二、处理器管理

2.1处理器

2.1.1处理器与寄存器

处理器系统：可分为单处理器和多处理器系统

共享存储（紧密耦合）多出力机系统和分布存储（松散耦合）多处理机系统

指令放在指令寄存器（IR）

程序计数器（PC）存放下一条要执行的指令的主存地址

算术逻辑单元（ALU）在完成执行之后将结果汇总到标志寄存器中

计算完成方式：数据🡪通用寄存器🡪功能单元🡪主存

寄存器分为用户可见寄存器、控制与状态寄存器两类

用户可见寄存器：使用机器语言（如汇编语言）编程时会使用，可以减少访问主存的次数，提高程序处理数据的效率

控制与状态寄存器：记录操作系统在控制程序时程序的动态行为、结果等

程序的执行时一个动态行为，故除了主存代码区与数据区外处理器现场还记录了程序执行的动态行为，为了描述程序动态执行的行为，操作系统建立了程序状态字

程序状态字PSW（ProgramStatus Word ）：用来控制指令执行顺序并保留和指示与程序有关的系统状态，主要作用是实现程序状态的保护和恢复

每个程序都有一个与其执行相关的PSW，每个处理器都设置一个PSW寄存器程序占有处理器执行，它的PSW将占有硬件PSW寄存器

2.1.2指令与处理器模式

机器指令：计算机系统执行的基本指令，是处理器执行的重要基本单位

指令由一个或多个字节组成，包括操作码字段、一个或多个操作数地址字段，以及一些表征处理器状态的状态字和特征码，用以完成各种算数逻辑运算，数据传输，控制流跳转

指令的执行：取指令🡪解码🡪执行（取数据🡪运算🡪返回结果）

配置操作系统后，计算机软件系统分为两类：

（1）操作系统内核程序：通常被赋予较高特权，可以使用全部机器指令

（2）运行于从操作系统环境的普通应用程序：拥有较低的使用权限，只能使用资源管理命令之外的指令系统子集

特权指令：只能提供给操作系统的核心程序使用的指令，如启动I/O设备、设置时钟、控制中断屏蔽位、清主存、建立存储键，加载PSW等

非特权指令：在内核态和用户态下都能使用的指令，即操作系统程序能够执行全部机器指令，应用程序只能使用非特权指令

处理器状态：管理状态（特权状态系统模式、特态或管态）和用户状态（目标状态、用户模式、常态或目态）

内核态：操作系统管理程序运行时所处的状态，可认为处理器正在运行可信系统软件，全部机器指令都被允许在处理器上执行，程序可访问所有主存单元和系统资源，并具有改变处理器状态的能力

用户态： 正在运行非可信的应用程序，此时无法执行特权指令，且访问仅限于当前正在处理器上执行程序梭子啊的地址空间

大多数计算机系统的处理器由四种状态：0~3级，其中0级权限最高，3级权限最低

只有中断才会使吹起从用户态向内核态转换，包含三类通过广义的中断机制触发的情况：

（1）程序运行时产生异部中断时间（如I/O操作完成），运行程序被中断，转向中断处理程序处理

（2）程序主动同步请求操作系统服务，执行系统该调用，触发了系统异常，也成为自愿性中断

（3）程序运行时，执行指令出现问题从而产生同步异常事件（如用户态执行特权指令），运行程序被打断，转向异常处理程序工作，这属于程序性中断

用户栈：操作系统在用户进程空间中开辟的一块区域，用于保存应用程序的子程序（函数）见相互调用的参数、返回值、返回点以及子程序的局部变量

核心栈（系统栈、核心站）：主存中属于操作西戎内核空间的一块区域，用途包括：

（1）保存终端现场，对于嵌套中断，将被中断程序的线程信息一次压入核心栈，中断返回时逆序弹出

（2）保存操作系统程序（函数）见相互调用的参数、返回值、返回点以及程序局部变量

每个进程被创建时捆绑一个核心栈，具有可读、可写、不可执行的属性

进程由用户栈和核心栈，单硬件栈指针仅有一个

2.2中断

2.2.1概念

中断：程序执行过程中，遇到急需处理的事件时，暂时中止CPU上现行程序的运行，转去执行相应的事件处理程序，待处理完成后再返回原程序被中断处或调度其他程序执行的过程

中断：CPU对系统内或系统外异步事件的响应

异步事件：无一定时序关系的随机发生的事件

中断的作用：CPU的利用率、提高实时性

不同计算机系统通常配备有不同的终端装置，但在发生中断之后都能改变处理器内操作的执行顺序🡪中断是现代操作系统实现并发性的基础之一，操作系统内核时终端驱动的软件

广义中断：狭义中断+异常+系统异常

狭义中断：很多情况下简称为中断，来源于处理器外部的异步中断时间

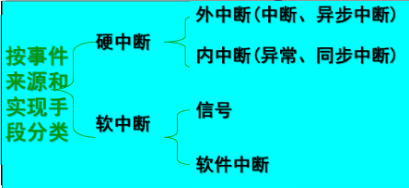
异常：正在运行的程序鱿鱼执行指令出现特殊情况时，由CPU控制器引向异常处理电路的同步中断事件

系统异常：又称为系统调用，是由于运行程序调用触发系统调用的陷入指令而引起的预设服务过程系统异常是由执行指令引起的，属于异常的一类，只不过是被引向CPU控制单元预设别的正常处理电路

中断与异常的区别

中断：由与现行指令无关的中断信号触发的（异步的），且中断的发生与CPU处在用户模式或内核模式无关，在两条机器指令之间才可响应中断，一般来说，中断处理程序提供的服务不是为当前进程所需的

异常：由处理器正在执行现行指令而引起的，一条指令执行期间允许响应异常，异常处理程序提供的服务是为当前进程所用的异常包括很多方面，有出错（fault）也有陷入（trap）



