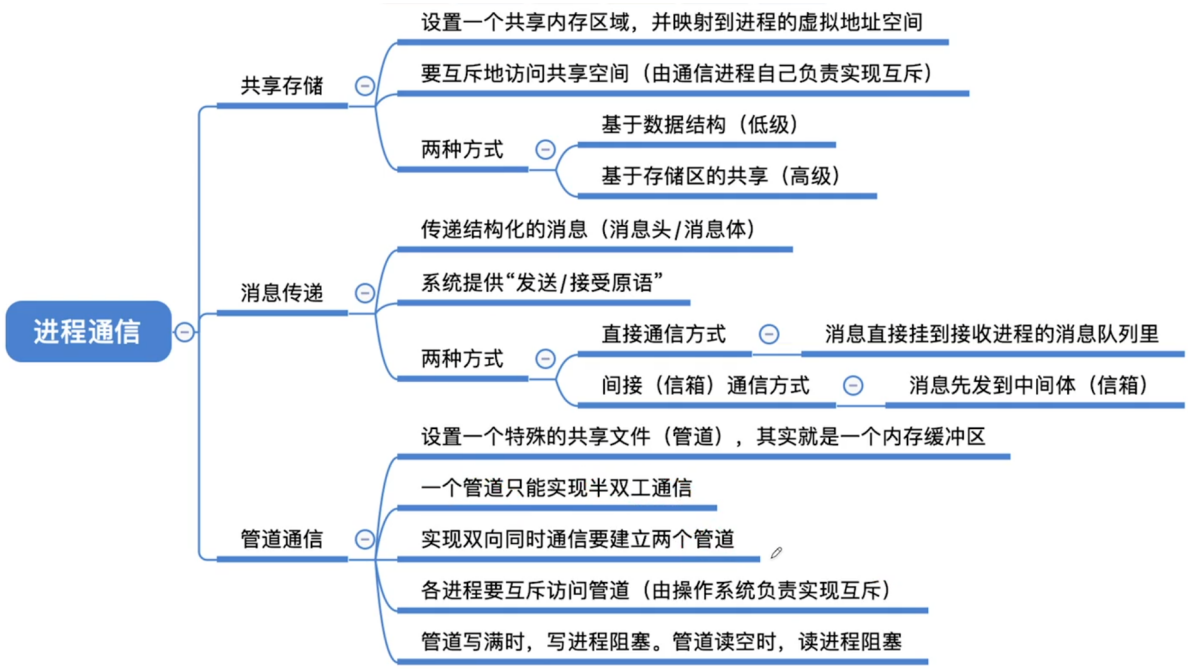
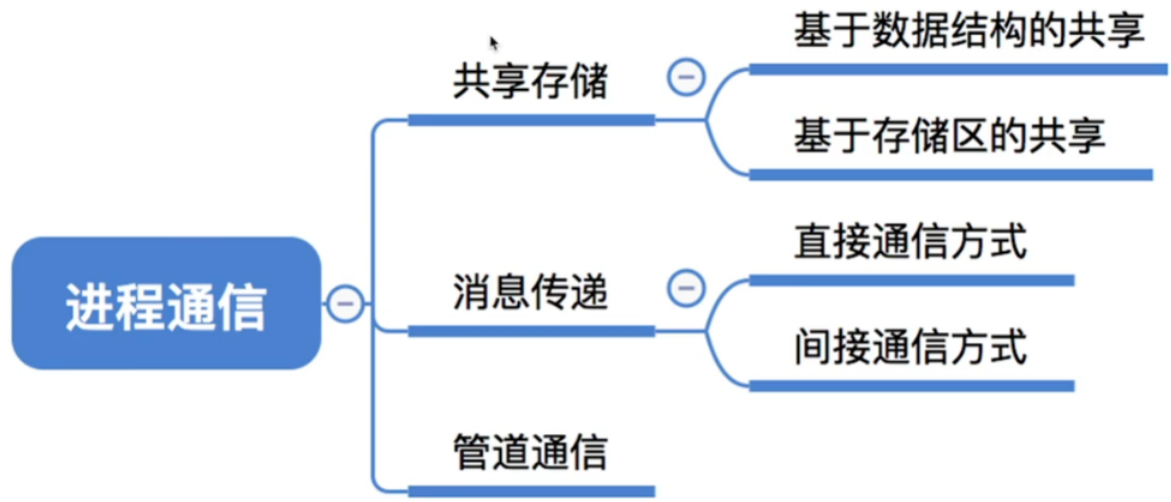
进程通信





进程间通信（Inter-Process Communication, IPC）是指两个进程之间产生数据交互。

为什么进程通信需要操作系统支持?

