1.2 PHP的启动与终止

PHP程序的启动可以看作有两个概念上的启动，终止也有两个概念上的终止。
其中一个是PHP作为Apache(拿它举例，板砖勿扔)的一个模块的启动与终止，
这次启动php会初始化一些必要数据，比如与宿主Apache有关的，并且这些数据是常驻内存的！
终止与之相对。
还有一个概念上的启动就是当Apache分配一个页面请求过来的时候，PHP会有一次启动与终止，这也是我们最常讨论的一种。

现在我们主要来看一个PHP扩展的生命旅程是怎样走完这四个过程的。

在最初的初始化时候，就是PHP随着Apache的启动而诞生在内存里的时候，
它会把自己所有已加载扩展的MINIT方法(全称Module Initialization，是由每个模块自己定义的函数。)都执行一遍。
在这个时间里，扩展可以定义一些自己的常量、类、资源等所有会被用户端的PHP脚本用到的东西。
但你要记住，这里定义的东东都会随着Apache常驻内存，可以被所有请求使用，直到Apache卸载掉PHP模块！

内核中预置了PHP_MINIT_FUNCTION宏函数，来帮助我们实现这个功能：

```
//抛弃作者那个例子，书才看两页整那样的例子太复杂了！
//walu是我扩展的名称
int time_of_minit;//在MINIT()中初始化，在每次页面请求中输出，看看是否变化
PHP_MINIT_FUNCTION(walu)
{
    time_of_minit=time(NULL);//我们在MINIT启动中对他初始化
    return SUCCESS;//返回SUCCESS代表正常，返回FALIURE就不会加载这个扩展了。
}
```

当一个页面请求到来时候，PHP会迅速的开辟一个新的环境，并重新扫描自己的各个扩展，
遍历执行它们各自的RINIT方法(俗称Request Initialization)，
这时候一个扩展可能会初始化在本次请求中会使用到的变量等，
还会初始化等会儿用户端（即PHP脚本）中的变量之类的，内核预置了PHP_RINIT_FUNCTION()这个宏函数来帮我们实现这个功能：

```
int time_of_rinit;//在RINIT里初始化，看看每次页面请求的时候是否变化。
PHP_RINIT_FUNCTION(walu)
{
    time_of_rinit=time(NULL);
    return SUCCESS;
}
```

好了，现在这个页面请求执行的差不多了，可能是顺利的走到了自己文件的最后，
也可能是出师未捷，半道被用户给die或者exit了，
这时候PHP便会启动回收程序，收拾这个请求留下的烂摊子。
它这次会执行所有已加载扩展的RSHUTDOWN（俗称Request Shutdown）方法，
这时候扩展可以抓紧利用内核中的变量表之类的做一些事情，
因为一旦PHP把所有扩展的RSHUTDOWN方法执行完，
便会释放掉这次请求使用过的所有东西，
包括变量表的所有变量、所有在这次请求中申请的内存等等。

内核预置了PHP_RSHUTDOWN_FUNCTION宏函数来帮助我们实现这个功能

```
PHP_RSHUTDOWN_FUNCTION(walu)
{
    FILE *fp=fopen("time_rshutdown.txt","a+");
    fprintf(fp,"%ld\n",time(NULL));//让我们看看是不是每次请求结束都会在这个文件里追加数据
    fclose(fp);
    return SUCCESS;
}
```

前面该启动的也启动了，该结束的也结束了，现在该Apache老人家歇歇的时候，当Apache通知PHP自己要Stop的时候，PHP便进入MSHUTDOWN（俗称Module Shutdown）阶段。这时候PHP便会给所有扩展下最后通牒，如果哪个扩展还有未了的心愿，就放在自己MSHUTDOWN方法里，这可是最后的机会了，一旦PHP把扩展的MSHUTDOWN执行完，便会进入自毁程序，这里一定要把自己擅自申请的内存给释放掉，否则就杯具了。

内核中预置了PHP_MSHUTDOWN_FUNCTION宏函数来帮助我们实现这个功能：

```
PHP_MSHUTDOWN_FUNCTION(walu)
{
    FILE *fp=fopen("time_mshutdown.txt","a+");
    fprintf(fp,"%ld\n",time(NULL));
    return SUCCESS;
}
```

这四个宏都是在walu.c里完成最终实现的，而他们的则是在/main/php.h里被定义的(其实也是调用的别的宏，本节最后我把这几个宏给展开了，供有需要的人查看)。

好了，现在我们本节内容说完了，下面我们把所有的代码合在一起，并预测一下应该出现的结果：

```
//这些代码都在walu.c里面，不在.h里

int time_of_minit;//在MINIT中初始化，在每次页面请求中输出，看看是否变化
PHP_MINIT_FUNCTION(walu)
{
    time_of_minit=time(NULL);//我们在MINIT启动中对他初始化
    return SUCCESS;
}

int time_of_rinit;//在RINIT里初始化，看看每次页面请求的时候是否变化。
PHP_RINIT_FUNCTION(walu)
{
    time_of_rinit=time(NULL);
    return SUCCESS;
}

PHP_RSHUTDOWN_FUNCTION(walu)
{
    FILE *fp=fopen("/cnan/www/erzha/time_rshutdown.txt","a+");//请确保文件可写，否则apache会莫名崩
溃
    fprintf(fp,"%d\n",time(NULL));//让我们看看是不是每次请求结束都会在这个文件里追加数据
    fclose(fp);
    return SUCCESS;
}

PHP_MSHUTDOWN_FUNCTION(walu)
{
    FILE *fp=fopen("/cnan/www/erzha/time_mshutdown.txt","a+");//请确保文件可写，否则apache会莫名崩
溃
    fprintf(fp,"%d\n",time(NULL));
    return SUCCESS;
}

//我们在页面里输出time_of_minit和time_of_rinit的值
PHP_FUNCTION(walu_test)
{
    php_printf("%d&lt;br /&gt;",time_of_minit);
    php_printf("%d&lt;br /&gt;",time_of_rinit);
    return;
}
```

- time_of_minit的值每次请求都不变。
- time_of_rinit的值每次请求都改变。
- 每次页面请求结束都会往time_rshutdown.txt中写入数据。
- 只有在apache结束后time_mshutdown.txt才写入有数据。

> 多谢 闸北陆小洪 指出的有关time_of_rinit的笔误。

上面便是PHP中典型的启动-终止模型，实际情况可能因为模式不同而有所变化，
到底PHP的启动-终止会有多少种不同变化方式，请看下一节。

links

- 目录
- 上一节: 让我们从SAPI开始
    - 下一节: PHP的生命周期