# 进程通信——消息队列

**实验目的**

1. 了解Linux系统的进程间通信机构（IPC）；
2. 理解Linux关于消息队列的概念；
3. 掌握Linux支持消息队列的系统调用；
4. 巩固进程同步概念。

**实验内容**

消息队列保存在内核中，是一个由消息组成的链表。

有关消息队列的系统调用：

* 创建或访问消息队列

\* int msgget(key\_t key,int msgflg);

* 操作消息队列

\* int msgsnd(int msqid,const void \*msg,size\_t nbytes,int msgflg);

\* ssize\_t msgrcv(int msqid,void \*msg,size\_t nbytes,long msgtype,int msgflg);

* 控制消息队列

int msgctl(int msqid,int cmd,struct msqid\_ds \*buf);

**实验步骤**

**（一）消息的发送与接收**

实现并发进程间消息的发送与接收。编制两个程序client.c和server.c，分别用于消息的发送与接收，如后所示。

server建立一个key为75的消息队列，等待其它进程发来的消息。当遇到类型为1的消息，则作为结束信号，取消该队列，并退出server。server每接收到一个消息后显示一句“(server)received”。

client使用 key为75的消息队列，先后发送类型从10到1的消息，然后退出。最后一个消息，即是server端需要的结束信号。client每发送一条消息后显示一句“(client)sent”。

client 和server 分别发送和接收了10条消息。观察运行结果，注意发送方发送消息和接收方接收消息的顺序。

|  |  |
| --- | --- |
| #include <sys/types.h>  #include <sys/msg.h>  #include <sys/ipc.h>  #define MSGKEY 75  struct msgform  {  long mtype;  char mtext[1000];  }msg;  int msgqid;  void client()  {  int i;  //打开75#消息队列  msgqid=msgget(MSGKEY,0777);  for(i=10;i>=1;i--)  {  msg.mtype=i;  printf("(client)sent\n");  //发送消息  msgsnd(msgqid,&msg,1024,0);  }  exit(0);  }  main( )  {  client( );  } | #include <sys/types.h>  #include <sys/msg.h>  #include <sys/ipc.h>  #define MSGKEY 75  struct msgform  {  long mtype;  char mtext[1000];  }msg;  int msgqid;  void server( )  {  //创建75#消息队列  msgqid=msgget(MSGKEY,0777|IPC\_CREAT);  do  {  //接收消息  msgrcv(msgqid,&msg,1030,0,0);  printf("(server)received\n");  }while(msg.mtype!=1);  //删除消息队列，归还资源  msgctl(msgqid,IPC\_RMID,0);  exit(0);  }  main( )  {  server( );  } |

