第1部分 操作系统概论名词解释

脱机输入 / 输出

具体的输入 / 输出不需要在主计算机上进行的方式也称“脱机输入 / 输出”

批处理

作业是由操作系统成批地进行处理，操作系统能自动地从输入池读入下一个作业，并予以运行和输出，如此直到整批作业全部处理完毕。

SPOOLING

由操作系统将磁盘模拟为输入 / 输出设备的处理方式称为SPOOLING（Simultaneous Peripheral Operating On Line），即“并行的外部设备操作联机”，也称“假脱机”。SPOOLING系统是以磁盘为几乎无限巨大的缓冲区来解决低速的I/O设备与高速的CPU之间的速度匹配问题。

分时系统

为了降低交互式系统的等待时间和运行时间的比率，系统通过多台终端同时向很多用户提供运行环境，这种分时系统就能以合理的成本向用户提供交互式使用计算机的方便。

多路性

一台主机可连接多台终端，多个终端用户可以同时使用计算机，共享系统的硬软件资源。

交互性

用户能与系统进行对话。在一个多步骤作业的运行过程中，用户能通过键盘等设备输入数据或命令，系统获得用户的输入后做出响应，显示执行的状况或结果。

实时操作系统

是一种能在限定的时间内对输入进行快速处理并做出响应的计算机处理系统

多处理机系统

一个计算机系统中可具有多个CPU或处理机。一般用微处理器构成阵列系统，其运算速度可以达到上万亿次，

作业

请求计算机完成的一个完整的处理任务称为作业，它可以包括几个程序的相继执行。一个复杂的作业可由多个作业步组成，如编译、运行、打印一个程序的全部工作是一个作业，其中相对独立的每一部分称为作业步。

进程（不支持线程的进程）

程序在一个数据集合上的运行活动，它是系统进行资源分配和调度的一个可并发执行的独立单位。

并发

并发是指在某一时间间隔内计算机系统内存在着多个程序活动。并发是从宏观上（这种“宏观”也许不到一秒的时间）看多个程序的运行活动，这些程序在串行地、交错地运行，由操作系统负责这些程序之间的运行切换，人们从外部宏观上观察，有多个程序都在系统中运行。

虚拟

例如操作系统将一台互斥共享设备虚拟成同时共享设备。

共享

共享是指多个用户或程序共享系统的软、硬件资源。

不确定性

不确定性指的是使用同样一个数据集的同一个程序在同样的计算机环境下运行，每次执行的顺序和所需的时间都不相同。操作系统的不确定性不是指程序执行结果的不确定

第2部分 存储管理名词解释

符号名地址

由定义在源程序变量标识符号决定的数据存放地址。

虚拟地址（相对地址、程序地址、逻辑地址）

源程序经汇编或编译后得到的是目标代码程序，由于编译程序无法确定目标代码在执行时所驻留的实际内存地址，故一般总是从零号单元开始为其编址，并顺序分配所有的符号名所对应的地址单元。由于目标代码中所有的地址值都相对于以“0”为起始的地址，而不是真实的内存地址，故称这类地址为相对地址、程序地址、逻辑地址或虚拟地址。

物理地址

指令中指定的直接内存地址

地址重定位

当装入程序将可执行代码装入内存时，程序的逻辑地址与程序在内存的物理地址一般是不相同的，必须通过地址转换将逻辑地址转换成内存地址，这个过程称为地址重定位。

静态重定位

源程序经编译和连接后生成目标代码中的地址是以0为起始地址的相对地址。当需要执行时，由装入程序运行重定位程序模块，根据作业在本次分配到的内存起始地址，将可执行目标代码装到指定内存地址中，并修改所有有关地址部分的值。修改的方式是对每一个逻辑地址的值加上内存区首地址（或称基地址）值。

