

假设if中的两个fork分别记录为fork1和fork2;

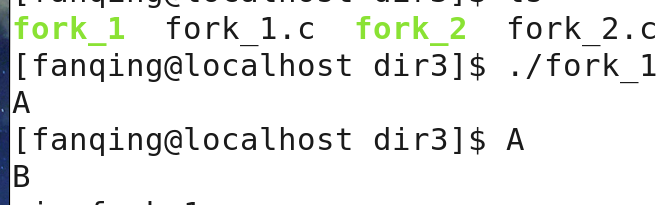
当进行到fork1时，返回值为其子进程的PID不为零直接执行父进程中的A并开辟子进程不会在父进程中进行fork2的判断，但其子进程中的fork1的返回值为0，所以还要判断继承过来的fork2，此时fork2的返回值是其子进程PID所以也不为零，直接执行输出A并开辟当前进程的子进程，在这个进程中因为其fork1和fork2的返回值都为0，所以其执行else中输出B。

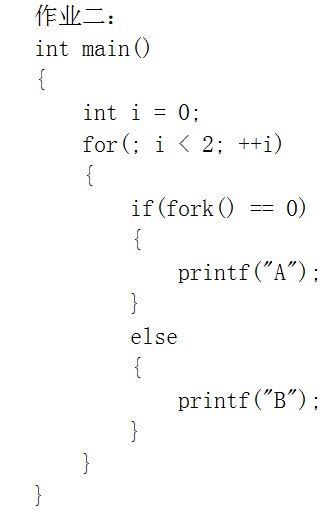
Fork返回值不为0 Fork2返回值不为0 fork1与fork2返回值都为零

输出A

子进程 输出A 子进程 输出B

执行结果





分析：i=0时，先判断Fork然后直接开辟子进程，在父进程中的Fork不为0，所以在其缓冲区中放入B，而其子进程中fork=0，所以直接在其缓冲区中放入A，因为其继承父进程时父进程的缓冲区中并没有东西，所以只有一个A；在子进程中i++为1时再次判断一次Fork，再开辟一个子进程（子孙进程），但当前子进程所要继承的缓冲区中已有一个A，所以将其继承过来，并且在其父进程中输出B，在这个子进程中输出A；返回到最初的父进程，i++为1，然后再一次fork开辟一个子进程，在这个子进程中要继承父进程缓冲区中的B，并且自身输出A，其父进程再输出一个B。即共4A4B.

B i=0时开辟的子进程 A

B B

I=1时开辟的子进程 I=1时开辟的子进程

B A

A A

执行结果

