电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2015220103022

姓 名 张健顺

（实验） 课程名称 **操作系统基础**

理论教师 杨霞

实验教师 杨霞

**电子科技大学教务处制表**

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：张健顺 学号：2015220103022 指导教师：杨霞**

**实验地点：信软楼西304 实验时间：2017.04.27**

**一、实验名称：利用管道实现两个进程的通信**

**二、实验学时：4**

**三、实验目的：**

1. **熟悉Linux下的应用程序开发**
2. **熟悉Linux的进程控制原语的使用**
3. **掌握Linux操作系统的进程间通信机制管道的使用。**
4. **掌握Linux操作系统中父进程与子进程的同步。**

**四、实验原理：**

首先创建两个子进程，注意Linux下使用fork()函数创建进程的方法。父进程和两个子进程间需要同步，使用waitpid（）函数实现父进程等待子进程运行完毕后从管道中读取数据并打印，只有子进程将数据写入管道后，父进程才能够执行打开管道操作。

由于fork函数让子进程完整地拷贝了父进程的整个地址空间，所以子进程都有管道的读端和写端。所以在相关进程中最好关掉不用的那一端。根据要求，“父进程先接收子进程P1发来的消息，然后再接收子进程P2发来的消息。”存在两个同步问题，两个子进程和父进程之间（先子写后父读）同步、子进程1和子进程2之间（先1写，再2写）

**五、实验内容：**

在Linux系统中使用系统调用fork()创建两个子进程，使用系统调用pipe()建立一个管道，两个子进程分别向管道各写一句话：

Child process 1 is sending a message!

Child process 2 is sending a message!

而父进程则从管道中读出来自于两个子进程的信息，显示在屏幕上。然后分别结束两个子进程的运行。

要求：

1.父进程先接收子进程P1发来的消息，然后再接收子进程P2发来的消息。

2.在Linux平台下实现。

**六、实验器材（设备、元器件）：**

1. 学生每人一台PC，安装Windows8操作系统。
2. 个人PC安装VMware虚拟机和Ubuntu系统。

**七、实验步骤：**

步骤1：创建一个管道。

步骤2：创建子进程1，向管道中写入“Child process 1 is sending a message!”，并做好跟父进程的同步执行。

步骤3：创建子进程2，向管道中写入“Child process 2 is sending a message!”，并做好跟父进程的同步执行。

步骤4：父进程从管道中读取数据，并打印输出。

**八、实验结果与分析（含重要数据结果分析或核心代码流程分析）**

核心思想：

使用管道实现两个进程之间的通信，管道有两端，读端和写端，两个进程之间的通信则可以采用分别调用读端和写端来实现，多个子进程分别向缓冲区中写入信息。同时为了保证先写后读，父进程需要等待子进程的执行。将会使用到相关的wait函数。进程将数据写入管道后，父进程才能够执行打开管道操作。在读写数据时最好是将不是用的一端给关闭掉。

程序源码：

#include <stdio.h>

#include <unistd.h>

#include <string.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

int main()

{

//fileDest[0] is the reader of the pipe

//fileDest[1] is the writer of the pipe

int fileDest[2];

//define the buffer

char write\_buffer[200],read\_buffer[200];

int length;

//pipe() is to create pipe

pipe(fileDest);

pid\_t child1;

pid\_t child2;

//use fork() to creat progress

child1 = fork();

child2 = fork();

if(child1 == 0)

{

//close the reader of the pipe

close(fileDest[0]);

//sprintf(var,"...") is to write the ... to the var

sprintf(write\_buffer,"Child process 1 is sending a message!\n");

length = strlen(write\_buffer);

//writes the data (length) from the write\_buffer to fileDest[1](the writer of the pipe)

write(fileDest[1],write\_buffer,length);

return 0;

}

else if(child1 > 0)

{

//father progress waits the child1 progress to run

waitpid(child1,NULL,0);

if(child2 == 0)

{

close(fileDest[0]);

sprintf(write\_buffer,"Child process 2 is sending a message!\n");

length = strlen(write\_buffer);

write(fileDest[1],write\_buffer,length);

return 0;

}else if(child2 > 0)

{

waitpid(child2,NULL,0);

close(fileDest[1]);

//read the data (length) from the write\_buffer to fileDest[0](the reader of the pipe)

read(fileDest[0],read\_buffer,length);

printf("%s\n",read\_buffer);

return 0;