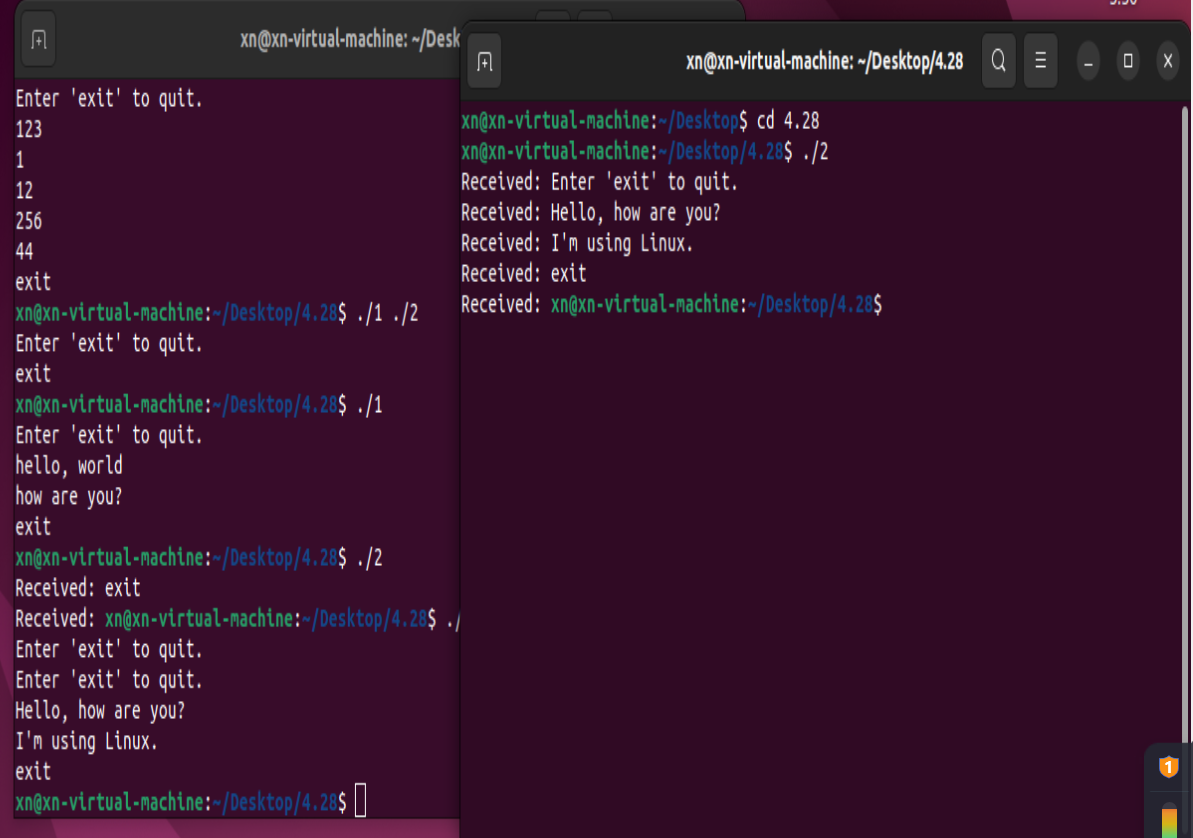
消息发送方client.c和消息接收方server.c

**（二）编写程序**

一、使用消息队列，实现具有下列功能两个程序（进程）：

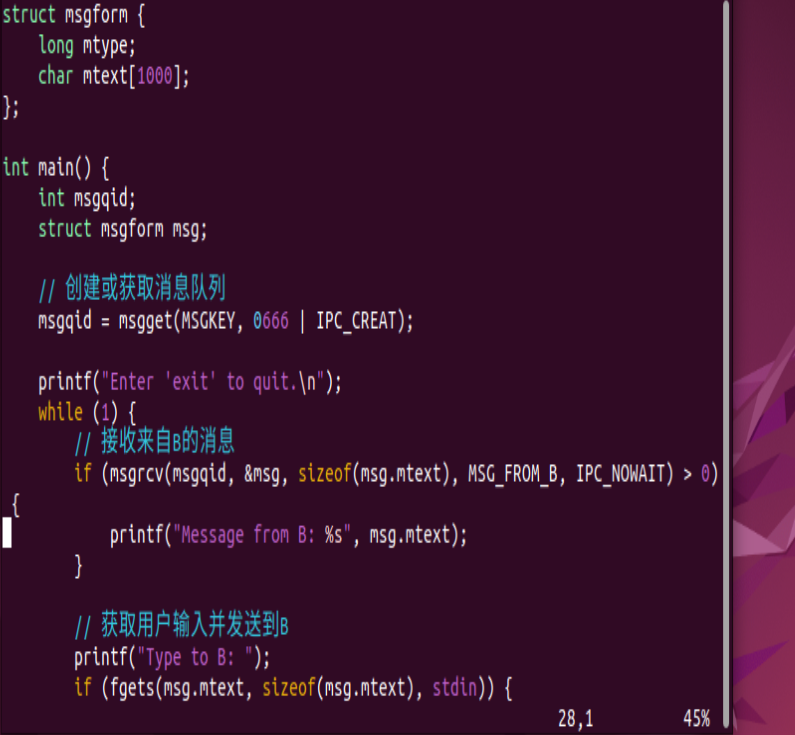
1. 程序A负责接受用户来自键盘的输入；
2. 程序B负责实时输出用户由程序A接收的字符。

