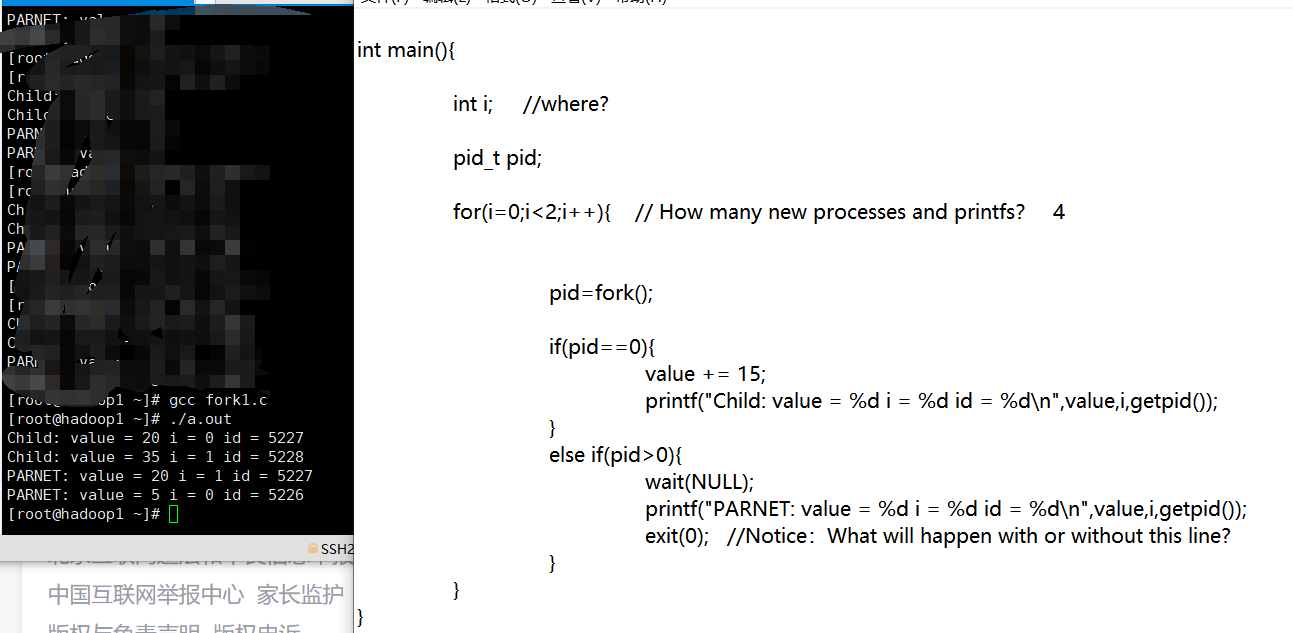
1. fork()相关



可以看到我在这里输出了value,i和进程id

* 1. where is i and value?

可以看出输出命令提示符的进程与多进程的输出之间有并发，输出结果总体顺序不确定，但是阅读完相关资料后才发现，其顺序很明显是5226，5227，5227，5228.每一行的value与i所属的进程均为该行所在的id数，如第一行value=20，i=0，id=5227，是第一个循环的子进程，以此类推，第四行是第一个循环的父进程，第二行是第二个循环的子进程，第三行是第二个循环的父进程。i和value所在进程id为该行末尾，这是我理解的“where”指的是所属进程id。

