嵌入式系统工程师

消息队列、共享内存

大纲

- 消息队列(message queue)
- 共享内存(shared memory)

大纲

- 消息队列(message queue)
- 共享内存(shared memory)

消息队列概述

- 消息队列是消息的链表，存放在内存中，由内核维护
- 消息队列的特点
  - 消息队列允许一个或多个进程向它写入或者读取消息，并且每条消息都有类型。
  - 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。
  - 与无名管道、有名管道一样，从消息队列中读出消息，消息队列中数据会被删除。
  - 同样消息队列中的消息是有格式的。

消息队列概述

- 只有内核重启或人工删除时，该消息才会被删除，若不人工删除消息队列，消息队列会一直存在于内存中
- 消息队列标识符，来标识消息队列。消息队列在整个系统中是唯一的。
- 在Linux操作系统中消息队列限制值如下:
  - 消息队列个数最多为16个
  - 消息队列总容量最多为16384字节
  - 每个消息内容最多为8192字节

消息队列

- 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ID。
- key值可以是人为指定的，也可以通过ftok函数获得。

消息队列

- #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。

消息队列

- 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ID。

参数:

- key: IPC键值
- msgflg: 标识函数的行为: IPC_CREAT(创建)或IPC_EXCL(如果已经存在则返回失败)。

返回值:

成功: 消息队列的标识符，失败: 返回-1。

消息队列

- 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

- 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）。

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: 消息队列的队列ID。

msgp: 待发送消息结构体的地址。

msgsz: 消息正文的字节数。

消息队列

- 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。

消息队列

- 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ID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ID，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。

消息队列

- 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，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。

消息队列

- 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。

消息队列

- 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ID

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。

消息队列

- 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

01_message_queue_read.c

大纲

- 消息队列(message queue)
- 共享内存(shared memory)

