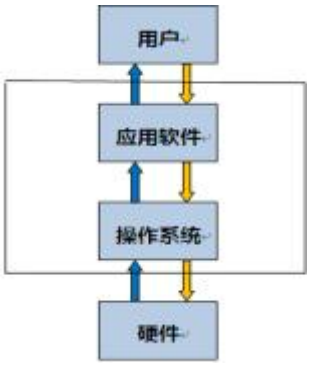
操作系统

第一节 基础知识

一、操作系统的概念

操作系统也称计算机管理控制程序，是控制和管理计算机系统内各种软、硬件资源，合理有效地组织计算机系统的工作流程，为用户提供一个使用方便、可扩展的工作环境，从而起到连接计算机用户接口的作用。

对于计算机系统而言，操作系统是核心和基石，它就是控制整个系统的神经中枢，一个庞大的管理控制程序。常见的操作系统有 DOS、OS/2、UNIX、Linux、Windows 等。



二、操作系统的功能

（一）处理器管理

一般情况下，有多个任务需要完成，而 CPU 在同一时刻只能处理单个任务。为了保证完成多个作业并提高 CPU 效率，需要操作系统进行统一的管理和调度，将 CPU 的时间合理、动态地分配给各个任务。

（二）内存管理

操作系统对内存的存储空间进行统一安排和管理，通过对内存空间进行合理分配，保证各作业所占用的内存空间不发生冲突，以便用户能合理地使用内存。

（三）设备管理

主要任务是方便用户使用各种输入/输出设备。当用户使用外部设备时，通过它向系统提出申请，由操作系统进行统一分配；当用户的程序需要用到某个外设时，由操作系统负责驱动该外部设备。

（四）文件管理

文件是计算机的软件资源，程序和数据都是以文件的形式存储在外存设备上的。文件管理功能主要包括外存空间的管理和回收，对文件进行存取、检索、更新，及有效地实现文件共享。

（五）作业管理

作业是用户请求完成的一个独立任务。操作系统提供一套控制操作命令，用来控制程序的运行，在多道作业程序运行时，操作系统使用户合理地共享计算机系统资源。

三、操作系统的分类

操作系统按不同的分类标准可以有不同的分类方法。

（一）按计算机系统分类

1.单用户操作系统

一个用户使用一台计算机的操作系统，这个用户占用计算机的全部资源，常见的有PC DOS、MS DOS等。

2.多用户操作系统

多个用户使用一套计算机的操作系统，这些用户共享计算机的硬件和软件资源，常见的有 UNIX 操作系统。

（二）按计算机的工作过程分类

1.批处理系统

批处理系统是指用户可以将一个或多个任务交给计算机操作系统，由系统运行直到结束。批处理系统又可分为单道批处理系统和多道批处理系统。

单道批处理系统：用户一次提供已安排好先后顺序的多个任务，由操作系统按照顺序自动完成任务运行的全过程。

多道批处理系统：又称多任务操作系统，用户提供多个任务同时运行，用户将要执行的任务和要求一起交代给计算机，然后由操作系统调度并协调各个作业运行，直到完成全部运行结果。

2.分时操作系统

通常用于多用户操作系统。同时在一台计算机上连接多个终端，操作系统按预先分配给各个终端的时间片段轮流为各个用户终端服务，使各个终端用户分时共享计算机系统的资源。

3.实时操作系统

实时操作系统是一种时间性强、反应快速的操作系统，它在规定时间内处理完任务并输出结果。这类操作系统应该具备实时性和可靠性。

	分时系统	**实时系统**
交互能力	强（通用系统）	弱（专用系统）
响应时间	秒级	及时，毫秒/微妙级
可靠性	一般要求	要求更高

四、常见的操作系统

操作系统	**介绍**
DOS	Disk Operation System（磁盘操作系统），它采用的是命令行形式，靠输入命令来进行人机对话，并通过命令的形式把指令传给计算机。
Windows	是美国微软开发的窗口化操作系统，采用了 GUI 图形化操作模式，比起从前的指令操作系统如 DOS 更为人性化。Windows 操作系统是目前世界上使用最广泛的操作系统。目前最新的版本是 Windows 10。
UNIX	多用户、多任务操作系统，属于分时操作系统。支持并行处理、多处理器、独有的 64 位计算能力。它由三部分组成：内核 Kernel（UNIX 操作系统核心），外壳 shell（命令解释层），工具和应用程序。

Linux	来源于 UNIX，是一个开放源码的操作系统，在 AT&T 的贝尔实验室开发出来，目前使用的 Linux 系统既有 32 位的，也有 64 位的。可靠、高效、使用成本基本为零，成为一种重要的操作系统。
Netware	是一个开放的网络服务器平台，可方便对其进行扩充。它满足不同类型的网络间实现相互通信的需要，同时也不需专用服务器，是目前 PC 局域网中最主要的操作系统。

第二节 进程管理

在未配置 OS 的系统中，程序的执行方式是顺序执行，即必须在一个程序执行完后，才允许另一个程序执行；在多道程序环境下，则允许多个程序并发执行。程序的这两种执行方式间有着显著的不同，也正是程序并发执行时的这种特征，才在操作系统中引入了进程的概念。