链表:5226->5227->5228

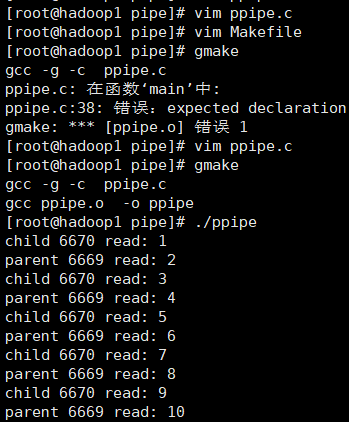
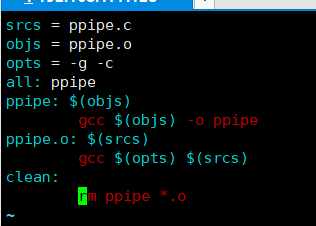
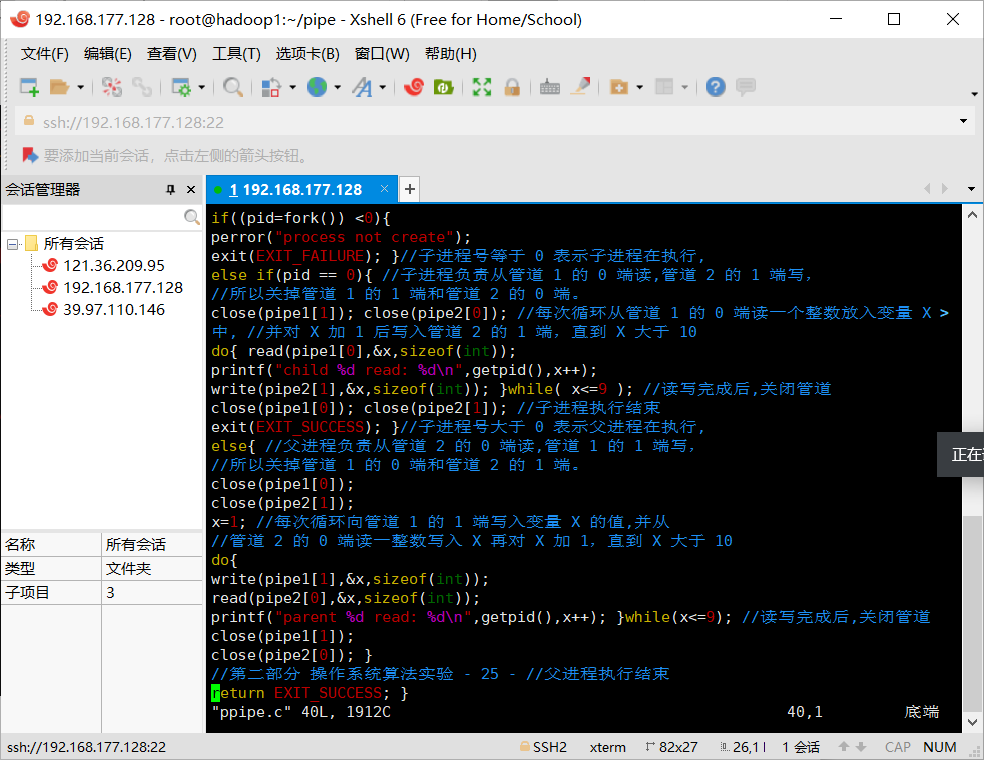
* 1. How many new processes and printfs?

不难发现，出现了四行，也就是打印出了四下。有三个进程，但是要说新进程，如果说没有fork(),那么应该是两个进程，一次循环代表一次进程，但是用了fork(),就应该是多了一个新进程, id为5226，5227，5228.

* 1. What will happen with or without this line?

会继续一遍循环。为什么？在p5227和p5228执行完第二个循环后，main函数就该退出了，也即进程该死亡了，因为它已经做完所有事情了。p5227和p5228死亡后，p5229，p5230就没有父进程了，这在操作系统是不被允许的，所以p5227和p5228的父进程就被置为p1了，p1是永远不会死亡的。

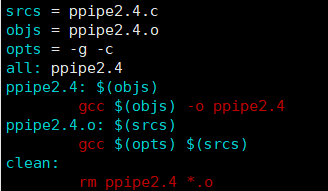
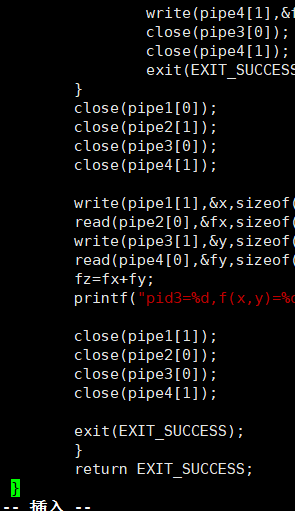
2.管道通信



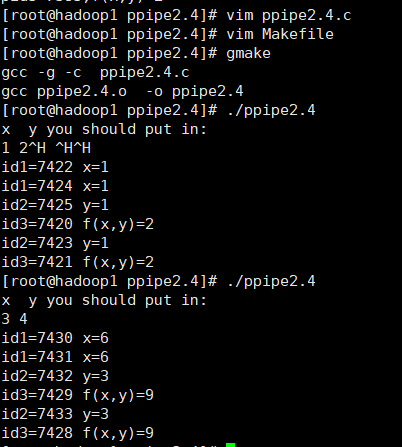
可以看出来程序通过父子进程合作使最终正确输出.

子进程负责从管道1的0端读,管道 2 的 1 端写,父进程负责从管道 2 的 0 端读,管道 1 的 1 端写,子进程每次循环从管道 1 的 0 端读一个整数放入变量X中,并对 X 加 1 后写入管道 2 的 1 端，直到 X 大于 10;父进程每次循环向管道 1 的 1 端写入变量X的值,并从管道2的0端读一整数写入X再对X加1,直到X大于10.

3.附加题独立实验



ppipe2.4.c makefile



输出结果