**进程间通信之消息队列**

1. **原理讲解**

**消息队列(也叫报文队列)，它是System V中进程间通信中的一种。消息队列就是一个消息的链表。我们可以把这个消息看成一个记录[固定的格式]。这个记录中包含了很多的信息，并且具有一定的格式和优先级。对消息队列有写权限的进程可以向中按照一定的规则添加新消息；对消息队列有读权限的进程则可以从消息队列中读走消息。，我们的操作系统提供了一个struct msg\_ids结构体来记录消息队列的全局数据结构。**

**struct msqid\_ds{**

**struct ipc\_perm msg\_perm; //被用来传递ipc操作权限信息**

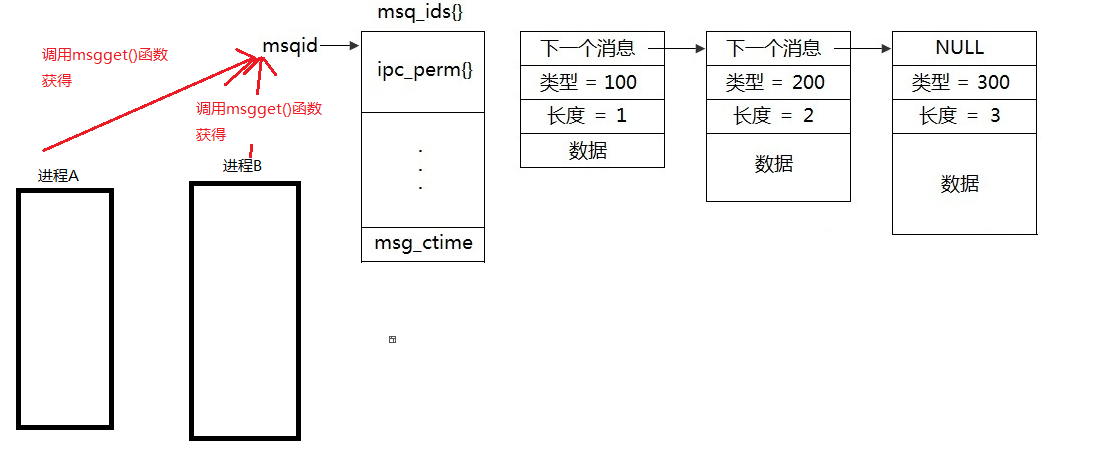
**struct msg \*msg\_first; //指向消息队列第一个结点**

