变量：

内存中用于临时存储数据的一个空间，空间有一个名子

变量都是以$开头

预定义变量：

$\_GET

$\_POST

$\_RQUEST

$\_SERVER

$\_SESSION

$\_COOKIE

$\_FILES

$GLOBAL

可变变量

$$变量名;

$v = ‘age’;

$age = 20;

echo $$v

常量

define(‘常量名’,值【,true|false】);

const 常量名=值;

魔术常量

\_\_FILE\_\_

\_\_LINE\_\_

\_\_DIR\_\_

\_\_FUNCTION\_\_

\_\_CLASS\_\_

\_\_METHOD\_\_

\_\_NAMESPACE\_\_

数据类型

标量类型

int

float

string

boolean

复合类型

array

object

特殊类型

null

resource 资源类型

也是一种特殊的变量，资源类型的不能让程序员直接定义。必须使用php为我们提供的资源操作函数来获取。

类型转换

自动转换，当参与运算的两个操作数数据类型不一致的时候，会根据运算符自动进行转换。

强制转换，其他数据类型转换为布尔型。

(boolean)变量

0、0.0、null、’’、’0’、空数组 等效于布尔false的值

类型判断

is\_类型名()

is\_string()

is\_int()

is\_float()

is\_bool()

is\_scalar()

is\_array()

is\_object()

is\_null()

isset(); 判断一个变量的值是否为null，如果是Null返回false

empty() 判断一个变量的值是否为空，

一、进制

1、相关概念

数码：某一种进制中所出现的数字

基数：某一种进制中数码的个数(逢几进一)

位权：用于衡量某一种进制中数码所在的位置的大小的值。

2、计算机常用进制

①、十进制

数码：0 至 9

基数：10

位权：

(12345)10

按权展开式：1\*104 + 2\*103 + 3\*102 + 4\*101 + 5\*100

10000 1000 100 10 1

②、八进制

数码：0 至 7

基数：8

位权：

(12345)8

按权展开式：1\*84 + 2\*83 + 3\*82 + 4\*81 + 5\*80

4096 512 64 8 1

③、十六进制

数码：0 至 9 a 至 f a(十进制中的10) f(十进制中的15)

基数：16

位权：

(12345)16

按权展开式：1\*164 + 2\*163 + 3\*162 + 4\*161 + 5\*160

256 16 1

④、二进制

数码：0、1

基数：2

位权：

(11111)2

按权展开式：1\*24 + 1\*23 + 1\*22 + 1\*21 + 1\*20

16 8 4 2 1

2、进制转换（了解）

任何一种进制按权展开最终的结果都是十进制的值。

所谓的按权展开就是，某一种进制中，各个位上的数码与当前位的位权的乘积之和。就是按权那个

①、二进制转换为十进制：

8421BCD码

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 位 | 位7 | 位6 | 位5 | 位4 | 位3 | 位2 | 位1 | 位0 |
| 权 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 二进制 |  |  | 1 | 0 | 1 | 1 | 0 | 1 |

二进制转换为十进制就是将1位对应的权相加即可得到对应的十进制数。

二、运算符

1、赋值运算符

=

运算规则：

将赋值号右边的值赋值给左边的变量。左边必须是变量不能是式子。

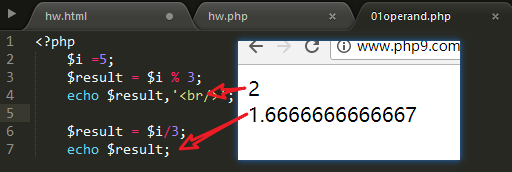


2、算术运算符

+、-、\*、/、%

% 取模(取余)