一、程序的顺序执行及其特征

（一）程序顺序执行的概念

通常可以把一个应用程序分成若干个程序段，在各程序段之间，必须按照某种先后次序顺序执行，仅当前一操作(程序段)执行完后，才能执行后继操作。

（二）程序顺序执行时的特征

顺序性：处理器的操作严格按照程序所规定的顺序执行，即每一操作必须在上一个操作结束之后开始。

封闭性：程序是在封闭的环境下执行的，即程序运行时独占全机资源状态，(除初始状态外)只有本程序才能改变它。程序一旦开始执行，其执行结果就不受外界因素影响。

可再现性：只要程序执行时的环境和初始条件相同，当程序重复执行时，不论它是从头到尾不停地执行，还是“停停走走”地执行，都将获得相同的结果。

程序顺序执行时的特性，为程序员检测和校正程序的错误带来了很大的方便。

二、程序的并发执行及其特征

程序的并发执行，虽然提高了系统吞吐量，但也产生了下述一些与程序顺序执行时不同的特征。

（一）间断性

程序在并发执行时，由于它们共享系统资源，以及为完成同一项任务而相互合作，致使在这些并发执行的程序之间，形成了相互制约的关系。

（二）失去封闭性

程序在并发执行时，是多个程序共享系统中的各种资源，因而这些资源的状态将由多个程序来改变，致使程序的运行失去了封闭性。

（三）不可再现性

程序在并发执行时，由于失去了封闭性，也将导致其失去可再现性。

顺序执行和并发执行的区别：

顺序执行	并发执行
程序顺序执行	间断执行，多个程序各自在“走走停停”中进行
程序具有封闭性	程序失去封闭性
独享资源	共享资源
具有可再现性	失去可再现性
	有直接和间接的相互制约

三、多道程序设计概念及其优点

多道程序设计：是在一台计算机上同时运行两个或多个程序。

多道程序设计的特点：多个程序共享系统资源、多个程序并发执行。

多道程序设计的优点：提高资源利用率、增加系统吞吐量。

四、进程的特征与状态

（一）进程的定义和特征

为使程序能并发执行，且为了对并发执行的程序加以描述和控制，人们引入了“进程”的概念。

（二）进程与程序的主要区别

程序是永存的，进程是暂时的。

程序是静态的概念，进程是动态的概念。

进程由三部分组成：程序+数据+进程控制块（描述进程活动情况的数据结构）。

进程的三种基本状态，进程执行时的间断性决定了进程可能具有多种状态。事实上，运行中的进程可能具有以下三种基本状态。

1.就绪状态

当进程已分配到除 CPU 以外的所有必要资源后，只要再获得 CPU，便可立即执行，进程这时的状态称为就绪状态。在一个系统中处于就绪状态的进程可能有多个，通常将它们排成一个队列，称为就绪队列。

2.执行状态

进程已获得 CPU，其程序正在执行。在单处理器系统中，只有一个进程处于执行状态；在多处理器系统中，则有多个进程处于执行状态。

3.阻塞状态

正在执行的进程由于发生某事件而暂时无法继续执行时，便放弃处理器而处于暂停状态，亦即进程的执行受到阻塞，把这种暂停状态称为阻塞状态，有时也称为等待状态或封锁状态。

（三）进程状态的转换

就绪态->运行态

运行态->就绪态

运行态->阻塞态

阻塞态->就绪态

五、临界资源、临界区

临界资源：一次仅允许一个进程使用的资源。

临界区：在每个进程中访问临界资源的那段程序。

互斥进入临界区的准则：

如果有若干进程要求进入空闲的临界区，一次仅允许一个进程进入。

任何时候，处于临界区内的进程不可多于一个。如已有进程进入自己的临界区，则其他所有试图进入临界区的进程必须等待。

进入临界区的进程要在有限时间内退出，以便其他进程能及时进入自己的临界区。

如果进程不能进入自己的临界区，则应让出 CPU，避免进程出现“忙等”现象。

六、线程

（一）线程的概念及由来

自从在 20 世纪 60 年代人们提出了进程的概念后，在 OS 中一直都是以进程作为能拥有资源和独立运行的基本单位的。直到 20 世纪 80 年代中期，人们又提出了比进程更小的能独立运行的基本单位——线程(Threads)，试图用它来提高系统内程序并发执行的程度，从而可进一步提高系统的吞吐量。

（二）线程与进程的比较

线程具有许多传统进程所具有的特征，所以又称为轻型进程(Light-Weight Process)或进程元，相应地把传统进程称为重型进程(Heavy-Weight Process)，传统进程相当于只有一个线程的任务。

七、进程的调度

进程调度算法：解决以何种次序对各就绪进程进行处理器的分配以及按何种时间比例让进程占用处理器。

进程调度的方式通常有可剥夺和非剥夺方式两种。

所谓可剥夺方式，是指就绪队列中一旦有优先级高于当前运行进程的优先级的进程存在时，便立即发生进程调度，转让处理器。

非剥夺方式则是指即使在就绪队列中存在有优先级高于当前运行进程的进程，当前进程仍将继续占有处理器，直到该进程完成或某种事件发生（如 I/O 事件）让出处理器。

因为“可剥夺”优先级调度始终保证在处理器上运行的是优先级最高的进程，这样，当处理器正在运行某个进程时，很可能会被其他优先级更高的进程“抢占”引起处理器调度，和“非剥夺”算法相比，前者的调度次数会更频繁，而每调度一次都会引起保护现场、恢复现场的工作，所以“可剥夺”的优先级调度算法开销更大。

进程调度常用算法如下：

（一）先来先服务调度算法（FIFO）

这种调度算法是按照进程进入就绪队列的先后次序选择可以占用处理器的进程。当有进程就绪时，把该进程排入就绪队列的末尾，而进程调度总是把处理器分配给就绪队列中的第一个进程。一旦一个进程占有了处理器，它就一直运行下去，直到因等待某事件或进程完成了工作才让出处理器。

