南京大学
NANJING UNIVERSITY

计算机操作系统
复习

南京大学软件学院

南京大学
NANJING UNIVERSITY

课程学习目标

* 明确计算机操作系统的作用与功能
* 掌握操作系统实现的基本原理与方法
  * 在微观上，掌握设计实现各个操作系统模块的方法、策略与算法
  * 在宏观上，掌握操作系统的结构和设计实现方法，进一步了解大型软件系统的结构和设计实现方法
* 掌握并发程序设计的基本方法

南京大学
NANJING UNIVERSITY

操作系统课程的教学原则

* 用系统的观点、全局的观点、整体的观点来看待操作系统的实现
* 理解软硬件协同解决问题的方法
* 理解分而治之、分层实现在复杂软件系统实现中的重要作用
* 用工程师的立场来看待操作系统的实现
* 理解文化在操作系统实现中的重要作用

计算机专业考研全国统考《操作系统》考查目标

1. 了解操作系统在计算机系统中的作用、地位、发展和特点。
2. 理解操作系统的基本概念、原理，掌握操作系统设计方法与实现技术。
3. 能够运用所学的操作系统原理、方法与技术分析问题和解决问题。

4

一、操作系统概述

* 一、操作系统概述 (Chap 1)
  * 操作系统的概念、特征、功能和提供的服务
  * 操作系统的发展与分类
  * 操作系统的运行环境

5

二、进程管理

二、进程管理 (Chap 2.3~2.4)

* (一)进程与线程(2.3~2.4)
  1. 进程概念
  2. 进程的状态与转换
  3. 进程控制
  4. 进程组织 (进程的队列)
  5. 进程通信 (6.5) (共享存储系统；消息传递系统；管道通信)
  6. 线程概念与多线程模型 (2.4)

6

二、进程管理

* (二)处理机调度 (2.5)
  * 1.调度的基本概念
  * 2.调度时机、切换与过程
  * 3.调度的基本准则
  * 4.调度方式抢占与非抢占
  * 5.典型调度算法 (2.5.2)
    * 先来先服务调度算法；短作业(短任务、短进程、短线程)优先调度算法；时间片轮转调度算法；优先级调度算法；高响应比优先调度算法；多级反馈队列调度算法。

7

二、进程管理

* (三)进程同步 (Chap6)
  1. 进程同步的基本概念 (6.1)
  2. 实现临界区互斥的基本方法 (6.2)
     - 软件实现方法；硬件实现方法。
  3. 信号量 (6.3)
  4. 管程 (6.4)
  5. 经典同步问题 (6.3~6.4)
     - 生产者-消费者问题；读者-写者问题；哲学家进餐问题。

8

二、进程管理

* （四）死锁 (Chap 6.6)
  1. 死锁的概念
  2. 死锁处理策略
  3. 死锁预防
  4. 死锁避免
     - 系统安全状态：银行家算法。
  5. 死锁检测和解除

9

三、内存管理

* 三、内存管理 (Chap 3.1~3.4)

(一)内存管理基础 (3.1~3.2)

1. 内存管理概念
   - 程序装入与链接；逻辑地址与物理地址空间；内存保护。
2. 交换与覆盖
3. 连续分配管理方式
   - 单一连续分配；分区分配。
4. 非连续分配管理方式:分页管理方式；分段管理方式；段页式管理方式。

10

三、内存管理

* (二) 虚拟内存管理 (Chap 3.3~3.4)
  1. 虚拟内存基本概念
  2. 请求分页管理方式
  3. 页面置换算法：最佳置换算法(OPT)；先进先出置换算法(FIFO)；最近最少使用置换算法(LRU)；时钟置换算法(CLOCK)。
  4. 页面分配策略
  5. 抖动：抖动现象；工作集。
  6. 请求分段管理方式
  7. 请求段页式管理方式

11

四、文件管理

文件管理 (Chap 5)

* (一) 文件系统基础

1. 文件概念
2. 文件结构
   - 顺序文件；索引文件；索引顺序文件。
3. 目录结构
   - 文件控制块和索引节点；单级目录结构和两级目录结构；树形目录结构；图形目录结构。
4. 文件共享
   - 共享动机；共享方式；共享语义。
5. 文件保护
   - 访问类型；访问控制。

12

四、文件管理

* (二) 文件系统基础 (chap 5.2~5.3)
  1. 文件系统实现
  2. 文件系统层次结构
  3. 目录实现
  4. 文件实现
* (三) 磁盘组织与管理 (chap 4.4)
  1. 磁盘的结构
  2. 磁盘调度算法
  3. 磁盘的管理

13

五、输入输出(I/O)管理

* (一) 输入输出(I/O)管理 (Chap 4)
  1. I/O管理概述
  2. I/O设备
  3. I/O管理目标
  4. I/O管理功能
  5. I/O应用接口
  6. I/O控制方式
* (二) I/O核心子系统
  1. I/O调度概念
  2. 高速缓存与缓冲区
  3. 设备分配与回收
  4. 假脱机技术(SPOOLing)(4.5)
  5. 出错处理

14

考研题型分析

(全国统考)

专业试卷总分150分(4门课),

含数据结构45分、组成原理45分、操作系统35分、网络25分

2013年考研，南大软院改为自主命题，组成原理改为《软件工程》

* 单项选择题
  * 概念为主
  * 小型计算题
* 综合应用题
  * 算法题(计算)
  * 互斥与同步问题：信号量与PV操作，管程方法

15

计算题型分析

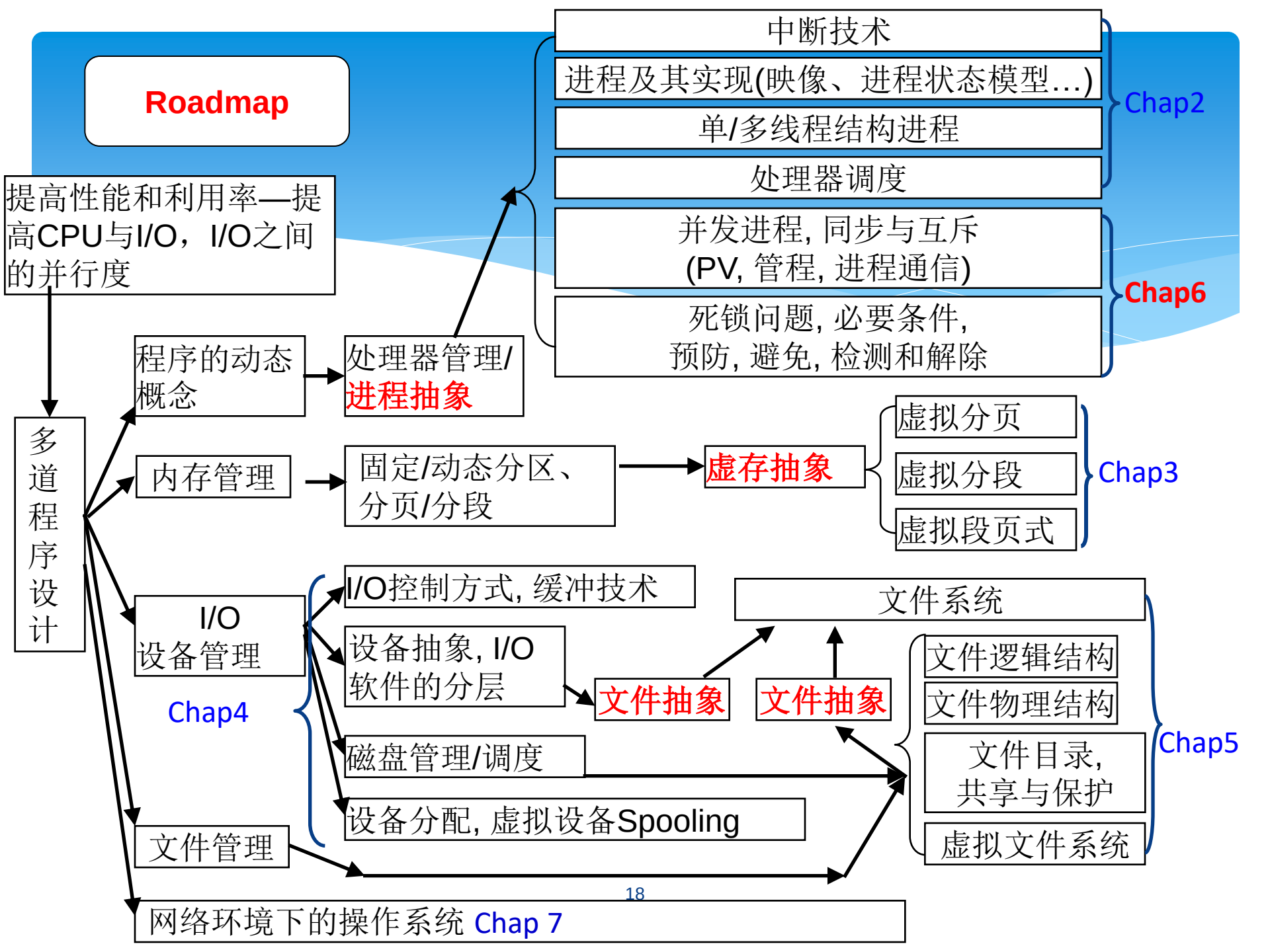
* 多道程序设计
* CPU调度算法
* 死锁避免银行家算法，死锁检测
* 连续分配，分区分配：适配算法，伙伴系统
* 地址转换计算：分页管理方式；分段管理方式。
* 页面置换算法：
* 抖动现象，工作集
* 磁盘调度算法
* 文件系统的计算
* PV操作、管程

16

期末考试题型

* 一、选择题 (50分)
* 二、简答题 (12分)
* 三、计算题/应用题/算法题 (38分)

17

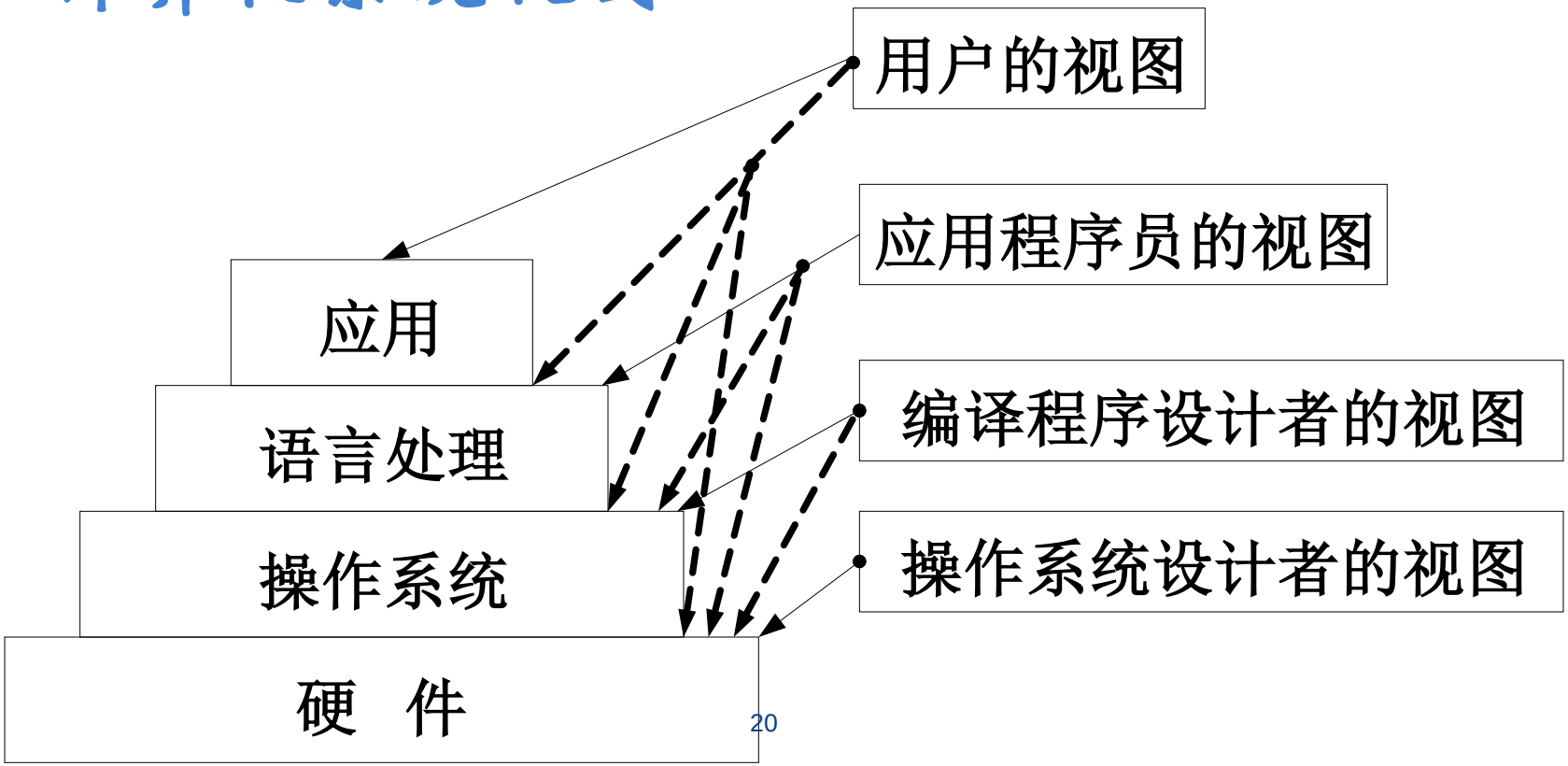

Roadmap
提高性能和利用率—提高CPU与I/O，I/O之间的并行度
中断技术
进程及其实现(映像、进程状态模型…)
单/多线程结构进程
处理器调度
Chap2
并发进程, 同步与互斥
(PV, 管程, 进程通信)
死锁问题, 必要条件,
预防, 避免, 检测和解除
Chap6
多道程序设计
程序的动态概念
处理器管理/
进程抽象
内存管理
固定/动态分区、
分页/分段
虚存抽象
虚拟分页
虚拟分段
虚拟段页式
Chap3
I/O
设备管理
Chap4
I/O控制方式, 缓冲技术
设备抽象, I/O
软件的分层
文件抽象
文件抽象
磁盘管理/调度
设备分配, 虚拟设备Spooling
文件系统
文件逻辑结构
文件物理结构
文件目录,
共享与保护
虚拟文件系统
Chap5
文件管理
网络环境下的操作系统 Chap 7

第一章 计算机系统概述
本主题教学目标

* 了解计算机硬件与操作技术的发展
  * 概念：分时操作系统、实时操作系统
* 掌握多道程序设计的概念
* 掌握计算机系统的组成
* 了解计算机体系结构与计算机总线、处理器、存储器、I/O设备以及I/O控制方式
* 掌握计算机系统的层次结构

