo wordwrap($str,15);	目标字符串|最大宽数	折行后的新字符串
str_shuffle()	随机地打乱字符串中所有字符	echo str_shuffle("Hello World");	目标字符串	顺序打乱后的字符串
parse_str()	将字符串解析成变量	parse_str("id=23&name=John%20Adams",$myArray); print_r($myArray);	要解析的字符串|存储变量的数组名称	返回Array([id] => 23 [name] => John Adams)
number_format()	通过千位分组来格式化数字		要格式化的数字|规定多少个小数|规定用作小数点的字符串|规定用作千位分隔符的字符串	1,000,000 1,000,000.00 1,000,000,00
大小写转换:				
strtolower()	字符串转为小写	echo strtolower("Hello WORLD!");	目标字符串	小写字符串
strtoupper()	字符串转为大写	echo strtoupper("Hello WORLD!");		大写字符串
ucfirst()	字符串首字母大写	echo ucfirst("hello world");		Hello world
ucwords()	字符串每个单词首字符转为大写	echo ucwords("hello world");		Hello World
html标签关联:				
htmlentities()	把字符转为HTML实体	$str = "John & 'Adams'"; echo htmlentities($str, ENT_COMPAT);		John & 'Adams'
htmlspecialchars()	预定义字符转html编码			
nl2br()	\n转义为 标签	echo nl2br("One line.\nAnother line.");		处理后的字符串
strip_tags()	剥去 HTML、XML 以及 PHP 的标签	echo strip_tags("Hello world!");		
addcslashes()	在指定的字符前添加反斜线转义字符串中字符	$str = "Hello, my name is John Adams."; echo $str; echo addcslashes($str,'m');	目标字符串|指定的特定字符或字符范围	
stripcslashes()	删除由addcslashes()添加的反斜线	echo stripcslashes("Hello, \my na\me is Kai Ji\m.");	目标字符串	Hello, my name is Kai Jim.
addslashes()	指定预定义字符前添加反斜线	$str = "Who's John Adams?";echo addslashes($str);		把目标串中的' " \和null进行转义处理
stripslashes()	删除由addslashes()添加的转义字符	echo stripslashes("Who\'s John Adams?");		清除转义符号Who's John Adams?
quotemeta()	在字符串中某些预定义的字符前添加反斜线	$str = "Hello world. (can you hear me?)"; echo quotemeta($str);		Hello world\. \(can you hear me\?\)
chr()	从指定的 ASCII 值返回字符	echo chr(052);	ASCII 值	返回对应的字符//*
ord()	返回字符串第一个字符的 ASCII 值	echo ord("hello");	字符串	第一个字符的 ASCII 值
字符串比较:				
strcasecmp()	不区分大小写比较两字符串	echo strcasecmp("Hello world!","HELLO WORLD!");	两个目标字符串	大1|等0|小-1
strcmp()	区分大小写比较两字符串			
strncmp()	比较字符串前n个字符,区分大小写	int strncmp (string $str1 , string $str2 , int $len)		
strncasecmp()	比较字符串前n个字符,不区分大小写	int strncasecmp (string $str1 , string $str2 , int $len)		
strnatcmp()	自然顺序法比较字符串长度,区分大小写	int strnatcmp (string $str1 , string $str2)	目标字符串	
strnatcasecmp()	自然顺序法比较字符串长度,不区分大小写	int strnatcasecmp (string $str1 , string $str2)		
字符串切割与拼接:				
chunk_split()	将字符串分成小块	str chunk_split(str $body[,int $len[,str $end]])	$body目标字串,$len长度,$str插入结束符	分割后的字符串
strtok()	切开字符串	str strtok(str $str,str $token)	目标字符串$str，以$token为标志切割	返回切割后的字符串
explode()	使用一个字符串为标志分割另一个字符串	array explode(str $sep,str $str[,int $limit])	$sep为分割符,$str目标字符串,$limit返回数组最多包含元素数	字符串被分割后形成的数组
implode()	同join,将数组值用预订字符连接成字符串	string implode (string $glue , array $pieces)	$glue默认，用''则直接相连	
substr()	截取字符串	string substr (string $string , int $start [, int $length])		
字符串查找替换:				
str_replace()	字符串替换操作,区分大小写	mix str_replace(mix $search,,mix $replace,mix $subject[,int &$num])	$search查找的字符串,$replace替换的字符串,$subject被查找字串,&$num	返回替换后的结果
str_ireplace()	字符串替换操作,不区分大小写	mix str_ireplace (mix $search , mix $replace , mix $subject [, int &$count])	$search查找的字符串,$replace替换的字符串,$subject被查找字串,&$num	返回替换后的结果
substr_count()	统计一个字符串,在另一个字符串中出现次数	int substr_count (string $haystack , string $needle [, int $offset = 0 [, int $length]])		
		mixed substr_replace (mixed $string , string		