2.2.2中断源

中断源：引起中断的事件，即中断请求信号的来源

硬件故障中断：由计算机系统的关键性硬件故障导致的，如电源故障、主存故障、奇偶校验中断等，排除这种故障需要人工干预，如复位、设置或替换中断处理程序的工作：

（1）保护现场，停止设备工作，停止处理器运行

（2）向操作员报告故障信息

（3）并对故障造成的破坏进行估计和恢复

程序性中断事件：由处理器执行机器指令出错或异常引起的，其处理原则如下：

（1）除数为零、操作数溢出等算术异常：简单处理，报告用户，由用户判断计算机结果是否可信，也可以由用户编写中断续元程序，操作系统把处理权交给用户程序

（2）非法指令、用户态使用特权指令、地址越界、非法存取等指令异常：终止进程

（3）终止进程指令：终止进程

（4）虚拟地址异常：发现了不在实际主存的地址，需要调整主存后重新执行指令

自愿性中断事件：又称系统调用，是由程序执行陷入指令引起的，表示当前运行程序对操作系统功能的调用

陷入指令：一种为操作系统专门设计的机器指令，包括操作码和功能号两部分，前者表示此指令是陷入指令，后者则表示具体的系统调用服务要求，共性处理流程：

（1）程序执行陷人指令，并通过适当方式指明系统调用号

（2）通过系统陷阱机制进入系统调用处理程序，现场信息被保护到核心栈，按功能

号实现跳转

（3）通过系统调用人口表找到相应系统调用服务例程的入口地址

（4）执行系统调用服务例程，结束后返回系统调用的下一条指令

I/O中断事件：来自设备执行输入/输出的过程，I/O 中断的处理原则如下：

（1）I/O操作正常结束：本进程设置为就绪态，让等待传输的下一个进程占有设备或通道并启动数据传输

（2）I/O操作发生故障：先向设备发送命令索取状态字，分析产生故障的确切原因再进行复执或请求人工干预

（3）I/O操作发生异常：分析情况，采取相应措施并向操作员报告，如通知操作员换卷、装纸等

外部中断事件：来自CPU现场之外的信号，其处理原则如下

（1）时钟中断：计时

（2）间隔时钟中断：时间片到，被中断进程就要让出处理器进入就绪队列

（3）设备报告与结束中断：调整设备表，例如打印机连接到计算机系统后，操作系统就要进行中断并调整资源表来对该设备进行管理

（4）键盘/鼠标信号中断：根据信号做出相应的反应

（5）关机/重启动中断：写回文件，停止设备与 CPU

2.2.3 中断系统

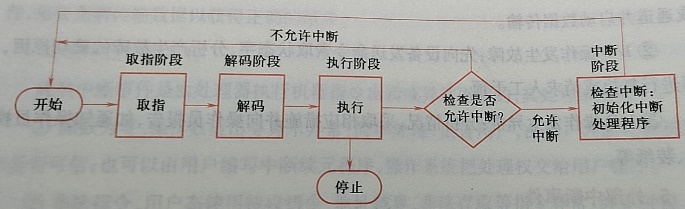
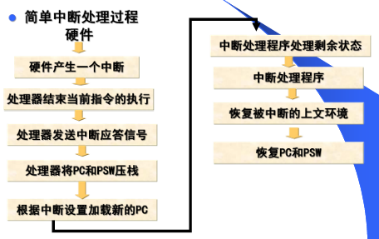
中断系统：计算机系统中响应和处理中断的系统，包括硬件子系统和软件子系统两部分

硬件子系统主要完成中断响应的工作

软件子系统主要完成中断处理的过程

中断系统是操作系统实现的基础支撑，也是在操作系统设计中使用软硬件协同方法的一个典型例子

中断响应



硬件实现的中断装置在计算机系统中通常可以将其分为以下三类：

1. 对于来自处理器外的异步中断事件，通常设置中断控制器

（2）对于请求操作系统服务的自愿性中断事件，通常设置系统陷阱

（3）对于处理器执行指令引发的程序性中断事件，通常设置陷阱

响应中断的典型响应过程如下：

（1）发现中断源，判断这些中断是否应该屏蔽，当存在有多个需要响应的中断源时，

根据规定的优先级选择一个提出中断请求

（2）保存当前程序的PSW和PC到核心栈，准备中断当前程序的执行（中断处理结

束后，可以根据该PSW 和PC 返回中断点继续运行）

（3）将处理器状态从用户态切换至内核态，调出响应中断处理程序的PSW和PC

转向操作系统的中断处理程序开始工作

操作系统的中断处理程序处理完毕后，需要调用一条中断返回指令，将内核态改为用户态，然后返回下一个被选中待执行的进程考虑到操作系统的性能设计需要或处理器调度的现场情况，被选中的进程可以是别的进程，也可以是刚刚被中断的进程

中断处理程序：操作系统中处理中断事件的控制程序，其主要工作是处理中断以及恢复正常工作

处理中断包括下列4项主要工作：

（1）保护处理器现场出于实现成本的考虑，模式切换时核心栈中只保存了PSW和PC，未保留整个处理器现场（即CPU 中所有寄存器和高级别 Cache 的内容），这些现场信息应由中断处理程序来完成保存

