



“操作系统原理与实践”实验报告

[地址映射与共享](#)

我感觉本次实验的精华全部都在前面跟踪地址映射过程。实验文档写的很详细了，大家一定要跟着做一下。本次跟上次的套路几乎一样。刚好还要用到上次的信号量，幸亏我没有删除本地的。先进行ubuntu本地测试。producer.c

实验数据

学习时间	210分钟
操作时间	57分钟
按键次数	1143次
实验次数	5次
报告字数	6613字
是否完成	完成

评分

未评分

下一篇

篇

相关报告

- 操作系统原理与实践: 熟悉实验环境 实验报告
- 操作系统原理与实践: 熟悉实验环境 实验报告
- 操作系统原理与实践: 基于内核栈切换的进程切换 实验报告
- 操作系统原理与实践: 熟悉实验环境 实验报告
- 操作系统原理与实践: 信号量的实现和应用 实验报告

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <semaphore.h>
#include <sys/wait.h>
#include <sys/shm.h>
#include <sys/ipc.h>
#include <fcntl.h>

#define PRODUCE_NUM 800
#define BUFF_SIZE 10
#define SHM_KEY 12345

void die(const char *p)
{
    puts(p);
    exit(1);
}

void producer(sem_t * Empty, sem_t * Full, sem_t * Mutex, int shm_id)
{
    int *p;
    int i, location=0;
    if((p = (int *)shmat(shm_id, NULL, 0)) == -1)
        die("link error!");
    for(i=0; i<PRODUCE_NUM; i++)
    {
        sem_wait(Empty);
        sem_wait(Mutex);
        p[location] = i;
        sem_post(Full);
        sem_post(Mutex);
        location = (location+1)%10;
    }
    if(shmdt(p) == -1)
        die("reless shm error");
    return;
}

int main(int argn, char* argc[])
{
    sem_t *semEmpty, *semFull, *semMutex;
    int i;
    int shm_id;

    sem_unlink("Mutex");
    sem_unlink("Full");
    sem_unlink("Empty");

    semEmpty = sem_open("Empty", O_CREAT|O_EXCL, 0600, BUFF_SIZE);
    semFull = sem_open("Full", O_CREAT|O_EXCL, 0600, 0);
    semMutex = sem_open("Mutex", O_CREAT|O_EXCL, 0600, 1);

    if((shm_id = shmget(SHM_KEY, BUFF_SIZE*sizeof(int), IPC_CREAT|0666)) == -1)
        die("shmget failed!");
    printf("create a shared memory segment sucessfully:%d\n", shm_id);
    producer(semEmpty, semFull, semMutex, shm_id);
    return 0;
}

```

consumer.c

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <semaphore.h>
#include <sys/wait.h>
#include <sys/shm.h>
#include <sys/ipc.h>
#include <fcntl.h>

#define PRODUCE_NUM 800
#define BUFF_SIZE 10
#define SHM_KEY 12345

void die(const char *p)
{
    puts(p);
    exit(1);
}

int main(int argn, char* argc[])
{
    sem_t *semEmpty,*semFull,*semMutex;
    int shm_id,location = 0, pid;
    int *p,i;
    semEmpty = sem_open("Empty",0,0600,BUFF_SIZE);
    semFull = sem_open("Full", 0,0600, 0);
    semMutex = sem_open("Mutex",0,0600, 1);
    if((shm_id = shmget(SHM_KEY, BUFF_SIZE*sizeof(int), IPC_CREAT|0666)) == -1)
        die("shmget failed!");
    printf("%d",shm_id);
    if((p = (int *)shmat(shm_id, NULL, 0)) == -1)
        die("link error!");
    for(i=0; i<PRODUCE_NUM; i++)
    {
        sem_wait(semFull);
        sem_wait(semMutex);
        printf("%d:%d\n", pid, p[location]);
        sem_post(semEmpty);
        sem_post(semMutex);
        location = (location+1)%10;
    }
    if(shmdt(p) == -1)
        die("reless shm error");
    if(shmctl(shm_id,IPC_RMID,NULL) == -1)
        die("remove shm error");
    printf("close succeed\n");
    sem_unlink("Mutex");
    sem_unlink("Full");
    sem_unlink("Empty");
    return 0;
}

```

```

neuron@neuron-virtual-machine: ~/桌面
neuron@neuron-virtual-machine: ~/桌面 x neuron@neuron-virtual-machine: ~/桌面 x
neuron@neuron-virtual-machine:~/桌面$ gcc producer.c -o producer -pthread
gcc: error: unrecognized command line option '-pthread'
neuron@neuron-virtual-machine:~/桌面$ gcc producer.c -o producer -pthread
producer.c: In function 'producer':
producer.c:23:42: warning: comparison between pointer and integer
    if((p = (int *)shmat(shm_id, NULL, 0)) == -1)
                                   ^
neuron@neuron-virtual-machine:~/桌面$ gcc consumer.c -o consumer -pthread
consumer.c: In function 'main':
consumer.c:30:9: warning: format '%d' expects argument of type 'int', but argume
nt 2 has type 'sem_t * {aka union <anonymous> *}' [-Wformat=]
    printf("%d",semEmpty);
    ^
consumer.c:33:42: warning: comparison between pointer and integer
    if((p = (int *)shmat(shm_id, NULL, 0)) == -1)
                                   ^
neuron@neuron-virtual-machine:~/桌面$ gcc consumer.c -o consumer -pthread
consumer.c: In function 'main':
consumer.c:29:42: warning: comparison between pointer and integer
    if((p = (int *)shmat(shm_id, NULL, 0)) == -1)
                                   ^
neuron@neuron-virtual-machine:~/桌面$ ./producer
create a shared memory segment sucessfully:23625739
neuron@neuron-virtual-machine:~/桌面$