substr_replace()	替换字符串中某串为另一个字符串	$replacement , int $start [, int $length])		
similar_text()	返回两字符串相同字符的数量	int similar_text(str $str1,str $str2)	两个比较的字符串	整形,相同字符数量
strrchr()	返回一个字符串在另一个字符串中最后一次出现位置开始到末尾的字符串	string strrchr (string $haystack , mixed $needle)		
strstr()	返回一个字符串在另一个字符串中开始位置到结束的字符串	string strstr (string $str, string $needle , bool $before_needle)		
strchr()	strstr()的别名,返回一个字符串在另一个字符串中首次出现的位置开始到末尾的字符串	string strstr (string $haystack , mixed $needle [, bool $before_needle = false])		
stristr()	返回一个字符串在另一个字符串中开始位置到结束的字符串，不区分大小写	string stristr (string $haystack , mixed $needle [, bool $before_needle = false])		
strtr()	转换字符串中的某些字符	string strtr (string $str , string $from , string $to)		
strpos()	寻找字符串中某字符最先出现的位置	int strpos (string $haystack , mixed $needle [, int $offset = 0])		
stripos()	寻找字符串中某字符最先出现的位置,不区分大小写	int stripos (string $haystack , string $needle [, int $offset])		
strrpos()	寻找某字符串中某字符最后出现的位置	int strrpos (string $haystack , string $needle [, int $offset = 0])		
strripos()	寻找某字符串中某字符最后出现的位置,不区分大小写	int strripos (string $haystack , string $needle [, int $offset])		
strspn()	返回字符串中首次符合mask的子字符串长度	int strspn (string $str1 , string $str2 [, int $start [, int $length]])		
strcspn()	返回字符串中不符合mask的字符串的长度	int strcspn (string $str1 , string $str2 [, int $start [, int $length]])	$str1被查询，$str2查询字符串，$start开始查询的字符，$length查询长度	返回从开始到第几个字符
字符串统计:				
str_word_count()	统计字符串含有的单词数	mix str_word_count(str $str,[])	目标字符串	统计处的数量
strlen()	统计字符串长度	int strlen(str $str)	目标字符串	整型长度
count_chars()	统计字符串中所有字母出现次数(0..255)	mixed count_chars (string $string [, int $mode])		
字符串编码:				
md5()	字符串md5编码	$str = "Hello"; echo md5($str);		

