# 操作系统实验

消息通信扩展实验

| | |
|---|---|
| 学号： | **2220191195** |
| 姓名： | 李毓琪 |
| 选课序号： | **4** |

## 一、 实验名称：

消息通信扩展实验

## 二、 实验目的：

掌握 Linux 进程软中断通信的基本原理和实现方法。

掌握消息队列的通信机制以及原理，掌握通信相关系统调用的使用方法。

相关理论知识：

软终端信号处理，主要是实现同一用户的各进程之间的通信。

kill(pid,sig)：发送信号

signal(sig,func)；将信号与指定函数进行绑定。func 为 SIG_IGN 时为该进程忽略该信号。sig 的值为 SIGINT 时为中断信号如 del 或者 Ctrl + C，由用户产生。sig 的值为 SIGINFO 时为 BSD signal 使用 Ctrl + T 触发信号。

打开消息队列 int msgget(key_t key,int msgflg|IPC_CREATE);

发送消息队列 int msgsnd(int msgid, const void *msgp , size_t msgsize, int flags);

接收消息队列 int msgrcv(int msgid, const void *msgp , size_t msgsize, int flags);

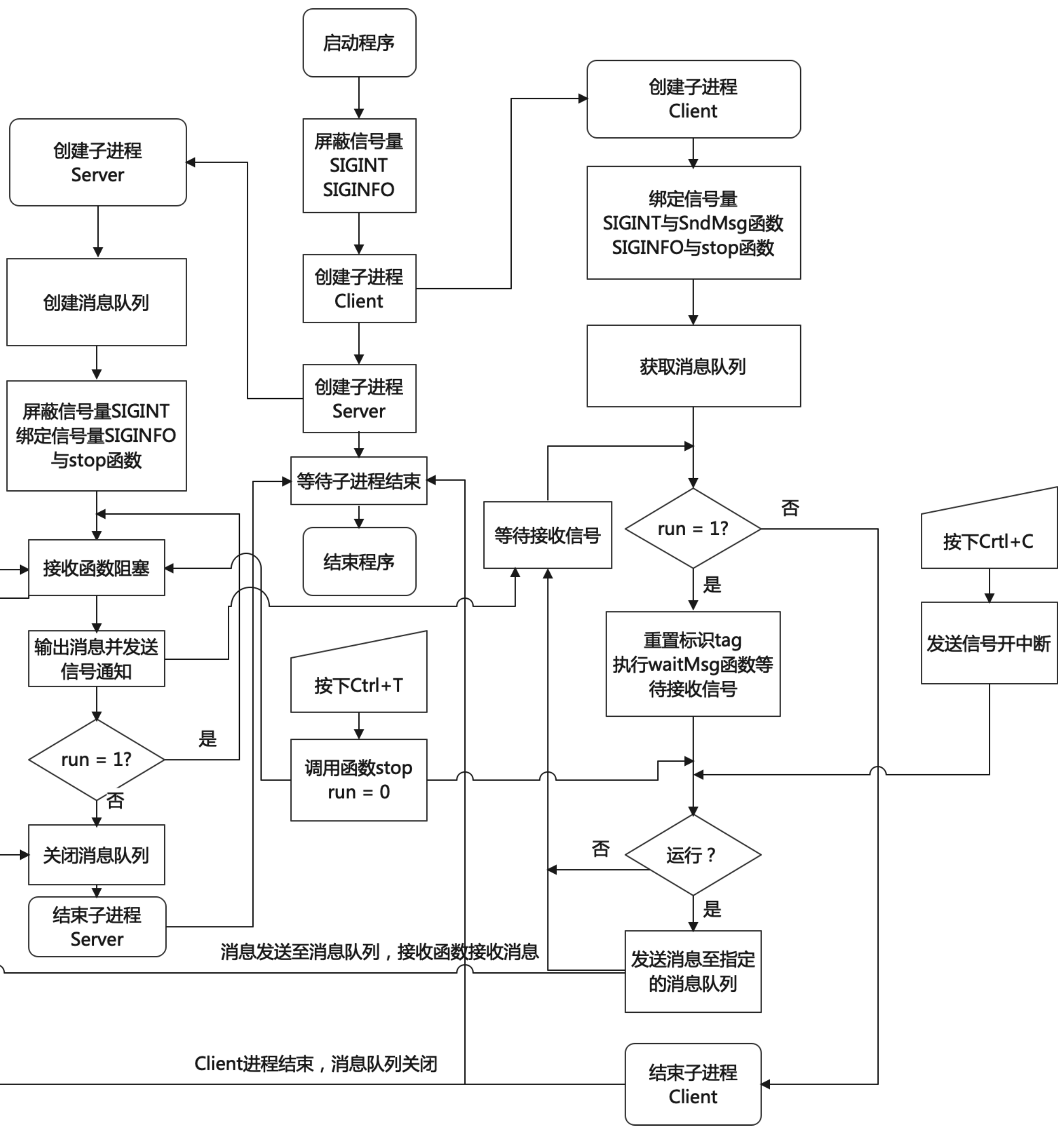
设置消息队列属性 int msgctl(int msgid ,int cmd, struct msqid_ds *data);当 cmd 为 IPC_RMID 时，清除消息队列。