}

}

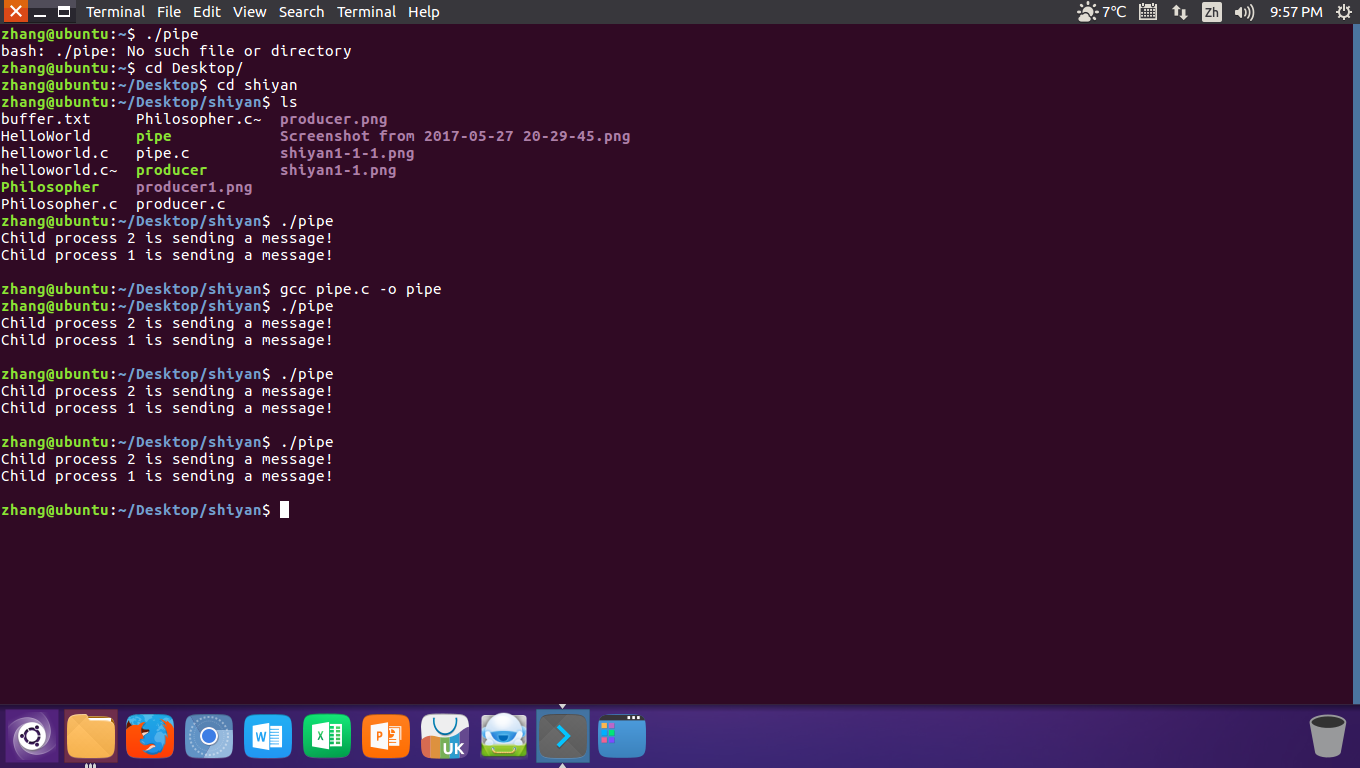
return 0;

}

程序分析：

在完成相关管道、读写缓冲区、子进程的创建以后，判断子进程创建的状态（是否成功），若子进程1创建成功，则向管道中写入相关数据，又因为父进程和两个子进程间需要同步，使用waitpid()函数实现父进程等待子进程运行完毕后从管道中读取数据并打印，只有子进程将数据写入管道后，父进程才能够执行打开管道操作。在读写数据时，为了避免占用相同的地址空间，在读写时关闭管道中不需要使用的一端。

结果截图：



父进程从管道中读出消息

**九、总结及心得体会：**

通过本次实验，了解了linux下的进程创建方法，以及一些进程操作的相关指令。同时也感受到了进程之间相互通信的应用场景，而管道又是一种进程间相互通信的方式。理解了管道的含义以及管道的使用策略。与此同时，加深了对进程间的同步问题的理解。除此之外，也回顾了C语言的读写操作的相关知识。自己也尝试着实现linux和windows文件共享，了解了一些linux下的基本的局域网配置的命令原语和相关知识。

1. **对本实验过程及方法、手段的改进建议：**

希望能通过更多的创建进程的方式（函数）来创建进程，同时理解他们之间的异同以及各自的应用场景。对于进程间的通信和同步，希望能有一些更为生动实际的例子来进行详细阐述。

**报告评分：**

**指导教师签字：**