动态重定位

将程序在装入内存时，不必修改程序的逻辑地址值，程序执行期间在访问内存之前，再实时地将逻辑地址变换成物理地址。动态重定位要靠硬件地址变换机构实现。

单一连续区存储管理

操作系统管理一块单一的用户内存区，一个作业在运行要独占整个用户区。

固定分区管理

在系统初始化时就把存储空间划分成若干个分区（这些分区的大小可以不同），以支持不同的作业对内存大小需求的不同。

可变分区存储管理

可变分区存储管理法是等到作业运行需要内存时向系统申请时，从若干空闲的内存分区区按要求选择并中“挖”一块出来，其大小等于作业所需内存大小，

首次适应法

采用首次适应法为作业分配大小为size的内存空间时，总是从表的起始端的低地址部分开始查找，当第一次找到大于或等于申请大小的空闲区时，就按所需大小分配给作业。如果分配后原空闲区还有剩余空间，就修改原存储区表项。

循环首次适应法

循环首次适应法分配时总是从起始查找指针所指的表项开始查找，第一次找到满足要求的空闲区时，就分配所需大小的空闲区，修改表项，并调整起始查找指针，使其指向队列中被分配的后面的那块空闲区。下次分配时就从新指向的那块空闲区开始查找。

最佳适应算法

在所有大于或等于要求分配长度的空闲分区中挑选一个最小的分区，即该分区对所要求分配的大小来说，是最适合的。

最差适应算法

最差适应法所分割的空闲存储区是所有空闲分区中的最大的一块。

覆盖

将一个大程序按程序的逻辑结构划分成若干个程序（或数据）段，并将不会同时执行，从而就不必同时装入内存的程序段分在一组内，该组称为覆盖段。这个覆盖段可分配到同一个称为覆盖区的存储区域。

交换

任一时刻主存中只保留一个完整的用户作业。当该作业的时间片用完或因等待某一事件而不能继续运行时，系统就挑选下一个作业进入主存运行。为了减少在主存和辅存间传输的数据量，可以只将原作业的一部分保存到辅存中去，只要释放的主存空间刚好够装入下一个运行作业就行。在以后的适当时间，作业移出的部分可装入到原来的存储区中继续运行下去。这种技术称之为交换技术，也叫“滚进滚出”。

虚拟存储器

在主存中可只装入最近经常要访问的某些区域的指令和数据，剩余部分就暂时不必装入，等到以后要访问到它们时再调入内存。如果主存较紧张，必要时可将已不大访问的信息调出内存，再执行调入操作。由于作业的指令和数据可以存放在外存中，用户的程序就不受实际内存大小的限制，好像计算机系统向用户系统提供了容量极大的“主存”，而这个大容量的“主存”是靠存储管理的软件和硬件通过大容量的辅存作为后援存储器扩充而获得的，是程序设计员感觉到的，而实际上并不存在的存储器，故称虚拟存储器。

页式存储管理

页式存储管理的基本思想是把作业的虚拟地址空间划分成若干长度相等的页（page），也称虚页，每一个作业的虚页都从0开始编号。主存也划分成若干与虚页长度相等的页架（frame），也称页框或实页，主存的页架也从0开始编号。程序装入时，每一个虚页装到主存中的一个页架中，这些页架可以是不连续的。

页表

每一个作业的虚页号到内存的页架号之间的映射关系的表。

联想寄存器

是一种按内容进行并行查找的一组快速寄存器。当用作为页面快表时，在其输入端有一个输入值页号p时，在联想寄存器中存放页号为p的那一项就立即选中，并输出其变换值页架号b。由于访问联想寄存器比访问主存快得多，故极大地提高了地址变换速度。

快表

很多页式系统都配有一组快速寄存器，用来存放当前运行作业的页表表项，以加速地址变换过程，这种页表称之为快表。快表由CPU中的高速cache或联想寄存器构成。

请求分页

其基本思想是对于每一个运行作业，只装入当前运行需要的一部分页面集合。当作业运行时需要访问其他不在主存中的虚页时，硬件产生“缺页中断”，如主存资源紧张，可在原先装入主存的页面中选择一个或多个页，将其换出到辅存中，再把所需的页调入主存。请求式分页系统将主存和辅存这两级存储器融合成逻辑上统一的整体，故在这种系统中能运行比可用主存更大的作业或在相同容量的主存中并发运行更多的作业。