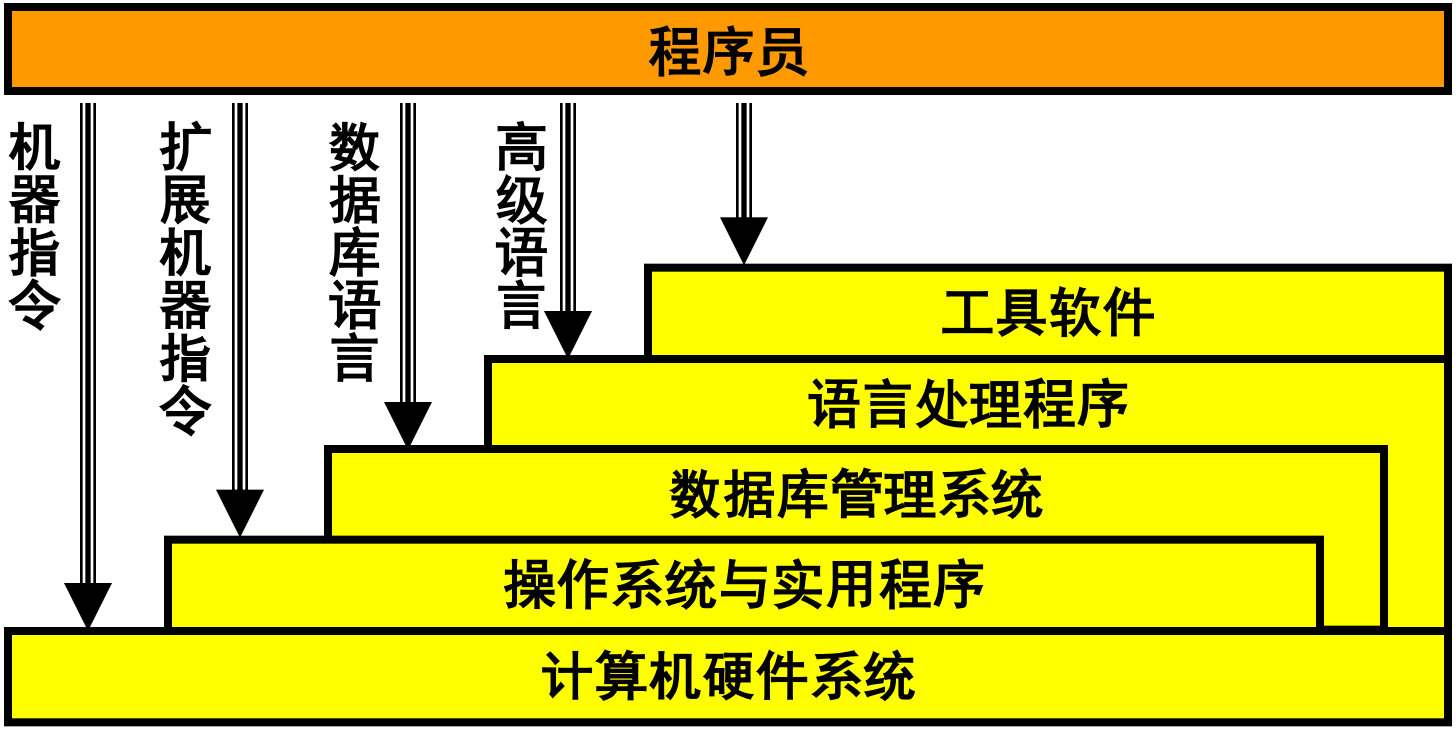
计算机系统的层次结构

* 计算机系统视图



南京大学
NANJING UNIVERSITY

程序员的视图



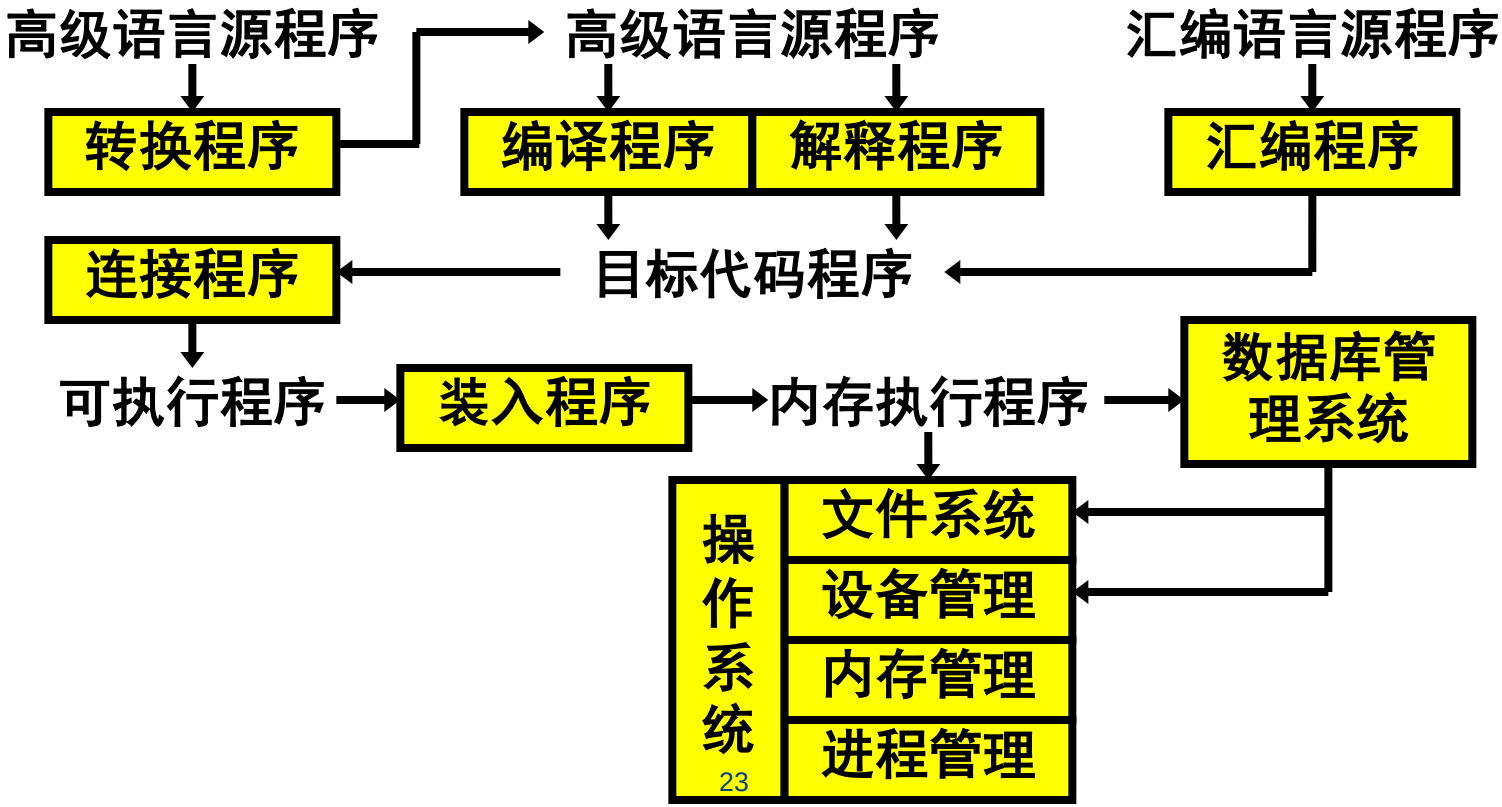
南京大学
NANJING UNIVERSITY

软件开发的不同层次

* 计算机硬件系统：机器语言
* 操作系统之资源管理：机器语言+广义指令(扩充了硬件资源管理)
* 操作系统之文件系统：机器语言+系统调用(扩充了信息资源管理)
* 数据库管理系统：+数据库语言(扩充了功能更强的信息资源管理)
* 语言处理程序：面向问题的语言

南京大学
NANJING UNIVERSITY

计算机程序的执行过程



操作系统概述
本主题教学目标

1. 掌握处理器
2. 掌握操作系统管理的资源
3. 掌握操作系统的用户接口
4. 了解操作系统的类型
5. 了解操作系统的结构
6. 了解操作系统主流产品

南京大学
NANJING UNIVERSITY

批处理操作系统

* 成批处理作业
* 作业控制语言与作业说明书
* 脱机工作方式
* 追求系统效率与吞吐量

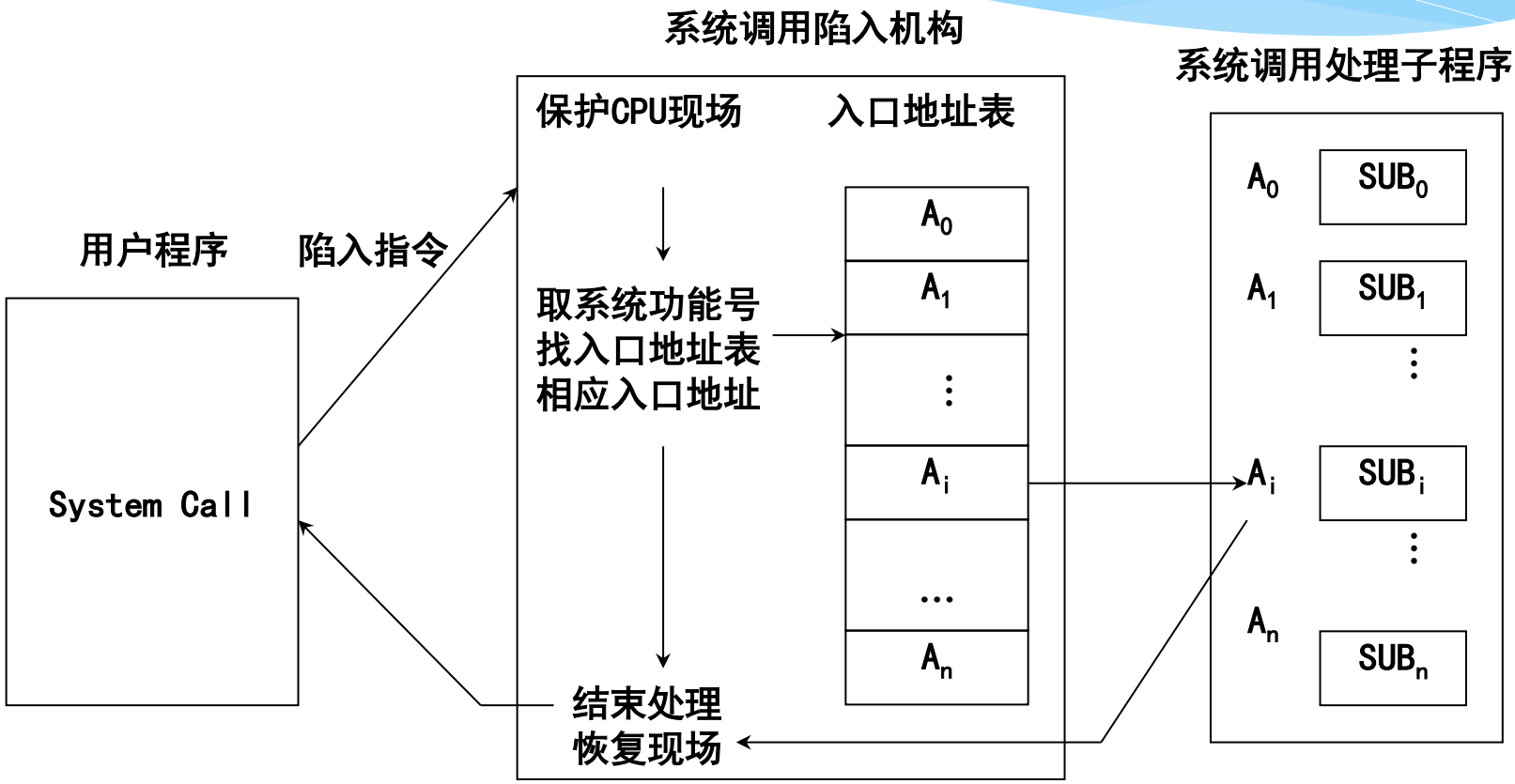
分时操作系统

* 用户通过终端直接控制程序执行
* 交互式工作方式
* 交互型、友善性、快速响应
* 今天最常见的计算机操作方式

实时操作系统

* 事件驱动，有较高时间要求
* 实时操作系统的分类
  * 过程控制系统
  * 信息查询系统
  * 事务处理系统
* 过程控制系统的处理步骤：数据采集、加工处理、操作控制、反馈处理

系统调用的实现过程


系统调用陷入机构
系统调用处理子程序
保护CPU现场
入口地址表
用户程序
陷入指令
取系统功能号
找入口地址表
相应入口地址
System Call
结束处理
恢复现场
A0
A1
Ai
An
SUB0
SUB1
SUBi
SUBn

操作系统结构分类

1. 单体式结构
2. 层次式结构
3. 虚拟机结构
4. 微内核结构
5. 客户/服务器结构

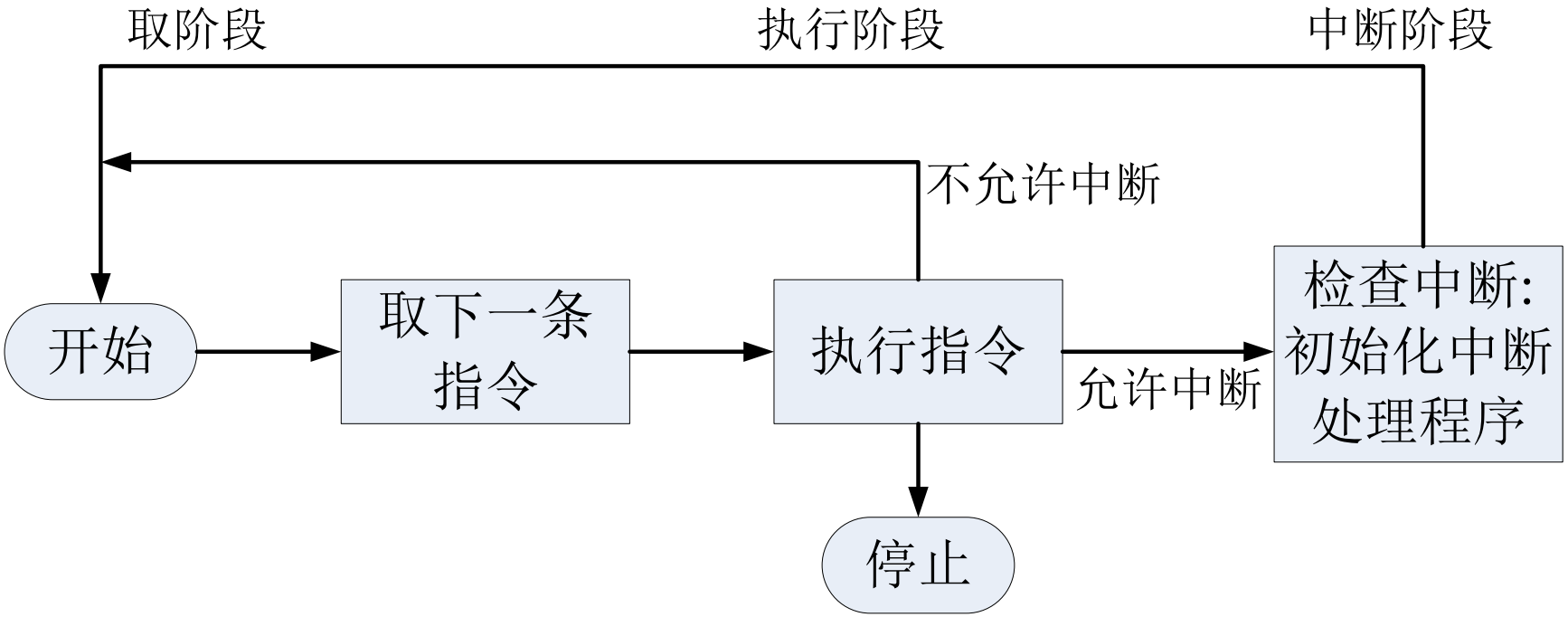
中断处理
本主题教学目标

1. 了解处理器寄存器
2. 掌握处理器状态、特权指令、程序状态字
3. 掌握指令执行周期
4. 了解指令流水线
5. 掌握中断和中断源
6. 掌握中断响应和处理的过程
7. 掌握中断的优先级和多重中断

