姓名：吴涵 学号：202101000720 成绩：

# 实验31共享内存通信

1、共享内存通信

源程序：

（1）shmmutexwrite.c：

#include <semaphore.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

#include <sys/shm.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <string.h>

#include <unistd.h>

#define BUFFER\_SIZE 10

#define sem\_name "mysem"

int main()

{

struct Stu

{

char name[10];

int score;

};

int shmid;

sem\_t \*sem;

int score = 60, i = 1;

char buff[BUFFER\_SIZE];

key\_t shmkey;

shmkey = ftok("shmmutexread.c", 0);

sem = sem\_open(sem\_name, O\_CREAT, 0644, 1);

if (sem == SEM\_FAILED)

{

printf("unable to creat semaphore!");

sem\_unlink(sem\_name); // 删除有名信号量

exit(-1);

}

shmid = shmget(shmkey, 1024, 0666 | IPC\_CREAT);

/\*创建IPC键值为shmkey的共享内存，其大小为1024字节，允许读写\*/

if (shmid == -1)

printf("creat shm is fail\n");

struct Stu \*addr;

addr = (struct Stu \*)shmat(shmid, 0, 0);

if (addr == (struct Stu \*)-1)

printf("shm shmat is fail\n");

addr->score = 0;

printf("写进程映射的共享内存地址=%p\n", addr);

do

{

sem\_wait(sem);

memset(buff, 0, BUFFER\_SIZE);

memset((addr + i)->name, 0, BUFFER\_SIZE);

printf("写进程:输入一些姓名（不超过10个字符）到共享内存(输入'quit' 退出):\n");

if (fgets(buff, BUFFER\_SIZE, stdin) == NULL)

{

sem\_post(sem);

break;

}

strncpy((addr + i)->name, buff, strlen(buff) - 1);

(addr + i)->score = ++score;

addr->score++;

i++;

sem\_post(sem);

sleep(1);

} while (strncmp(buff, "quit", 4) != 0);

if (shmdt(addr) == -1) /\*将共享内存与当前进程断开\*/

printf("shmdt is fail\n");

sem\_close(sem); // 关闭有名信号量

sem\_unlink(sem\_name); // 删除有名信号量

}

（2）shmmutexread.c：

#include <semaphore.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

#include <sys/shm.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <string.h>

#define sem\_name "mysem"

int main()

{

int shmid;

sem\_t \*sem;

int i = 1;

key\_t shmkey;

shmkey = ftok("shmmutexread.c", 0);

struct Stu

{

char name[10];

int score;

};

sem = sem\_open(sem\_name, 0, 0644, 0);

if (sem == SEM\_FAILED)

{

printf("unable to open semaphore!");

sem\_close(sem);

exit(-1);

}

shmid = shmget(shmkey, 0, 0666);

if (shmid == -1)

{

printf("creat shm is fail\n");

exit(0);

}

struct Stu \*addr;

addr = (struct Stu \*)shmat(shmid, 0, 0);

if (addr == (struct Stu \*)-1)

{

printf("shm shmat is fail\n");

exit(0);

}

printf("读进程映射的共享内存地址=%p\n", addr);

do

{

sem\_wait(sem);

if (addr->score > 0)

{

printf("\n读进程:绑定到共享内存 %p:姓名 %d %s , 分值%d \n", addr, i, (addr + i)->name, (addr + i)->score);

addr->score--;

if (strncmp((addr + i)->name, "quit", 4) == 0)

break;

i++;

}

sem\_post(sem);

} while (1);

sem\_close(sem);

if (shmdt(addr) == -1)

printf("shmdt is fail\n");

if (shmctl(shmid, IPC\_RMID, NULL) == -1)

printf("shmctl delete error\n");