数组函数

函数名	描述	实例	输入	输出
数组创建:				
array()	生成一个数组	$a=array("Dog","Cat","Horse"); print_r($a);	数组值或,键=>值	一个数组型变量
array_combine()	生成一个数组,用一个数组的值作为键名,另一个数组值作为值	$a1=array("a","b","c","d"); $a2=array("Cat","Dog","Horse","Cow"); print_r(array_combine($a1,$a2));	$a1为提供键,$a2提供值	合成后的数组
range()	创建并返回一个包含指定范围的元素的数组。	$number = range(0,50,10); print_r ($number);	0是最小值,50是最大值,10是步长	合成后的数组
compact()	创建一个由参数所带变量组成的数组	$firstname = "Peter"; $lastname = "Griffin"; $age = "38"; $result = compact("firstname", "lastname", "age"); print_r($result);	变量或数组	返回由变量名为键,变量值为值的数组,变量以为多维数组.会递归处理
array_fill()	用给定的填充(值生成)数组	$a=array_fill(2,3,"Dog"); print_r($a);	2是键,3是填充的数量,'Dog'为填充内容	返回完成的数组
数组合并和拆分:				
array_chunk()	把一个数组分割为新的数组块	$a=array("a"=>"Cat","b"=>"Dog","c"=>"Horse","d"=>"Cow"); print_r(array_chunk($a,2));	一个数组	分割后的多维数组，规定每个新数组包含2个
array_merge()	把两个或多个数组合并为一个数组。	$a1=array("a"=>"Horse","b"=>"Dog"); $a2=array("c"=>"Cow","b"=>"Cat"); print_r(array_merge($a1,$a2));	两个数组	返回完成后的数组
array_slice()	在数组中根据条件取出一段值，并返回。	$a=array(0=>"Dog",1=>"Cat",2=>"Horse",3=>"Bird"); print_r(array_slice($a,1,2));	一个数组	1为从'Cat'开始，2为返回两个元素
数组比较:				
array_diff()	返回两个数组的差集数组	$a1=array(0=>"Cat",1=>"Dog",2=>"Horse"); $a2=array(3=>"Horse",4=>"Dog",5=>"Fish"); print_r(array_diff($a1,$a2));	两个或多个数组	返回'Cat'，$a1与$a2的不同之处
array_intersect()	返回两个或多个数组的交集数组			返回'Dog'和'Horse',$a1与$a2的相同之处
数组查找替换:				
array_search()	在数组中查找一个值，返回一个键，没有返回返回假	$a=array("a"=>"Dog","b"=>"Cat","c"=>"Horse"); echo array_search("Dog",$a);	一个数组	成功返回键名,失败返回false

array_splice()	把数组中一部分删除用其他值替代	$a1=array(0=>"Dog",1=>"Cat",2=>"Horse",3=>"Bird");
		$a2=array(0=>"Tiger",1=>"Lion");
		array_splice($a1,0,2,$a2);
		print_r($a1);
			一个或多个数组	$a1被移除的部分由$a2补全

array_sum()	返回数组中所有值的总和	$a=array(0=>"5",1=>"15",2=>"25"); echo array_sum($a);	一个数组	返回和

in_array()	在数组中搜索给定的值,区分大小写	$people = array("Peter", "Joe", "Glenn", "Cleveland");
		if (in_array("Glenn",$people){
		echo "Match found";}else{
		echo "Match not found";}
			需要搜索的值|数组	true/false

array_key_exists()	判断某个数组中是否存在指定的 key		需要搜索的键名|数组	
数组指针操作:				
key()	返回数组内部指针当前指向元素的键名			
current()	返回数组中的当前元素（单元）。			
next()	把指向当前元素的指针移动到下一个元素的位置，并返回当前元素的值			
prev()	把指向当前元素的指针移动到上一个元素的位置，并返回当前元素的值			
end()	将数组内部指针指向最后一个元素，并返回该元素的值（如果成功）			
reset()	把数组的内部指针指向第一个元素，并返回这个元素的值			
list()	用数组中的元素为一组变量赋值	$my_array=array("Dog","Cat","Horse"); list($a, $b, $c) = $my_array;	$a, $b, $c为需要赋值的变量	变量分别匹配数组中的值
array_shift()	删除数组中的第一个元素，并返回被删除元素的值	$a=array("a"=>"Dog","b"=>"Cat","c"=>"Horse"); echo array_shift($a); print_r ($a);		

array_unshift()	在数组开头插入一个或多个元素	$a=array("a"=>"Cat","b"=>"Dog");
		array_unshift($a,"Horse");
		print_r($a);		

array_push()	向数组最后压入一个或多个元素	$a=array("Dog","Cat"); array_push($a,"Horse","Bird");	目标数组|需要压入的值	返回新的数组

		print_r($a);		

array_pop()	取得（删除）数组中的最后一个元素	$a=array("Dog","Cat","Horse");
		array_pop($a);
		print_r($a);
			$a为目标数组	返回数组剩余元素

