第一章 操作系统概论

1. 操作系统的定义

OS 是一组控制和管理计算机硬件和软件资源，合理地对各类作业进行调度，方便用户使用计算机的程序集合。即为用户程序提供服务，是用户与硬件系统之间的接口。

2. 操作系统的作用

OS 是计算机系统的核心，负责管理整个计算机系统的软硬件资源，制定各种资源的分配策略，调度系统中运行的用户程序，协调用户对资源的需求，从而使真个计算机系统高效有序的工作

3. 操作系统的常见的分类

①批处理操作系统：单道批处理（自动、顺序 and 单道性），多道批处理（宏观上并行，微观上串行，资源利用率高系统吞吐量大；用户响应时间长无人机交互能力）

②分时操作系统：时间片轮转；同时、交互、独立 and 及时性

③实时操作系统：在事先定义好的时间内对事件进行响应处理；及时 and 可靠性

④网络 OS 分布式 OS and 嵌入式 OS

4、多道程序技术的定义，引入多道程序技术的原因和目的

多道程序技术允许多个程序同时进入内存并且允许它们在 CPU 中交替运行。当一道程序因 I/O 请求而暂停运行时，CPU 立即转去运行其他道程序，使资源得到充分利用，工作效率成倍提高。

引入多道程序技术为了进一步提高资源利用率和系统的吞吐量，充分发挥计算机系统部件的并行性

5. 操作系统与硬件、其他系统软件以及用户之间的关系

操作系统是覆盖在硬件上的第一层软件，它直接管理计算机的硬件和软件资源，并向用户提供良好的界面；

操作系统是一种特殊的系统软件，其他系统软件运行在操作系统的基础之上，可获得操作系统提供的大量服务，也就是说操作系统是其他系统软件与硬件之间的接口；

而一般用户使用计算机除了需要操作系统支持外，还需要用到大量的其他系统软件和应用软件，以使其工作更加方便和高效

6、多道程序技术的定义及在 OS 中引入该技术所带来了的好处

多道程序技术即是指在内存中存放多道作业,运行结束或出错,自动调度内存中另一道作业运行。多道程序主要优点如下:
(1)资源利用率高。由于内存中装入了多道程序,使它们共享资源,保持系统资源处于忙碌状态,从而使各种资源得以充分利用。
(2)系统吞吐量大。由于 CPU 和其它系统资源保持“忙碌”状态,而且仅当作业完成或运行不下去时才切换,系统开销小,所以吞吐量大。

7、推动批处理系统和分时系统形成和发展的主要动力

（1）推动批处理系统形成和发展的主要动力：不断提高系统资源利用率和提高系统吞吐量：脱机输入/输出技术的应用和作业的自动过渡大大地提供了 I/O 的速度及 I/O 设备与 CPU 并行工作的程度，减少了主机 CPU 的空闲时间；多道程序设计技术的应用更进一步提高了 CPU、内存和 I/O 设备的利用率及系统的吞吐量。

（2）推动分时系统形成和发展的主要动力是为了更好地满足用户的需要：CPU 的分时使用缩短了作业的平均周转时间；人机交互能力的提供使用户能方面地直接控制自己的作业；主机的共享使多个用户能够同时使用同一台计算机独立、互不干扰地处理自己的作业。

8、实现分时系统的关键问题

实现分时系统的关键问题是使用户能与自己的作业交互作用即用户在自己的终端上键入一命令以请求系统服务后系统能及时地接收并处理该命令并在用户能够接受的时延内将结果返回给用户。 及时地接收命令和返回输出结果是比较容易做到的一般只要在系统中配置一个多路卡并为每个终端配置一个缓冲区用来暂存用户键入的命令和输出的结果便可以了。

9、时间片的划分原则

时间片即 CPU 分配给各个程序的时间，即该进程允许运行的时间，使各个程序从表面上看是同时进行的。如果在时间片结束时进程还在运行，则 CPU 将被剥夺并分配给另一个进程。如果进程在时间片结束前阻塞或结束，则 CPU 当即进行切换。而不会造成 CPU 资源浪费。

10、从交互性，及时性以及可靠性三个方面，比较分时系统与实时系统。

①及时性。实时信息处理系统对实时性的要求与分时系统类似，都是以人所能接受的等待时间来确定的;而实时控制系统的及时性，则是以控制对象所要求的开始截止时间或完成截止时间来确定的，一般为秒级到毫秒级，甚至有的要低于 100 微秒。

②交互性。实时信息处理系统虽然也具有交互性，但在这里用户与系统的交互仅限于访问系统中某些特定的专用服务程序。它不像分时系统那样能向终端用户提供数据处理和资源共享等服务。

③可靠性。分时系统虽然也要求系统可靠，但相比之下，实时系统则要求系统具有高度的可靠性。因为任何差错都可能带来巨大的经济损失，甚至是无法预料的灾难性后果，所以在实

时系统中，往往都采取了多级容错措施来保障系统的安全性及数据的安全性。

11、操作系统的特征

（1）并发

两个或多个事件在同一时间间隔内发生。

（区别与并行的不同：并行性是指操作系统具有同时进行运算或操作的特性，同一时刻进行两种或以上的工作。）

（2）共享

①互斥共享：一段时间内只允许一个进程使用，只有当前作业结束释放后，才允许其他作业使用

②同时共享：系统资源允许一个时间段由多个进程同时（宏观）访问，微观上是交替访问

（3）虚拟

通过某种技术将一个物理实体变为若干个逻辑上对应的功能。

（4）异步

多道程序环境多个程序并发执行，但由于资源有限，进程的执行并不是一贯到底，而是以不可预知的速度向前推进。

12. 微内核结构的优点

微内核结构是以客户/服务器体系结构为基础，采用面向对象技术的结构。

①灵活性。微内核短小精干，仅提供最基本最必要的服务。

②开放性。操作系统除内核以外的功能都可用服务器的形式建立在内核之上，系统的开发者基于这种结构框架，可以方便地设计、开发、集成自己的新系统。

③可扩充性。采用微内核的操作系统，对于实现、安装、调试一个系统是很容易的。用户可以重写他们已有的不满意的服务

第二章进程描述与控制/线程

1. 进程的基本状态

①创建态。进程刚被创建时的状态，尚未进入就绪态。

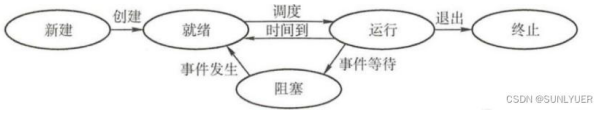
②就绪态。进程获得了除处理机外的所有资源，只需获得处理机就可以运行，如果多个进程处于就绪态，则它们放在就绪队列中。

