任务：

第二章

（1） 用C语言完成线程的创建和撤消，并按先来先服务方式对多个线程进行调度。

创建：Int create(char \*name,codeptr code,int stck) ；

撤销：void destroy（int id）；

（2） 将线程调度算法修改为时间片轮转算法，实现时间片轮转调度。（也可以结合优先权，实现优先权加时间片轮转算法的线程调度。）

void interrupt my\_swtch(void)：该函数主要解决两种原因引起的调度：线程执行完毕或正在执行的线程因等待某事件发生而不能继续执行。

void interrupt new\_int8(void)：该函数主要解决因时间片到时引起的调度，这可通过截取时钟中断（int 08）来完成。

（3）改变时间片的大小，观察结果的变化。思考：为什么时间片不能太小或太大。

（4）假设两个线程共用同一软件资源（如某一变量，或某一数据结构），请用记录型信号量来实现对它的互斥访问。

void p(semaphore \*sem)

void v(semaphore \*sem)

（5）假设有两个线程共享一个可存放５个整数的缓冲，其中一个线程不停地计算１至５０的平方，并将结果放入缓冲中，另一个线程不断地从缓冲中取出结果，并将它们打印出来，请用记录型信号量实现这一生产者和消费者的同步问题。

void p(semaphore \*sem)

void v(semaphore \*sem)

（6）实现消息缓冲通信，并与4、5中的简单通信进行比较。

获取空闲缓冲区函数getbuf()

插入缓冲区到缓冲队列函数insert()

发送原语send()

获取消息缓冲区函数remov()

接收原语receive()

（7）思考：在线程间进行消息缓冲通信时，若对消息队列的访问没有满足互斥要求，情况将会怎样？

第四章

**2.** 编写三个不同的程序cmd1.c，cmd2.c，cmd3.c，每个程序输出一句话，分别编译成可执行文件cmd1，cmd2，cmd3。然后再编写一个程序，模拟shell程序的功能，能根据用户输入的字符串（表示相应的命令名），去为相应的命令创建子进程并让它去执行相应的程序，而父进程则等待子进程结束，然后再等待接收下一条命令。如果接收到的命令为exit，则父进程结束；如果接收到的命令是无效命令，则显示“Command not found”，继续等待。

**3.** 由父进程创建一个管道，然后再创建2个子进程，并由这两个兄弟进程利用管道进行进程通信：子进程1使用管道的写端，子进程2使用管道的读端。通信的具体内容可根据自己的需要随意设计，要求能试验阻塞型读写过程中的各种情况。运行程序，观察各种情况下，进程实际读写的字节数以及进程阻塞唤醒的情况。

int pipe(int filedes[2]);

int mkfifo(const char \* pathname, mode\_t mode)// 有名管道通过mkfifo创建

int open(const char \*pathname, int flags);//对有名管道的open操作必须遵循下列规则

read、write系统调用对管道进行读写操作

int close(const char \*pathname)//读写完成应使用close系统调用关闭有名管道。

**4.** 编写程序创建两个线程：sender线程和receive线程，其中sender线程运行函数sender()，它创建一个消息队列，然后，循环等待用户通过终端输入一串字符，将这串字符通过消息队列发送给receiver线程，直到用户输入“exit”为止；最后，它向receiver线程发送消息“end”，并且等待receiver的应答，等到应答消息后，将接收到的应答信息显示在终端屏幕上，删除相关消息队列，结束程序的运行。Receiver线程运行receive()，它通过消息队列接收来自sender的消息，将消息显示在终端屏幕上，直至收到内容为“end”的消息为止，此时，它向sender发送一个应答消息“over”，结束程序的运行。使用无名信号量实现两个线程之间的同步与互斥。

int msgget(key\_t key, int msgflg)

//如果IPC\_CREAT单独使用，semget()为一个新创建的消息队列返回标识数，或者返回具有相同键值的已存在消息队列标识数。如果IPC\_EXCL与IPC\_CREAT一起使用，要么创建一个新的队列并返回它的标识数，如果队列已存在，则返回-1。

int msgsnd(int msqid, struct msgbuf \*msgp, size\_t msgsz, int msgflg);

//在标识数为msqid的消息队列中添加一个消息，即向标识数为msqid的消息队列发送一个消息。

ssize\_t msgrcv(int msqid, struct msgbuf \*msgp, size\_t msgsz, long msgtyp, int msgflg);

//如果传递给参数msgflg的值为IPC\_NOWAIT，并且没有可取的消息，那么给调用进程返回ENOMSG错误码，否则，调用进程阻塞，直到一条满足要求的消息到达消息队列。如果进程正在等待消息，而相应的消息队列被删除，则返回EIDRM。如果当进程正在等待消息时，捕获到了一个信号，则返回EINTR

int msgctl(int msqid, int cmd, struct msqid\_ds \*buf);

//获取或设置消息队列的属性信息，或者删除消息队列

5．编写程序sender，它创建一个共享内存，然后等待用户通过终端输入一串字符，并将这串字符通过共享内存发送给receiver；最后，它等待receiver的应答，等到应答消息后，将接收到的应答信息显示在终端屏幕上，删除共享内存，结束程序的运行。编写receiver程序，它通过共享内存接收来自sender的消息，将消息显示在终端屏幕上，然后再通过该共享内存向sender发送一个应答消息“over”，结束程序的运行。使用无名信号量或System V信号量实现两个进程对共享内存的互斥使用。

int shmget(key\_t, int size, int shmflg);//创建一块共享内存。

void \*shmat(int shmid, const void \*shmaddr, int shmgflg);

//将一块已存在共享内存映射（map）到一个进程的地址空间。

int shmdt（const void \*shmaddr）；

//取消一个进程的地址空间中一块共享内存块的映射（unmap）。

int shmctl (int shmid, int cmd, struct shmid\_ds \*buff);

//是管理共享内存的函数，用于执行对共享内存的各种控制命令。

第二章 1 （ 2，3） 4 5 6

第四章 2 3 4 5