（2）分析被中断进程的 PSW 中断码字段，识别中断源

（3）根据中断源对中断事件进行具体处理

（4）恢复正常操作

恢复正常操作包括两种情况：

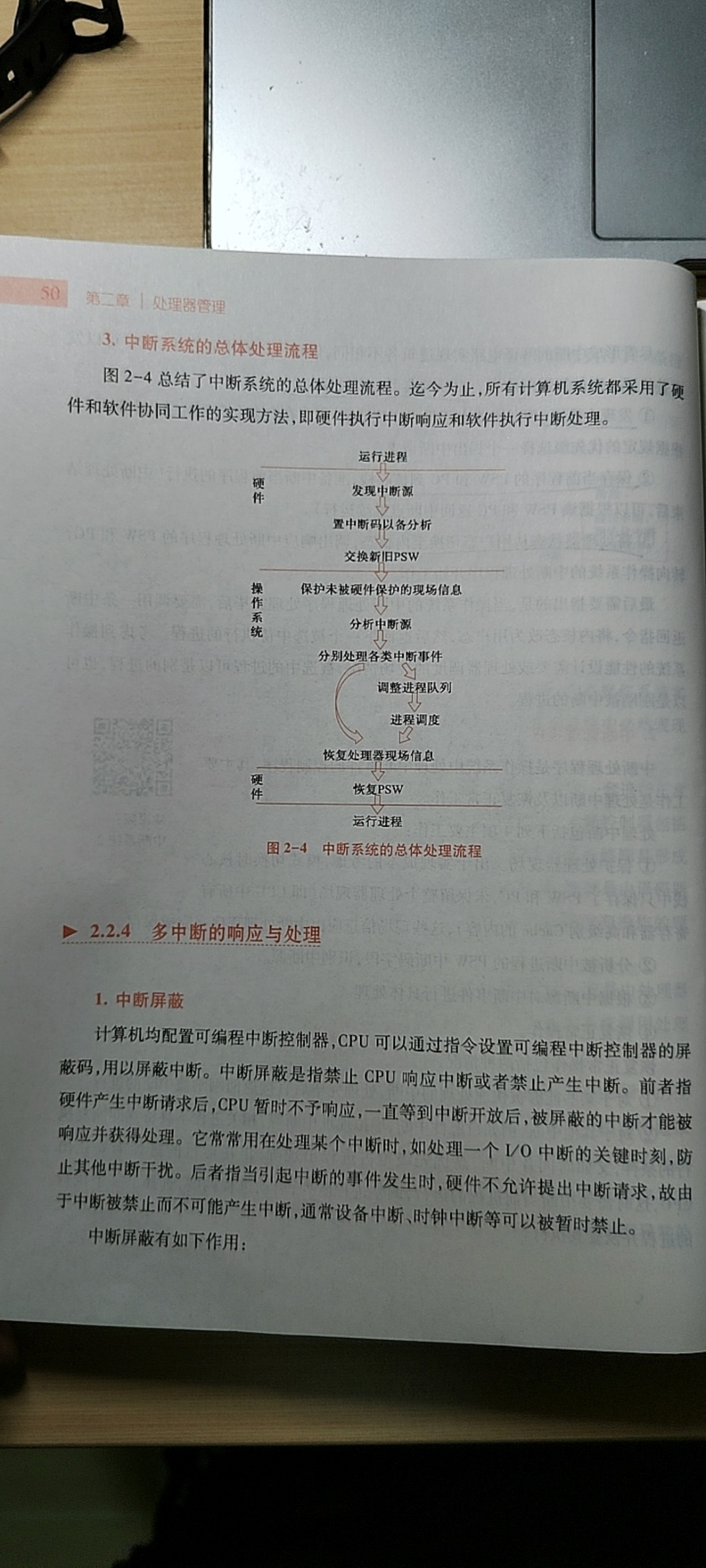
（1）对于某些能够被快速处理的中断，处理完毕后直接返回刚刚被中断的进程

（2）对于其他中断，例如自愿性中断、虚拟地址中断等，相关进程不具备立即恢复执

行的条件，又如时间片中断、高优先级抢占进程出现等，正在运行的进程需要让出

CPU，这时需要中断当前进程的运行，调整进程队列，启动进程调度，选择下一个执行

的进程并恢复其执行



计算机系统基本都采用了硬件和软件协同工作的实现方法，即硬件执行中断响应和软件执行中断处理

2.2.4 多中断的响应与处理

中断屏蔽：计算机均配置可编程中断控制器，CPU可以通过指令设置可编程中断控制器的屏蔽码，用以屏蔽中断中断屏蔽是指禁止CPU响应中断或者禁止产生中断前者指硬件产生中断请求后，CPU暂时不予响应，一直等到中断开放后，被屏蔽的中断才能被响应并获得处理

中断屏蔽有如下作用：

（1）延迟或禁止某些中断的响应，系统程序执行过程中不希望产生干扰事件，以免

共享数据结构受到破坏

（2）协调中断响应与中断处理的关系，确保高优先级中断可以打断低优先级中断

反之却不能

（3）防止同级中断相互干扰，在处理某优先级中断事件时必须屏蔽该级中断，以免

造成混乱

中断优先级：以不发生中断丢失为前提，将紧迫程度相当的中断源归为同一级别，紧迫程度差别大的中断源归于不同级别，级别高的中断有优先获得响应的权利中断装置所预设的中断响应顺序称为中断优先级

中断优先级从高到低：硬件故障中断、自愿性中断、程序性中断外部中断和I/O中断

多重中断事件处理：在计算机系统运行过程中可能同时出现多个中断，或者前一个中断尚未处理完又发生了新的中断，于是CPU就有可能暂停正在运行的中断处理程序，转而执行新的中断处理程序，这就叫作多重中断事件处理或中断嵌套处理

2.3进程管理

2.3.1进程及其状态

顺序程序的特征：

（1）程序执行的顺序性

（2）程序执行的封闭性：独占资源，执行过程中不受外界影响

（3）程序执行结果的确定性：程序结果的可再现性

（3）程序运行结果与程序执行速度无关，只要初始状态相同，结果应相同

前趋图：一个有向无循环图(DAG)，用于描述程序段或进程之间执行的前后关系

结点：描述一个程序段或进程，或一条语句

有向边：结点之间的偏序或前趋关系“→”

Pi→Pj：Pi必须在Pj开始之前完成，则Pi是Pj的直接前趋，Pj是Pi的直接后继

初始结点：没有前趋的结点

终止结点：没有后继的结点

前趋图中不能出现循环

并发环境：一定时间内，物理机器上有两个或两个以上的程序同处于开始运行但尚未结束的状态，并且次序不是事先确定的

并发程序的特征：

（1）并发环境下程序的执行是间断性：执行--停--执行

（2）失去封闭性：系统资源被多个进程使用，程序不能独占系统资源

（3）执行结果的不可再现性，并发程序执行的结果与其执行的相对速度有关，是不确定的

（4）独立性和制约性：独立的相对速度、起始时间，进程之间可相互作用（相互制约）

进程是操作系统中最基本、最重要的概念，是在多道程序系统出现后，为了刻画系统内部动态状况、描述运行程序活动规律而引进的新概念

进程的定义：正在运行的程序的抽象

（1）进程是程序的一次执行

（2）进程是一个程序及其数据在处理机上顺序执行时发生活动

（3）进程是程序在一个数据集合上的运行过程，是系统进资源分配和调度的一个独立单位

从原理角度看，进程是支持程序执行的一种系统进程及其制，它对处理器上运行程序的活动规律进行抽象

从实现角度看，进程是一种数据结构，用来准确地刻画运行程序的状态和系统的动态变化状况进程概念既能描述程序的并发执行，又能共享系统的资源，所以现代操作系统都围绕着进程概念来设计和构造

进程刻画了运行程序之间的并发性在多道程序环境下，各运行程序在处理器上交替运行，运行程序的任意两条指令之间都可能发生中断事件而引发程序切换因此运行程序的执行可能不是连续的，而是走走停停的这种运行程序的并发性是操作系统的重要特征，操作系统必须保证各无关的运行程序的封闭性与计算结果的唯一性

进程🡪动态 程序🡪静态

“程序”自身只是计算任务的指令和数据的描述，这种静态概念无法刻画程序的并发性，系统要寻找能够描述程序并发执行过程的概念，这就是进程

可再入程序：能够被多个程序同时调用的程序，可再入程序是纯代码，在执行过程中不能被修改，由调用它的各应用程序提供数据工作区

进程是既能描述程序的并发执行，又能共享系统资源的一个基本单位

进程是具有独立功能的程序在某个数据集合上的一次运行活动，也是操作系统进行资源分配和保护的基本单位

从操作系统实现者的观点出发，一个进程包括如下5个实体部分：

（1）(操作系统管理运行程序的)数据结构P;

（2）(运行程序的)主存或虚拟主存代码C;

（3）(运行程序的)主存或虚拟主存数据 D;

（4）(运行程序的)通用寄存器信息 R;

