# 实验内容

## 实验任务：

使用linux系统提供的系统调用msgget(),msgrev(),msgctl()编制一个长度为1K的消息发送和接受的程序。

## 实验要求：

* 用一个程序作为“引子”，先后fork()两个子进程，父进程执行服务端SERVER程序，两个子进程分别执行客户端CLIENT1程序和客户端CLIENT2程序，进行通信。
* 由SERVER端创建一个Key为75的消息队列，等待CLIENT1端进程和CLIENT2端进程发来的消息。当收到CLIENT1端消息编号为1的消息，以及收到CLIENT2端消息编号为1的消息后，作为结束信号，删除该队列，并结束SERVER。其中，SERVER每接受到一个消息后显示一句“(SERVER SERVER端pid) received message 消息的编号 from CLIENT CLIENT端pid”,然后发送一个返回消息给对应CLIENT端，显示一句“(SERVER SERVER端pid) sent return message 消息的编号 of CLIENT CLIENT端pid”。
* CLIENT1端使用key为75的消息队列，先后发送消息编号从10到1的消息，然后退出。CLIENT1每发送一条消息后显示一句“(CLIENT1 CLIENT1端pid) sent message 消息的编号”,然后等待接受SERVER端返回给自己的消息后，显示一句“(CLIENT1 CLIENT1端pid) received return message 消息的编号 from SERVER端pid”，再发送下一条消息。
* CLIENT2端使用key为75的消息队列，先后发送消息编号从10到1的消息，然后退出。CLIENT2每发送一条消息后显示一句“(CLIENT2 CLIENT2端pid) sent message 消息的编号”,然后等待接受SERVER端返回给自己的消息后，显示一句“(CLIENT2 CLIENT2端pid) received return message 消息的编号 from SERVER端pid”，再发送下一条消息。

# 程序代码

//xiaorong@bnu.edu.cn  
  
#include <stdio.h>  
#include <sys/types.h>  
#include <sys/msg.h>   
#include <unistd.h>  
#include <stdlib.h>  
#include <string.h>  
  
  
#define MSGKEY 75  
  
#define StoC1 1  
#define StoC2 2  
  
#define CtoS 3  
  
  
int CLIENT(int arg);  
int SERVER();  
  
pid\_t c1\_pid,c2\_pid;  
  
/\*  
请设计消息结构struct msgform，每个消息至少包含消息类型、消息编号、进程pmtpye;  
请设计消息类型，保证：服务端可以接收所有客户端发给服务端的消息；  
每个客户端只能接收服务端发给自己的返回消息，而不会接收发给其他客户端的消息；  
服务端和客户端都不能接收自己发出去的消息。  
\*/  
struct msg\_info{  
 int msg\_id;  
 int sender\_pid;  
};  
  
struct s\_msg{  
 long mtpye;  
 struct msg\_info msgInfo;  
};  
  
int main(){  
 /\*  
 用一个程序作为“引子”，先后fork()两个子进程  
 父进程执行服务端SERVER程序，两个子进程分别执行客户端CLIENT1程序和客户端CLIENT2程序，进行通信。  
 \*/  
 c1\_pid=fork();  
 c2\_pid=fork();  
 if(c1\_pid==0 && c2\_pid!=0){  
 CLIENT(StoC1);  
 }  
 else if(c2\_pid==0 && c1\_pid!=0){  
 CLIENT(StoC2);  
 }  
 else if(c2\_pid!=0 && c1\_pid!=0){  
 //printf("server is running");  
 SERVER();  
 }  
  
 return 0;  
}  
  
/\*  
由SERVER端创建一个Key为75的消息队列  
等待CLIENT1端进程和CLIENT2端进程发来的消息  
当收到CLIENT1端消息编号为1的消息  
以及收到CLIENT2端消息编号为1的消息后  
作为结束信号，删除该队列，并结束SERVER。  
其中，SERVER每接受到一个消息后显示一句  
“(SERVER \*SERVER端pid\*) received message \*消息的编号\*  
 from CLIENT \*CLIENT端pid\*”  
 ,然后发送一个返回消息给对应CLIENT端，显示一句“(SERVER \*SERVER端pmtpye\*) sent return message \*消息的编号\* of CLIENT \*CLIENT端pmtpye\*”。  
\*/  
  
  
int SERVER(){  
 //让server端真正跑起来...  
  