③运行态。进程获得处理机运行。在单处理机同一时刻只能有一个进程在处理机上运行

④阻塞态。进程正在等待某一资源而暂停运行的状态，如等待某资源或等待输入输出完成，如果有多个阻塞进程，则将他们存入阻塞队列中。

⑤结束态。进程正在从系统中消失，可能正常结束，也可能被动结束。系统需要将该进程置为结束态回收相关资源。

2. 进程状态转换图



①就绪态→运行态：处于就绪态的进程得到处理机的调度，获得处理机资源，便可运行

②运行态→就绪态：非抢占式操作系统中是时间片用完了；抢占式操作系统中是有更高优先级的进程进入，就会抢占处理机，导致原进程重新回到就绪态

③运行态→阻塞态：运行中的进程，需要其他的资源，就会进入阻塞态

④阻塞态→就绪态：之前所需的资源获取到之后，就重新进入就绪态

3. 进程挂起的定义

将处于就绪或阻塞态的进程由内存换到外存

4. 产生挂起的原因

1）终端用户的请求。当终端用户因为在运行期间发现问题而无法执行、需要暂时停止正在运行的进程，若该进程处于就绪状态，挂起后，该进程就处于静止状态，也称为就绪静止。

2）父进程请求。有时父进程根据运行需要，在执行期间需要挂起自己的某个子进程，以便考查和修改该子进程，或者协调各子进程间的活动。

3）负荷调节的需要。当实时操作系统中存在较重的工作负荷时，为了保证实时系统的及时、可靠性，需要把系统中不重要的进程挂起，实现系统的正常执行。

4）操作系统的需要。操作系统有时会根据需要挂起某些进程，完成资源使用情况或运行记录情况的检查。

5. 操作系统中用于进程控制的原语，这些原语的作用

①create 原语：创建新进程

②撤销原语：终止进程

③阻塞原语 block：将进程由运行态转变为阻塞态

④唤醒原语：将进程从等待队列中移出，并置其状态为就绪态

6. 进程的组成结构

进程是一个独立的运行单位，也是操作系统进行资源分配和调度的基本单位，一般包括进程控制块，程序段和数据段三部分

7、进程控制块的定义

PCB 是进程实体地一部分，是进程存在地唯一标志，是操作系统中最重要地记录型数据结构

8. 进程控制块的组成

①进程标识符

②处理及状态

③进程调度信息

④进程控制信息

9. PCB 的作用

PCB 的作用是使一个在多道程序环境下不能独立运行的程序成为一个能独立运行的基本单位，能与其他进程并发执行的进程。

10. 一个经典的 PCB 中应该具备哪几类信息

处理机状态信息；进程调度信息；进程控制信息

11、创建一个新进程的主要步骤

①为新进程分配一个唯一的进程标识号，并申请一个空白的 PCB 。

②为进程分配资源，为新进程的程序和数据及用户栈分配必要的内存空间。若资源不足(如内存空间)，则并不是创建失败，而是处于阻塞态，等待内存资源。

③初始化 PCB，主要包括初始化标志信息、初始化处理机状态信息和初始化处理机控制信息，以及设置进程的优先级等。

④将新进程插入就绪队列，等待被调度运行。

12、进程和程序的区别

（1）进程是一个动态的概念，而程序是一个静态的概念，程序是指令的有序集合，无执行含义，进程则强调执行的过程。

（2）进程具有并行特征（独立性、异步性），程序则没有。

（3）不同的进程可以包含同一个程序，同一程序在执行中也可以产生多个进程

13、进程的执行模式

用户模式；内核模式

14、从调度性，并发性，拥有资源，独立性，系统开销以及对多处理机的支持等方面比较对进程和线程

1)调度。在引入线程的操作系统中线程是独立调度的基本单位，进程是拥有资源的基本单位；在同一进程中线慢的切换不会引起进程切换。在不同进程中进行线程切换，如从一个进程内的线程切换到另一个进程中的线程时，会引起进程切换。

2)并发性。在引入线程的操作系统中。不仅进程之间可以并发执行，而且多个线程之间也可以并发执行从而使操作系统具有更好的并发性。提高了系统的吞吐量。

3)拥有资源。不论是传统操作系统还是没有线程的操作系统.进程都是拥有资源的基本单位；而线程不拥有系统资源(只有一点必不可少的资源)，但线程可以访问其隶属进程的系统资源。

4)系统开销。由于创建或撤销进程时，系统都要为之分配或回收资源，如内存空间、IO 设备等，因此操作系统所付出的开销远大于创建或撤销线程时的开销；而线程切换时只需保存和设置少量寄存器内容，开销很小。由于同一进程中的多个线程具有相同的地址空间，致使它们之间的同步和通信的实现，也变得比较容易。且无需操作系统的干预。

15、用户级线程 ULT 的定义

用户级线程是指不依赖于操作系统核心，由应用进程利用线程库提供创建、同步、调度和管理线程的函数来控制的线程。

16. 内核支持线程 KLT 的定义

内核级线程是指依赖于内核，由操作系统内核完成创建和撤销工作的线程。

17. 线程与进程的关系

①一个进程可以有多个线程，但至少有一个线程；而一个线程只能在一个进程的地址空间内活动。

②资源分配给进程，同一个进程的所有线程共享该进程所有资源。

③CPU 分配给线程，即真正在处理器运行的是线程。

④线程在执行过程中需要协作同步，不同进程的线程间要利用消息通信的办法实现同步。

第三章 并发控制——互斥与同步

1、临界资源的使用应遵循的基本原则

对临界资源的访问必须互斥地进行

2、 临界区的定义

进程中访问临界资源的那段代码。

3. 进程在访问临界区时要遵循的原则

1)空闲让进。临界区空闲时，可以允许一个请求进入临界区的进程立即进入临界区。

2)忙则等待。当已有进程进入临界区时，其他试图进入临界区的进程必须等待。

3)有限等待。对请求访问的进程，应保证能在有限时间内进入临界区。

4)让权等待。当进程不能进入临界区时，应立即释放处理器，防止进程忙等待。

4、中断处理过程

①关中断：CPU 响应中断后，首先要保护程序现场状态，在保护的过程中 CPU 不可以响应更高优先级的中断请求。如果响应了更高优先级的程序，那低优先级的程序现场保存不完整，之后处理完中断后就不能正确的回到中断前的状态

②保存断点：为保证中断服务程序执行完毕能正确的返回到原来的程序，必须将程序断点（程序计数器 PC 的值）保存起来

③寻找中断服务程序入口地址：实质是取出中断服务程序的入口地址送入到程序计数器 PC 中入口地址的获取：查询"中断向量表"，找到相应的中断处理程序在内存中的存放位置

