共享内存

概述

■ 共享内存允许两个或者多个进程共享物理内存的同一块区域（通常被称为段）。由于一个共享内存段会称为一个进程用户空间的一部分，因此这种 IPC 机制无需内核介入。所有需要做的就是让一个进程将数据复制进共享内存中，并且这部分数据会对其他所有共享同一个段的进程可用。
■ 与管道等要求发送进程将数据从用户空间的缓冲区复制进内核内存和接收进程将数据从内核内存复制进用户空间的缓冲区的做法相比，这种 IPC 技术的速度更快。

共享内存操作步骤

■ 调用 shmget() 创建一个新共享内存段或取得一个既有共享内存段的标识符（即由其他进程创建的共享内存段）。这个调用将返回后续调用中需要用到的共享内存标识符。 --> 创建共享内存或者获取共享内存首地址
■ 使用 shmat() 来附上共享内存段，即使该段成为调用进程的虚拟内存的一部分。 ---> 将共享内存附着在当前进程
■ 此刻在程序中可以像对待其他可用内存那样对待这个共享内存段。为引用这块共享内存，程序需要使用由 shmat() 调用返回的 addr 值，它是一个指向进程的虚拟地址空间中该共享内存段的起点的指针。
■ 调用 shmdt() 来分离共享内存段。在这个调用之后，进程就无法再引用这块共享内存了。这一步是可选的，并且在进程终止时会自动完成这一步。
■ 调用 shmctl() 来删除共享内存段。只有当当前所有附加内存段的进程都与之分离之后内存段才会销毁。只有一个进程需要执行这一步。

相关函数及其详解

```
■ int shmget(key_t key, size_t size, int shmflg);
■ void *shmat(int shmid, const void *shmaddr, int shmflg);
■ int shmdt(const void *shmaddr);
■ int shmctl(int shmid, int cmd, struct shmid_ds *buf);
■ key_t ftok(const char *pathname, int proj_id);
```

```
#include <sys/ipc.h>
#include <sys/shm.h>

int shmget(key_t key, size_t size, int shmflg);
    - 功能：创建一个新的共享内存段，或者获取一个既有的共享内存段的标识。
        新创建的内存段中的数据都会被初始化为0
    - 参数：
        - key : key_t类型是一个整型，通过这个找到或者创建一个共享内存。
                一般使用16进制表示，非0值，也可以是10进制，它会默认转换成16进制
        - size: 共享内存的大小
        - shmflg: 属性
            - 访问权限
            - 附加属性：创建/判断共享内存是不是存在
                - 创建：IPC_CREAT
                - 判断共享内存是否存在： IPC_EXCL , 需要和IPC_CREAT一起使用
                    IPC_CREAT | IPC_EXCL | 0664
        - 返回值：
            失败：-1 并设置错误号
            成功：>0 返回共享内存的引用的ID，后面操作共享内存都是通过这个值。
```

```
void *shmat(int shmid, const void *shmaddr, int shmflg);
    - 功能：和当前的进程进行关联
    - 参数：
        - shmid : 共享内存的标识（ID）,由shmget返回值获取
        - shmaddr: 申请的共享内存的起始地址，指定NULL，内核指定
        - shmflg : 对共享内存的操作
            - 读 ： SHM_RDONLY, 必须要有读权限
            - 读写： 0
    - 返回值：
        成功：返回共享内存的首（起始）地址。  失败(void *) -1


int shmdt(const void *shmaddr);
    - 功能：解除当前进程和共享内存的关联
    - 参数：
        shmaddr：共享内存的首地址
    - 返回值：成功 0， 失败 -1

int shmctl(int shmid, int cmd, struct shmid_ds *buf);
    - 功能：对共享内存进行操作。删除共享内存，共享内存要删除才会消失，创建共享内存的进行被销毁了对共享内存是没有任何影响。
    - 参数：
        - shmid: 共享内存的ID
        - cmd : 要做的操作
            - IPC_STAT : 获取共享内存的当前的状态
            - IPC_SET : 设置共享内存的状态
            - IPC_RMID: 标记共享内存被销毁,当共享内存没有进程与之关联才会被销毁
        - buf：需要设置或者获取的共享内存的属性信息
            - IPC_STAT : buf存储数据
            - IPC_SET : buf中需要初始化数据，设置到内核中
            - IPC_RMID : 没有用，NULL

key_t ftok(const char *pathname, int proj_id);
    - 功能：根据指定的路径名，和int值，生成一个共享内存的key
    - 参数：
        - pathname:指定一个存在的路径
            /home/nowcoder/Linux/a.txt
            /
        - proj_id: int类型的值，但是这系统调用只会使用其中的1个字节
                   范围 ： 0-255  一般指定一个字符 'a'
```