（5）(操作系统控制程序执行的)程序状态字信息 PSW

2.进程的状态

进程在执行过程中要定义运行、就绪和等待三种进程状态

（1）运行（nunning）态：进程占有处理器正在运行的状态

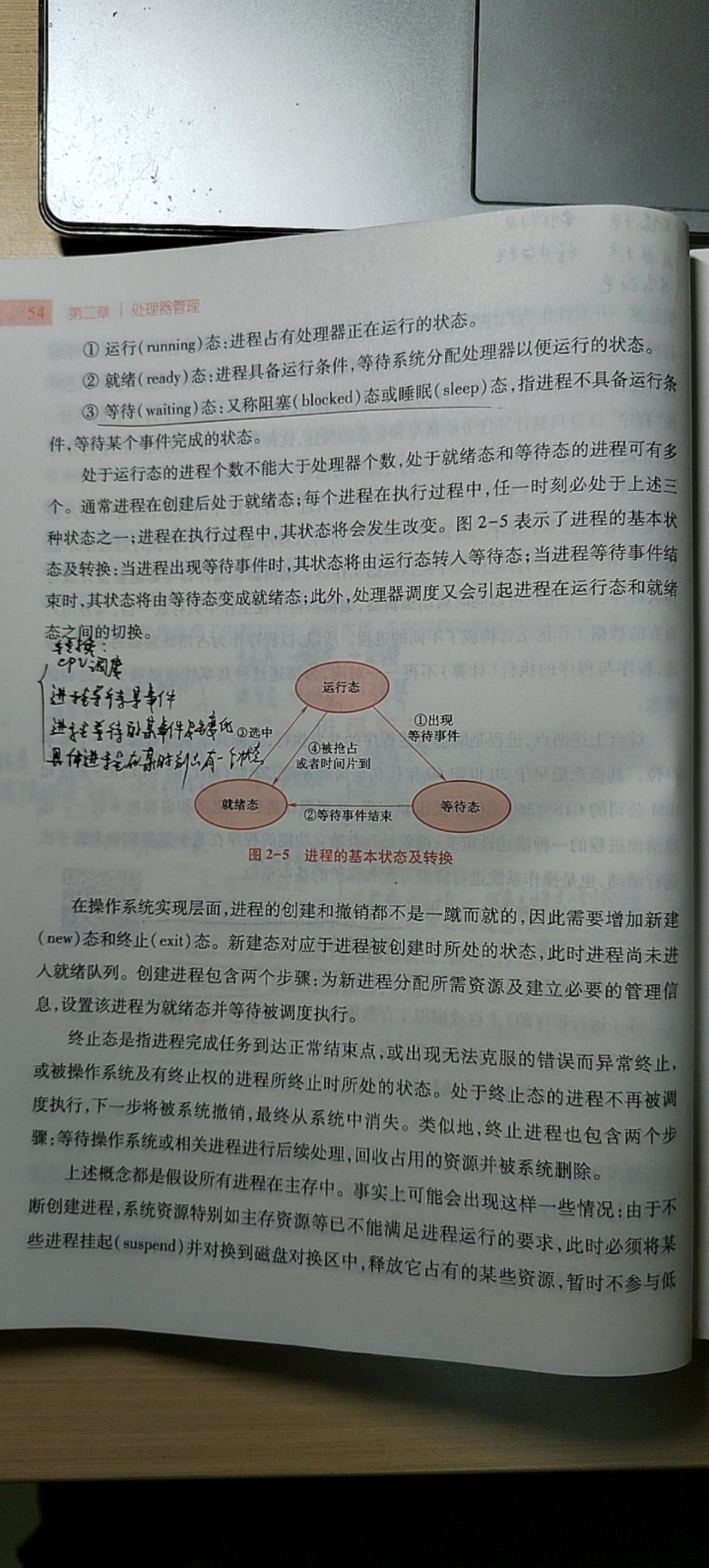
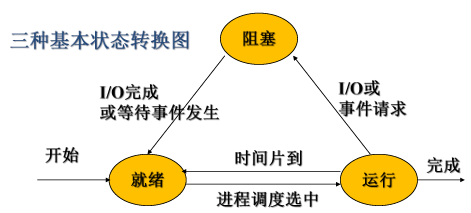
（2）就绪（ready）态：进程具备运行条件，等待系统分配处理器以便运行的状态