进程是分配系统资源的单位（包括内存地址空间），因此各进程拥有的内存地址空间相互独立。

为了保证安全，一个进程不能直接访问另一个进程的地址空间。

共享存储

Linux中，如何实现共享内存：

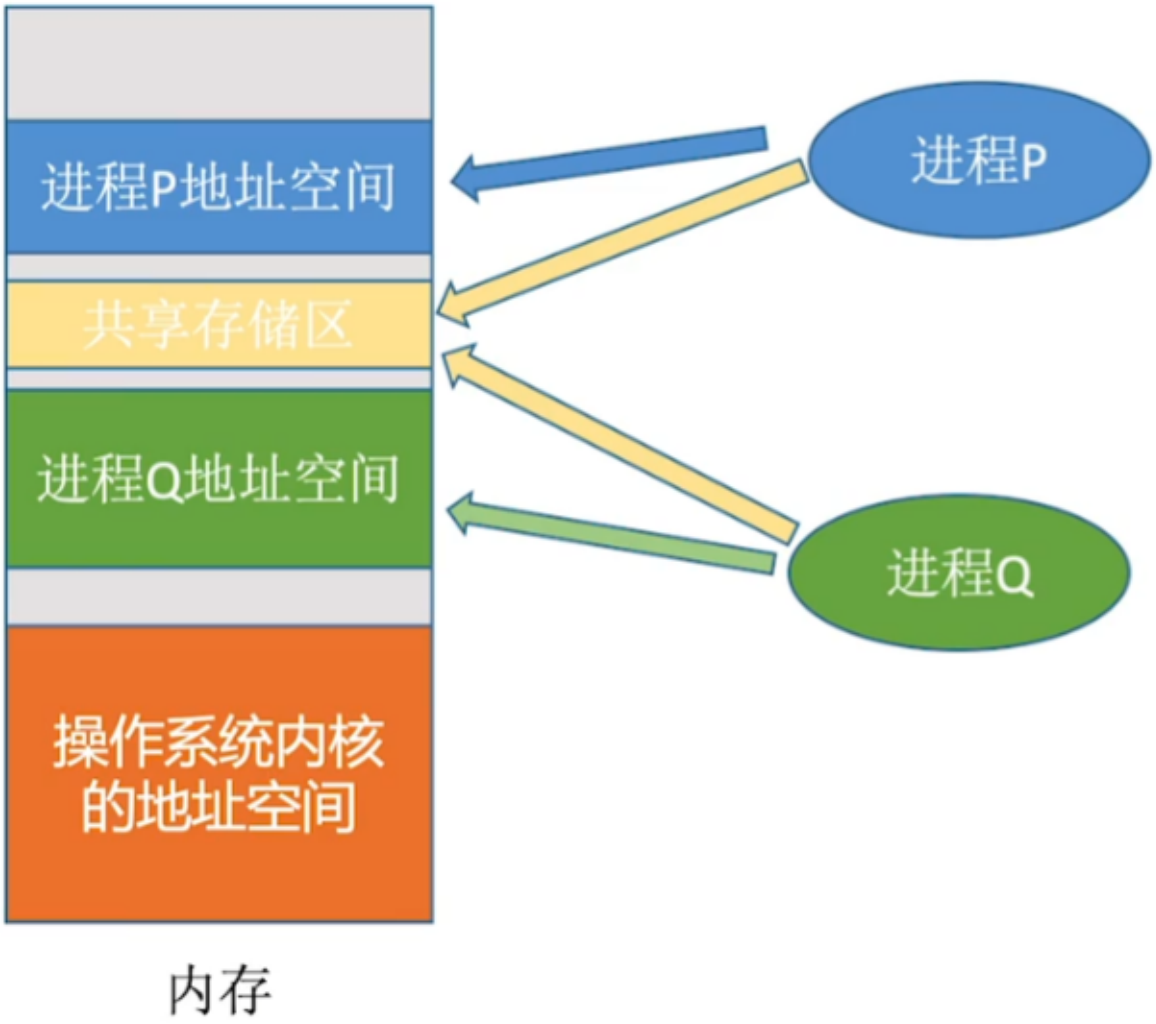
```
int shm_open(...);//通过shm_open系统调用，申请一片共享内存区
void * mmap(...);//通过mmap系统调用，将共享内存区映射到进程自己的地址空间
```

