# LINUX 安装 PHP

前言：

各软件包下载地址：

Apache下载地址：

<http://httpd.apache.org/download.cgi>

http://archive.apache.org/dist/httpd/

Mysql下载地址：

<https://dev.mysql.com/downloads/mysql/>

http://mirror.bit.edu.cn/mysql/Downloads/

Php下载地址：

<http://php.net/downloads.php>

<http://php.net/releases/>

Php扩展下载：

<http://pecl.php.net/>

<http://windows.php.net/downloads/pecl/releases/>

<http://downloads.php.net/pierre/>

依赖包下载地址：<http://download.chinaunix.net/>

Rpm包下载地址：

<http://www.rpmfind.net/linux/>

[https://pkgs.org/download/](https://pkgs.org/download/pcre)

开源软件镜像服务下载地址：<http://mirror.bit.edu.cn/web/>

Linux系统下载地址：

<https://www.linux.org/pages/download/>

<http://archive.kernel.org/centos-vault/>

Libxml2下载地址：

<http://xmlsoft.org/sources/>

## 一，源码安装

测试安装环境： CentOS6.8 64位。

安装位置： /usr/local/

软件包存放位置： /home/

系统支持：gcc，gcc-c++，make编译环境

如无gcc，gcc-c++，make编译环境，则按以下步骤下载安装编译工具包（已安装则忽略）

1. 使用rpm方式安装编译工具包

下载以下软件包（操作系统不同，软件包也不一样）:

ppl-0.10.2-11.el6.x86\_64.rpm

cloog-ppl-0.15.7-1.2.el6.x86\_64.rpm

cpp-4.4.7-17.el6.x86\_64.rpm

mpfr-2.4.1-6.el6.x86\_64.rpm

gcc-4.4.7-17.el6.x86\_64.rpm

gcc-c++-4.4.7-17.el6.x86\_64.rpm

gcc-java-4.4.7-17.el6.x86\_64.rpm

libstdc++-devel-4.4.7-17.el6.x86\_64.rpm

安装顺序：

1，rpm -ivh ppl-0.10.2-11.el6.x86\_64.rpm

2，rpm -ivh cloog-ppl-0.15.7-1.2.el6.x86\_64.rpm

3，rpm -ivh mpfr-2.4.1-6.el6.x86\_64.rpm

4，rpm -ivh cpp-4.4.7-17.el6.x86\_64.rpm

5，rpm -ivh gcc-4.4.7-17.el6.x86\_64.rpm

6，rpm -ivh gcc-c++-4.4.7-17.el6.x86\_64.rpm

7，rpm -ivh libstdc++-devel-4.4.7-17.el6.x86\_64.rpm

8，rpm -ivh gcc-c++-4.4.7-17.el6.x86\_64.rpm

9，rpm -ivh gcc-java-4.4.7-17.el6.x86\_64.rpm(编译java需要，可选)

安装成功后，写一个c语言（hello.c）文件：

内容如下: #include<stdio.h>

main(){

printf("hello word\n");

}

保存退出。

执行gcc hello.c，在查看目录，若出现a.out文件，则运行./a.out，可运行表示安装成功。

B) 使用yum方式安装编译工具包

前提：电脑必须可以联网。

执行命令: yum -y install gcc gcc-c++ make

### 安装前准备软件包

autoconf-2.69.tar.gz 可选

zlib-1.2.11.tar.gz 必备包(已安装则忽略)

libxml2-2.7.4.tar.gz php扩展包

libmcrypt-2.5.8.tar.gz php扩展包

jpegsrc.v9.tar.gz GD依赖包

libpng-1.6.32.tar.gz GD依赖包

freetype-2.4.8.tar.bz2 GD依赖包

libgd-2.2.3.tar.gz php扩展包

php-5.6.32.tar.gz php安装包

使用高版本的GD库的话，可能需要以下两个包：

libXpm-3.5.10-2.el6.x86\_64.rpm

libXpm-devel-3.5.10-2.el6.x86\_64.rpm

### 安装zlib(已安装则忽略)

**Shell># cd /home/**

**Shell># tar -zxvf zlib-1.2.11.tar.gz**

**Shell># cd zlib-1.2.11.tar.gz**

**Shell># ./configure**

**Shell># make && make install**

注意：

1，若要指定zlib安装路径的话，则在/configure 之后追加 --prefix=安装路径，例如：./configure --prefix=/usr/local/zlib

2，如果系统是64位，需要使用64位元的方法编译安装zlib:

CFLAGS=”-03 -fPIC” ./configusr --prefix=/usr/local/zlib

3，若想知道./configure 有什么参数可以配置? 可以使用：

./configure --help【适用所有源码编译的软件】

4，若要指定安装路径的话，记得先卸载系统自带的zlib库，并且卸载时间必须在安装zlib时已经使用make，但未使用make install 的时候卸载。

5，指定安装好zlib后，在编辑/etc/ld.so.conf 文件，在下面加入一行/usr/local/zlib/lib，保存退出。再使用ldconfig更新缓存即可。

### 安装autoconf(可选)

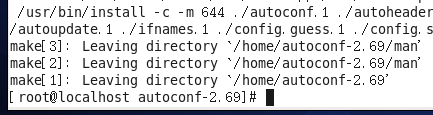
**Shell># cd /home/**

**Shell># tar -zxvf autoconf-2.69.tar.gz**

**Shell># cd autoconf-2.69**

**Shell># ./configure**

**Shell># make && make install**

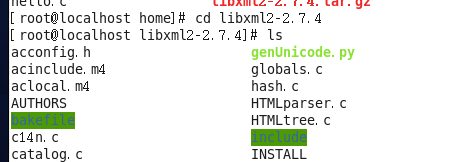


### 安装libxml2

**Shell># cd /home**

**Shell># tar -zxvf libxml2-2.7.4.tar.gz**

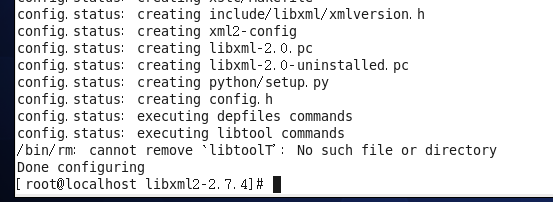
**Shell># cd libxml2-2.7.4**



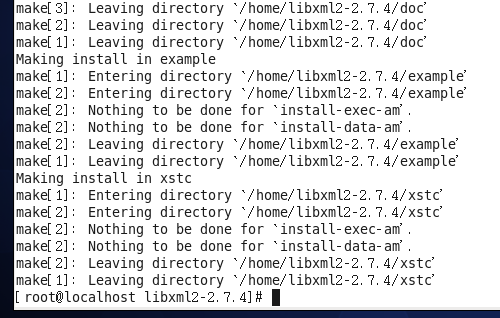
**Shell># ./configure --prefix=/usr/local/libxml2 --without-zlib**