数组键值操作：				

shuffle()	将数组打乱,保留键名	$my_array = array("a" => "Dog", "b" => "Cat");
		shuffle($my_array);
		print_r($my_array);
			一个或多个数组	顺序打乱后的数组

count()	计算数组中的单元数目或对象中的属性个数	$people = array("Peter", "Joe", "Glenn", "Cleveland"); $result = count($people); echo $result;	数组	输出元素个数
array_flip()	返回一个键值反转后的数组	$a=array(0=>"Dog",1=>"Cat",2=>"Horse");print_r(array_flip($a));		返回完成后的数组
array_keys()	返回数组所有的键,组成一个数组	$a=array("a"=>"Horse","b"=>"Cat","c"=>"Dog"); print_r(array_keys($a));		返回由键名组成的数组
array_values()	返回数组中所有值，组成一个数组	同上		返回由键值组成的数组
array_reverse()	返回一个元素顺序相反的数组	同上		元素顺序相反的一个数组，键名和键值依然
array_count_values()	统计数组中所有的值出现的次数	$a=array("Cat","Dog","Horse","Dog"); print_r(array_count_values($a));	返回数组	原键值为新键名，次数为新键值
array_rand()	从数组中随机抽取一个或多个元素,注意是键名!!!	$a=array("a"=>"Dog","b"=>"Cat","c"=>"Horse"); print_r(array_rand($a,1));	$a为目标数组，1为抽取第几个元素的键名	返回第1个元素的键名b
each()				
array_unique()	删除重复值，返回剩余数组	$a=array("a"=>"Cat","b"=>"Dog","c"=>"Cat"); print_r(array_unique($a));	数组	返回无重复值数组，键名不变
数组排序：				

sort()	按升序对给定数组的值排序,不保留键名	$my_array = array("a" => "Dog", "b" => "Cat", "c" => "Horse");
		sort($my_array);
		print_r($my_array);
				true/false

rsort()	对数组逆向排序,不保留键名			
asort()	对数组排序,保持索引关系			
arsort()	对数组逆向排序,保持索引关系			
ksort()	按键名对数组排序			
krsort()	将数组按照键逆向排序			
natsort()	用自然顺序算法对数组中的元素排序			
natcasesort()	自然排序,不区分大小写			

文件系统函数

函数名	描述	实例	输入	输出
fopen()	打开文件或者 URL	$handle = fopen("ftp://user:password@example.com/somefile.txt", "w");	resource fopen (string filename, string mode [, bool use_include_path [, resource zcontext]])	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	$handle = fopen('somefile.txt', 'r'); fclose($handle);	bool fclose(resource handle)	如果成功则返回 TRUE，失败则返回 FALSE
文件属性				

file_exists()	检查文件或目录是否存在	$filename = '/path/to/foo.txt';
		if (file_exists($filename)) {
		echo "exists";
		} else {
		echo "does not exist";
		}
			bool file_exists (string filename)	指定的文件或目录存在则返回 TRUE，否则返回 I

filesize()	取得文件大小	$filename = 'somefile.txt'; echo $filename . ': ' . filesize($filename) . 'bytes';	int filesize (string $filename)	返回文件大小的字节数，如果出错返回 FALSE 并一条 E_WARNING 级的错误

is_readable()	判断给定文件是否可读	$filename = 'test.txt';
		if (is_readable($filename)) {
		echo '可读';
		} else {
		echo '不可读';
		}
			bool is_readable (string $filename)	如果由 filename 指定的文件或目录存在并且可返回 TRUE

is_writable()	判断给定文件是否可写	$filename = 'test.txt';
		if (is_writable($filename)) {
		echo '可写';
		} else {
		echo '不可写';
		}
			bool is_writable (string $filename)	如果文件存在并且可写则返回 TRUE。filename 可以是一个允许进行是否可写检查的目录名

is_executable()	判断给定文件是否可执行	$file = 'setup.exe';
		if (is_executable($file)) {
		echo '可执行';
		} else {
		echo '不可执行';
		}
			bool is_executable (string $filename)	如果文件存在且可执行则返回 TRUE

