操作系统（2）

- 进程和线程的区别?
- 进程　　线程
- 定义　　进程是资源分配的最小单位。　　线程是程序执行的最小单位。
- 关系　　一个进程可以拥有多个线程。　　一个线程只能属于一个进程。
- 资源　　进程拥有独立的资源，系统为每一个进程分配独立的地址空间，建立数据表来维护代码段、堆栈段和数据段。　　同一进程内的线程共享本进程的资源。 线程是共享进程中的数据的，使用相同的地址空间。
- 通信　　进程之间的通信需要以通信的方式（IPC）进行。　　线程之间的通信更方便，同一进程下的线程共享全局变量、静态变量等数据。不过如何处理好同步与互斥是编写多线程程序的难点。
- 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。　　CPU创建和切换一个线程的花费比进程要小很多。
- 健壮性　　多进程程序更健壮。一个进程死掉并不会对另外一个进程造成影响，因为进程有自己独立的地址空间。　　多线程程序只要有一个线程死掉，整个进程也死掉了。
- 进程间通信（IPC，Inter-Process Communication）方式
- 1. 共享内存
- 多个进程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的更新，该方式需要依靠某种同步操作，如互斥锁、信号量等。
- 2. 消息队列
- 在消息的传输过程中保存消息的容器。具有写权限的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
- 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，用来控制多个进程对共享资源的访问。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间的同步手段。
- 5. 套接字
- 可用于网络中不同机器之间的进程间通信，是一种更为一般的进程间通信机制，应用广泛。
- 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。
- 线程间通信方式
- 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。
- 1. 锁机制（本质是共享内存）
- ·　　互斥锁（ReentrantLock）提供了以排他方式防止数据结构被并发修改的方法。

- · 条件变量（Condition）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。
- · 读写锁（ReentrantReadWriteLock）允许多个线程同时读共享数据，而对写操作是互斥的。
- 2. 信号量机制（Semaphore）
- 信号量包括无名线程信号量和命名线程信号量：无名信号量只用于线程间的同步；有名信号量只用于进程间通信。
- 3. 信号机制（Signal）
- 类似进程间的信号处理。
- 进程调度算法
- 1.先来先服务 （FCFS，first come first served）
- 进程按照请求CPU的顺序使用CPU。
- 优点：易于理解且实现简单，只需要一个队列(FIFO)，且相当公平。
- 缺点：比较有利于长进程，而不利于短进程，有利于CPU 繁忙的进程，而不利于I/O 繁忙的进程。
- 2.最短作业优先（SJF，Shortest Job First） /短进程优先(SPN，Shortest Process Next)
- 对预计执行时间短的进程优先分派处理机。通常后来的短进程不抢先正在执行的进程。
- 优点：相比FCFS 算法，该算法可改善平均周转时间和平均带权周转时间，缩短进程的等待时间，提高系统的吞吐量。
- 缺点：对长进程非常不利，可能长时间得不到执行，且未能依据进程的紧迫程度来划分执行的优先级，以及难以准确估计进程的执行时间，从而影响调度性能。
- 3.最高响应比优先法(HRRN，Highest Response Ratio Next)
- HRRN调度策略同时考虑每个作业的等待时间长短和估计需要的执行时间长短，从中选出响应比最高的作业投入执行。这种算法是介于FCFS和SJF之间的一种折中算法。
- 原理：响应比R定义如下： R =(W+T)/T = 1+W/T
- 其中T为该作业估计需要的执行时间，W为作业在后备状态队列中的等待时间。每当要进行作业调度时，系统计算每个作业的响应比，选择其中R最大者投入执行。
- 优点：由于长作业也有机会投入运行。
- 缺点：由于每次调度前要计算响应比，系统开销也要相应增加。
- 4.时间片轮转算法（RR，Round-Robin）
- 采用剥夺策略。时间片轮转调度是一种最古老，最简单，最公平且使用最广的算法，又称RR调度。每个进程被分配一个时间段，称作它的时间片，即该进程允许运行的时间。
- 原理：让就绪进程以FCFS 的方式按时间片轮流使用CPU 的调度方式，即将系统中所有的就绪进程按照FCFS 原则排成一个队列，每次调度时将CPU 分派给队首进程，让其执行一个时间片，时间片的长度从几个ms 到几百ms。在一个时间片结束时，发生时钟中断，调度程序据此暂停当前进程的执行，将其送到就绪队列的末尾，并通过上下文切换执行当前的队首进程，进程可以未使用完一个时间片，就出让CPU（如阻塞）。
- 算法优点：时间片轮转调度算法的特点是简单易行、平均响应时间短。
- 算法缺点：不利于处理紧急作业。在时间片轮转算法中，时间片的大小对系统性能的影响很大，因此时间片的大小应选择恰当。
- 怎样确定时间片的大小：
- · 系统对响应时间的要求
- · 就绪队列中进程的数目
- · 系统的处理能力
- 5.多级反馈队列(MultilevelFeedback Queue)