**解释：--prefix 软件安装路径**

**--without-zlib 不需要zlib代码优化支持**



**Shell># make && make install**

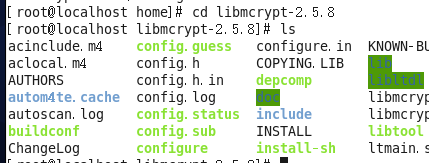


### 安装libmcrypt

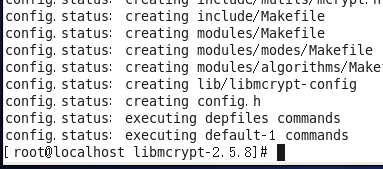
**Shell># cd /home/**

**Shell># tar -zxvf libmcrypt-2.5.8.tar.gz**

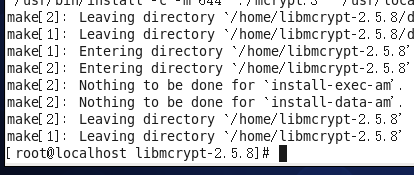
**Shell># cd libmcrypt-2.5.8**



**Shell># ./configure --prefix=/usr/local/libmcrypt**



**Shell># make && make install**

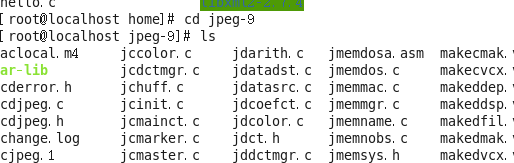


### 安装jpegsrc

**Shell># cd /home/**

**Shell># tar -zxvf jpegsrc.v9.tar.gz**

**Shell># cd jpegsrc-9**



**Shell># ./configure --prefix=/usr/local/jpeg9 \**

**> --enable-shared --enable-static**

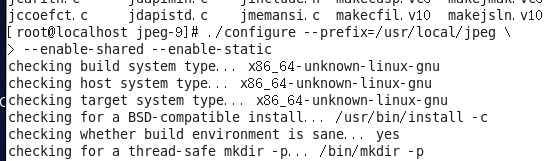
**解释：**

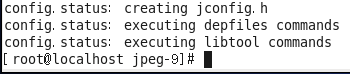
**--enable-shared 把jpeg需要的函数库程序编译到该软件里边**

**优点：函数调用速度快 缺点：软件本身比较大**

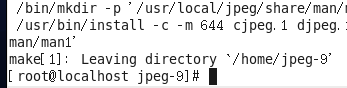
**--enable-static 静态方式函数处理，需要什么函数就include进来**

**优点：软件本身比较小 缺点：函数调用速度慢**





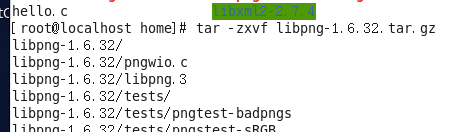
**Shell># make && make install**



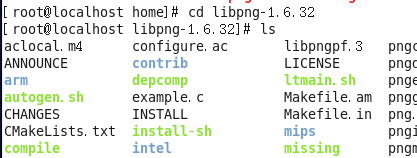
### 安装libpng

**Shell># cd /home/**

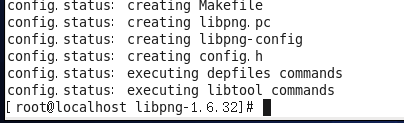
**Shell># tar -zxvf libpng-1.6.32.tar.gz**



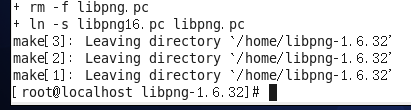
**Shell># cd libpng-1.6.32**



**Shell># ./configure --prefix=/usr/local/libpng**



**Shell># make && make install**

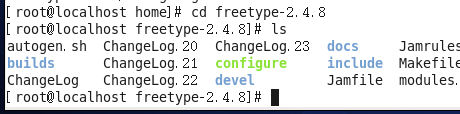


### 安装freetype

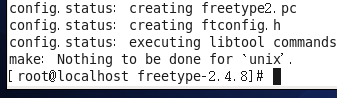
**Shell># cd /home/**

**Shell># tar -jxvf freetype-2.4.8.tar.bz2**

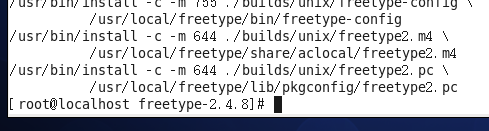
**Shell># cd freetype-2.4.8**



**Shell># ./configure --prefix=/usr/local/freetype**



**Shell># make && make install**



注意安装freetype时有可能提示如下错误：

**either set the LIBPNG\_CFLAGS and LIBPNG\_LDFLAGS environment variables**

这是提示找不到libpng的安装位置，所以需要将libpng的bin目录配置到环境变量中：

解决方法：

Shell># cd /etc/

Shell># vim /etc/profile

找到：export 开头那行，在上面加入以下两行

**LIB\_PNG=/usr/local/libpng/bin**

**PATH=$LIB\_PNG:$PATH**

  export PATH ......【该行无需修改】

Shell># ./etc/profile

Shell># rm -rf /usr/local/freetype

Shell># rm -rf /home/freetype-2.4.8

重新解压安装freetype。

### 安装libXpm

**Shell># cd /home/**

**Shell># rpm -ivh libXpm-3.5.10-2.el6.x86\_64.rpm**

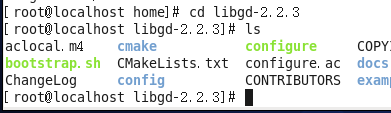
**Shell># rpm -ivh libXpm-devel-3.5.10-2.el6.x86\_64.rpm**

### 安装libgd

**Shell># cd /home/**

**Shell># tar -zxvf libgd-2.2.3.tar.gz**

**Shell># cd libgd-2.2.3**



**Shell># ./configure --prefix=/usr/local/gd \**

**> --with-jpeg=/usr/local/jpeg9 \**

**> --with-png=/usr/local/libpng/ \**

**> --with-freetype=/usr/local/freetype/ \**

**> --with-zlib**

解释：

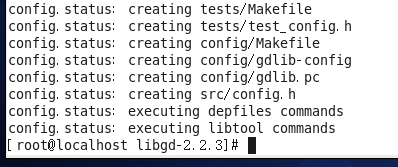
--with-jpeg=$PATH 支持jpeg格式图片

--with-png=$PATH 支持 png 格式图片

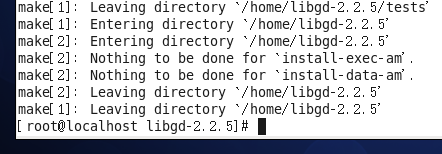
--with-freetype=$PATH 支持图片写入字体

--with-zlib 支持图片代码压缩

注：--with-扩展名称 若不指定位置，则会自动到系统默认安装位置查询



**Shell># make && make install**



提示：若编译失败，提示错误则重新安装高版本的GD库或者2.1以下版本的GD库。

### 安装php

**Shell># cd /home**

**Shell># tar --zxvf php-5.6.32.tar.gz**

**Shell># cd php-5.6.32**



**Shell># ./configure --prefix=/usr/local/php \**

**> --with-config-file-path=/usr/local/php/etc \ （可选）**

**> --with-apxs2=/usr/local/httpd24/bin/apxs \**

**> --with-mysql=mysqlnd \**

**> --with-mysqli=mysqlnd \**

**> --with-pdo-mysql=mysqlnd \**

**> --with-zlib --with-libxml-dir=/usr/local/libxml2/ \**

**> --with-freetype-dir=/usr/local/freetype/ \**

**> --with-mcrypt=/usr/local/libmcrypt/ \ （可选）**

**> --with-png-dir=/usr/local/libpng/ \**

**> --with-jpeg-dir=/usr/local/jpeg9/ \**

**> --with-gd=/usr/local/gd/ \**

**> --enable-mbstring=all \**

**> --enable-mbregex \**

**> --enable-shared \**

**> --enable-sockets \ （可选）**

**> --enable-soap \ （可选）**

**> --enable-pdo （可选）**

解释：

--prefix=/usr/local/php

php安装路径

--with-config-file-path=/usr/local/php/etc

指定php配置文件存放路径

--with-apxs2=/usr/local/httpd24/bin/apxs

apache支持，其作用：

1，给apache生成PHP模块

2，修改httpd.conf文件，使其引入php模块

--with-mysql=mysqlnd

--with-pdo-mysql=mysqlnd

--with-mysqli=mysqlnd

当前环境没有安装mysql的情况下，使用mysqlnd做为扩展功能模块路径的话，则mysqlnd将会使用MySQL本机驱动。若已安装mysql，则填写mysql功能模块的路径。