工作集

当前运行需要的一部分页面的集合。

页面淘汰

请求分页系统中的程序在运行时，当发现某页的内容未被调入主存，就要通过缺页中断处理程序调入该页。如这时主存中还有空闲的页架，那么只需要分配给调入页即可；但如果此时主存中所有页架都已分配出去，就只能从已占用的页架中挑选出一个页面，释放其所占的内存空间，即将其“淘汰”，以腾出空页架以装入新页。

最优淘汰算法

就是淘汰那些从当前时刻起在页面流中不再出现的页，如没有这类页，则淘汰一个在页面流中最晚出现的页。

先进先出淘汰算法

总是淘汰最早调入主存的页面。

最近最少使用淘汰算法（LRU，Least Recently Used）

比较最近一段时间里对各个页面的访问频率，淘汰访问频率最低的页面。实际上，很多系统都将该算法实现为淘汰“最近一段时间内最久没有访问”

最近未使用淘汰算法（NUR，Not Used Recently）

淘汰最近一段时间内未曾访问过的某一页面。该算法的一个实施不仅能考虑最近未曾访问过的页，还能优先挑选页面数据未曾修改过的页，这样可减少将淘汰页写回辅存的开销。

段式存储管理

用户可以根据逻辑结构将程序分成若干段，每一段的虚拟地址空间各自都从0开始编址，因此整个作业的虚拟地址空间是二维的。类似于页式管理，段式管理要通过一个段表来进行地址变换。

段页式存储管理

段页式存储管理的基本思想是将面向用户的程序地址空间分为段，系统为每一段分配和管理实存时再分页，这样可以保持分段管理系统的便于模块化设计、允许分段动态扩展、动态链接、分段的共享和段地址的保护等诸优点，也便于保持页式存储管理系统提供的大容量的虚拟存储器、没有页外碎片存在、无需紧凑内存、从而更有效地利用主存、且对用户透明的优点。

第3部分 进程管理名词解释

进程

进程（不支持线程的进程）是程序处于一个执行环境中在一个数据集上的运行过程，它是系统进行资源分配和调度的一个可并发执行的独立单位。

进程控制块PCB（Process Control Block）

系统用于查询和控制进程运行的档案，它描述进程的特征，记载进程的历史，决定进程的命运。

执行（Running）状态

进程占用了CPU，正在执行指令的状态。

就绪（Ready）状态

进程拥有除了CPU之外的任何其他的资源和运行条件，只是由于还没有给它分配CPU而处于下一个执行阶段的起跑线上，它已“万事俱备，只欠东风”，因此就绪状态进程在逻辑上是可执行的。在一个系统中可以有多个进程处于就绪状态，通常将它们排在一个（或多个）就绪队列中。

阻塞（Blocked）状态

阻塞状态也称睡眠状态、封锁状态或挂起状态等。某些系统中这些状态有微小的差别。一个进程因某个原因（或事件）暂时无法继续运行下去，因此放弃了CPU，等待影响它运行的因素消除。引起进程阻塞的原因很多，如进程在等待用户输入数据，或等待I/O设备空闲，或等待其他进程发一个同步信号等。一个进程进入了阻塞状态后，系统根据不同的原因将它们排入某一个阻塞队列中。

高级调度

又称长程调度、作业调度或接纳调度等，它决定处于输入池中的哪个后备作业可以调入主系统做好运行的准备，成为一个或一组就绪进程。系统中高级调度的执行频度较低，一个作业只需经过一次高级调度。

中级调度

又称中程调度，它决定处于交换区中的就绪进程中哪一个可以调入内存，以便直接参与对CPU的竞争。在内存资源紧张时，为了将进程调入内存，必须将内存中处于阻塞状态的进程调至交换区，以便为调入进程腾出空间。这相当于使处于内存中的进程和处于盘交换区中的进程交换了位置，故中级调度又称为“对换调度”。中级调度是为了缓解内存资源的紧张状