 int msgqid,pid;  
 struct s\_msg rcv\_msg\_1;  
 struct s\_msg snd\_msg\_1;  
 struct s\_msg rcv\_msg\_2;  
 struct s\_msg snd\_msg\_2;  
 snd\_msg\_1.mtpye=StoC1;  
 snd\_msg\_2.mtpye=StoC2;  
  
 if ((msgqid=msgget(MSGKEY,0777|IPC\_CREAT))==-1) /\*建立消息队列失败\*/  
 {  
 printf("This message queue does not exist in server. \n");  
 perror("Server msg\_queue Error:");  
 return 1;  
 }  
 pid=getpid();  
 printf("server is called,my pid is %d\n",pid);  
 int counter=10;  
   
 while(1){  
 printf("server is looping and counter = %d\n",counter);  
 msgrcv(msgqid,&rcv\_msg\_1,sizeof(rcv\_msg\_1.msgInfo),CtoS,0);  
 msgrcv(msgqid,&rcv\_msg\_2,sizeof(rcv\_msg\_2.msgInfo),CtoS,0);  
 printf("(SERVER %d) received message %d from CLIENT %d\n",pid,rcv\_msg\_1.msgInfo.msg\_id,c1\_pid);  
 printf("(SERVER %d) received message %d from CLIENT %d\n",pid,rcv\_msg\_2.msgInfo.msg\_id,c2\_pid);  
 snd\_msg\_1.msgInfo.msg\_id=counter;  
 snd\_msg\_2.msgInfo.msg\_id=counter;  
 snd\_msg\_1.msgInfo.sender\_pid=getpid();  
 snd\_msg\_2.msgInfo.sender\_pid=getpid();  
 msgsnd(msgqid,&snd\_msg\_1,sizeof(snd\_msg\_1.msgInfo),0);  
 msgsnd(msgqid,&snd\_msg\_2,sizeof(snd\_msg\_2.msgInfo),0);  
 printf("(SERVER %d) send return message %d to CLIENT %d\n",pid,snd\_msg\_1.msgInfo.msg\_id,c1\_pid);  
 printf("(SERVER %d) send return message %d to CLIENT %d\n",pid,snd\_msg\_2.msgInfo.msg\_id,c2\_pid);  
 if(rcv\_msg\_1.msgInfo.msg\_id==1 || rcv\_msg\_2.msgInfo.msg\_id==1){  
 break;   
 }  
 counter--;  
 }  
 return 0;  
}  
  
int CLIENT(int arg){  
 /\*  
 CLIENT1端使用key为75的消息队列，先后发送消息编号从10到1的消息，然后退出。  
 CLIENT1每发送一条消息后显示一句“(CLIENT1 CLIENT1端pmtpye) sent message 消息的编号”,  
 然后等待接受SERVER端返回给自己的消息后，显示一句  
 “(CLIENT1 CLIENT1端pmtpye) received return message 消息的编号 from SERVER端pmtpye”，  
 再发送下一条消息。  
 \*/  
 printf("client is called, my pmtpye is %d, my arg is %d\n",getpid(),arg);  
 int msgqid,s\_pid;  
 struct s\_msg rcv\_msg;  
 struct s\_msg snd\_msg;  
 snd\_msg.mtpye=CtoS;  
  
 while((msgqid=msgget(MSGKEY,0777))==-1);  
  