注：通过增加页表项/段表项即可将同一片共享内存区映射到各个进程的地址空间中

为避免出错，各个进程对共享空间的访问应该是互斥的。

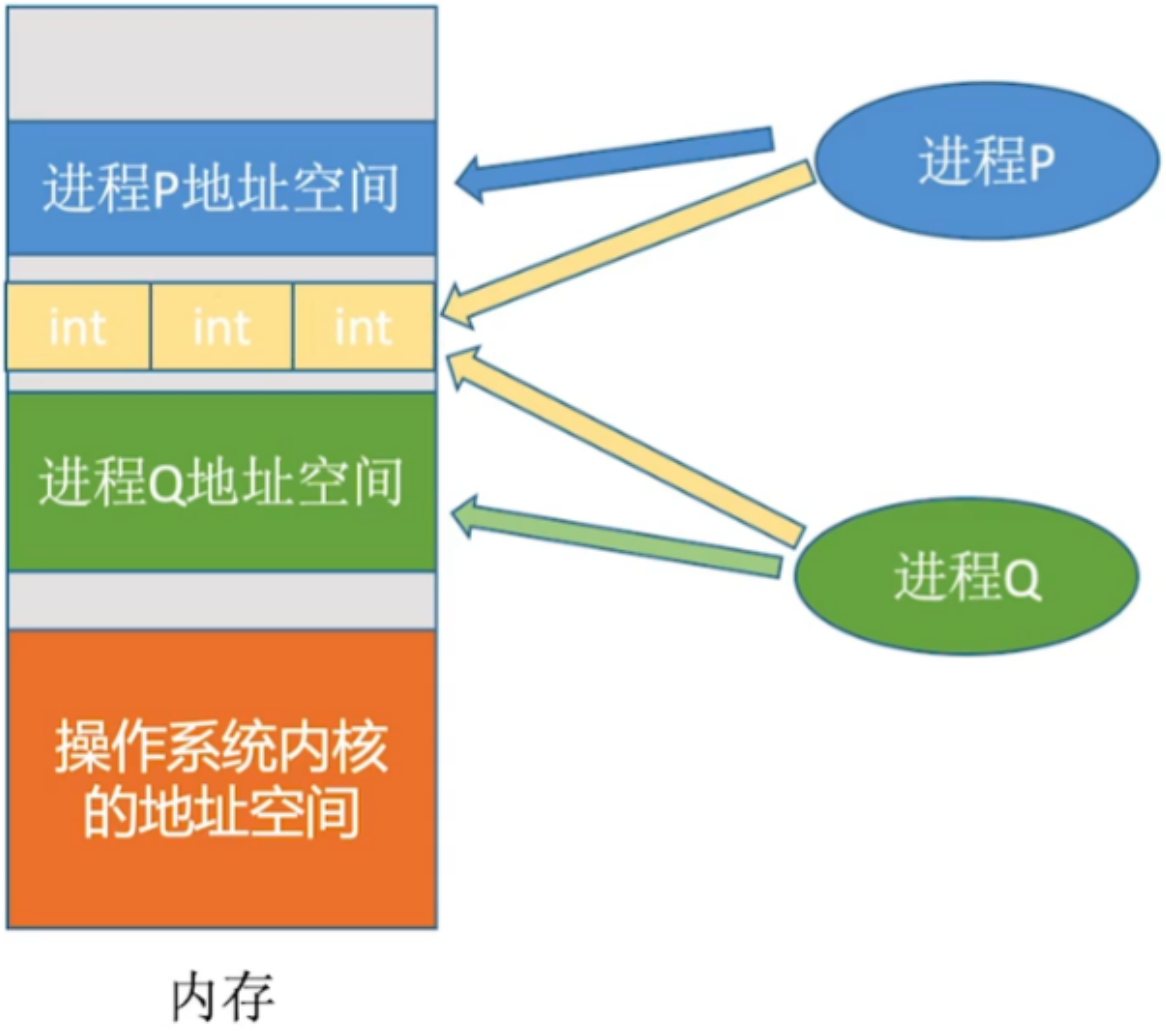
各个进程可使用操作系统内核提供的同步互斥工具（如P、V操作）

基于存储区的共享



操作系统在内存中划分一块共享存储区，数据的形式、存放位置都由通信进程控制，而不是操作系统。这种共享方式速度很快，是一种高级通信的方式。

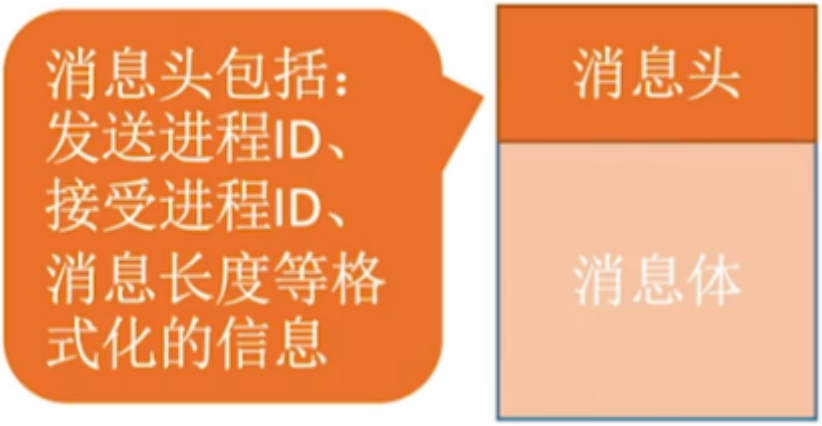
基于数据结构的共享



比如共享空间里只能放一个长度为10的数组。这种共享方式速度慢、限制多，是一种低级通信方式。

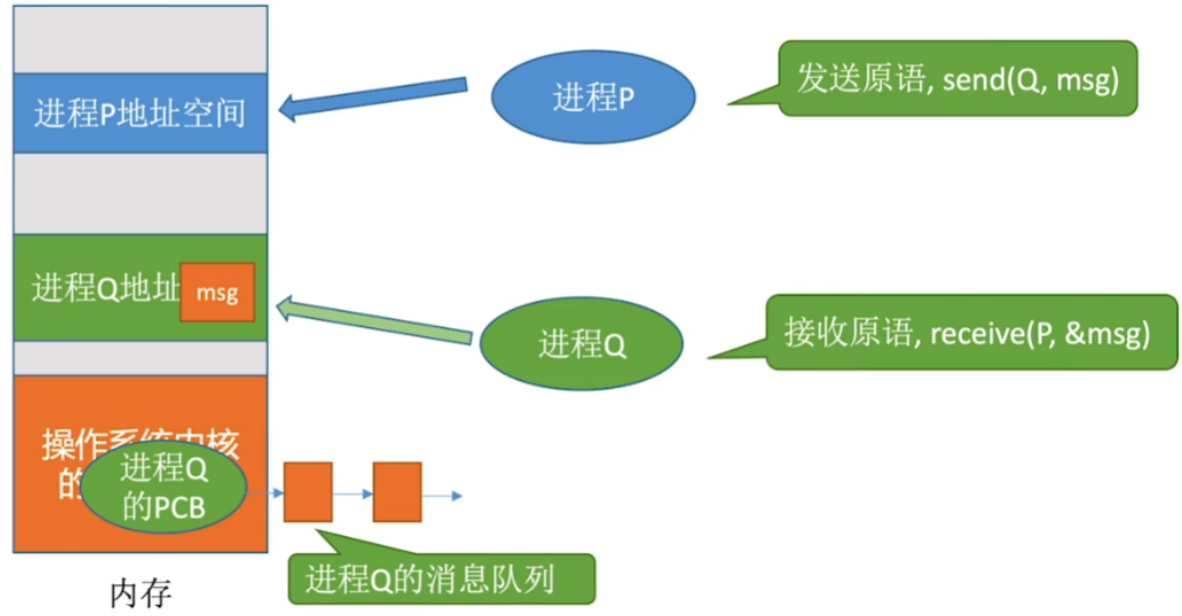