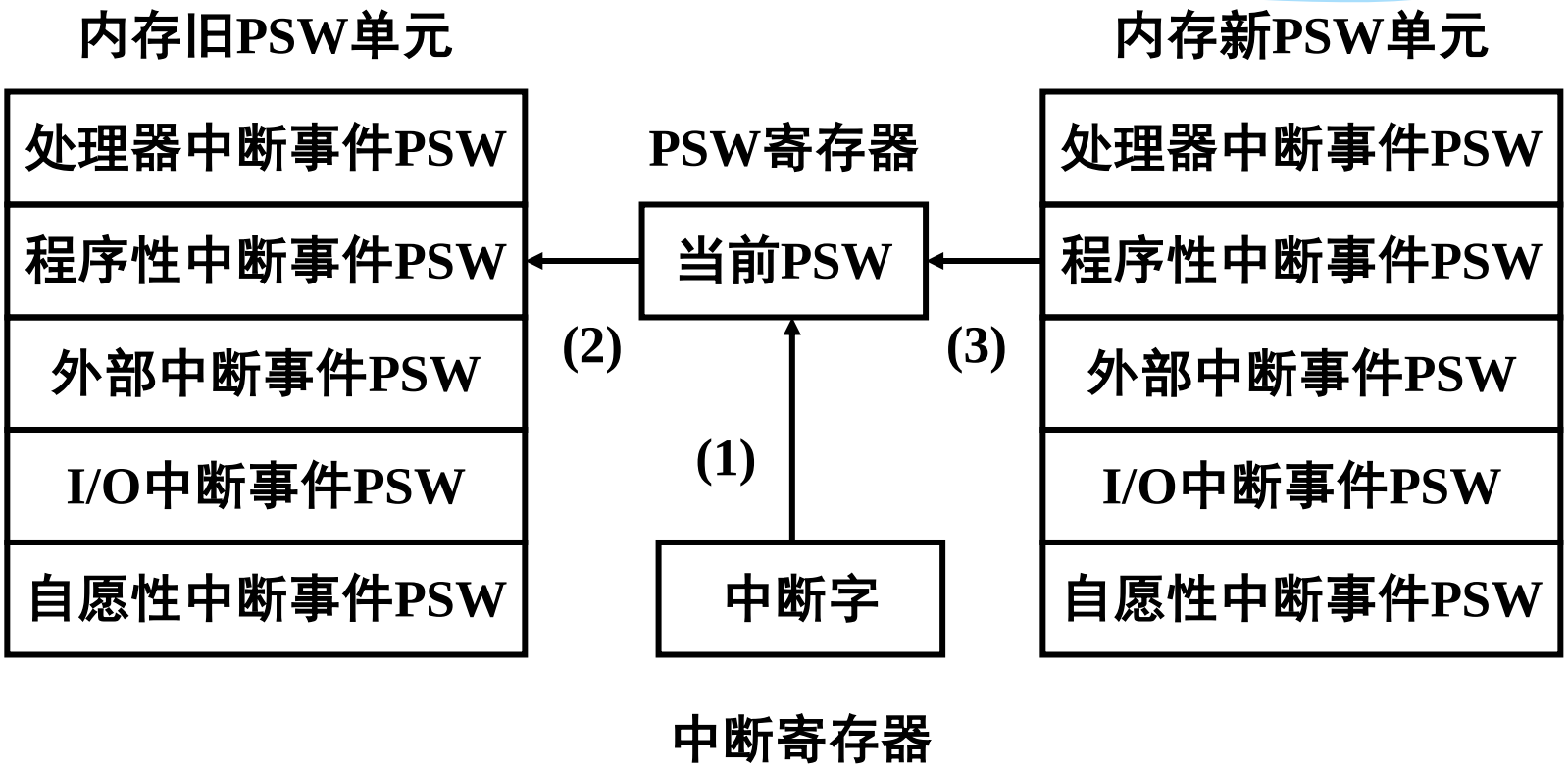
特权指令和处理器状态

* 从资源管理和控制程序执行的角度出发，必须设置特权指令，提供给操作系统的核心程序使用
* 处理器状态
  * 管理状态(特权状态、系统模式、特态或管态)：处理器可以执行全部指令，使用所有资源，并具有改变处理器状态的能力
  * 用户状态(目标状态、用户模式、常态或目态)：处理器只能执行非特权指令
  * 核心状态、管理状态和用户状态

中断与指令周期

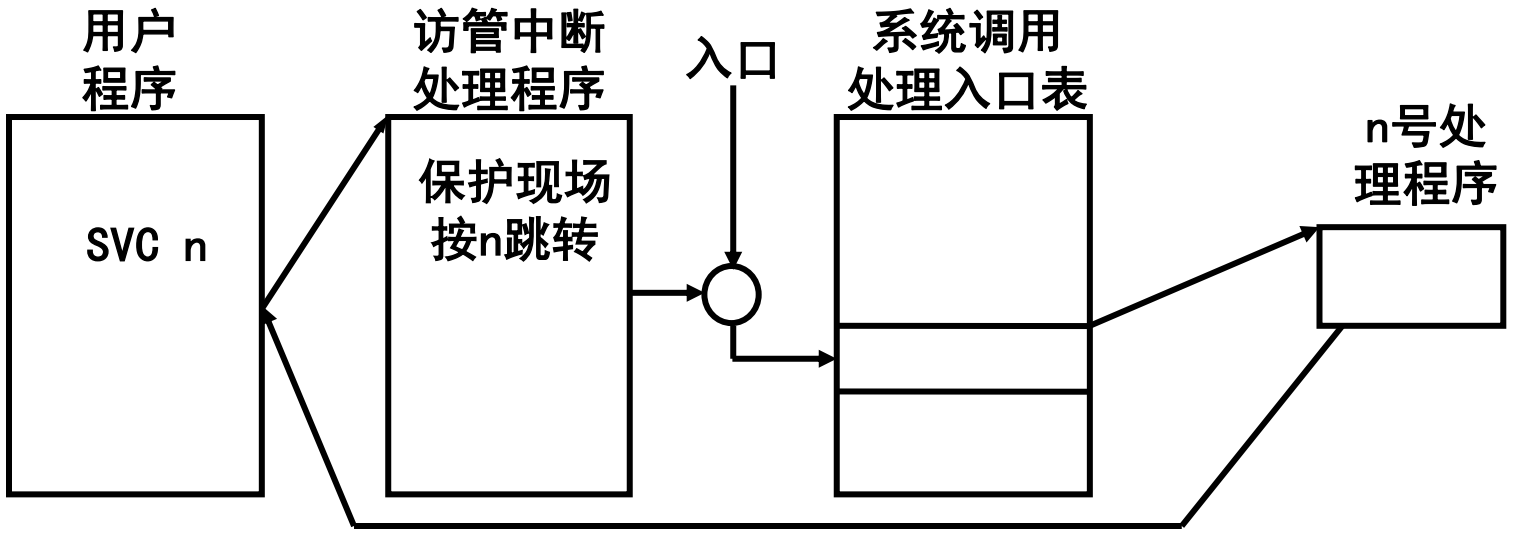


中断装置与中断响应

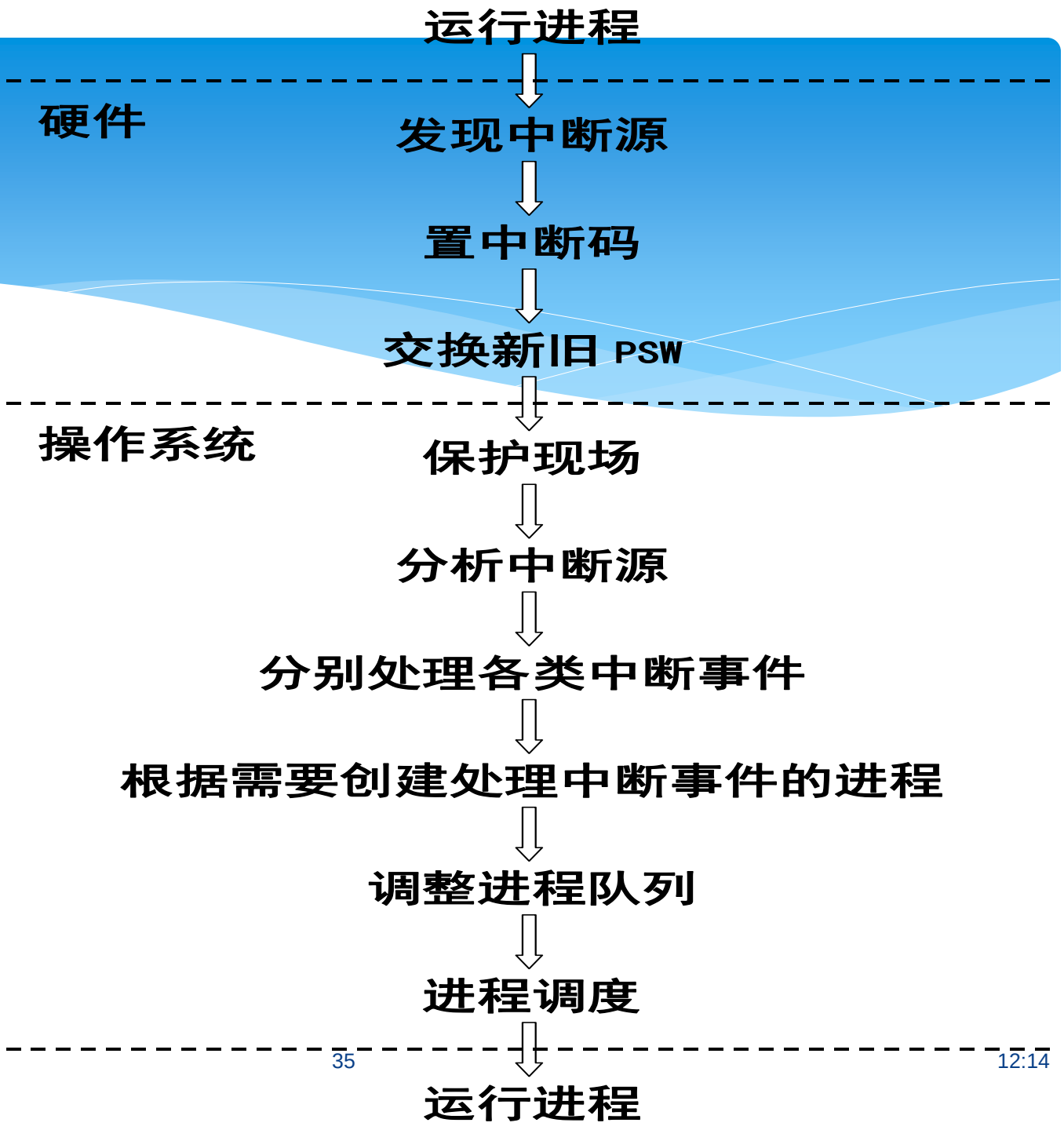

内存旧PSW单元
处理器中断事件PSW
程序性中断事件PSW
外部中断事件PSW
I/O中断事件PSW
自愿性中断事件PSW
PSW寄存器
当前PSW
(2)
(1)
中断字
(3)
内存新PSW单元
处理器中断事件PSW
程序性中断事件PSW
外部中断事件PSW
I/O中断事件PSW
自愿性中断事件PSW
中断寄存器

自愿性中断事件的处理

* 用户程序执行系统调用(访管指令,广义指令)；操作系统把系统调用参数作为中断字，分析检查后进行相应处理



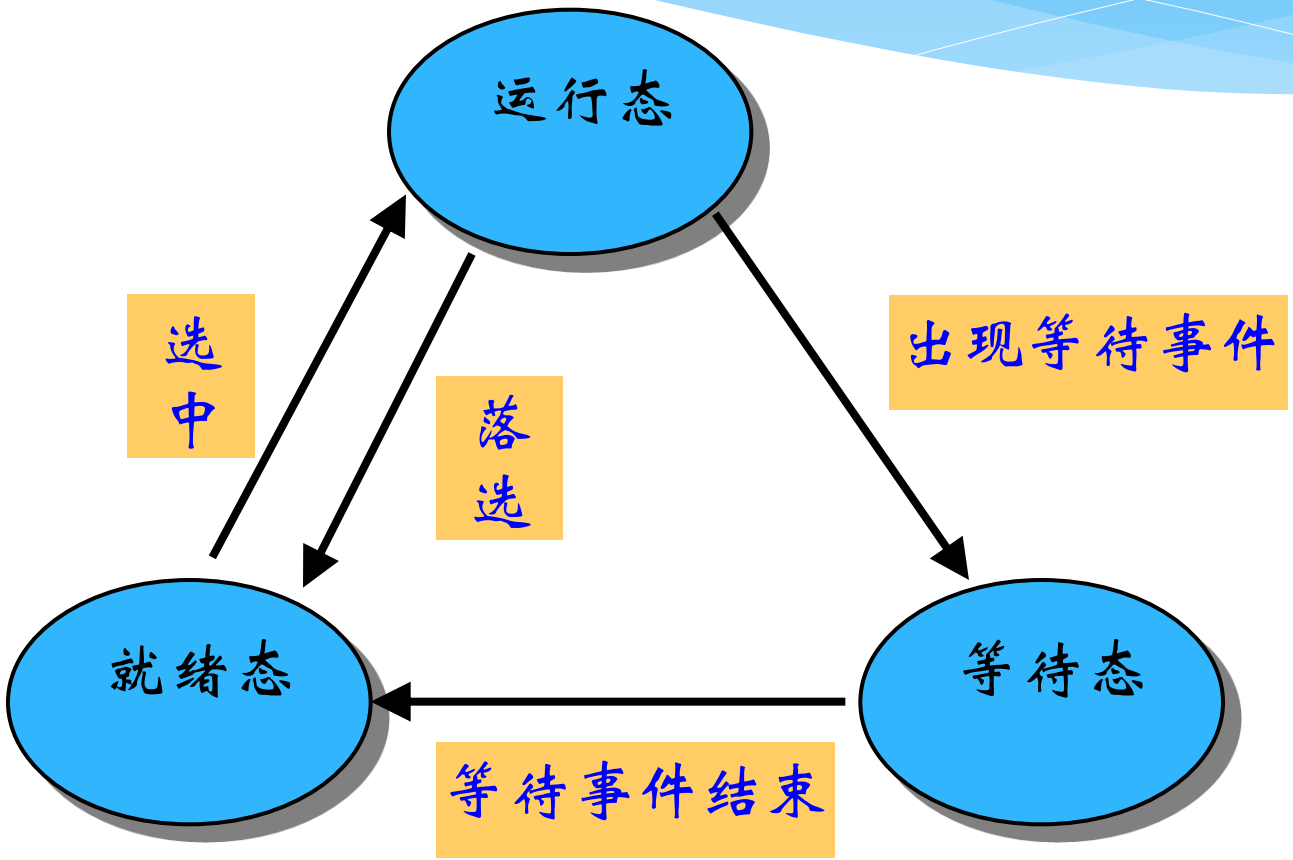
中断控制流程


运行进程
硬件
发现中断源
置中断码
交换新旧 PSW
操作系统
保护现场
分析中断源
分别处理各类中断事件
根据需要创建处理中断事件的进程
调整进程队列
进程调度
运行进程

