|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | **刘家豪** | **学号** | **18** |
| **实验题目** | **进程控制和进程通信** | | |
| **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!**  利用fork()创建进程，singal()接收信号并转入相应的函数执行，kill()发送信号给相应的进程。用getpid()获取当前正在执行进程的pid，getppid()来获取当前执行进程的父进程的pid。而且利用锁函数lockf()来进行进程p1和p2的互斥操作。由此可以编写符合相应要求的C代码。并进行编译执行，得到的结果如下：    **（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(SIGINT,SIG\_IGN)和signal(SIGQUIT,SIG\_IGN)分别为忽略键信号以及忽略中断信号。因此，如果在父进程所执行的代码中加入上述两个语句，会使得父进程无法继续执行接收中断信号后需要执行的代码，即忽略了中断信号，不对中断信号作出任何的响应。结果如下：    **（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。**  利用pthread\_create()创建多个线程，sleep()暂停以此进行速度的同步，以及对信号量的使用以此实现多线程之间的互斥操作，建立了缓冲区数组，由此模拟了生产者/消费者问题的具体实现。可以据此编写出代码，编译执行结果如下：    在使用gcc编译的过程中，需要使用到其他的选项，分别是-lpthread和-finput-charset=GB2312,第一个选项的意思是链接了线程库，由此可以使用线程中的API，第二个选项的意思是指定了编译此C程序的编码方式为GB2312，由此可以保证最后编译后的程序执行时输出的语言为中文，而不是一堆乱码 | | |
| **总结** | 1. 掌握了使用了有关进程控制和通信的一些API的用法 2. 明白如何在Linux操作系统进行进程的创建和终止 3. 理解了Linux操作系统里提供的“软中断”机制的实施过程 4. 了解了几种典型的进程通信机制如管道通信等 5. 掌握了使用信号量来进行多线程之间的通信 6. 成功的模拟了生产者和消费者问题的解决情况 | | |
| **日期** | 2020.06.08 | **成绩** |  |