（二）优先数调度算法（HPF）

对每个进程确定一个优先数，进程调度总是让具有最高优先数的进程先使用处理器。如果进程具有相同的优先数，则对这些有相同优先数的进程再按先来先服务的次序分配处理器。就绪队列中进程可按优先数从大到小排列，这样，进程调度也总是把处理器分配给就绪队列中的第一个进程。

进程被创建时系统为其确定一个优先数，进程的优先数可以是固定的，也可随进程的执行过程而动态变化。

优先数调度算法分为“非抢占式”的与“可抢占式”的两种。

（三）时间片轮转调度算法（RR）

系统规定一个“时间片”的值。调度算法让就绪进程按就绪的先后次序排成队列，每次总是选择就绪队列中的第一个进程占用处理器，但规定只能使用一个“时间片”。如果一个时间片用完，进程工作尚未结束，则它也必须让出处理器而被重新排到就绪队列的末尾，等待再次运行，当再次轮到运行时，

重新开始使用一个新的时间片。这样，就绪队列中的进程就依次轮流地占用处理器运行。

轮转法主要用于分时系统的调度算法，它具有较好的响应时间，且对每个进程来说都具有较好的公平性。

先来先服务调度算法是“非抢占式”的；“优先数调度算法”可以是“非抢占式”的，也可以是“抢占式”的；“时间片轮转调度算法”是一种“抢占式”的。

八、多道程序系统中进程的并发执行

一组在逻辑上互相独立的程序或程序段在执行过程中，其执行时间在客观上互相重叠，即一个程序段的执行尚未结束，另一个程序段的执行已经开始的这种执行方式。

程序的并发执行可进一步分为两种，第一种是多道程序系统的程序执行环境变化所引起的多道程序的并发执行；第二种并发执行是在某道程序的几个程序段中，包含着一部分可以同时执行或顺序颠倒执行的代码。

为了合理利用系统资源，更好地发挥各种资源的效益，使各种物理设备之间的时间性限制条件减少到最低限度，最大限度地提高系统的效率，因而引出了多道程序方法。其实质是减少程序的顺序性，提高系统的并行性。

第三节 作业管理

一、作业管理的基本概念

（一）作业

作业是用户交给计算机的具有独立功能的任务。

作业在系统中存在与否的唯一标志是作业控制块（JCB），系统是根据作业控制块 JCB 来感知作业的存在的。

采用批处理控制方式的作业，用户把对作业执行的控制意图用作业控制语言写成一份说明书，连同该作业的源程序和初始数据一起输入到计算机系统，系统就可按用户说明书来控制作业的执行。作业执行过程中用户不能干预，一切由系统自动地控制作业的执行。

（二）作业步

作业中每个步骤称为作业步。各作业步之间相对独立，又相互关联。

（三）作业流

作业流是指在批处理系统中把一批作业安排在输入设备上，然后依次读入系统进行处理，从而形成了作业流。

二、作业管理的功能

（一）作业调度定义

作业调度是按某种算法从后备作业队列中选择作业进入主存，并为作业做运行前的准备和完成后的善后工作。

作业调度从处于后备状态的队列中选取适当的作业投入运行。从作业提交给系统到作业完成的时间间隔叫做周转时间，是作业等待时间和运行时间之和。等待时间是指作业从进入后备队列到被调度程序选中时的时间间隔。

每个作业的周转时间=执行结束时间—进入输入井时间

一个作业从进入系统到运行结束，一般要经历进入、后备、运行和完成四个阶段。

（二）作业调度的目标

使作业运行最大限度的发挥各种资源的利用率，并保持系统内各种活动的充分运行。

（三）作业调度的主要任务

1. 按某种算法从后备队列中选择作业。

2. 为选中的作业分配资源。

3. 为选中的作业建立相应的进程。

4. 为选中的作业构造相应的数据结构。

5. 作业结束时完成该作业的善后处理（回收资源等）。

（四）主要作业调度算法

1.先来先服务算法（FCFS）

该算法是一种较简单的调度算法，它是按照作业进入输入井的先后次序来挑选作业，先进入的作业优先被挑选。但要注意，不是先进入的一定先被选中，只有满足必要条件的作业才可能被选中。

假设有三道作业，它们的提交时间及运行时间由下表给出：

作业	提交时间（小时）	执行时间（小时）
1	10:00	2
2	10:20	1
3	10:40	0.5

求作业的开始时间，结束时间和周转时间：

作业	提交时间（小时）	执行时间（小时）	开始时间	结束时间	周转时间
1	10:00	2	10:00	12:00	2
2	10:20	1	12:00	13:00	2 小时 40 分
3	10:40	0.5	13:00	13:30	2 小时 50 分

2.短作业优先算法（SJF）

这种算法要求用户预先估计自己作业所需要计算的时间，并在作业说明书中说明。调度时优先选择计算时间短且资源能得到满足的作业。这种算法使作业的平均等待时间最短，并能降低作业的平均周转时间，从而提高系统的吞吐能力。

在一个多道程序设计系统中，不采用移动技术的可变分区方式管理内存。设用户空间为 100K，主存空间采用最先适应分配算法，采用计算机时间短的作业优先算法管理作业。今有如下所示的作业序列，请分别列出各个作业的开始执行时间、完成时间和周转时间（忽略系统开销）。

作业名	进入输入井时间	需计算时间	主存需求量
JOB1	8.0 时	1 小时	20K
JOB2	8.2 时	0.6 小时	60K
JOB3	8.4 时	0.5 小时	25K
JOB4	8.6 时	0.4 小时	20K

