UNIX 操作系统用消息队列实现进程通信的程序设计方法

沈华峰　（中国工商银行绍兴市分行　312000）

摘要：消息队列、共享内存、信号是 UNIX 操作系统环境中实现进程通信的主要手段。本文主要介绍消息队列的基本概念及系统实现方法，并结合实际例子，详细说明利用消息队列进行进程通信的程序设计方法。

关键词：UNIX 操作系统　消息队列　进程通信

采用 CLIENT/SERVER 结构的软件系统的处理模式是：CLIENT 向 SERVER 提出交易请求，SERVER 进行相应的数据处理后，把处理结果返回给 CLIENT。在基于 CLIENT/SERVER 结构的实际应用系统中，如储蓄临柜业务软件系统，SERVER 负责开销户、帐户查询等后台数据库的操作，CLIENT 处理前台柜员界面和帐户数据输入输出。这就需要设立一种机制，保证 CLIENT 进程和 SERVER 进程能有效地进行数据交换。UNIX 操作系统核心提供的消息机制允许进程发送格式化的数据流到其他进程，能够满足 CLIENT 方和 SERVER 方进行数据交换的需要。

一、消息队列的基本概念

将消息按队列结构进行组织管理形成消息队列。其基本设计思想是：由系统管理一组缓冲区，其中每个缓冲区可以存放一个消息；每个消息由消息类型和消息正文组成；当进程要发送消息时，先要向系统申请一缓冲区，然后把信息写进去，接着再把缓冲区送到接收进程的消息队列中；接收进程在适当的时候从消息队列中取出消息，并释放有关缓冲区。

消息队列机构是 UNIX 核心提供实现进程通信的有效手段之一，具有很多特点。首先，队列结构具有先进先出(FIFO)的特点。消息在队尾被插入，在队首被取出。这种机制能保证消息严格遵守顺序操作，可用于请求/响应模式中的事件调度处理。其次，接收进程可以有选择地接收某个队列中的消息。第三，消息正文的大小和内容可由用户定义，使发送方与接收方可定义一致的数据格式，进行数据交换。第四，信息的发送方和接收方不固定。通信双方可以使用同一个消息队列，也可以使用若干个消息队列。第五，消息队列具有操作权限控制机制，只有授权的进程才能存取消息队列。

二、消息队列的系统实现与应用程序设计

UNIX 操作系统核心提供了完善的消息管理机制，包括消息的创建、操作及控制功能。下面这些函数定义了消息队列的应用程序接口。

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

int msgget (key, msgflg)
key-t key;
int msgflg;

int msgsnd (msqid, msgp, msgsz, msgflg)
int msqid;
struct msgbuf * msgp;
int msgsz, msgflg;

int msgrcv (msqid, msgp, msgsz, msgtyp, msgflg)
int msqid;
struct msgbuf * msgp;
int msgsz;
long msgtyp;
int msgflg;

int msgctl (msqid, cmd, buf)
int msqid, cmd;
struct msqid-ds * buf;
```

1. 消息队列的创建

每个消息队列使用一个消息队列标识号(msqid)来

唯一标识该队列以便用于其他系统调用。msgget()函数用于返回(有可能是创建)与参数 key 相应的 msqid,参数 key 为一非负整数。

(1) 当相应的 msqid 已被创建, msgget()返回 msqid。

(2) 当相应的 msqid 不存在,msgget()创建 msqid 及消息队列的数据结构,返回 msqid。

key 可以被指定为 IPC-PRIVATE 以保证返回一个未用的 msqid。

消息队列的属性由 msgflg 决定。msgflg 的值必须是一个八进制数,它指定了消息队列的操作权限与控制域。

操作权限	八进制值
用户可读	00400
用户可写	00200
组用户可读	00040
组用户可写	00020
其他用户可读	00004
其他用户可写	00002

通过把对应操作权限的八进制值相加得到消息队列的操作权限。如果希望用户可读可写,组用户可读可写,那么操作权限的八进制值代码为 00660。

控制域是预先定义在 ipc.h 头文件中的常量。可取 IPC-CREAT 和 IPC-CREAT|IPC-EXCL 之一。msgflg 的值由操作权限与控制域按位"或"操作得到。

(1) 当 msgflg&IPC-CREAT 为真,且与 key 相应的 msqid 不存在时,msgget(key,(IPC-CREAT|操作权限))创建消息队列及相应的数据结构,返回 msqid。

(2) 当与 key 相应的 msqid 已存在,则 msgget(key,(IPC-CREAT|操作权限))返回 msqid。

(3) 当与 key 相应的 msqid 已存在,msgget(key,((IPC-CREAT|IPC-EXCL)|操作权限))调用失败。也就是说,可以用 IPC-EXCL 控制域来迫使系统通知错误。

消息队列创建成功后,可以用系统命令 ipcs 查询消息队列的状态,在 SCO UNIX 3.2.4.2 版本中,敲入 ipcs -q -a

