## 实验五；管道通信

1. 阅读以下程序，编译并运行程序，分析程序执行过程和结果，注释程序主要语句。

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include ”apue.h” //err\_quit函数的头文件

main()

{

int filedes[2];

char buffer[80];

/\* pipe()会建立一个管道，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，filedes[1]则为管道的写入端。若成功则返回零，否则返回-1。\*/

if(pipe(filedes)<0) //如果创建管程失败

err\_quit(“pipe error”); //err\_quit：错误处理函数

if(fork()>0){ //创建子进程成功

//进入父程序

char s[ ] = “hello!\n”;

close(filedes[0]); //关闭管道的读取端

write(filedes[1],s,sizeof(s)); //把字符串s写入管道。

close(filedes[1]); //关闭管道的写入端

}

else{

//进入子程序

close(filedes[1]); //关闭管道的写入端

read(filedes[0],buffer,80);//从管道读取端读取80字节数据并保存到buffer里

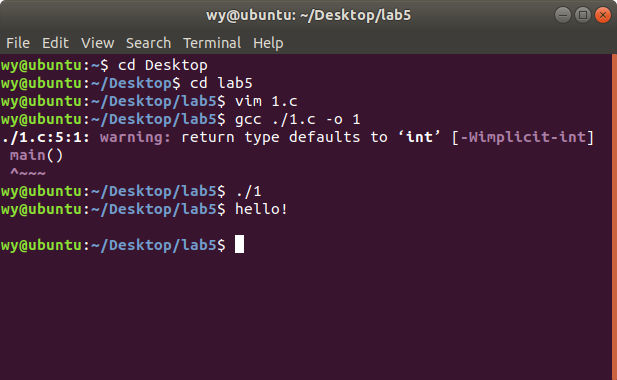
printf(“%s”,buffer);

close(filedes[0]); //关闭管道的读取端

}

}

实验结果：



1. 阅读以下程序，编译并运行程序，分析程序执行过程和结果，注释程序主要语句。

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

main()

{

char buffer[80];

int fd;

char \*FIFO=”2-2”;

unlink(FIFO);//如果有FIFO文件，就先删掉

mkfifo(FIFO,0666); //创建一个FIFO文件，FIFO是管道名，0666是权限

if(fork()>0){ //创建子程序

//进入父程序

char s[ ] = “hello!\n”;

fd = open (FIFO,O\_WRONLY);以只写方式打开FIFO文件

write(fd,s,sizeof(s)); //把s写入FIFO文件

close(fd); //关闭FIFO文件

}else{

//进入子程序

fd= open(FIFO,O\_RDONLY); //以只读方式打开FIFO文件

read(fd,buffer,80); //从FIFO文件里读取80字节到buffer指针所指的内存中

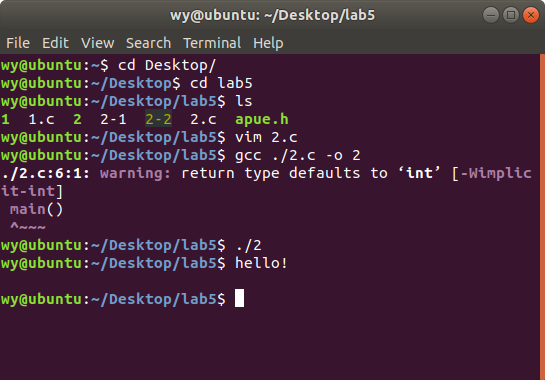
printf(“%s”,buffer);

close(fd); //关闭FIFO文件

}

}

实验结果：



1. 阅读以下程序，编译并运行程序，分析程序执行过程和结果，注释程序主要语句。

#include<stdio.h>

main()

{

FILE \* fp; //定义文件指针

char buffer[80];

fp=popen(“cat /etc/passwd”,”r”);

//用只读的方式打开/etc/passwd里的文件将其写入到fp中

/\*popen()会调用fork()产生子进程，然后从子进程中调用/bin/sh -c来执行第一个参数表示的指令。第二个参数里，“r”代表读取，“w”代表写入，依照该值，popen()会建立管道连到子进程的标准输出设备或标准输入设备，然后返回一个文件指针。随后进程便可利用此文件指针来读取子进程的输出设备或是写入到子进程的标准输入设备中。\*/

fgets(buffer,sizeof(buffer),fp); //获取字符串并保存到buffer中

printf(“%s”,buffer);

pclose(fp); // pclose()用来关闭由popen函数所建立的管道及文件指针。

}

实验结果：

