山东大学 软件 学院

**操作系统** 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202000300125 | 姓名：贾星宇 | | 班级：2020级5班 |
| 实验编号：实验二 | | | |
| 实验题目：进程通信实验 | | | |
| 实验学时：4 | | 实验日期：2022年4月13日 | |
| 实验目的：  通过 Linux 系统中管道通信机制，加深对于进程通信概念的理解，观察和体验并发进程间的通信和协作的效果，练习利用无名管道进行进程通信的编程和调试技术。 | | | |
| 硬件环境：  宿主机：lntel(R) Core(TM) i5-10210U CPU @ 1.60G Hz 2 .1 1 G Hz  虚拟机：ubuntu-20.04.1-desktop-amd64 | | | |
| 软件环境：  宿主机：Windows 10 家庭中文版  虚拟机：Code::Blocks | | | |
| 实验步骤与内容：  1.分析并编写示例实验代码，并运行:      运行结果如下：    可以看到以上程序的执行中父子进程合作将整数 X 的值从 1 加到了 10。  2.分析独立实验要求：  设有二元函数 f(x,y) = f(x) + f(y)  其中： f(x) = f(x-1) \* x (x >1)  f(x)=1 (x=1)  f(y) = f(y-1) + f(y-2) (y> 2)  f(y)=1 (y=1,2)  请编程建立 3 个并发协作进程，它们分别完成 f(x,y)、f(x)、f(y)  回答疑问：最后得到f（x,y）时候x和y满足x==y，而且为了避免无限循环，添加限制条件让x与y循环到9.        运行产生结果：    发现完成了f(x) f(y)和f(x,y)的运算，符合题目要求 | | | |
| 结论分析与体会：  【个人对示例程序的理解】  程序运行开始，创建两个管道pipe1，2并且对每个管道，0为读，1为写。如果管道创建失败，则返回报错。否则开始创建子进程。如果创建失败则报错，否则开始执行下面的代码：  子进程和父进程并发执行，首先看子进程：  子进程复制了fork后面的代码，并且进入pid==0的if判断语句内运行  子进程读管道1，写往管道2，所以关掉1的1和2的0；  随后开始读管道1的值，由于此时管道1没有值，所以子进程被阻塞  来到父进程：  父进程进入最后一个else判断分支：  父进程读管道2，向管道1写，所以关掉2的1和1的0；  由于管道被读取之后便成为空管道，所以管道2为空  首先父进程令x=1，并且把x写往管道1，管道1有了值【1】  随后父进程开始读管道2的值，由于此时管道2没有值，所以被阻塞  回到子进程  由于管道1中被写入了值【1】，所以子进程可以读管道1的值并且打印出来；  由于管道被读取之后便成为空管道，所以管道1成为空  随后将读到的值加1并送入管道2，管道2拥有值【2】  由于x=2，所以循环继续。  子进程再次回到读管道，由于刚刚读取了管道1的值所以为空，被阻塞  回到父进程  父进程由于管道2被子进程写入了【2】，因此可以读出来，随后打印此值并且将x加到3，再次循环写入管道1；由于刚刚2被读出来了，所以被阻塞。  【总结】  管道的特点为每次只可以写/读，而且如果为空，读进程阻塞；为满，写进程阻塞；读完后管道变为空。  此程序，首先父进程写入管道1值【1】，子进程读出管道1的值【1】并自增，写入管道2【2】，父进程读出管道2【2】并自增，写入管道1【3】，子进程读出管道1【3】并自增，写入管道2【4】，父进程读出管道2【4】并自增……  【独立实验撰写思路】  首先创建两个管道1、2，以及变量x、y（初始值为1）；然后检查两个管道是否被创建成功。  创建成功后父进程0便开始创建子进程1，返回值pid>=0说明进程被创建成功。  开始进程1  进入pid==0的判断分支语句中。首先关闭管道1、2的读和管道1的写，然后设置fx初始值为1，进入循环语句  当x是1时，fx是1；否则fx=fx\*x；（等号右边的fx其实是f（x-1））  随后便将fx值写入管道1，再进行下一次循环，直到算出f（9）。  然后关掉管道1的写。  对进程0来说，它进入下方的else判断分支语句。首先再次创建一个子进程进程2。如果创建成功——  开始进程2  关掉管道1的读写和管道2的读，像进程1一样写fy到管道2，直到写完f（9）.  在子进程1、2向管道写数据的同时，父进程进入最后一个else判断语句  开始父进程  关掉管道1、2的写功能，同时开始读，每次读管道1和管道2的一组int大小的数据，并且存入x和y，将他们加起来并打印，实现f（x）+f（y）  \*管道的行为类似于队列数据结构，是FIFO(先进先出)，而且读写的大小在这里不必匹配。我们一次可以写入512个字节，而在管道另一端一次读取1个字节。  总结：父进程0完成f(x)+f(y)，子进程1完成f（x）并存入管道1，父进程完成f（y）并存入管道。  【疑惑总结】  1.对于独立实验中并发的理解  由于父子进程是并发执行的，分析这三个进程的执行顺序：  子进程1不断向管道1中写入f（x）的值，由于没有阻塞因素影响，所以它会一直循环下去直到x==9。  父进程在关掉两个管道的写端口后开始读，读第一个管道没有问题，因为里面有值；但读第二个管道的时候需要等待子进程2向管道2中写入f（y）的值，然后才能读到并且加起来。因此形成了实验结果中的交替进行的场面。  2.父子进程close问题  它们close只是close了自己进程对管道的读写控制，并不影响其他子进程，因为fork系统调用会复制一份给子进程。同时close是为了让进程结束后可以正常结束，而不是出于挂起状态，防止无用功的产生。 | | | |