Swoole源码学习记录

我接触PHP的时间不长，最开始只认为PHP是用来做网站开发，是一个比JSP要简单的语言。后来，因为工作需要，一位学长建议我使用Ngnix + PHP 搭建应用服务器，并建议使用现有的框架。一番搜索之下，我意外发现了Swoole <http://www.swoole.com/>

接下来的半年里，我一直使用Swoole扩展作为我的服务器核心。Swoole稳定而高效的性能以及优秀的架构设计使我的开发变得简单、高效。因此，我希望能够更加深入的了解Swoole的核心，学习它的架构设计，也能锻炼自己的C语言能力。

因此，我将不定期更细这一系列，将我在阅读、理解Swoole源码过程中的心得体会记录下来，也希望我的记录能帮助那些同样希望理解Swoole源码的人。本人能力有限，C语言能力也只能说刚刚入门，难免会有误解的地方，希望大家能及时指正。

## Swoole版本：1.7.4-stable

Swoole没有采用多线程模型而是使用的多进程模型，在一定程度上减少了访问数据时加锁解锁的开销，但同时也引入了新的需求：共享内存。

直接上源码吧，在Swoole中，对于ShareMemory的声明在swoole.h 464行-478行，如下：

#define SW\_SHM\_MMAP\_FILE\_LEN 64

typedef struct \_swShareMemory\_mmap

{

int size;

char mapfile[SW\_SHM\_MMAP\_FILE\_LEN];

int tmpfd;

int key;

int shmid;

void \*mem;

} swShareMemory;

void \*swShareMemory\_mmap\_create(swShareMemory \*object, int size, char \*mapfile);

int swShareMemory\_mmap\_free(swShareMemory \*object);

void \*swShareMemory\_sysv\_create(swShareMemory \*object, int size, int key);

int swShareMemory\_sysv\_free(swShareMemory \*object, int rm);

这里，Rango声明了一个结构体swShareMemory表示共享内存的结构。在内存中，一个完整的swShareMemory存储形式如下：

swShareMemory

mem

其中，首部swShareMemory存放了共享内存的相关信息，尾部的void \*mem指针指向共享内存的起始地址。int size代表共享内存的大小(不包括swShareMemory结构体大小)，char mapfile[]代表共享内存使用的内存映射文件的文件名，int tmpfd为内存映射文件的描述符，int key代表使用Linux自带的shm系列函数创建的共享内存的key值，int shmid为shm系列函数创建的共享内存的id（类似于fd）。

接下来看对应的四个处理函数，从名字上可以区分，mmap代表使用内存映射文件的共享内存，sysv代表使用shm系列函数的共享内存。这四个函数的实现写在ShareMemory.c文件中。这里重点解析 swShareMemory\_mmap\_create 方法和swShareMemory\_sysv\_create 方法。

下面贴上swShareMemory\_mmap\_create的核心代码：

#ifdef MAP\_ANONYMOUS

flag |= MAP\_ANONYMOUS;

#else

if(mapfile == NULL)

{

mapfile = "/dev/zero";

}

if((tmpfd = open(mapfile, O\_RDWR)) < 0)

{

return NULL;

}

strncpy(object->mapfile, mapfile, SW\_SHM\_MMAP\_FILE\_LEN);

object->tmpfd = tmpfd;

#endif

mem = mmap(NULL, size, PROT\_READ | PROT\_WRITE, flag, tmpfd, 0);

先解释一下 MAP\_ANONYMOUS 宏，该宏定义在 mman-linux.h 内，定义如下：

# define MAP\_ANONYMOUS 0x20 /\* Don't use a file. \*/

使用这个宏代表使用mmap生成的映射区不与任何文件关联。我的理解是：此处创建一个仅存在与内存中虚拟文件，用来存放需要共享的内容。

其次解释 /dev/zero 。熟悉Linux的应该都知道 /dev/zero 和 /de/null , 这是Linux系统中的两个特殊文件，/dev/zero的特性是，所有写入该文件的数据都会消失，如果从该文件中读取内容，只能读取到连续的‘\0’。

源码解释：

如果定义了 MAP\_ANONYMOUS 宏，则添加进flag中。否则，判定mapfile是否为空，如果为空，则指定mapfile为/dev/zero ，然后打开文件，将描述符赋值给tmpfd。最后，通过mmap打开大小为size的映射内存，指定内存可读可写，并将内存地址赋值给mem。

swShareMemory\_sysv\_create的核心源码：

if (key == 0)

{

key = IPC\_PRIVATE;

}

if ((shmid = shmget(key, size, SHM\_R | SHM\_W | IPC\_CREAT)) < 0)

{

swWarn("shmget() failed. Error: %s[%d]", strerror(errno), errno);

return NULL;

}

if ((mem = shmat(shmid, NULL, 0)) < 0)

{

swWarn("shmat() failed. Error: %s[%d]", strerror(errno), errno);

return NULL;

}

else

{

object->key = key;

object->shmid = shmid;

object->size = size;

object->mem = mem;

return mem;

}

源码解释：

如果key 为0，则表示需要创建新的共享内存，则将key赋值为IPC\_PRIVATE 宏。然后，调用shmget方法，如果key为IPC\_PRIVATE，则创建一个大小为size的可读可写的共享内存，并获取到共享内存标识符shmid，否则获取到IPC值为key的共享内存标识符shmid。接着，调用shmat方法获取到shmid对应的共享内存首地址，并赋值给mem。

两个free方法则是分别调用munmap方法和shmctl方法释放内存，不再贴上源码。

时间缘故，这次分析就到这里，下一篇将介绍Swoole的三种类型的MemoryPool的具体实现。