## Inter-process communication

目录

[Inter-process communication 1](#_Toc25445)

[1.进程间通信（IPC 1](#_Toc4296)

[Ⅰ管道 1](#_Toc2960)

[Ⅱ命名管道 FIFIO 3](#_Toc15765)

[Ⅲ消息队列 6](#_Toc32478)

[Ⅳ共享内存 10](#_Toc14758)

[Ⅴ信号量 12](#_Toc16213)

[Ⅵ套接字 14](#_Toc13929)

[2.线程间通信 15](#_Toc31287)

[Ⅰ互斥锁 15](#_Toc12125)

[Ⅱ读写锁 15](#_Toc23343)

[Ⅲ条件变量 16](#_Toc20250)

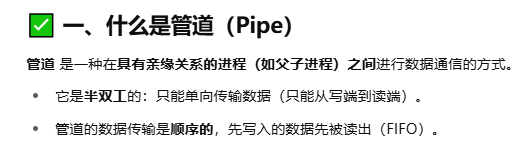
[Ⅳ信号量 16](#_Toc16801)

[Ⅴ互斥与同步 16](#_Toc8673)

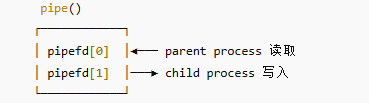
## 1.进程间通信（IPC

管道、消息队列、信号量、共享内存、套接字

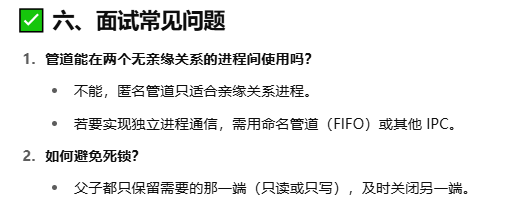
### Ⅰ管道







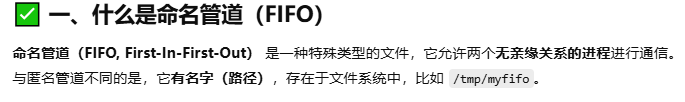




我们平时写的应用文件里的fd是文件描述符，不是管道



### Ⅱ命名管道 FIFIO









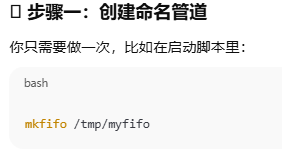
命名管道的使用：

①创建

因为管道是单向的，有一个写端，

②写端写入数据

③读端读取数据





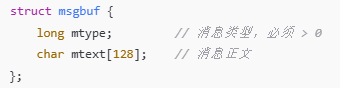




### Ⅲ消息队列

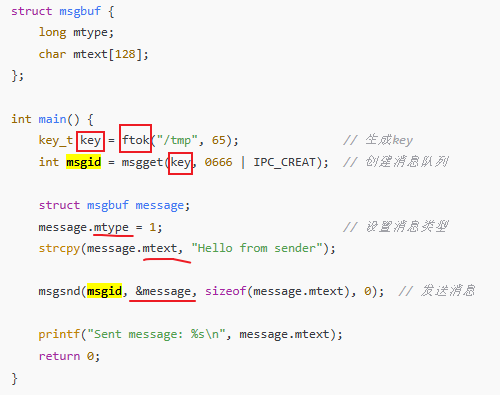






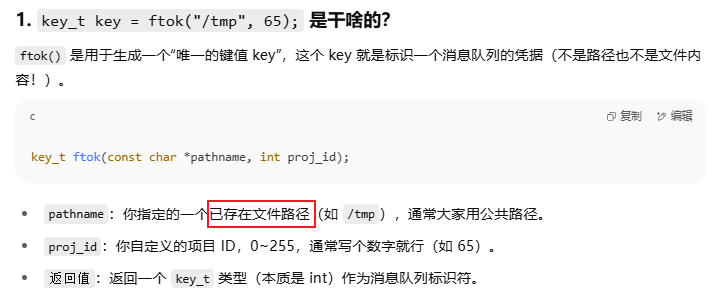
发送使用方法：

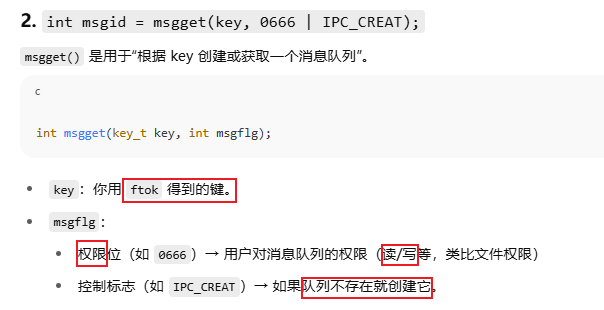
（吐槽）创建消息队列这么麻烦的吗



队列头/id

消息队列创建相关API：