}

编译链接命令：

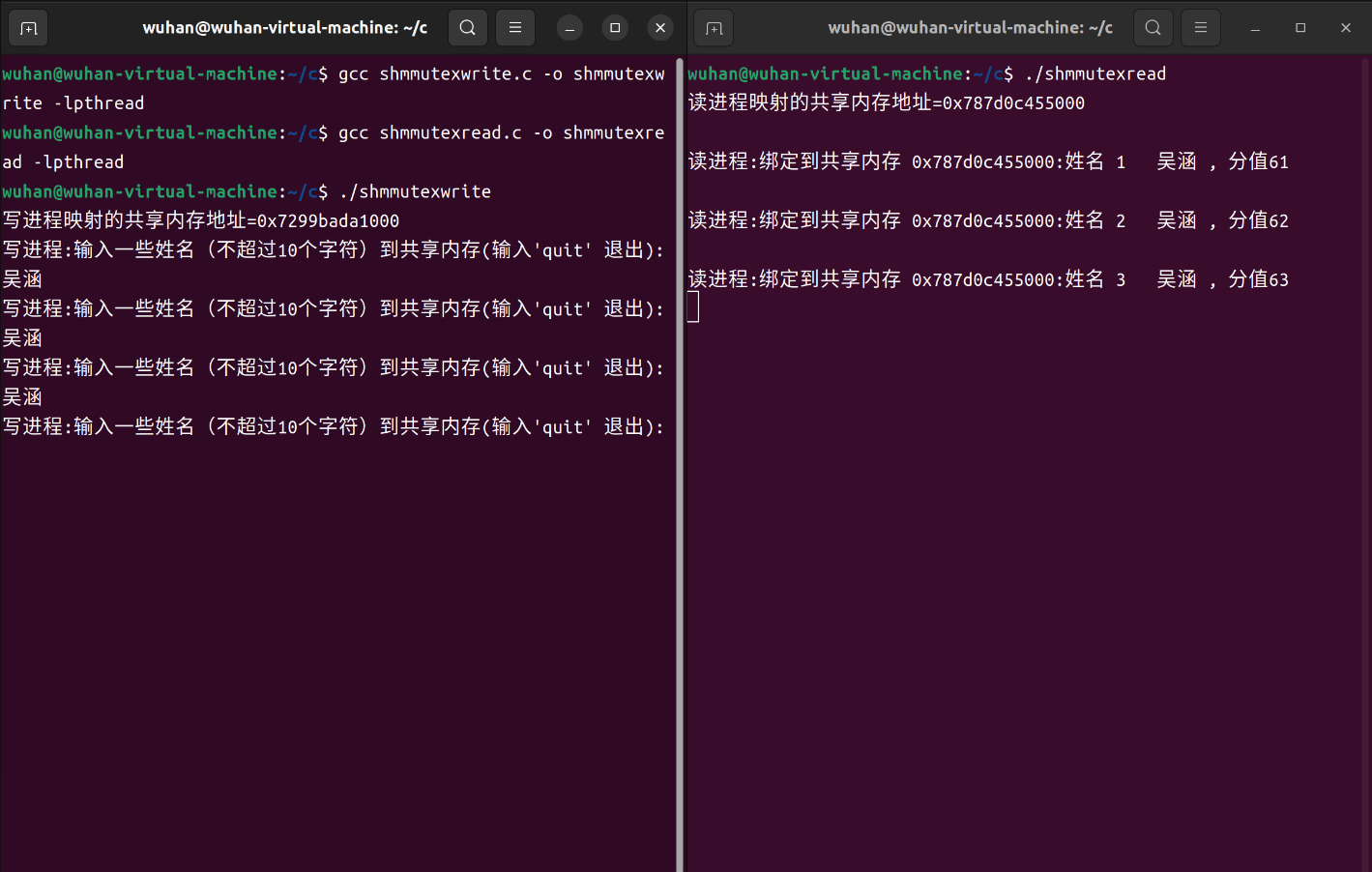
gcc shmmutexwrite.c -o shmmutexwrite -lpthread

gcc shmmutexread.c -o shmmutexread –lpthread

运行命令：

./shmmutexwrite

./shmmutexread

交互与结果：