filectime()	获取文件的创建时间	$filename = 'somefile.txt'; echo filectime($filename);	int filectime (string $filename)	时间以 Unix 时间戳的方式返回，如果出错则返 FALSE
filemtime()	获取文件的修改时间	$filename = 'somefile.txt'; echo filemtime($filename);	int filemtime (string $filename)	返回文件上次被修改的时间，出错时返回 FALSE 间以 Unix时间戳的方式返回
fileatime()	获取文件的上次访问时间	$filename = 'somefile.txt'; echo fileatime($filename);	int fileatime (string $filename)	返回文件上次被访问的时间，如果出错则返回 FALSE。时间以Unix时间戳的方式返回
stat()	获取文件大部分属性值	$filename = 'somefile.txt'; var_dump(fileatime($filename));	array stat (string $filename)	返回由 filename 指定的文件的统计信息
文件操作				

fwrite()	写入文件	$filename = 'test.txt';
		$somecontent = "添加这些文字到文件\n";
		$handle = fopen($filename, 'a');
		fwrite($handle, $somecontent);
		fclose($handle);
			int fwrite (resource handle, string string [, int length])	把 string 的内容写入 文件指针 handle 处。指定了 length ，当写入了 length 个字节或者了 string 以后，写入就会停止，视乎先碰到哪况

fputs()	同上			
fread()	读取文件	$filename = "/usr/local/something.txt"; $handle = fopen($filename, "r"); $contents = fread($handle, filesize ($filename)); fclose($handle);	string fread (int handle, int length)从文件指针 handle，读取最多 length 个字节	从文件指针 handle 读取最多 length 个字

feof()	检测文件指针是否到了文件结束的位置	$file = @fopen("no_such_file", "r");
		while (!feof($file)) {
		}
		fclose($file);
			bool feof (resource handle)	如果文件指针到了 EOF 或者出错时则返回 TRUE 则返回一个错误（包括 socket 超时），其它情 返回 FALSE

fgets()	从文件指针中读取一行	$handle = @fopen("/tmp/inputfile.txt", "r");
		if ($handle) {
		while (!feof($handle)) {
		$buffer = fgets($handle, 4096);
		echo $buffer;
		}
		fclose($handle);
		}
			string fgets (int handle [, int length])	从 handle 指向的文件中读取一行并返回长度最 length - 1 字节的字符串。碰到换行符（包括在值中）、EOF 或者已经读取了 length - 1 字节止（看先碰到那一种情况）。如果没有指定 leng 则默认为 1K，或者说 1024 字节。

fgetc()	从文件指针中读取字符	$fp = fopen('somefile.txt', 'r');
		if (!$fp) {
		echo 'Could not open file somefile.txt';
		}
		while (false !== ($char = fgetc($fp))) {
		echo "$char\n";
		}
			string fgetc (resource $handle)	返回一个包含有一个字符的字符串，该字符从 h 指向的文件中得到。碰到 EOF 则返回 FALS

file()	把整个文件读入一个数组中	$lines = file('http://www.example.com/');
		// 在数组中循环，显示 HTML 的源文件并加上行号。
		foreach ($lines as $line_num => $line) {
		echo "Line #{$line_num} : " .
		htmlspecialchars($line) . "
		\n";
		}
		// 另一个例子将 web 页面读入字符串。参见
		file_get_contents()。
		$html = implode('', file
		('http://www.example.com/'));
			array file (string $filename [, int
			$use_include_path [, resource $context]])
				数组中的每个单元都是文件中相应的一行，包括
				符在内。如果失败 file() 返回 FALSE

readfile()	输出一个文件		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 的字节数。如果出错返回 FALSE
file_get_contents()	将整个文件读入一个字符串	echo file_get_contents('http://www.baidu.com');	string file_get_contents (string $filename [, bool $use_include_path [, resource $context [, int $offset [, int $maxlen]]]])	
file_put_contents()	将一个字符串写入文件	file_put_contents('1.txt','aa');	int file_put_contents (string $filename , string $data [, int $flags [, resource $context]])	该函数将返回写入到文件内数据的字节数

ftell()	返回文件指针读/写的位置	$fp=fopen('tx.txt','r');
		fseek($fp,10);
		echo ftell($fp);
		fread($fp,4);
		echo ftell($fp);
			int ftell (resource $handle)	返回由 handle 指定的文件指针的位置，也就是
				流中的偏移量