（3）等待（ waiing）态：又称阻塞（blocked）态或睡眠（seep）态，指进程不具备运行条件，等待某个事件完成的状态

处于运行态的进程个数不能大于处理器个数，处于就绪态和等待态的进程可有多个

进程在创建后处于就绪态，每个进程在执行过程中，任一时刻必处于三种状态之一，进程在执行过程中，其状态将会发生改变

当进程出现等待事件时，其状态将由运行态转入等待态，当进程等待事件结束时，其状态将由等待态变成就绪态，此外，处理器调度又会引起进程在运行态和就绪态之间切换

进程之间的转换并非都可逆

进程之间的转换并非都主动

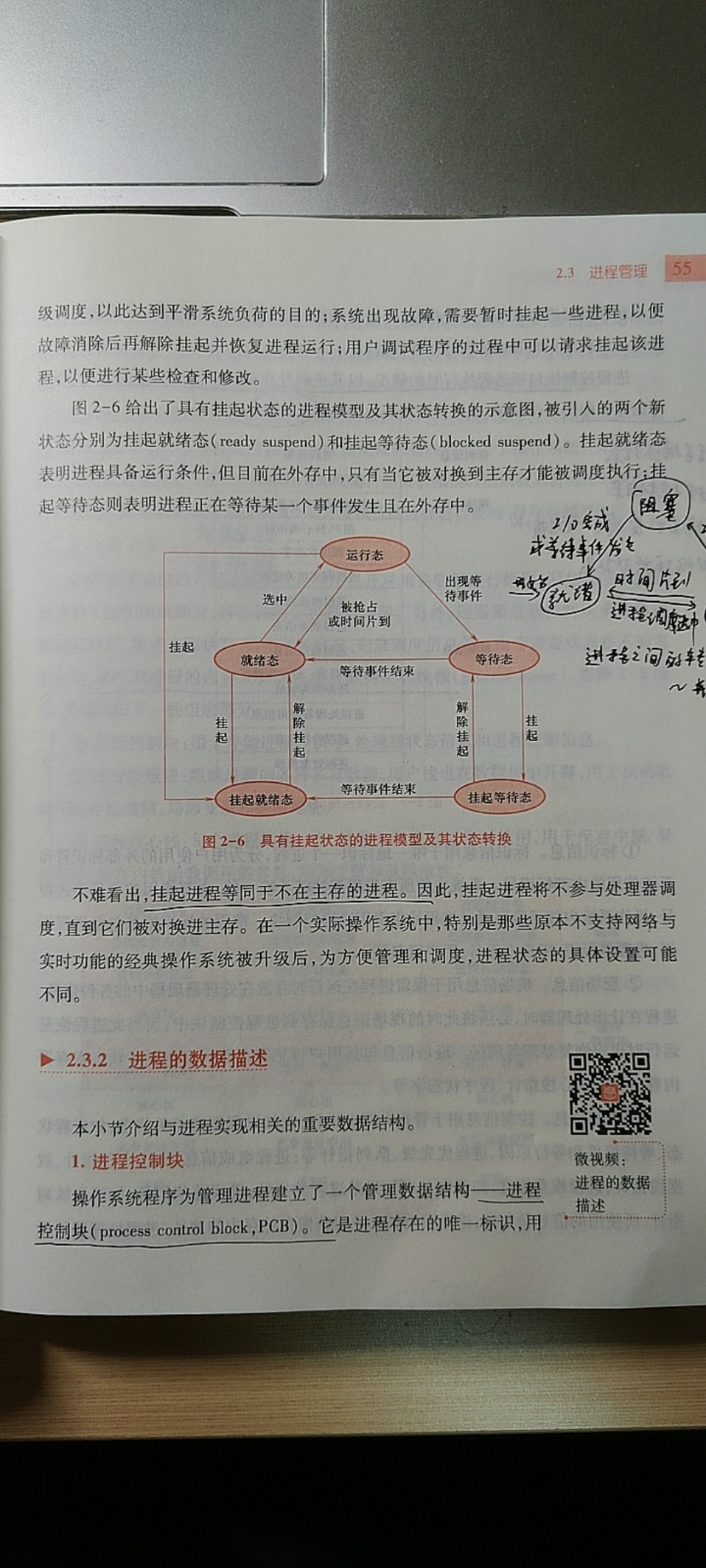
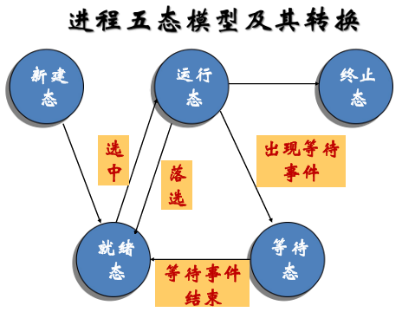
引起进程状态转换的原因

（1）CPU调度

（2）进程在等待某一事件

（3）进程等待的某一事件发生变化

（4）具体进程在某一时刻只有一个状态



挂起：

（1）系统资源特别如主存资源等已不能满足进程运行的要求，此时必须将某些进程挂起（suspemd）并对换到磁盘对换区中，释放它占有的某些资源，暂时不参与低级调度，以此达到平滑系统负荷的目的

（2）系统出现故障，需要暂时挂起一些进程，以便故障消除后再解除挂起并恢复进程运行，用户调试程序的过程中可以请求挂起该进程，以便进行某些检查和修改

挂起进程等同于不在主存的进程因此，挂起进程将不参与处理器调度，直到它们被对换进主存

挂起进程具有如下特征：

（1）该进程不能立即被执行

（2）挂起进程可能会等待事件，但所等待事件是独立于挂起条件的，事件结束并不能导致进程具备执行条件

（3）进程进入挂起状态是由于操作系统、父进程或进程本身阻止它的运行

（4）结束进程挂起状态的命令只能通过操作系统或父进程发出

2.3.2 进程的数据描述

本小节介绍与进程实现相关的重要数据结构

1.进程控制块

操作系统程序为管理进程建立了一个管理数据结构--进程描述控制块（process control block，PCB）它是进程存在的唯一标识，用来记录和刻画进程状态及环境信息，是进程动态特征的汇集，也是操作系统掌握进程的唯一资料结构和管理进程的主要依据

进程控制块应包含以下三类信息

（1）标识信息：标识信息用于唯一地标识一个进程，分为用户使用的外部标识符和系统使用的内部标识号

（2）现场信息：用于保留进程在运行时存放在处理器现场中的各种信息

（3）控制信息：用于管理和调度进程，包括进程调度相关信息，进程组成信息，进程族系信息，进程通信信息，进程处理器使用信息，进程特权信息，进程资源清单

或者以下几种：

进程描述信息、进程控制信息、所拥有的资源和使用情息、CPU现场保护信息

进程控制块是操作系统中最为重要的数据结构，它包含管理进程所需要的全部信

息，实际上定义了一个进程的当前状态，其使用权和修改权均属于操作系统

PCB表组织方式：

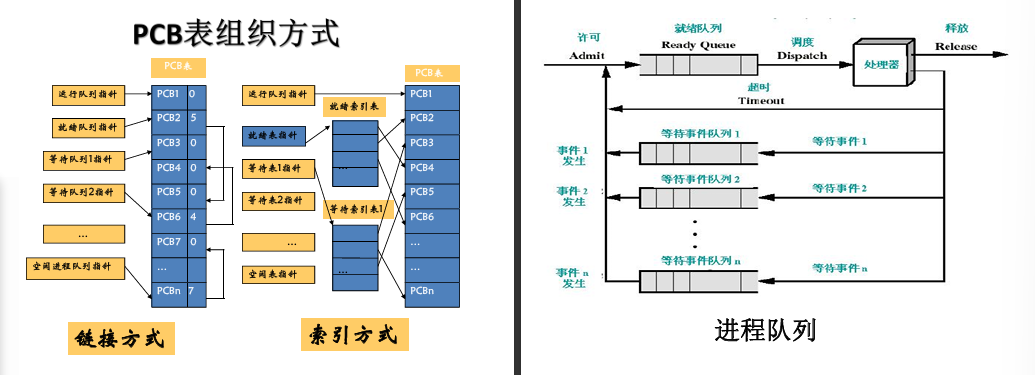
（1）PCB表：系统把所有PCB组织在一起，并把它们放在内存的固定区域，构成PCB表