```
$ ipcs -q -a
IPC status from /dev/kmem as of Tue Mar 31 06:37:44 1998
T ID KEY MODE OWNER GROUP CREATOR CGROUP CBYTES QNUM QBYTES LSPID LRPID STIME RTIME CTIME
Message Queues:
q 200 0x00000001 -Rrw-rw---- icbc informix icbc informix 0 0 16384 658 676 6:37:43 6:37:43 6:31:39
$
```

消息队列标识号 msqid = 200, key = 0x00000001,操作权限为 rw-rw---(660),队列创建者为 icbc 用户,属于 informix 组,字节数 cbytes = 0,消息数 qnum = 0,消息队列的最大空间为 16384bytes,最后一个发送消息的进程号 lspid = 658,发送时间 stime = 06:37:43,最后一个接收消息的进程号 lrpid = 676,接收时间 rtime = 06:37:43,队列改变时间 ctime = 06:31:39。

2. 消息队列的操作

消息队列的操作包括向消息队列发送消息和从消息队列接收消息。分别由系统调用 msgsnd()和 msgrcv()实现。

(1) msgsnd()

msgsnd()调用接收四个参数,调用成功时返回 0,否则返回 -1。其中:

msqid 是通过 msgget()创建的消息队列标识号。

msgp 是指向消息结构的指针,该指针所指结构包括消息类型和消息正文。

```
struct msgbuf {
long mtype; /* message type */
char mtext[]; /* message text */
};
```

msgsz 是消息结构中定义的字符数组 mtext 的长度。其最大值由系统参数 MSGMAX 决定。

msgflg 定义当消息队列空间溢出(即队列满)时系统应采取的行动。

①当 msgflg&IPC-NOWAIT 为真,msgsnd()不发送消息,立即返回。

②当 msgflg&IPC-NOWAIT 为假,msgsnd()处于"挂起"状态,等待下列事件的发生:

·溢出条件不再存在

·消息队列被删除

·捕获用户信号

(2) msgrcv()

msgrcv()系统调用接收五个参数,调用成功时返回消息队列正文长度,否则返回 -1。

msqid msgp 参数含义与 msgsnd()中的一样。

msgsz 指定被接收消息的长度,如果它的值小于 mtext 数组的长度,则:

①若 msgflg&MSG-NOERROR 为真，则消息正文按 msgsz 大小截取，系统不报告错误消息。

②若 msgflg&MSG-NOERROR 为假，则 msgsnd()返回 -1。

msgtyp 参数用于指定接收消息的类型，定义如下：

①msgtyp = 0，接收消息队列上的第一个消息。

②msgtyp > 0，接收消息队列中第一个类型为 msgtyp 的消息。

③msgtyp < 0，接收消息队列中类型小于或等于 msgtyp 绝对值最小的消息。

msgflg 用于定义当消息队列上没有指定的消息或消息队列为空时，msgrcv()应采取的行动。

①若 msgflg&IPC-NOWAIT 为真，则 msgrcv()返回 -1。

②若 msgflg&IPC-NOWAIT 为假，则 msgrcv()处于"挂起"状态，等待下列事件的发生：

·队列中指定类型的消息被放入

·消息队列被删除

·捕获用户信号

3. 消息队列的控制

系统调用 msgctl()实现对消息队列的控制，它接收三个参数。调用成功返回 0，否则返回 -1。

msqid 指定被控制的消息队列标识号。

cmd 可被下列的控制命令之一所代替：

①IPC-STAT，将指定消息队列的状态结构(msqid-ds)信息取出放入 buf 指向的缓冲区。

②IPC-SET，设置消息队列操作权限及消息的长度。

③IPC-RMID，释放指定的消息队列及相关的数据结构。

只有消息队列的创建者或超级用户才能执行 IPC-SET 或 IPC-RMID 控制命令。

buf 是指向消息队列结构的指针，该指针所指的结构定义在 msg.h 头文件以及 ipc.h 头文件中

```
struct msqid-ds {
    struct ipc-perm msg-perm; /* operation permission struct */
    struct msg *msg-first; /* ptr to first message on q */
    struct msg *msg-last; /* ptr to last message on q */
    ushort msg-cbytes; /* current # bytes on q */
    unsigned short msg-qnum; /* # of messages on q */
    unsigned short msg-qbytes; /* max # of bytes on q */
    ushort msg-lspid; /* pid of last msgsnd */
    ushort msg-lrpid; /* pid of last msgrcv */
    time-t msg-stime; /* last msgsnd time */
    time-t msg-rtime; /* last msgrcv time */
    time-t msg-ctime; /* last change time */
};
struct ipc-perm {
    uid-t uid; /* owner's user id */
    gid-t gid; /* owner's group id */
    uid-t cuid; /* creator's user id */
    gid-t cgid; /* creator's group id */
    mode-t mode; /* access modes */
    unsigned short seq; /* slot usage sequence number */
    key-t key; /* key */
};
struct msg {
    struct msg *msg-next; /* ptr to next message on q */
    long msg-type; /* message type */
    short msg-ts; /* message text size */
    short msg-spot; /* message text map address */
};
```

下面给出一个运用消息队列系统接口实现 client/server 进程通信的程序实例，该程序在 SCO UNIX 3.2.4.2 环境中调试通过。（程序清单略 编者注）

(1) 首先运行 SERVER 程序 server，创建消息队列。

(2) 运行 CLIENT 程序 client，client 进程把自己的进程号及消息发送时间作为消息正文发送给消息队列。

(3) server 进程从消息队列接收到该消息后，把消息正文中的 client 进程号设置到消息类型中，把处理的请求数和时间作为消息正文发送到消息队列。

(4) client 进程读取消息队列中消息类型为本进程号的消息，并打印结果。

(5) 在收到 100 条请求后，server 进程在删除消息队列后终止运行。

(来稿时间：1998 年 4 月)