|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | 何飞亚 | **学号** | 09 |
| **实验题目** | **第三次实验** | | |
| **实验内容** | （1）编制一段程序，使用系统调用 fork()创建两个子进程，再用系统调用 signal()让父进程捕捉键盘上来的中断信号（即按【Del】键），当捕捉到中断信号后，父进程用系统调用kill()向两个子进程发出信号，子进程捕捉到信号后，分别输出下列信息后终止：  child process1（pid=XX ,ppid=XX） is killed by parent!  child process2 (pid=XX ,ppid=XX) is killed by parent!  父进程等待两个子进程终止后，输出以下信息后终止：  Parent(pid=XX) process is killed!  代码部分：    编译和运行：    如果不使用ctrl+C中断，程序将在五秒后中断  （2）在上面的程序中增加语句signal(SIGNAL,SIG-IGN)和signal(SIGQUIT,SIG-IGN)，观察执行结果，并分析原因。  代码部分：  编译和运行部分：    运行时，由于忽略中断信号，程序不能自行中断，利用ctrl+C进行中断  （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，消费者便可以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。  代码部分：    编译和运行部分： | | |
| **总结** | 1. fork()创建一个新进程，pid=0：创建子进程，从子进程返回的 id 值;大于0：从父进程返回的子进程 id 值;-1：创建失败; 2. lockf(files,function,size):用作锁定文件的某些段或者整个文件 3. 进程标识符管理：   int getpid():取得当前进程的标识符；  int getppid():取得当前进程的父进程ID   1. 发送信号kill(pid,sig):本进程将指定信号sig发送给指定进程pid 2. 接受信号signal(sig,func),本进程接受到其他进程发送给他的信号后，完成指定的功能func 3. SIG\_IGN：忽视该信号的出现 4. msgget(key,flag):获得一个消息的描述符，该描述符指定一个消息队列以便用于其他系统调用。 5. msgsnd(id,msgp,size,flag):发送一消息。 6. msgrcv(id,msgp,size,type,flag):接受一消息。 7. msgctl(id,cmd,buf):查询一个消息描述符的状态，设置它的状态及删除一个消息描述符。 8. shmget(key,size,flag):获得一个共享存储区。 9. shmat(id,addr,flag):从逻辑上将一个共享存储区附接到进程的虚拟地址空间上。 10. shmdt(addr):把一个共享存储区从指定进程的虚地址空间分开。 11. shmctl(id,cmd,buf):对与共享存储区关联的各种参数进行操作，从而对共享存储区进行控制。 12. Linux线程的基本使用方法 13. Linux线程之互斥锁 14. POSIX信号量 | | |
| **日期** | 2020.6.5 | **成绩** |  |