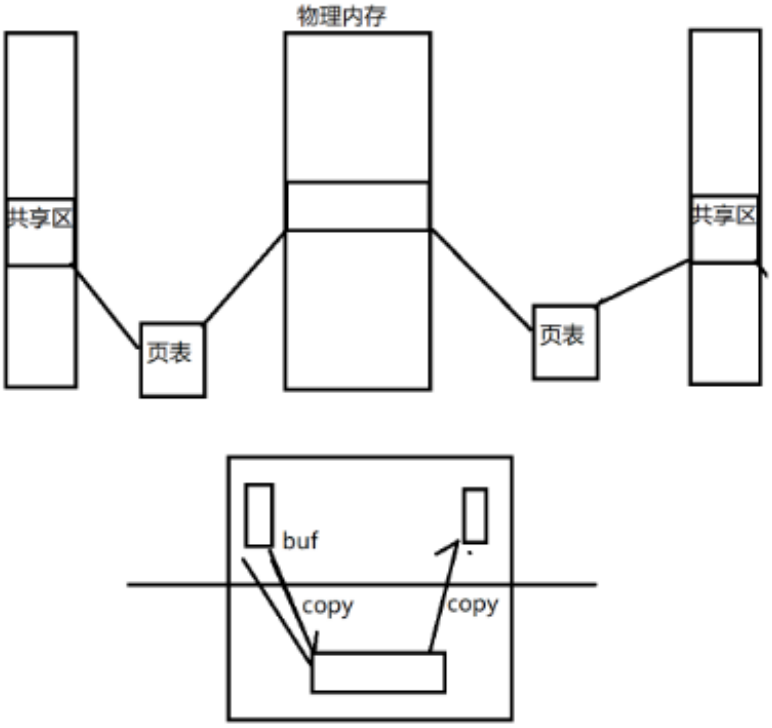
共享内存:将同一块物理内存映射到进程各自的虚拟地址空间中就可以实现数据共享，因为都可以通过自己的虚拟地址进行访问

特性:

1、最快的进程间通信方式



因为共享内存直接通过虚拟地址访问物理内存，进行数据共享

而其它通信方式，需要先将数据拷贝入内核，再从内核拷贝出来，才能实现通信

因此共享内存的通信相较于其它方式，少了两次数据拷贝操作，因此速度最快

共享内存的代码操作:

1.创建/打开共享内存

2.将共享内存映射到进程的虚拟地址空间

3.通过任意的内存操作,对共享内存进行操作memcpy/strcpy

4.解除映射关系(共享内存会记录当前的进程映射链接数)

5.删除共享内存(只有当前的进程映射链接数为0的时候才会删除，不为0，则不会立即删除，而是禁止继续链接，等到为0的时候再删除)

#include <sys/ipc.h>

#include <sys/shm.h>

1.创建/打开

int shmget(key_ key, size_t size, int shmflg);

key:是共享内存的标识符---多个进程就是通过同-一个标识符才能访问同-块共享内存
IPC PRIVATE-- -私有-只能用于具有亲缘关系的进程间通信
size:要创建的共享内存大小--以内存页为单位的
shmflg: IPC CREAT--共享内存不存在则创建,存在则打开
IPC _EXCL:与IPC _CREAT同时使用，不存在则创建，存在则报错
mode权限: 0777 IPC CREAT | 0777
返回值:共享内存在代码中的操作句柄参数中的key是共享内存的标识