**struct msg \*msg\_last; //指向详细队列最后一个结点**

**……;**

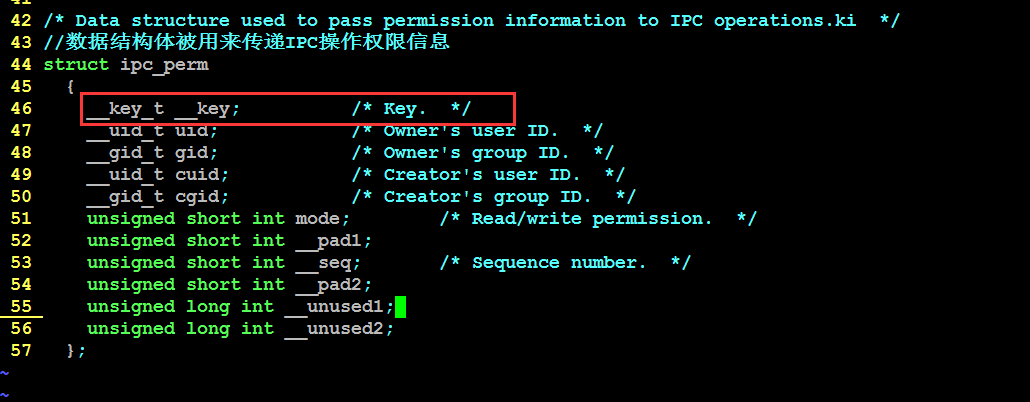
**};**

**我们可以将内核中某个特定的消息队列画成一个消息链表。如下图所示，假如一个具有三个消息的队列。长度分别为1byte，2byte和3byte，消息类型(type)为100，200，300.为那么这些消息的排序如下图。**



**OK,了解了之后，我们来回顾一下，我们以下学习的IPC通信，都是通过一个key值来标识一个ID,那么消息队列的可key值保存在哪里呢？我们来看看我们的**

**Ipc\_perm结构。**



1. **常用操作**

**<1>创建消息队列 msgget()**

**<2>发送消息队列 msgnd()，我们按照一定的类型发送即可。**

**<3>接收消息队列的信息 msgrcv（）**

**<4>删除消息队列 msgctl（）**

**1）创建消息队列**

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

**功能：创建消息队列信息。得到其ID**

**参数：**

**@key IPC\_PRIVATE 或者 ftok()**

**@msgflg**

**IPC\_CREAT | 0666 对应的消息队列不存在，则创建, 存在直接返回ID**

**或 IPC\_CREAT | IPC\_EXEL | 0666 对应的消息队列段存在则调用失败。**

**否则，创建新的消息队列段。**

**返回值：成功返回显示消息队列的ID，失败返回-1**

**2）发送消息**

**int msgsnd(int msqid, const void \*msgp, size\_t msgsz, int msgflg);**

**功能:向消息队列中添加消息**

**参数:**

**msqid 消息队列的ID**

**msgp 消息存放的地址**

**msgsz 消息正文的大小--- sizeof(msg\_t) - sizeof(long)**

**msgflg**

**0:阻塞的方式发送消息**

**IPC\_NOWAIT:非阻塞发送消息(当消息队列中没有可用空间时，不阻塞，立即返回一个错误码EAGAIN)**

**返回值：**

**成功返回0,失败返回-1**

**注意：msgp在系统内部一个指定了一个消息的模板。用于我们发送和接收消息**

**struct msgbuf{**

**long mtype; //消息的类型**

**char mtext[1]; //消息的正文。