实验六 进程间通信——共享内存

**一、实验目的**

1、了解采用共享内存通信的原理。

2、掌握共享内存的创建及使用方法。

**二、实验原理**

1、使用共享内存实现进程间通信的步骤：

（1）shmget()创建共享内存。

（2）shmat()映射共享内存。

（3）读写进程使用共享内存通信。

（4）shmdt()撤销映射。

（5）shmctl()的IPC\_RMID参数，删除共享内存。

**三、实验内容**

1、创建写进程，其功能为创建共享内存并映射到当前进程地址空间。接着从标准输入向共享内存中循环写入数据，直至写入“end”后结束退出。

2、读进程使用和写进程相同的key打开共享内存，将共享内存映射到当前进程地址空间，并从共享内存中循环读取数据并输出至终端。

3、通过信号量方式解决读写进程的同步问题。

4、通信完成后，撤销共享内存和进程间的映射，并删除共享内存，删除信号量。

**四、实验代码**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<unistd.h>**

**#include<sys/types.h>**

**#include<sys/wait.h>**

**#include<sys/ipc.h>**

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

**#include<sys/sem.h>**

**#include<string.h>**

**#include<assert.h>**

**union semun {**

**int val; /\* Value for SETVAL \*/**

**struct semid\_ds \*buf; /\* Buffer for IPC\_STAT, IPC\_SET \*/**

**unsigned short \*array; /\* Array for GETALL, SETALL \*/**

**struct seminfo \*\_\_buf; /\* Buffer for IPC\_INFO**

**(Linux-specific) \*/**

**};**

**static void semaphore\_p(int sem\_id);**

**static void semaphore\_v(int sem\_id);**

**int main(int arc, char \*argv[])**

**{**

**key\_t key = ftok(".", 1);**

**assert(key != -1);**

**int shmid = shmget(key, 4096, IPC\_CREAT | IPC\_EXCL | 0600);**

**assert(shmid != -1);**

**char \*shm = shmat(shmid, NULL, 0);**

**int semid = semget(key, 1, IPC\_CREAT | IPC\_EXCL | 0600);**

**assert(semid != -1);**

**union semun sem\_union;**

**sem\_union.val = 1;**

**assert(semctl(semid, 0, SETVAL, sem\_union) != -1);**

**pid\_t pid = fork();**

**assert(pid != -1);**

**if(pid == 0){**

**sleep(1);**

**semaphore\_p(semid);**

**printf("\nread process[%d] lock....