运算规则：求两个数相除，除不尽的部分。



在实际开发中，我们经常使用%运算判断奇偶数。

3、自操作运算符

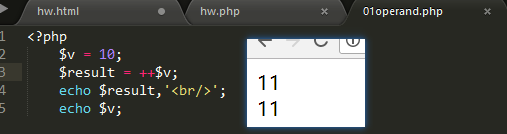
++ 自增、--自减

自增运算符

前自增 规则：先对变量的值自增1，再使用变量的新值参与式子的运算。

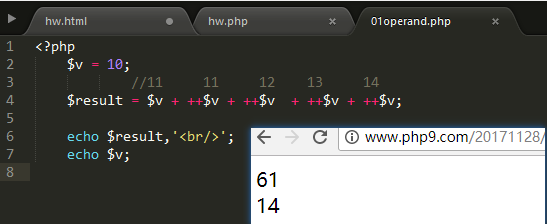
后自增 规则：先使用变量的原值参与式子的运算，再对变量的值进行加1

示例：前自增



示例：后自增





说明：

前++是先更改内存中变量的值，再将新值读取到这个式子中。

后++是先取内存中的这个变量的值，放在式子中，再对变量的值加1

自减运算符

前自减 规则：先对变量的值自减1，再使用变量的新值参与式子的运算。

后自减 规则：先使用变量的原值参与式子的运算，再对变量的值进行减1

$i+=2 等效于 $i=$i+2;

$i-=2 等效于 $i=$i-2;

$i\*=2 等效于 $i=$i\*2;

$i/=2 等效于 $i=$i/2;

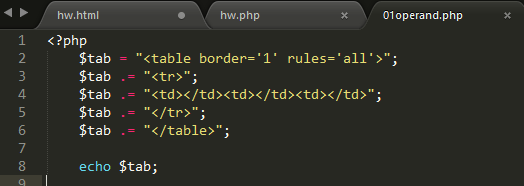
$i%=2 等效于 $i=$i%2;

4、字符串运算符

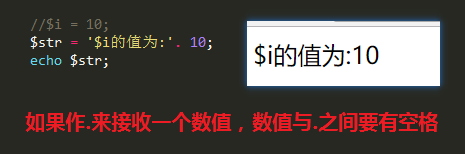
**. 对字符串进行拼接**

**.= 对字符串进行拼接**

示例：



注意：



5、比较运算符

>、<、>=、<=、==、!=、===、!==

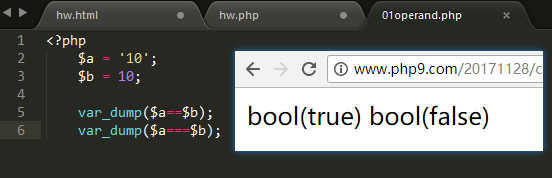
前提：

与变量相关的有两部分：其一是值，其二是类型。

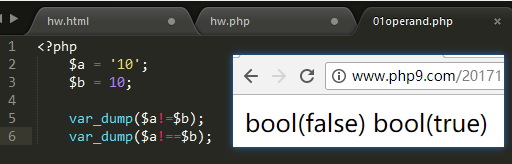
== 用于判断两个数的值是否相等

=== 同时判断变量的值与类型是否相同，如果相同返回true，

示例：



!== 判断两个变量的值或类型至少有一方面不相同



提示：

比较运算符得到的最终结果boolean类型的值

6、逻辑运算符

&&、||、!、and、or

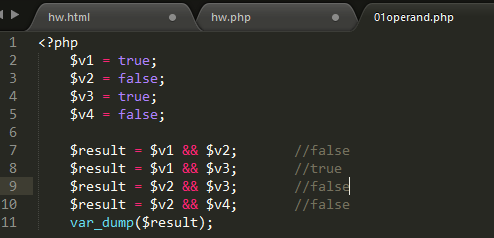
&& 逻辑与

运算规则：

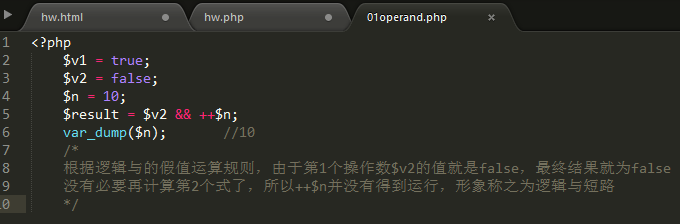
真值运算规则：参与运算的两个操作数，如果同时为true或能转换为true，那么最终的结果就是true

