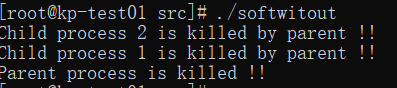
**实验2-1 进程的软中断通信**

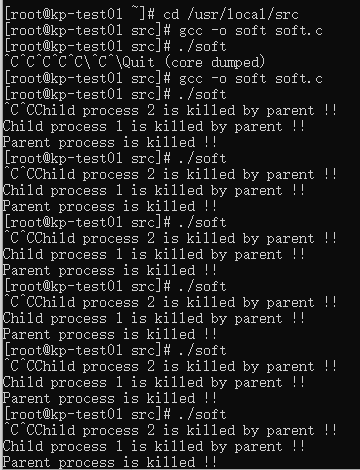
**1. 实验要求：使用系统调用fork()创建两个子进程，再用系统调用signal()让父进程捕捉键盘上发出的中断信号（即按delete键），当父进程接收到这两个软中断的某一个后，父进程用系统调用kill()向两个子进程分别发出整数值为16和17软中断信号，子进程获得对应软中断信号，然后分别输出下列信息后终止： Child process 1 is killed by parent !! Child process 2 is killed by parent !! 父进程调用wait()函数等待两个子进程终止后，输入以下信息，结束进程执行：Parent process is killed!!**

代码运行情况如下：



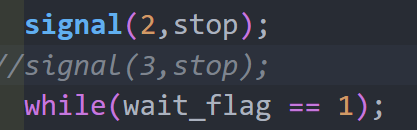
* **1、根据流程图编写程序，猜想一下这个程序的运行结果，然后多次运行，观察Delete/quit键前后，会出现什么结果？分析原因。**

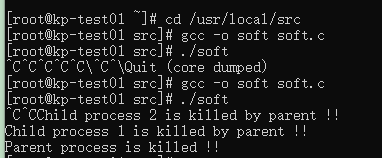
理论上会出现1，2被kill的输出前后不一样，（但是试了很多次没有出现1在2前面的情况）。这是因为每个进程收到信号的时间随机。



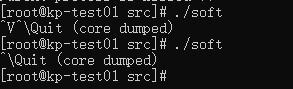
* **2、如果程序运行界面上显示“Child process 1 is killed by parent !! Child process 2 is killed by parent !!”，五秒之后显示“Parent process is killed !!”，怎样修改程序使得只有接收到相应的中断信号后再发生跳转，执行输出？**