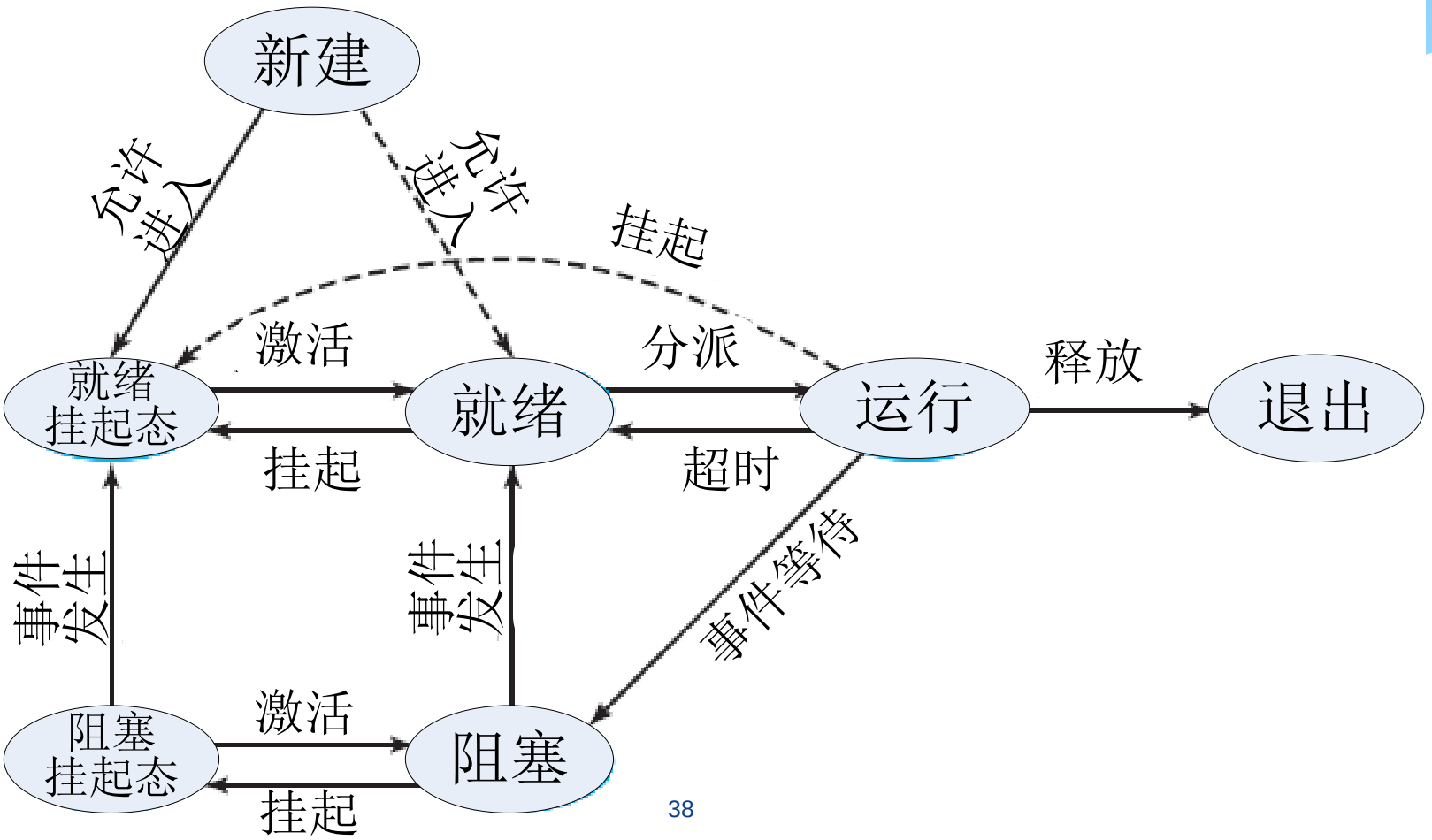
第二章 处理器管理
本主题教学目标

1. 掌握进程的概念，可再入过程
2. 掌握进程的状态、进程的挂起，以及队列实现模型
3. 掌握操作系统的控制结构
4. 掌握进程描述与控制的数据结构
5. 掌握处理机模式的概念
6. 掌握进程创建、模式切换、进程切换、进程队列、进程原语等进程实现的原理
7. 了解操作系统的执行模型

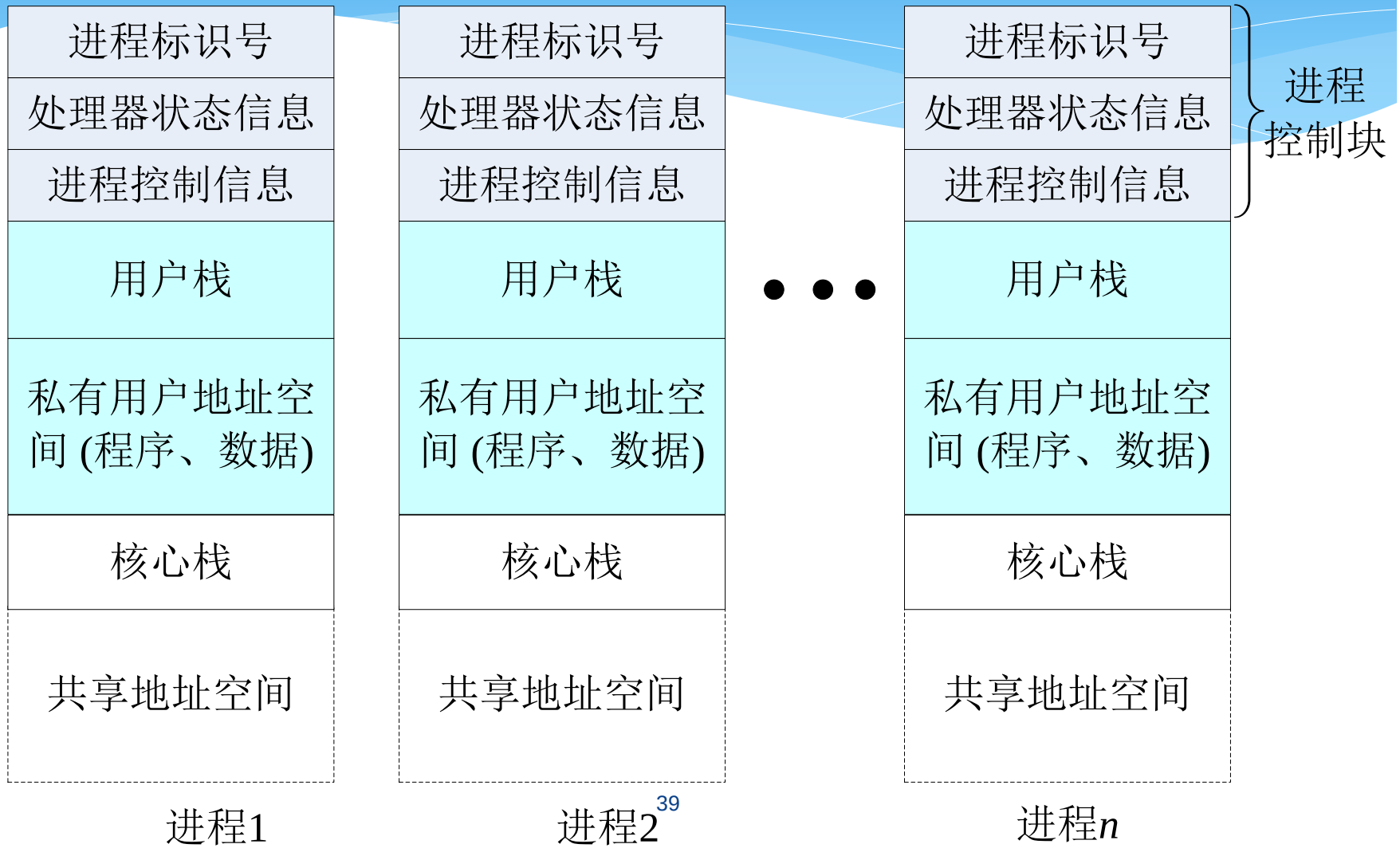
三状态模型及其转换



含两个挂起态(七状态模型)



虚拟内存中的用户进程


进程标识号
处理器状态信息
进程控制信息
用户栈
私有用户地址空间 (程序、数据)
核心栈
共享地址空间
进程1
进程标识号
处理器状态信息
进程控制信息
用户栈
私有用户地址空间 (程序、数据)
核心栈
共享地址空间
进程2
进程标识号
处理器状态信息
进程控制信息
用户栈
私有用户地址空间 (程序、数据)
核心栈
共享地址空间
进程n
进程控制块

多线程技术
本主题教学目标

1. 掌握多线程环境下进程和线程的概念
2. 掌握线程的三种实现模型
3. 了解Windows的进程和线程管理
4. 掌握Solaris的进程和线程管理模型

单线程结构进程给并发程序设计效率带来问题

* 进程切换开销大
* 进程通信开销大
* 限制了进程并发的粒度
* 不适合并行计算的要求

南京大学
NANJING UNIVERSITY

线程的概念(1)

解决问题的基本思路：

* 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，不需要频繁地切换和保护资源；
* 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，在这种指导思想下，产生了线程的概念。

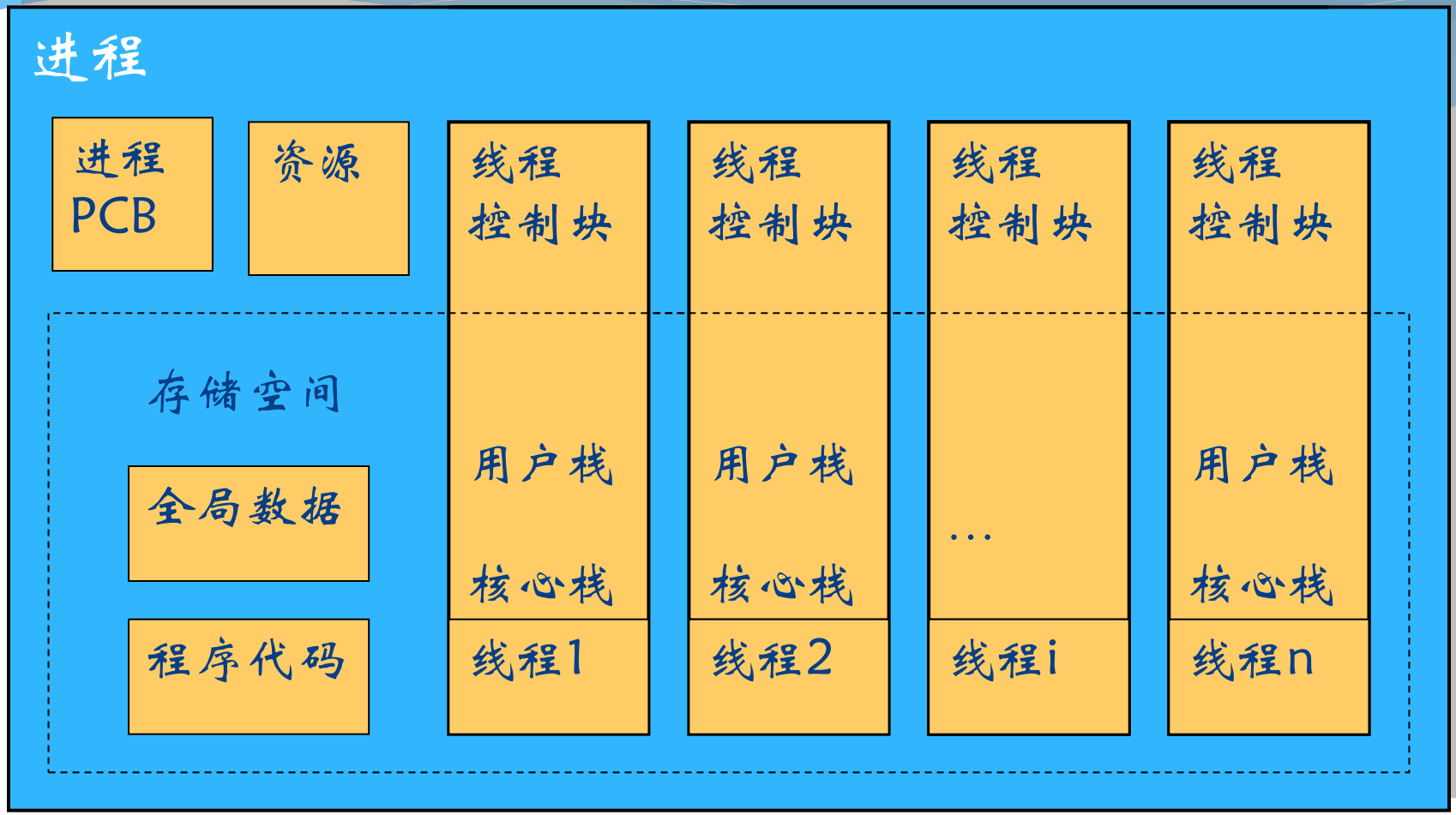
42

线程的概念(2)

* 操作系统中引入进程的目的是为了使多个程序并发执行，以改善资源使用率和提高系统效率，
* 操作系统中再引入线程，则是为了减少程序并发执行时所付出的时空开销，使得并发粒度更细、并发性更好。

43

多线程结构的进程


进程
进程
PCB
资源
线程
控制块
线程
控制块
线程
控制块
线程
控制块
存储空间
全局数据
程序代码
用户栈
核心栈
用户栈
核心栈
...
用户栈
核心栈
线程1
线程2
线程i
线程n

44

南京大学
NANJING UNIVERSITY

多线程结构的进程

* 线程是进程的组成部分，每个进程内允许包含多个并发执行的实体(控制流)
* 线程作为处理器调度和分派的一级单位
* 线程的状态(运行态、就绪态、阻塞态)

45

南京大学
NANJING UNIVERSITY

线程组成

* 线程唯一标识符及线程状态信息
* 未运行时保存的线程上下文，可把线程看成是进程中一个独立的程序计数器在操作
* 核心栈，核心态下工作时，保存参数
* 用于存放线程局部变量及用户栈的私有存储区