接收消息队列：（只接收队列头（id）为1的消息队列

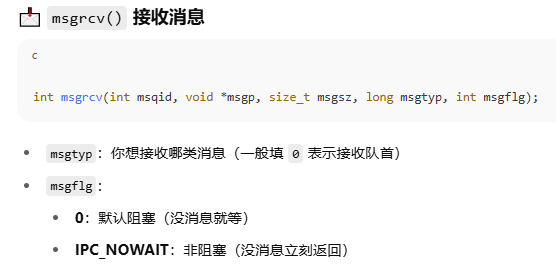






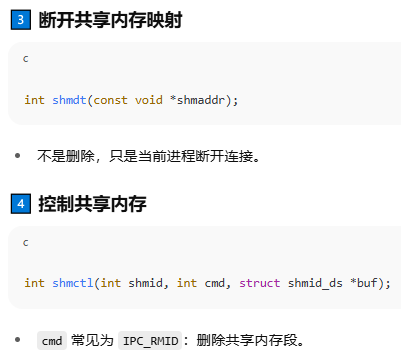






### Ⅳ共享内存











**共享内存 + 信号量** 的完整通信示例：

（（（（（待补充

### Ⅴ信号量

①有名/无名信号量（对比system V更高级

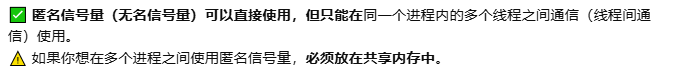


有名信号量实例：





为什么进程间通信，无名信号量要放到共享内存中：

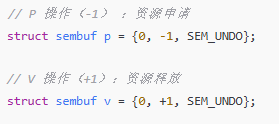


匿名信号量是使用 sem\_init() 创建的，不绑定系统名字（不像 sem\_open()）。这种信号量直接存在于用户定义的变量中



②System V信号量





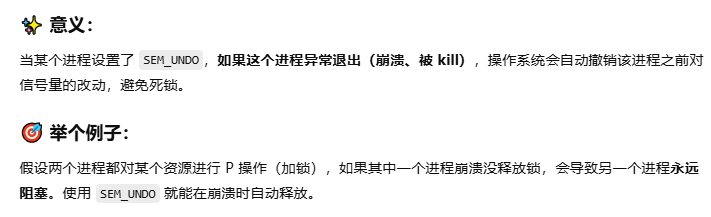




SEM\_UNDO这个参数的含义

struct sembuf p = {0, -1, SEM\_UNDO};

信号量编号， P/V操作， xxx参数



### Ⅵ套接字

这个太复杂了，涉及网络知识，暂不学习。

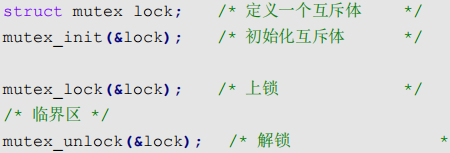
## 2.线程间通信

比起进程复杂的通信机制（管道、匿名管道、消息队列、信号量、共享内存、内存映射以及socket等），线程间通信要简单的多。

因为同一进程的不同线程共享同一份全局内存区域，其中包括初始化数据段、未初始化数据段，以及堆内存段，所以线程之间可以方便、快速地共享信息。只需要将数据复制到共享（全局或堆）变量中即可。不过，要避免出现多个线程试图同时修改同一份信息。

### Ⅰ互斥锁

互斥量本质上说是一把锁，在访问共享资源前对互斥量进行加锁，在访问完成后释放互斥量。对互斥量进行枷锁以后，其他线程再次对互斥量加锁的线程都会被阻塞直到当前线程释放该互斥锁。如果释放互斥量时有一个以上的线程阻塞，那么所有该锁上的阻塞线程都会变成可运行状态，第一个变成运行状态的线程可以对互斥量加锁，其他线程就会看到互斥量依然是锁着，只能再次阻塞等待它重新变成可用，这样，一次只有一个线程可以向前执行。