3.最高响应比作业优先算法

最高响应比优先算法综合考虑等待时间和计算时间，把响应比定义为：

响应比=等待时间/计算时间

可以看出计算时间短的作业响应比较高，所以能被优先选中；但等待时间长的作业响应比也会较高，这样就不会因不断地有小作业进入输入井而使大作业无限制地被推迟。

4.优先数调度算法

系统为每一作业确定一个优先级，优先级高的作业优先被选取。优先级的确定可根据作业的缓急程度、估计计算时间、作业等待时间、资源申请情况、付费情况等因素综合考虑，既照顾用户要求，也考虑系统效率。

5.均衡调度算法

根据各作业对不同资源的申请进行调度，其目标是使系统中的各类资源能均衡利用，避免资源忙闲不均的情况。

第四节 处理器调度与死锁

在多道程序环境下，主存中有着多个进程，其数目往往多于处理器数目。这就要求系统能按某种算法，动态地把处理器分配给就绪队列中的一个进程，使之执行。分配处理器的任务是由处理器调度程序完成的。由于处理器是最重要的计算机资源，提高处理器的利用率及改善系统性能(吞吐量、响应时间)，在很大程度上取决于处理器调度性能的好坏，因而，处理器调度便成为操作系统设计的中心问题之一。

一、处理器的三级调度

处理器的三级调度：高级调度、中级调度、低级调度。

（一）高级调度（又称为作业调度）

主要功能：按照某种原则从磁盘某些盘区的作业队列中选取作业进入内存，为作业做好运行前的准备工作以及作业完成后的善后工作。具体如下：

1. 记录系统中各个作业的情况。
2. 按照某种调度算法从后备作业队列中挑选作业。
3. 为选中的作业分配内存和外设等资源。
4. 为选中的作业建立相应的进程。
5. 作业结束后进行善后处理工作。

作业调度程序挑选作业进入内存是一个宏观的概念，被选进内存运行的作业只是具有了竞争 CPU 的机会（将来真正在 CPU 上运行的是该作业的一个进程），进程调度程序才是真正让某个就绪进程到 CPU 上运行。

（二）中级调度

主要功能：决定哪些进程暂时不允许参与竞争 CPU 资源，起到短期调整系统负荷的作用。

使用的方法：通过挂起和解除挂起一些进程，达到操作系统负载平衡的目的。

（三）低级调度（进程调度）

主要功能：按照某种原则将 CPU 分配给就绪进程，执行低级调度的程序称为进程调度程序，它实现 CPU 在进程间的转换，必须常驻内存，是操作系统内核的主要部分。

作业调度 （宏观调度）	为进程活动做准备，即有获得处理器的资格	调度次数	有的系统不设作业调度
进程调度 （微观调度）	使进程活动起来，即分配得到了处理器	调度频率高	进程调度必不可少

二、产生死锁的原因和必要条件

（一）死锁的产生

概念：指多个进程因竞争共享资源而造成的一种僵局，若无外力作用，这些进程都将永远不能再向前推进。

产生原因：竞争系统资源、进程的推进顺序不当。

必要条件：要产生死锁，四个条件必同时成立。

1. 互斥条件：进程要求对所分配的资源进行排他性控制，即在一段时间内某资源仅为一进程所占用。

2. 请求和保持条件：当进程因请求资源而阻塞时，对已获得的资源保持不放。

3. 不剥夺条件：进程已获得的资源在未使用完之前，不能剥夺，只能在用完时由自己释放。

4. 环路等待条件：在发生死锁时，必然存在一个进程--资源的环形链。

安全状态与不安全状态：安全状态指系统能按某种进程顺序来为每个进程分配其所需资源，直至最大需求，使每个进程都可顺利完成。若系统不存在这样一个序列，则称系统处于不安全状态。

（二）解决死锁的基本方法

1.预防死锁

资源一次性分配（破坏请求和保持条件）。

可剥夺资源：即当某进程新的资源未满足时，释放已占有的资源（破坏不可剥夺条件）。

资源有序分配法：系统给每类资源赋予一个编号，每一个进程按编号递增的顺序请求资源，释放则相反（破坏环路等待条件）。

2.避免死锁

预防死锁的几种策略，会严重地损害系统性能。因此在避免死锁时，要施加较弱的限制，从而获得较满意的系统性能。由于在避免死锁的策略中，允许进程动态地申请资源，因而，系统在进行资源分配之前预先计算资源分配的安全性。若此次分配不会导致系统进入不安全状态，则将资源分配给进程；否则，进程等待。其中最具有代表性的避免死锁算法是银行家算法。（银行家算法的基本思想：分配资源之前，判断系统是否是安全的；若是，才分配）。