（2）PCB表的大小决定了系统中最多可同时存在的进程个数，称为系统的并发度

（3）链接结构：同一状态进程的PCB组成一个链表，不同状态对应多个不同的链表🡪就绪链表、阻塞链表

（4）索引结构：对具有相同状态的进程，分别设置各自的PCB索引表，表明PCB在PCB表中的地址

（5）进程队列：运行队列、就绪队列、等待队列



程序段+数据段

进程映像（processimage）：某时刻进程的内容及其状态集合称为，其包括以下一些组成部分：

（1）进程控制块：用于存储进程标识号、处理器状态信息和进程控制信息

（2）进程数据块：组成进程的各种私有数据，用户栈也在数据块中开辟，用于在函数调用时存放栈帧、局部变量和返回地址

（3）进程核心栈：每个进程捆绑一个，进程在内核态工作时使用，用于保存中断/异

常现场，保存内核函数调用的参数、局部变量和返回地址

进程上下文（processcontext）：进程物理实体和支持进程运行的环境合称为进程上下文

进程在其当前上下文中运行，当系统调度新进程占有处理器时，新老进程随之发生上下文

切换

进程上下文由以下三部分组成

（1）用户级上下文（user levelcontext）：由程序块（可执行的机器指令序列）、数据块

（进程可访问的信息）、共享主存区（进程通信使用的主存区）、用户栈（存放函数调用

过程中的信息）组成，它们占用进程的虚存空间

（2）寄存器上下文（register context）：由处理器状态寄存器（进程当前状态）、指令计

数器（下一条执行的指令地址）、栈指针（指向用户栈或核心栈当前地址）、通用寄存器

等组成

（3）系统级上下文（system level context）：由进程控制块（进程的状态）、主存管理信

息（进程页表或段表）核心栈（进程内核态运行时的工作区）等操作系统管理进程所

需要的信息组成

2.3.3 进程管理的实现

队列管理程序：操作系统将进程组织成若干个队列进行管理

进程最终会在处理器上通过执行一条系统调用进入结束终止的完成状态

队列管理程序是操作系统实现进程管理的核心模块

操作系统建立多个进程队列（包括就绪队列和等待队列），这些队列按照需求组织成先进先出的队列或优先队列（根据优先数来排列的队列，人队时将根据优先级找到相应位置插入）

队列中进程可以通过进程控制块的队列指引元采用单/双指引元或索引连接

进程控制程序：包括进程的创建、进程的撤销、进程的阻塞、进程的唤醒、进程的挂

起、进程的激活，以及进程特权的修改等操作，它们往往被抽象成一组系统调用，在实现上是一个处理的流程这些控制和管理功能由操作系统中的原语实现

原语：在管态下执行、完成系统特定功能的过程

原语和机器指令类似，其特点是执行过程中不允许被中断，是一个不可分割的基本单位，原语的执行是顺序的而不可能是并发的

进程控制是进程管理的重要组成部分，需要对操作系统的核心数据结构进行操作

原语的实行方法：以系统调用方式提供原语接口，且采用屏蔽中断的方式来实现原语功能，以保证原语操作不被打断

进程的创建

操作系统启动进行初始化时，会创建承担系统资源分配和控制管理的一些系统进

程同时，还会创建一个所有用户进程的祖先，其他用户进程是在用户作业（程序）被

提交与选中时创建的

操作系统把具有“创建”与“被创建”关系的进程称为父子进程，其中生成进程称为父进程，被生成进程称为子进程父进程可以创建多个子进程，从而形成树状族系关系

进程创建过程如下

（1）在主进程表中增加一项，并从PCB池中取一个空白PCB

（2）为新进程的进程映像分配地址空间（传递环境变量，构造共享地址空间）

（3）为新进程分配资源除内存空间外，还有其他各种资源

（4）查找辅存，找到进程正文段并装到正文区

（5）初始化进程控制块，为新进程分配进程标识符，初始化PSW

（6）加入就绪进程队列，或直接将进程投入运行

（7）通知操作系统的某些模块

进程的创建

Linux fork()创建子进程但父子进程不共享内容

Linux clone()允许定义父子进程共享的内容

Linux中认为线程就是共享上下文的进程

fork调用结果

仅仅被调用一次，却能够返回两次，可能有三种不同的返回值：

在父进程中，fork返回新创建子进程的进程ID

在子进程中，fork返回0

如果出现错误，fork返回一个负值

进程的撤销

进程完成特定工作或出现严重错误后，必须被撒销

进程撤销可分为正常撤销和非正常撤销

进程运行过程中出现借误或异常进程撤销的主要原因有进程运行结束进程执行非法令

进程将调用撤销原语来终止自己或子进程具体步骤：

（1）根据撤销进程的标识号，从相应队列中查找并移除它

（2）将此进程所拥有的资源归还给父进程或操作系统，若此进程拥有子进程，先撤销其所有子进程，以防止它们脱离控制

（3）回收进程控制块，并将其归还至进程控制块池

（4）转向进程调度程序工作

进程的阻塞和唤醒

进程阻塞是指使进程让出处理器转而等待一个事件

进程通常调用阻塞原语来阻塞自己，所以阻塞是进程的自主行为，是一个同步事件当等待事件完成时会产生一个中断激活操作系统

进程阻塞步骤如下：

（1）停止进程执行，将现场信息保存到进程控制块

（2）修改进程控制块的有关内容，如进程状态由运行态改为等待态等，并将状态已修改的进程移入相应事件的等待队列

（3）转向进程调度程序，调度其他进程运行

进程唤醒步骤如下

（1）从相应的等待队列中移出进程

（2）修改进程控制块的有关内容，如进程状态改为就绪态，并将进程移入就绪队列

（3）若被唤醒的进程比当前运行进程的优先级高，重新设置调度标志

进程的挂起和激活

当出现引起挂起的事件时，系统或进程会将指定进程或处于等待态的进程挂起

其执行过程大致是：

（1）检查要被挂起的进程状态，若处于活动就绪态，就修改为挂起就绪态，若处于等待态，就修改为挂起等待态

（2）被挂起进程的进程控制块的非常驻部分要交换到磁盘对换区

当系统资源尤其是主存资源充裕或请求激活指定进程时，系统或相关进程会将指定进程激活，具体工作是：

（1）将被挂起进程的进程控制块的非常驻部分调入主存，修改其状态，挂起等待态改为等待态，挂起就绪态改为就绪态

（2）将进程移入相应队列中

挂起操作既可由进程自己也可由其他进程调用，但是激活操作只能由其他过程调用

2.3.4进程切换与模式切换

2.3.4 进程切换与模式切换

中断是激活操作系统的唯一手段，它暂停当前运行进程的执行，将处理器切换至内核态，内核获得处理器控制权之后，如果需要就可以实现进程切换

进程切换必定在内核态而非用户态发生

进程切换实现步骤如下

（1）保存被中断进程的处理器现场信息

（2）修改被中断进程的进程控制块的有关信息

（3）把被中断进程的进程控制块加入到有关队列

（4）选择占用处理器运行的另一个进程

（5）修改被选中进程的进程控制块的有关信息

（6）设置被选中进程的地址空间，恢复存储管理信息

（7）根据被选中进程的上下文信息来恢复处理器现场

调度和切换时机：请求调度的事件发生后，就会运行低级调度程序，低级调度程序选中新的就绪进程后，就会进行上下文切换。实际上，由于种种原因，调度和切换并不一定能一气呵成

