# 概述

消息队列提供了一种在两个不相关的进程之间传递数据的简单高效的方法

**消息队列常用操作函数如下：**

#include <sys/msg.h>

#include <sys/types.h>

#include <sys/ipc.h>

key\_t ftok(const char \*pathname, int proj\_id);

int msgget(key\_t key, int msgflg);

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

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

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

# 生成消息队列标识符

key\_t标识消息队列标识符

key\_t 值可以是人为指定的，也可以通过 ftok() 函数获得。

#include <sys/types.h>

#include <sys/ipc.h>

key\_t ftok(const char \*pathname, int proj\_id);

功能：

获取键（key）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：key 值

失败：-1

# 消息队列的创建

#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。

参数：

key: ftok() 返回的 key 值

msgflg: 标识函数的行为及消息队列的权限，其取值如下：

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

返回值：

成功：消息队列的标识符

失败：-1

示例代码如下：

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <string.h>

int main(int argc, char \*argv[])

{

     key\_t key;

int msgqid;

key = ftok(".", 2012); // key 值

     // 创建消息队列

msgqid = msgget(key, IPC\_CREAT | 0666);

    return 0;

}

运行结果如下：



**消息结构**

在学习消息队列读写操作前，我们先学习消息队列的消息格式：

typedef struct \_msg

{

    long mtype;       // 消息类型

    char mtext[100];  // 消息正文成员mtext

    //…… ……           // 消息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）。

## ****向消息队列发送消息****

#include <sys/msg.h>

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

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，其取值如下：

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：-1

## 从消息队列获取消息

#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。接收成功会删除消息。

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（常用）。

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。

msgflg：函数的控制属性。其取值如下：

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

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：读取消息的长度

失败：-1

## 消息队列的控制

#include <sys/msg.h>

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。其取值如下：

IPC\_RMID：删除由 msqid 指示的消息队列

IPC\_STAT：把消息队列的属性备份到 buf 里。

IPC\_SET：消息队列原来的属性值清空，再由 buf 来替换。

buf：msqid\_ds 数据类型的地址，用来存放或更改消息队列的属性。

返回值：

成功：0

失败：-1

# 实践示例

写端示例代码如下：

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <string.h>

// 消息结果体

typedef struct \_msg

{

    long mtype;

    char mtext[50];

} MSG;

int main(int argc, char \*argv[])

{

    key\_t key;

    int msgqid;

MSG msg;

key = ftok("./", 2015); // key 值

    // 创建消息队列

    msgqid = msgget(key, IPC\_CREAT | 0666);

    if (msgqid == -1)

    {

        perror("msgget");

        exit(-1);

}

    msg.mtype = 10;                // 消息类型

strcpy(msg.mtext, "hello lh"); // 正文内容

    // 发送消息

msgsnd(msgqid, &msg, sizeof(msg) - sizeof(long), 0);

    return 0;

}

读端代码：

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <string.h>

// 消息结构体

typedef struct \_msg

{

    long mtype;

    char mtext[50];

} MSG;

int main(int argc, char \*argv[])

{

    key\_t key;

int msgqid;

key = ftok("./", 2015); // key 值

    // 创建消息队列

    msgqid = msgget(key, IPC\_CREAT | 0666);

    if (msgqid == -1)

    {

        perror("msgget");

        exit(-1);

}

    MSG msg;

memset(&msg, 0, sizeof(msg));

    // 接收消息类型为10的消息

    msgrcv(msgqid, &msg, sizeof(msg) - sizeof(long), (long)10, 0);

printf("msg.mtext=%s\n", msg.mtext);

    // 把消息队列删除

    // IPC\_RMID：删除标志位

msgctl(msgqid, IPC\_RMID, NULL);

    return 0;

}

运行结果：