--with-freetype-dir=/usr/local/freetype/

--with-zlib

--with-libxml-dir=/usr/local/libxml2/

--with-freetype-dir=/usr/local/freetype/

--with-mcrypt=/usr/local/libmcrypt/

--with-png-dir=/usr/local/png/

--with-jpeg-dir=/usr/local/jpeg9/

--with-gd=/usr/local/gd/

以上都是已安装的功能模块扩展

--enable-mbstring

宽字节函数库对php的支持（中文字符串处理支持）。默认开启php\_mbstring

--enable-mbregex

正则表达式对PHP的支持

--enable-shared

把所有的php扩展全部安装到php软件本身

--enable-sockets

变量激活socket通信特征

--enable-soap

变量激活SOAP和web service支持

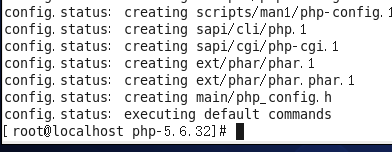
--enable-pdo

激活pdo

更详细的配置参数，可参考：

<http://www.cnblogs.com/HKUI/p/5137115.html>





**Shell># make && make install**

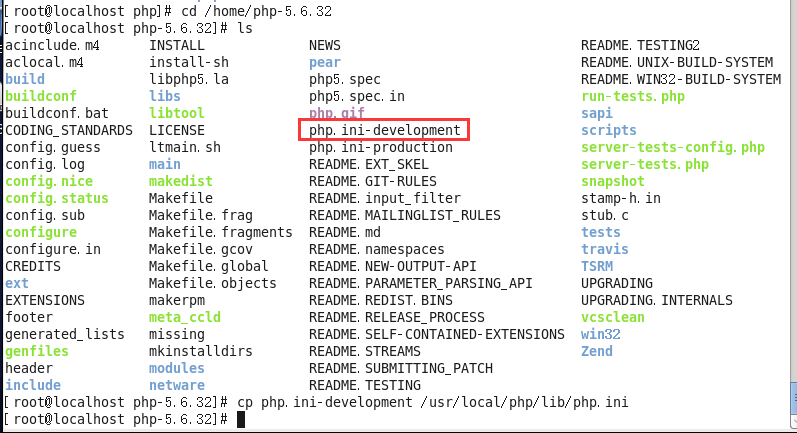


### 配置PHP

1. 拷贝php.ini文件到 /etc 目录

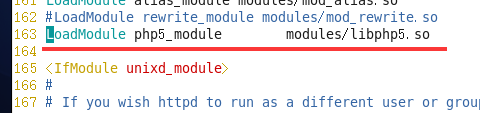
**Shell># cd /home/php-5.6.32/**

**Shell># cp php.ini-development /usr/local/php/lib/php.ini**



1. 配置apache使其支持PHP

**php安装完成后，在/usr/local/httpd24/conf/httpd.conf文件中，会自动引入PHP模块：**

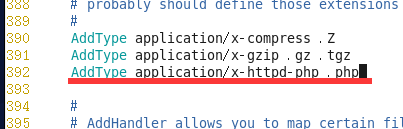


编辑apache配置文件：

Shell># vim /usr/local/httpd24/conf/httpd.conf

1，在httpd.conf(apache 主配置文件)中增加：

AddType application/x-httpd-php .php



2，找到下面这段话：

<IfModule dir\_module>

DirectoryIndex index.html

</IfModule>

在index.html前面添加index.php

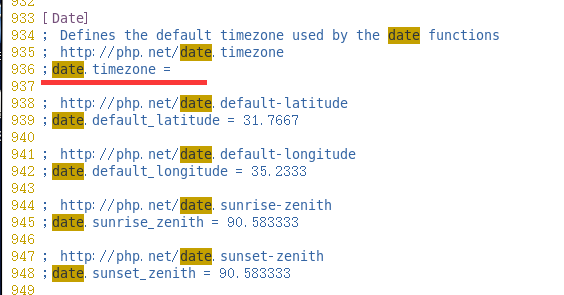


1. 编写php文件，配置是否成功

index.php文件内容：<?php echo phpinfo(); ?>



注：若提示时区错误，在/usr/local/php/lib/php.ini 文件中设置。



可输入：PRC 或者 Asia/shanghai，前面;号要去掉。

### 可能发生的编译错误

在编译PHP的时候会出现的错误情况！

a) 按照以上配置，make时可能会报如下错误：

/usr/local/src/php-5.5.6/ext/gd/gd.c:57:22: error: X11/xpm.h: No such file or directory

make: \*\*\* [ext/gd/gd.lo] Error 1

问题说明：这是提示gd库不支持xpm

解决方法：

1，安装libxpm

Shell># yum -y install libXpm

Shell># yum -y install libXpm-devel

2，按照上面的方法重新安装gd库。（需要先卸载GD库）

Shell># rm -rf /usr/loca/gd

Shell># rm -rf /home/libgd-2.2.3

Shell># cd /home

Shell># tar -zxvf libgd-2.2.3.tar.gz

Shell># cd libgd-2.2.3

Shell># ./configure --prefix=/usr/local/gd \

> --with-jpeg=/usr/local/jpeg9/ \

> --with-png=/usr/local/libpng/ \

> --with-freetype=/usr/local/freetype \

> --with-zlib

Shell># make && make install

3，再次按照上面的方法安装php

Shell># rm -rf /usr/local/php

Shell># rm -rf /home/php-5.6.32

Shell># cd /home/

Shell># tar -zxvf php-5.6.32.tar.gz

Shell># cd php-5.6.32

Shell># ./configure --prefix=/usr/local/php \

> --with-config-file-path=/usr/local/php/etc \ （可选）

> --with-apxs2=/usr/local/httpd24/bin/apxs \

> --with-mysql=mysqlnd \

> --with-mysqli=mysqlnd \

> --with-pdo-mysql=mysqlnd \

> --with-zlib --with-libxml-dir=/usr/local/libxml2/ \

> --with-freetype-dir=/usr/local/freetype/ \

> --with-mcrypt=/usr/local/libmcrypt/ \ （可选）

> --with-png-dir=/usr/local/libpng/ \

> --with-jpeg-dir=/usr/local/jpeg9/ \

> --with-gd=/usr/local/gd/ \

> --enable-mbstring=all \

> --enable-mbregex \

> --enable-shared \

> --enable-sockets \ （可选）

> --enable-soap \ （可选）

> --enable-pdo （可选）

Shell># make && make install

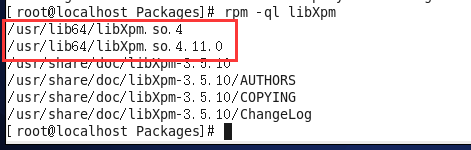
b) 检测编译环境报错：

ext/gd/gd.c:1330: undefined reference to `XpmLibraryVersion'

