简单点，先来实现一个PHP扩展的hello world。

注意，以下所有操作都是基于linux系统(推荐centos和ubuntu, Mac系统应该类似），PHP5.5以上，建议读者使用5.5以上版本测试本文例程，PHP内核是由C语言写成的，所以扩展基本也是用C/C++编写。

话不多说，请睇下面:

一、下载PHP源码到本地目录

解压后进入PHP源码的ext目录，在此目录下有一个名为`ext_skel`的shell脚本文件。接下来我们将使用它来生成我们的扩展的基本骨架。当然，如果你够牛也可以不用它，直接自己编写必要的文件。

二、生成扩展的基本骨架

在ext目录执行命令:

```
./ext_skel --extname=foobar
```

如果在此目录没有写权限请自觉加sudo。命令执行完毕后输出:

```
Creating directory foobarCreating basic files: config.m4
config.w32 .svnignore foobar.c php_foobar.h CREDITS
EXPERIMENTAL tests/001.phpt foobar.php [done].To use your
new extension, you will have toexecute the following
steps:1.  $ cd ..2.  $ vi ext/foobar/config.m43.  $
./buildconf4.  $ ./configure --[with|enable]-foobar5.  $
make6.  $ ./sapi/cli/php -f ext/foobar/foobar.php7.  $ vi
ext/foobar/foobar.c8.  $ makeRepeat steps 3-6until you are
satisfied with ext/foobar/config.m4 andstep 6 confirms that
your moduleiscompiledinto PHP. Then, start writingcode
andrepeat the last two steps as often as necessary.
```

人品好的话将看到上面的输出，这表示已成功生成名为`foobar`的PHP扩展的基本骨架，在当前目录生成了一个`foobar`的文件夹，我们扩展的所有代码都将放在此目录下（使用了第三方的库的扩展另当别论）。先别着急弄懂上面提示的内容，以后你会知道的，就像小时候妈妈经常跟你说：等你长大了就懂了！

三、编辑config.m4文件

`ls`扩展目录foobar,发现里面有几个文件:

```
config.m4  config.w32  CREDITS  EXPERIMENTAL  foobar.c
foobar.php  php_foobar.h  tests
```

用VIM打开扩展目录下的config.m4文件，找到下面几行：

```
16 dnl PHP_ARG_ENABLE(foobar, whether to enable foobar
support,17 dnl Make sure that the comment is aligned:18 dnl
[  --enable-foobar           Enable foobar support])
```

前面的数字是它所在的行数(下同)，不是文件内容，去掉16和18行前面`dnl`字符，`dnl`注释开始，我们要把这两行的注释符号去掉，17行不用管。`wq`保存文件。

config.m4文件其实有很多内容，初始入门教程就不仔细说明各部分内容，因为实在我也不太懂！它的作用是配置扩展的行为，比如说明扩展编译选项，是否使用第三方库，扩展的源码组成等等。

四、编辑php_foobar.h文件：声明一个函数

`php_foobar.h`是一个C头文件，我们需要在这个头文件里声明一个方法

vim编辑`php_foobar.h`文件,找到下面这一行：

```
47 PHP_FUNCTION(confirm_foobar_compiled);   /* For testing,
remove later. */
```

这上ext_skel工具生成扩展骨架的时候自动声明的一个函数，仅用于测试，你可以去掉，也可以保留！在这一行的下面添加一行:

```
48 PHP_FUNCTION(halo);
```

这就声明了一个名为halo的PHP空间的函数，在PHP的代码里就可以像普通函数一样调用它。当然，目前到这一步还不行，因为这里只是声明，还没有定义它的行为。我们将在foobar.c文件中编写它的函数体。

四、编辑foobar.c文件：定义函数体

foobar.c是扩展主要实现的地方，找到下面几行:

```
41 const zend_function_entry foobar_functions[] = {42
PHP_FE(confirm_foobar_compiled, NULL)        /* For testing,
remove later. */43      PHP_FE_END  /* Must be the last line
in foobar_functions[] */44 };
```

在42行下面添加PHP_FE(halo, NULL)。注意不要添加任何分号。如下:

```
41 const zend_function_entry foobar_functions[] = {42
PHP_FE(confirm_foobar_compiled, NULL)        /* For testing,
remove later. */43      PHP_FE(halo, NULL)44      PHP_FE_END
/* Must be the last line in foobar_functions[] */45 };
```