消息传递

进程间的数据交换以格式化的消息（Message）为单位。进程通过操作系统提供的”发送消息/接收消息“两个原语进行数据交换。



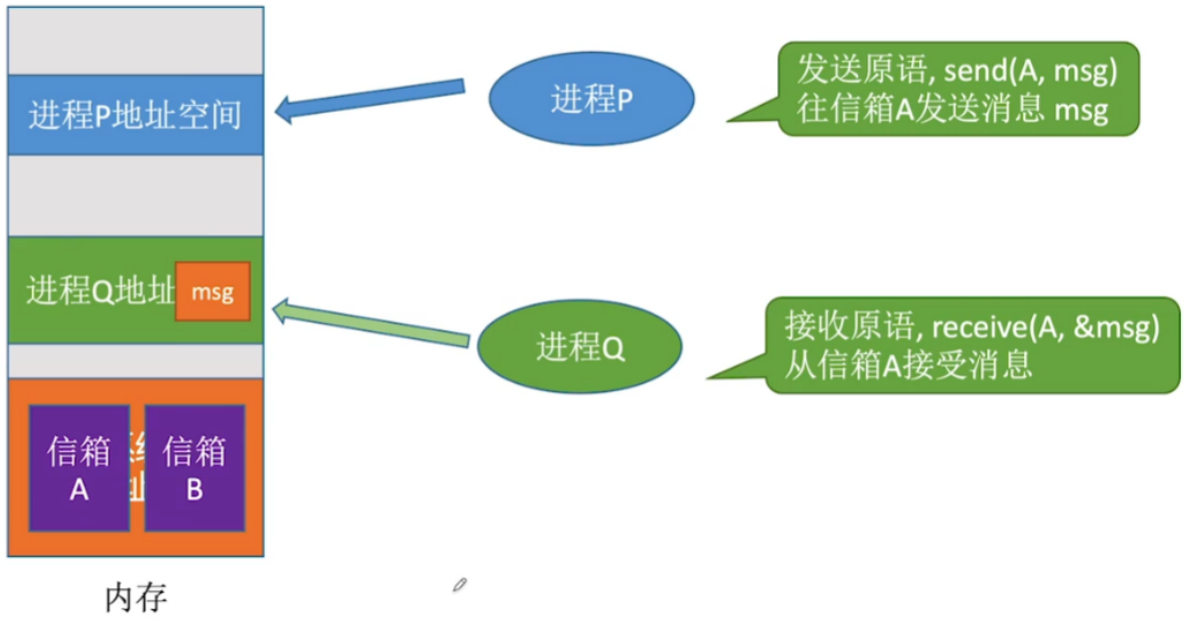
直接通信方式

消息发送进程要指明接收进程的ID，点名道姓的消息传递。



间接通信方式

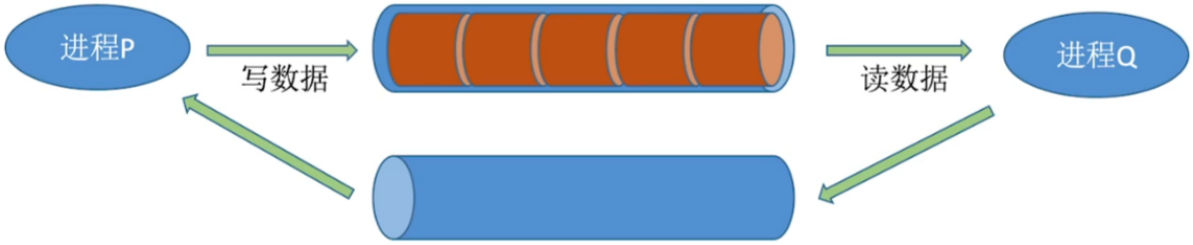
通过"信箱"间接地通信。因此又称"信箱通信方式"。以"信箱"作为中间实体进行消息传递。



可以多个进程往同一个信箱send消息，也可以多个进程从同一个信箱中receive消息

管道通信

管道是一个特殊的共享文件，又名pipe文件。其实就是在内存中开辟一个大小固定的内存缓冲区



1. 管道只能采用半双工通信，某一时间段只能实现单向的传输。如果要实现双向同时通信，则需要设置两个管道。
2. 各进程要互斥地访问管道（由操作系统实现）
3. 当管道写满时，写进程将阻塞，直到读进程将管道中的数据取走，即可唤醒写进程。
4. 当管道读空时，读进程将阻塞，直到写进程往管道中写入数据，即可唤醒读进程。
5. 管道中的数据一旦被读出，就彻底消失。因此，当多个进程读同一个管道时，可能会错乱。对此，通常有两种解决方案：
    - 一个管道允许多个写进程，一个读进程；
    - 允许有多个写进程，多个读进程，但系统会让各个读进程轮流从管道中读数据（Linux方案）。
6. 写进程往管道写数据，即便管道没被写满，只要管道没空，读进程就可以从管道读数据
7. 读进程从管道读数据，即便管道没被读空，只要管道没满，写进程就可以往管道写数据