46

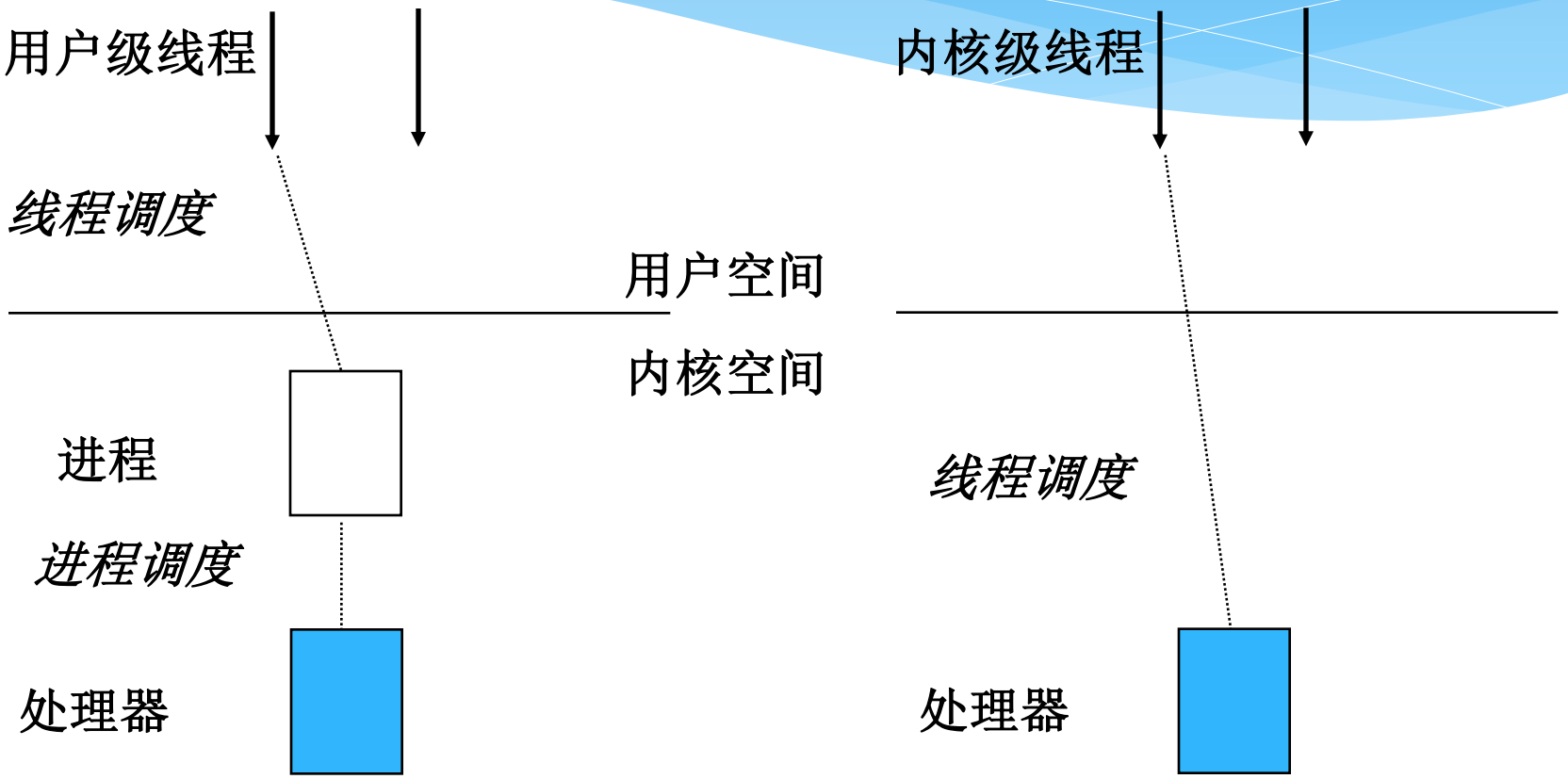
南京大学
NANJING UNIVERSITY

并发多线程程序设计的优点

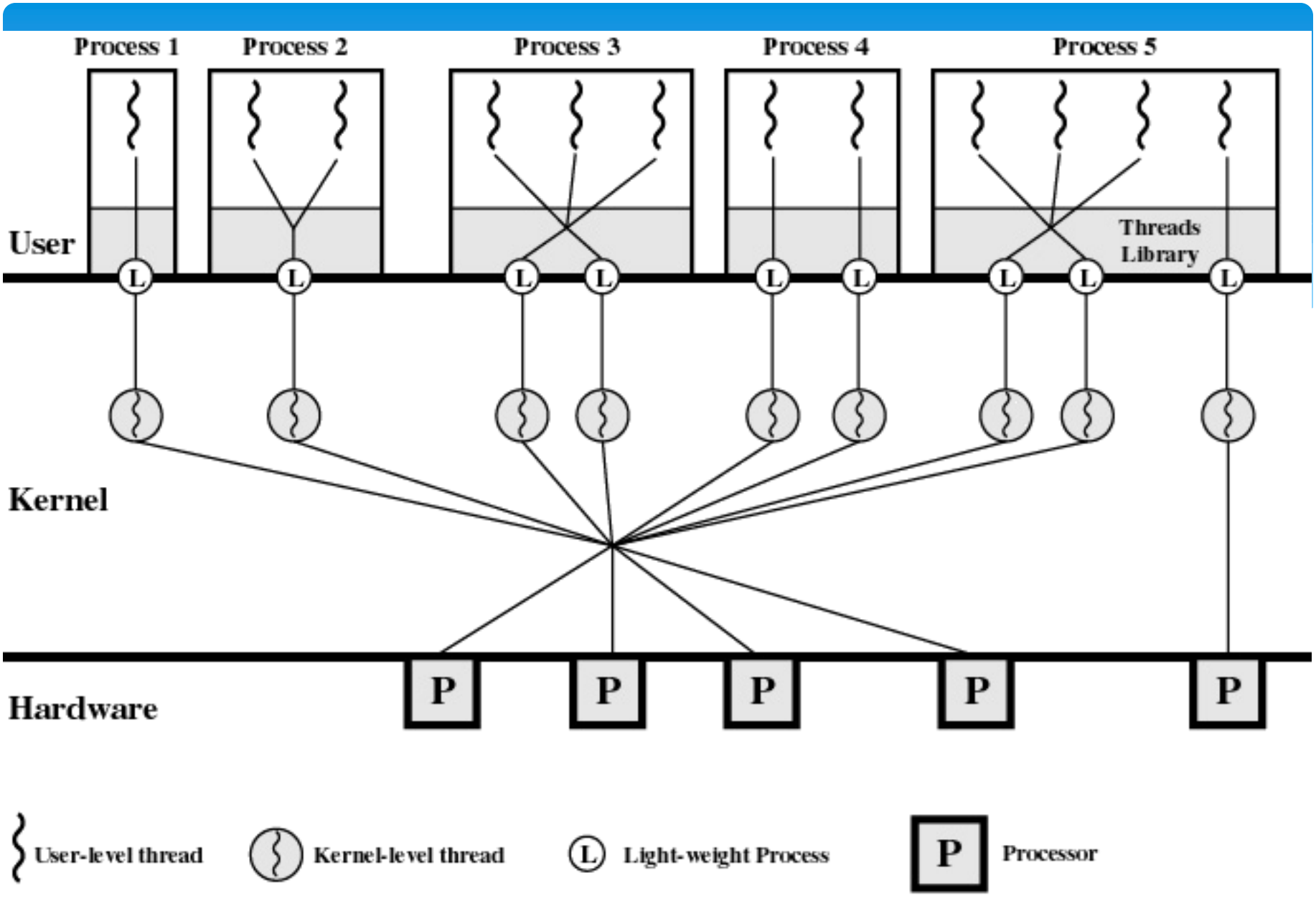
- 快速线程切换
- 减少(系统)管理开销
- (线程)通信易于实现
- 便于共享资源
- 并行程度提高

47

用户级线程 vs. 内核级线程

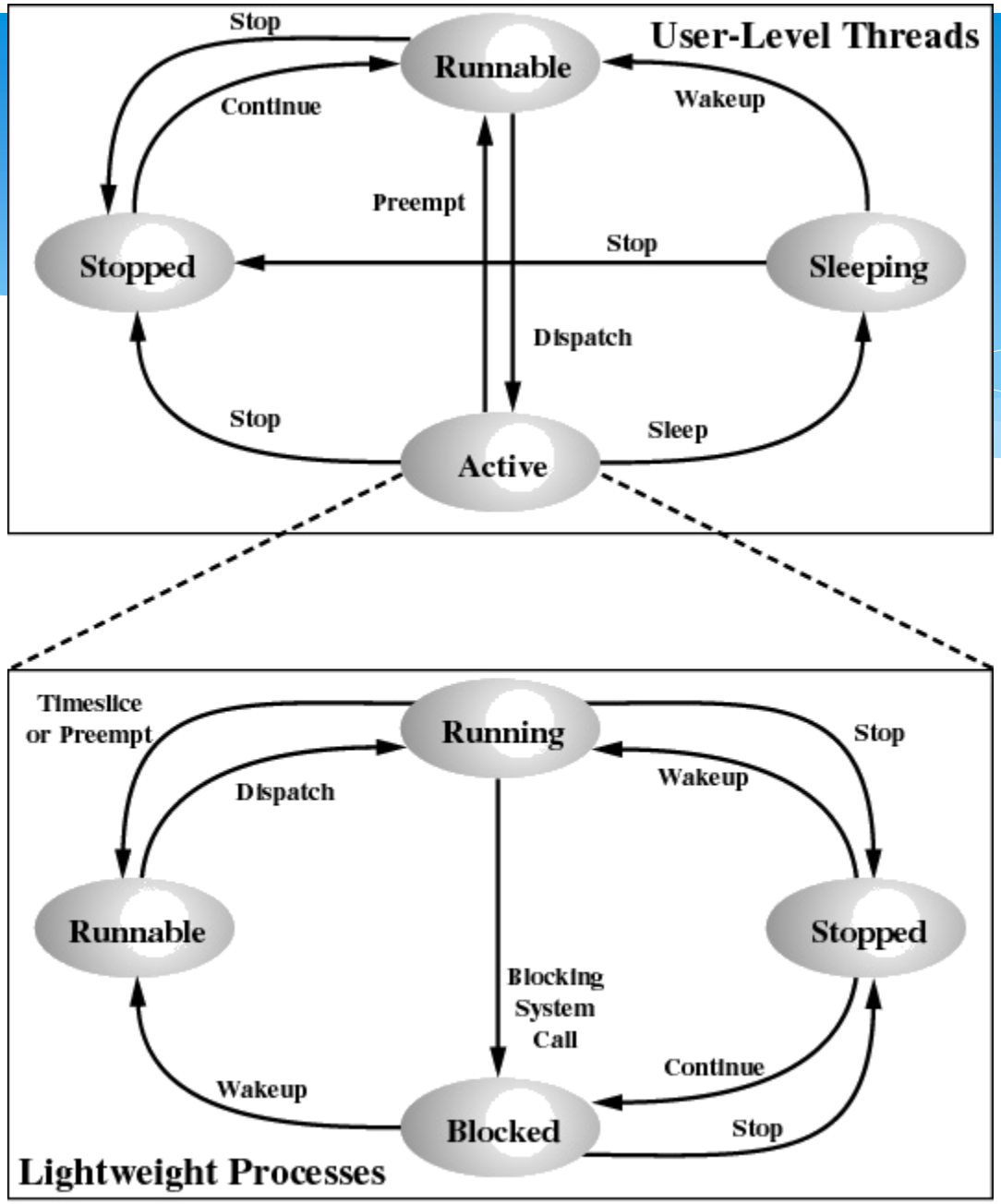


48



Solaris系统: 混合多线程结构

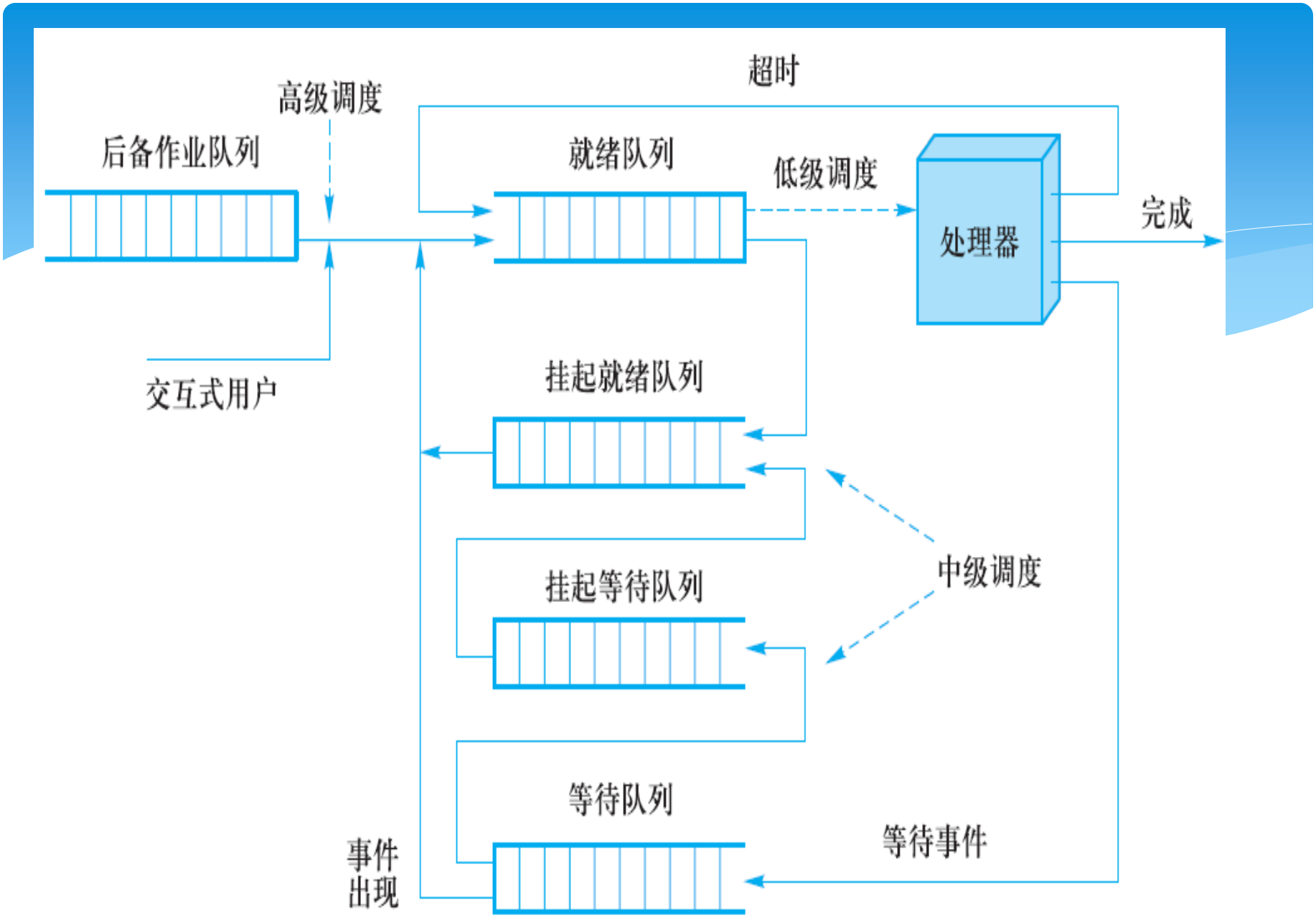
49


User-Level Threads
Runnable
Stop
Continue
Wakeup
Preempt
Stopped
Stop
Sleeping
Dispatch
Stop
Sleep
Active
Lightweight Processes
Running
Timeslice
or Preempt
Stop
Dispatch
Wakeup
Runnable
Stopped
Blocking
System
Call
Continue
Wakeup
Blocked
Stop