互斥锁形成的临界区内的变量受保护。执行时不会被打断，被中断打扰。

### Ⅱ读写锁

### Ⅲ条件变量

条件变量是线程可用的另一种同步机制。互斥量用于上锁，条件变量则用于等待，并且条件变量总是需要与互斥量一起使用，运行线程以无竞争的方式等待特定的条件发生。

条件变量本身是由互斥量保护的，线程在改变条件变量之前必须首先锁住互斥量。其他线程在获得互斥量之前不会察觉到这种变化，因为互斥量必须在锁定之后才能计算条件。

### Ⅳ信号量

### Ⅴ互斥与同步

互斥与同步的区别：

互斥：是指某一资源同时只允许一个访问者对其进行访问，具有唯一性和排它性。但互斥无法限制访问者对资源的访问顺序，即访问是无序的。

同步：主要是流程上的概念，是指在互斥的基础上（大多数情况），通过其它机制实现访问者对资源的有序访问。在大多数情况下，同步已经实现了互斥，特别是所有写入资源的情况必定是互斥的。少数情况是指可以允许多个访问者同时访问资源。

互斥锁、条件变量和信号量的区别：

互斥锁：互斥，一个线程占用了某个资源，那么其它的线程就无法访问，直到这个线程解锁，其它线程才可以访问。

条件变量：同步，一个线程完成了某一个动作就通过条件变量发送信号告诉别的线程，别的线程再进行某些动作。条件变量必须和互斥锁配合使用。

信号量：同步，一个线程完成了某一个动作就通过信号量告诉别的线程，别的线程再进行某些动作。而且信号量有一个更加强大的功能，信号量可以用作为资源计数器，把信号量的值初始化为某个资源当前可用的数量，使用一个之后递减，归还一个之后递增。

另外还有以下几点需要注意：

1、信号量可以模拟条件变量，因为条件变量和互斥量配合使用，相当于信号量模拟条件变量和互斥量的组合。在生产者消费者线程池中，生产者生产数据后就会发送一个信号 pthread\_cond\_signal通知消费者线程，消费者线程通过pthread\_cond\_wait等待到了信号就可以继续执行。这是用条件变量和互斥锁实现生产者消费者线程的同步，用信号量一样可以实现！

2、信号量可以模拟互斥量，因为互斥量只能为加锁或解锁（0 or 1），信号量值可以为非负整数，也就是说，一个互斥量只能用于一个资源的互斥访问，它不能实现多个资源的多线程互斥问题。信号量可以实现多个同类资源的多线程互斥和同步。当信号量为单值信号量时，就完成一个资源的互斥访问。前面说了，信号量主要用做多线程多任务之间的同步，而同步能够控制线程访问的流程，当信号量为单值时，必须有线程释放，其他线程才能获得，同一个时刻只有一个线程在运行（注意，这个运行不一定是访问资源，可能是计算）。如果线程是在访问资源，就相当于实现了对这个资源的互斥访问。

3、互斥锁是为上锁而优化的；条件变量是为等待而优化的； 信号量既可用于上锁，也可用于等待，因此会有更多的开销和更高的复杂性。

4、互斥锁，条件变量都只用于同一个进程的各线程间，而信号量（有名信号量）可用于不同进程间的同步。当信号量用于进程间同步时，要求信号量建立在共享内存区。

5、互斥量必须由同一线程获取以及释放，信号量和条件变量则可以由一个线程释放，另一个线程得到。

6、信号量的递增和减少会被系统自动记住，系统内部的计数器实现信号量，不必担心丢失，而唤醒一个条件变量时，如果没有相应的线程在等待该条件变量，此次唤醒会被丢失。

Ⅵ

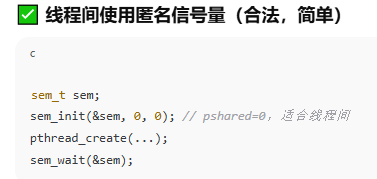
Ⅶ

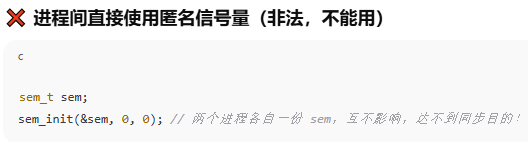
## 注)进程通信&线程通信中信号量的区分

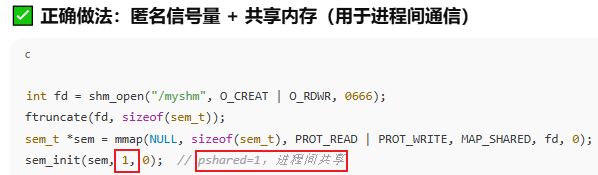












**总结：**