命令:
ipcs查看进程间的通信资源
-m共享内存 -q消息队列 -s信号量

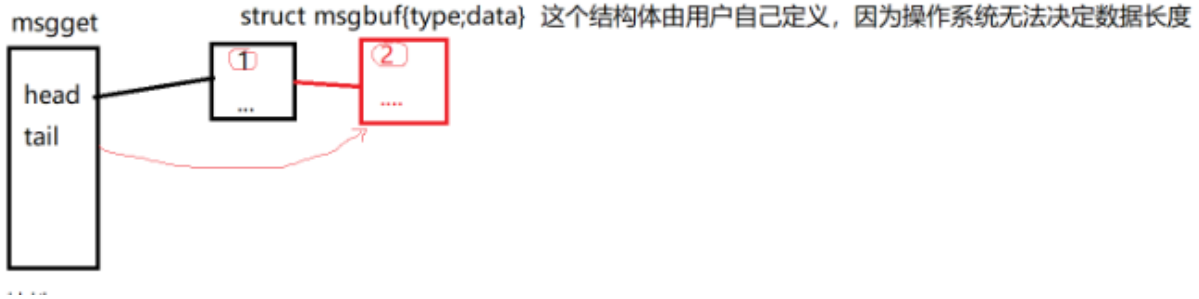
特性:生命周期随内核
进程间通信标识KEY值的生成:
#define proj id 0x12345678
key_t ftok(const char *pathname, int proj_ id);
从文件的inode节点号中取出-部分数据，从proj id中取出一部分数据进行合并生成key值
这种计算key的方法非常依赖文件---意味着如果文件被删除了,则计算得到的key会发生改变;
多个进程无法映射同一块物理内存

void *shmat(int shmid, const void *shmaddr, int shmflg);
shmid:创建共享内存时，返回的操作句柄
shmaddr:共享内存映射在虚拟地址空间中的首地址，通常置NULL,让操作系统来分配一个合适的地址即可
shmflg:在堆共享内存具有权限的情况下，可以设置在代码中的操作权限SHM_ RDONLY则为只读(前提是具有共享内存的可读权限)，默认为0时，则为可读可写
返回值:返回共享内存在虚拟地址空间中实际映射的首地址---通过这个地址就可以访问共享内存了

int shmct(int shmid, int cmd, struct shmid_ ds *buf); --- 共享内存的信息获取/设置
shmid:操作句柄
cmd:堆共享内存想要进行的操作 IPC_ RMID - 标记共享内存为将要销毁---实际销毁是在映射链接数为0的时候

buf:在设置/获取共享内存信息的时候，通过这个结构体返回数据，或者设置新数据

共享内存:
本质原理:进程间通过映射链接同-块物理内存实现数据共享
特性:
1.最快的进程间通信方式
2.生命周期随内核
3.共享内存的操作并不安全(不具备同步互斥)
多个进程同时操作有可能会造成数据二义---操作过程需要被保护
消息队列:
本质原理:在操作系统内核中创建了一个优先级队列，多个进程间通过向队列中添加数据块或者获取数据块实现数据通信



特性：
1、生命周期随内核
2、消息队列也是自带同步与互斥
3、消息队列所能存储的数据是有最大长度限制的；

信号量:
信号量本身并不是用于实现通信的，而是用于实现进程间同步与互斥的(保护进程间对临界资源进行访问时不会出现数据二义性)
同步:通过一-定的条件判断，实现进程间对临界资源访问的时序合理性
互斥:通过同一时间的唯一访问， 实现进程间对临界资源访问的安全性
临界资源:进程间都能访问到的资源
临界区:对临界资源进行操作的代码区域

本质:计数器+ pcb等待队列->使一个进程陷入休眠的接口/唤醒一个休眠进程的接口
停车场---停车位(资源) --- 合理性:有停车位就去停车，没有停车位就应该等着
停车场外的计数牌子:计数器--统计资源的数量，通过数量判断当前的访问是否合理
信号量同步的实现:

数据是资源，初始化的时候有多少数据资源，计数器就会初始化为相应的数量;
进程A去获取数据资源，若计数器>0;表示有资源，可以直接获取，计数器-1;若计数器< =0表示没有资源，则需要调用使一个进程陷入休眠的接口让进程进行等待，计数器-1;
进程B产生了- -个资源，判断若计数器<0，则调用唤醒一个休眠进程的接口 唤醒一个等待中进程A， 计数器+1;若计数器>=0,则没有人等待，则直接计数器+1;

信号量实现互斥:保证同一-时间只有一个进程能够访问资源
信号量本身是一个计数器,计数器为0的时候就会让继续访问资源的进程陷入等待,等待其它进程唤醒若一个计数器的计数最大只有1;表示只有-个资源，也就意味着只有-一个人能访问

在进程A访问临界资源时，将计数从1减成0; (其它进程这时候若继续访问就会陷入等待)
等到进程A对临界资源访问完毕，将计数器进行+1 (唤醒 了等待的进程，大家就可以重新抢夺资源了)