实验四 进程通信

**学号： 20020006107 姓名： 王义钧 年级： 2020**

**一．PIPE系统调用的基本使用方法**

**1.具体要求与步骤**

1. 编写一C/C++语言程序（程序名为basicpipe.c/basicpipe.cpp），使用系统调用pipe()实现基本的消息传递。
2. 多次连续反复运行这个程序，观察屏幕显示结果，试简单分析其原因。
3. 可以使用实验报告模板中所推荐的代码实现，但是要求为代码添加注释，对代码关键逻辑步骤进行解释。在代码头部添加如代码 1所示式样的头部版权声明。使用星号、井号、等号、破折号等各类符号对版权声明添加边框，并拼出19os的式样。

**2.实验结果截图**

**结果分析：**

在本次实验中，要实现父进程与子进程的通信过程，通过pipe()建立管道，并将文件描述词由参数 filedes 数组返回。filedes[0]为管道里的读取端，所以pipe用read调用的filedes[1]则为管道的写入端。

父进程先通过write( filedes[1], s, sizeof(s) )，将内容字符串写入管道写入端filedes[1]中，；在子进程中，通过read(filedes[0], buf, sizeof(buf))读取管道读取缓冲区的内容。

经过程序的多次运行，其运行结果为：父进程先获得处理机组员，打印：“This is in the father process,here write a string to the pipe”，并将“Hello world , this is write by pipe.”写入filedes[1]，然后进入子进程，先打印“This is in the child process,here read a string from the pipe.”，然后读取filedes[0]，再打印“Hello world , this is write by pipe.”。

**尝试情况：**

假设在父进程，使用sleep(3)让其休息3s，这样子进程优先获得处理机资源，打印“This is in the child process,here read a string from the pipe.”，但是从管道读不到信息，便阻塞自己，执行父进程完成写管道的操作后，子进程再向屏幕输出。



图 1 实验代码

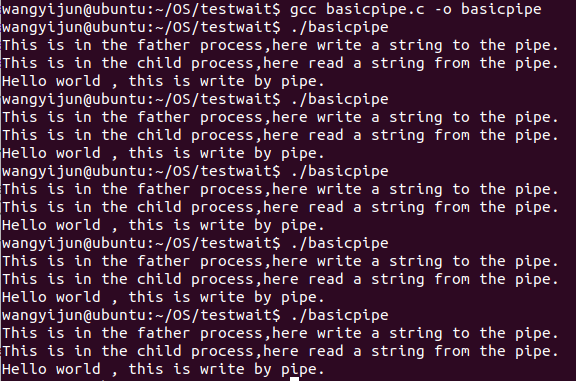


图 2 父子进程通信

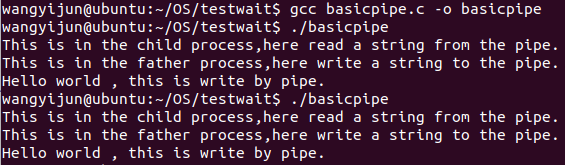


图 3 子进程阻塞 等待父进程

**二. PIPE实现进程间通信**

**1.具体要求与步骤**

1. 编写一C/C++语言程序（程序名为testpipe.c/testpipe.cpp），使用系统调用pipe( )尝试在多个进程间进行通信。
2. 要求实现的功能是，用管道来实现父子进程间通信。子进程向父进程发送字符串“[进程名称] is sending a message to parent!”；父进程则从管道中读出子进程发来的消息，并将其显示到屏幕上，然后终止。多次连续反复运行这个程序，观察屏幕显示结果的顺序，试简单分析其原因。
3. 可以使用实验报告模板中所推荐的代码实现，但是要求为代码添加注释，对代码关键逻辑步骤进行解释。在代码头部添加如代码 1所示式样的头部版权声明。使用星号、井号、等号、破折号等各类符号对版权声明添加边框，并拼出19os的式样。

**2.实验结果截图**

**结果分析：**

父进程创建了子进程1和子进程2，创建了文件描述词fd，通过pipe进行通信。

在子进程1中，首先将字符串“Child Process 1 is sending a message!”送入缓冲区S，再利用S写入管道，随后sleep 3s。

在子进程2中，与子进程1大部分比较类似，不同的地方为字符串“Child Process 2 is sending a message!”。

在父进程中，首先利用wait等待两个子进程都完成工作，再连续两次从管道的输出端口获取信息并向屏幕输出。在多次执行的过程中，由于两个子进程执行的先后顺序不相同，输入管道中的字符串的顺序也不相同，故父进程中输出来自两个子进程的信息顺序也不相同。

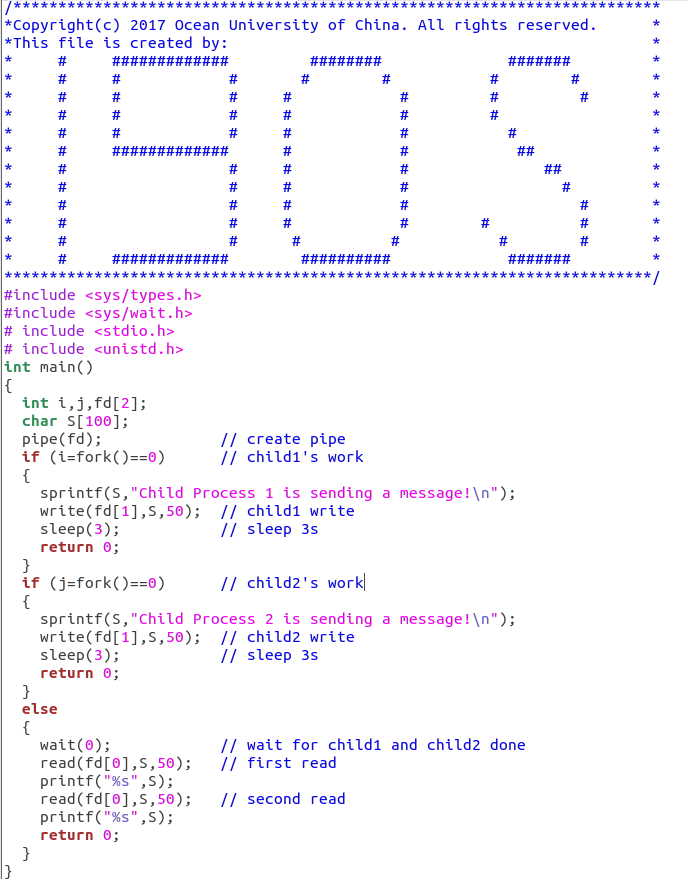


图 4 实验代码

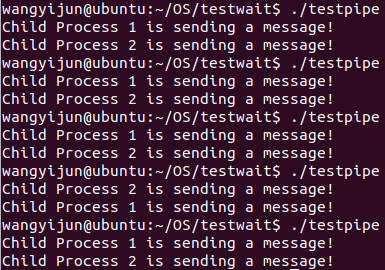


图 5 两个子进程交叉顺序