这一步是向PHP空间注册一个函数，名字就是刚才在php_foobar.h文件声明的halo。

接下来是真正编写halo函数实现的时候。

在文件末尾添加以下代码：

```
169 PHP_FUNCTION(halo){170     php_printf("hello
world!");171 }
```

从代码看出:halo函数只是打印一串字符串hello world，不做其他任何事情。

foobar.c文件内容很多，每个代码段都有相应的注释说明，仔细研究一下，应该还是大概能懂是什么意思！如果看不明白也没关系，还是妈妈那句话：等你长大(看多)了就懂了

OK！编码完毕，下面就是把扩展编译进PHP，供PHP代码调用！

五、编译安装扩展

扩展编译分动态编译和静态编译两种方法！今天我们先讨论动态编译。有兴趣的同学可以自行研究一下静态编译是什么鬼！

在扩展目录中执行phpize命令。一定要在扩展的目录执行才有效，否则将得到一个错误提示。

如果提示没有找到命令，请检查系统没有安装php-dev工具集，如果是源码编译安装的PHP，一般在php的bin目录下面，如果通过yum或apt安装的PHP请确认是否安装过php-dev或者php-devel,安装过的话应该直接就能运行phpize命令。也可通过find / -name phpize命令来找到phpize的路径，然后带路径执行，如果系统安装了多个版本的PHP，最好是指定路径的phpize来指定使用的PHP版本！还找不到的话就GOOGLE一下吧！

phpize命令的正常输出如下:

```
Configuringfor:PHPApiVersion:20121113ZendModuleApiNo:2012121
2ZendExtensionApiNo:220121212
```

它表明的是当前使用的PHP内核的版本。

执行完`phpize`命令，细心的你会发现扩展目录下多出了好多文件，我的意见是不用管这些文件是干嘛用的，当然有兴趣也可以研究一下！

下一步就是`configure`,详细命令如下：

```
./configure --enable-foobar --with-config-path=/usr/local/php/bin/php-config
```

`configure`需要两个选项：`--enable-foobar`表示启用这个扩展; `--with-config-path=/usr/local/php/bin/php-config`, 指定了`php-config`的路径，一般源码编译安装的PHP和多版本环境都需要指定这个选项，`apt`和`yum`安装的都是在默认路径，可以不特别指定。

`configure`之后又多了好多文件，再一次无视它们吧！命令输出好长的一陀东西。

`configure`完之后就是`make`了。

什么？`command not found`?先安装`gcc`和`make`吧! 方法请Google!

顺利`make`完之后，会在扩展的目录下的`modules`子目录多了一个`foobar.so`的文件，

它就是我们刚才编写的扩展的最终产物。聪明的你一定已经想到：这TM不就是个动态库吗？而我只能说：你说对了！动态编译产生的是动态库文件。

OK，扩展编译完了，需要在PHP中使用扩展，复制`foobar.so`的完整路径，vim打开PHP的配置文件`php.ini`，在文件的末尾加入以下内容:

```
extension=/root/php-5.5.38/ext/foobar/modules/foobar.so #这是在我的系统foobar.so路径
```

保存退出，重启一下php-fpm（或apache,nginx什么的，如果不确定，就都重启

吧！请不要在生产环境瞎搞）

有些人就忍不住要问一下了：为什么有些扩展在`php.ini`的配置中不用带路径？其实我们的扩展一样也是可以的。在`make`命令之后多执行一步：`make install`，如果不是`root`权限，请自觉加`sudo`或切到`root`用户下执行。扩展就会安装到相应PHP版本的默认扩展加载路径！然后在`php.ini`的配置中就只要简单写上扩展名加`so`即可.

```
extension=foobar.so
```

至此，一个简单的PHP扩展就完成了。是不是有点小激动?

下面验证一番，在WEB目录新建一个php文件，如`info.php`,写入如下代码：

```
phpinfo();
```

地球人都知道这个函数调用是做什么的吧?

在浏览器执行这个文件!看到以下输出就说明扩展安装成功了：

foobar

foobar support	enabled

直接修改`info.php`文件吧，调用我们刚才在扩展中定义的函数：

```
phphalo();
```

刷新浏览器，如果人品不太坏的话应该就能看到以下的输出：

```
hello world!
```

BingGo!

如果你觉得这文章不错，请给我点个赞吧^_^!

六、扩展阅读

- PHP扩展开发及内核应用
- TIPI项目
- longmon@github