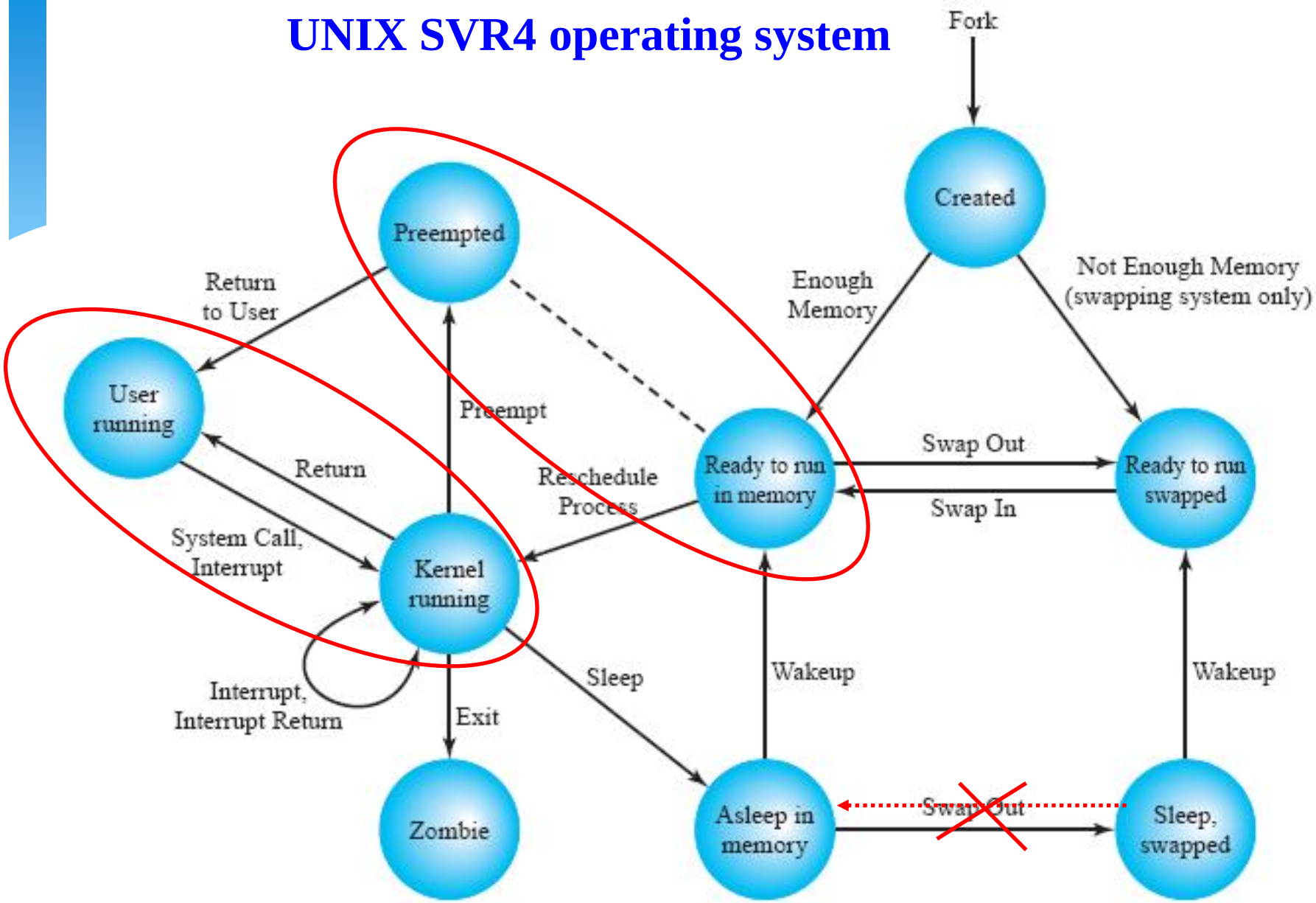
50

处理器调度
本主题教学目标

1. 掌握调度的层次
2. 掌握进程的挂起态
3. 掌握低级调度及其策略
4. 掌握作业调度及其策略


超时
高级调度
后备作业队列
就绪队列
低级调度
处理器
完成
交互式用户
挂起就绪队列
中级调度
挂起等待队列
等待队列
事件
出现
等待事件

UNIX SVR4 operating system

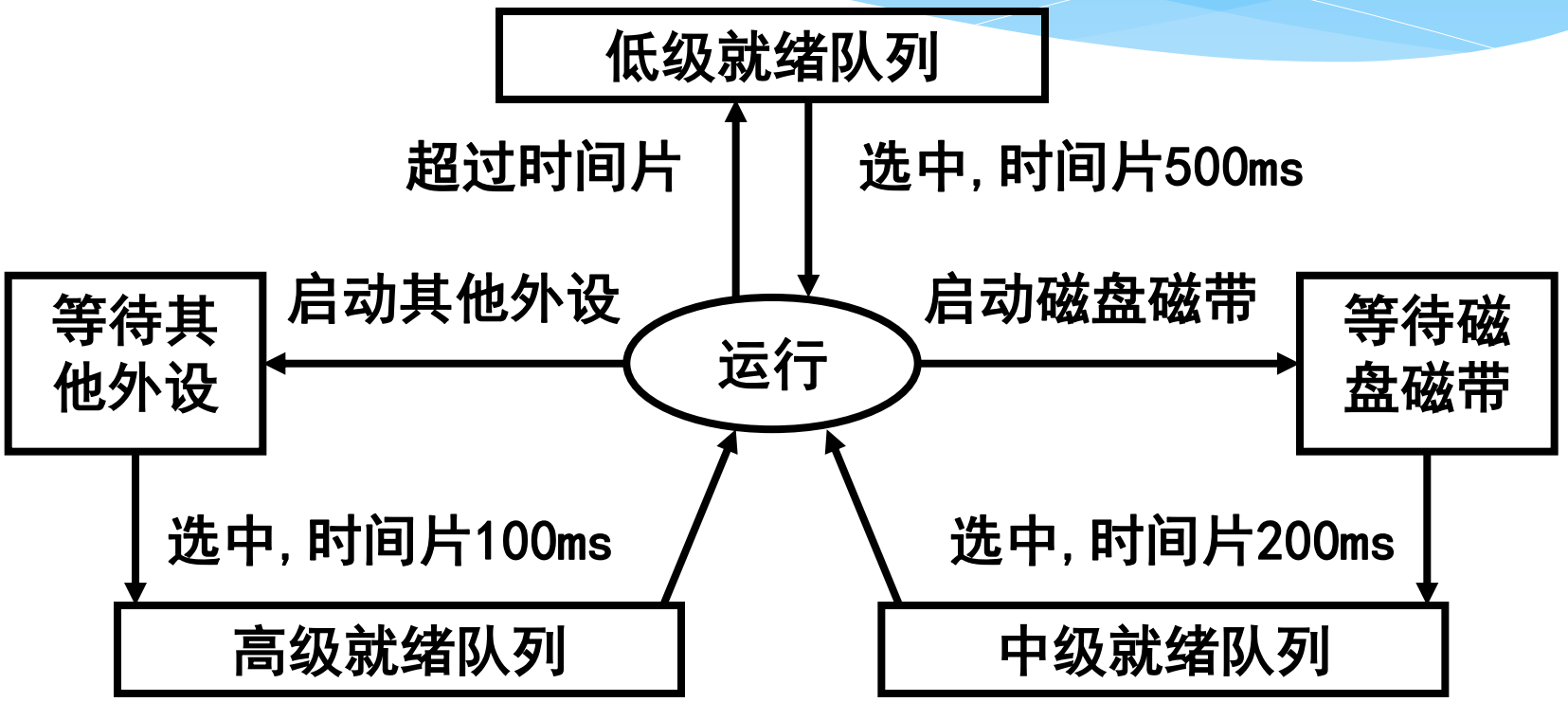


53

处理器调度算法

* FCFS (先来先服务)
* RR (时间片轮转)
* SPN (最短进程优先)
* SRT (最短剩余时间优先)
* HRRF (最高响应比优先)
* Feedback (多级反馈调度)

Feedback


低级就绪队列
超过时间片
选中，时间片500ms
等待其他外设
启动其他外设
运行
启动磁盘磁带
等待磁盘磁带
选中，时间片100ms
选中，时间片200ms
高级就绪队列
中级就绪队列

55

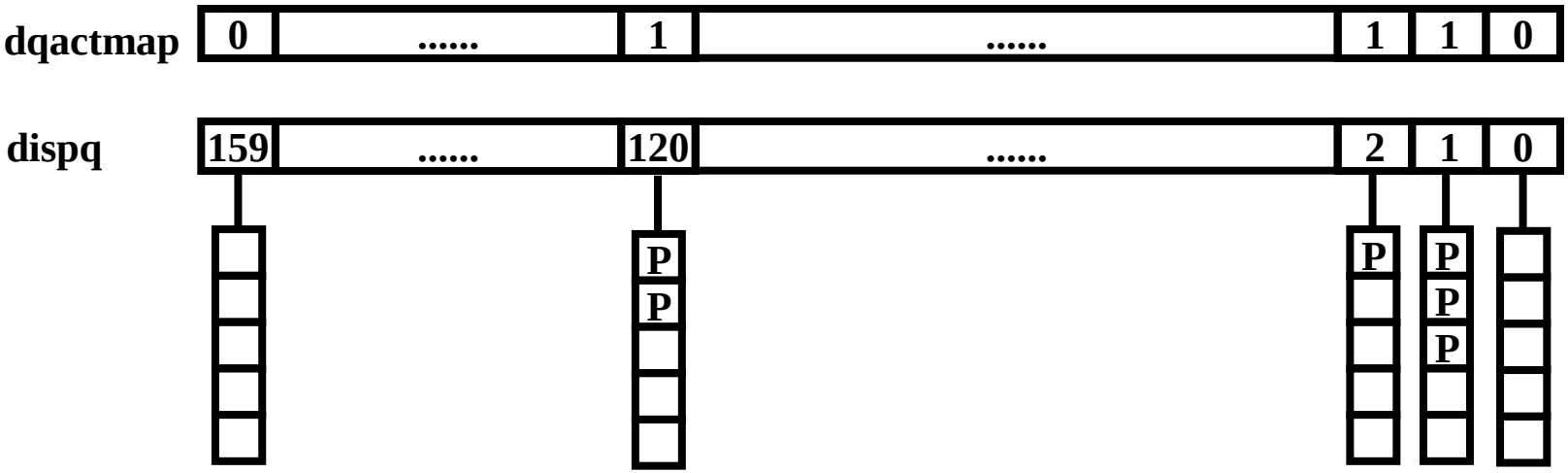
传统Unix系统的调度(例)

* 多级反馈队列，每个优先级队列使用时间片轮转
* 每秒重新计算每个进程的优先级
* 给每个进程赋予基本优先级的目的是把所有进程划分成固定的优先级区
* 可控调节因子

56

Unix SVR4调度算法(例)

* 多级反馈队列，每一个优先数都对应于一个就绪进程队列
* 实时优先级层次：优先数和时间片都是固定的，在抢占点执行抢占
* 分时优先级层次：优先数和时间片是可变的，从0优先数的100ms到59优先数的10ms

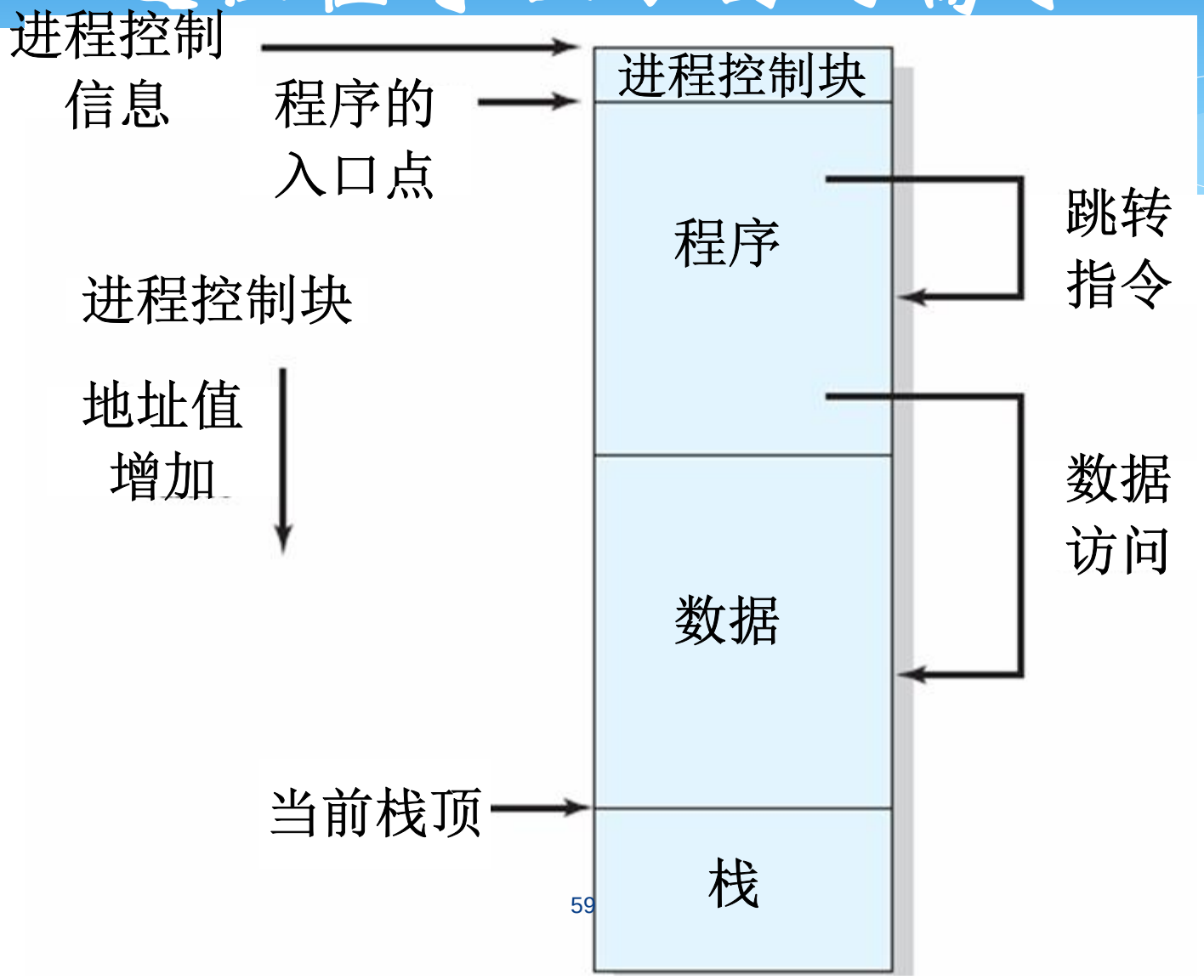


57

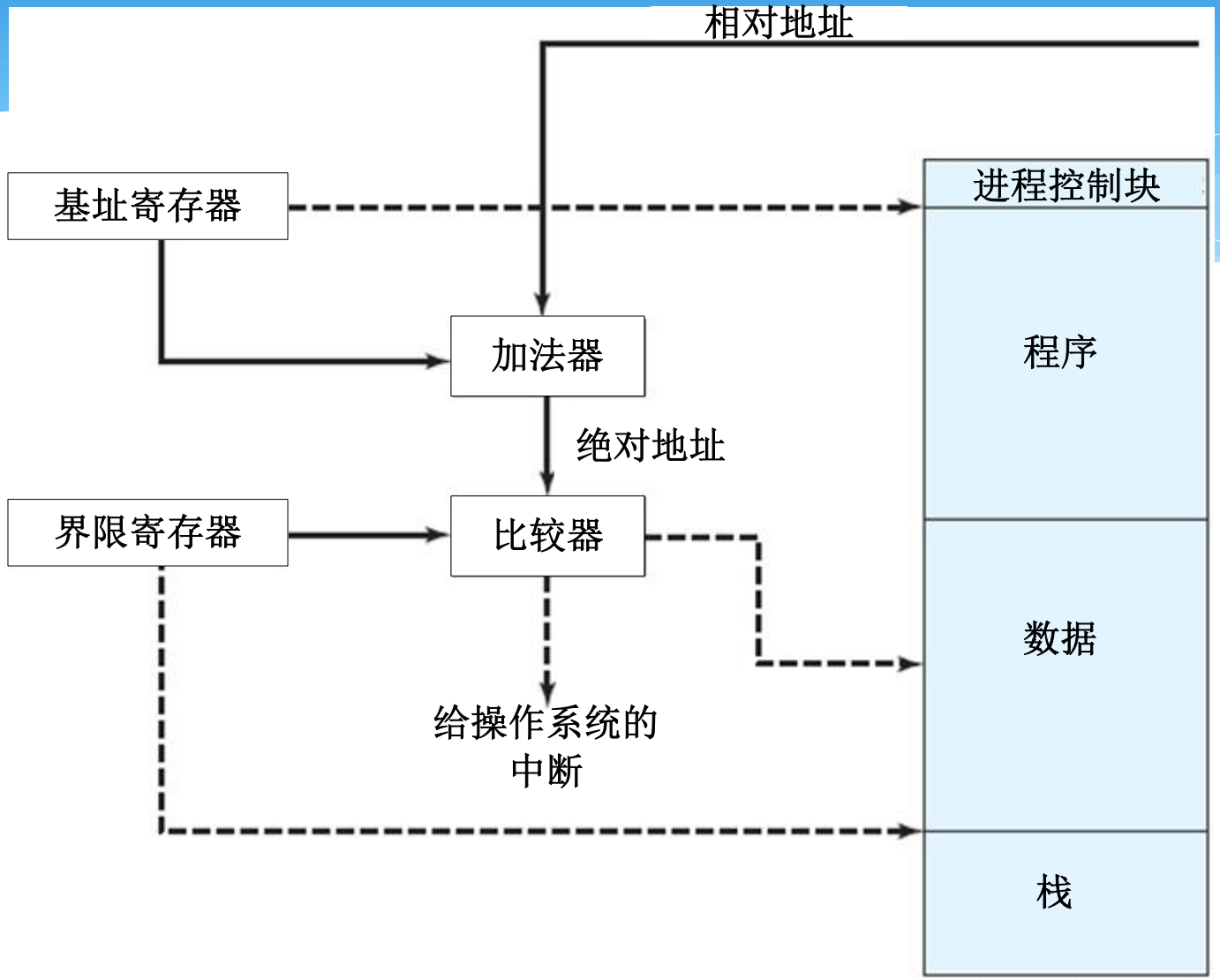
第三章 存储管理
本主题教学目标

1. 掌握存储管理的基本内容
2. 掌握固定分区存储管理
3. 掌握可变分区存储管理
4. 掌握对换、重定位、地址等概念
5. 初步掌握段式存储管理
6. 初步掌握页式存储管理(页表、多级页表、反置页表)
7. 掌握Cache的工作原理