态，在多道程序范畴内实现进程动态覆盖和进程级的虚拟存储器技术。一个进程在其运行期间可能需要经过多次中级调度。

低级调度

又称短程调度或进程调度。它决定驻在内存中的哪一个就绪进程可以占用CPU，使其获得实实在在的执行权力，故低级调度又可称处理机调度或分派调度。低级调度执行频度很高。

先来先服务（FIFO）调度算法

先来先服务算法是按照进程到达就绪队列的时间次序分配处理机，这是一种不可抢占式的简单算法。

时间片轮转法

进程按到达的时间排在一个先进先出就绪队列中，调度程序每次选择队首的就绪进程，使其占用处理机，并运行一段称为“时间片”的固定时间间隔。在这个时间片内，如运行任务完成或因I/O等原因进入了阻塞状态，该进程就提前退出执行队列，调度程序就使就绪队列中的下一个进程占用处理机，使用一个时间片。当一个进程耗费完了一个时间片而尚未执行完毕，调度程序就强迫它放弃处理机，使其重新排到就绪队列末尾，再等待一个轮转周期。

优先级调度算法

为了能反映出各种进程的重要和紧迫程度，系统赋予每一个进程一个优先数，用优先数表示该进程的优先级。调度程序总是从就绪队列中挑选一个优先级最高的进程，使之占用处理机。优先级调度算法分为两类，一类是静态优先级法，另一类是动态优先级法。

静态优先级法

在一个进程创建时就赋予它一个优先级，在进程运行期间该优先级保持不变。

动态优先级法

能反映进程在运行过程中不同阶段的优先级变化情况。例如，一个总体CPU忙的进程在其I/O阶段就应提高其优先级，一旦在此阶段需要占用CPU，就应当尽快满足要求，以使它能

尽快地启动下一次I/O操作。反之，一个总体I/O繁忙的进程，可能在输入一批数据后的一段时间内，需要大量的计算或数据处理时间，这是它的CPU繁忙阶段，这时该进程的优先级就不必像I/O繁忙阶段那样高。一个运行到某一阶段的进程，需要和用户交互才能正确运行下去，也应当在该阶段提高优先级，以减少用户等待的时间。

线程

一个进程内部可以有一至多个线程，每一个线程具有如下特征：

n 线程的执行状态（运行、就绪等）；

n 当不处于执行状态时保存的线程上下文环境；

n 一个执行栈；

n 存取所属进程内的主存和其他资源，在本进程的范围内与所有线程共享这些资源。

线程带来的关键好处是提高了操作系统的性能。在一个现存的进程中创建一个新的线程的时间远小于创建一个新的进程。研究表明，创建一个新进程的开销是一个线程的10倍。终止一个线程的时间也较小。在同一个进程内部两个线程的切换开销比进程之间的切换开销小得多。这样，一个应用要实现为一组相关的执行单元，那么用一组线程执行而不是用一组分开的进程执行，其效率就要高得多。

第4部分 进程通信名词解释

互斥

两个或两个以上的进程竞争某些同时只能被一个进程使用的资源的情况下，就需要一种互斥机构来协调，控制为这些进程分配资源的次序。

同步

两个或两个以上的进程要协作完成一个任务，它们之间就要互相配合，需要在某些动作之间进行同步，即一个进程的某些动作与协作进程某些动作之间在时序上要有一定的关系。如果

协作进程的某些操作没有完成，那么进程就要在执行路径的某些点上等待这些操作的完成，之后才能继续执行下去。

临界资源

在一段时间内只能允许一个进程访问的资源称为临界资源。

临界段

进程执行的访问临界资源的程序段称为临界段或互斥段。

锁

锁有两个状态：一个是打开状态；另一个是关闭状态。故锁可以用布尔变量表示。在C语言中，锁变量可以定义为char或int类型变量。用对锁变量x的访问，可以控制临界段的执行。