## 三、 实验内容：

在原程序 3_16 基础之上，使用软中断的机制，未接收到用户发送

的信号时，发送端进程处于死循环卡住，当其接收到用户发送的信号时，跳出死循环，发送消息至消息队列，然后再次陷入死循环等待用户的下一次信号的发送。接收端在指定类型的消息队列为空的时候陷入阻塞状态，当发送端发送完成之后接收端才可以接收消息。接收完数据之后，接收端向发送端发送信号，发送端接收到信号之后用户才能继续发送消息至消息队列。

程序流程图：



## 四、 程序：

```
#include <signal.h>
#include <sys/msg.h>
#include <sys/ipc.h>
#include <sys/types.h>
#include <string.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#define MSGKEY 75

struct msgform{
	long mtype;
	char msgtext[1030];
}msg;

int msgqid,p1,p2;
int tag = 1;
int run = 1;
int ok = 1;
int alreadyGet = 1;

void waitMsg()
{
	while(alreadyGet == 0);
	printf("Waiting for you Command...\n");
	while(tag == 0)
	{
		if(run != 1)
		{
			ok = 0;
			return;
		}
	};
	alreadyGet = 0;
}
void Get()
{
	alreadyGet = 1;
}
void sndMsg()
{
	tag = 1;
```

```
}
void stop()
{
	run = 0;
}
void SERVER()
{
	printf("Server Create Success!\n");
	msgqid = msgget(MSGKEY,0777|IPC_CREAT);
	signal(SIGINT,SIG_IGN);
	signal(SIGINFO,stop);
	do
	{
		msgrcv(msgqid,&msg,1030,0,0);
		printf("(server)receive message !");
		printf("%s",msg.msgtext);
		kill(p2,SIGUSR1);
	}while(run == 1);
	msgctl(msgqid,IPC_RMID,0);
	exit(0);
}
void CLIENT()
{
	printf("Client Create Success!\n");
	int i;
	char string_i[20];
	signal(SIGINT, sndMsg);
	signal(SIGINFO, stop);
	signal(SIGUSR1,Get);
	sleep(1);
	msgqid = msgget(MSGKEY,0777);
	for(i = 1;run == 1;i++){
		tag = 0;
		waitMsg();
		if(ok == 1){
			msg.mtype = 1;
			printf("\n(client)send\n");
			sprintf(msg.msgtext,"the content of message is: ");
			sprintf(string_i,"message    %d",i);
			strcat(msg.msgtext,string_i);
			strcat(msg.msgtext,"\n");
			msgsnd(msgqid,&msg,1030,0);
		}
	}
	exit(0);
}
int main()
{
	signal(SIGINT,SIG_IGN);
	signal(SIGINFO,SIG_IGN);
	printf("The System Open Success! You can use Ctrl+C to Send a message to Server from Client, and you can use Ctrl+T to quit the system\n");
	while((p2 = fork()) == -1);
	if(!p2) CLIENT();
	while((p1 = fork()) == -1);
	if(!p1) SERVER();
	wait(0);
	wait(0);
}
```