通常的做法：由内核置上请求调度标志，延迟到上述工作完成后再进行调度和进程上下文切换

处理器模式切换：当中断发生时，暂时中断正在执行的用户进程，把进程从用户状态切换到内核状态，去执行操作系统例行程序以获得服务，这就是一次模式切换

内核在被中断了的进程的上下文中对这个中断事件作处理，即使该中断可能不是此进程引起的

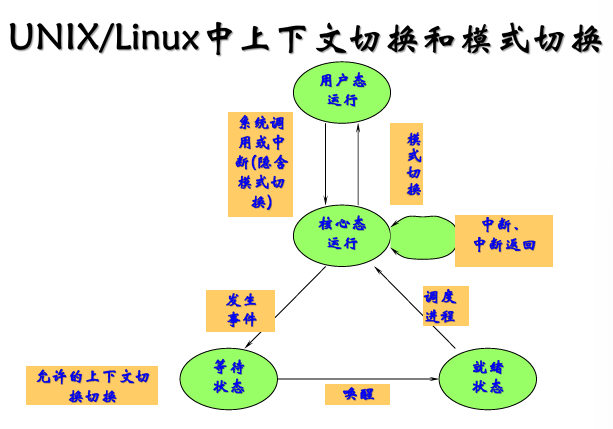
模式切换的步骤

（1）保存被中断进程的处理器现场信息

（2）处理器从用户态切换到核心态，以便执行服务程序或中断处理程序

（3）如果处理中断，可根据规定的中断级设置中断屏蔽位

（4）根据系统调用号或中断号，从系统调用表或中断入口表找到服务程序或中断处理程序地址



2.3.5 操作系统的执行模式

略

2.4多线程技术

2.4.1 多线程环境概述

单线程结构进程给并发程序设计效率带来问题：

（1）进程切换开销大

（2）进程通信代价大

（3）进程之间的并发性粒度较粗，并发度不高

（4）不适合并行计算和分布并行计算的要求

（4）不适合客户/服务器计算的要求。

线程：一个进程内的基本调度单位

思想：把进程的两项功能--“独立分配资源”与“被调度分派执行”分离开来，进程作为系统资源分配和保护的独立单位，不需要频繁地切换，线程作为系统调度和分派的基本单位，能轻

装运行，会被频繁地调度和切换

多线程环境中的进程可定义为：操作系统中除处理器以外的资源分配和保护的基本单位它有一个独立的虚拟地址空间，用来容纳进程映像，并以进程为勇位对各种资源(如文件、O 设备等)实施保护

线程是进程中能够并发执行的实体，是进程的组成部分，也是处理器调度和分配的基本单位进程包含多个线程（控制流），这些线程共享进程所获得的主存空间和资源，可以为完成某一项任务而协同工作

线程的组成：

（1）有一个唯一的标识符

（2）有一组表示处理机状态的寄存器

（3）有两个堆栈，分别用于用户态执行和核心态执行

（3）有一个独立的程序计数器

线程的适用范围

优点:

（1）多任务处理时减少切换时间

（2）线程创建、结束所需的雄的系统开销较小

（3）简化程序的结构、提高执行效率

典型的应用：服务器中的文件管理和通信控制、前后台处理、异步处理

线程具有以下特性：

（1）线程唯一标识符及线程状态信息

（2）线程是一条执行路径，有独立的程序计数器

（3）线程有执行栈和存放局部变量的私有存储空间

（4）可访问所属进程的主存和资源，并与该进程中的其他线程共享这些资源

进程可以分为两部分：资源集合和线程集合

进程要支撑线程运行，为线程提供虚拟地址空间和各种资源

线程封装执行信息，包括对状态信息、寄存器、执行栈(用户栈与核心栈)和局部变量、过程调用参数、返回值等私有部分的管理

线程也有生命周期，由于同一进程中的多线程会竞争处理器资源，故存在运行、就绪、等待和终止等状态，其状态转换与进程类似

由于线程不是资源拥有单位，挂起状态对于线程是没有意义的

一个进程中的若干线程可以有多种组织方式：

（1）调度员/工作者模式：进程中的一个线程担任调度员，接收和处理工作请求，其他线程是工作者线程，由调度员线程分配任务并唤醒工作者线程

（2）组模式：进程中的各个线程看作同一组，都可以取得并处理工作请求，不存在调度员线程有时每个线程被设计成专门执行特定的任务，同时建立相应的任务队列

（3）流水线模式：线程排成某个次序，第一个线程所产生的数据传送给下一个线程进行处理，以此类推；数据按照排定次序由线程依次传递以完成被请求的任务

五个基本操作：派生、阻塞、激活、调度、结束

进程与线程的比较：

（1）进程是资源分配的基本单位，处理机的调度单位，有完整的虚拟地址

（2）线程与资源无关，与进程中的其他线程共享资源

（3）同一进程中的线程共享同一地址空间。不同的进程有不同的虚拟地址空间

（4）线程由相关堆栈寄存器和线程控制表组成

（5）线程切换式系统开销较小，进程切换时涉及资源指针保存和地址空间变化

（6）进程的调度由OS内核完成

2.4.2内核级线程、用户级线程

基本类型：用户级线程ULT(如Java,Informix)、内核级线程KLT(如OS/2)、混合式线程(如，Solaris)

线程的执行：

用户级线程系统：以进程为单位调度

核心级线程系统：以线程为单位调度，通过提供API函数进行

内核级线程：

（1）线程管理工作由操作系统内核完成，并提供线程应用程序接口来使用线程

（2）当任务提交给操作系统执行时，内核为其创建进程和个基线程

（3）线程在执行过程中可通过内核的创建线程原语来创建其他线程，应用程序的所有线程均在一个进程中获得支持

（4）内核需要为进程及进程中的单个线程维护现场信息，所以应在内核空间中建立和维护进程控制块及线程控制块(threadconnbock,TCB)。

（5）处理器调度以线程为单位，即内核的处理器调度程序直接选中某个就绪的线程

内核级线程优点：

（1）在多处理器上能够同时调度同一进程的多线程并行执行

（2）若进程的一个线程被阻塞，内核能够调度同一进程的其他线程占有处理器运行，也

可以运行其他进程的线程

（3）由于内核级线程只有很小的数据结构和堆栈，切换速度快

（4）内核自身也可用多线程技术实现,从而提高系统的执行效率

内核级线程缺点：线程在用户态运行，而线程的调度和管理在内核实现，在同一进程中控制权从一个线程传送到另一个线程时，需要进行用户态一内核态一用户态模式切换，系统开销较大

用户级线程：

（1）由应用程序完成所有线程的管理

（2）通过线程库(用户空间)