假值运算规则：参与运算的两个操作数，只要有一个为false或能转换为false，那么最终的结果就是false

示例：



**逻辑与短路：**



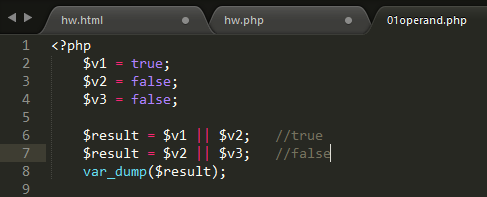
|| 逻辑或

运算规则：

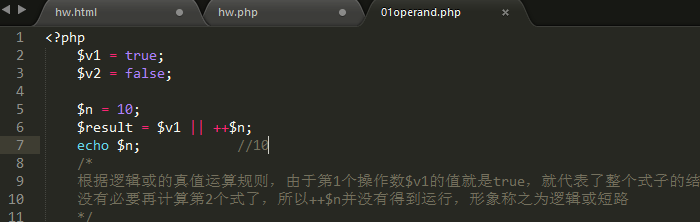
真值运算规则：参与运算的两个操作数，只要有一个为true或能转换为true，那么最终的结果就为true

假值运算规则：参与运算的两个操作数，全部为false或能够转换为false，那么最终的结果才是false

示例：

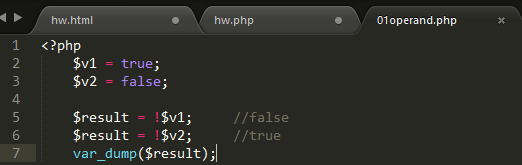


逻辑或短路：



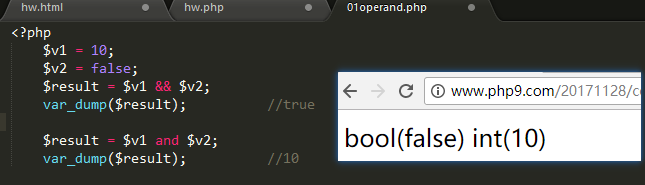
! 逻辑非

运算规则：对操作数据进行取反。



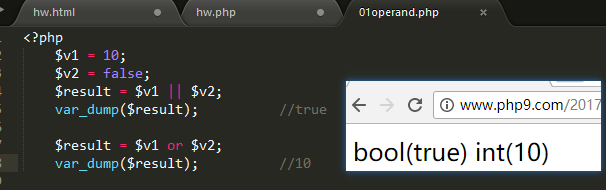
and 逻辑与

运算规则与&&运算规则相同，唯一区别是&&的优先级高于=，而and的优先级是低于=



or 逻辑或

运算规则与||运算规则相同，唯一的区别是||的优先级高于=，而or的优先级是低于=



运算符的优先级：

在一个式子中可能会出现多种运算符，但运算符之间会有优先级。

**口诀：**

**单、算、关、逻、条、赋、逗**

7、条件运算符

三元运算符

语法：

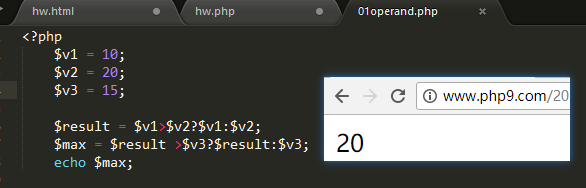
表达式?表达式A:表达式B;