进入中断服务程序，开始执行中断服务程序

④保护现场和屏蔽字：进入中断服务程序后的第一件事，就是保护现场和屏蔽字，主要是保存程序状态字 PSWR 和某些通用寄存器的内容

⑤开中断：允许响应更高级中断此时如果有高级中断到来，可以去执行高级中断，因为被中断程序的现场信息已经被保存，响应高级中断不会导致被中断程序在恢复时现场信息不完整

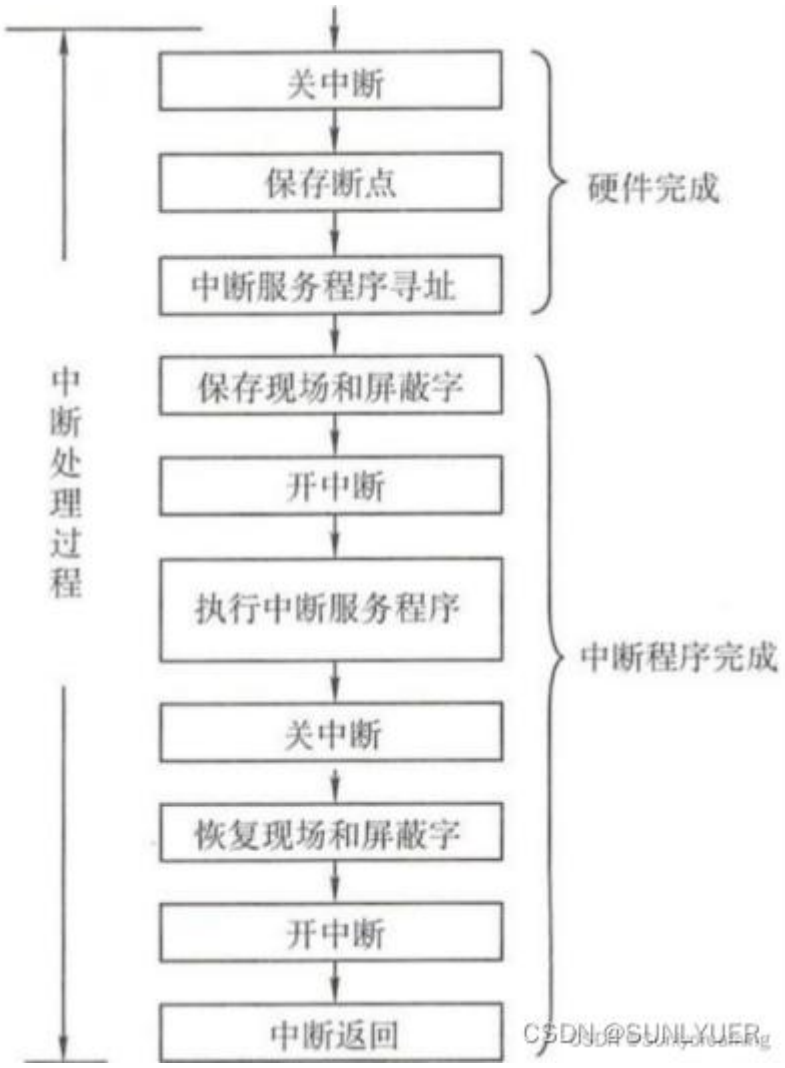
⑥中断服务程序处理：执行中断服务程序，这也是中断的目的所在

⑦关中断：执行完中断服务程序，在程序结束前，要恢复现场，这个操作也不可以被打断

⑧恢复现场：中断服务程序将之前保存的相关寄存器的值恢复到原来的状态

⑨开中断：中断服务程序即将结束，重新打开中断，允许响应其他中断

⑩返回到断点继续执行：中断服务程序的最后一条指令通常是一条中断返回指令，使其返回到程序断点处，以便原程序继续执行



5. 如何保证诸进程互斥地访问临界资源？

空闲让进；忙则等待；有限等待；忙则等待

6. 如何用硬件方法解决互斥（包含两种方法）

①中断屏蔽：CPU 的进程切换是通过中断完成的，所以屏蔽中断可以可以让当前正在运行的进程顺利执行，防止其他进程进入临界区，进而保证互斥的实现，典型步骤为：关中断→临界区→开中断。

该方法简单高效，但是只适用于操作系统的内核进程，不适用于用户进程；处理机的执行效率下降；若关中断后没有开中断系统可能会终止。

②硬件指令：TestAndSet 指令：该指令是原子操作（执行过程不允许被中断），用于将读出的指定标志设为真；Swap 指令：用于交换两个字节的内容

7. 什么是“忙-等”，它的缺点

“忙等”指：“不让权”的等待，即进程因某事件的发生而无法继续执行时它仍占有 CPU 并通过不断地执行循环测试指令来等待该事件的完成。

“忙等”的主要缺点是浪费 CPU 的时间另外它还可能引起预料不到的后果。

例如考虑某个采取高优先权优先调度原则的系统目前有两个进程 A 和 B 共享某个临界资源 A 的优先权较高 B 的优先权较低且 B 已处于临界区内而 A 欲进入自己的临界区则 A、B 都不可能继续向前推进陷入“死等”状态。

8. 如何利用信号量解决互斥问题«

互斥是不同进程对同一信号量执行 PV 操作；解决互斥，即要求单位时间内只允许一个进程访问临界资源

进程 P1,P2 并发执行，二者有各自的临界区，但系统要求每次只能又一个进程进入自己的临界区，因此设置信号量 S 初值为 1（可用资源数为 1），把临界区置于 P,V 操作之间，实现两个进程对临界资源的互斥访问：

```
semaphore S=1;     //初始化信号量
P1(){
     ...
     P(S);         //申请访问临界资源，并加锁
     P1的临界区;
     V(S);         //访问结束，释放资源，解锁
}
P2(){
     ...
     P(S);         //申请访问临界资源，并加锁
     P2的临界区;
     V(S);         //访问结束，释放资源，解锁
}
```


若临界区中没有进程，任意进程进入就要执行 P 操作，加锁 S--为 0，然后进入；若临界区中有进程，此时 S 为 0，想要执行 P 操作便会自我阻塞，直到临界区的进程退出，这样就实现了互斥。

9. 信号量的 P，V 操作

P 操作（wait(s)，value--），表示进程申请一个该类资源；

V 操作（signal(s)，value++），表示进程释放一个资源；

10. 管程的组成

①管程的名称

②局部于管程内部的共享结构数据说明

③对该数据结构进行操作的一组过程(或函数)