问题说明：没有找到xpm库的安装位置

解决方法：

1，rpm -ql libXpm : 查询出libXpm的安装位置，发现在/usr/lib64/下



2，重新安装PHP，在./configure中增加 --with-xpm-dir=/usr/lib64/

Shell># cd php-5.6.32

Shell># ./configure --prefix=/usr/local/php \

> --with-config-file-path=/usr/local/php/etc \ （可选）

> --with-apxs2=/usr/local/httpd24/bin/apxs \

> --with-mysql=mysqlnd \

> --with-mysqli=mysqlnd \

> --with-pdo-mysql=mysqlnd \

> --with-zlib --with-libxml-dir=/usr/local/libxml2/ \

> --with-freetype-dir=/usr/local/freetype/ \

> --with-mcrypt=/usr/local/libmcrypt/ \ （可选）

> --with-png-dir=/usr/local/libpng/ \

> --with-jpeg-dir=/usr/local/jpeg9/ \

> --with-gd=/usr/local/gd/ \

> --with-xpm-dir=/usr/lib64/ \

> --enable-mbstring=all \

> --enable-mbregex \

> --enable-shared \

> --enable-sockets \ （可选）

> --enable-soap \ （可选）

> --enable-pdo （可选）

Shell># make && make install

### 卸载源码安装的PHP

步骤：

1，进入php的解压目录

2，执行make uninstall（若提示无该命令的话，则查看Makefile文件，主要看install部分，从其中找出tar格式的文件被复制到了什么路径，然后进入相应的目录进行删除即可）

3，删除./configure指定php的安装路径。如:/usr/local/php

4，删除php解压目录

**Shell># cd /home/php-5.6.32/**

**Shell># make uninstall**

**Shell># rm -rf /usr/local/php**

**Shell># cd /home/**

**Shell># rm -rf php-5.6.32**

### 源码软件安装错误解决

1. 已执行configure操作

》根据正确的参数重新configure即可

2，已经执行configure、make操作

》删除解压后的目录，重新解压、configure、make

3，已经执行configure、make、make install

》删除安装后的文件（有指定安装目录情况/usr/local/httpd24）

》删除解压后的目录

》重新解压，重新configure，重新make，重新make install

## 二，rpm安装

1. 准备php安装包

php-5.3.3-47.el6.x86\_64.rpm

php-cli-5.3.3-47.el6.x86\_64.rpm （依赖包，必须）

php-common-5.3.3-47.el6.x86\_64.rpm （依赖包，必须）

php-xml-5.3.3-47.el6.x86\_64.rpm （可选）

php-xmlrpc-5.3.3-47.el6.x86\_64.rpm （可选）

B) 安装php

**安装顺序如下：**

**Shell># rpm -ivh php-cli-5.3.3-47.el6.x86\_64.rpm**

**Shell># rpm -ivh php-common-5.3.3-47.el6.x86\_64.rpm**

**Shell># rpm -ivh php-5.3.3-47.el6.x86\_64.rpm**

**以下php扩展包，顺序随意：**

**Shell># rpm -ivh php-xml-5.3.3-47.el6.x86\_64.rpm**

**Shell># rpm -ivh php-xmlrpc-5.3.3-47.el6.x86\_64.rpm**

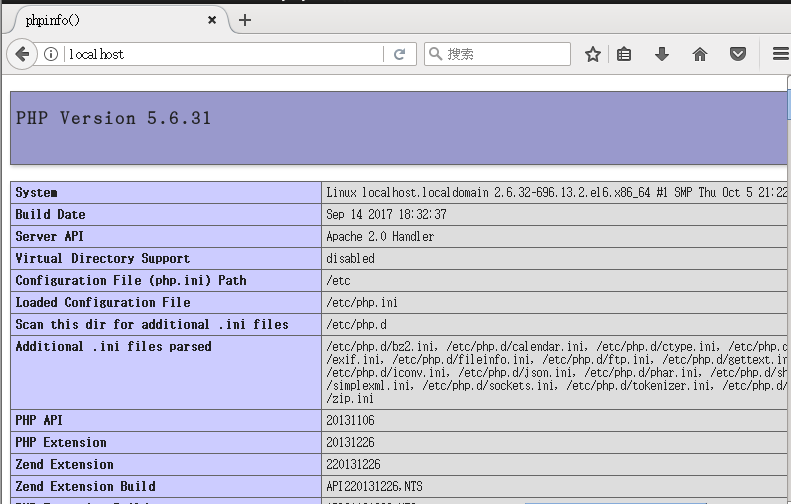
C) 编写php文件测试

由于apache是使用yum方式安装：

站点目录位于：/var/www/html/

Apache配置文件位置位于：/etc/httpd/conf/httpd.conf

Apache模块存放位置：/etc/httpd/modules/ 或者 /usr/lib64/httpd/modules/



注：若访问无效（无法解析php文件）的话。（yum方式安装基本上会自动配置好）

则编辑 apache 配置文件，使其支持php。步骤参考：一-12

D) rpm方式安装的php路径

CentOS下：

PHP主目录 /etc/php.d/   
 PHP配置文件 /etc/php.ini   
 PHP模块位置 /usr/lib/php/ 或者 /usr/lib64/php/

Ubuntu下：

PHP主目录 /etc/php5/   
 PHP配置文件 /etc/php5/apache2/php.ini   
 PHP模块位置 /usr/lib/php5/

E) 增加php扩展

**前提：需要下载相应扩展的rpm包。**

**使用命令: rpm -ivh xxxxxx.rpm 即可。**

**安装完成后，该扩展会自动开启。**

**例如安装soap扩展(存放在/home/目录下)：**

**php-soap-5.3.3-47.el6.x86\_64.rpm**

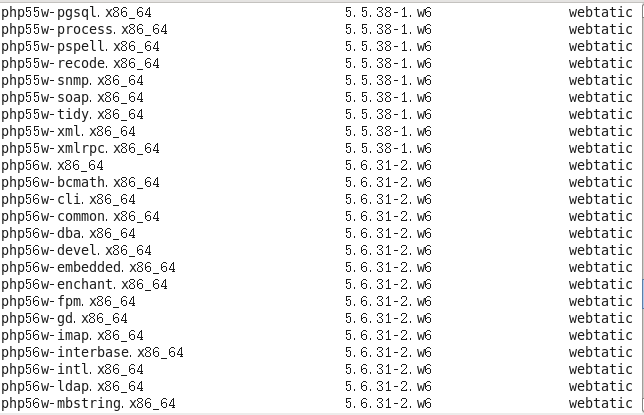
**Shell># cd /home/**

**Shell># rpm -ivh php-soap-5.3.3-47.el6.x86\_64.rpm**

## 三，yum安装

1) 可根据yum搜索的php资源包列表，安装合适的php版本：

**Shell># yum list | grep php**



2) 安装php:

**Shell># yum -y install php56w**

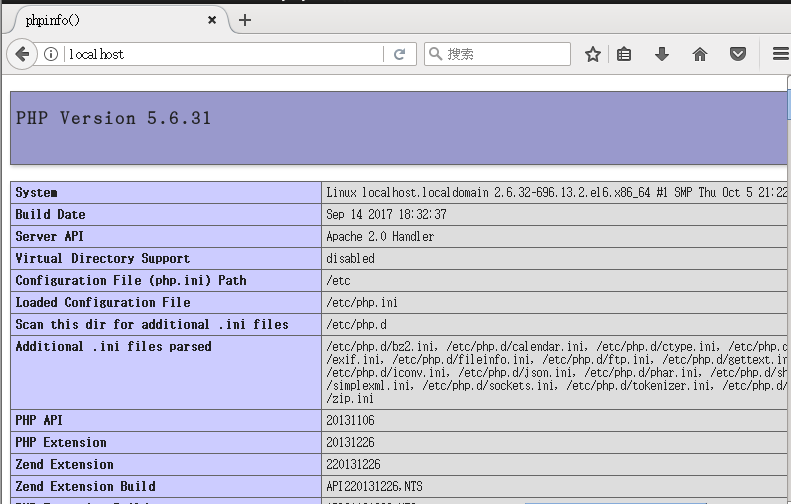
1. 编写php文件测试

由于apache是使用yum方式安装：

站点目录位于：/var/www/html/

Apache配置文件位置位于：/etc/httpd/conf/httpd.conf

Apache模块存放位置：/etc/httpd/modules/ 或者 /usr/lib64/httpd/modules/



注：若访问无效（无法解析php文件）的话。（yum方式安装基本上会自动配置好）

则编辑 apache 配置文件，使其支持php。步骤参考：一-12

1. yum安装php路径

CentOS下：

PHP主目录 /etc/php.d/   
 PHP配置文件 /etc/php.ini   
 PHP模块位置 /usr/lib/php/ 或者 /usr/lib64/php/