3.检测死锁

首先为每个进程和每个资源指定一个唯一的号码，然后建立资源分配表和进程等待表。

4.解除死锁

当发现有进程死锁后，便应立即把它从死锁状态中解脱出来。

5.剥夺资源

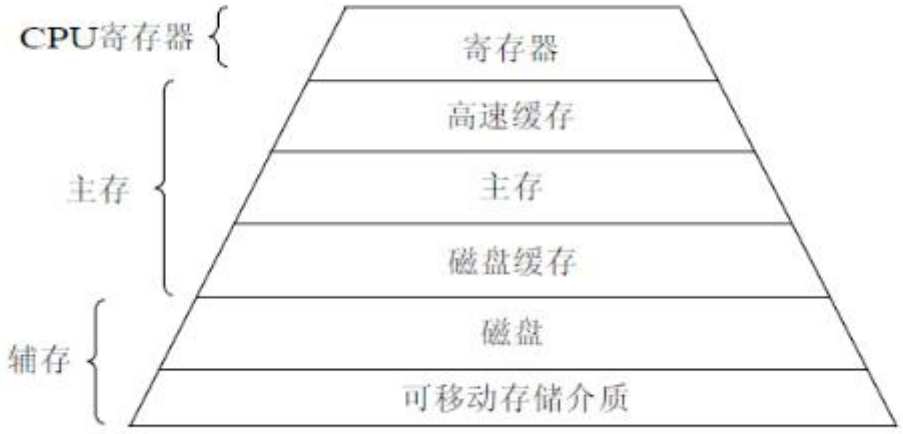
从其他进程剥夺足够数量的资源给死锁进程，以解除死锁状态。

6.撤销进程

可以直接撤销死锁进程或撤销代价最小的进程，直至有足够的资源可用，死锁状态、消除为止；所谓代价是指优先级、运行代价、进程的重要性和价值等。

第五节 存储器管理

对于通用计算机而言，存储层次至少应具有三级：最高层为 CPU 寄存器，中间为主存，最底层是辅存。在较高档的计算机中，还可以根据具体的功能分工细划为寄存器、高速缓存、主存储器、磁盘缓存、固定磁盘、可移动存储介质等 6 层。



计算机系统存储层次示意图

一、主存储器与寄存器

（一）主存储器

主存储器(简称内存或主存)是计算机系统中一个主要部件，用于保存进程运行时的程序和数据，也称可执行存储器。CPU 的控制部件只能从主存储器中取得指令和数据，数据能够从主存储器读取并将它们装入到寄存器中，或者从寄存器存入到主存储器。CPU 与外围设备交换的信息一般也依托于主存储器地址空间。由于主存储器的访问速度远低于 CPU 执行指令的速度，为缓和这一矛盾，在计算机系统中引入了寄存器和高速缓存。

（二）寄存器

寄存器访问速度最快，能完全与 CPU 协调工作，但价格却十分昂贵，因此容量不可能做得很大。寄存器用于加速存储器的访问速度，如用寄存器存放操作数，或用作地址寄存器加快地址转换速度等。

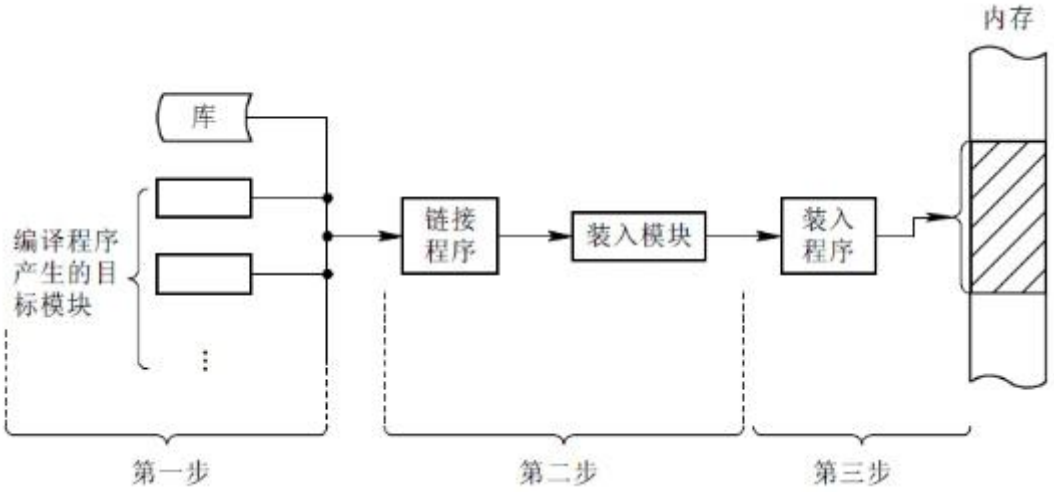
二、程序的装入和链接

在多道程序环境下，要使程序运行，必须先为之创建进程。而创建进程的第一件事，便是将程序和数据装入内存。如何将一个用户源程序变为一个可在内存中执行的程序，通常都要经过以下几个步骤。