④对局部于管程内部的共享数据设置初始值的语句

11. 管程的特性

①管程内的局部变量只能被局限于管程内的过程所访问;反之亦然，即局部于管程内的过程只能访问管程内的变量。

②任何进程只能通过调用管程提供的过程入口进入管程。对于上例中外部进程只能通过 take_away()过程来申请一个资源。

③任一时刻，最多只能有一个进程在管程中执行，从而实现进程互斥。管程是一种编程语言的构件，它的实现需要编译器的支持

12、消息传递原语

Send 原语 and Receive 原语：成对出现，一般组合为阻塞发送，阻塞接收；无阻塞发送，阻塞接收；无阻塞发送，无阻塞接收

第四章 死锁处理
1、死锁的定义

死锁是指多个进程因竞争资源而造成的一种僵局（互相等待），若无外力作用，这些进程都无法继续向前推进。

2. 死锁的四个必要条件

①互斥：同一时间间隔内只有一个进程占有资源

②不可剥夺：进程获得的资源只能由自己主动释放，不能被其他进程强行占有

③请求和保持：已经拥有资源的进程提出新的资源请求，但该资源已被其他进程占有，则该进程被阻塞，但以获得的资源依旧保持不放

④循环等待：存在一种进程资源的循环等待链，链中每一个进程已获得的资源同时被下一个进程所请求

3. 产生死锁的根本原因

①竞争系统资源：由于不可剥夺资源无法满足多个进程运行的需要，因此这些进

程对不可剥夺资源竞争陷入僵局，进而产生死锁。

②进程推进非法：当两个进程 P1,P2 分别占有资源 1,2，若 P1 再申请资源 2，P2 再申请资源 1，则两个进程都会因为缺少资源陷入僵局，即进程运行过程中的请求和释放资源顺序的不当也会导致死锁。

4. 解决死锁的三种方法

①死锁预防：设置条件去破坏死锁产生的四个条件中的一个或几个

②死锁避免：在资源动态分配的过程中，采取某种策略避免系统进入不安全状态

③死锁的检测解除：进程运行过程中，运行死锁的发生，通过检测机构检测死锁，采取剥夺资源和撤销进程等措施将系统从死锁中解脱出来

5. 静态资源分配策略

破坏请求与保持条件，采用静态分配，在进程处于就绪状态就一次性申请进程运行时所需要的全部资源，若全部资源申请完成立即运行进程，且所有资源运行期间只归当前进程所有，不再被其他进程请求。

6、预防死锁的方法

采用静态资源分配策略;

采用按需资源分配策略

7、什么是安全状态

安全状态是在某一时刻，系统能够按某种顺序 or 序列（eg：P1，P2，..Pn)来为每个进程分配其所需的资源，直至每个进程都能获得最大资源的需求，保证所有进程都顺序完成，此时称该序列为安全序列。只要能找出一个安全序列，系统就为安全状态；如果系统中一个安全序列都不存在，则该系统处于不安全状态。

8. 如何判断系统是否处于安全状态«

例题：T0 时刻是否为安全状态?若是，请给出安全序列

T_0时刻的资源分配表

进程 \ 资源情况	Max A B C	Allocation A B C	Available A B C
P_0	7 5 3	0 1 0	
P_1	3 2 2	2 0 0 (3 0 2)	3 3 2 (2 3 0)
P_2	9 0 2	3 0 2	
P_3	2 2 2	2 1 1	
P_4	4 3 3	0 0 2	

[分析]:

1、①Need = Max − Allocation = $\begin{bmatrix} 7 & 4 & 3 \\ 1 & 2 & 2 \\ 6 & 0 & 0 \\ 0 & 1 & 1 \\ 4 & 3 & 1 \end{bmatrix}$ $\begin{matrix} P_0 \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{matrix}$ work = Available = (3, 3, 2)

P_1 P_3

② In Need, work > (1, 2, 2) and (0, 1, 1). So we can choose P_1 or P_3 here, I choose P_1

③ P_1 finished, free all resources, update work = (3, 3, 2) + Allocation[P_1] = (5, 3, 2)

use this work repeat ②

finally get (P_1, P_3, P_4, P_2, P_0).

进程	Work			Need			Allocation			Work+Allocation			Finish
	A	B	C	A	B	C	A	B	C	A	B	C	
P1	3	3	2	1	2	2	2	0	0	5	3	2	True
P3	5	3	2	0	1	1	2	1	1	7	4	3	True
P4	7	4	3	4	3	1	0	0	2	7	4	5	True
P2	7	4	5	6	0	0	3	0	2	10	4	7	True
P0	10	4	7	7	4	3	0	1	0	10	5	7	True

9. 银行家算法的基本思想

核心思想：进程运行之前先声明对各种资源的最大需求量，当进程在执行中继续申请资源时，先测试该进程已占用的资源数与本次申请的资源数之和是否超过该进程声明的最大需求量。若超过则拒绝分配资源、若未超过则再测试系统现存的资源能否满足该进程尚需的最大资源量，若能满足则按当前的申请量分配资源，否则也要推迟分配。

10. 资源分配图及其化简

在资源分配图中，找出既不阻塞又不孤点的进程（即找出一条有向边与它相连，且该有向边对应资源的申请数量小于等于系统中已有的空闲资源数量），若所有连接该进程的边均满足条件，则该进程能继续运行直至完成，然后释放它占有的所有资源。消去它所有的请求边和分配边，使之成为孤立的结点。判断某种资源是否有空间，应用它的资源数量减去它在资源分配图中的出度。根据上述方法进行一系列简化后，若能消去图中所有的边，则称该图是可完全简化的。

第五章　内存管理

1、虚拟存储器的定义

虚拟存储器就是仅把作业的一部分装入内存便可运行的存储器系统。具体说就是指具有请求调入功能和置换功能，能从逻辑上对内存容量进行扩充的一种存储系统

2. 实现虚拟存储器的关键技术

虚拟存储器的的实现,都是建立在离散分配的存储管理方式的基础上,如请求分页、请求分段和请求段页式三种存储管理方式；同时需要动态重定位，即重定位技术以及重定位寄存器。

3. 交换技术

交换技术就是把暂时不用的某个程序及数据的一部分或全部从内存中移到外存中去,腾出必要的内存空间;或把指定的程序或数据从外存中读到相应的内存中,并将控制权转给它

换出—把处于等待（阻塞）状态的进程从内存移到外存，将内存空间腾出来

换入—将准备好竞争 CPU 运行的进程从外存转移到内存

4. 逻辑地址、物理地址的定义

①逻辑地址:源代码经过编译后,目标程序中所用的地址就是逻辑地址.每个目标模块都从 0 号单元开始编址,其对应的地址范围即逻辑地址空间;不同的进程可以拥有相同的逻辑地址,可以映射到主存的不同位置

②物理地址:内存中物理单元的集合,地址转换的最终地址,进程执行指令和访问数据最后都要通过物理地址从主存中存取

5. 重定位的定义

将地址空间中使用的逻辑地址转换为内存空间中的物理地址的地址转换，也叫地址映射

根据地址转换的时间和手段，把重定位分为静态重定位和动态重定位两种

6. 三种分区法

①固定分区法：它将用户内存划分为若干大小固定的区域,每个区域只装入一道作业

②动态分区法：动态分区分配不预先划分内存，当进程装入内存时，根据进程的大小动态建立分区，使分区的大小正好适合进程的需要

③可重定位分区法：

7. 什么是拼凑

为了使分散、较小的空闲区得到合理的使用，把所有的碎片合并为一个连续区，也就是移动某些已分配区域的内容，使所有作业的分区连在一起，把空闲区留在另一端

8. 首次适应算法、最佳适应算法、最坏适应算法中存储空间的回收与分配

①首次适应：空闲分区以地址递增的次序链接。当一个作业到达时，从该表中顺序检索，找到大小能满足要求的第一个空闲分区。余下的空用分区仍然保留。若从头到尾都不存在符合条件的分区，则分配失败。

