|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | 戚\*宝 | **学号** | 27 |
| **实验题目** | Lab3 | | |
| **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!  首先是创建两个子进程，先是创建一个进程，在判断父进程的情况下再创建另一个子进程，在主进程中用signal()注册两个信号，当父进程收到 . 时给两个子进程发信号，子进程开始处理      （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。  signal(SIGNAL,SIG-IGN)不合法，不能添加，signal(SIGQUIT,SIG-IGN)添加后执行结果应该是子进程退出，但等待的父进程收不到子进程退出的信号，从而处于一直等待状态，不输出语句。然而，由于程序的问题，这种情况在我这没有发生，子进程结束后，父进程也结束了    （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。  每一个生产者都要把自己生产的产品放入缓冲池，每个消费者从缓冲池中取走产品消费。在这种情况下，生产者消费者线程同步，因为只有通过互通消息才知道是否能存入产品或者取走产品。他们之间也存在互斥，即生产者消费者必须互斥访问缓冲池，即不能有两个以上的线程同时进行  最终执行结果如下： | | |
| **总结** | （1）掌握Linux操作系统的进程创建和终止，fork,wait,exit；  （2）学会了Linux操作系统提供的“软中断”机制实现进程间的同步，主要是signal()函数的使用。  （3）了解几种典型的进程通信机制（例如：管道通信，消息队列，共享内存、套接字等）以及使用信号量进行进程通信。 | | |
| **日期** | 2020.06.06 | **成绩** |  |