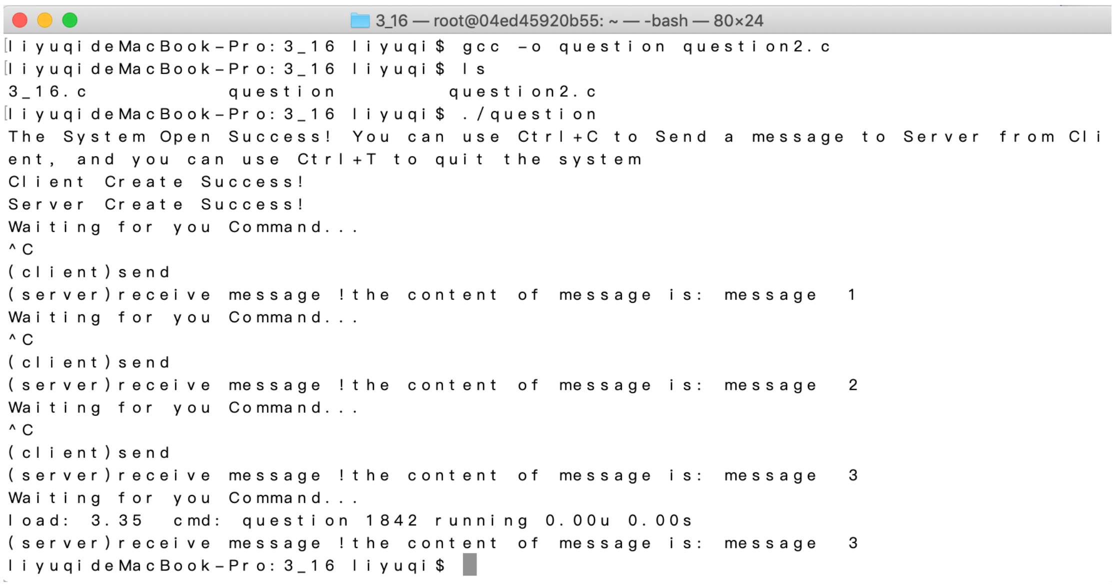
## 五、 运行结果:

使用 gcc -o question question2.c 指令将源文件进行编译生成 question 可执行文件。

使用./question 指令运行可执行文件。

按下 Ctrl + C 使得 Client 端向 Server 端发送消息。

按下 Ctrl + T  结束 Server 进程以及 Client 进程。

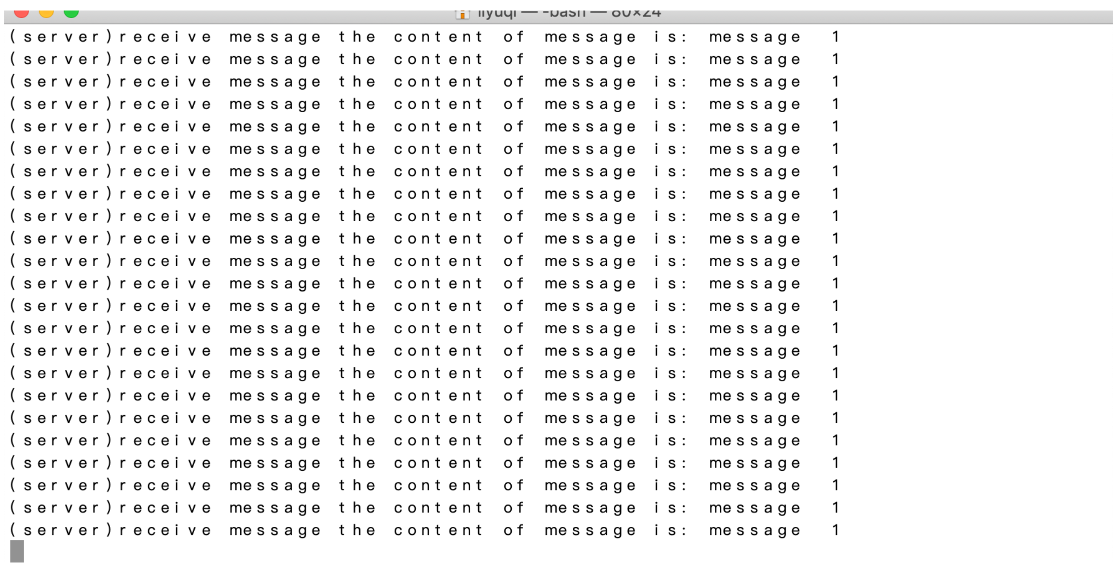


```
liyuqideMacBook-Pro:3_16 liyuqi$ gcc -o question question2.c
liyuqideMacBook-Pro:3_16 liyuqi$ ls
3_16.c          question        question2.c
liyuqideMacBook-Pro:3_16 liyuqi$ ./question
The System Open Success! You can use Ctrl+C to Send a message to Server from Cli
ent, and you can use Ctrl+T to quit the system
Client Create Success!
Server Create Success!
Waiting for you Command...
^C
(client)send
(server)receive message !the content of message is: message  1
Waiting for you Command...
^C
(client)send
(server)receive message !the content of message is: message  2
Waiting for you Command...
^C
(client)send
(server)receive message !the content of message is: message  3
Waiting for you Command...
load: 3.35  cmd: question 1842 running 0.00u 0.00s
(server)receive message !the content of message is: message  3
liyuqideMacBook-Pro:3_16 liyuqi$ 
```