②最佳适应：空闲分区按容量递增的方式形成分区链，当作业到达时，从该表中检索出第一个能满足要求的空闲分区分配给它。该方法碎片最小，如果剩余空闲分区太小，则将整个分区全部分配给它。

③最坏适应：空闲分区以容量递减的次序链接。当一个作业到达时，从该表中检索第一个能满足要求的空闲分区，即挑选出最大的分区，该方法碎片最大，但剩余的空闲分区可再次使用。

9. 页面的定义

将一个进程的逻辑地址空间划分为若干个大小相等的部分，每个部分称为页面

10. 页面的大小如何划分？

页面的大小也应该适中，一般为 2 的整数幂(518B~8KB)，页面太小会使进程的页面过多，导致页表过长占用大量内存，增加硬件地址转换的开销；页面过大会使页内碎片增多，降低内存的利用率。

11. 物理块的定义

将内存空间划分成与页面大小相同的若干个储存块，即为物理块或页框

12. 什么是页表？页表的作用是什么？页表结构？

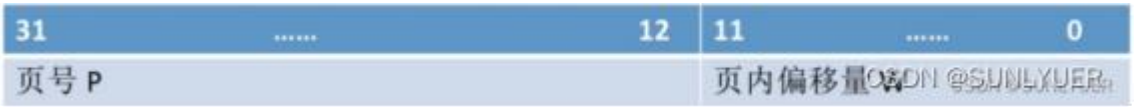
为了知晓每个进程的页面在内存中存放的位置，操作系统为每个进程建立一张页表，一个进程对应一张表，进程的每一页对应一个页表项（页号+块号），记

录进程页面和实际存放的内存块之间的对应关系

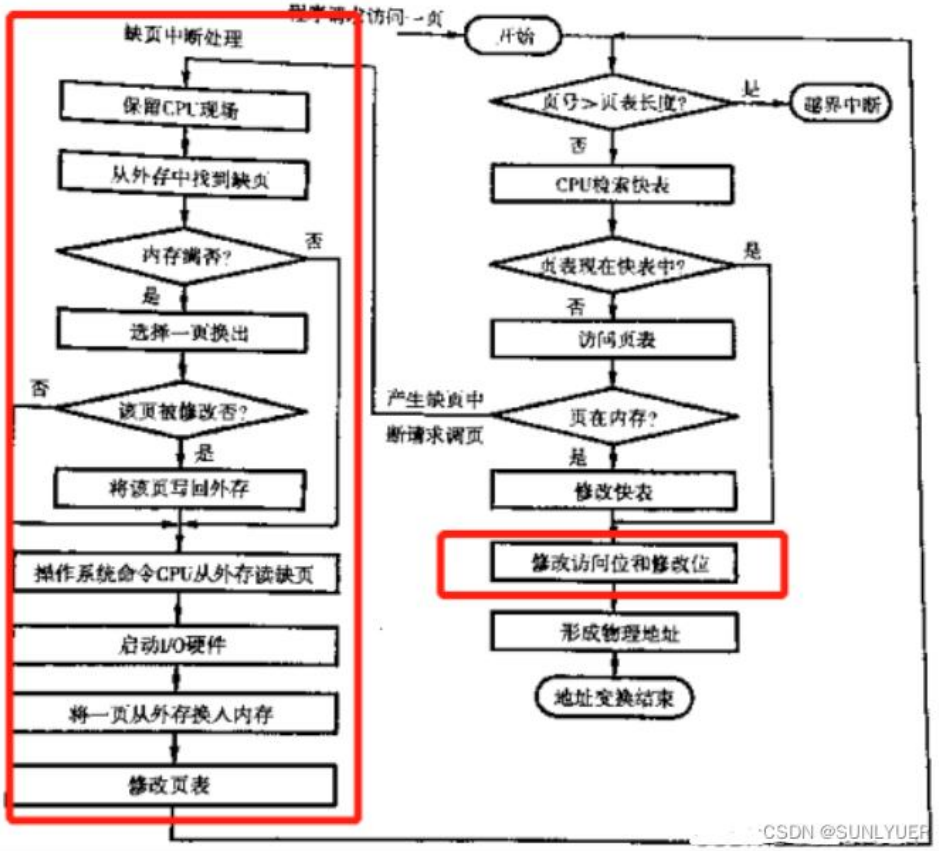
13. 什么是纯分页系统

纯分页系统是指在调度一个作业时，必须把它的所有页一次装入到内存的物理块中，当物理块不足，则该作业必须等待，直到有足够的物理块为止，这时系统再调度另外的作业

14. 分页存储管理系统中的逻辑地址结构



15. 请求分页系统中的地址转换过程



16. 什么是分段存储管理

它将进程的地址空间按照自身的逻辑关系划分为若干个段，每个段可以定义一组相对完整的逻辑性息，每个段从 0 开始编址（段名可供程序员编程）；内存以段为单位进行分配，每个段在内存中占用连续的空间，但可以不相邻（离散分配）

17. 段的定义

段是一组逻辑信息的集合。每个段都有自己的名字和长度从 0 开始编址，常用一个段号来代替段名

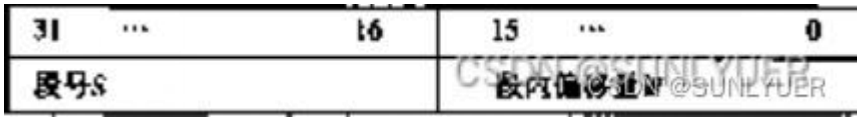
18. 段的大小如何划分

段的大小由相应的逻辑信息组的长度决定，所以各段的长度可以不同

19. 段表结构

每个进程都有一张逻辑空间与内存空间映射的段表，每个段表项对应进程的一段，并记录了段的长度和段在内存中的起始位置

20.分段式系统中的逻辑地址结构



21. 分页和分段存储管理的区别

1)页是信息的物理单位，分页是由于系统管理的需要而不是用户的需要，为了实现离散分配，提高内存的利用率；段则是信息的逻辑单位，分段的主要目的是更好满足用户需求，是对用户可见的。