\n", getpid());**

**printf("read data from shared memory:\n");**

**printf("%s", shm);**

**printf("read process[%d] unlock....\n", getpid());**

**semaphore\_v(semid);**

**exit(0);**

**}**

**char buffer[64];**

**char \*work = shm;**

**semaphore\_p(semid);**

**printf("write process[%d] lock....\n", getpid());**

**printf("please input your data:\n");**

**while(1){**

**char \*temp = fgets(buffer, 64, stdin);**

**int flag = strcmp(temp, "end\n");**

**while(\*temp){**

**\*work++ = \*temp++;**

**}**

**if(flag == 0){**

**break;**

**}**

**}**

**\*work = '\0';**

**printf("write process[%d] unlock....\n", getpid());**

**semaphore\_v(semid);**

**wait(NULL);**

**union semun sem\_union2;**

**semctl(semid, 0, IPC\_RMID, sem\_union2);**

**// delete memory map**

**shmdt(shm);**

**// delete shared memory id**

**shmctl(shmid, IPC\_RMID, NULL);**

**return 0;**

**}**

**static void semaphore\_p(int sem\_id){**

**struct sembuf sem\_b;**

**sem\_b.sem\_num = 0;**

**sem\_b.sem\_op = -1; //P()**

**sem\_b.sem\_flg = SEM\_UNDO;**

**assert(semop(sem\_id, &sem\_b, 1) != -1);**

**}**

**static void semaphore\_v(int sem\_id){**

**struct sembuf sem\_b;**

**sem\_b.sem\_num = 0;**

**sem\_b.sem\_op = 1; //V()**

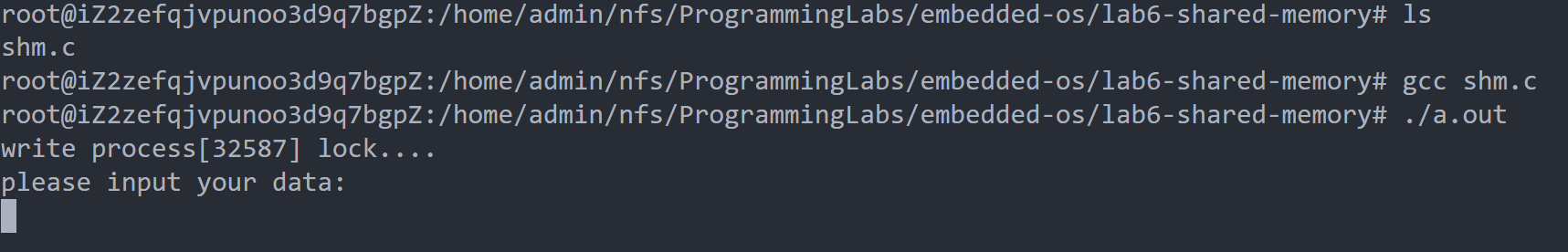
**sem\_b.sem\_flg = SEM\_UNDO;**

**assert(semop(sem\_id, &sem\_b, 1) != -1);**

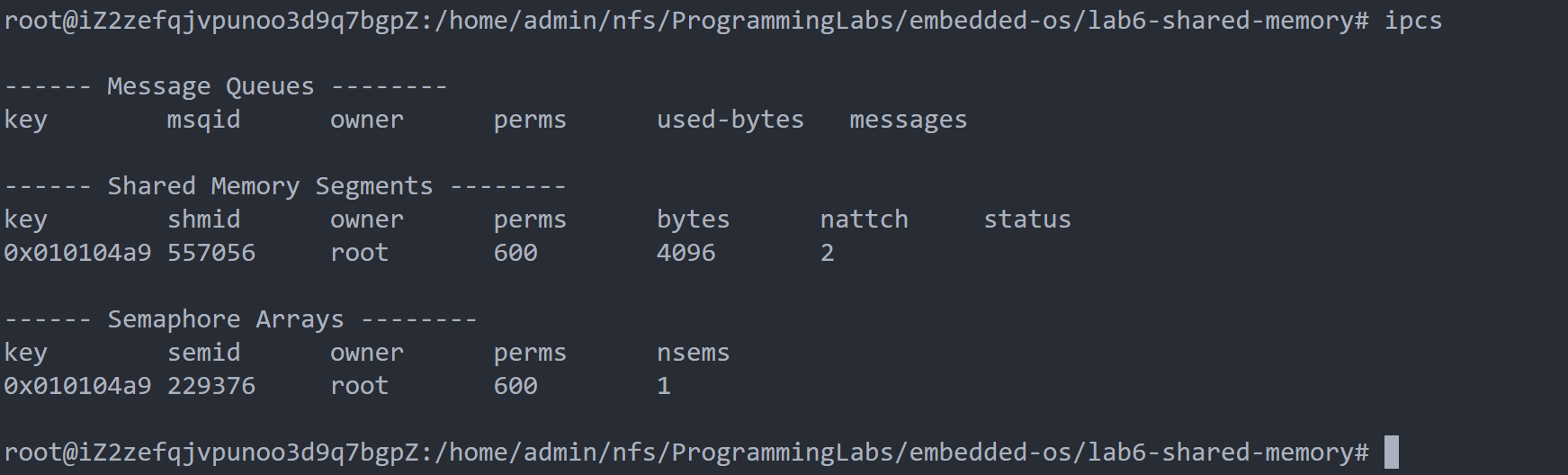
**}**

截图

1. 编译并运行



1. 打开另一个终端，使用ipcs查询进程间通信状态



1. 在写进程中输入数据, 并且以end结束