（3）一组管理线程的过程

（4）核心不知道线程的存在

（5）线程切换不需要核心态特权

（6）调度是应用特定的

线程库：一个用户级线程管理的例行程序包，主要功能有线程的创建、调度、管理等

在用户级线程下，线程库是线程的运行支撑环境

用户级线程优点是：

（1）线程切换无须使用内核特权方式，所有线程管理的数据结构均在用户空间中，可以节省模式切换开销和内核的宝贵资源

（2）允许进程按照应用的特定需要选择调度算法，且线程库的线程调度算法与操作系统的低级调度算法无关

（3）能够运行在任何操作系统上,内核无须做任何改变

用户级线程缺点：由于大多数系统调用是阻塞型的，一个用户级线程的阻塞将引起整个进程阻塞,使用户级线程不能发挥多处理器并行工作的优点

Jackeling技术：修改操作系统的系统调用，一旦发现线程要进入等待状态，就调用Jacketing程序,让操作系统将处理器控制权切换到用户进程的用户级线程调度程序，由用户级线程调度程序去选择下一个线程继续占有处理器运行

2.4.3多线程实行的混合策略

1.多线程实现的混合策略

用户级多线程技术适合于解决逻辑并行性问题

内核级多线程技术适合于解决物理并行性问题。

采用混合式策略实现多线程技术有以下一些特点:

（1）组合了用户级多线程策略和内核级多线程策略，可以提供各种复杂的语义、纯用户级线程、纯内核级线程以及折中方案

（2）线程的创建完全在用户空间完成，其调度和同步也完全在用户空间完成，线程的切换代价比较低

（3）多个用户级线程被映射到一些内核级多线程上，可以通过调节两类线程数量以达到一个应用整体最佳的有效解决方案

（4）该策略既可以有效利用物理并行性，又可以有效利用逻辑并行性，且对于各种各样的并行性,可以通过折中得到非常好的处理效果

由操作系统来调度的内核级线程的状态与进程三态完全一致,包括运行(mnning)态、可运行(munnable)态(就绪态)和阻塞(blocked)态(等待态)

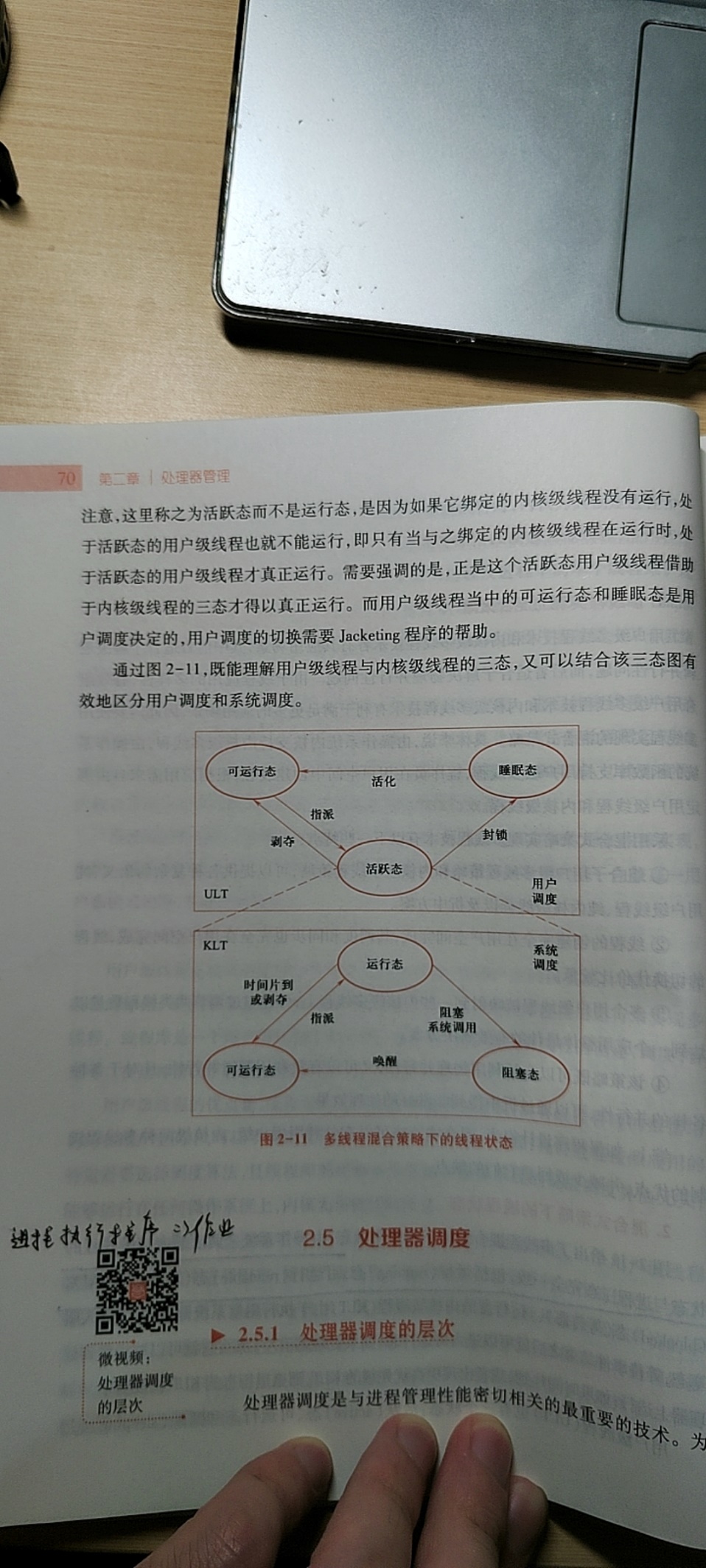
运行态的内核级线程(KLT)由于执行阻塞系统调用，可以进入阻塞态，等待事件结束之后就可以进入可运行态；如果处理器有空，那么它就可以被指派到处理器上运行；如果时间片到，或者出现更高优先级的 KLT，那么运行态的 KLT 就被剥夺

用户级线程(ULT)也有三个状态：活跃(acive)态、可运行态和睡眠(sleeping)态

活跃态而不是运行态，是因为如果它绑定的内核级线程没有运行，处于活跃态的用户级线程也就不能运行，即只有当与之绑定的内核级线程在运行时，处于活跃态的用户级线程才真正运行

正是这个活跃态，用户级线程借助于内核级线程的三态才得以真正运行

用户级线程当中的可运行态和睡眠态是用户调度决定的,用户调度的切换需要Jacketing 程序的帮助



2.5 处理器调度

2.5.1 处理器调度的层次

处理器调度是与进程管理性能密切相关的最重要的技术。为

2.5.2处理器调度算法