|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | 刘良伟 | **学号** | 4062017019 |
| **实验题目** | （1）掌握Linux操作系统的进程创建和终止；（2）利用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以及使用信号量进行进程通信。 | | |
| **实验内容** | **lab3\_1**：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!（2）在上面的程序中增加语句signal(SIGNAL,SIG-IGN)和signal(SIGQUIT,SIG-IGN)（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。  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ctrl c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，输出结果信息。  源代码如下：    结果分析：Pid为3638的父进程创建了两个子进程，在接收到中断信号后父进程被终止，同时由于子进程没有忽略该中断信号，也被终止，没能打印出子进程信息。  **Lab3\_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。    结果分析：Pid为3853的父进程创建了两个pid分别为3854和3855的两个子进程，因为signal(SIGINT, SIG-IGN)和signal(SIGQUIT, SIG-IGN)屏蔽了从键盘上传来的中断信号，所以子进程忽略了中断信号，所以能打印出信息。  **lab3\_3**：使用多线程和信号量解决生产者/消费者问题  源代码如下：    在终端中执行高gcc task3.c -o task3 -之后： | | |
| **总结** | 1.通过这次实验，我进一步熟悉了linux系统，也学会了进程的创建过程。  2.对生产者、消费者问题加深了自己的理解。对于子进程和父进程也有了一定的了解。 | | |
| **日期** | 2020.06.09 | **成绩** |  |