**操作系统实验**

**消息通信扩展实验**

|  |  |
| --- | --- |
| **学号：** | **2220193626** |
| **姓名：** | **李子毅** |
| **选课序号：** | **51** |

**消息通信扩展实验**

1. 实验名称：消息通信扩展实验
2. 实验目的:学习软中断和消息队列的相关理论，并熟悉linux下与操作系统相关的库的使用。
3. 实验内容：通过消息队列来发送和接收消息，通过软中断来控制发送方发送消息的频率，实现了消息的一发一收，在设计过程中，对于原有的消息队列的示例程序进行了修改，将服务器端设置为主进程，而客户端设置为子进程，用以通过getpid和getppid得到两个进程的进程号，这样就能够实现进程间的信号发送与关联，当客户端发送一条信息后进入阻塞状态，直到服务器端读取完消息，然后发送信号给客户端，客户端才能继续发送消息。

4.#include <stdio.h>

#include <sys/types.h>

#include <sys/msg.h>

#include <sys/ipc.h>

#include <stdlib.h>

#include<string.h>

#include<unistd.h>

#include<sys/wait.h>

#include<signal.h>

#define MSGKEY 75

//定义消息队列结构

struct msgfrom{

long mtype;//消息类型

char msgtext[1030];//存放消息内容

}msg;

int msgqid,i,j,wait\_mark;

void waiting()

{

while(wait\_mark!=0);

}

void stop()

{

wait\_mark=0;

}

//客户端

void CLIENT()

{

int i;

char string\_i[5];//存放信息

msgqid=msgget(MSGKEY,0777);//0777：文件打开,该操作是打开消息队列

j=getpid();//获得服务器端pid

for(i=10;i>=1;i--)

{

//发送消息

msg.mtype=i;

printf("客户端发出消息\n");

sprintf(msg.msgtext,"消息内容为：");

sprintf(string\_i,"Hello %d",i);

strcat(msg.msgtext,string\_i);

strcat(msg.msgtext,"\n");

msgsnd(msgqid,&msg,1030,0);

wait\_mark=1;

signal(17,stop);//关联信号

waiting();//等待服务器端处理

}

exit(0);

}

void SERVER()

{

msgqid=msgget(MSGKEY,0777|IPC\_CREAT);

do

{

msgrcv(msgqid,&msg,1030,0,0);

printf("服务器收到来自 message %d的消息\n",msg.mtype);

printf("%s",msg.msgtext);

kill(j,17);

}while (msg.mtype!=1);

msgctl(msgqid,IPC\_RMID,0);

exit(0);

}

int main()

{

//主进程为服务器，子进程为客户端

while((i=fork())==-1);

if(i>0)

{

SERVER();

}

else

{

CLIENT();

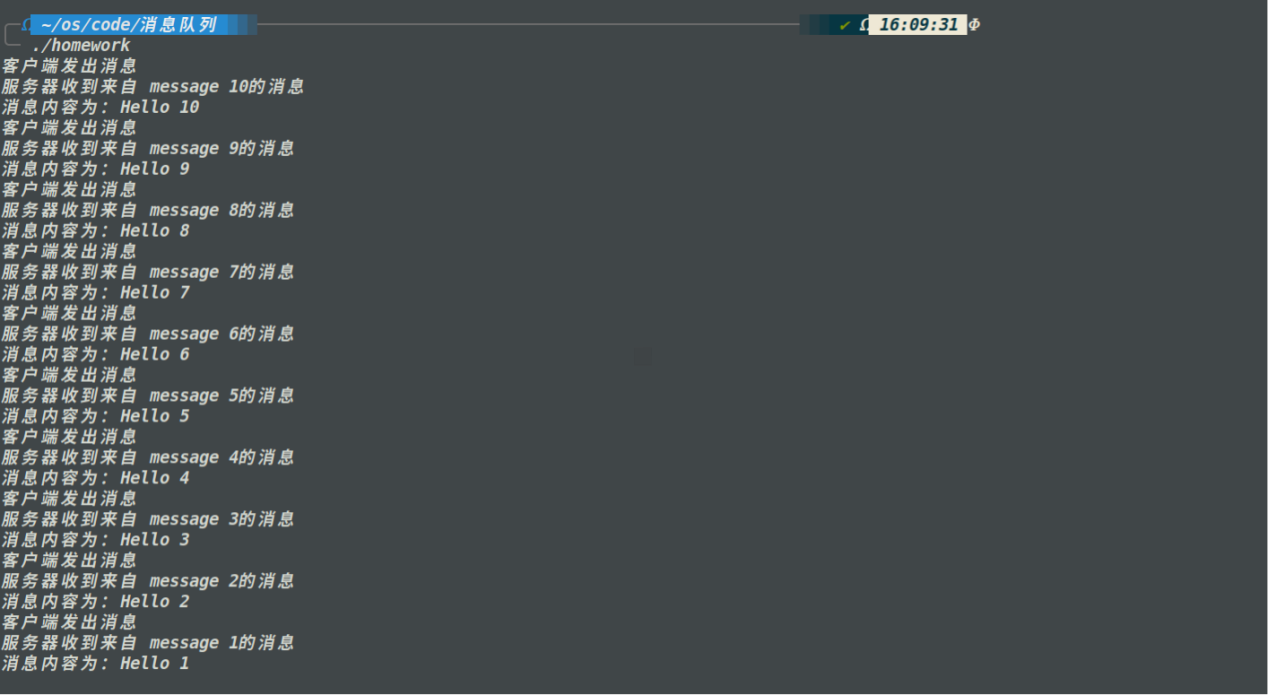
}

wait(0);

wait(0);

}

5.运行结果：



1. 问题及解决方法：
2. 原程序中，客户端和服务器端均为主进程的子进程，是两个并列的关系，这样的话就很难将他们两个关联起来，所以我将服务器端修改为主进程，客户端设置为服务器端的子进程，再定义一个共享的全局变量，这个变量的值为客户端进程中调用getpid之后的返回值，当然，这里意识到如果是两个并列的进程也可以用这个方法，但是主从进程的好处就在于如果在一端使用getpid和getppid就可以获得两个进程所有的进程号，并且后续进行其他操作也很方便。
3. 发送和关联信号的时机：客户端应该在一开始就关联信号，而服务器端由于有一个固定的读取方式，如果消息队列里面没有消息就不读取，所以无论怎么调度，都会是客户端的程序先发送，只是服务器端如何发送信号的问题，这里我选择先读取，再发送关联信号，这样的话逻辑就是：客户端先发送第一条消息并将信号与stop函数关联->服务器端读取，调用kill函数->客户端收到信号，继续发送。这里还有一个坑就是signal函数里的信号的值，与不同的操作系统有关，ubuntu系统的话，建议使用16或者17.
4. 写完代码后采用gcc命令编译运行，出现找不到input file的情况，解决方式为：是每次写完代码之后没有保存所致，将IDE设置为自动保存，并每次需要重新编译运行新的代码，出现问题时采用gdb调试