 for(int i=10;i>0;i--){  
 // 对消息内容进行赋值  
 //sprintf(msg.msg\_str, "(CLIENT1 %d) sent message %d\n", getpmtpye(), i);->没必要这样  
 //为mtpye赋值  
 snd\_msg.msgInfo.msg\_id=i;  
 msgsnd(msgqid, &snd\_msg, sizeof(snd\_msg.msgInfo), 0); // 发送消息 msg 到 msgmtpye 消息队列  
 printf("(CLIENT %d) sent message %d\n", getpid(), i);  
 msgrcv(msgqid, &rcv\_msg, sizeof(rcv\_msg.msgInfo), arg, 0); // 接收消息  
 s\_pid=rcv\_msg.msgInfo.sender\_pid;  
 printf("(CLIENT %d) received return message %d from SERVER%d\n", getpid(), rcv\_msg.msgInfo.msg\_id, s\_pid);  
 }  
 return 0;  
}

# 运行结果

以下是我编译并运行程序得到的结果，由输出可见我们达到了要求

juanitahowe@JuanitadeMacBook-Air ipcs #optional  
juanitahowe@JuanitadeMacBook-Air ipcrm -q <exist message queue id> #optional  
juanitahowe@JuanitadeMacBook-Air lab02 % gcc msg\_queue.c  
juanitahowe@JuanitadeMacBook-Air lab02 % ./a.out   
server is called,my pid is 70015  
server is looping and counter = 10  
client is called, my pmtpye is 70017, my arg is 2  
(CLIENT 70017) sent message 10  
client is called, my pmtpye is 70016, my arg is 1  
(CLIENT 70016) sent message 10  
(SERVER 70015) received message 10 from CLIENT 70016  
(SERVER 70015) received message 10 from CLIENT 70017  
(SERVER 70015) send return message 10 to CLIENT 70016  
(CLIENT 70016) received return message 10 from SERVER70015  
(CLIENT 70016) sent message 9  
(CLIENT 70017) received return message 10 from SERVER70015  
(CLIENT 70017) sent message 9  
(SERVER 70015) send return message 10 to CLIENT 70017  
server is looping and counter = 9  
(SERVER 70015) received message 9 from CLIENT 70016  
(SERVER 70015) received message 9 from CLIENT 70017  
(SERVER 70015) send return message 9 to CLIENT 70016  
(SERVER 70015) send return message 9 to CLIENT 70017  
server is looping and counter = 8  
(CLIENT 70016) received return message 9 from SERVER70015  
(CLIENT 70017) received return message 9 from SERVER70015  
(CLIENT 70016) sent message 8  
(CLIENT 70017) sent message 8  
(SERVER 70015) received message 8 from CLIENT 70016  
(SERVER 70015) received message 8 from CLIENT 70017  
(SERVER 70015) send return message 8 to CLIENT 70016  
(SERVER 70015) send return message 8 to CLIENT 70017  
server is looping and counter = 7  
(CLIENT 70016) received return message 8 from SERVER70015  
(CLIENT 70016) sent message 7  
(CLIENT 70017) received return message 8 from SERVER70015  
(CLIENT 70017) sent message 7  
(SERVER 70015) received message 7 from CLIENT 70016  
(SERVER 70015) received message 7 from CLIENT 70017  
(SERVER 70015) send return message 7 to CLIENT 70016  
(SERVER 70015) send return message 7 to CLIENT 70017  
server is looping and counter = 6  
(CLIENT 70016) received return message 7 from SERVER70015  
(CLIENT 70016) sent message 6  
(CLIENT 70017) received return message 7 from SERVER70015  
(CLIENT 70017) sent message 6  
(SERVER 70015) received message 6 from CLIENT 70016  
(SERVER 70015) received message 6 from CLIENT 70017  
(SERVER 70015) send return message 6 to CLIENT 70016  
(SERVER 70015) send return message 6 to CLIENT 70017  
(CLIENT 70016) received return message 6 from SERVER70015  
server is looping and counter = 5  
(CLIENT 70016) sent message 5  
(CLIENT 70017) received return message 6 from SERVER70015  
(CLIENT 70017) sent message 5  
(SERVER 70015) received message 5 from CLIENT 70016  
(SERVER 70015) received message 5 from CLIENT 70017  
(SERVER 70015) send return message 5 to CLIENT 70016  
(SERVER 70015) send return message 5 to CLIENT 70017  
server is looping and counter = 4  
(CLIENT 70016) received return message 5 from SERVER70015  
(CLIENT 70017) received return message 5 from SERVER70015  
(CLIENT 70016) sent message 4  
(CLIENT 70017) sent message 4  
(SERVER 70015) received message 4 from CLIENT 70016  
(SERVER 70015) received message 4 from CLIENT 70017  
(SERVER 70015) send return message 4 to CLIENT 70016  
(SERVER 70015) send return message 4 to CLIENT 70017  
server is looping and counter = 3  
(CLIENT 70017) received return message 4 from SERVER70015  
(CLIENT 70016) received return message 4 from SERVER70015  
(CLIENT 70017) sent message 3  
(CLIENT 70016) sent message 3  
(SERVER 70015) received message 3 from CLIENT 70016  
(SERVER 70015) received message 3 from CLIENT 70017  
(SERVER 70015) send return message 3 to CLIENT 70016  
(SERVER 70015) send return message 3 to CLIENT 70017  
(CLIENT 70016) received return message 3 from SERVER70015  
server is looping and counter = 2  
(CLIENT 70016) sent message 2  
(CLIENT 70017) received return message 3 from SERVER70015  
(CLIENT 70017) sent message 2  
(SERVER 70015) received message 2 from CLIENT 70016  
(SERVER 70015) received message 2 from CLIENT 70017  
(SERVER 70015) send return message 2 to CLIENT 70016  
(SERVER 70015) send return message 2 to CLIENT 70017  
server is looping and counter = 1  
(CLIENT 70017) received return message 2 from SERVER70015  
(CLIENT 70016) received return message 2 from SERVER70015  
(CLIENT 70017) sent message 1  
(CLIENT 70016) sent message 1  
(SERVER 70015) received message 1 from CLIENT 70016  
(SERVER 70015) received message 1 from CLIENT 70017  
(SERVER 70015) send return message 1 to CLIENT 70016  
(SERVER 70015) send return message 1 to CLIENT 70017  
(CLIENT 70016) received return message 1 from SERVER70015  
(CLIENT 70017) received return message 1 from SERVER70015

