# 嵌入式软件开发技术与工具·小组报告

组员：隆非非 卓佳佳 刘政

## **一、实验目的**

#### **1.学会创建多进程程序，理解fork()，vfork() 创建的子进程的特点。**

#### **2.学会使用有名管道和无名管道实现进程间通信。**

#### **3.学会使用信号机制实现进程间通信。**

## **二、实验内容**

#### **1.编写fork()，vfork()多进程程序，通过输出结果比较所创建的子进程的不同之处。**

#### **2.编写fork()多进程程序，进程间模拟信号通信，包括信号发送、捕捉与处理。**

#### **3.编写fork()多进程程序，进程间实现有名管道和无名管道通信。具体要求如下：**

(1)编写1个服务器程序server.c和2个客户端程序client1.c, client2.c ，服务器和客户端之间采用FIFO管道通信； (2)服务器和客户端各自创建自己的子进程，父进程负责 完成FIFO管道通信，子进程负责输出接收到的信息， 父子进程间采用无名管道通信。

## **三、实验过程与结果**

**1 编写fork()，vfork()多进程程序，通过输出结果 比较所创建的子进程的不同之处。**

**先是fork()**

#include <unistd.h>

#include <stdio.h>

int main()

{

int num=1;

pid\_t fpid;

fpid = fork();//或者vfork

if (fpid < 0)

printf("error in fork!");

else if (fpid == 0) {

printf("i am kid, my id is %d\n", getpid());

num++;

printf("num is %d\n",num);

\_exit(0);

}

else {

printf("i am dad, my id is %d\n", getpid());

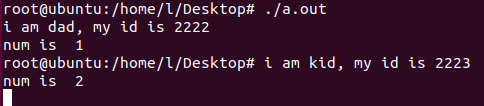
printf("num is %d\n",num);

}

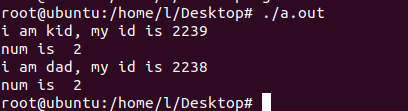
return 0;

}

结果：



然后是**vfork()**



总结：

fork 是 创建一个子进程，并把父进程的内存数据copy到子进程中。

vfork是 创建一个子进程，并和父进程的内存数据share一起用。

**2 编写fork()多进程程序，进程间模拟信号通信， 包括信号发送、捕捉与处理。**

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

#include<string.h>

int main(void){

pid\_t pid;

int fd[2],nbytes;

char string[100];

char readbuffer[80];

int \*write\_fd = &fd[1];

int \*read\_fd = &fd[0];

printf("Please input data:");

scanf("%s",string);

int result = pipe(fd);

if(-1 == result)

{

perror("pipe");

return -1;

}

pid=fork();

if(-1 == pid)

{

perror("fork");

return -1;

}

else if(0 == pid)

{

printf("this is child %d\n", getpid());

close(\*read\_fd);

result = write(\*write\_fd,string,strlen(string));

return 0;

}

else

{

printf("this is parent %d\n", getpid());

close(\*write\_fd);

nbytes = read(\*read\_fd,readbuffer,sizeof(readbuffer)-1);

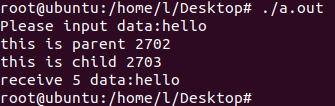
printf("receive %d data:%s\n",nbytes,readbuffer);

}

return 0;

