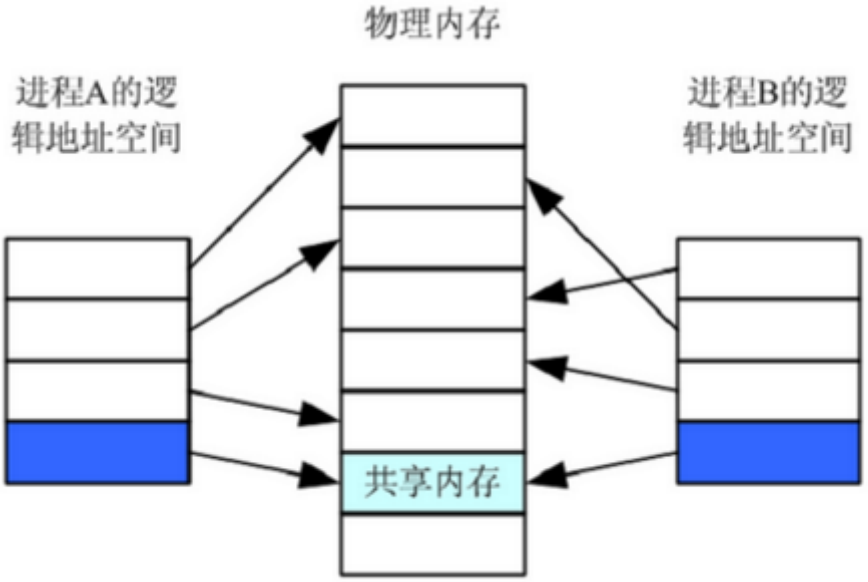
共享内存是进程间通信中最简单的方式之一。共享内存允许两个或更多的进程访问同一块内存，就如同malloc()函数向不同进程返回了指向同一个物理内存区域的指针。当一个进程改变了这块地址中的内容的时候，其他进程都会察觉这个更改。



共享内存的特点：

1.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。

2.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。

实现过程：

a.创建共享内存。

1.通过ftok()，获取key。 系统建立进程间通讯时必须指定一个ID值。通常情况下，该id值通过ftok函数得到。

2.使用key来创建一个共享内存shmget()

1、shmget()函数

该函数用来创建共享内存，它的原型为：

```
int shmget(key_t key, size_t size, int shmflg);
```

第一个参数，与信号量的semget函数一样，程序需要提供一个参数key（非0整数），它有效地为共享内存段命名，shmget()函数成功时返回一个与key相关的共享内存标识符（非负整数），用于后续的共享内存函数。调用失败返回-1.

不相关的进程可以通过该函数的返回值访问同一共享内存，它代表程序可能要使用的某个资源，程序对所有共享内存的访问都是间接的，程序先通过调用shmget()函数并提供一个键，再由系统生成一个相应的共享内存标识符（shmget()函数的返回值），只有shmget()函数才直接使用信号量键，所有其他的信号量函数使用由semget函数返回的信号量标识符。

第二个参数，size以字节为单位指定需要共享的内存容量

第三个参数，shmflg是权限标志，它的作用与open函数的mode参数一样，如果要想在key标识的共享内存不存在时，创建它的话，可以与IPC_CREAT做或操作。共享内存的权限标志与文件的读写权限一样，举例来说，0644,它表示允许一个进程创建的共享内存被内存创建者所拥有的进程向共享内存读取和写入数据，同时其他用户创建的进程只能读取共享内存。

3. 映射共享内存(得到虚拟地址)，shmat()

2、shmat()函数　-- at：attach

第一次创建完共享内存时，它还不能被任何进程访问，shmat()函数的作用就是用来启动对该共享内存的访问，并把共享内存连接到当前进程的地址空间。它的原型如下：

```
void *shmat(int shm_id, const void *shm_addr, int shmflg);
```

第一个参数，shm_id是由shmget()函数返回的共享内存标识。

第二个参数，shm_addr指定共享内存连接到当前进程中的地址位置，通常为空，表示让系统来选择共享内存的地址。

第三个参数，shm_flg是一组标志位，通常为0。

调用成功时返回一个指向共享内存第一个字节的指针，如果调用失败返回-1.

4. 使用共享内存，往共享内存中写入数据

5. 解除映射shmat()

6. 如果共享内存不再使用，可以使用shmctl()销毁共享内存。

```
{
    // 生成一个key
    key_t key = ftok("./", 66);

    // 创建共享内存，返回一个id
    int shmid = shmget(key, 8, IPC_CREAT|0666|IPC_EXCL);
    if(-1 == shmid)
    {
        perror("shmget failed");
        exit(1);
    }

    // 映射共享内存，得到虚拟地址
    void *p = shmat(shmid, 0, 0);
    if((void*)-1 == p)
    {
        perror("shmat failed");
        exit(2);
    }

    // 写共享内存
    int *pp = p;
    *pp = 0x12345678;
    *(pp + 1) = 0xffffffff;

    // 解除映射
    if(-1 == shmdt(p))
    {
        perror("shmdt failed");
        exit(3);
    }
    printf("解除映射成功，点击回车销毁共享内存\n");
    getchar();

    // 销毁共享内存
    if(-1 == shmctl(shmid, IPC_RMID, NULL))
    {
        perror("shmctl failed");
        exit(4);
    }

    return 0;
```

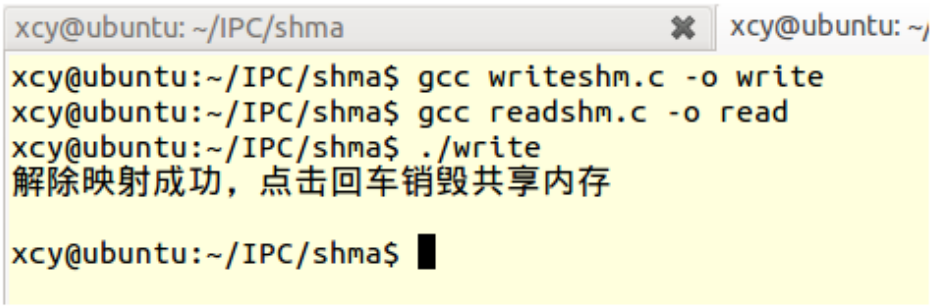
b. 读取共享内存区域

1. 获得key，ftok()
2. 使用key来获得一个共享内存shmget()
3. 映射共享内存(得到虚拟地址)，shmat()
4. 使用共享内存，读取共享内存中的数据
5. 解除映射shmdt()

```
int main()
{
    // 生成一个key
    key_t key = ftok("./", 66);

    // 获取共享内存，返回一个id
    int shmid = shmget(key, 0, 0);
    if(-1 == shmid)
    {
        perror("shmget failed");
        exit(1);
    }

    // 映射共享内存，得到虚拟地址
    void *p = shmat(shmid, 0, 0);
    if((void*)-1 == p)
    {
        perror("shmat failed");
        exit(2);
    }

    // 读共享内存
    int x = *(int *)p;
    int y = *((int *)p + 1);
    printf("从共享内存中都取了: 0x%x 和 0x%x \n", x, y);

    // 解除映射
    if(-1 == shmdt(p))
    {
        perror("shmdt failed");
        exit(3);
    }

    return 0;
}
```

运行结果：

writeshma：



readshma：