信号灯（信号量）

信号灯定义成具有整型值，并能对其施加以下3种操作的变量，除了这3种操作之外的任何操作都不能测试和处理信号灯的值。

① 初始化操作，信号灯能初始化为非负的值。

② Wait操作，能减小信号灯的值，如结果值为负，执行Wait操作的进程就被封锁。

③ Signal操作，能增加信号灯的值，如果结果值非正，那么原先因执行Wait操作而阻塞的进程被解除阻塞。

生产者和消费者问题

生产者和消费者问题是通过有限的缓冲区（仓库）将一群生产者P1，P2，…，Pk和一群消费者C1，C2，…，Cm联系起来，通过信号灯实现生产者和消费者的同步与互斥。

进程间的消息

类似于用户之间通过电子邮件系统进行通信。消息通信的基本思想是由系统的消息通信机构统一管理一组空闲的消息缓冲区，一个进程要向另一个进程发送消息，先要向系统申请一个缓冲区，填写了消息正文和其他有关消息的特征和控制信息后，通过消息通信机构将该消息送到接收进程的消息队列中。接收进程在一个适当时机从消息队列中移出一个消息，读取所有的信息后，再释放消息缓冲区。

一个消息缓冲区的数据结构中除了要包含消息的正文外，一般还要包含其他有关的控制信息

共享存储区

该机制可以把内存中的一个区域连入多个进程的虚拟地址空间。这样，当一个进程对该地址空间写入数据后，另一个进程就可以从自己所连入的虚拟地址空间直接读出共享存储区中的数据，就如同进程存取自己的私有数据一样方便。

管道

管道是一种信息流缓冲机构，它用于连接发送进程和接收进程，以实现它们之间的数据通信。管道不同于一般的数据缓冲，它以先进先出（FIFO）的方式组织数据的传输。发送进程能把信息以流的形式源源不断地写入管道中，接收进程能以与发送进程写入时的相同顺序读出管道中的信息。

信号

信号是一取值为1~19（MAX_SIGS）的某个整数，可以在进程之间传送，用于通知进程发生了某种异常事件，需要执行事先安排好的动作。每个进程在运行中的某几个时机要主动通过信号机制检查是否有信号到达，如有，便中断正在执行的程序，转入对应的事件处理程序。事件处理完毕，再返回断点继续执行原先的程序。这样的信号处理过程与硬件中断处理很相似，故称之为“软中断”。

死锁

因一组进程为了竞争系统资源或相互间的通信而发生永久性的阻塞。

死锁预防

就是在设计系统时，使该系统能预先排除死锁的可能性。

死锁的避免

在执行时动态地进行审慎的判断，以保证运行不会到达死锁这一点上。由于采用死锁避免的方法要进行动态判断：如果同意为进程分配当前的资源，会不会导致死锁？这样，死锁的避免就需要知道进程以后阶段的资源请求。

死锁的检测

操作系统定期地执行检查算法，以判断是否存在形成死锁的循环等待链。

死锁解除

一旦检测到死锁，就需要采用某种方法解除死锁。

第5部分 设备管理名词解释

中断

当一个正在运行的过程被另外一个过程打断，停止运行过程而转向执行另一过程的活动。在计算机系统中，中断有如下6种基本类型。

（1）I/O中断

（2）时钟中断

（3）系统请求中断

（4）报警中断

（5）程序错误中断

（6）机器错误中断

中断处理程序

也叫做中断服务程序，由操作系统执行相应处理程序，提供所需的服务。

I/O通道

有很强I/O处理功能的智能部件，它可以独立地完成系统处理器交付的I/O操作任务，此时，I/O总线直接与通道相连而不与处理器相连。通道具有自己专门的指令集，即通道指令，用于与连接到总线上的I/O控制器通信，在设备与存储器之间传送数据。系统主处理器只需进行I/O操作的委托，其后的所有I/O操作均由通道自己进行。通道执行来自主处理器的通道程序，完成后只需向系统处理器发出中断，请求结束。可见，这种方式可以形成系统处理器与I/O设备之间的并行操作。

字节多路通道

主要用于连接大量低、中速、以字节作为传输单位的I/O设备。

数据选择通道

主要用以支持高速设备（如磁盘），每次只对一个设备进行数据传输。

成组多路通道

成组多路通道以分时方式同时执行几道通道程序，每条通道指令可以传送一组数据，所以成组多路通道既具有选择通道传输速率较高的优点，也具有字节多路通道分时操作可以同时管理多台设备I/O操作的优点。