进程在寻址方面的需求

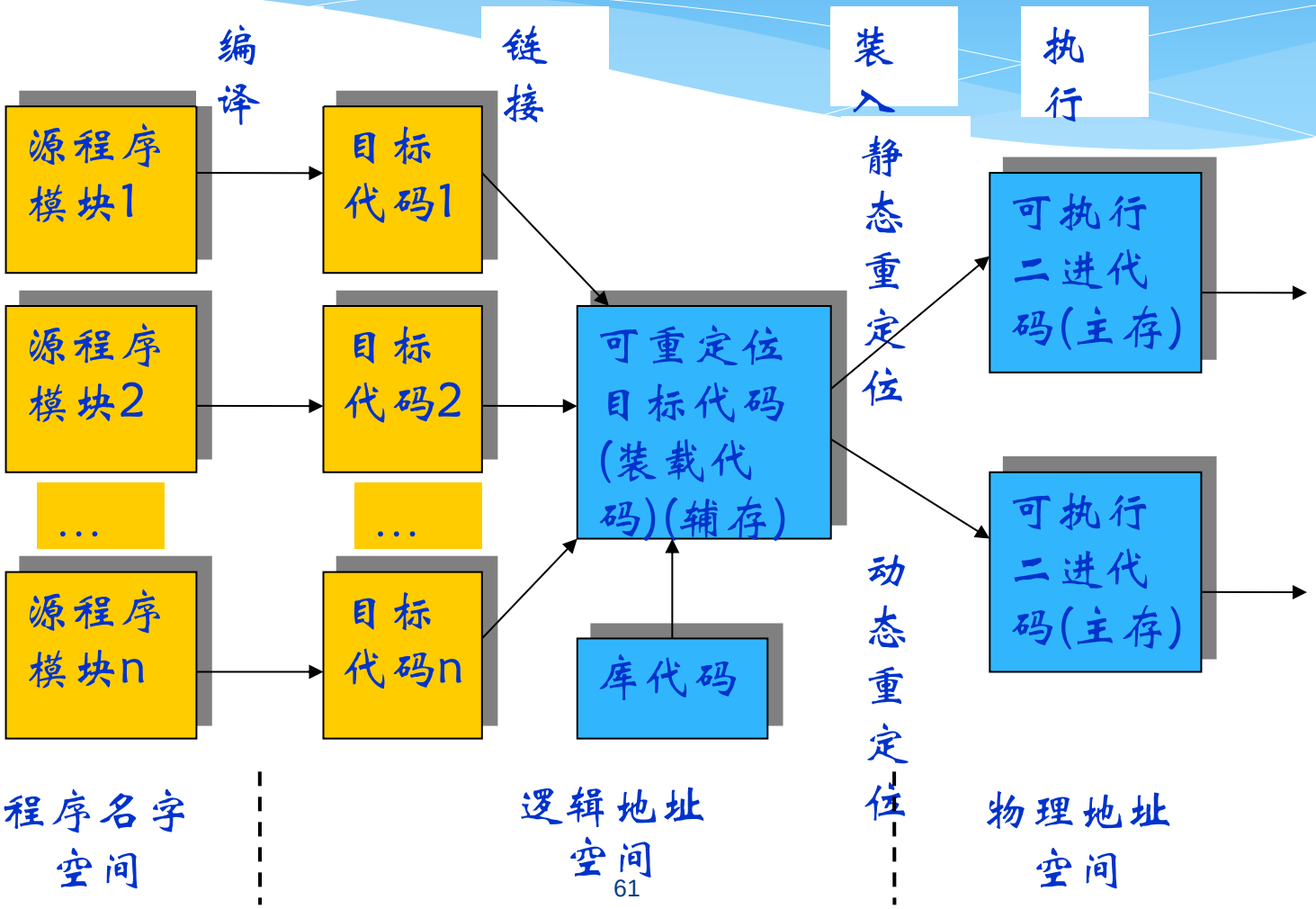


重定位


相对地址
基址寄存器
进程控制块
加法器
程序
绝对地址
界限寄存器
比较器
数据
给操作系统的
中断
栈
主存中的进程映像

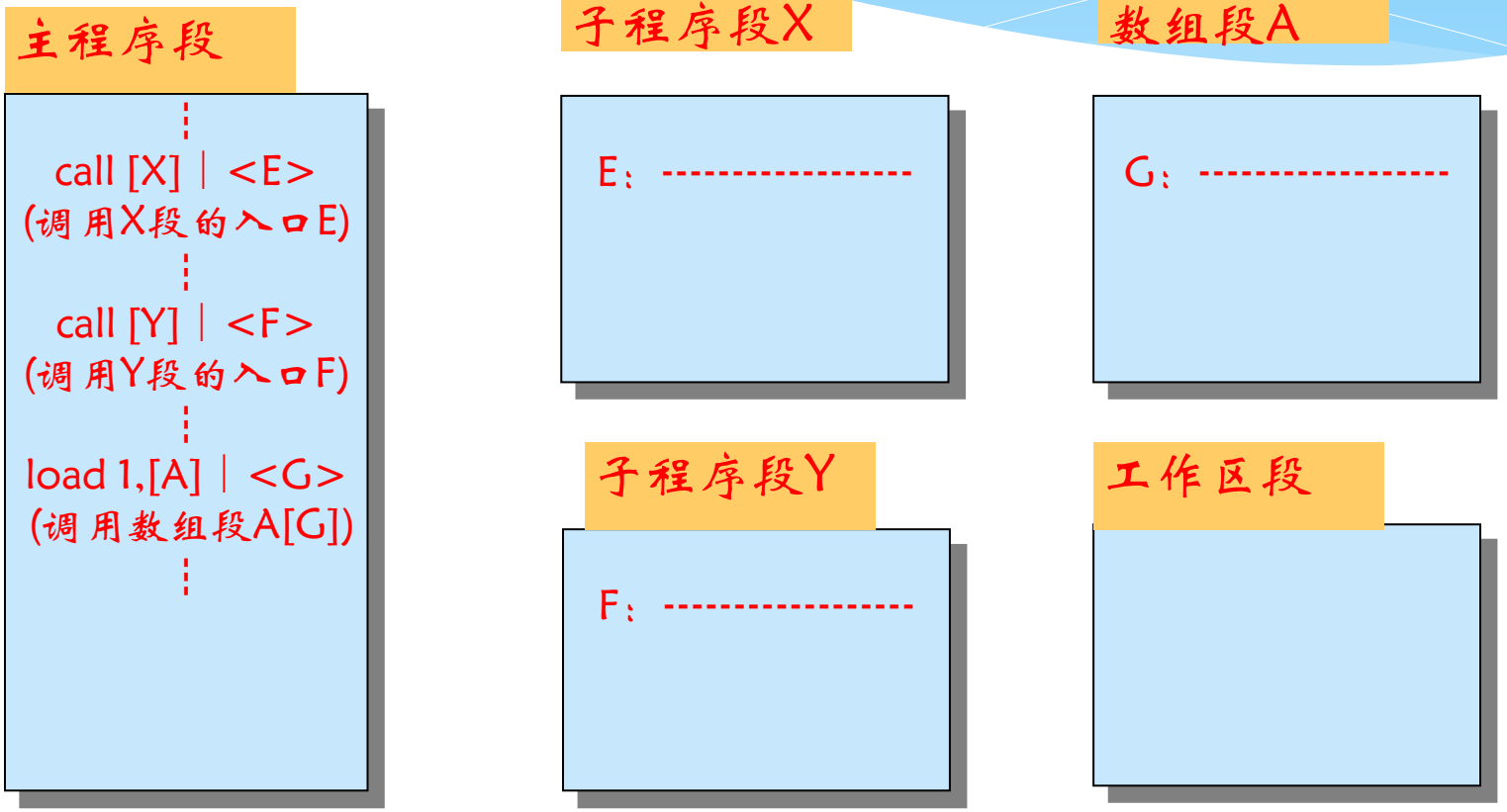
地址转换与存储保护

程序的编译、链接、装入和执行

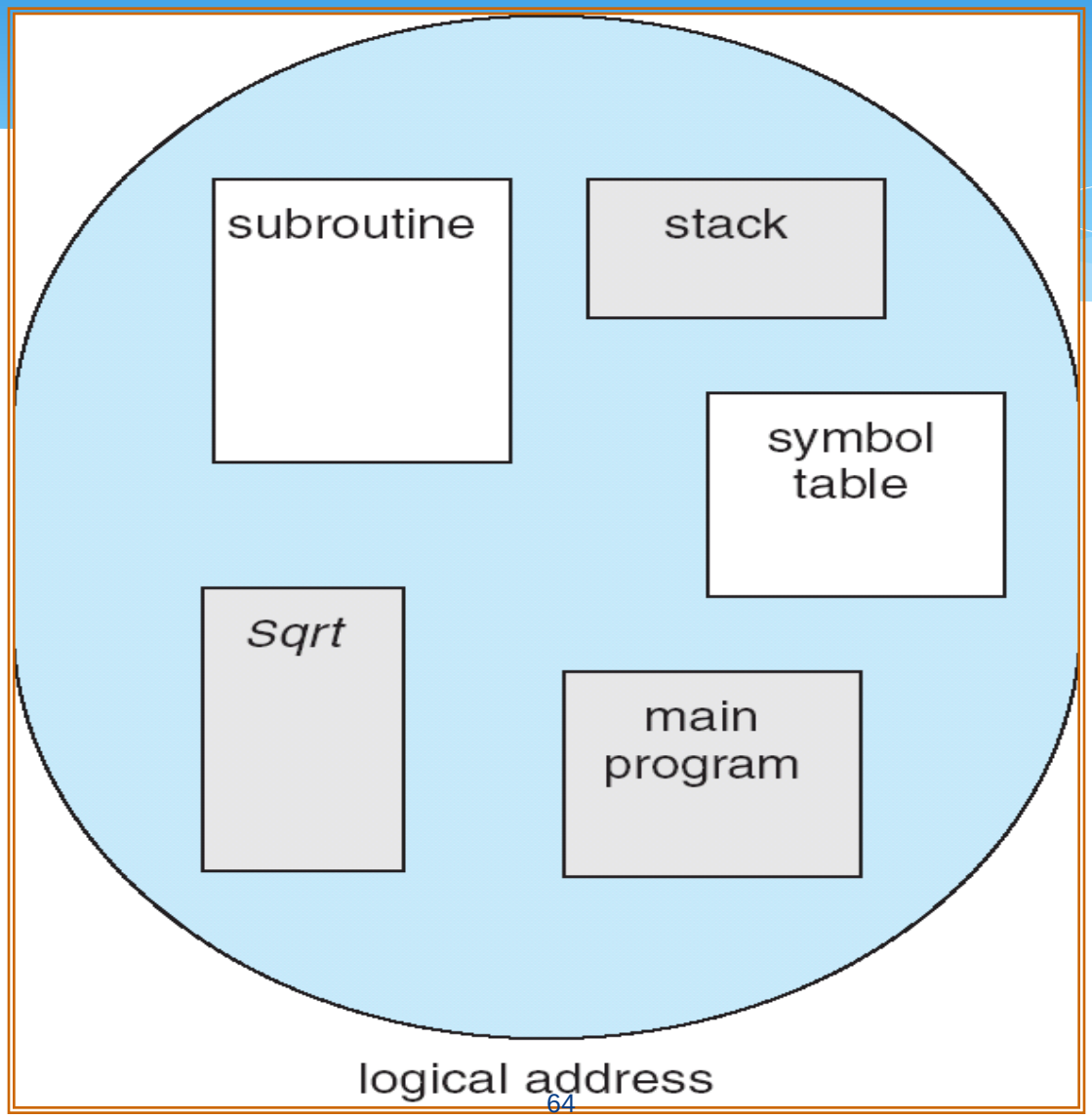


* 固定分区
* 动态分区
* 伙伴系统

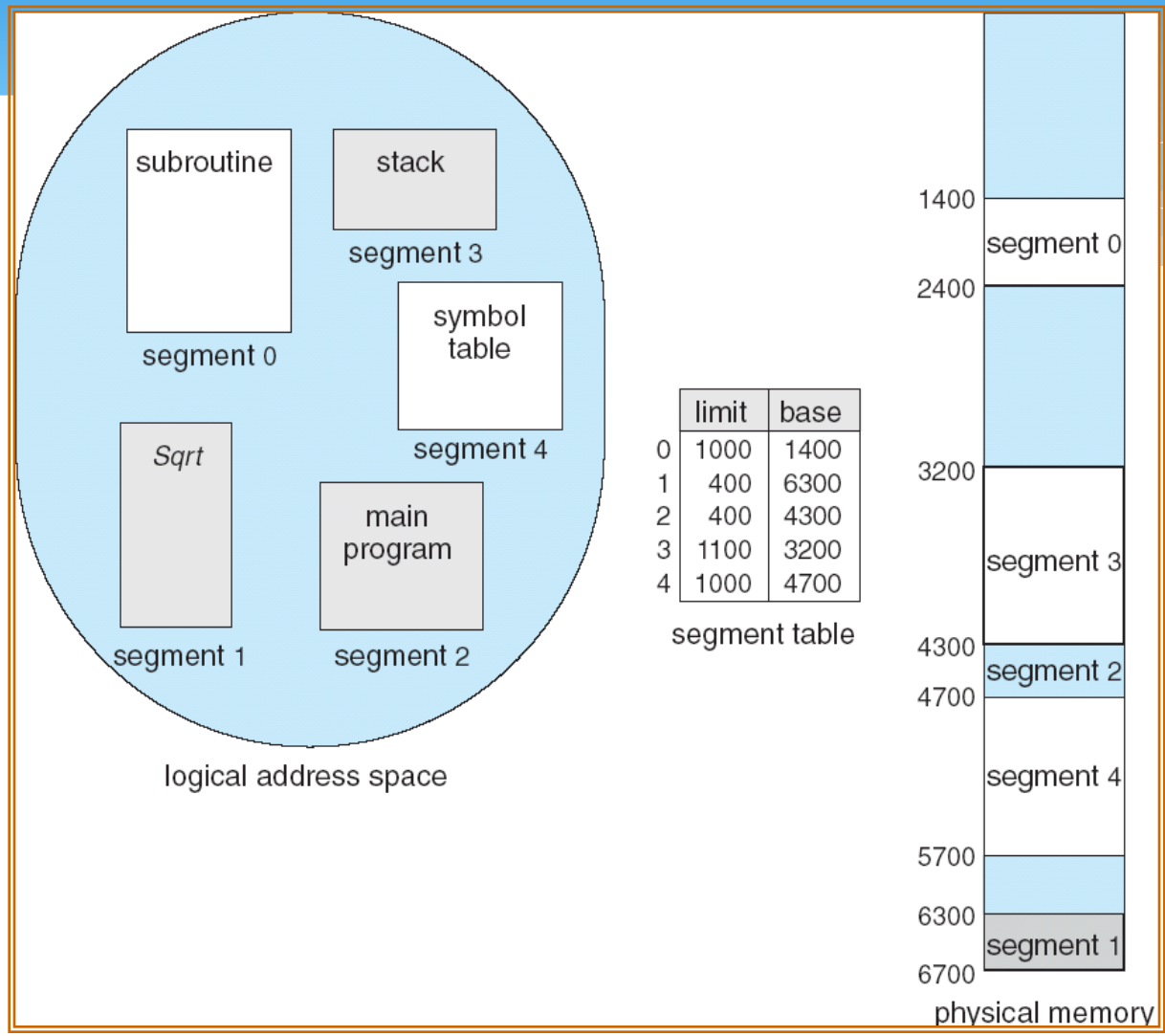
模块化程序设计的分段结构



程序的用户视图

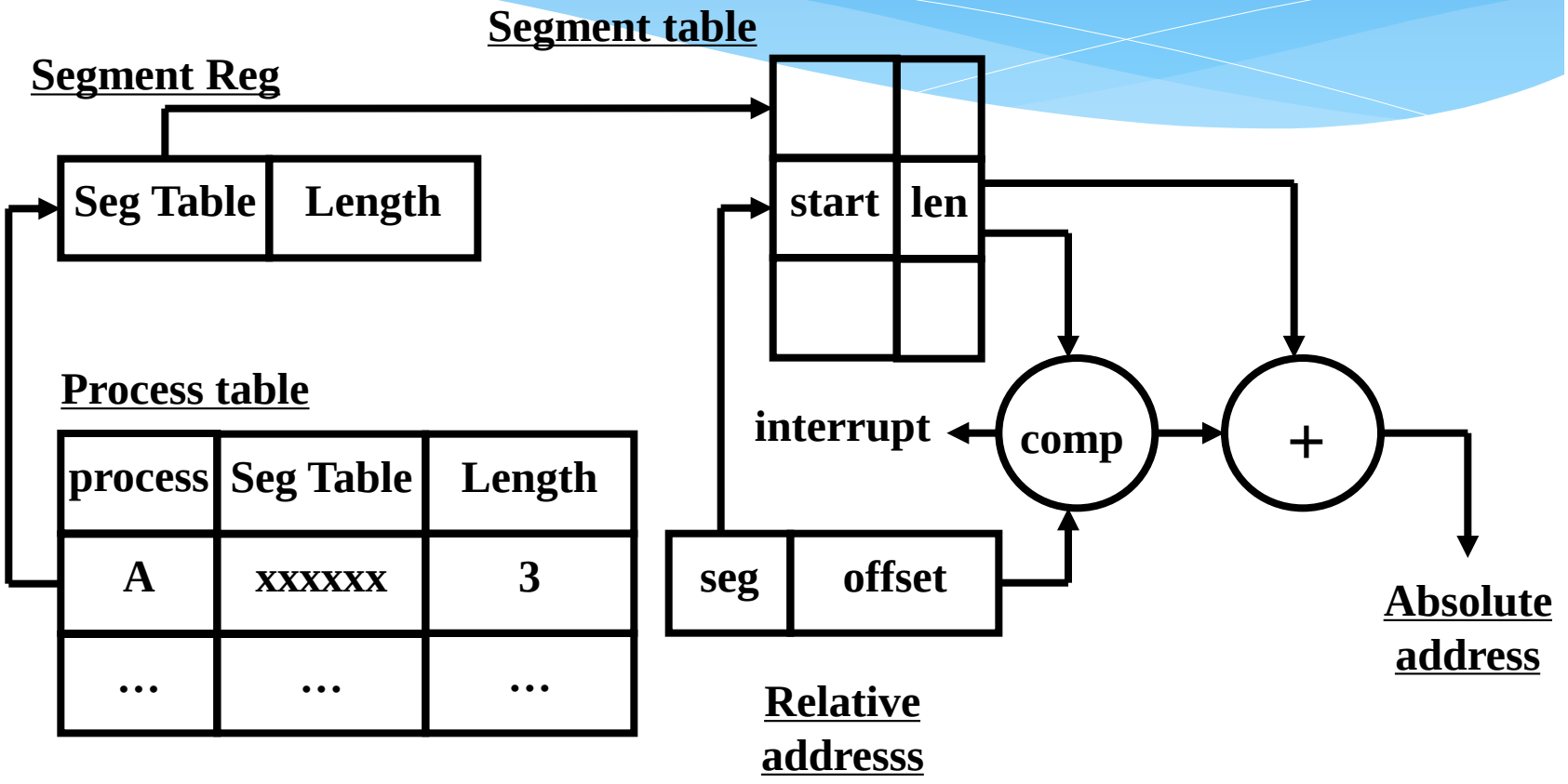

subroutine
stack
symbol table
sqrt
main program
logical address

分段（例）


subroutine
stack
segment 3
symbol table
segment 0
segment 4
Sqrt
main program
segment 1
segment 2
logical address space
limit base
0 1000 1400
1 400 6300
