Part------操作系统专题

1.进程间的通信方式（IPC）？进程调度方法？

1. 共享内存

可以说这是最有用的进程间通信方式。它使得多个进程可以访问同一块内存空间，不同进程可以及时看到对方进程中对共享内存中数据得更新。这种方式需要依靠某种同步操作，如互斥锁和信号量等。

2. 消息队列

“消息队列”是在消息的传输过程中保存消息的容器。具有写权限得进程可以按照一定得规则向消息队列中添加新信息；对消息队列有读权限得进程则可以从消息队列中读取信息。

3. 信号

信号是一种比较复杂的通信方式，用于通知接收进程某个事件已经发生。

4. 信号量

信号量是一个计数器，可以用来控制多个进程对共享资源的访问。它常作为一种锁机制，防止某进程正在访问共享资源时，其他进程也访问该资源。因此，主要作为进程间以及同一进程内不同线程之间的同步手段。

5. 套接字

这是一种更为一般得进程间通信机制，它可用于网络中不同机器之间的进程间通信，应用非常广泛。

6. 普通管道

普通管道是一种半双工的通信方式，数据只能单向流动，而且只能在具有父子关系的进程间使用。

7. 有名管道

有名管道也是半双工的通信方式，但是它允许无亲缘关系进程间的通信。

进程调度方法：

1. 先来先服务调度算法。
2. 短作业（进程）优先调度算法。
3. 优先权调度算法。
4. 高响应比优先调度算法。
5. 基于时间片的轮转调度算法。
6. 多级反馈队列调度算法。

2.线程间通信的方式？

锁机制：包括互斥锁、条件变量、读写锁

*互斥锁提供了以排他方式防止数据结构被并发修改的方法。

*读写锁允许多个线程同时读共享数据，而对写操作是互斥的。

*条件变量可以以原子的方式阻塞进程，直到某个特定条件为真为止。对条件的测试是在互斥锁的保护下进行的。条件变量始终与互斥锁一起使用。

信号量机制(Semaphore)：包括无名线程信号量和命名线程信号量。
信号机制(Signal)：类似进程间的信号处理。

线程间的通信目的主要是用于线程同步，所以线程没有像进程通信中的用于数据交换的通信机制。

3.操作系统由哪些部分组成？

进程管理，存储管理，设备管理，文件管理，程序接口，用户界面。

4.用户态和系统态在什么时候进行切换？平时用的都是 64 位系统，那它和 32 位系统相比，有什么区别和优点？

以下三种情况会导致用户态到内核态的切换：
1. 系统调用；
2. 异常，比如缺页异常；
3. 外围设备的中断，当外围设备完成用户请求的操作后，会向 CPU 发出相应的中断信号。

1）寻址能力不同；2)运算速度不同。

5.选择一个你熟悉的磁盘臂调度算法进行简单描述。

1. 先来先服务
2. 最短寻道时间优先
3. SCAN 算法
4. CSCAN 算法

6.进程和线程的区别？

(1)调度：线程作为调度和分配的基本单位，进程作为拥有资源的基本单位 。

(2)并发性：不仅进程之间可以并发执行，同一个进程的多个线程之间也可并发执行。

(3)拥有资源：进程是拥有资源的一个独立单位，线程自己基本上不拥有系统资源，只拥有一点在运行中必不可少的资源（如程序计数器、一组寄存器和栈），但是它可以与同属一个进程的其他线程共享进程所拥有的全部资源。进程之间是不能共享地址空间的，而线程是共享着所在进程的地址空间的。

(4)系统开销：在创建或撤消进程时，由于系统都要为之分配和回收资源，导致系统的开销明显大于创建或撤消线程时的开销。

7.操作系统的换页方法。

FIFO、LRU、OPT 等。

8.哲学家进餐问题的无死锁算法。

A.原理：仅当哲学家的左右两支筷子都可用时,才允许他拿起筷子进餐。

方法 1：利用 AND 型信号量机制实现：根据课程讲述，在一个原语中，将一段代码同时需要的多个临界资源，要么全部分配给它，要么一个都不分配，因此不会出现死锁的情形。当某些资源不够时阻塞调用进程;由于等待队列的存在，使得对资源的请求满足 FIFO 的要求，因此不会出现饥饿的情形。

伪码：

```
semaphore chopstick[5]={1，1，1，1，1};
void philosopher(int I)
{
while(true)
{
think();
Swait(chopstick[(I+1)]%5,chopstick[I]);
eat();
Ssignal(chopstick[(I+1)]%5,chopstick[I]);
}
}
```

方法 2：利用信号量的保护机制实现。通过信号量 mutex 对 eat（）之前的取左侧和右侧筷子的操作进行保护，使之成为一个原子操作，这样可以防止死锁的出现。

伪码：

```
semaphore mutex = 1 ;
semaphore chopstick[5]={1，1，1，1，1};
void philosopher(int I)
{
while(true)
{
think();
wait(mutex);
wait(chopstick[(I+1)]%5);
wait(chopstick[I]);
signal(mutex);
```

```
eat();
signal(chopstick[(I+1)]%5);
signal(chopstick[I]);
}
}
```

B. 原理：规定奇数号的哲学家先拿起他左边的筷子, 然后再去拿他右边的筷子; 而偶数号的哲学家则相反。按此规定, 将是 1, 4 号哲学家竞争 4 号筷子, 2, 3 号哲学家竞争 2 号筷子. 即五个哲学家都竞争偶数号筷子, 获得后, 再去竞争奇数号筷子, 最后总会有一个哲学家能获得两支筷子而进餐。而申请不到的哲学家进入阻塞等待队列，根据 FIFO 原则，则先申请的哲学家会较先可以吃饭，因此不会出现饿死的哲学家。伪码：

```
semaphore chopstick[5]={1，1，1，1，1};
void philosopher(int i)
{
while(true)
{
think();
if(i%2 == 0) //偶数哲学家，先右后左。
{
wait (chopstick[ i + 1 ] mod 5) ;
wait (chopstick[ i]) ;
eat();
signal (chopstick[ i + 1 ] mod 5) ;
signal (chopstick[ i]) ;
}
Else //奇数哲学家，先左后右。
{
wait (chopstick[ i]) ;
wait (chopstick[ i + 1 ] mod 5) ;
eat();
signal (chopstick[ i]) ;
signal (chopstick[ i + 1 ] mod 5) ;
}
}
```

9.操作系统的内存管理。

1 分区管理

2 分页管理

3 分段管理

4.段页式管理

Part------操作系统专题（补 1）

线程同步的机制。

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 。

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。

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。