进程间通信

消息队列

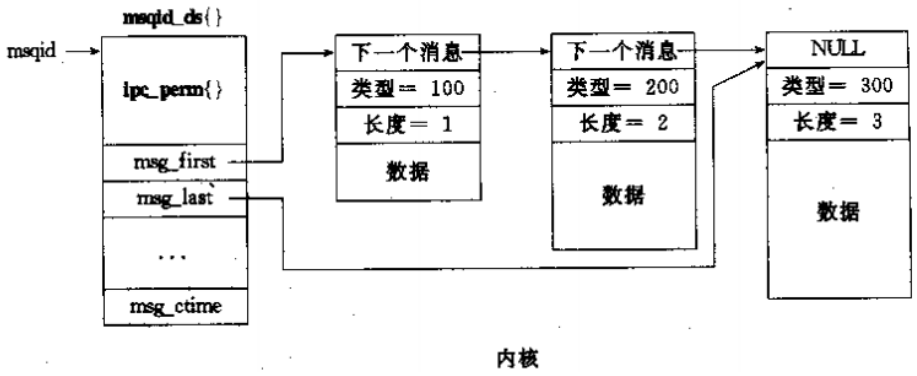
1.1 特点：

消息队列是 IPC 对象的一种

消息队列由消息队列 ID 来唯一标识

消息队列就是一个消息的列表。用户可以在消息队列中添加消息、读取消息等。

消息队列可以按照类型来发送/接收消息



1.2 步骤：

1）创建 key 值 ftok

2）创建或打开消息队列 msgget

3）添加消息/读取消息 msgsnd/msgrcv

4）删除消息队列 msgctl

1.3 操作命令

ipcs -q ：查看消息队列

ipcrm -q msgid ：删除消息队列

1.4. 函数接口

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

功能：创建或打开一个消息队列

参数： key 值

flag：创建消息队列的权限 IPC_CREAT|IPC_EXCL|0666

返回值：成功：msgid

失败：-1

```
int msgsnd(int msqid, const void *msgp, size_t size, int flag);
```

功能：添加消息

参数：msqid：消息队列的 ID

msgp：指向消息的指针。常用消息结构 msgbuf 如下：

```
struct msgbuf{
  long mtype;         //消息类型
  char mtext[N]};     //消息正文
```

size：发送的消息正文的字节数

flag：IPC_NOWAIT 消息没有发送完成函数也会立即返回

0：直到发送完成函数才返回

返回值：成功：0

失败：-1

使用：msgsnd(msgid, &msg,sizeof(msg)-sizeof(long), 0)

注意：消息结构除了第一个成员必须为 long 类型外，其他成员可以根据应用的需求自行定义。

```
int msgrcv(int msgid,  void* msgp,  size_t  size,  long msgtype,
int  flag);
```

功能：读取消息

参数：msgid：消息队列的 ID

msgp：存放读取消息的空间

size：接受的消息正文的字节数

msgtype：0：接收消息队列中第一个消息。

大于 0：接收消息队列中第一个类型为 msgtyp 的消息.

小于 0：接收消息队列中类型值不小于 msgtyp 的绝对值且类型值又最小的消息。

flag：0：若无消息函数会一直阻塞

IPC_NOWAIT：若没有消息，进程会立即返回 ENOMSG

返回值：成功：接收到的消息的长度

失败：-1

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

功能：对消息队列的操作，删除消息队列

参数：msqid：消息队列的队列 ID

例子：

```
struct msgbuf{
    long type;
    int num;
    char buf[32];
};
int main(int argc, char const *argv[])
{
    key_t key;
    int msgid;
    //创建 key 值
    key = ftok("./app", 'b');
    if (key < 0)
    {
        perror("ftok err");
        return -1;
    }
    printf("%#x\n", key);
    //创建消息队列
    msgid = msgget(key, IPC_CREAT|IPC_EXCL|0666);
    if(msgid < 0)
    {
        if(errno == EEXIST)
            msgid = msgget(key, 0666);
        else
        {
            perror("msgget err");
            return -1;
        }
    }
    //添加消息
    int size = sizeof(struct msgbuf)-sizeof(long);
    struct msgbuf msg = {1, 100, "hello"};
    struct msgbuf msg1 = {2, 200, "world"};
    msgsnd(msgid, &msg, size, 0);
    msgsnd(msgid, &msg1, size, 0);
    //读取消息
    struct msgbuf m;
    msgrcv(msgid, &m, size, 2, 0);
    printf("%d %s\n", m.num, m.buf);
    //删除消息队列
    msgctl(msgid, IPC_RMID, NULL);
    return 0;
```

练习：两个进程通过消息队列进行通信，一个进程从终端输入下发的指令，另一个进程接收指令，并打印对应操作语句。

如果输入 LED ON，另一个进程输出 “开灯”

如果输入 LED OFF，另一个进程输出 “关灯”

回顾

IO：

- 标准 IO：有缓冲机制、围绕流、stdin、stdout、stderr
  - 缓冲区：全缓存(文件)、行缓存(终端)、不缓存(sdterr)
  - 函数：打开：fopen/freopen、关闭：fclose
    - 读写：fgetc\fputc、fgets\fputs、fread\fwrite
    - 定位：fseek\rewind\ftell
    - cat、wc -l、head、cp
- 文件 IO：无缓冲机制、围绕文件描述符、0\1\2、任意类型文件除 d
  - 函数：打开：open、关闭：close 读写：read/write 定位：lseek
    - cp
- 文件属性获取：stat ls -l
- 目录操作：围绕目录流，opendir\readdir\closedir
    - ls
- 库：静态库、动态库

进程

- 进程基础：概念、三个段、分类、状态;
  - 创建进程：fork、孤儿进程、僵尸进程
  - 回收进程：wait、waitpid
  - 结束进程：exit、_exit
  - 获取进程号：getpid、getppid
  - 守护进程：特点、步骤
- 线程基础：线程和进程区别、
  - 创建线程：pthread_create、回收线程：pthread_join、结束线程：pthread_exit
  - 同步：信号量 sem_init、sem_wait、sem_post
  - 互斥：互斥锁 pthread_mutex_init、pthread_mutex_lock、pthread_mutex_unlock

条件变量：和互斥锁搭配实现同步

pthread_cond_init、pthread_cond_wait、pthread_cond_signal

进程间通信：

无名管道：亲缘关系，固定读端 fd[0]和写端 fd[1]、半双工

pipe

有名管道：不相关进程通信，文件 IO，路径中存在管道文件

mkfifo，open、read、write、close

信号：异步，处理方式：忽略、捕捉、执行缺省操作

kill、raise、pause、signal

共享内存：效率最高

步骤：shmget、shmat、shmdt、shmctl

信号灯集：实现同步

步骤：semget、semctl、semop

消息队列：按照消息类型添加消息和读取消息

步骤：msgget、msgsnd、msgrcv、msgctl