**

**};**

**但是，大家可以看到，我们的上面的模板中的消息一般不止一个字节。所以我们一般**

**自定义封装的结构体如下。**

**typedef struct{**

**long type ; //第一个字段必须是消息的类型**

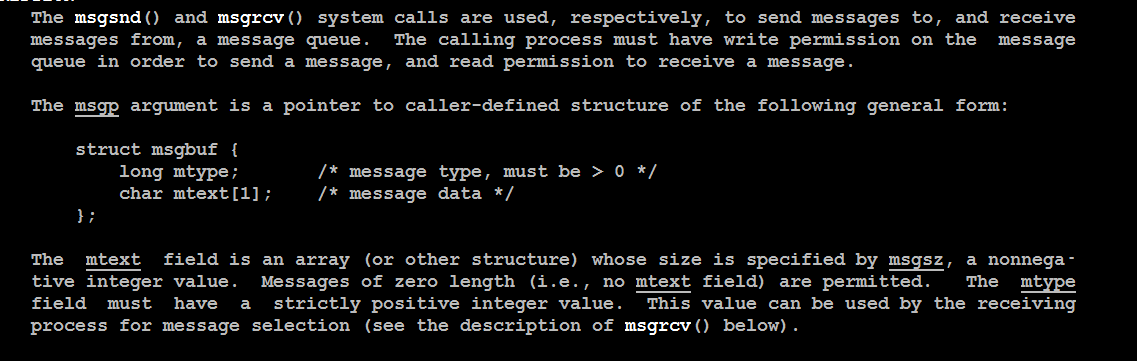
**//消息的正文 [自己设定],例如**

**char msgbuf[1024];**

**}msg\_t;**

**//消息正文的大小[除掉消息类型之后的大小]**

**#define MSG\_LEN (sizeof(msg\_t) – sizeof(long))**



**翻译如下：**

**那个msgsnd()和msgecv()函数被调用，分别用于从一个消息队列中发送消息和接收消息。那个调用**

**进程若是要发送消息，必须要有写权限，若是接收消息，必须要有读权限。**

**那个msgp 是一个访问结构体的指针，我们提供了下列的模板**

**struct msgbuf{**

**long mtype;    //消息的类型**

**char mtext[1];  //消息的正文**

**};**

**我们的mtext是一个数组[或者说一个结构]，它的大小由我们的msgsz这个参数指定，它是一个正整数值。**

**消息的长度也可以为0。那个mtype参数表示了一个消息的类型，它必须被指定为一个正整数。它可以**

**被用来接收消息选择的过程。**

**例如：**

**typedef struct**

**{**

**long type; //消息类型**

**char buf[1024] ;//正文的消息**

**}msg\_t;**

**#define MSGLEN (sizeof(msg\_t) – sizeof(long)) //正文消息长度**

**msg\_t msg;**

**msg\_t \*pmsg = (msg\_t \*)malloc(sizeof(msg\_t));**

**msg.type = 100;**

**strcpy(msg.buf,”hello”);**

**msgsnd(msgid,&msg,MSGLEN);**

**3）接收消息**

**ssize\_t msgrcv(int msqid, void \*msgp, size\_t msgsz, long msgtyp,int msgflg);**

**功能:接收指定类型的消息**