关于共享内存的相关问题

```
问题1：操作系统如何知道一块共享内存被多少个进程关联？
    - 共享内存维护了一个结构体struct shmid_ds 这个结构体中有一个成员 shm_nattch
    - shm_nattach 记录了关联的进程个数

问题2：可不可以对共享内存进行多次删除 shmctl
    - 可以的
    - 因为shmctl 标记删除共享内存，不是直接删除
    - 什么时候真正删除呢?
        当和共享内存关联的进程数为0的时候，就真正被删除
    - 当共享内存的key为0的时候，表示共享内存被标记删除了
        如果一个进程和共享内存取消关联，那么这个进程就不能继续操作这个共享内存。也不能进行关联。
```

共享内存和内存映射区别

1.共享内存可以直接创建，内存映射需要磁盘文件（匿名映射除外）
2.共享内存效果更高
3.内存
        所有的进程操作的是同一块共享内存。
        内存映射，每个进程在自己的虚拟地址空间中有一个独立的内存。
4.数据安全
        - 进程突然退出
            共享内存还存在
            内存映射区消失
        - 运行进程的电脑死机，宕机了
            数据存在在共享内存中，没有了
            内存映射区的数据 ，由于磁盘文件中的数据还在，所以内存映射区的数据还存在。

  5.生命周期
        - 内存映射区：进程退出，内存映射区销毁
        - 共享内存：进程退出，共享内存还在，标记删除（所有的关联的进程数为0），或者关机
            如果一个进程退出，会自动和共享内存进行取消关联。

操作共享内存的相关命令

■ ipcs 用法
 ipcs -a // 打印当前系统中所有的进程间通信方式的信息
 ipcs -m // 打印出使用共享内存进行进程间通信的信息
 ipcs -q // 打印出使用消息队列进行进程间通信的信息
 ipcs -s // 打印出使用信号进行进程间通信的信息
■ ipcrm 用法
 ipcrm -M shmkey // 移除用shmkey创建的共享内存段
 ipcrm -m shmid // 移除用shmid标识的共享内存段
 ipcrm -Q msgkey // 移除用msqkey创建的消息队列
 ipcrm -q msqid // 移除用msqid标识的消息队列
 ipcrm -S semkey // 移除用semkey创建的信号
 ipcrm -s semid // 移除用semid标识的信号

示例

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <string.h>

int main() {

    // 1.创建一个共享内存
    int shmid = shmget(100, 4096, IPC_CREAT|0664);
    printf("shmid : %d\n", shmid);

    // 2.和当前进程进行关联
    void * ptr = shmat(shmid, NULL, 0);

    char * str = "helloworld";

    // 3.写数据
```

```
    memcpy(ptr, str, strlen(str) + 1);

    printf("按任意键继续\n");
    getchar();

    // 4.解除关联
    shmdt(ptr);

    // 5.删除共享内存
    shmctl(shmid, IPC_RMID, NULL);

    return 0;
}
```

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <string.h>

int main() {

    // 1.获取一个共享内存
    int shmid = shmget(100, 0, IPC_CREAT);
    printf("shmid : %d\n", shmid);

    // 2.和当前进程进行关联
    void * ptr = shmat(shmid, NULL, 0);

    // 3.读数据
    printf("%s\n", (char *)ptr);

    printf("按任意键继续\n");
    getchar();

    // 4.解除关联
    shmdt(ptr);

    // 5.删除共享内存
    shmctl(shmid, IPC_RMID, NULL);

    return 0;
}
```