|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | 李昊阳 | **学号** | 14 |
| **实验题目** | 进程控制和进程通信 | | |
| **实验环境** | **虚拟机：VMware Workstation 15.5.0 Pro**  **Linux版本：Ubuntu-19.10** | | |
| **实验内容** | 1. 通过C语言编程，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!  首先使用gcc命令编译lab3.c文件：    运行a.out，分别使用Ctrl+C中断进程以及不使用中断信号，观察实验结果：     1. 在上面的程序中增加语句signal(SIGINT,SIG-IGN)和signal(SIGQUIT,SIG-IGN)，观察执行结果，并分析原因。   在程序中添加语句：    首先只添加语句signal(SIGINT,SIG-IGN)，此时无论发不发送中断信号程序子进程和父进程都会被kill：    之后只添加语句signal(SIGQUIT,SIG-IGN)，此时若发送中断信号则父进程会被kill：    实验结果分析：进程的软中断通信是利用函数signal()和kill()来  实现的，signal(SIGINT,SlG\_SGN）和signal(SIGQUlT,SIG\_IGN)  分别为忽略键入信号以及忽略中断信号。子进程中的signal()函数通过接收父进程的kill()函数发来的信号来执行输出终止操作。在子进程中开始添加signal(SIGINT,SlG\_SGN） 则子程忽略键盘中断信号， 此时按ctrl-c  不会将子进程终止， 程序可以正常运行， 而在子进程中开始添加  signal(SlGQUIT,SIG\_IGN), 则相应的中断信号被忽略，只有父进程中断。   1. 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。   通过gcc生成文件proandcus，其中-lpthread是在链接阶段，链接指定库，之后执行：    可看出在一个队列中生产者和消费者相互工作的情况，初始状态缓冲区为空，此时只允许生产者生产资源放入缓冲区，而消费者试图取缓冲区产品时，将进入等待队列，等待资源释放，在等待队列中的第一个进程将得到资源的控制权。当缓冲区产品满了，将拒绝生产者向缓冲区输送资源。生产者每隔三秒向缓冲区输送产品，缓冲区满了，则等待消费者消费产品后再将产品输送进去；消费者每隔六秒向缓冲区取得产品，缓冲区为空，则等待生产者输送产品后再消费产品。符合生产者消费者模型。 | | |
| **总结** | 加深了对Linux系统命令行运用的熟悉程度，以及进程相关的命令fork、wait、sleep等命令的理解；了解了操作系统中进程的创建、中断、管理的方式，对课上所学内容加深了理解，并对多线程和信号量解决生产者消费者问题有了直观地理解。另外回顾了C语言的使用，加深了对Linux系统的了解并熟悉了Ubuntu的系统框架和gcc的使用方法，希望在以后的学习中能够提高对Linux系统操作方法的熟练程度。 | | |
| **日期** | 2020.06.07 | **成绩** |  |