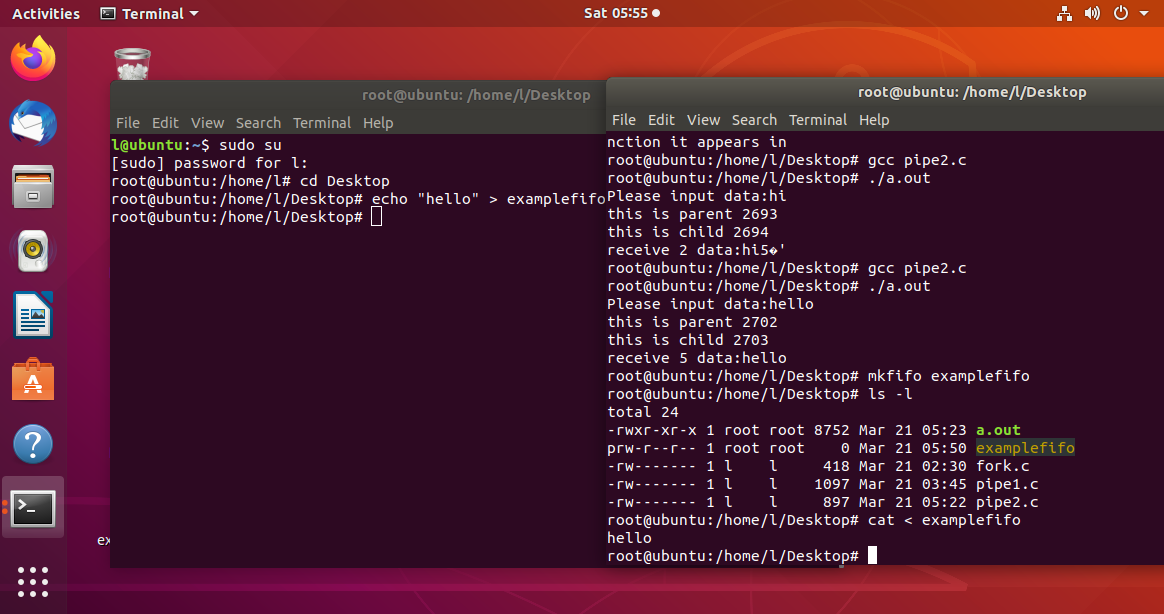
}

结果：

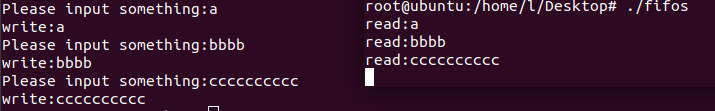


**3 编写fork()多进程程序，进程间实现有名管道和 无名管道通信。**

先尝试在shell中使用fifo

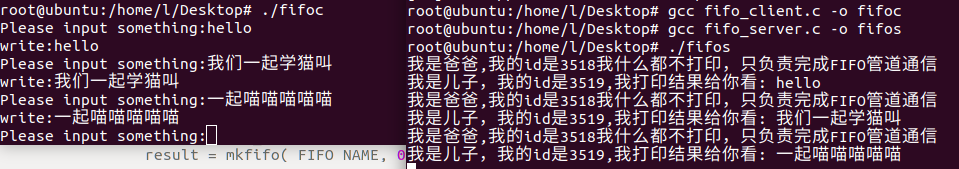


再尝试在程序中使用fifo管道：



最后增加上fork和pipe部分：

最终结果：



代码：

Server：

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/stat.h>

#define FIFO\_NAME "myfifo"

int main()

{

int result;

int fifo\_fd;

char buffer[32];

int buffer\_len;

pid\_t pid;

int fd[2],nbytes;

char readbuffer[80];

int \*write\_fd = &fd[1];

int \*read\_fd = &fd[0];

int my\_result = pipe(fd);

if(-1 == my\_result)

{

perror("pipe");

return -1;

}

pid=fork();

if(-1 == pid)

{

perror("fork");

return -1;

}

else if(0 == pid)

{

do

{

close(\*write\_fd);

nbytes = read(\*read\_fd,readbuffer,sizeof(readbuffer)-1);

printf("我是儿子，我的id是%d,我打印结果给你看: %s\n",getpid(),readbuffer);

}

while( memcmp( buffer, "close", 5 ) != 0 );

return 0;

}

else

{

/\* 先删除之前可能遗留的管道文件，然后再次创建它 \*/

//fifo file

unlink( FIFO\_NAME );

result = mkfifo( FIFO\_NAME, 0777 );

if( result != 0 )

{

printf( "error:can't create a fifo.\n" );

return 1;

}

/\* 以只读的方式打开管道文件 \*/

fifo\_fd = open( FIFO\_NAME, O\_RDONLY );

if( fifo\_fd < 0 )

{

printf( "error:can't open a fifo.\n" );

return 1;

}

close(\*read\_fd);

/\* 循环从管到文件中读取数据 \*/

do

{

//memset是计算机中C/C++语言初始化函数。作用是将某一块内存中的内容全部设置为指定的值

//这个函数通常为新申请的内存做初始化工作。

memset( buffer, 0, 32 );

buffer\_len = read( fifo\_fd, buffer, 31 );

buffer[buffer\_len] = '\0';//str end flag

printf("我是爸爸,我的id是%d我什么都不打印，只负责完成FIFO管道通信\n", getpid());

write(\*write\_fd,buffer,strlen(buffer));

}

while( memcmp( buffer, "close", 5 ) != 0 );

close( fifo\_fd );

unlink( FIFO\_NAME );

}

return 0;

}

Client：

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/stat.h>

#define FIFO\_NAME "myfifo"

int main()

{

int result;

int fifo\_fd;

char buffer[32];

int buffer\_len;

/\* 判断一下管道文件是否存在，不存在就退出程序 \*/

result = access( FIFO\_NAME, F\_OK );

if( result == -1 )

{

printf( "error:can't find the fifo.\n" );

return 1;

}

/\* 以只写的方式打开管到文件 \*/

fifo\_fd = open( FIFO\_NAME, O\_WRONLY );

if( fifo\_fd < 0 )

{

printf( "error:can't open a fifo.\n" );

return 1;

}

/\* 循环向管到文件中写入数据 \*/

do

{

memset( buffer, 0, 32 );

printf( "Please input something:" );

scanf( "%s", buffer );

buffer\_len = write( fifo\_fd, buffer, strlen( buffer ) );

printf( "write:%s\n", buffer );

}

while( memcmp( buffer, "close", 5 ) != 0 );

close( fifo\_fd );

unlink( FIFO\_NAME );

return 0;

}

## **实验总结**

掌握了fork、pipe和fifo的基本用法

## **代码**

见文中