fseek()	在文件指针中定位	$fp=fopen('tx.txt','r');
		fseek($fp,10);
		echo ftell($fp);
		fread($fp,4);
		echo ftell($fp);
			int fseek (resource $handle , int $offset [, int
			$whence])
				成功则返回 0；否则返回 -1

rewind()	倒回文件指针的位置	$fp=fopen('tx.txt','r');
		fseek($fp,3);
		echo ftell($fp);
		fread($fp,4);
		rewind($fp);
		echo ftell($fp);
			bool rewind (resource $handle)	如果成功则返回 TRUE，失败则返回 FALSE

flock()	轻便的咨询文件锁定	$fp=fopen('tx.txt','r');
		flock($fp, LOCK_SH);//共享锁
		//flock($fp, LOCK_EX);//独立锁，写文件时用它打开
		//flock($fp, LOCK_NB);//附加锁
		flock($fp, LOCK_UN);//释放锁
		fclose($fp);
			bool flock (int $handle , int $operation [, int
			&$wouldblock])
				如果成功则返回 TRUE，失败则返回 FALSE

目录

basename()	返回路径中的文件名部分	path = "/home/httpd/html/index.php"; $file = basename($path); $file = basename($path,".php");	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 函数返回基本的文件名。如果文件名是以 suffi 束的，那这一部分也会被去掉
dirname()	返回路径中的目录部分	$path = "/etc/passwd"; $file = dirname($path);	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 函数返回去掉文件名后的目录名

pathinfo()	返回文件路径的信息	echo '<pre>';
		print_r(pathinfo("/www/htdocs/index.html"));
		echo '</pre>';
			mixed pathinfo (string $path [, int $options])	返回一个关联数组包含有 path 的信息

opendir()	打开目录句柄	$fp=opendir('E:/xampp/htdocs/php/study/19');
		echo readdir($fp);
		closedir($fp);
			resource opendir (string $path [, resource
			$context])
				如果成功则返回目录句柄的 resource，失败则
				FALSE

readdir()	从目录句柄中读取条目	$fp=opendir('E:/xampp/htdocs/php/study/19');
		echo readdir($fp);
		closedir($fp);
			string readdir (resource $dir_handle)	返回目录中下一个文件的文件名。文件名以在文
				统中的排序返回

closedir()	关闭目录句柄	$fp=opendir('E:/xampp/htdocs/php/study/19');
		echo readdir($fp);
		closedir($fp);
			void closedir (resource $dir_handle)	关闭由 dir_handle 指定的目录流。流必须之前
				opendir() 所打开

rewinddir()	倒回目录句柄	$fp=opendir('E:/xampp/htdocs/php/study/19');
		echo readdir($fp).'
		';
		echo readdir($fp).'
		';
		echo readdir($fp).'
		';
		rewinddir($fp);
		echo readdir($fp).'
		';
		closedir($fp);
			void rewinddir (resource $dir_handle)	将 dir_handle 指定的目录流重置到目录的开

mkdir()	新建目录	mkdir('123');	bool mkdir (string $pathname [, int $mode [, bool $recursive [, resource $context]]])	尝试新建一个由 pathname 指定的目录
rmdir()	删除目录	rmdir('123');	bool rmdir (string $dirname)	尝试删除 dirname 所指定的目录。 该目录必须 的，而且要有相应的权限。如果成功则返回 TRU 败则返回 FALSE
unlink()	删除文件	unlink('123/1.txt'); rmdir('123');	bool unlink (string $filename)	删除 filename 。和 Unix C 的 unlink() 函数 似。如果成功则返回 TRUE，失败则返回 FALS
copy()	拷贝文件	copy('index.php','index.php.bak');	bool copy (string $source , string $dest)	将文件从 source 拷贝到 dest 。如果成功则 TRUE，失败则返回 FALSE
rename()	重命名一个文件或目录	rename('tx.txt','txt.txt');	bool rename (string $oldname , string $newname [, resource $context])	如果成功则返回 TRUE，失败则返回 FALSE