Ubuntu下：

PHP主目录 /etc/php5/   
 PHP配置文件 /etc/php5/apache2/php.ini   
 PHP模块位置 /usr/lib/php5/

1. 增加php扩展

**Shell># yum -y install php56w-mysql php56w-gd**

**解释：**

**php56w-mysql**

**php56w 安装的php名称（当前实例安装的是: php56w）**

**mysql 扩展名称，yum方式安装好后会自动开启**

**注：多个扩展包，使用空格隔开。**

## 四，增加PHP扩展

**源码编译的PHP自带一些扩展文件：cd php-5.x.x/etc/ 目录。**

**若需要编译第三方扩展，则自行下载源码扩展包**

这里以编译fileinfo扩展为例：

环境：

源码编译PHP

目录解压位置：/home/php-5.6.32

PHP安装位置：/usr/local/php

Php.ini文件存放位置：/usr/local/php/lib/

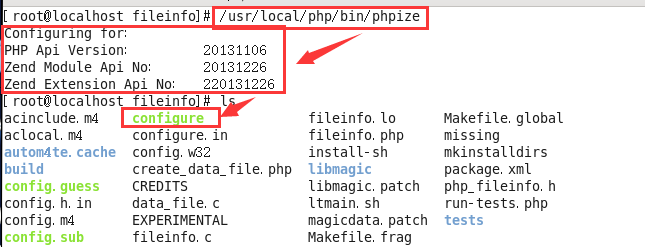
1，找到扩展包

**Shell># cd /home/php-5.6.32/etc/fileinfo**



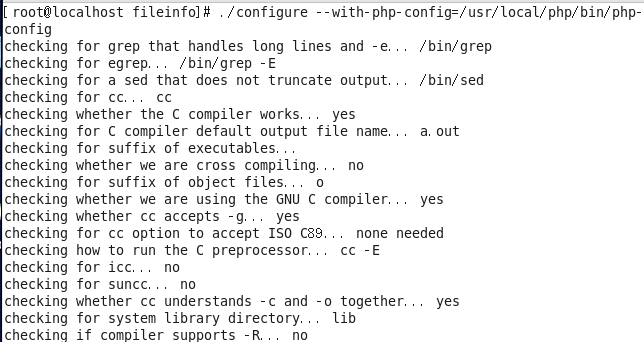
2，编译

**Shell># /usr/local/php/bin/phpize**

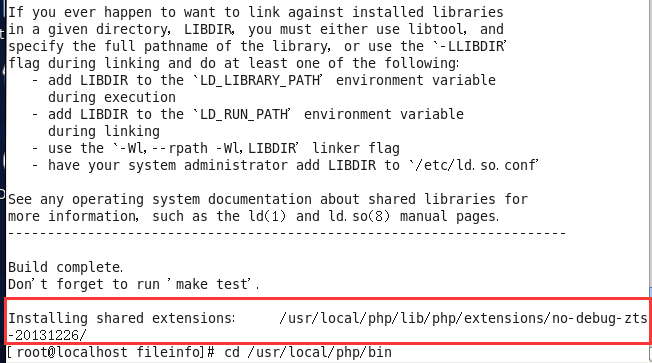


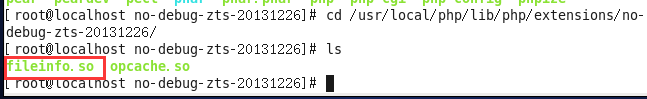
**Shell># ./configure \**

**> --with-php-config=/usr/local/php/bin/php-config**



**Shell># make && make install**





解释：

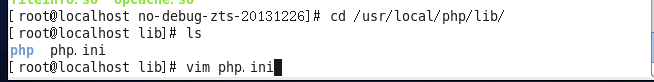
首先执行phpize，它是用来安装php扩展的工具，若果你设置了环境变量，则可以使用phpize，若为设置，请使用完整路径。这个工具会在当前目录下生成configure文件。

--with-php-config=/usr/local/php/bin/php-config 中的php-config 路径根据实际情况填写。

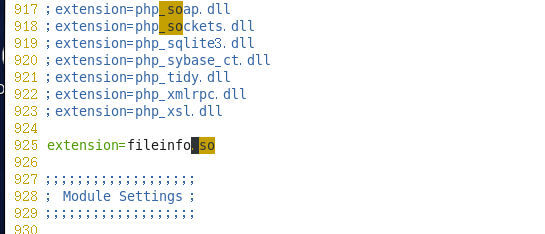
编译完成后，会在php/lib/php/extensions/(在php.ini的extension\_dir可找到)目录下找到生成的fileinfo.so文件

3，添加配置

extension = “fileinfo.so”







添加完后，保存退出，重新启动web(apache)服务器。

## 五，配置详解

PHP配置文件详解php.ini

[PHP]

; PHP还是一个不断发展的工具，其功能还在不断地删减

; 而php.ini的设置更改可以反映出相当的变化，

; 在使用新的PHP版本前，研究一下php.ini会有好处的

;;;;;;;;;;;;;;;;;;;

; 关于这个文件 ;

;;;;;;;;;;;;;;;;;;;

; 这个文件控制了PHP许多方面的观点。为了让PHP读取这个文件，它必须被命名为

; 'php.ini'。PHP 将在这些地方依次查找该文件：当前工作目录；环境变量PHPRC

; 指明的路径；编译时指定的路径。

; 在windows下，编译时的路径是Windows安装目录。

; 在命令行模式下，php.ini的查找路径可以用 -c 参数替代。

; 该文件的语法非常简单。空白字符和用分号';'开始的行被简单地忽略（就象你可能

; 猜到的一样）。 章节标题（例如 : [Foo]）也被简单地忽略，即使将来它们可能

; 有某种的意义。

;

; 指示被指定使用如下语法：

; 指示标识符 = 值

; directive = value

; 指示标识符 是 \*大小写敏感的\* - foo=bar 不同于 FOO = bar。

;

; 值可以是一个字符串，一个数字，一个 PHP 常量 (如： E\_ALL or M\_PI), INI 常量中的

; 一个 (On, Off, True, False, Yes, No and None) ，或是一个表达式

; (如: E\_ALL & ~E\_NOTICE), 或是用引号括起来的字符串(" foo" ).

