学号:031602523 姓名:刘宏岩 学院:数计学院 专业:计算机类 班级:05班

《Linux 操作系统设计实践》实验二:进程通信

实验环境：Ubuntu16.04

实验内容

- 本次实验，我先学习了共享内存的概念，然后学习了相关的函数，整理成笔记：

1. shmget函数

- 该函数用来创建虚拟内存。

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

int shmget(key_t key, size_t size, int shmflg);
```

- 第一个参数，与信号量的semget函数一样，需要提供一个参数key（非0整数），它有效地为共享内存段命名。
- 第二个参数，size以字节为单位指定需要共享的内存容量。
- 第三个参数，shmflg是权限标志，它的作用与open函数的mode参数一样，如果要想在key标识的共享内存不存在时，创建它的话，可以与IPC_CREAT做或操作。共享内存的权限标志与文件的读写权限一样，举例来说。
- shmget函数成功时返回一个与key相关的共享内存标识符（非负整数），用于后续的共享内存函数。调用失败返回-1。

2. shmat函数

- 第一次创建完共享内存时，它还不能被任何进程访问，shmat函数的作用就是用来启动对该共享内存的访问，并把共享内存连接到当前进程的地址空间。它的原型如下：

```
void *shmat(int shm_id, const void *shm_addr, int shmflg);
```

- 第一个参数，shm_id是由shmget函数返回的共享内存标识。
- 第二个参数，shm_addr指定共享内存连接到当前进程中的地址位置，通常为空，表示让系统来选择共享内存的地址。
- 第三个参数，shm_flg是一组标志位，通常为0。
- 调用成功时返回一个指向共享内存第一个字节的指针，如果调用失败返回-1.

3. shmdt函数

- 该函数用于将共享内存从当前进程中分离。注意，将共享内存分离并不是删除它，只是使该共享内存对当前进程不再可用。它的原型如下：

```
int shmdt(const void *shmaddr);
```

- 参数shmaddr是shmat函数返回的地址指针，调用成功时返回0，失败时返回-1。

4. shmctl函数

- 与信号量的semctl函数一样，用来控制共享内存，它的原型如下：

```
    int shmctl(int shm_id, int command, struct shmid_ds *buf);
```

- 第一个参数，shm_id是shmget函数返回的共享内存标识符。
- 第二个参数，command是要采取的操作，它可以取下面的三个值 ：

```
IPC_STAT：把shmid_ds结构中的数据设置为共享内存的当前关联值，即用共享内存的当

IPC_SET：如果进程有足够的权限，就把共享内存的当前关联值设置为shmid_ds结构中给

IPC_RMID：删除共享内存段
```

- 第三个参数，buf是一个结构体指针，它指向共享内存模式和访问权限的结构。shmid_ds结构至少包括以下成员：

```
struct shmid_ds
{
    uid_t shm_perm.uid;
    uid_t shm_perm.gid;
    mode_t shm_perm.mode;
};
```

我的代码

- 进程1

```
// writeshm.c

#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>

int main()
{
    // 生成一个key
    key_t key = ftok("./", 66);

    // 创建共享内存，返回一个id
    int shmid = shmget(key, 8, IPC_CREAT|0666|IPC_EXCL);
    if(-1 == shmid)
    {
        perror("shmget failed");
        exit(1);
    }

    // 映射共享内存，得到虚拟地址
    void *p = shmat(shmid, 0, 0);
    if((void*)-1 == p)
    {
        perror("shmat failed");
        exit(2);
    }

    // 写共享内存
    int *pp = p;
    *pp = 0x12345678;
    *(pp + 1) = 0xffffffff;
```

```
    // 解除映射
    if(-1 == shmdt(p))
    {
        perror("shmdt failed");
        exit(3);
    }
    printf("解除映射成功，点击回车销毁共享内存\n");
    getchar();

    // 销毁共享内存
    if(-1 == shmctl(shmid, IPC_RMID, NULL))
    {
        perror("shmctl failed");
        exit(4);
    }

    return 0;
}
```

- 进程2

```
// readshm.c

#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>

int main()
{
    // 生成一个key
    key_t key = ftok("./", 66);

    // 获取共享内存，返回一个id
    int shmid = shmget(key, 0, 0);
    if(-1 == shmid)
    {
        perror("shmget failed");
        exit(1);
    }

    // 映射共享内存，得到虚拟地址
    void *p = shmat(shmid, 0, 0);
    if((void*)-1 == p)
    {
```

```
        perror("shmat failed");
        exit(2);
    }

    // 读共享内存
    int x = *(int *)p;
    int y = *((int *)p + 1);
    printf("从共享内存中都取了：0x%x 和 0x%x \n", x, y);

    // 解除映射
    if(-1 == shmdt(p))
    {
        perror("shmdt failed");
        exit(3);
    }

    return 0;
}
```

- 运行结果