2)页的大小固定由系统决定；而段的长度却不固定，取决于用户所编写的程序。

3)分页的作业地址空间是一维的，即单一的线性地址空间；而分段的作业地址空间是二维的，要标识一个地址出了给出段内地址外还需给出段号

4)在一个进程中，段表只有一个，而页表可能有多个。

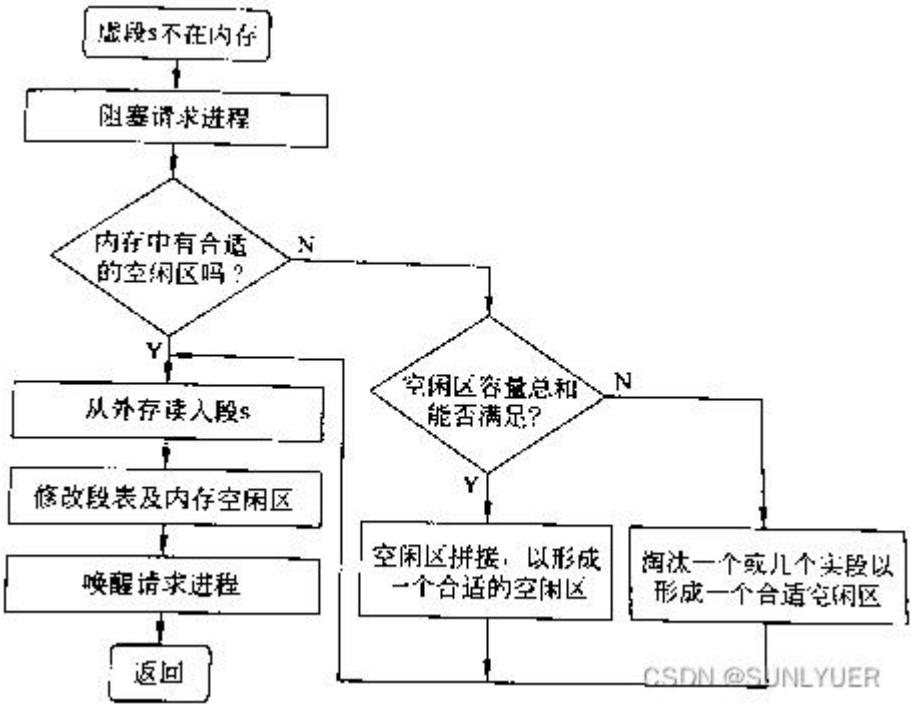
5)分段比分页更容易实现信息的共享和保护。

6)单级页表和分段访问一个逻辑地址均需要两次访存。

23、缺段中断如何处理

在请求分段系统中,每当进程要访问的段还没有调人内存时，就由缺段中断机构产生-一个缺段中断信号,然后,操作系统将通过缺段中断处理程序把所需图

5.20 请求分段式的地址转换过程的段调人内存。与缺页中断机构一样，缺段中断机构在一条指令的执行期间产生和处理中断,并且在一条指令执行期间可能产生多次缺段中断。与页不同,段是不定长的，故对缺段中断的处理将比缺页中断的处理复杂得多。



24. 什么是段页式存储管理

段页式存储管理是页式和段式管理方式的结合，按程序分段，段内再分页。也就是用户程序被逻辑划分为若干段，每段又分成若干页面，内存划分成对应大小的块

25. 段页式存储管理中的段表、页表结构

进程中每个段对应一个段表项,每个段表项由段号、页表长度、页表存放块号(页表起始地址)组成。每个段表项长度相等，段号是隐含的。

每个页面对应一个页表项，每个页表项由页号、页面存放的内存块号组成。每个页表项长度相等，页号是隐含的。

26. 页面置换算法（要求掌握算法的英文缩写方式）

①最佳置换 OPT：选择淘汰以后永不使用的页面，or 在最长时间内不再被访问的页面，这样可以保证获得最低的缺页率。但是操作系统无法预知接下来访问的是哪个页面，所以实际上 OPT 是无法实现的

②先进先出 FIFO：每次淘汰最早进入内存的页面（or 再内存停留最久的），根据调入内存的顺序将页面排序，置换时选择头页面即可

③最近最久未使用 LRU：每次淘汰最近最久未被使用的页面，可以用页表项中的访问字段记录该页面从上次访问以来所经历的时间，淘汰时选择现有页面中该值最大的

④时钟置换 CLOCK27.