```

```
神经元 (核心已转储)  
neuron@neuron-virtual-machine:~/桌面$ ./consumer 23625739  
59114:0  
59114:1  
59114:2  
59114:3  
59114:4  
59114:5  
59114:6  
59114:7  
59114:8  
59114:9  
59114:10  
59114:11  
59114:12  
59114:13  
59114:14  
59114:15  
59114:16  
59114:17  
59114:18  
59114:19  
59114:20
```

图片中./consumer有数字不用加这是我测试截的图 然后是系统调用 shm.h

```
#define __LIBRARY__  
#include <unistd.h>  
#include <linux/kernel.h>  
#include <linux/sched.h>  
#include <linux/mm.h>  
#include <errno.h>  
  
#define SHM_SIZE 20  
  
typedef struct  
{  
    unsigned int key;  
    unsigned int size;  
    unsigned long page;  
}shm_ds;  
  
_syscall1(int, shmat, int, shmid);  
_syscall2(int, shmget, unsigned int, key, size_t, size);
```

shm.c

```

#include <shm.h>

static shm_ds shm_ds_list[SHM_SIZE] = {0};

int sys_shmget(unsigned int key, size_t size)
{
    int i;
    void *page;
    if(size > PAGE_SIZE)
        return -EINVAL;
    page = get_free_page();
    if(!page)
        return -ENOMEM;
    for(i=0; i<SHM_SIZE; i++)
    {
        if(shm_ds_list[i].key == key)
            return i;
    }
    for(i=0; i<SHM_SIZE; i++)
    {
        if(shm_ds_list[i].key == 0)
        {
            shm_ds_list[i].page = page;
            shm_ds_list[i].key = key;
            shm_ds_list[i].size = size;
            return i;
        }
    }
    return -1;
}

void *sys_shmat(int shmid)
{
    int i;
    void *loc;
    if(0 < shmid || SHM_SIZE <= shmid)
        return -EINVAL;
    loc = current->brk + get_base(current->ldt[1]);
    current->brk += PAGE_SIZE;
    for(i=0; i<SHM_SIZE; i++)
    {
        if(shm_ds_list[shmid].key != 0)
        {
            put_page(shm_ds_list[shmid].page, loc);
            return current->brk - PAGE_SIZE;
        }
    }
    return -1;
}

```

我只按照老师的标准做了两个，没有做释放和删除的函数。你可以加上，也不复杂。其实这最麻烦的地方我觉得在于put_page，但是可以直接点用，想来直接写也是可以的。下面修改相应的producer.c

```

#include <stdio.h>
#include <stdlib.h>
#include <semaphore.h>
#include <sys/wait.h>
#include <sys/shm.h>
#include <fcntl.h>

#define PRODUCE_NUM 800
#define BUFF_SIZE 10
#define SHM_KEY 12345

void die(const char *p)
{
    puts(p);
    exit(1);
}

void producer(sem_t * Empty, sem_t * Full, sem_t * Mutex, int shm_id)
{
    int *p;
    int i, location=0;
    if((p = (int *)shmat(shm_id)) == -1)
        die("link error!");
    for(i=0; i<PRODUCE_NUM; i++)
    {
        sem_wait(Empty);
        sem_wait(Mutex);
        p[location] = i;
        sem_post(Full);
        sem_post(Mutex);
        location = (location+1)%10;
    }
    /*if(shmdt(p) == -1)
        die("reless shm error");
    */
    return;
}

int main(int argn, char* argc[])
{
    sem_t *semEmpty, *semFull, *semMutex;
    int i;
    int shm_id;

    sem_unlink("Mutex");
    sem_unlink("Full");
    sem_unlink("Empty");

    semEmpty = sem_open("Empty", BUFF_SIZE);
    semFull = sem_open("Full", 0);
    semMutex = sem_open("Mutex", 1);

    if((shm_id = shmget(SHM_KEY, BUFF_SIZE*sizeof(int))) == -1)
        die("shmget failed!");
    printf("create a shared memory segment sucessfully:%d\n", shm_id);
    producer(semEmpty, semFull, semMutex, shm_id);
    return 0;
}

