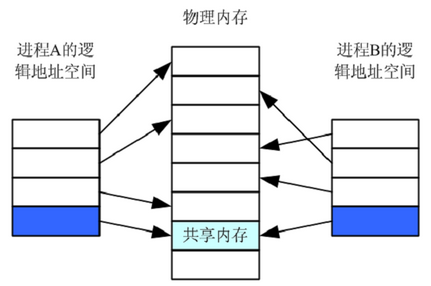
# 概述

[共享内存](http://baike.baidu.com/link?url=MdyPihmS_tWLwgWL5CMzaTrwDFHu6euAJJUAjKvlzbJmRw7RfhmkBWwAloo7Y65hLY-kQdHsbqWYP2wc2fk8yq)是进程间通信中最简单的方式之一。



# 常用函数

**1）创建共享内存**

#include <sys/ipc.h>

#include <sys/shm.h>

int shmget(key\_t key, size\_t size,int shmflg);

功能：

创建或打开一块共享内存区。

参数：

**key：**进程间通信键值，ftok() 的返回值。

**size**：该共享存储段的长度(字节)。

**shmflg**：标识函数的行为及共享内存的权限，其取值如下：

**IPC\_CREAT**：如果不存在就创建  
 **IPC\_EXCL**：  如果已经存在则返回失败  
 **位或权限位**：共享内存的访问权限，格式和 open() 函数的 mode\_t 一样

返回值：

成功：共享内存标识符。

失败：-1。

示例代码如下：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#define BUFSZ 1024

int main(int argc, char \*argv[])

{

    int shmid;

key\_t key;

    key = ftok("./", 2015);

    if (key == -1)

    {

        perror("ftok");

}

    //创建共享内存

    shmid = shmget(key, BUFSZ, IPC\_CREAT | 0666);

    if (shmid < 0)

    {

        perror("shmget");

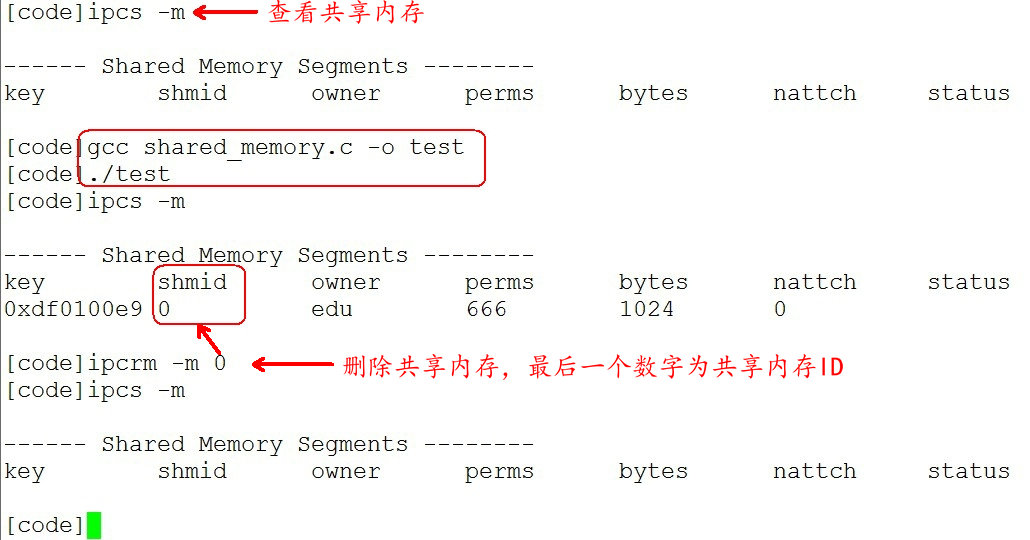
        exit(-1);

}

    return 0;

}

运行结果如下：



**2）共享内存映射**

#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，shmget() 的返回值。

**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**。

**shmflg**：共享内存段的访问权限和映射条件( **通常为 0** )，具体取值如下：

**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）

返回值：

成功：共享内存段映射地址( 相当于这个指针就指向此共享内存 )

失败：-1

**3）解除共享内存映射**

#include <sys/types.h>

#include <sys/shm.h>

void \*shmat(int shmid, const void \*shmaddr, int shmflg);

功能：

将共享内存和当前进程数据段地址分离

参数：

**shmaddr**：共享内存映射地址。

返回值：

成功：0

失败：-1

**4）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：

共享内存属性的控制。

参数：

**shmid**：共享内存标识符。

**cmd**：函数功能的控制，其取值如下：

IPC\_RMID：删除。(**常用**)

IPC\_SET：共享内存原来的属性值替换为 buf 里的属性值。

IPC\_STAT：共享内存原来的属性值备份到 buf 里。

SHM\_LOCK：锁定共享内存段( 超级用户 )，禁止内存交换。

SHM\_UNLOCK：解锁共享内存段。

**buf**：shmid\_ds 数据类型的地址，用来存放或修改共享内存的属性。

返回值：

成功：0

失败：-1

# 实战示例

创建两个进程，在 A 进程中创建一个共享内存，并向其写入数据，通过 B 进程从共享内存中读取数据。

写端代码如下：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#define BUFSZ 512

int main(int argc, char \*argv[])

{

    int shmid;

    int ret;

    key\_t key;

char \*shmadd;

    //创建key值

    key = ftok("../", 2015);

    if (key == -1)

    {

        perror("ftok");

}

    //创建共享内存

    shmid = shmget(key, BUFSZ, IPC\_CREAT | 0666);

    if (shmid < 0)

    {

        perror("shmget");

        exit(-1);

}

    //映射

    shmadd = shmat(shmid, NULL, 0);

    if (shmadd < 0)

    {

        perror("shmat");

        \_exit(-1);

}

    //拷贝数据至共享内存区

    printf("copy data to shared-memory\n");

    // 共享内存清空

    bzero(shmadd, BUFSZ);

    // 向共享内存写入 字符串

strcpy(shmadd, "how are you, lh\n");

    return 0;

}

读代码：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#define BUFSZ 512

int main(int argc, char \*argv[])

{

    int shmid;

    int ret;

    key\_t key;

char \*shmadd;

    //创建key值

    key = ftok("../", 2015);

    if (key == -1)

    {

        perror("ftok");

}

system("ipcs -m"); //查看共享内存

    //打开共享内存

    shmid = shmget(key, BUFSZ, IPC\_CREAT | 0666);

    if (shmid < 0)

    {

        perror("shmget");

        exit(-1);

}

    //映射共享内存

    shmadd = shmat(shmid, NULL, 0);

    if (shmadd < 0)

    {

        perror("shmat");

        exit(-1);

}

    //读共享内存区数据

printf("data = [%s]\n", shmadd);

    //分离共享内存

    ret = shmdt(shmadd);

    if (ret < 0)

    {

        perror("shmdt");

        exit(1);

    }

    else

    {

        printf("deleted shared-memory\n");

}

    //删除共享内存

shmctl(shmid, IPC\_RMID, NULL);

system("ipcs -m"); //查看共享内存

    return 0;

}

运行结果：