## 六、 实验总结

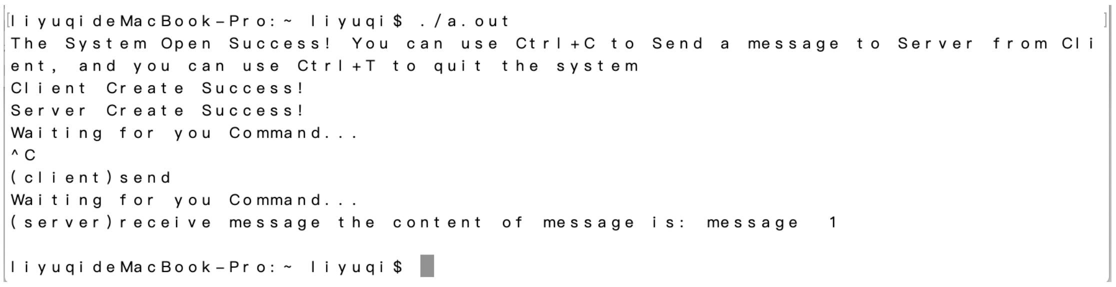
问题 1:发送端发送完数据之后接收端一直输出同一个消息。

解决办法：修改发送端的 string_i 的大小即可解决。



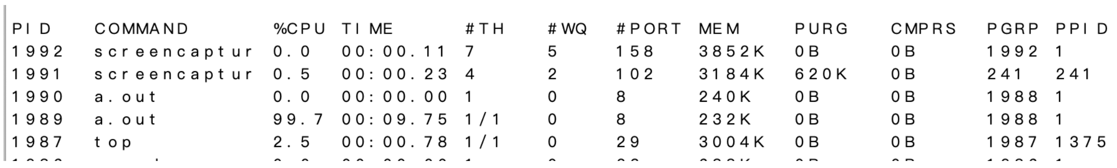
```
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
(server)receive message the content of message is: message  1
```

问题 2: 按下 Ctrl + C 之后程序结束并产生两个孤儿进程。



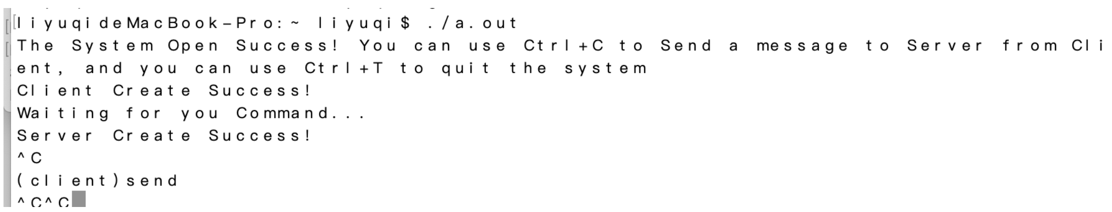
```
liyuqideMacBook-Pro:~ liyuqi$ ./a.out
The System Open Success! You can use Ctrl+C to Send a message to Server from Cli
ent, and you can use Ctrl+T to quit the system
Client Create Success!
Server Create Success!
Waiting for you Command...
^C
(client)send
Waiting for you Command...
(server)receive message the content of message is: message  1

liyuqideMacBook-Pro:~ liyuqi$
```



```
PID    COMMAND       %CPU  TIME      #TH   #WQ  #PORT  MEM    PURG   CMPRS  PGRP PPID
1992   screencaptur  0.0   00:00.11  7     5    158    3852K  0B     0B     1992 1
1991   screencaptur  0.5   00:00.23  4     2    102    3184K  620K   0B     241  241
1990   a.out         0.0   00:00.00  1     0    8      240K   0B     0B     1988 1
1989   a.out         99.7  00:09.75  1/1   0    8      232K   0B     0B     1988 1
1987   top           2.5   00:00.78  1/1   0    29     3004K  0B     0B     1987 1375
```