运算规则：

先计算表达式是否成立，如果成立取表达式A的值，如果不成立取表达式B的值。

就是算简单的分支语句

示例：



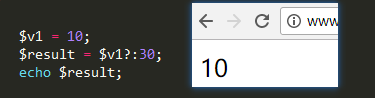
扩展：了解

语法：

变量?:表达式B

运算规则：

如果变量能转换为true，那么就取变量，否则取表达式B。



8、错误控制运算符

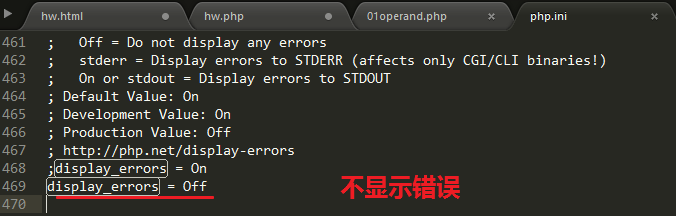
①、错误抑制符

@ 错误抑制符

作用是不显示错误信息。



②、php.ini中隐藏错误



说明：

此种配置影响是全局。所有的使用此份php的配置文件的站点都会受此影响。

③、脚本级的错误控制

脚本级的错误控制仅限于当前的Php脚本文件。

ini\_set() 主要用于在Php脚本中来设置php.ini中的配置项

语法：

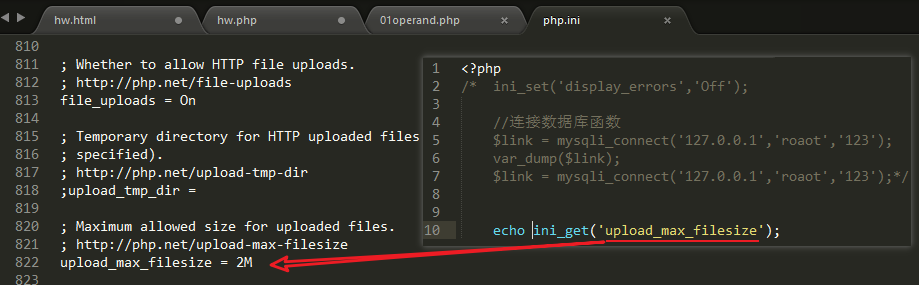
ini\_set(配置项名,值);

示例：



ini\_get(配置项名) 主要是用于获取php.ini配置文件中的配置项的值

示例：



9、位运算符(了解)

位运算都是对一个数的二进制形式进行运算。

8421BCD码

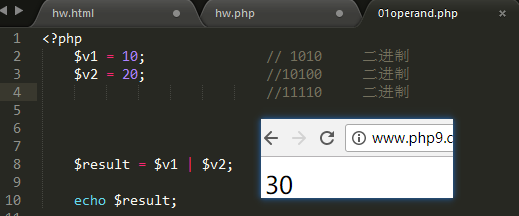
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 位 | 位7 | 位6 | 位5 | 位4 | 位3 | 位2 | 位1 | 位0 |
| 权 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 二进制 | 1 |  | 1 |  | 1 | 0 | 1 | 0 |

| 按位或

运算规则：

参与运算的两个操作数的二进制形式对应的位进行与运算，如果对应的位只要有1个为1那么最终的结果位就1，如果都为0，最终结果位为0.

10 || 20 true | true true

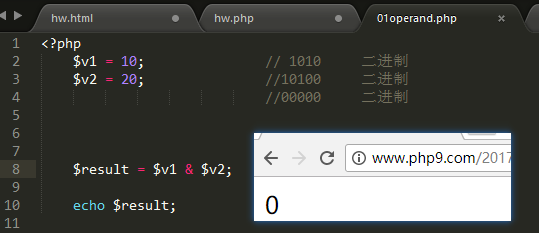


& 按位与

运算规则：

参与运算的两个操作数的二进制形式对应的位，全部为1，最终的结果位为1，不是为0

示例：

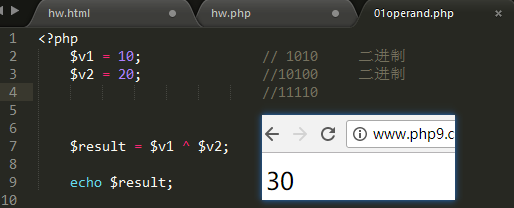


^ 按位异或

运算规则：

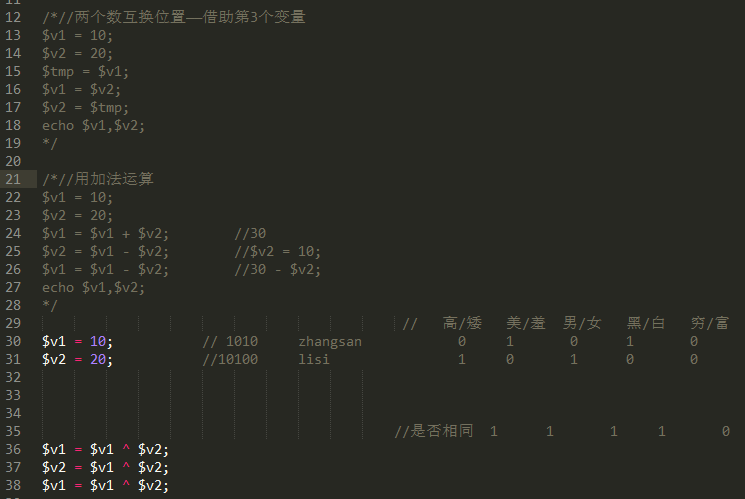
参与运算的两个操作数的二进制形式对应的位，如果相同为0，不同为1

示例：



主要是用于记录同点。

扩展：



~ 按位非(单目运算符)