共享内存

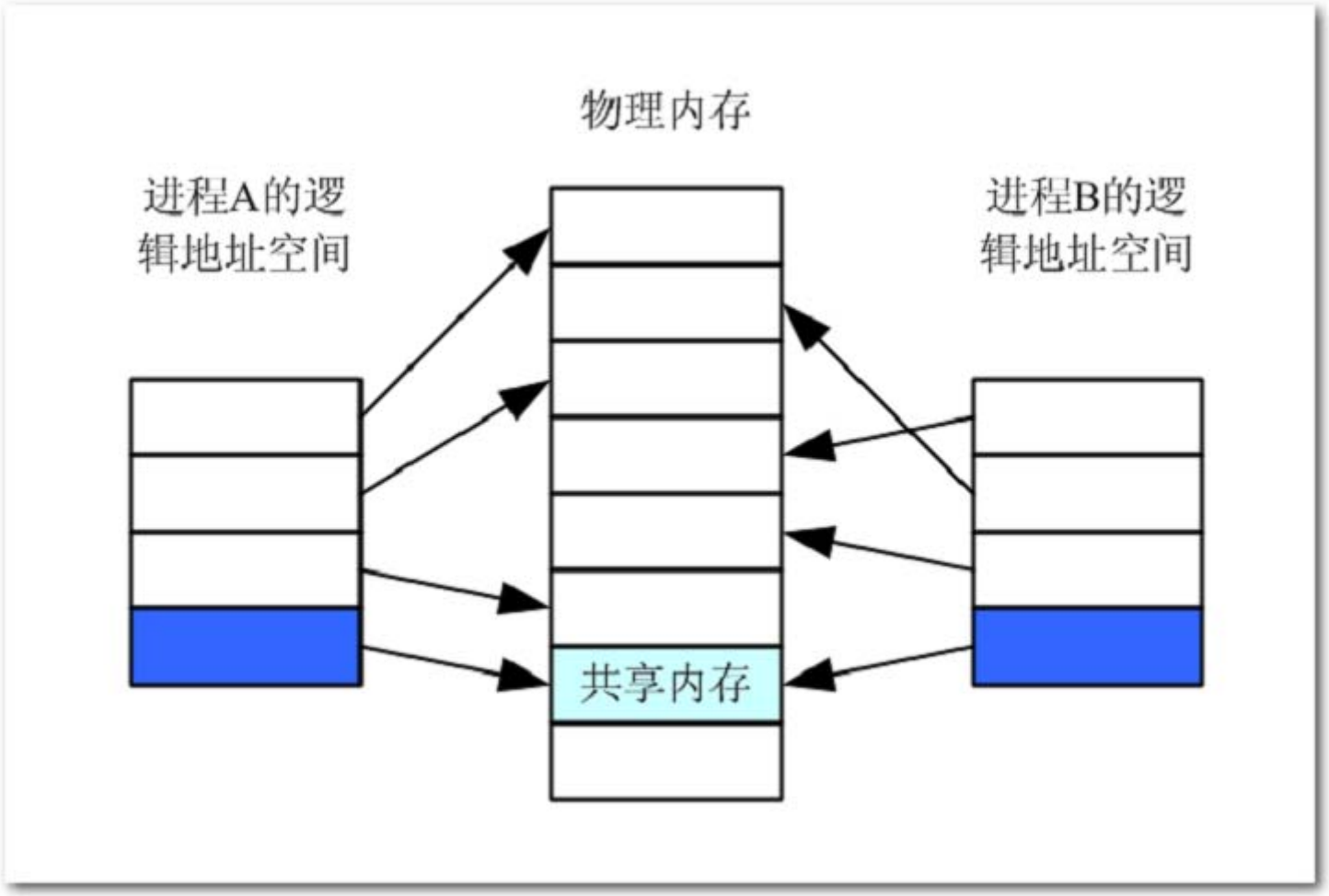
共享内存允许两个或者多个进程共享给定的存储区域。

共享内存是进程间共享数据的一种最快的方法，一个进程向共享的内存区域写入了数据，共享这个内存区域的所有进程就可以立刻看到其中的内容。

使用共享内存要注意的是多个进程之间对一个给定存储区访问的互斥。若一个进程正在向共享内存区写数据，则在它做完这一步操作前，别的进程不应当去读、写这些数据。

共享内存

- 共享内存示意图



共享内存

- 在Linux操作系统中共享内存限制值如下
  - 共享存储段的最大字节数：33554432
  - 共享存储段的最小字节数：1
  - 系统中共享存储段的最大段数：4096
  - 每个进程共享存储段的最大段数：4096

共享内存

- 获得一个共享存储标识符

```
#include <sys/ipc.h>
#include <sys/shm.h>
int shmget(key_t key, size_t size,intshmflg);
```

功能:创建或打开一块共享内存区

参数:

key: IPC键值

size: 该共享存储段的长度(字节)

shmflg: 用来标识函数的行为

共享内存

- 参数:
  - shmflg: 用来标识函数的行为

    IPC_CREAT: 如果不存在就创建

    IPC_EXCL: 如果已经存在则返回失败

    IPC_NOWAIT: 调用进程会立即返回。若发生错误则返回-1。
  - SHM_R: 可读
  - SHM_W: 可写
- 返回值:
  - 成功: 返回共享内存标识符。
  - 失败: 返回－1。

共享内存

- 使用shell命令操作共享内存:

  查看共享内存

  ```
  ipcs -m
  ```

  删除共享内存

  ```
  ipcrm -m shmid
  ```

共享内存

- 共享内存映射(attach)

```
#include <sys/types.h>
#include <sys/shm.h>
void *shmat(int shmid, const void *shmaddr,
                                    int shmflg);
```

功能:

将一个共享内存段映射到调用进程的数据段中。

参数:

shmid: 共享内存标识符。

shmaddr: 共享内存映射地址(若为NULL则由系统自动指定)，推荐使用NULL。

共享内存

- 参数:

  shmflg: 共享内存段的访问权限和映射条件

  - 0: 共享内存具有可读可写权限。
  - SHM_RDONLY: 只读。
  - SHM_RND: （shmaddr非空时才有效）

    没有指定SHM_RND则此段连接到shmaddr所指定的地址上(shmaddr必需页对齐)。

    指定了SHM_RND则此段连接到shmaddr-shmaddr%SHMLAB 所表示的地址上。

- 返回值:
  - 成功: 返回共享内存段首地址
  - 失败: 返回 -1

共享内存

- 注：

  shmat函数使用的时候第二个和第三个参数一般设为NULL和0，即系统自动指定共享内存地址，并且共享内存可读可写。

共享内存

- 解除共享内存映射(detach)

```
#include <sys/types.h>
#include <sys/shm.h>
int shmdt(const void *shmaddr);
```

功能:

将共享内存和当前进程分离(仅仅是断开联系并不删除共享内存)。

参数:

- shmaddr: 共享内存映射地址。

返回值:

- 成功返回 0，失败返回 -1。

共享内存

- 共享内存控制

```
#include <sys/ipc.h>
#include <sys/shm.h>
int shmctl(int shmid, int cmd,
                        struct shmid_ds *buf);
```

功能：共享内存空间的控制。

参数：

- shmid：共享内存标识符。
- cmd：函数功能的控制。
- buf：shmid_ds数据类型的地址，用来存放或更改消息队列的属性。

共享内存

参数:

- cmd: 函数功能的控制
  - IPC_RMID: 删除。
  - IPC_SET: 设置shmid_ds参数。
  - IPC_STAT: 保存shmid_ds参数。
  - SHM_LOCK: 锁定共享内存段(超级用户)。
  - SHM_UNLOCK: 解锁共享内存段。
- 返回值:

  成功返回 0，失败返回 -1。

例: 02_shared_memory_write.c

02_shared_memory_read.c

共享内存

注意：

SHM_LOCK用于锁定内存，禁止内存交换。并不代表共享内存被锁定后禁止其它进程访问。其真正的意义是：被锁定的内存不允许被交换到虚拟内存中。

这样做的优势在于让共享内存一直处于内存中，从而提高程序性能。