解决方法：父进程未屏蔽 SIGINT 以及 SIGINFO 两个信号量，导致接收到该信号的时候，父进程结束进程，而子进程屏蔽该信号，进程未中断。导致产生孤儿进程。应当在父进程里加上 signal(SIGINT,SIG_IGN)以及 signal(SIGINFO,SIG_IGN)屏蔽这两个信号。

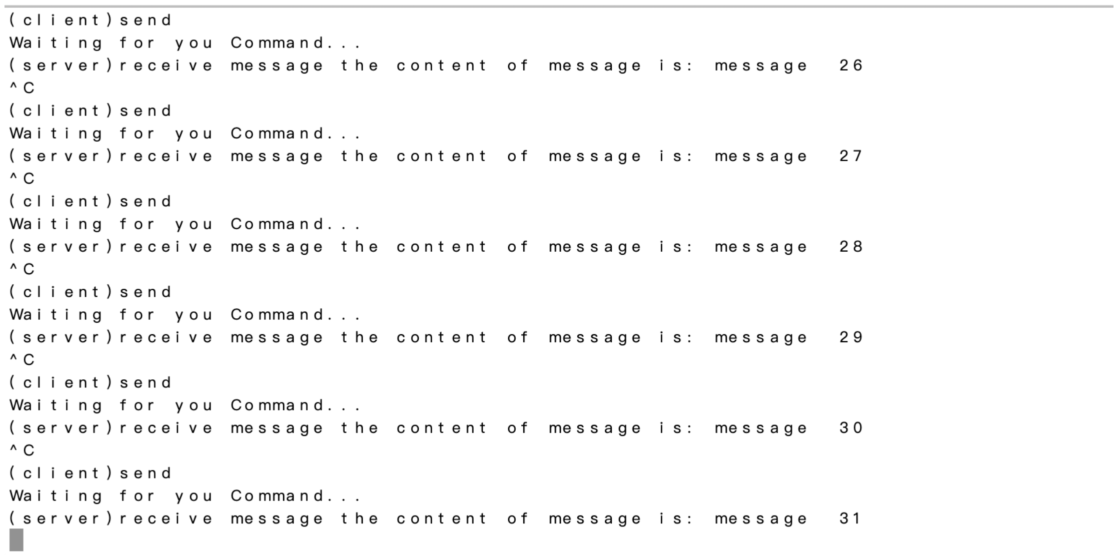


```
liyuqideMacBook-Pro:~ liyuqi$ ./a.out
The System Open Success! You can use Ctrl+C to Send a message to Server from Cli
ent, and you can use Ctrl+T to quit the system
Client Create Success!
Waiting for you Command...
Server Create Success!
^C
(client)send
^C^C
```

问题 3:发送信号之后，接收端未收到信号程序陷入阻塞状态。由于

Server 端未创建消息队列，而 Client 端获取消息队列错误，获取空队列。导致两个进程的消息队列 ID 不同，Server 端接收不到 Client 端发送的消息。

解决方法：在 Client 端获取消息队列之前使用 sleep 函数，等待 Server 端完成消息队列的创建，即可解决该问题



```
(client)send
Waiting for you Command...
(server)receive message the content of message is: message  26
^C
(client)send
Waiting for you Command...
(server)receive message the content of message is: message  27
^C
(client)send
Waiting for you Command...
(server)receive message the content of message is: message  28
^C
(client)send
Waiting for you Command...
(server)receive message the content of message is: message  29
^C
(client)send
Waiting for you Command...
(server)receive message the content of message is: message  30
^C
(client)send
Waiting for you Command...
(server)receive message the content of message is: message  31
```

问题 4:输出的顺序不符合常理， 应当等待接受进程接收完消息并输出之后，发送端再提醒用户发送消息。

解决方法：使用软中断，当发送端发送数据之前陷入阻塞状态，当接收端接收完数据并输出之后向发送端发送信号，使其跳出阻塞状态，从而才能输出提示并继续接收用户的输入。