```

consumer.c

```

#include <stdio.h>
#include <stdlib.h>
#include <semaphore.h>
#include <sys/wait.h>
#include <sys/shm.h>
#include <fcntl.h>

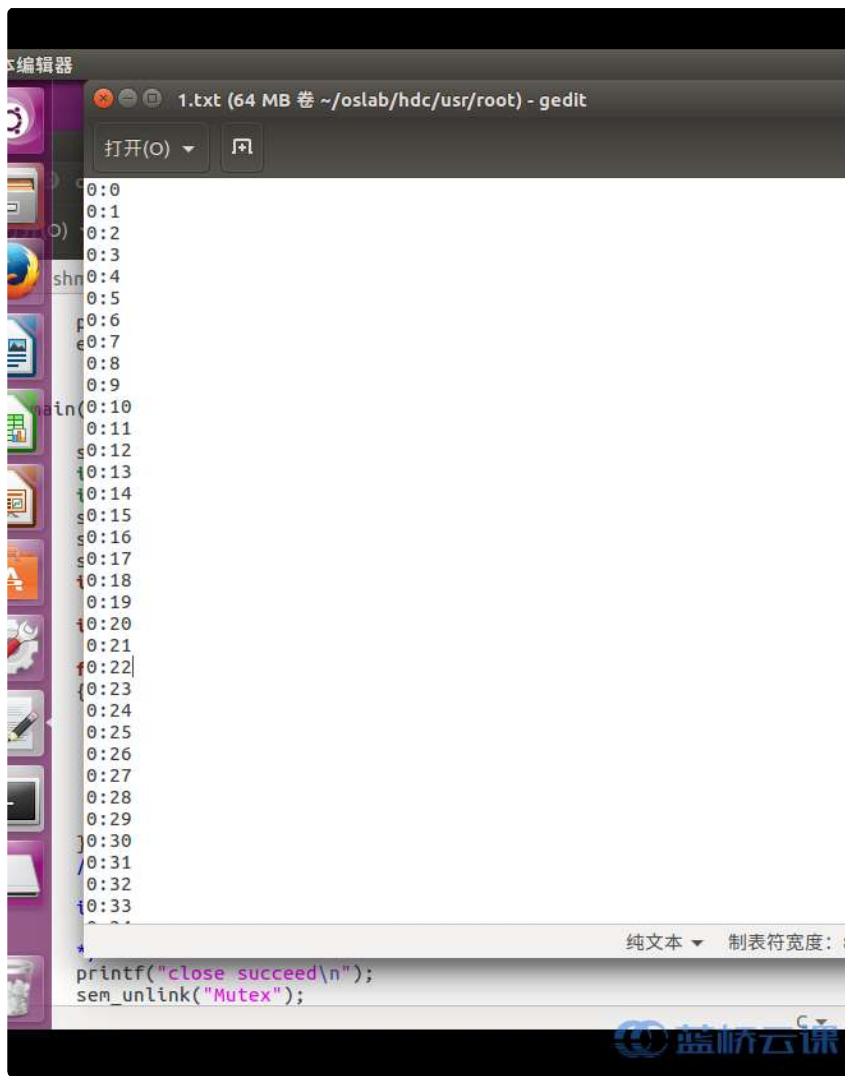
#define PRODUCE_NUM 800
#define BUFF_SIZE 10
#define SHM_KEY 12345

void die(const char *p)
{
    puts(p);
    exit(1);
}

int main(int argn, char* argc[])
{
    sem_t *semEmpty,*semFull,*semMutex;
    int shm_id,location = 0, pid;
    int *p,i;
    semEmpty = sem_open("Empty",BUFF_SIZE);
    semFull = sem_open("Full", 0);
    semMutex = sem_open("Mutex",1);
    if((shm_id = shmget(SHM_KEY, BUFF_SIZE*sizeof(int))) == -1)
        die("shmget failed!");
    if((p = (int *)shmat(shm_id)) == -1)
        die("link error!");
    for(i=0; i<PRODUCE_NUM; i++)
    {
        sem_wait(semFull);
        sem_wait(semMutex);
        printf("%d:%d\n", pid, p[location]);
        sem_post(semEmpty);
        sem_post(semMutex);
        location = (location+1)%10;
    }
    /*if(shmdt(p) == -1)
        die("reless shm error");
    if(shmctl(shm_id,IPC_RMID,NULL) == -1)
        die("remove shm error");
    */
    printf("close succeed\n");
    sem_unlink("Mutex");
    sem_unlink("Full");
    sem_unlink("Empty");
    return 0;
}

```

运行结果



那个pid之所以是0是因为我忘了get_pid()

看得出来代码的结构有些不合理。我是直接修改上一次实验用过的。实验之后发现的问题比解决的要多得多。绝知此事要躬行。

哎刚交的女朋友没俩星期就分手了。好伤心。完全没心情学习。程序员找个女盆友好难啊。



B **I**

[Markdown 语法](#)

请输入想说的话

0 / 2000

发表评论

最新评论



连接高校和企业



公司

关于我们
联系我们
加入我们

产品与服务

会员服务
蓝桥杯大赛
实战训练营
就业班

合作

1+X证书
高校实验教学
企业内训
合办学院

学习路径

Python学习路径
Linux学习路径
大数据学习路径
Java学习路径

