

第六章：消息队列

消息队列是消息的链表，存放在内存中，由内核维护

6.1 消息队列概述

消息队列的特点

- 1、消息队列中的消息是有类型的。
- 2、消息队列中的消息是有格式的。
- 3、消息队列可以实现消息的随机查询。消息不一定要以先进先出的次序读取，编程时可以按消息的类型读取。
- 4、消息队列允许一个或多个进程向它写入或者读取消息。
- 5、与无名管道、命名管道一样，从消息队列中读出消息，消息队列中对应的数据都会被删除。
- 6、每个消息队列都有消息队列标识符，消息队列的标识符在整个系统中是唯一的。
- 7、只有内核重启或人工删除消息队列时，该消息队列才会被删除。若不人工删除消息队列，消息队列会一直存在于系统中。

在 ubuntu 12.04 中消息队列限制值如下：

每个消息内容最多为 8K 字节

每个消息队列容量最多为 16K 字节

系统中消息队列个数最多为 1609 个

系统中消息个数最多为 16384 个

System V 提供的 IPC 通信机制需要一个 key 值，通过 key 值就可在系统内获得一个唯一的消息队列标识符。key 值可以是人为指定的，也可以通过 ftok 函数获得。

6.1.1 ftok 函数

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

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

```
key_t ftok(const char *pathname, int proj_id);
```

功能：

获得项目相关的唯一的 IPC 键值。

参数：

pathname：路径名

proj_id：项目 ID，非 0 整数(只有低 8 位有效)

返回值：

成功返回 key 值，失败返回 -1

6.2 消息队列的操作

6.2.1 创建消息队列

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

```
int msgget(key_t key, int msgflg);
```

功能:

创建一个新的或打开一个已经存在的消息队列。不同的进程调用此函数，只要用相同的 key 值就能得到同一个消息队列的标识符。

参数:

key: IPC 键值。

msgflg: 标识函数的行为及消息队列的权限。

msgflg 的取值:

IPC_CREAT: 创建消息队列。

IPC_EXCL: 检测消息队列是否存在。

位或权限位: 消息队列位或权限位后可以设置消息队列的访问权限，格式和 open 函数的 mode_t 一样，但可执行权限未使用。

返回值:

成功: 消息队列的标识符，失败: 返回-1。

使用 shell 命令操作消息队列:

查看消息队列

```
ipcs -q
```

删除消息队列

```
ipcrm -q msqid
```

消息队列的消息的格式。

```
typedef struct _msg
```

```
{
```

```
    long mtype;    /*消息类型*/
```

```
    char mtext[100]; /*消息正文*/
```

```
    ... /*消息的正文可以有多个成员*/
```

```
}MSG;
```

消息类型必须是长整型的，而且必须是结构体类型的第一个成员，类型下面是消息正文，正文可以有多个成员（正文成员可以是任意数据类型的）。

6.2.2 发送消息

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

```
int msgsnd(int msqid, const void *msgp, size_t msgsz, int msgflg);
```

做真实的自己，用良心做教育

功能:

将新消息添加到消息队列。

参数:

msqid: 消息队列的标识符。

msgp: 待发送消息结构体的地址。

msgsz: 消息正文的字节数。

msgflg: 函数的控制属性

0: msgsnd 调用阻塞直到条件满足为止。

IPC_NOWAIT: 若消息没有立即发送则调用该函数的进程会立即返回。

返回值:

成功: 0; 失败: 返回-1。

6.2.3 接收消息

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

```
ssize_t msgrcv(int msqid, void *msgp, size_t msgsz, long msgtyp, int msgflg);
```

功能:

从标识符为 msqid 的消息队列中接收一个消息。一旦接收消息成功，则消息在消息队列中被删除。

参数:

msqid: 消息队列的标识符，代表要从哪个消息列中获取消息。

msgp: 存放消息结构体的地址。

msgsz: 消息正文的字节数。

msgtyp: 消息的类型、可以有以下几种类型

msgtyp = 0: 返回队列中的第一个消息

msgtyp > 0: 返回队列中消息类型为 msgtyp 的消息

msgtyp < 0: 返回队列中消息类型值小于或等于 msgtyp 绝对值的消息，如果这种消息有若干个，则取类型值最小的消息。

注意:

若消息队列中有多种类型的消息，msgrcv 获取消息的时候按消息类型获取，不是先进先出的。

在获取某类型消息的时候，若队列中有多条此类型的消息，则获取最先添加的消息，即先进先出原则。

msgflg: 函数的控制属性

0: msgrcv 调用阻塞直到接收消息成功为止。

MSG_NOERROR: 若返回的消息字节数比 nbytes 字节数多，则消息就会截短到 nbytes 字节，且不通知消息发送进程。

IPC_NOWAIT: 调用进程会立即返回。若没有收到消息则立即返回-1。

返回值:

成功返回读取消息的长度，失败返回-1。

6.2.4 消息队列的控制

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

做真实的自己，用良心做教育

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

功能:

对消息队列进行各种控制，如修改消息队列的属性，或删除消息消息队列。

参数:

msqid: 消息队列的标识符。

cmd: 函数功能的控制。

buf: msqid_ds 数据类型的地址，用来存放或更改消息队列的属性。

cmd: 函数功能的控制

IPC_RMID: 删除由 msqid 指示的消息队列，将它从系统中删除并破坏相关数据结构。

IPC_STAT: 将 msqid 相关的数据结构中各个元素的当前值存入到由 buf 指向的结构中。

IPC_SET: 将 msqid 相关的数据结构中的元素设置为由 buf 指向的结构中的对应值。

返回值:

成功: 返回 0; 失败: 返回 -1

例: 01_message_queue_write.c

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

typedef struct _msg
{
    long mtype;
    char mtext[50];
}MSG;

int main(int argc, char *argv[])
{
    key_t key;
    int msgqid;
    MSG msg;

    key = ftok(".", 2012);
    msgqid = msgget(key, IPC_CREAT|0666);
    if(msgqid == -1)
    {
```

```
perror("msgget");
exit(-1);
}
msg.mtype = 10;
strcpy(msg.mtext, "hello world");
msgsnd(msgqid, &msg, sizeof(msg.mtext), 0);
return 0;
}
```

例：01_message_queue_read.c

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

typedef struct _msg
{
    long mtype;
    char mtext[50];
}MSG;

int main(int argc, char *argv[])
{
    key_t key;
    int msgqid;
    MSG msg;

    key = ftok(".", 2012);
    msgqid = msgget(key, IPC_CREAT|0666);
    if(msgqid == -1)
    {
        perror("msgget");
        exit(-1);
    }
}
```

```
msgrcv(msgqid, &msg, sizeof(msg.mtext), 10, 0);  
printf("msg.mtext=%s\n", msg.mtext);  
msgctl(msgqid, IPC_RMID, NULL);  
return 0;  
}
```

千锋教育