# 总结和感想

## 关于系统调用

系统调用的参数比较复杂，而且若是一个参数搞错则效果与预想效果偏差极大，而且较难debug，于是一定要阅读明白之后再写程序，其中重要的细节如下：

### msg\_queue 里传递的消息格式

在msg\_queue里传递的格式有特定的要求: \* 结构体里的第一个元素是一个数据类型为long的message type \* 实际接收消息的缓冲区，建议为一个数组或者结构体较为便利，具体细节参考[这个回答](https://stackoverflow.com/questions/66843903/structure-of-msg-in-ipc-message-queues)

struct msgbuf {  
 long mtype; /\* message type, must be > 0 \*/  
 char mtext[1]; /\* message data \*/  
};

### msgget

注意事项：server端flag=0777|IPC\_CREAT，0777代表文件权限，含义是表示创建的消息队列的访问权限为所有者、所属组和其他用户都具有读、写和执行权限，IPC\_CREAT代表创建队列。而client端flag=0777，无需重复创建新的队列

### msgrcv 中msgtyp参数

* msgtyp：指定要接收的消息类型。
  + 如果为 0，则表示接收队列中的第一条消息。
  + 如果不为0，接受mtype==msgtype的消息

## 关于实验设计

在这个实验中，如何设计msgtyp也是一个重要的点，需注意我们要区分server发送给c1还是c2的信息，以防止一个进程饥饿