文件的上传与下载

is_uploaded_file()	判断文件是否是通过 HTTP POST 上传的	if(is_uploaded_file($_FILES['bus']['tmp_name'])){
		if(move_uploaded_file($_FILES['bus']['tmp_name'],
		$NewPath)){
		echo '上传成功
		<img src="'.$NewPath.'">';
		}else{
		exit('失败');
		}
		}else{
		exit('不是上传文件');
		}
			bool is_uploaded_file (string $filename)	

move_uploaded_file()	将上传的文件移动到新位置	if(is_uploaded_file($_FILES['bus']['tmp_name'])){
		if(move_uploaded_file($_FILES['bus']['tmp_name'],
		$NewPath)){
		echo '上传成功
		<img src="'.$NewPath.'">';
		}else{
		exit('失败');
		}
		}else{
		exit('不是上传文件');
		}
			bool move_uploaded_file (string $filename , string
			$destination)	

时间函数

函数名	描述	实例	输入	输出
time()	返回当前的 Unix 时间戳	time();	int time (void)	返回自从 Unix 纪元（格林威治时间 1970 年 1 日 00:00:00）到当前时间的秒数
mktime()	取得一个日期的 Unix 时间戳	mktime(0, 0, 0, 4, 25, 2012);	int mktime ([int $hour [, int $minute [, int $second [, int $month [, int $day [, int $year [, int $is_dst]]]]]])	
date()	格式化一个本地时间 / 日期	date('Y年m月d日 H:i:s');	string date (string $format [, int $timestamp])	2012年04月25日 20:45:54

checkdate()	验证一个格里高里日期	if(checkdate(6,31,2012)){
		echo '成立';
		}else{
		echo '不成立';
		}
			bool checkdate (int $month , int $day , int $year
			)
				如果给出的日期有效则返回 TRUE，否则返回 FA

date_default_timezone_set()	设定用于一个脚本中所有日期时间 函数的默认时区	date_default_timezone_set('PRC');	bool date_default_timezone_set (string $timezone_identifier)	
		$t=getdate();		返回一个根据 timestamp 得出的包含有日期信息

getdate()	取得日期 / 时间信息	var_dump($t);	array getdate ([int $timestamp])	合数组。如果没有给出时间戳则认为是当前本地

strtotime()	将任何英文文本的日期时间描述解析为 Unix 时间戳	echo strtotime("now");
		echo strtotime("10 September 2000");
		echo strtotime("+1 day");
		echo strtotime("+1 week");
		echo strtotime("+1 week 2 days 4 hours 2 seconds");
		echo strtotime("next Thursday");
		echo strtotime("last Monday");
			int strtotime (string $time [, int $now])	

microtime()	返回当前 Unix 时间戳和微秒数	$start=microtime(true); sleep(3); $stop=microtime(true); echo $stop-$start;	mixed microtime ([bool $get_as_float])	

正则表达式-元字符

元字符	含义	等价于
匹配范围		
\d	匹配任意一个十进制数字	[0-9]
\D	匹配除十进制数字以外的任意数字	[^0-9]
\s	匹配空白字符	[\n\f\r\t\v]
\S	匹配除空白字符以外的任意一个字符	[^\n\f\r\t\v]
\w	匹配任意一个数字、字母和下划线	[0-9a-zA-Z_]
\W	匹配除字母、数字和下划线以外的任意字符	[^0-9a-zA-Z_]
[]	1)用来表示范围。 2)匹配任意一个中括号中定义的原子	
[^]	中括号里面的^(抑扬符)：表示匹配任意一个除中括号里面定义的原子	
限定次数		
*	匹配0次、1次或多次其前的原子	{0,}
+	匹配1次或多次其前的原子	{1,}
?	匹配0次或1次其前的原子	{0,1}
{n}	表示其前的原子正好出现n次	
{n,}	表示其前的原子至少出现n次，最多不限制	
{m,n}	表示其前的原子最少出现m次，最多出现n次	
其它		
.	匹配除换行符(\n)以外的任意字符【windows下还匹配\f\r】	
|	两个或多个分支选择【优先级最低】	
^	匹配输入字符的开始位置	
$	匹配输入字符的结束位置	
\b	匹配词边界	
\B	匹配非词边界	
()	1)模式单元，把多个小原子组成一个大原子。2)可以改变优先级	