**一、进程/线程间同步机制**

临界区、互斥区、事件、信号量四种方式

临界区（Critical Section）、互斥量（Mutex）、信号量（Semaphore）、事件（Event）的区别

1、临界区：通过对多线程的串行化来访问公共资源或一段代码，速度快，适合控制数据访问。在任意时刻只允许一个线程对共享资源进行访问，如果有多个线程试图访问公共资源，那么在有一个线程进入后，其他试图访问公共资源的线程将被挂起，并一直等到进入临界区的线程离开，临界区在被释放后，其他线程才可以抢占。

2、互斥量：采用互斥对象机制。 只有拥有互斥对象的线程才有访问公共资源的权限，因为互斥对象只有一个，所以能保证公共资源不会同时被多个线程访问。互斥不仅能实现同一应用程序的公共资源安全共享，还能实现不同应用程序的公共资源安全共享 .互斥量比临界区复杂。因为使用互斥不仅仅能够在同一应用程序不同线程中实现资源的安全共享，而且可以在不同应用程序的线程之间实现对资源的安全共享。

3、信号量：它允许多个线程在同一时刻访问同一资源，但是需要限制在同一时刻访问此资源的最大线程数目 .信号量对象对线程的同步方式与前面几种方法不同，信号允许多个线程同时使用共享资源，这与操作系统中的PV操作相同。它指出了同时访问共享资源的线程最大数目。它允许多个线程在同一时刻访问同一资源，但是需要限制在同一时刻访问此资源的最大线程数目。

PV操作及信号量的概念都是由荷兰科学家E.W.Dijkstra提出的。信号量S是一个整数，S大于等于零时代表可供并发进程使用的资源实体数，但S小于零时则表示正在等待使用共享资源的进程数。

P操作申请资源：

　　（1）S减1；

　　（2）若S减1后仍大于等于零，则进程继续执行；

　　（3）若S减1后小于零，则该进程被阻塞后进入与该信号相对应的队列中，然后转入进程调度。

V操作释放资源：

　　（1）S加1；

　　（2）若相加结果大于零，则进程继续执行；

　　（3）若相加结果小于等于零，则从该信号的等待队列中唤醒一个等待进程，然后再返回原进程继续执行或转入进程调度。

4、事 件： 通过通知操作的方式来保持线程的同步，还可以方便实现对多个线程的优先级比较的操作 .

**总结：**

　　1． 互斥量与临界区的作用非常相似，但互斥量是可以命名的，也就是说它可以跨越进程使用。所以创建互斥量需要的资源更多，所以如果只为了在进程内部是用的话使用临界区会带来速度上的优势并能够减少资源占用量。因为互斥量是跨进程的互斥量一旦被创建，就可以通过名字打开它。

　　2． 互斥量（Mutex），信号灯（Semaphore），事件（Event）都可以被跨越进程使用来进行同步数据操作，而其他的对象与数据同步操作无关，但对于进程和线程来讲，如果进程和线程在运行状态则为无信号状态，在退出后为有信号状态。所以可以使用WaitForSingleObject来等待进程和线程退出。

　　3． 通过互斥量可以指定资源被独占的方式使用，但如果有下面一种情况通过互斥量就无法处理，比如现在一位用户购买了一份三个并发访问许可的数据库系统，可以根据用户购买的访问许可数量来决定有多少个线程/进程能同时进行数据库操作，这时候如果利用互斥量就没有办法完成这个要求，信号灯对象可以说是一种资源计数器。

**二、进程间通信方式**

由于比较容易混淆，我们把进程间通信方法也列在这里做比较。

程间通信就是在不同进程之间传播或交换信息，那么不同进程之间存在着什么双方都可以访问的介质呢？进程的用户空间是互相独立的，一般而言是不能互相访问的，唯一的例外是共享内存区。但是，系统空间却是“公共场所”，所以内核显然可以提供这样的条件。除此以外，那就是双方都可以访问的外设了。在这个意义上，两个进程当然也可以通过磁盘上的普通文件交换信息，或者通过“注册表”或其它数据库中的某些表项和记录交换信息。广义上这也是进程间通信的手段，但是一般都不把这算作“进程间通信”。因为那些通信手段的效率太低了，而人们对进程间通信的要求是要有一定的实时性。

进程间通信主要包括管道, 系统IPC(包括消息队列,信号量,共享存储), SOCKET.

管道分为有名管道和无名管道，无名管道只能用于亲属进程之间的通信，而有名管道则可用于无亲属关系的进程之间。

消息队列用于运行于同一台机器上的进程间通信，与管道相似；

共享内存通常由一个进程创建，其余进程对这块内存区进行读写。得到共享内存有两种方式：映射/dev/mem设备和内存映像文件。前一种方式不给系统带来额外的开销，但在现实中并不常用，因为它控制存取的是实际的物理内存；

本质上，信号量是一个计数器，它用来记录对某个资源（如共享内存）的存取状况。一般说来，为了获得共享资源，进程需要执行下列操作：

　　（1）测试控制该资源的信号量；

　　（2）若此信号量的值为正，则允许进行使用该资源，进程将进号量减1；

　　（3）若此信号量为0，则该资源目前不可用，进程进入睡眠状态，直至信号量值大于0，进程被唤醒，转入步骤（1）；

　　（4）当进程不再使用一个信号量控制的资源时，信号量值加1，如果此时有进程正在睡眠等待此信号量，则唤醒此进程。

套接字通信并不为Linux所专有，在所有提供了TCP/IP协议栈的操作系统中几乎都提供了socket，而所有这样操作系统，对套接字的编程方法几乎是完全一样的

**三、进程/线程同步机制与进程间通信机制比较**

很明显2者有类似，但是差别很大

同步主要是临界区、互斥、信号量、事件

进程间通信是管道、内存共享、消息队列、信号量、socket

共通之处是，信号量和消息（事件）

**其他资料：**

进程间通讯(IPC)方法主要有以下几种：

管道/FIFO/共享内存/消息队列/信号

1.管道中还有命名管道和非命名管道(即匿名管道)之分，非命名管道(即匿名管道)只能用于父子进程通讯，命名管道可用于非父子进程，命名管道就是FIFO，管道是先进先出的通讯方式

2.消息队列是用于两个进程之间的通讯，首先在一个进程中创建一个消息队列，然后再往消息队列中写数据，而另一个进程则从那个消息队列中取数据。需要注意的是，消息队列是用创建文件的方式建立的，如果一个进程向某个消息队列中写入了数据之后，另一个进程并没有取出数据，即使向消息队列中写数据的进程已经结束，保存在消息队列中的数据并没有消失，也就是说下次再从这个消息队列读数据的时候，就是上次的数据。

3.信号量，它与WINDOWS下的信号量是一样的，所以就不用多说了

4.共享内存，类似于WINDOWS下的DLL中的共享变量，但LINUX下的共享内存区不需要像DLL这样的东西，只要首先创建一个共享内存区，其它进程按照一定的步骤就能访问到这个共享内存区中的数据，当然可读可写。

**以上几种方式的比较：**

1.管道：速度慢，容量有限，只有父子进程能通讯

2.FIFO：任何进程间都能通讯，但速度慢

3.消息队列：容量受到系统限制，且要注意第一次读的时候，要考虑上一次没有读完数据的问题

4.信号量：不能传递复杂消息，只能用来同步

5.共享内存区：能够很容易控制容量，速度快，但要保持同步，比如一个进程在写的时候，另一个进程要注意读写的问题，相当于线程中的线程安全，当然，共享内存区同样可以用作线程间通讯，不过没这个必要，线程间本来就已经共享了同一进程内的一块内存