设备表

设备表有多个表项，每个表项对应一个设备，表项的内容包括设备标识符、设备地址、设备属性、设备状态、设备驱动程序地址、设备等待队列指针等。

设备开关表

针对各类设备不同的物理特性，系统为它们各自设置了一套子程序，它们包括打开、关闭和启动子程序。系统为每类设备又设置了一数据结构，存放这些程序的入口地址，该数据结构称为设备开关。

寻道时间

把磁头移到相应的道上或柱面上的时间。

等待时间

一旦磁到达指定磁道，必须等待所需要的扇区转到读／写磁头下，这个延迟时间叫做等待时间。

传送时间

信息实际在盘和内存之间进行传送的时间。

第6部分 文件系统名词解释

文件

数据的集合。文件被用户和应用程序当做单个实体对待，可以用名字来引用，并可以产生和删除。存取控制通常施加到文件一级。

目录

目录含有文件的信息，包括文件的属性、位置和文件主等，其中大部分信息特别是与存储有关的信息是由操作系统管理的。目录本身也是文件，各种文件管理例行程序要通过操作系统存取目录文件。从用户的观点看，目录提供了为用户和应用程序所知的文件名到文件实体本身的映射。因此，每一个文件目录项包含文件名。

文件的权限

文件系统要提供能控制特定文件存取的操作。典型的是能授权用户或用户组有存取文件的读、写和执行权限。

文件的系统调用

文件系统向用户提供的有关使用文件的系统功能，主要有：创建和取消文件、打开和关闭文件、对文件进行读 / 写操作和设置文件的读写位置、改变工作目录、 改变文件的权限和和文件控制。

文件的标准子例程

文件的标准I/O库主要通过用户态空间的自动缓冲机构以及数据类型转化和格式化的I/O，向程序员提供了效率高、功能强和可移植的文件访问或字符串处理功能。在UNIX的标准I/O库中，通过一个FILE类型结构建立与打开文件的联系。这种组织形式称为流（stream）。

索引节点

每一个文件有一组控制信息，其中包括文件名、文件主、文件大小、访问权限、存取时间以及文件的数据存放在哪些磁盘块中等信息。UNIX为了实施文件的共享和提高目录的检索速度，只将文件名从文件控制块信息中抽出来，其余所有的控制信息构成了文件的索引节点（Index Node），简称I节点。

文件索引结构

记录了文件中所有盘块的地址，即是文件的逻辑块号与物理块号的映射。

打开文件结构

为了提高打开文件后对文件的访问速度，当读取该文件的控制节点后，应当在对整个文件的访问期间内，在内存中保存该节点的副本，并能通过一个方便快速的途径存取它，这就要建立打开文件管理机构。在Unix系统中打开文件的管理机构包括三部分，它们是内存索引节点、系统打开文件控制块和进程打开文件表。

文件系统超级块

超级块中存放文件系统核心管理数据，它包含以下几部分内容：文件系统各部分所占的盘块总数、超级块直接管理的空闲I节点数和空闲I节点索引表、超级块直接管理的空闲盘块数和

空闲盘块索引表、文件系统的类型、时间和状态信息等。

管道文件

连接前后两个进程的打开文件，前一个进程可以向该文件中顺序地写入数据，后一个进程可以从该文件中顺序地读出数据。数据的写入和读出以先进先出的方式进行，并由系统自动地处理两个进程间的调度、同步和数据缓冲，这类文件就称为管道（pipe）文件，简称管道，或称FIFO。

无名管道

无名，不能通过类似文件名来访问，只能在与创建进程的同一进程族内传递数据的按先进先出方法组织的打开文件。

有名管道

像普通文件一样有其目录项，在文件系统中能长久地存在，任何有访问权限的用户都可以通过路径名来打开它，进而存取其中数据，因此无关的进程就可以通过有名管道进行通信。

虚拟文件系统

一个虚拟文件系统，不管位于什么具体的设备上，必须保持同样的方式和接口来进行操作。具体地说，当一个进程调用文件系统例程时，内核调用VFS函数（这个函数是和具体结构无关的），并将这个调用传递给物理文件系统中的相应函数,该函数与具体的物理结构有关。

原文：https://blog.csdn.net/SpadgerZ/article/details/52802242