;

; INI 文件的表达式被限制于位运算符和括号。

; | bitwise OR

; & bitwise AND

; ~ bitwise NOT

; ! boolean NOT

;

; 布尔标志可用 1, On, True or Yes 这些值置于开的状态。

; 它们可用 0, Off, False or No 这些值置于关的状态。

;

; 一个空字符串可以用在等号后不写任何东西表示，或者用 None 关键字:

;

; foo = ; 将foo置为空字符串

; foo = none ; 将foo置为空字符串

; foo = " none" ; 将foo置为字符串'none'

;

; 如果你值设置中使用常量，而这些常量属于动态调入的扩展库（不是 PHP 的扩展，就是

; Zend 的扩展），你仅可以调入这些扩展的行\*之后\*使用这些常量。

;

; 所有在 php.ini-dist 文件里设定的值与内建的默认值相同（这是说，如果 php.ini

; 没被使用或者你删掉了这些行，默认值与之相同）。

;;;;;;;;;;;;;;;;;;;;

; 语言选项 ;

;;;;;;;;;;;;;;;;;;;;

engine = On

; 使 PHP scripting language engine（PHP 脚本语言引擎）在 Apache下有效。

short\_open\_tag = On

; 允许 tags 将被识别。

asp\_tags = Off

; 允许ASP-style tags

precision = 14

; 浮点类型数显示时的有效位数

y2k\_compliance = Off

; 是否打开 2000年适应 (可能在非Y2K适应的浏览器中导致问题)

output\_buffering = Off

; 输出缓存允许你甚至在输出正文内容之后发送 header（标头，包括cookies）行

; 其代价是输出层减慢一点点速度。你可以使用输出缓存在运行时打开输出缓存，

; 或者在这里将指示设为 On 而使得所有文件的输出缓存打开。

output\_handler = ; 你可以重定向你的脚本的所有输出到一个函数，

; 那样做可能对处理或以日志记录它有用。

; 例如若你将这个output\_handler 设为" ob\_gzhandler" ,

; 则输出会被透明地为支持gzip或deflate编码的浏览器压缩。

; 设一个输出处理器自动地打开输出缓冲。

implicit\_flush = Off

; 强制flush（刷新）让PHP 告诉输出层在每个输出块之后自动刷新自身数据。

; 这等效于在每个 print() 或 echo() 调用和每个 HTML 块后调用flush()函数。

; 打开这项设置会导致严重的运行时冲突，建议仅在debug过程中打开。

allow\_call\_time\_pass\_reference = On

; 是否让强迫函数调用时按引用传递参数。这一方法遭到抗议，

; 并可能在将来版本的PHP/Zend里不再支持。

; 受到鼓励的指定哪些参数按引用传递的方法是在函数声明里。

; 你被鼓励尝试关闭这一选项并确认你的脚本仍能正常工作，以保证在将来版本的语言里

; 它们仍能工作。（你将在每次使用该特点时得到一个警告，而参数将按值而不是按引用

; 传递）。

; Safe Mode 安全模式

safe\_mode = Off

safe\_mode\_exec\_dir =

safe\_mode\_allowed\_env\_vars = PHP\_

; ？Setting certain environment variables

; ？may be a potential security breach.

; 该指示包含用逗号分隔的前缀列表。安全模式中，用户仅可以替换

; 以在此列出的前缀开头的环境变量的值。

; 默认地，用户将仅能 设定以PHP\_开头的环境变量，（如: PHP\_FOO=BAR）。

; 注意: 如果这一指示为空，PHP 将让用户更改任意环境变量!

safe\_mode\_protected\_env\_vars = LD\_LIBRARY\_PATH

; 这条指示包含一个用逗号分隔的环境变量列表，那是最终用户将不能用putenv () 更改的。

; 这些变量甚至在safe\_mode\_allowed\_env\_vars 设置为允许的情况下得到保护。

disable\_functions =

; 这条指示让你可以为了安全的原因让特定函数失效。

; 它接受一个用逗号分隔的函数名列表。

; 这条指示 \*不受\* 安全模式是否打开的影响。

; 语法高亮模式的色彩。

; 只要能被接受的东西就能工作。

highlight.string = #DD0000

highlight.comment = #FF8000

highlight.keyword = #007700

highlight.bg = #FFFFFF

highlight.default = #0000BB

highlight.html = #000000

; Misc 杂项

expose\_php = Off

; 决定 PHP 是否标示它装在服务器上的事实（例如：加在它 — PHP— 给Web服务

; 发送的信号上）。

; （我个人的意见，在出现什么power-by的header的时候，把这关掉。）

; 它不会有安全上的威胁, 但它使检查你的服务器上是否安装了PHP成为了可能。

;;;;;;;;;;;;;;;;;;;

; Resource Limits ;

;;;;;;;;;;;;;;;;;;;

max\_execution\_time = 30 ; 每个脚本的最大执行时间, 按秒计

memory\_limit = 8388608 ; 一个脚本最大可使用的内存总量 (这里是8MB)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Error handling and logging ;

; 出错控制和登记 ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; 错误报告是按位的。或者将数字加起来得到想要的错误报告等级。

; E\_ALL - 所有的错误和警告

; E\_ERROR - 致命性运行时错

; E\_WARNING - 运行时警告（非致命性错）

; E\_PARSE - 编译时解析错误

; E\_NOTICE - 运行时提醒(这些经常是是你的代码的bug引起的，

;也可能是有意的行为造成的。(如：基于未初始化的变量自动初始化为一个

;空字符串的事实而使用一个未初始化的变量)

; E\_CORE\_ERROR - 发生于PHP启动时初始化过程中的致命错误

; E\_CORE\_WARNING - 发生于PHP启动时初始化过程中的警告(非致命性错)

; E\_COMPILE\_ERROR - 编译时致命性错

; E\_COMPILE\_WARNING - 编译时警告(非致命性错)

; E\_USER\_ERROR - 用户产生的出错消息

; E\_USER\_WARNING - 用户产生的警告消息

; E\_USER\_NOTICE - 用户产生的提醒消息

; 例子:

; error\_reporting = E\_ALL & ~E\_NOTICE ; 显示所有的错误，除了提醒

; error\_reporting = E\_COMPILE\_ERROR|E\_ERROR|E\_CORE\_ERROR ; 仅显示错误

error\_reporting = E\_ALL & ~E\_NOTICE ; 显示所有的错误，除了提醒

display\_errors = On ; 显示出错误信息(作为输出的一部分)

; 在最终发布的web站点上，强烈建议你关掉这个特性，并使用

; 错误日志代替（参看下面）。

; 在最终发布的web站点继续让 display\_errors 有效可能

; 暴露一些有关安全的信息，例如你的web服务上的文件路径、

; 你的数据库规划或别的信息。

display\_startup\_errors = Off ; 甚至当display\_erroes打开了，发生于PHP的启动的步骤中

; 的错误也不会被显示。

; 强烈建议保持使 display\_startup\_errors 关闭，

; 除了在改错过程中。

log\_errors = Off ; 在日志文件里记录错误（服务器指定的日志，stderr标准错误输出，或error\_log(下面的））

; 正如上面说明的那样，强烈建议你在最终发布的web站点以日志记录错误

; 取代直接错误输出。

track\_errors = Off ; 保存最近一个 错误/警告 消息于变量 $php\_errormsg (boolean)

;error\_prepend\_string = " " ; 于错误信息前输出的字符串

;error\_append\_string = " " ; 于错误信息后输出的字符串

;error\_log = filename ; 记录错误日志于指定文件

;error\_log = syslog ; 记录错误日志于系统日志 syslog (NT 下的事件日志, Windows 95下无效)

warn\_plus\_overloading = Off ; 当将‘ +’ 用于字符串时警告

;;;;;;;;;;;;;;;;;

; Data Handling ;

