|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | 张昊哲 | **学号** | 40 |
| **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!  编程完后保存为3.c源文件，在终端执行gcc 3.c命令生成a.out文件，  输入./a.out执行源文件      此时已经执行fork（），在等待终端输入中断指令。当输入中断指令后，由于ctrl+c信号会并发传到每个进程中，进程受到该信号会立刻终止。当子进程收到ctrl+c信号时，就终止了，根本不会等父进程传来的软中断信号，因此也就不会打印出child process1 is killed和child process2 is killed。    （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。  再加入上述语句后，再次运行结果如下：    其原因是signal(SIGINT, SIG-IGN)和signal(SIGQUIT, SIG-IGN)的作用是屏蔽从键盘上传来的中断信号，因此子进程可以接收到父进程传来的软中断信号，进而将那两句话打印出来。  （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。  代码如图所示：        在终端输入gcc 4.c -o 4 -lpthread将其变成可执行文件，在输入./4执行。结果如图所示： | | |
| **总结** | 1，了解了Linux操作系统的进程创建和终止，以及fork（）创建进程的  过程。  2，了解Linux操作系统提供的“软中断”机制实现进程间的同步  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** | 2020.06.08 | **成绩** |  |