编译：由编译程序(Compiler)将用户源代码编译成若干个目标模块(Object Module)。

链接：由链接程序(Linker)将编译后形成的一组目标模块，以及它们所需要的库函数链接在一起，形成一个完整的装入模块(Load Module)。

装入：由装入程序(Loader)将装入模块装入内存。



三、存储器有关概念

逻辑地址：用户程序经编译之后的每个目标模块都以 0 为基地址顺序编址。

物理地址：内存中各物理单元的地址是从统一的基地址顺序编址。

重定位：把逻辑地址转变为内存的物理地址的过程。

静态重定位：是在目标程序装入内存时，由装入程序对目标程序中的指令和数据的地址进行修改，即把程序的逻辑地址都改成实际的内存地址。重定位在程序装入时一次完成。

动态重定位：在程序执行期间，每次访问内存之间进行重定位，这种变换是靠硬件地址变换机构实现的。

碎片：内存中容量太小、无法被利用的小分区。

四、存储管理基本技术

存储管理的目的和功能：

对主存空间进行分配和管理；

提高主存的利用率；

“扩充”主存容量；

实现地址的变换。

分区法：把内存划分成若干分区，每个分区里容纳一个作业。

固定分区：分区的个数、分区的大小固定不变；每个分区只能放一道作业。

优点：管理方式简单。

缺点：内存空间利用率低。

动态分区法：分区大小和个数依作业情况而定；作业进入内存时才建分区。

优点：按需分配内存。

缺点：产生大量碎片。

重定位分区分配：通过紧缩可解决碎片问题；作业在内存中可以移动。

优点：解决了碎片的问题，提高了主存利用率。

缺点：增加了开销，需消耗大量的 CPU 时间。

对换技术：作业（或进程）在内存和磁盘之间交换，换出暂时不能运行的作业（或进程）；换入具备运行条件的作业（或进程）。

五、虚拟存储器

虚拟存储器：是由操作系统提供的一个假想的特大存储器。

虚拟存储器的基本特征：

虚拟扩充：不是物理上，而是逻辑上扩充了内存容量。

部分装入：每个作业不是全部一次性地装入内存，而是只装入一部分。

离散分配：不必占用连续的空间，而是“见缝插针”。

多次对换：所需的全部程序和数据要分成多次调入内存。

虚拟存储器受到的限制：指令中表示地址的字长、外存的容量。

六、分页存储管理技术

（一）分页的概念

逻辑空间等分为页，物理空间等分为块，与页面大小相同。

逻辑地址表示：（如，页面大小为 1K）。

内存分配原则：以块为单位，逻辑上相邻的页可以分配在不相邻的内存块中。

页表：实现从页号到物理块号的地址映射。

地址映射：由硬件完成。

（二）请求分页的基本思想

地址空间分页，内存分块，页与块大小相同； 作业部分装入内存；作业所占的各块不连续；硬件通过页表生成访内地址。若缺页，进行缺页中断处理，换入内存。 利用页表可加速地址转换。

七、分段存储管理技术

（一）分段的概念

逻辑空间分段：段是信息的逻辑单位，每段对应一个相应的程序模块，有完整的逻辑意义。

程序的地址结构：逻辑地址表示（二维的地址空间）。

内存分配：内存以段为单位进行分配，每个段单独占用一块连续的内存分区。

段表：实现每个逻辑段到物理内存中分区位置的映射。

地址转换：将逻辑地址转换成内存的物理地址，完成地址重定位。需要指出的是，地址转换是操作系统的地址变换机构自行完成的，无需用户干预，这样我们使用操作系统时，才方便而可靠。

（二）分页与分段的区别

分页	信息的物理单位	大小一样，由系统固定	地址空间是一维的
分段	信息的逻辑单位	大小不等，由用户确定	地址空间是二维的

八、虚存中的置换算法

先进先出法（FIFO）：将最先进入内存的页换出内存。

最佳置换法（OPT）：将将来不再被使用或是最远的将来才被访问的页换出内存。

最近最少使用置换法（LRU）：将最近一段时间里最久没有使用过的页面换出内存。

最近未使用置换法（NUR）：是 LRU 近似方法，比较容易实现，开销也比较小。

实现方法：在存储分块表的每一表项中增加一个引用位，操作系统定期地将它们置为 0。当某一页被访问时，由硬件将该位置 1。需要淘汰一页时，把该位为 0 的页淘汰出去，因为最近一段时间里它未被访问过。

第六节 设备管理

一、设备分类及设备标识

（一）设备分类

存储设备（外存、辅助存储器）：用于存储信息的设备。

输入/输出设备：用于输入/输出信息的设备。

（二）设备标识

设备绝对号：系统为设备指定的唯一代号。

设备相对号：用户自己规定的设备序号。

二、引入缓冲的目的和缓冲区的设置方式

（一）引入缓冲区的目的

缓和 CPU 与外设间速度不匹配的矛盾；

提高 CPU 与外设之间的并行性；

减少对 CPU 的中断次数。

（二）缓冲区的设置方式

单缓冲：当数据到达率与离去率相差很大时，可采用单缓冲方式。

双缓冲：当信息输入率和输出率相同（或相差不大）时，可利用双缓冲区，实现两者的并行。

多缓冲：对于阵发性的输入、输出，为了解决速度不匹配问题，可以设立多个缓冲区。

三、设备管理的目标

设备管理的目标：使用方便、与设备无关、效率高、管理统一。

四、设备管理功能

监视设备状态：记住所有设备、控制器和通道的状态，以便有效的调度和使用它们。

进行设备分配：按照设备的类型和系统中采用的分配算法，实施设备分配。这一功能由设备分配程序完成。

完成 I/O 操作：通常完成这一部分功能的程序叫做设备驱动程序。系统按照用户的要求调用具体的设备驱动程序，启动相应的设备，进行 I/O 操作；并且处理来自设备的中断。操作系统中每类设备都有自己的设备驱动程序。

缓冲管理与地址转换：由于外设与主机间的速度差异，大多数 I/O 操作都涉及到缓冲区，因此系统应对缓冲区进行管理。此外，用户程序应与实际使用的物理设备无关，这就需要将用户在程序中使用的逻辑设备地址转换成物理设备的地址。

五、常用设备分配技术

根据设备的使用性质，可将设备分成：独占设备、共享设备和虚拟设备。

独占设备：不能共享的设备。即：在一段时间内，该设备只允许一个进程独占，如打印机。

共享设备：可由若干个进程同时共享的设备，如磁盘机。

虚拟设备：是利用某种技术把独占设备改造成可由多个进程共享的设备。

针对三种设备采用三种分配技术：独占分配、共享分配和虚拟分配。

独占分配技术：是把独占设备固定地分配给一个进程，直至该进程完成 I/O 操作并释放它为止。

共享分配技术：通常适用于高速、大容量的直接存取存储设备。由多个进程共享一台设备，每个进程只用其中的一部分。

虚拟分配技术：利用共享设备去模拟独占设备，从而使独占设备成为可共享的、快速 I/O 的设备。实现虚拟分配的最有名的技术是 SPOOLing 技术，也称作假脱机操作。