运算规则：

对一个操作数的二进制形式中的每一位进行取反。

<< 左移

运算规则：

对二进制数据整体向左移指定的位数。左侧移出要被忽略，右侧补0



左移，可以快速对一个数加位。

>> 右移

运算规则：

对二进制数据整体向右移指定的位数。右侧移出要被忽略，左侧补0

示例：



三、原码、反码、补码(了解)

原码：

就是带有正负号的二进制数。

在计算机内所有的信息的表示都是使用0与1表示的，正负号与不例外。

人们规定，如果是正数，在最高位使用0表示+；如果是负数在最高位使用1表示-

反码：

正数的反码与原码相同

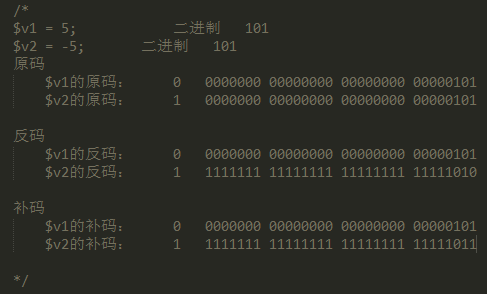
负数的反码，最高位的符号位不变，其余位取反。

补码：

正数的补码与原码相同

负数的补码是在反码的基础上加1

示例：

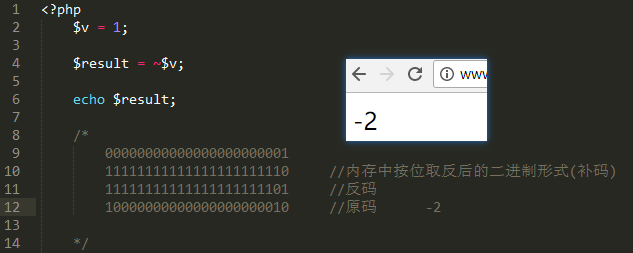


提示：

以上只是在内存中的一个表现形式，最终在显示时还要转换为十进制数。

对于正数，原码 到 反码 到 补码 没有任何的变动，所以找二进制对应的十进制数即可。

对于负数，先由补码反推出反码，再由反码反推出原码，再将原码转换为十进制



四、进制转换函数

dec decimal 十进制

bin binary 二进制

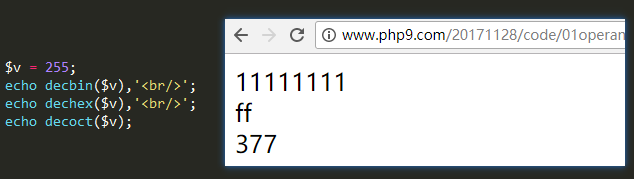
oct octet 八进制

hex hex 十六进制

decbin() 十进制转换为二进制

dechex() 十进制转换为十六进制

decoct() 十进制转换为八进制

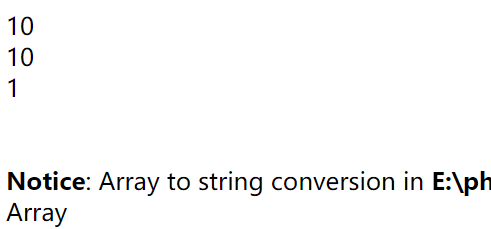


五、php的输出语法

echo

只能输出标量数据类型，对于任何数据都要转换为字符串输出

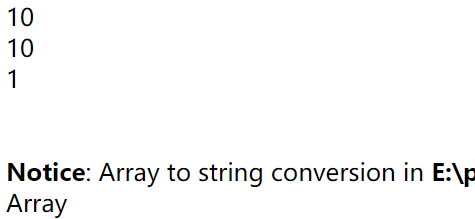
示例：



print()

只能输出标量数据类型，对于任何数据都要转换为字符串输出

示例：



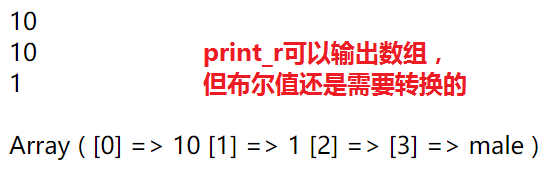
**echo 与print的区别：**

echo没有返回值，print有返回值。

print\_r()

