LinuxIPC 之消息队列

1.1. 消息队列

消息队列与 FIFO 很相似，都是一个队列结构，都可以有多个进程往队列里面写信息，多个进程从队列中读取信息。但 FIFO 需要读、写的两端事先都打开，才能够开始信息传递工作。而消息队列可以事先往队列中写信息，需要时再打开读取信息。但是，消息队列先打开读，仍然会阻塞，因为此时没有消息可读。

·**System V** IPC 机制：消息队列。

函数原型：

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
int msgget(key_t key, int msgflg);
int msgsnd(int msqid, struct msgbuf *msgp, size_t msgsz, int msgflg);
ssize_t    msgrcv(int msqid, struct msgbuf *msgp, size_t msgsz, long msgtyp, int msgflg);
int msgctl(int msqid, int cmd, struct msqid_ds *buf);
```

函数 msgget 创建和访问一个消息队列。该函数成功则返回一个唯一的消息队列标识符（类似于进程 ID 一样），失败则返回-1.

参数 key 是唯一标识一个消息队列的关键字，如果为 IPC_PRIVATE(值为 0)，用创建一个只有创建者进程才可以访问的消息队列，可以用于父子间通信；非 0 值的 key(可以通过 ftok 函数获得)表示创建一个可以被多个进程共享的消息队列；

参数 msgflg 指明队列的访问权限和创建标志，创建标志的可选值为 IPC_CREAT 和 IPC_EXCL 如果单独指定 IPC_CREAT,msgget 要么返回新创建的消息队列 id,要么返回具有相同 key 值的消息队列 id；如果 IPC_EXCL 和 IPC_CREAT 同时指明，则要么创建新的消息队列，要么当队列存在时，调用失败并返回-1。

函数 msgsnd 和 msgrcy 用来将消息添加到消息队列中和从一个消息队列中获取信息。

参数 msgid 指明消息队列的 ID; 通常是 msgget 函数成功的返回值。

参数 msgbuf 是消息结构体，它的长度必须小于系统规定的上限，必须以一个长整型成员变量开始，接收函数将用这个成员变量来确定消息的类型。必须重写这个结构体，其中第一个参数不能改，其他自定义。如下：

```
struct msgbuf {
    long mtype;         /* type of message */
    char mtext[1];      /* message text */
};
```

字段 mtype 是用户自己指定的消息类型（必须是正整数），该结构体第 2 个成员仅仅是一种说明性的结构，实际上用户可以使用任何类型的数据，就是消息内容；

参数 msgsz 是消息体的大小，每个消息体最大不要超过 4K;

参数 msgflg 可以为 0（通常为 0）或 IPC_NOWAIT，如果设置 IPC_NOWAIT,则 msgsnd 和 msgrcv 都不会阻塞，此时如果队列满并调用 msgsnd 或队列空时调用 msgrcv 将返回错误；

参数 msgtyp 有 3 种选项：

msgtyp == 0　　接收队列中的第 1 个消息（通常为 0）
msgtyp > 0　　接收对列中的第 1 个类型等于 msgtyp 的消息
msgtyp < 0　　接收其类型小于或等于 msgtyp 绝对值的第 1 个最低类型消息

函数 msgctl 是消息队列的控制函数，常用来删除消息队列。

参数 msqid 是由 msgget 返回的消息队列标识符。

参数 cmd 通常为 IPC_RMID 表示删除消息队列。

参数 buf 通常为 NULL 即可。

示例：有亲缘关系的消息队列（IPC_PRIVATE）：

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <sys/stat.h>
struct MSG{
    long mtype;
    char buf[64];
};
int main()
{
    int msgid = msgget((key_t)1234, 0666 | IPC_CREAT);
    if(msgid == -1)
    {
        perror("msgget error");
        exit(-1);
    }
    struct MSG msg;
    memset(&msg,0,sizeof(struct MSG));
    if(fork() > 0)
    {
        msg.mtype = 1;
        strcpy(msg.buf,"hello");
        msgsnd(msgid,&msg,sizeof(msg.buf),0);
        wait(NULL);
        msgctl(msgid, IPC_RMID, NULL);
        exit(0);
    }
    else
    {
        sleep(2);        //让父进程有时间往消息队列里面写
        msgrcv(msgid, &msg, sizeof(msg.buf), 1, 0);
```

```
            puts(msg.buf);
            exit(0);
        }
    }
```

示例：没有亲缘关系

消息发送

```
    #include <stdio.h>
    #include <string.h>
    #include <stdlib.h>
    #include <errno.h>
    #include <sys/types.h>
    #include <sys/ipc.h>
    #include <sys/msg.h>
    #include <sys/stat.h>
    #define     BUFFER    255
    struct msgtype {                                        //重新定义该结构体
        long mtype;                     //第一个参数不变
        char buffer[BUFFER+1];
    };

    int main(int argc,char **argv)
    {
        int msgid = msgget((key_t)1234,0666 | IPC_CREAT); //获取消息队列
        if(msgid == -1)
        {
            fprintf(stderr,"Creat Message    Error:%s\a\n",strerror(errno));
            exit(1);
        }

        struct msgtype msg;
        memset(&msg,0,sizeof(struct msgtype));
        msg.mtype = 1;                                          //给结构体的成员赋值
        strncpy(msg.buffer,"hello",BUFFER);
        msgsnd(msgid,&msg,sizeof(msg.buffer),0);        //发送信号

        memset(&msg,0, sizeof(msg));                    //清空结构体

        msgrcv(msgid,&msg, sizeof(msg.buffer),2,0);     //接收信号

        fprintf(stdout,"Client receive:%s\n",msg.buffer);
        exit(0);
    }
```

消息接收

```
    #include <stdio.h>
```

```
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/stat.h>
#include <sys/msg.h>
#define     BUFFER    255

struct msgtype {                                        //重定义消息结构体
    long mtype;
    char buffer[BUFFER+1];
};

int main()
{
    int msgid = msgget((key_t)1234, 0666 | IPC_CREAT);      //获得消息队列
    if(msgid == -1)
    {
        fprintf(stderr,"Creat Message   Error:%s\a\n",strerror(errno));
        exit(1);
    }

    struct msgtype msg;
    memset(&msg,0,sizeof(struct msgtype));
    while(1)
    {
        msgrcv(msgid,&msg, sizeof(msg.buffer),1,0);     //接收消息
        fprintf(stdout,"Server Receive:%s\n",msg.buffer);

        msg.mtype = 2;
        strncpy(msg.buffer,"world",BUFFER);
        msgsnd(msgid,&msg, sizeof(msg.buffer),0);       //发送消息
    }
    exit(0);
}
```