;;;;;;;;;;;;;;;;;

variables\_order = " EGPCS" ; 这条指示描述了PHP 记录

; GET, POST, Cookie, Environment and Built-in 这些变量的顺序。

; （以 G, P, C, E & S 代表，通常以 EGPCS 或 GPC 的方式引用）。

; 按从左到右记录，新值取代旧值。

register\_globals = On ; 是否将这些 EGPCS 变量注册为全局变量。

; 若你不想让用户数据不在全局范围内混乱的话，你可能想关闭它。

; 这和 track\_vars 连起来用更有意义 — 这样你可以通过

; $HTTP\_\*\_VARS[] 数组访问所有的GPC变量。

register\_argc\_argv = On ; 这条指示告诉 PHP 是否声明 argv和argc 变量

; （注：这里argv为数组,argc为变量数）

; （其中包含用GET方法传来的数据）。

; 若你不想用这些变量，你应当关掉它以提高性能。

track\_vars = On ; 使$HTTP\_\*\_VARS[]数组有效，这里\*在使用时用

; ENV, POST, GET, COOKIE or SERVER替换

post\_max\_size = 8M ; PHP将接受的POST数据最大大小。

gpc\_order = " GPC" ; 这条指示被人反对。用 variables\_order 代替。

; Magic quotes

magic\_quotes\_gpc = On ; 在输入的GET/POST/Cookie数据里使用魔术引用

; （原文就这样，呵呵，所谓magic quotes 应该是指用转义符加在引用性的控制字符上，如 '....）

magic\_quotes\_runtime= Off ; 对运行时产生的数据使用魔术引用，

; 例如：用SQL查询得到的数据，用exec()函数得到的数据，等等

magic\_quotes\_sybase = Off ; 采用 Sybase形式的魔术引用（用 '' 脱出 ' 而不用 '）

; 自动在 PHP 文档之前和之后添加文件

auto\_prepend\_file =

auto\_append\_file =

; 象4.04b4一样，PHP 默认��总是在 “ Content-type:” 头标输出一个字符的编码方式。

; 让输出字符集失效，只要设置为空。

; PHP 的内建默认值是 text/html

default\_mimetype = " text/html"

;default\_charset = " iso-8859-1"

;;;;;;;;;;;;;;;;;;;;;;;;;

; Paths and Directories ;

;;;;;;;;;;;;;;;;;;;;;;;;;

include\_path = ; include 路径设置，UNIX: " /path1:/path2" Windows: " \path1;\path2"

doc\_root = ; php 页面的根路径，仅在非空时有效

user\_dir = ; 告知 php 在使用 /~username 打开脚本时到哪个目录下去找，仅在非空时有效

;upload\_tmp\_dir = ; 存放用HTTP协议上载的文件的临时目录（在没指定时使用系统默认的）

upload\_max\_filesize = 2097152 ; 文件上载默认地限制为2 Meg

extension\_dir = c:\php\ ; 存放可加载的扩充库（模块）的目录

enable\_dl = On ; 是否使dl()有效。

; 在多线程的服务器上 dl()函数\*不能\*很好地工作，

; 例如IIS or Zeus，并在其上默认为禁止

;;;;;;;;;;;;;;;;

; File Uploads ;

;;;;;;;;;;;;;;;;

file\_uploads = On ; 是否允许HTTP方式文件上载

;upload\_tmp\_dir = ; 用于HTTP上载的文件的临时目录（未指定则使用系统默认）

upload\_max\_filesize = 2M ; 上载文件的最大许可大小

; Fopen wrappers ;

;;;;;;;;;;;;;;;;;;

allow\_url\_fopen = On ; 是否允许把URLs当作http:.. 或把文件当作ftp:...

;;;;;;;;;;;;;;;;;;;;;;

; 动态扩展 ;

; Dynamic Extensions ;

;;;;;;;;;;;;;;;;;;;;;;

; 若你希望一个扩展库自动加载，用下面的语法：

; extension=modulename.extension

; 例如，在windows上，

; extension=msql.dll

; or 在UNIX下,

; extension=msql.so

; 注意，这只应当是模块的名字，不需要目录信息放在里面。

; 用上面的 extension\_dir 指示指定扩展库的位置。

;Windows 扩展

;extension=php\_nsmail.dll

extension=php\_calendar.dll

;extension=php\_dbase.dll

;extension=php\_filepro.dll

extension=php\_gd.dll

;extension=php\_dbm.dll

;extension=php\_mssql.dll

;extension=php\_zlib.dll

;extension=php\_filepro.dll

;extension=php\_imap4r2.dll

;extension=php\_ldap.dll

;extension=php\_crypt.dll

;extension=php\_msql2.dll

;extension=php\_odbc.dll

; 注意， MySQL的支持现在是内建的，因此，不需要用它的dll

;;;;;;;;;;;;;;;;;;;

; 模块设定 ;

; Module Settings ;

;;;;;;;;;;;;;;;;;;;

[Syslog]

define\_syslog\_variables = Off ; 是否定义各种的系统日志变量

; 如：$LOG\_PID, $LOG\_CRON, 等等。

; 关掉它是个提高效率的好主意。

; 运行时，你可以调用函数define\_syslog\_variables()，来定义这些变量

[mail function]

SMTP = localhost ;仅用于win32系统

sendmail\_from = me@localhost.com ;仅用于win32系统

;sendmail\_path = ;仅用于unix, 也可支持参数（默认的是'sendmail -t -i'）

[Debugger]

debugger.host = localhost

debugger.port = 7869

debugger.enabled = False

[Logging]

; 这些配置指示用于示例的日志记录机制。

; 看 examples/README.logging 以得到更多的解释

;logging.method = db

;logging.directory = /path/to/log/directory

[Java]

;java.class.path = .\php\_java.jar

;java.home = c:\jdk

;java.library = c:\jdk\jre\bin\hotspot\jvm.dll

;java.library.path = .\

[SQL]

sql.safe\_mode = Off

[ODBC]

;uodbc.default\_db = Not yet implemented

;uodbc.default\_user = Not yet implemented

;uodbc.default\_pw = Not yet implemented

uodbc.allow\_persistent = On ; 允许或禁止 持久连接

uodbc.check\_persistent = On ; 在重用前检查连接是否还可用

uodbc.max\_persistent = -1 ; 持久连接的最大数。-1 代表无限制

uodbc.max\_links = -1 ; 连接的最大数目（持久和非持久）。-1 代表无限制

uodbc.defaultlrl = 4096 ; 控制 LONG 类型的字段。返回变量的字节数，0 代表通过（？）0 means passthru

uodbc.defaultbinmode = 1 ; 控制 二进制数据。0 代表?????Handling of binary data. 0 means passthru, 1 return as is, 2 convert to char

; 见有关 odbc\_binmode 和 odbc\_longreadlen 的文档以得到 uodbc.defaultlrl 和 uodbc.defaultbinmode 的解释。

[MySQL]

mysql.allow\_persistent = On ; 允许或禁止 持久连接

mysql.max\_persistent = -1 ; 持久连接的最大数。-1 代表无限制

mysql.max\_links = -1 ; 连接的最大数目（持久和非持久）。-1 代表无限制

mysql.default\_port = ; mysql\_connect() 使用的默认端口，如不设置，mysql\_connect()

; 将使用变量 $MYSQL\_TCP\_PORT，或在/etc/services 下的mysql-tcp 条目(unix)，

; 或在编译是定义的 MYSQL\_PORT(按这样的顺序)

; Win32环境，将仅检查MYSQL\_PORT。

mysql.default\_socket = ; 用于本地 MySql 连接的默认的套接字名。为空，使用 MYSQL 内建值

mysql.default\_host = ; mysql\_connect() 默认使用的主机（安全模式下无效）

mysql.default\_user = ; mysql\_connect() 默认使用的用户名（安全模式下无效）

mysql.default\_password = ; mysql\_connect() 默认使用的密码（安全模式下无效）

; 注意，在这个文件下保存密码通常是一个\*坏\*主意

; \*任何\*可以使用PHP访问的用户可以运行

; 'echo cfg\_get\_var(" mysql.default\_password" )'来显示那个密码!

