操作系统实验三报告

一、实验内容：

1）实现一个模拟的shell 2）实现一个管道通信程序 3）利用Linux消息队列通信机制实现两个线程间的通信 4）利用Linux共享内存通信机制实现两个进程间的通信

二、实验思路

（1）实验一模拟shell

pid = fork()会返回多个值，只需在fork()后使用多个判断语句即可。 pid<0表示错误，我打印error之后退出 pid=0表示子进程运行，使用execl替换进程，替换为我们想要的进程，如cmd.o。 pid>0表示父进程运行，使用wait(NULL)函数等待所有子进程的退出。

（2）实验二pipe

使用2个信号量：
write_psx = sem_open(SEM_W, O_CREAT, 0666, 1); read_psx = sem_open(SEM_R, O_CREAT, 0666, 0);

1）建立无名管道： int fd[2]; int ret = pipe(fd);//无名管道

2）先fork三个子进程，编写各自的操作 每个子进程等待各自的send信号量，再等待Mutex信号量，进行，完成后释放各自的receiver： P(write) 发送内容到管道 V(read) V(write) **父进程等待read信号量，接收管道内容** P(read) 接受内容

pipe默认大小

每次发送1024个字节消息给pipe，然后关闭read端，观察管道最大容量为65535。

（3）实验三消息队列

三个信号量： sem_init(&send_psx, 0, 1); sem_init(&recv_psx, 0, 0); sem_init(&final_recv, 0, 0); 发送方： While(1) { P（sen_psx） 发送信息 如果是exit V（recv_psx） break; V(send_psx); V(recv_psx); } P(final_recv); 接受方： while(1){ P(recv_psx); 接受信息 如果是exit V（final_recv） 退出 }

（3）实验四共享内存

三个信号量 sem_init(&send_psx, 0, 1); sem_init(&recv_psx, 0, 0); sem_init(&final_recv, 0, 0); shmid = shmget(key, SHM_SIZE, 0666|IPC_CREAT);//创建共享内存空间 shmp = shmat(shmid, NULL, 0);//启动对共享内存的访问

发送方

while(1)

{

P（send_psx）

接受terminal的信息并且把信息放进共享内存

如果是exit

V（recv_psx）；

V(send_psx);

}

P(final_psx)

接受方

while(1){

P（recv_psx）；

把数据从共享空间拷到自己空间

如果是exit

V（fianl_psx）；

}

close(all)//关闭信号量

三、遇到的问题

（1）实验一

遇到的问题及解决：cmd1.c要编译成可执行文件才能被shell调用。

（2）实验二

不能正常编译，会出现好多未定义

原因：这个是因为pthread并非Linux系统的默认库，编译时注意加上-lpthread参数，以调用链接库 我们再一次在终端输入：gcc -o consumer.out consumer.c -lpthread

（3）实验三

信号量初始值为0时会程序会永远等待下去sem_init(&send_psx, 0, 0);，形成死锁。

（4）实验四

支持多终端发送，单终端接收。 发送顺序不是固定。