六、IO 控制方式

在整个 IO 控制方式的发展过程中，始终贯穿着这样一条宗旨：即尽量减少主机对 IO 控制的干预，把主机从繁杂的 IO 控制事务中解脱出来，以便更多地去完成数据处理任务。

（一）程序 IO 方式

处理器对 IO 采用程序 IO 方式，即采用“忙——等待”方式，在处理器向控制器发出一条 IO 指令启动输入设备输入数据时，要同时把状态寄存器中的忙闲标志置为 1，然后便不断的循环测试，直到标志为 0。

当标志为 1 时，表示输入机未输完一个字符；

当标志为 0 时，表示输入机已经将输入数据送往控制器的数据寄存器中。

在这种方式中，由于 CPU 高速性和 IO 设备的低速性，致使 CPU 的绝大部分时间都处于忙等状态。造成了 CPU 极大的浪费。

如果当一个字符输入完后，由 IO 设备向 CPU 报告，这样就可以不让 CPU 忙等了，于是中断处理方式产生了。

（二）中断驱动 IO 控制方式

当某进程要启动某个 IO 设备工作时，便由 CPU 向相应的设备控制器发出一条 IO 命令，然后立即返回继续执行原来的任务。设备控制器于是按照该命令的要求去控制指定 IO 设备。此时，CPU 与 IO 设备并行操作。一旦数据进入数据寄存器，控制器便通过控制线向 CPU 发送一中断信息，由 CPU 检查输入过程中是否有错，若无错，便向控制器发送取走数据的信号，然后再通过控制器及数据线，将数据写入内存指定单元中。

（三）直接存储器访问 DMA IO 控制方式

虽然中断驱动 IO 比程序 IO 方式更有效，但注意到，它仍是以字（节）为单位进行 IO 的，每当完成一个字（节）的 IO 时，控制器便要向 CPU 请求一次中断。采用中断驱动 IO 方式时的 CPU，是以字（节）为单位进行干预的。如果将这种方式用于块设备的 IO 中，显然是极其低效的。为了进一步减少 CPU 对 IO 的干预而引入了直接存储器访问方式。

该方式的特点：

1. 数据传输的基本单位是数据块。

2. 所传送的数据是从设备直接送入内存，或者相反。

3. 仅在传送一个或多个数据块的开始或结束时，才需要 CPU 干预，整块数据的传送是在控制器的控制下。

可以看出：DMA 方式较之中断驱动方式，又是成百倍的减少了 CPU 对 IO 的干预，进一步提高了 CPU 与 IO 设备的并行操作程度。

DMA 控制器由三部分组成：

（1）主机与 DMA 控制器的接口。

（2）DMA 控制器与块设备的接口。

（3）IO 控制逻辑。

四类寄存器：

命令状态寄存器：用于接收从 CPU 发来的 IO 命令或有关控制信息，或设备的状态。

内存地址寄存器 MAR：在输入时，它存放把数据从设备传送起始目标地址；在输出时，它存放由内存到设备的内存源地址。

数据寄存器 DR：用于暂存从设备到内存，或从内存到设备的数据。

数据计数器 DC：存放本次 CPU 要读或写的字（节）数。

第七节 文件管理

一、文件及文件系统的概念

文件：是被命名的数据的集合体。

文件系统：是操作系统中负责操纵和管理文件的一整套设施，它实现文件的共享和保护，方便用户“按名存取”。

二、文件的类型

按性质和用途分类：系统文件、库文件、用户文件。

系统文件：由系统软件构成的文件，只允许用户通过系统调用或系统提供的专用命令来执行它们，不允许对其进行读写和修改。主要由操作系统核心和各种系统应用程序或实用工具程序和数据组成。

库文件：库文件允许用户对其进行读取和执行，但不允许对其进行修改 。主要由各种标准子程序库组成。

用户文件：是用户通过操作系统保存的用户文件，由文件的所有者或所有者授权的用户才能使用。主要由用户的源程序源代码、可执行目标程序的文件和用户数据库数据等组成。

按操作保护分类：只读文件、可读可写文件、可执行文件。

只读文件：只允许文件主及被核准的用户去读文件，而不允许写文件。

可读可写文件：允许文件主及被核准的用户去读和写文件。

可执行文件：允许文件主及被核准的用户去调用执行该文件而不允许读和写文件。

按用户观点分类（UNIX 系统文件分类）：

普通文件(常规文件)：是指系统中最一般组织格式的文件，一般是字符流组成的无结构文件。

目录文件：是由文件的目录信息构成的特殊文件，操作系统将目录也做成文件，便于统一管理特殊文件（设备驱动程序）。

按文件的逻辑结构分为：流式文件（无结构操作系统文件）、记录式文件（有结构的数据库文件）。

流式文件：这是直接由字符序列（字符流）所构成的文件，故又称为流式文件。大量的源程序、可执行文件、库函数等，所采用的就是无结构的文件形式，即流式文件。其长度以字节为单位。对流式文件的访问，则是采用读/写指针来指出下一个要访问的字符。可以把流式文件看作是记录式文件的一个特例。在 UNIX 系统中，所有的文件都被看作是流式文件，即使是有结构文件，也被视为流式文件，系统不对文件进行格式处理。