27. 什么是抖动

对于刚刚换出的页面马上又要换入内存，刚刚换入的页面马上又要换出内存，这种频繁的页面对换行为称为抖动

产生抖动的主要原因是分配给进程的物理块太少，进程频繁访问的页面数高于可用的物理块数，导致进程在对换花的时间超过执行时间，进程便处于抖动状态

28. 缺页中断率的定义

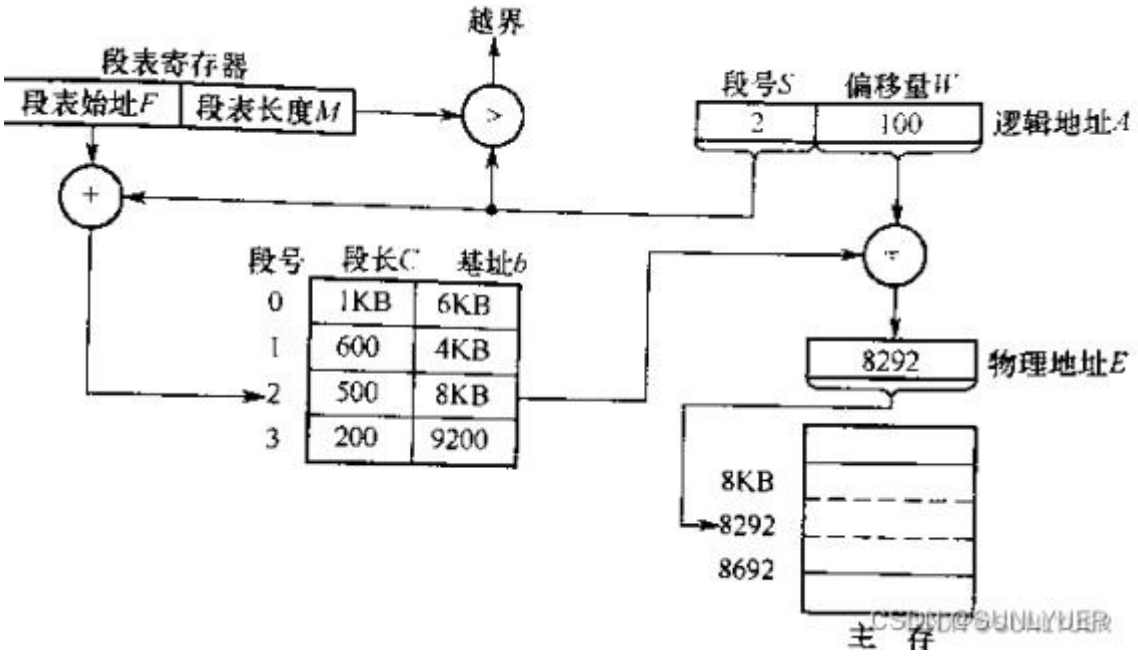
缺页中断次数/进程访问次数

29. 分段式存储系统中的地址转换过程

根据逻辑地址得到段号和偏移量，比较段号 S 和段表长度 M，若 S>=M 越界，否则继续；

查询段表，S 对应的段表项地址=段表始址 F+段号 S*段表长度 M，从而取出段长 C，若 C<=偏移量 W，则越界中断，否则继续；

得到物理地址 E=b+W，访问内存；



第六章　处理机调度

1、高级调度、中级调度、低级调度的任务

	要做什么	调度发生在..	发生频率	对进程状态的影响
高级调度（作业调度）	按照某种规则，从后备队列中选择合适的作业将其调入内存，并为其创建进程	外存→内存（面向作业）	最低	无→创建态→就绪态
中级调度（内存调度）	按照某种规则，从挂起队列中选择合适的进程将其数据调回内存	外存→内存（面向进程）	中等	挂起态→就绪态（阻塞挂起→阻塞态）
低级调度（进程调度）	按照某种规则，从就绪队列中选择一个进程为其分配处理机	内存→CPU	最高	就绪态→运行态

2. 引起进程调度的因素有哪些？

（1）正在执行的进程正常终止或异常终止。

（2）正在执行的进程因某种原因而阻塞。

（3）在引入时间片的系统中，时间片用完。

（4）在抢占调度方式中，就绪队列中某进程的优先权变得比当前正在执行的进程高，或者有优先权更高的进程进入就绪队列。

3、P159 所有的进程调度算法

（1）先来先服务 FCFS：每次从就绪队列中选择最先进入该队列的进程，将处理机分配给它，使之投入运行，直到完成或因某种原因而阻塞时才释放处理机。

FCFS 属于不可剥夺算法，算法简单，但效率低；对长作业比较有利，但对短作业不利（相对 SPF 和高响应比），若一个长作业先到达系统，就会使后面的许多短作业等待很长时间，因此它不能作为分时系统和实时系统的主要调度策略；有利于 CPU 繁忙型作业，而不利于 I/O 繁忙型作业。

（2）短作业有先 SJF：从就绪队列中选择一个或几个估计运行时间最短的进程，将处理机分配给它，使之立即执行，直到完成或发生某事件而阻塞时，才释放处理机。

①算法对长作业不利，SJF 调度算法中长作业的周转时间会增加。如果有一个长作业进人系统的后备队列，由于调度程序总是优先调度短作业，将导致长作业长期不被调度，可能会出现“饥饿”现象。

②没有考虑作业 的紧迫程度，因而不能保证紧迫性作业会得到及时处理。

③作业的长短只是根据用户所提供的估计执行时间而定的，而用户有可能会有意或无意地缩短其作业的估计运行时间，算法不一定能真正做到短作业优先调度。

④SJF 调度算法的平均等待时间、平均周转时间最少。

（3）高响应比优先 HRRN：先计算后备作业队列中每个作业的响应比，从中选出响应比最高的作业投入运行。

响应比＝(等待时间＋要求服务时间)/要求服务时间

①作业的等待时间相同时，要求服务时间越短，响应比越高，有利于短作业。

②要求服务时间相同时，作业的响应比由其等待时间决定，等待时间越长，其响应比越高，所以算法实现的又 是先来先服务。

③对于长作业，作业的响应比可以随等待时间的增加而提高，等待时间足够长时，其响应比便可升到很高，也可获得处理机。因此，克服了饥饿状态，兼顾了长作业。

（4）优先级调度：从就绪队列中选择优先级最高的进程，根据新的更高优先级进程能否抢占正在执行的进程，可分为非剥夺式优先级调度算法和剥夺式优先级调度算法

（5） 时间片轮转：进程调度程序总是按到达时间的先后次序选择就绪队列中的第一个进程执行，即先来先服务的原则，但仅运行一个时间片。在使用完一个时间片后，即使进程并未完成其运行，它也必须释放出（被剥夺）处理机给下一个就绪的进程，而被剥夺的进程返回到就绪队列的末尾重新排队，等候再次运行。

①若时间片足够大，以至所有进程均能在一个时间片执行完成，则退化为 FCFS

算法。

②若时间片很小，则处理机切换频繁，开销增大，进程使用时间减少。

（6）多级反馈队列 FB：多级反馈队列调度算法是时间片轮转调度算法和优先级调度算法的综合与发展，通过动态调整进程优先级和时间片大小，多级反馈队列调度算法可以兼顾多方面的系统目标。

4. 哪些是抢占式调度算法？哪些是非抢占调度？

①抢占式：时间片轮转；多级反馈队列；短作业优先；优先级调度

②非抢占式：短作业优先；高响应比优先；优先级调度

5. 高响应比优先调度算法的优点

①作业的等待时间相同时，要求服务时间越短，响应比越高，有利于短作业。

③对于长作业，作业的响应比可以随等待时间的增加而提高，等待时间足够长时，其响应比便可升到很高，也可获得处理机。因此，克服了饥饿状态，兼顾了长作业。

6. 周转时间，带权周转时间

①周转时间：从作业提交到作业完成所消耗的时间。包括作业等待、在就绪队列中排队在处理机上运行以及进行输人/输出操作等所花费时间的总和。

周转时间 T=作业完成时间-作业提交时间

③带权周转时间：带权周转时间是指作业周转时间与作业实际运行时间的比值。

作业的带权周转时间 w=作业的周转时间 T/实际运行时间 Ts

第七章 I/O 设备管理

1. 设备独立性

用户编程时使用的设备与实际使用的设备无关，用户程序在进行输入/输出的时候，不需要考虑具体的输入/输出设备，而可以用一种通用的方式进行输入/输出。

2. 设备逻辑名、设备物理名

3. 消息缓冲队列通信机制应具有的功能

（1）构成消息。发送进程在自己的工作区设置发送区 a 将消息正文和有关控制

信息填入其中。

(2)发送消息。将消息从发送区 a 复制到消息缓冲区并把它插入到目标进程的消息队列中。

(3)接收消息。由目标进程从自己的消息队列中找到第一个消息缓冲区并将其中的消息内容拷贝到本进程的接收区 b 中。

(4)互斥与同步。互斥是指保证在一段时间内只有一个进程对消息队列进行操作；同步是指在接收进程和发送进程之间进行协调。为此应在接收进程的 PCB 中设置用于实现互斥和同步的信号量。

4、I/O 通道的定义及分类

（1）I/O 通道是专门负责输入输出的处理机，它可以识别并执行一系列通道指令，在 DMA 方式的基础上，进一步减少 CPU 的干预，即把对一个数据块的读写的干预减少为对一组数据块的读写以及有关控制管理，进一步提高 CPU 的利用率

