4.6 进程间通讯 - 消息队列 (一)_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 4.6 进程间通讯 – 消息队列 (一) 涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

- IPC：Inter-Process Communication (进程间通讯)
- System V IPC 对象共有三种
    - 消息队列
    - 共享内存
    - 信号量
- System V IPC 是由内核维护的若干个对象, 通过 ipcs 命名查询

------ Message Queues --------

key msqid owner perms used-bytes messages

0x00019879 1 ben 666 0 0

------ Shared Memory Segments --------

key shmid owner perms bytes nattch status

0x00000000 11 ben 600 524288 2 dest

0x00000000 16 ben 600 524288 2 dest

0x00000000 35 ben 600 4194304 2 dest

------ Semaphore Arrays --------

key semid owner perms nsems

- 每个 IPC 对象都有一个唯一的 ID, 可以通过 ftok() 函数生成，fork 函数具体说明如下:

函数头文件

#include <sys/types.h>

#include <sys/ipc.h>

函数原型

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

函数参数

- pathname：文件路径名
- proj_id：8 bit 的 id 整数

函数返回值

- 成功: 返回合成的 key
- 失败：-1, 并设置 errno
- key 由 文件的 inode 节点号 与 proj_id 构成
- inode 节点号 ： 每个存在的文件操作系统都会有唯一的编号, 通过 ls – i 命令查看

3804600 –rwxrwxr–x 1 ben ben 547 Feb 9 2021 install_for_developers.sh

3804597 –rwxrwxr–x 1 ben ben 1914 Feb 9 2021 INSTALL.py

3804598 –rw–rw–r–– 1 ben ben 1075 Feb 9 2021 LICENSE

3804601 –rwxrwxr–x 1 ben ben 2260 Feb 9 2021 newycm_extra_conf.py

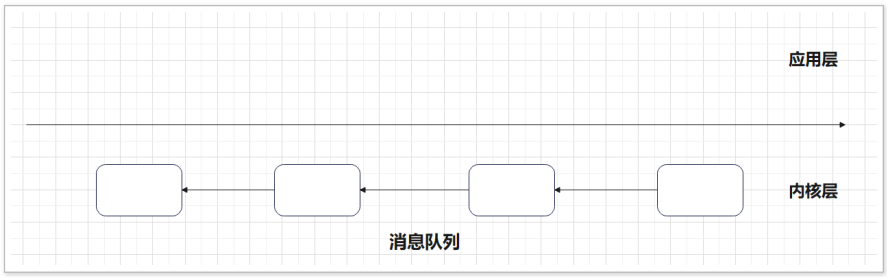
3804599 –rw–rw–r–– 1 ben ben 490 Feb 9 2021 README.md

3804606 –rw–rw–r–– 1 ben ben 1 Feb 9 2021 test.cpp

3804605 –rw–rw–r–– 1 ben ben 894 Feb 9 2021 ycm.cpp.qt.py

3804602 –rw–rw–r–– 1 ben ben 4899 Feb 9 2021 ycm.c.py

- 消息队列就是一个消息的列表, 进程可以在消息队列中添加消息和的读取消息
- 消息队列具有一定的 FIFO 特性, 具有无名管道与有名管道的各自的优势, 可以支持任意两个进程的进程间通讯



- 消息队列是属于 sytem ipc 的一种, 由内核维护与管理 可以通过 ipcs –q 查看

------ Message Queues --------

key msqid owner perms used–bytes messages

0x00019879 1 ben 666 0 0

函数头文件

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

函数原型

int msgget(key_t key, int msgflg);

函数参数

- key : 由 ftok 函数合成
- msgflg : 消息队列标志

- IPC_CREAT：创建标志
- IPC_EXCL：如果消息队列存在，则报错， errno 设置为 EEXIST
- 权限控制标志

函数返回值

- 成功：返回 消息队列 id
- 失败：返回 –1，并设置 errno

示例

创建一个消息队列, 并打印消息队列 ID

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


#define PATHNAME "."
#define PROID 10

int main(void)
{
    key_t key;
    int msgid;

    key = ftok(PATHNAME,PROID);
    if(key == -1){
        perror("fotk(): ");
        exit(EXIT_FAILURE);
    }

    msgid = msgget(key,IPC_CREAT | 0666);
    if(msgid == -1){
        perror("msgget(): ");
        exit(EXIT_FAILURE);
    }

    printf("msg id : %d\n",msgid);

    return 0;
}
```

------ Message Queues --------

key msqid owner perms used-bytes messages

0x00019879 1 ben 666 0 0

0x0a051dd0 2 ben 666 0 0

- 删除消息队列需要调用 msgctl 函数, 具体信息如下

函数头文件

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

函数原型

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

- 函数参数

- msqid : 消息队列 id
- cmd : 命令字
  - IPC_STAT： 获取 消息队列属性
  - IPC_SET : 设置消息队列属性
  - IPC_RMID : 删除消息队列属性 , 用此命名时，第三个参数为 NULL
- buf : 消息队列属性结构体对象指针

- 函数返回值

- 成功:
  - IPC_STAT, IPC_SET, and IPC_RMID 返回 0
- 失败: 返回 –1, 并设置 errno

- 消息队列属性结构体定义如下:
  - struct ipc_perm {
  - key_t __key; /* Key supplied to msgget(2) */
  - uid_t uid; /* Effective UID of owner */
  - gid_t gid; /* Effective GID of owner */
  - uid_t cuid; /* Effective UID of creator */
  - gid_t cgid; /* Effective GID of creator */
  - unsigned short mode; /* Permissions */
  - unsigned short __seq; /* Sequence number */
  - };

示例

在上一个示例的基础上，加上删除队列的代码

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


#define PATHNAME "."
#define PROID 10

int main(void)
{
        key_t key;
        int msgid;
        int ret;

        key = ftok(PATHNAME,PROID);
        if(key == -1){
                perror("fotk(): ");
                exit(EXIT_FAILURE);
        }

        msgid = msgget(key,IPC_CREAT | 0666);
        if(msgid == -1){
                perror("msgget(): ");
                exit(EXIT_FAILURE);
        }

        printf("msg id : %d\n",msgid);


        ret = msgctl(msgid,IPC_RMID,NULL);
        if(ret == -1){
                perror("msgctl(): ");
                exit(EXIT_FAILURE);
        }

        return 0;
}
```