; 而且当然地，任何有读该文件权力的用户也能看到那个密码。

[mSQL]

msql.allow\_persistent = On ; 允许或禁止 持久连接

msql.max\_persistent = -1 ; 持久连接的最大数。-1 代表无限制

msql.max\_links = -1 ; 连接的最大数目（持久和非持久）。-1 代表无限制

[PostgresSQL]

pgsql.allow\_persistent = On ; 允许或禁止 持久连接

pgsql.max\_persistent = -1 ; 持久连接的最大数。-1 代表无限制

pgsql.max\_links = -1 ; 连接的最大数目（持久和非持久）。-1 代表无限制

[Sybase]

sybase.allow\_persistent = On ; 允许或禁止 持久连接

sybase.max\_persistent = -1 ; 持久连接的最大数。-1 代表无限制

sybase.max\_links = -1 ; 连接的最大数目（持久和非持久）。-1 代表无限制

;sybase.interface\_file = " /usr/sybase/interfaces"

sybase.min\_error\_severity = 10 ; 显示的错误的最低严重性

sybase.min\_message\_severity = 10 ; 显示的消息的最低重要性

sybase.compatability\_mode = Off ; 与旧版的PHP 3.0 兼容的模式。若打开，这将导致 PHP 自动地

; 把根据结果的 Sybase 类型赋予它们，

; 而不是把它们全当成字符串。

; 这个兼容模式不会永远留着，

; 因此，将你的代码进行需要的修改，

; 并将该项关闭。

[Sybase-CT]

sybct.allow\_persistent = On ; 允许或禁止 持久连接

sybct.max\_persistent = -1 ; 持久连接的最大数。-1 代表无限制

sybct.max\_links = -1 ; 连接的最大数目（持久和非持久）。-1 代表无限制

sybct.min\_server\_severity = 10 ; 显示的错误的最低严重性

sybct.min\_client\_severity = 10 ; 显示的消息的最低重要性

[bcmath]

bcmath.scale = 0 ; 用于所有bcmath函数的10十进制数数字的个数number of decimal digits for all bcmath functions

[browscap]

;browscap = extra/browscap.ini

browscap = C:\WIN\SYSTEM\inetsrv\browscap.ini

[Informix]

ifx.default\_host = ; ifx\_connect() 默认使用的主机（安全模式下无效）

ifx.default\_user = ; ifx\_connect() 默认使用的用户名（安全模式下无效）

ifx.default\_password = ; ifx\_connect() 默认使用的密码（安全模式下无效）

ifx.allow\_persistent = On ; 允许或禁止 持久连接

ifx.max\_persistent = -1 ; 持久连接的最大数。-1 代表无限制

ifx.max\_links = -1 ; 连接的最大数目（持久和非持久）。-1 代表无限制

ifx.textasvarchar = 0 ; 若打开，select 状态符返回一个 ‘ text blob’ 字段的内容，而不是它的id

ifx.byteasvarchar = 0 ; 若打开，select 状态符返回一个 ‘ byte blob’ 字段的内容，而不是它的id

ifx.charasvarchar = 0 ; 追踪从固定长度的字符列里剥离的空格。

; 可能对 Informix SE 用户有效。

ifx.blobinfile = 0 ; 若打开，text和byte blobs 的内容被导出到一个文件

; 而不是保存到内存。

ifx.nullformat = 0 ; NULL（空）被作为空字段返回，除非，这里被设为1。

; 这种情况下（为1），NULL作为字串NULL返回。

[Session]

session.save\_handler = files ; 用于保存/取回数据的控制方式

session.save\_path = C:\win\temp ; 在 save\_handler 设为文件时传给控制器的参数，

; 这是数据文件将保存的路径。

session.use\_cookies = 1 ; 是否使用cookies

session.name = PHPSESSID

; 用在cookie里的session的名字

session.auto\_start = 0 ; 在请求启动时初始化session

session.cookie\_lifetime = 0 ; 为按秒记的cookie的保存时间，

; 或为0时，直到浏览器被重启

session.cookie\_path = / ; cookie的有效路径

session.cookie\_domain = ; cookie的有效域

session.serialize\_handler = php ; 用于连接数据的控制器

; php是 PHP 的标准控制器。

session.gc\_probability = 1 ; 按百分比的'garbage collection（碎片整理）'进程

; 在每次 session 初始化的时候开始的可能性。

session.gc\_maxlifetime = 1440 ; 在这里数字所指的秒数后，保存的数据将被视为

; '碎片(garbage)'并由gc 进程清理掉。

session.referer\_check = ; 检查 HTTP引用以使额外包含于URLs中的ids无效

session.entropy\_length = 0 ; 从文件中读取多少字节

session.entropy\_file = ; 指定这里建立 session id

; session.entropy\_length = 16

; session.entropy\_file = /dev/urandom

session.cache\_limiter = nocache ; 设为{nocache,private,public},以决定 HTTP 的

; 缓存问题

session.cache\_expire = 180 ; 文档在 n 分钟后过时

session.use\_trans\_sid = 1 ; 使用过渡性的 sid 支持，若编译时许可了

; --enable-trans-sid

url\_rewriter.tags = " a=href,area=href,frame=src,input=src,form=fakeentry"

[MSSQL]

;extension=php\_mssql.dll

mssql.allow\_persistent = On ; 允许或禁止 持久连接

mssql.max\_persistent = -1 ; 持久连接的最大数。-1 代表无限制

mssql.max\_links = -1 ; 连接的最大数目（持久和非持久）。-1 代表无限制

mssql.min\_error\_severity = 10 ; 显示的错误的最低严重性

mssql.min\_message\_severity = 10 ; 显示的消息的最低重要性

mssql.compatability\_mode = Off ; 与旧版的PHP 3.0 兼容的模式。

[Assertion]

; ？？？？？

;assert.active = On ; ？assert(expr); active by default

;assert.warning = On ; issue a PHP warning for each failed assertion.

;assert.bail = Off ; don't bail out by default.

;assert.callback = 0 ; user-function to be called if an assertion fails.

;assert.quiet\_eval = 0 ; eval the expression with current error\_reporting(). set to true if you want error\_reporting(0) around the eval().

[Ingres II]

ii.allow\_persistent = On ; 允许或禁止 持久连接

ii.max\_persistent = -1 ; 持久连接的最大数。-1 代表无限制

ii.max\_links = -1 ; 连接的最大数目（持久和非持久）。-1 代表无限制

ii.default\_database = ; 默认 database (format : [node\_id::]dbname[/srv\_class]

ii.default\_user = ; 默认 user

ii.default\_password = ; 默认 password

[Verisign Payflow Pro]

pfpro.defaulthost = " test.signio.com" ; 默认的 Signio 服务器

pfpro.defaultport = 443 ; 连接的默认端口

pfpro.defaulttimeout = 30 ; 按秒计的默认超时时间

; pfpro.proxyaddress = ; 默认的代理的 IP 地址（如果需要）

; pfpro.proxyport = ; 默认的代理的端口

; pfpro.proxylogon = ; 默认的代理的登录（logon 用户名）

; pfpro.proxypassword = ; 默认的代理的密码

[Sockets]

sockets.use\_system\_read = On ; 使用系统的read() 函数替代 php\_read()封装

; Local Variables: （局部变量）

; tab-width: 4

; End

来源：<http://www.linuxidc.com/Linux/2014-01/95999.htm>