实验七　进程间通信

一、实验目的

(1) 理解 Linux 关于进程间通信的概念。

(2) 掌握几种进程间通信的方法。

(3) 巩固进程同步概念和实现进程同步的方法。

二、实验内容

(1) 编写 C 程序，让父子两个进程通过消息队列相互聊天、发送消息，仅要求实现父（子）进程发送一条后接收一条，如此循环即可，不要求实现连续发送（接收）多条信息。其中每条消息大小不超过 1024 字节。

提示：针对消息队列的使用，一般需要用到以下函数，主要在头文件 sys/msg.h 中：

int msgget(key_t key, int msgflg); 创建一个消息队列，key 为一个整数，可以通过 key_t ftok(char * fname, int id)函数来创建。

int msgsnd(int id, const void *msgp, size_t size, int flag); 发送信息。

int msgrcv(int id, void *msgp, size_t size, long type, int flag); 接受信息。

在 msgsnd 和 msgrcv 中的 msgp 指向用户自定义的数据结构体。

消息结构体定义：

```
typedef struct{
        long msg_type;  //消息类型必须在第一个位置
    /*定义自己的数据内容*/
        ...
}msg_t;
```

int msgctl(int id, int cmd, struct msqid_ds *buf); 删除消息队列。

(2) 编写 C 程序，实现 reader 和 writer 两个进程，使它们通过共享内存交换数据：writer 从用户处获得输入，然后将其写入共享内存，reader 从共

享内存获取信息，再在屏幕上打印出来。

提示：针对共享内存的使用，一般需要用到以下函数，它们声明在头文件 sys/shm.h 中：

int shmget(key_t key, size_t size, int shmflg); 用来创建共享内存。

void *shmat(int shm_id, const void *shm_addr, int shmflg); 把共享内存连接到当前进程的地址空间。

int shmdt(const void *shmaddr); 将共享内存从当前进程中分离。

int shmctl(int shm_id, int command, struct shmid_ds *buf); 用来控制共享内存，第二个参数设置为 IPC_RMID 时，删除共享内存段。

(3) 使用多线程和信号量解决生产者/消费者问题：有一个长度为 N 的缓冲池被生产者和消费者共同使用。只要缓冲池未满，生产者就可以将消息送入缓冲池；只要缓冲池不空，消费者便可从缓冲池中取走一个消息。生产者向缓冲池放入消息的同时，消费者不能操作缓冲池，反之亦然。

提示：对于信号量的操作，一般需要用到以下函数，它们声明在头文件 semaphore.h 中：

int sem_init(sem_t *sem, int pshared, unsigned int value); 用于信号量的初始化。

int sem_wait(sem_t *sem); 执行信号量的 P 操作。

int sem_post(sem_t *sem); 执行信号量的 V 操作。

int sem_destroy(sem_t *sem); 用于对使用完的信号量的销毁。

三、实验要求

(1)在完成实验后，学生应具备对 Linux 下进程间通信方法的理解，能够灵活应用消息队列、共享内存、多线程和信号量进行编程，实现进程间通信。

(2)学生需要记录实验过程中编写的程序以及程序运行结果，在实验报告中提供相应的截图和解释。