（2）①字节多路通道：它适用于连接打印机、终端等低速或中速的 I/O 设备。这种通道以字节为单位交叉工作：当为一台设备传送一个字节后，立即转去为另一它设备传送一个字节。

②数组选择通道：它适用于连接磁盘、磁带等高速设备。这种通道以“组方式”工作，每次传送一批数据，传送速率很高，但在一段时间只能为一台设备服务。每当一个 I/O 请求处理完之后，就选择另一台设备并为其服务，可见通道利用率很低

③数组多路通道：这种通道综合了字节多路通道分时工作和选择通道传输速率高的特点，其实质是：对通道程序采用多道程序设计技术，使得与通道连接的设备可以并行工作。

5. 在设备管理中引入缓冲技术的原因

①缓和 CPU 与 I/O 设备间速度不匹配的矛盾

②减少 CPU 的中断频率，放宽对中断响应的限制

③提高 CPU 和 I/O 设备之间的并行性

6. 缓冲技术的分类

①单缓冲：所以采用单缓冲策略，处理一块数据平均耗时 Max(C, T)+M

②双缓冲：采用双缓冲策略，处理一块数据平均耗时 Max(M+C, T)

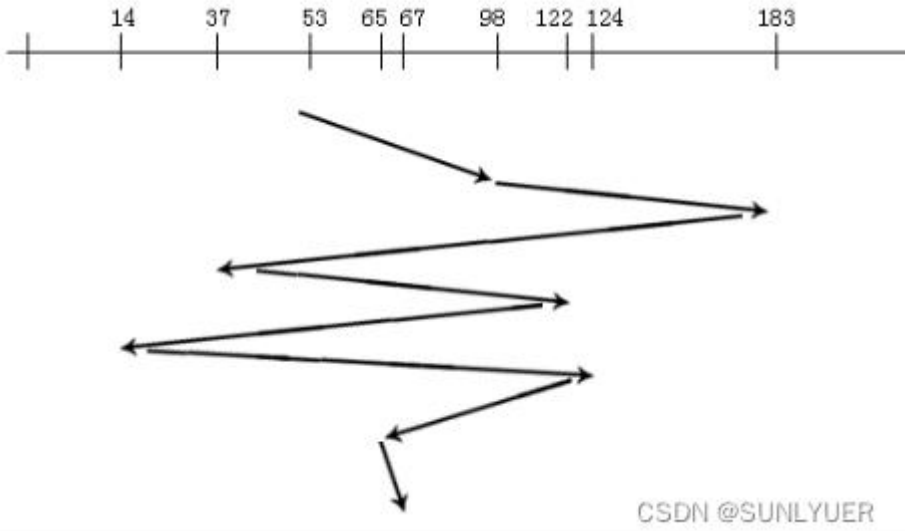
③循环缓冲

④缓冲池

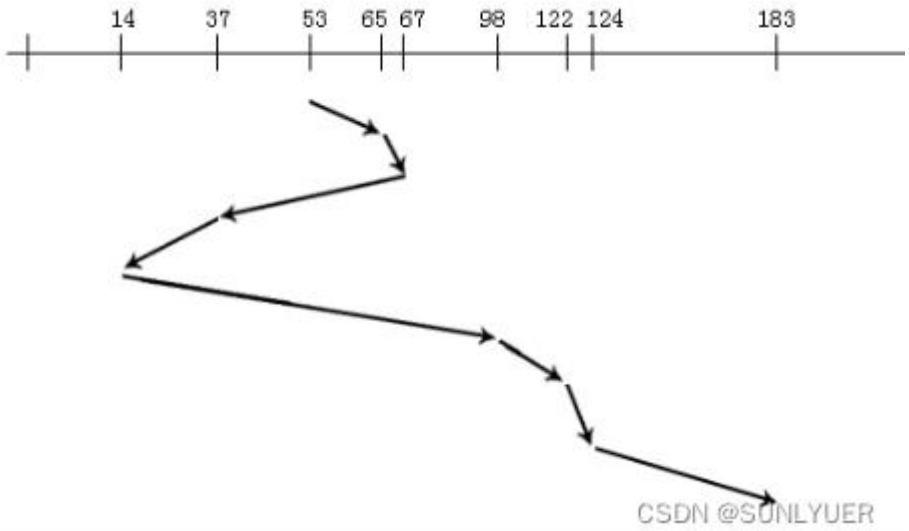
7. 磁盘调度算法

假设磁盘访问序列：98，183，37，122，14，124，65，67。读写头起始位置：53

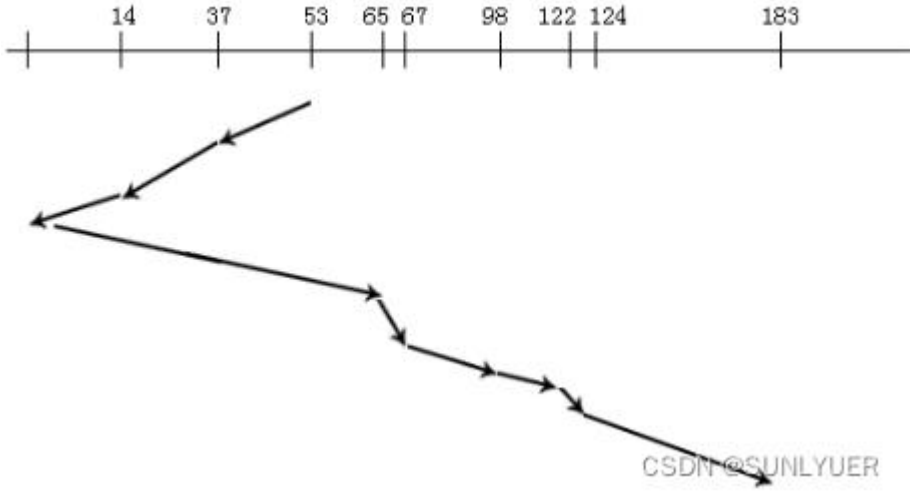
①先进先出 FCFS：进程处理起来非常简单，但平均寻道长度会很长，适用于访问请求不是很多的情况



②最短时间优先：已知磁头的初始位置，则最先被处理就是距离磁头位置最近的进程，处理完成后再处理距离当前磁道最近的进程，直到所有的进程被处理。该算法的优点是平均寻道长度会大大减少，缺点是距离初始磁头较远的服务长期得不到处理，产生“饥饿”现象。



③扫描策略：磁头仅沿一个方向进行扫描，在扫描途中完成所有没有完成的请求，直到磁头到达磁盘在这个方向上的最后一个磁道或者这个方向上最后一个请求所在的磁道。



④循环扫描策略：在磁盘扫描算法的基础上改变磁头的扫描路径：扫描到最内层之后从最外层向内继续扫描，即扫描方向一致，当磁头扫描到磁盘的最内层时，磁头跳转到磁盘最外层重新向内扫描，这样就可以有效的避免将已经扫描过的磁道重新扫