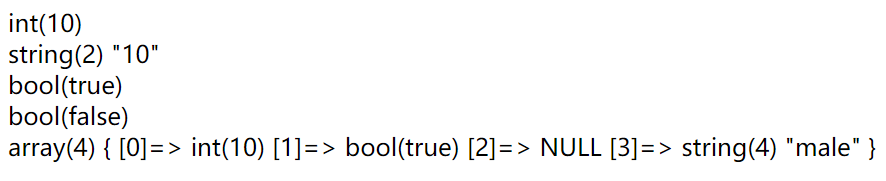
可以输出标量，及复合数据类型。

示例：



var\_dump();

主要是用于程序员进行代码调试，可以输出十分详细的信息。并不是为了输出信息给用户。



sprintf();

用于格式化输出。

语法：

sprint(格式化字符串,变量1,变量2,….)

说明：

格式化占位符有

%b 二进制

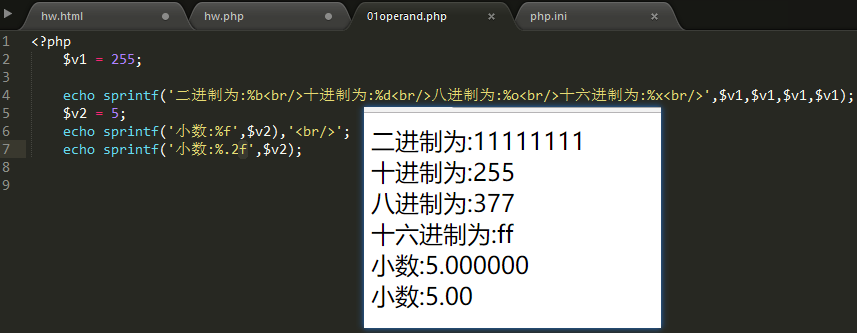
%d 十进制

%o 八进制

%f 浮点

%x 十六进制

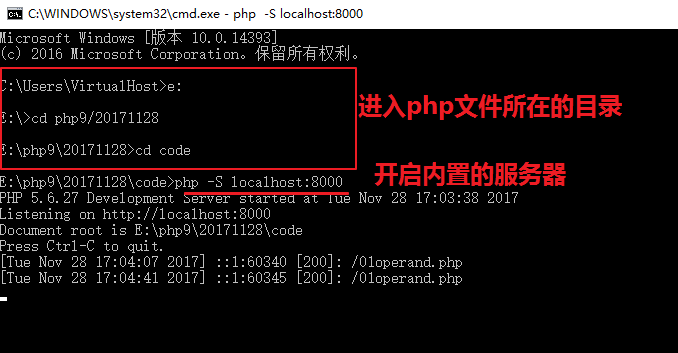
示例：

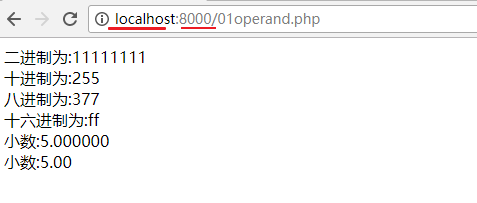


六、php.exe文件

1、php内置的web服务器

php.exe -S locahost:端口号



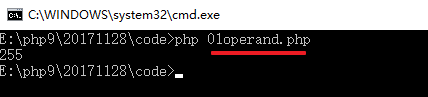


提示：

创建了内置的web服务器后，也需要使用浏览器访问php文件。

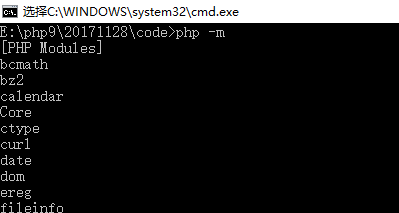
2、php\_cli

php的命令行模式

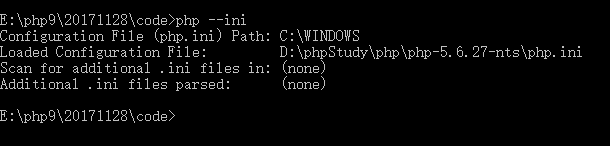


php –m

查看php所加载的功能模块



php --ini



php –f 文件的名



php –r “一行Php命令”