**参数:**

**msgid 消息队列ID**

**msgp 接收的消息存放的地址**

**msgsz 消息正文的大小**

**msgtyp 接收的消息的类型**

**msgflg 0(阻塞方式调用) 或 IPC\_NOWAIT (没有指定类型消息，不阻塞，立即返回)**

**返回值:**

**成功返回接收正文的大小，失败返回-1**

**例如：**

**接收消息类型为100的消息**

**msg\_t msg;**

**if(msgrcv(msgid,&msg,MSG\_LEN,100,0) < 0)**

**{**

**.....**

**}**

**printf("------------------------------\n");**

**printf("TYPE : %ld.\n",msg.type);**

**printf("MTXT : %s.\n",msg.buf);**

**printf("------------------------------\n");**

**4）删除消息队列**

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

**功能：对消息队列进行控制**

**参数：**

**@msgid 消息队列的id**

**@cmd 消息队列控制命令**

**IPC\_RMID 删除消息队列**

**@buf 填充当前进程的信息到msqid\_ds 结构体中。**

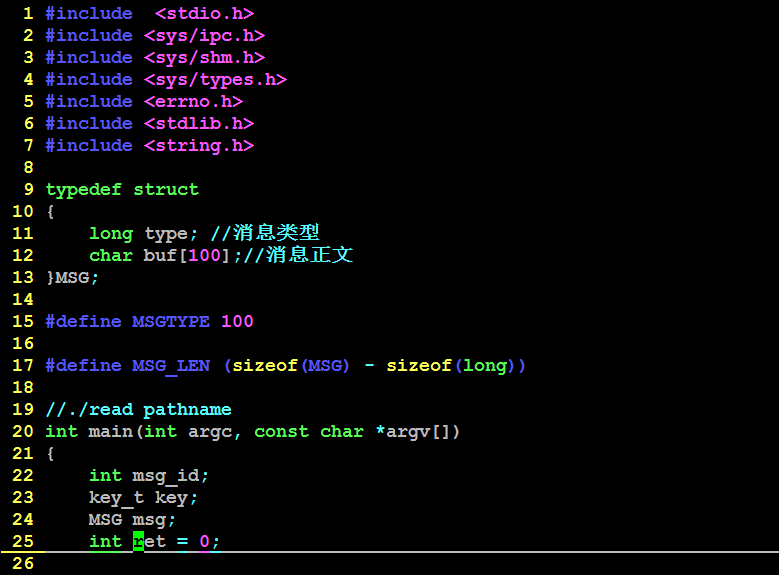
**一般我们置为NULL，不适用。**

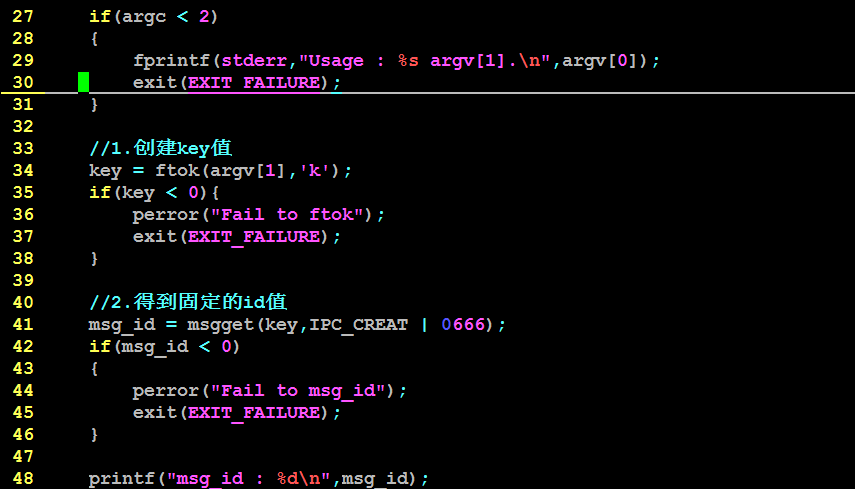
**例：msgctl(msqid,IPC\_RMID,NULL);**

**详见man手册**

**实例代码：**

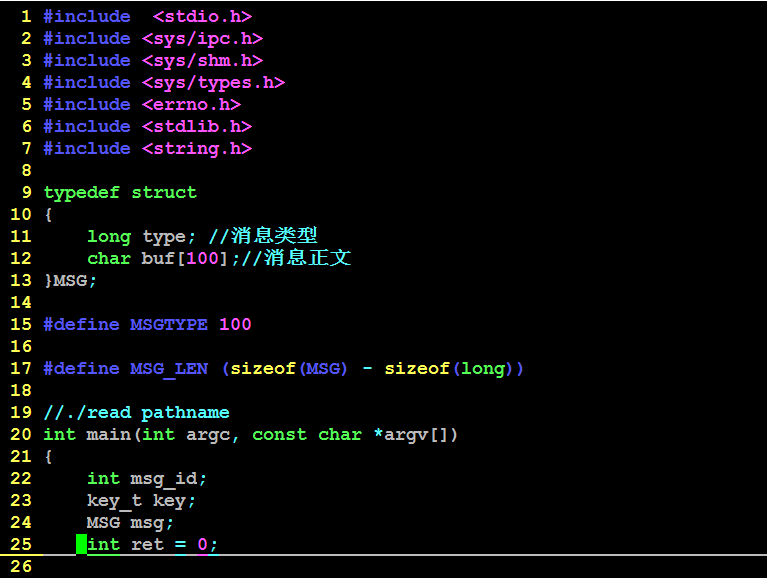
**msg\_read.c**

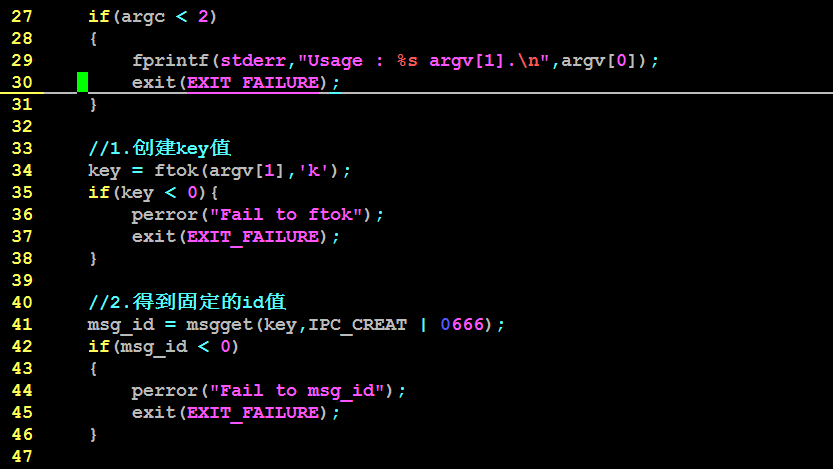


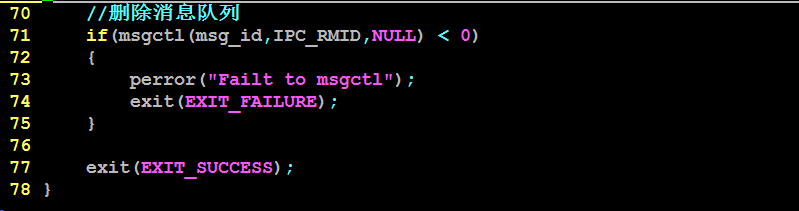




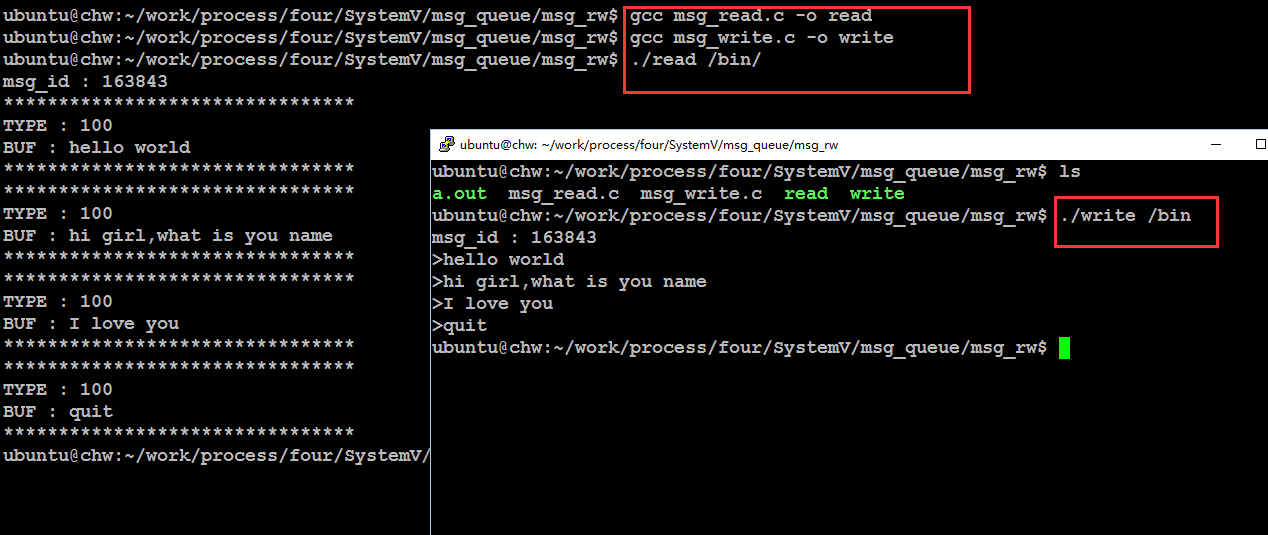
**msg\_write.c**







**运行结果：**



**练习：**

**通过消息队列让A终端上父子进程和B终端父进程了解，要求一边输入quit，所有进程**

**退出。[A和B终端都可以接收和发送消息]**