记录式文件：由若干个记录所构成的文件，故又称为记录式文件，也叫数据库文件。

三、文件系统的功能

文件系统是操作系统用于明确存储设备（常见的是磁盘，也有基于 NAND Flash 的固态硬盘）或分区上的文件的方法和数据结构；即在存储设备上组织文件的方法。

文件系统应具备以下功能：文件管理、目录管理、文件空间管理、文件共享和保护、提供方便的接口。

四、文件的存取方式

（一）顺序存取

顺序存取是按照文件的逻辑地址顺序存取。

固定长记录的顺序存取是十分简单的。读操作总是读出上一次读出的文件的下一个记录，同时，自动让文件记录读指针推进,以指向下一次要读出的记录位置。如果文件是可读可写的，再设置一个文件记录指针，它总指向下一次要写入记录的存放位置，执行写操作时，将一个记录写到文件末端。允许对这种文件进行前跳或后退N（整数）个记录的操作。顺序存取主要用于磁带文件，但也适用于磁盘上的顺序文件。

可变长记录的顺序文件，每个记录的长度信息存放于记录前面一个单元中，它的存取操作分两步进行。读出时，根据读指针值先读出存放记录长度的单元。然后，得到当前记录长后再把当前记录一起写到指针指向的记录位置，同时，调整写指针值。由于顺序文件是顺序存取的，可采用成组和分解操作来加速文件的输入输出。

（二）直接存取（随机存取法）

很多应用场合要求以任意次序直接读写某个记录。例如，航空订票系统，把特定航班的所有信息用航班号作标识，存放在某物理块中，用户预订某航班时，需要直接将该航班的信息取出。直接存取方法便适合于这类应用，它通常用于磁盘文件。

为了实现直接存取，一个文件可以看作由顺序编号的物理块组成的，这些块常常划成等长，作为定位和存取的一个最小单位，如一块为 1024 字节、4096 字节，视系统和应用而定。于是用户可以请求读块 22，然后，写块 48，再读块 9 等等。直接存取文件对读或写块的次序没有限制。用户提供给操作系统的是相对块号，它是相对于文件开始位置的一个位移量，而绝对块号则由系统换算得到。

（三）索引存取

第三种类型的存取是基于索引文件的索引存取方法。由于文件中的记录不按它在文件中的位置，而按它的记录键来编址，所以，用户提供给操作系统记录键后就可查找到所需记录。通常记录按记录键的某种顺序存放，例如，按代表健的字母先后次序来排序。对于这种文件，除可采用按键存取外，也可以采用顺序存取或直接存取的方法。信息块的地址都可以通过查找记录键而换算出。实际的系统中，大都采用多级索引，以加速记录查找过程。

五、文件的结构

（一）文件的逻辑结构

1.流式文件

构成文件的基本单位是字符，流式文件是有序字符的集合，其长度为该文件所包含的字符个数，因此又称为字符流文件。流式文件无结构，管理简单，用户可以方便地对其进行操作。

2.记录式文件

构成文件的基本单位是记录，记录式文件是一组有序记录的集合。记录是一个具有特定意义的信息单位。

记录式文件可分为定长记录文件和变长记录文件两种。

（二）文件的物理结构

1. 顺序结构；

2. 链接结构；

3. 随机文件结构（索引结构、Hash 结构、索引顺序结构）。

六、目录和目录结构

（一）文件控制块和文件目录

文件控制块：在文件系统内部给每个文件唯一地设置一个文件控制块，它用于描述和控制文件的数据结构，与文件一一对应。

文件目录：文件控制块的有序集合。

目录项：文件目录中的一个文件控制块。

目录文件：完全由目录项构成的文件。

（二）目录结构

单级目录：DOS2.0 版本以下采用，全部文件都登记在同一目录中。优点是简单，缺点是无法防止重名或被删，安全保密性差，目前已淘汰。

二级目录：为每个用户单独建立一个目录，各管辖自己下属的文件。产生于多用户分时系统，DOS2.0 版本以上采用，文件主目录（MFD）的表目按用户分，每个用户有一个用户文件目录（UFD）。优点是允许重名，提高搜索速度，缺点是不太适合大量用户和大量文件的大系统。

树型目录：多级目录结构的一种形式，形同一棵倒置的树。产生于 UNIX 操作系统，已被现代操作系统广泛采用。目录与文件在一起，目录也做成文件。操作系统中每一名字由“全路径”确定唯一文件，有根/茎/叶（端头）层次关系概念。

非循环图目录：又称带链接的树型目录，访问同一文件（或目录）可以有多条路径。UNIX 的文件系统是树型结构，而且是带链接的树型结构。

路径名：在树型目录中，同一目录中的各个文件不能同名，但不同目录中的文件可以同名。例如树型图中目录/usr 中都有名字为 fp 的项，但是它们代表了不同的文件。文件路径名有两种表示形式：绝对路径名和相对路径名。

绝对路径名（全路径名）：是从根目录开始到达所要查找文件的路径。例如，在 UNIX 系统中，以“/”表示根目录。图中两个 fp 文件的绝对路径名是：（root）/usr/fp；（root）/usr/m1/prog/fp；

相对路径名：系统为每个用户设置一个当前目录（又称工作目录），访问某个文件时，就从当前目录开始向下顺次检索。例如，如图当前目录是 usr，则有：

（root）/usr/fp；（绝对路径名）

fp；（当前路径省略路径名）

（root）/usr/m1/prog/fp；（绝对路径名）

m1/prog/fp；（相对路径名）