答：在父进程中引入信号量，在收到中断信号时被唤醒，即可执行



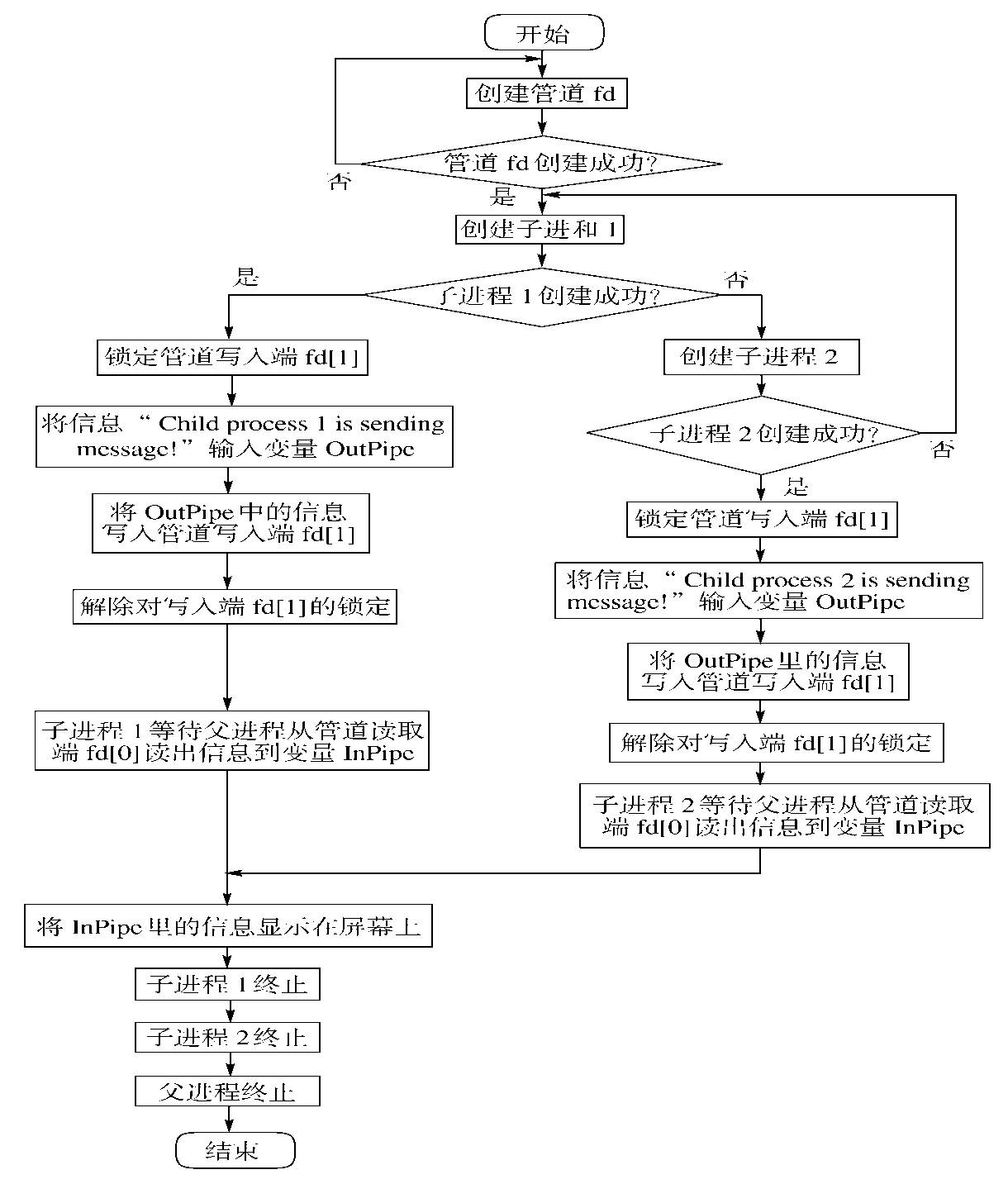


* **3、将本实验中通信产生的中断通过14 号信号值进行闹钟中断，将signal(3,stop)当中数字信号变为2，体会不同中断的执行样式，从而对软中断机制有一个更好的理解。**

输入quit之后程序直接退出

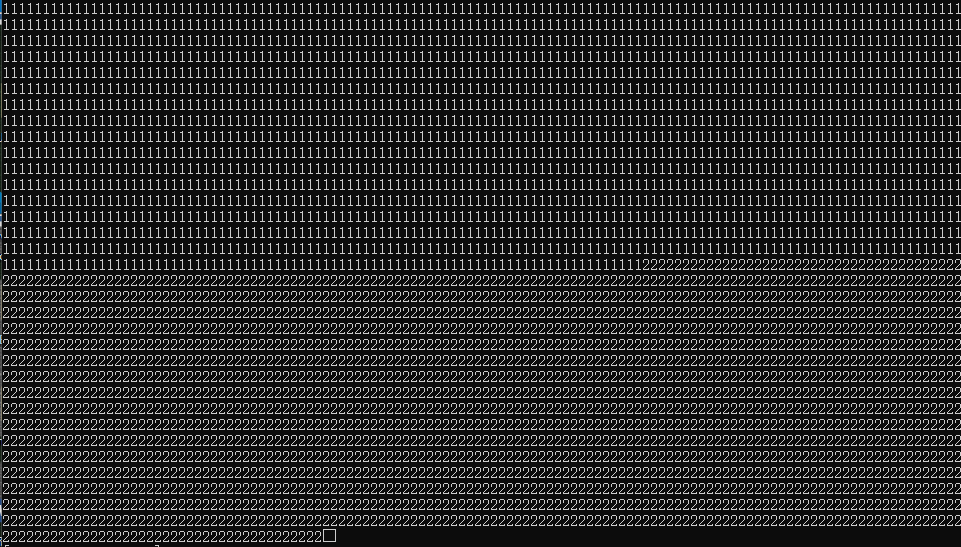
**实验2-2 进程的管道通信**

1. **实验要求：**

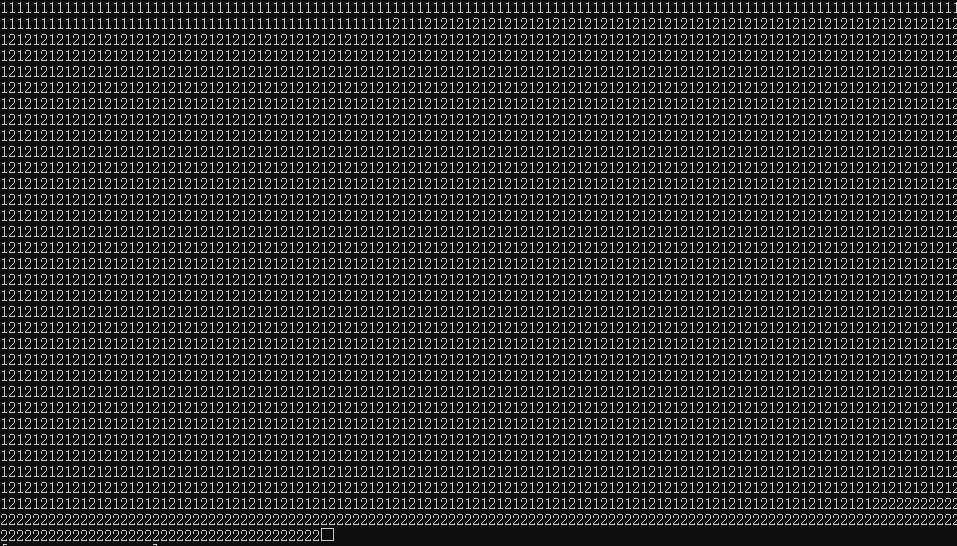


注：把信息改为2000个1，2000个2！

运行结果如下：



* **3） 修改程序并运行，体会互斥锁的作用，比较有锁和无锁程序的运行结果，并解释之。**

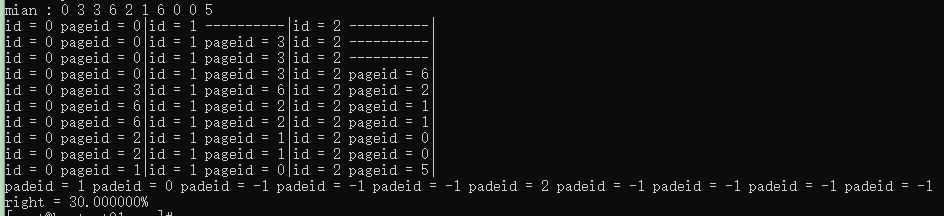


没有互斥锁，两个子进程没有顺序的往文件中写入数据

**实验2-4 页面的替换**

* **模拟实现FIFO算法，LRU算法**

FIFO算法运行结果如下：



LRU算法运行结果如下：