- 多级反馈队列调度算法是一种CPU处理机调度算法，UNIX操作系统采取的便是这种调度算法。
- 原理：
- a. 进程在进入待调度的队列等待时，首先进入优先级最高的Q1等待。
- b. 首先调度优先级高的队列中的进程。若高优先级中队列中已没有调度的进程，则调度次优先级队列中的进程。例如：Q1,Q2,Q3三个队列，只有在Q1中没有进程等待时才去调度Q2，同理，只有Q1,Q2都为空时才会去调度Q3。
- c. 对于同一个队列中的各个进程，按照时间片轮转法调度。比如Q1队列的时间片为N，那么Q1中的作业在经历了N个时间片后若还没有完成，则进入Q2队列等待，若Q2的时间片用完后作业还不能完成，一直进入下一级队列，直至完成。
- d. 在低优先级的队列中的进程在运行时，又有新到达的作业，那么在运行完这个时间片后，CPU马上分配给新到达的作业（抢占式）。
- 在多级反馈队列调度算法中，如果规定第一个队列的时间片略大于多数人机交互所需之处理时间时，便能够较好的满足各种类型用户的需要。
- 用户态(User Mode)和内核态(Kernel Mode)
- 用户态和内核态是操作系统的两种运行级别，用于区分不同程序的不同权利。
- · 用户态：只能受限地访问内存，且不允许访问外围设备，占用CPU的能力被剥夺，CPU资源可以被其他程序获取。
- · 内核态：CPU可以访问内存所有数据和外围设备的数据（如硬盘、网卡），CPU也可以将自己从一个程序切换到另一个程序。
- 用户态与内核态的切换
- 所有用户程序都是运行在用户态的，但是有时候程序确实需要做一些内核态的事情，例如从硬盘读取数据，或者从键盘获取输入等。而唯一可以做这些事情的就是操作系统，所以此时程序就需要先请求操作系统以程序的名义来执行这些操作。
- 系统调用机制：用户态切换到内核态，但是不能控制在内核态中执行的指令。在CPU中的实现称之为陷阱指令(Trap Instruction)。
- 工作流程如下:
- 1. 用户态程序将一些数据值放在寄存器中，或者使用参数创建一个堆栈（stack frame），以此表明需要操作系统提供的服务；
- 2. 用户态程序执行陷阱指令；
- 3. CPU切换到内核态，并跳到位于内存指定位置的指令；
- *这些指令是操作系统的一部分，他们具有内存保护，不可被用户态程序访问。这些指令称之为陷阱（trap）或者系统调用处理器（system call handler），他们会读取程序放入内存的数据参数，并执行程序请求的服务。
- 4. 系统调用完成后, 操作系统会重置CPU为用户态并返回系统调用的结果。
- 用户态和内核态在什么时候进行切换?
- 这3种方式是系统在运行时由用户态转到内核态的最主要方式，其中系统调用可以认为是用户进程主动发起的，异常和外围设备中断则是被动的。
- 1. 系统调用
- 用户态进程主动要求切换到内核态的一种方式，用户态进程通过系统调用申请使用操作系统提供的服务程序完成工作，比如fork()实际上就是执行了一个创建新进程的系统调用。而系统调用的机制其核心还是使用了操作系统为用户特别开放的一个中断来实现，例如Linux的int 80h中断。
- 2. 异常
- 当CPU在执行运行在用户态下的程序时，发生了某些事先不可知的异常，这时会触发由当前运行进程切换到处理此异常的内核相关程序中，也就转到了内核态，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，这时CPU会暂停执行下一条即将要执行的指令转而去执行与中断信号对应的处理程序，如果先前执行的指令是用户态下的程序，那么这个转换的过程自然也就发生了由用户态到内核态的切换。比如硬盘读写操作完成，系统会切换到硬盘读写的中断处理程序中执行后续操作等。
- 32位操作系统与64位操作系统，最大的区别是什么?
- 寻址能力不同（寻址能力指的是CPU所能访问的内存的地址范围的大小，也就是说是内存的存储单元的大小），是更大64位处理器的优势体现在系统对内存的控制上。由于地址使用的是特殊的整数，因此一个ALU(算术逻辑运算器)和寄存器可以处理更大的整数，也就的地址。比如，Windows Vista x64Edition支持多达128 GB的内存和多达16 TB的虚拟内存，而32位CPU和操作系统最大只可支持4G内存。
- 运算速度不同，64位CPUGPRs(General-Purpose Registers，通用寄存器)的数据宽度为64位，64位指令集可以运行64位数据指令，也就是说处理器一次可提取64位数据(只要两个指令，一次提取8个字节的数据)，比32位(需要四个指令,一次提取4个字节的数据)提高了一倍，理论上性能会相应提升1倍。