提示：程序A和程序B可分别在两个进程上同时运行。

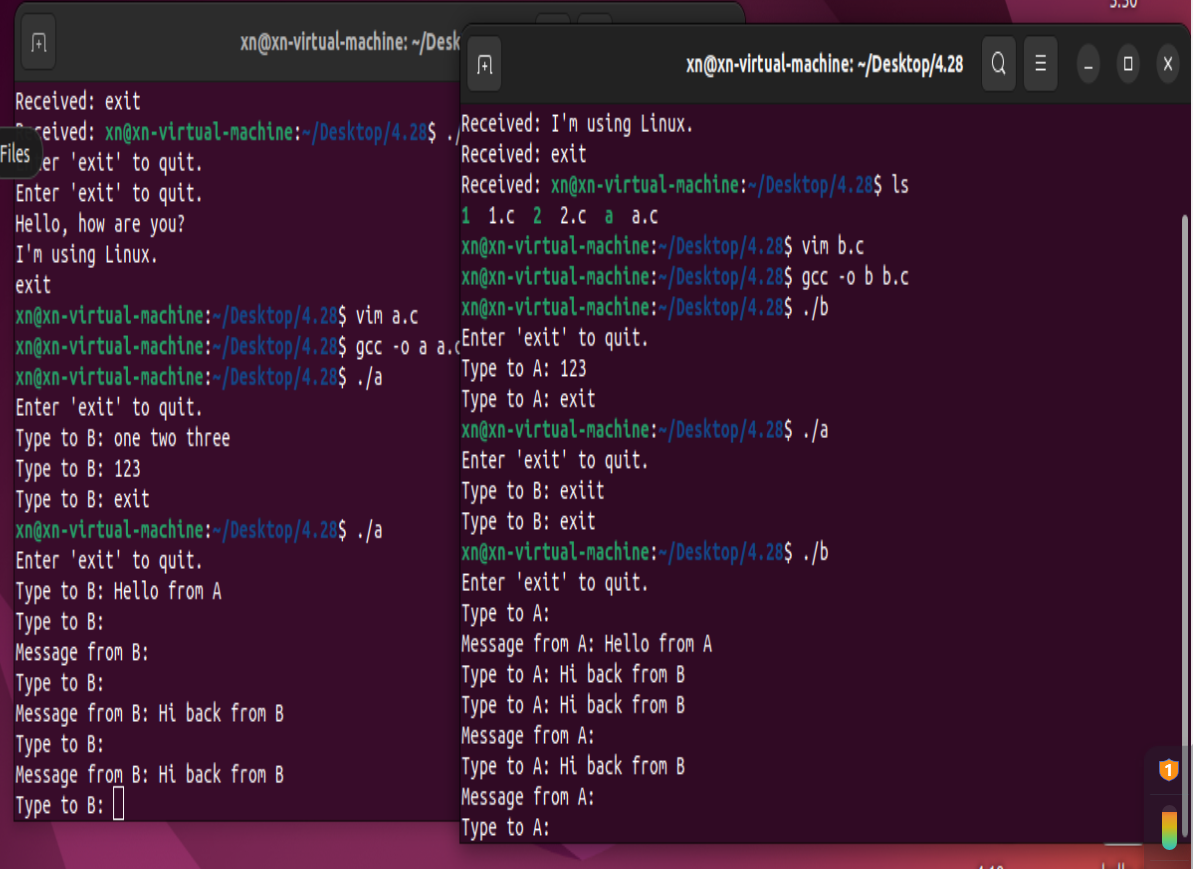


同时开两个端口进行运行

1. 在上题的基础上，使用消息队列，实现两个程序的双向通信，即：程序A接受的键盘输入，通过消息队列，在程序B的终端显示输出；同时程序B接受的键盘输入，也通过消息队列，在程序A的终端显示输出。提示：可通过建立多进程来实现。





1. 

两个端口同时进行