2 400 4300
3 1100 3200
4 1000 4700
segment table
1400
segment 0
2400
3200
segment 3
4300
segment 2
4700
segment 4
5700
6300
segment 1
6700
physical memory

南京大学 NANJING UNIVERSITY

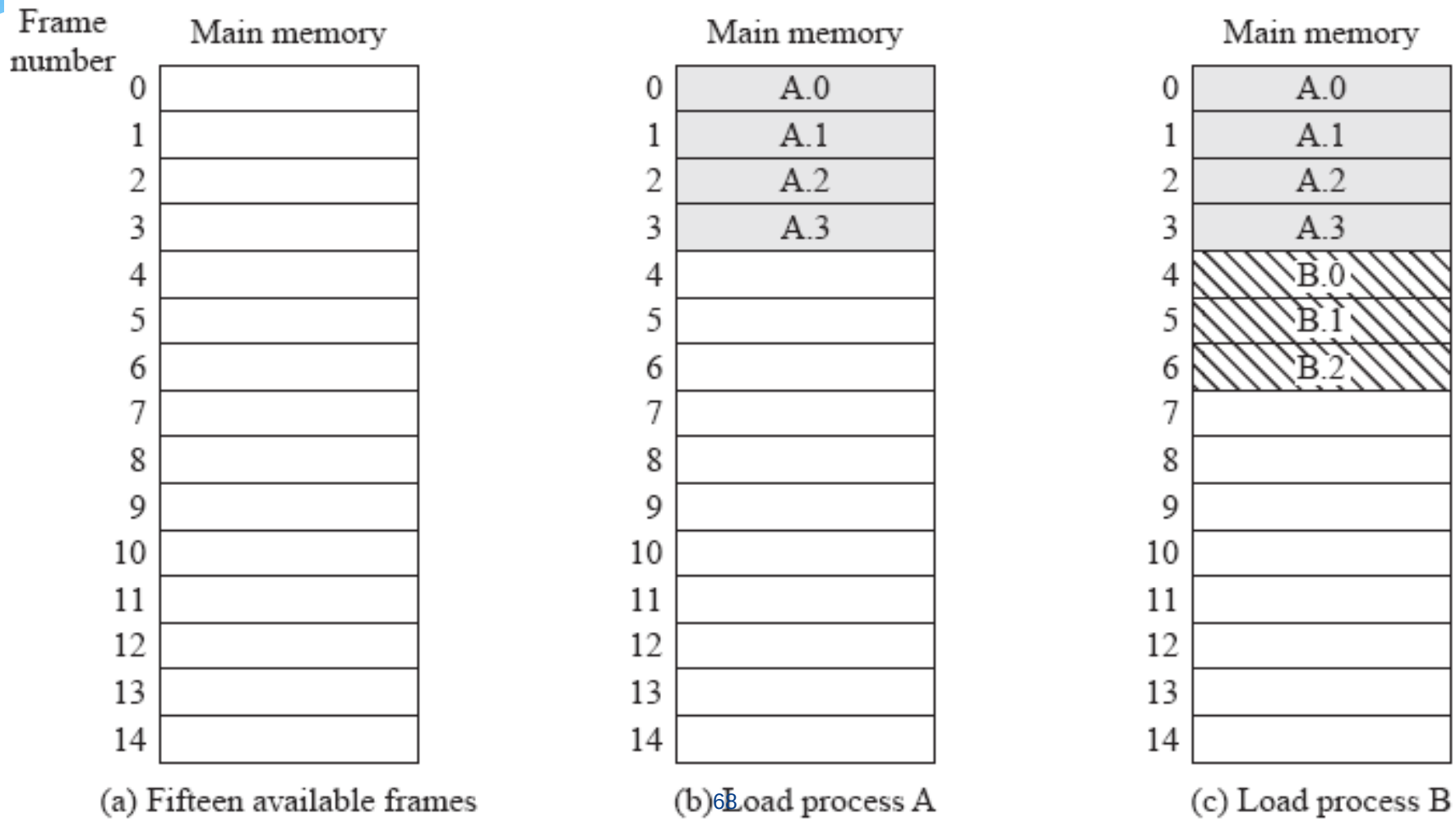
分段的重定位



分页

* 主存被划分成大小固定相等的块，且块相对比较小，每个进程也被分成同样大小的小块，
* 进程中称为页(page)的块可以指定到内存中称为页框(frame)或者页框的可用块
* 仅有一个简单的基址寄存器是不够的，操作系统需要为每个进程维护一个页表(page table)
* 页表给出了该进程的每一页对应的页框的位置
* 每个逻辑地址包括一个页号和在页中的偏移量

指定进程页到空闲页框

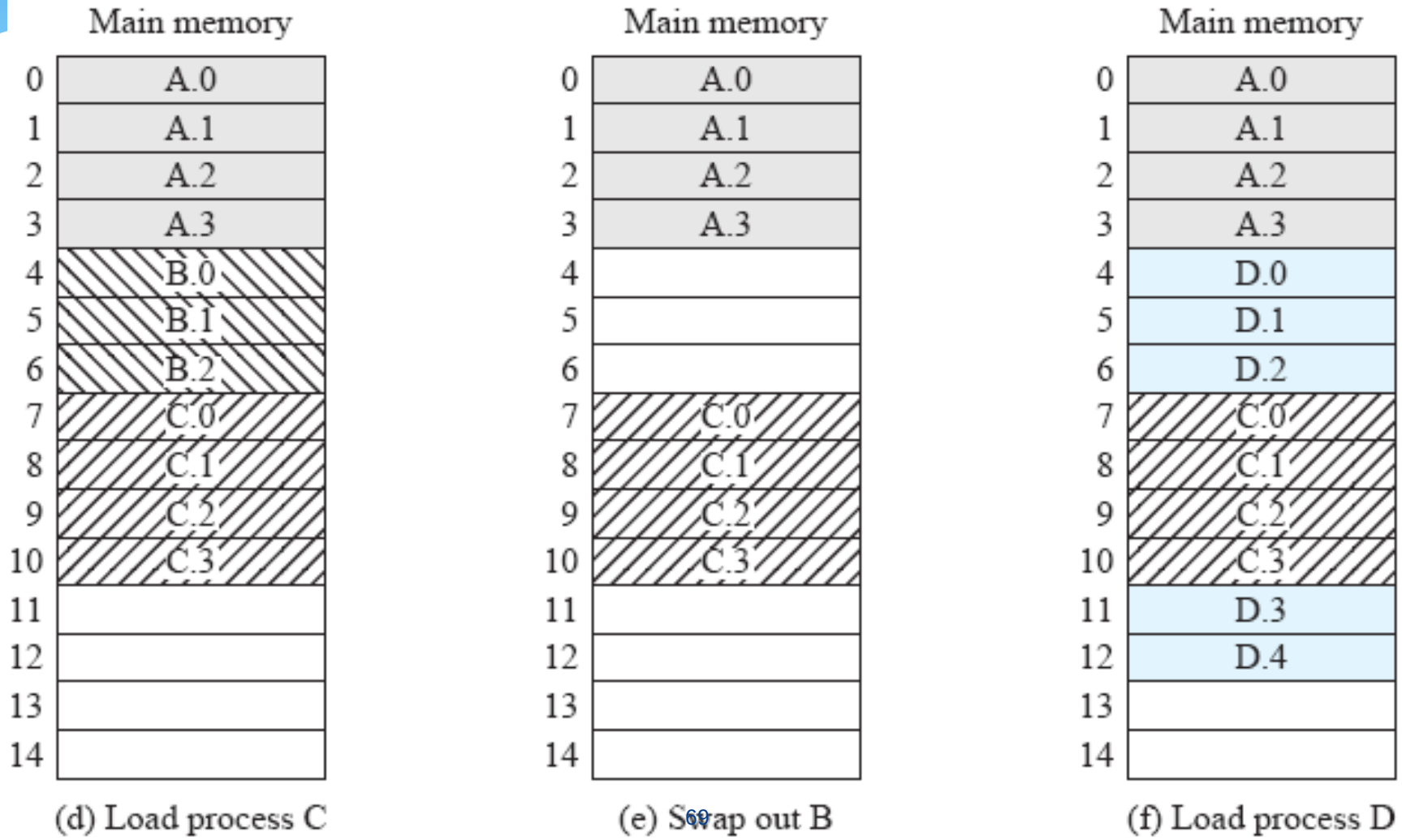


(a) Fifteen available frames

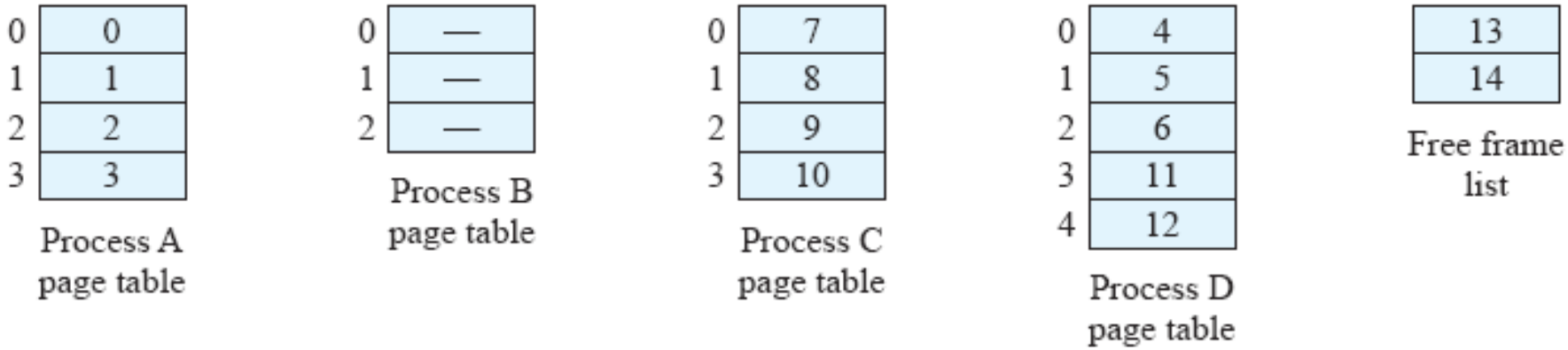
(b) Load process A

(c) Load process B

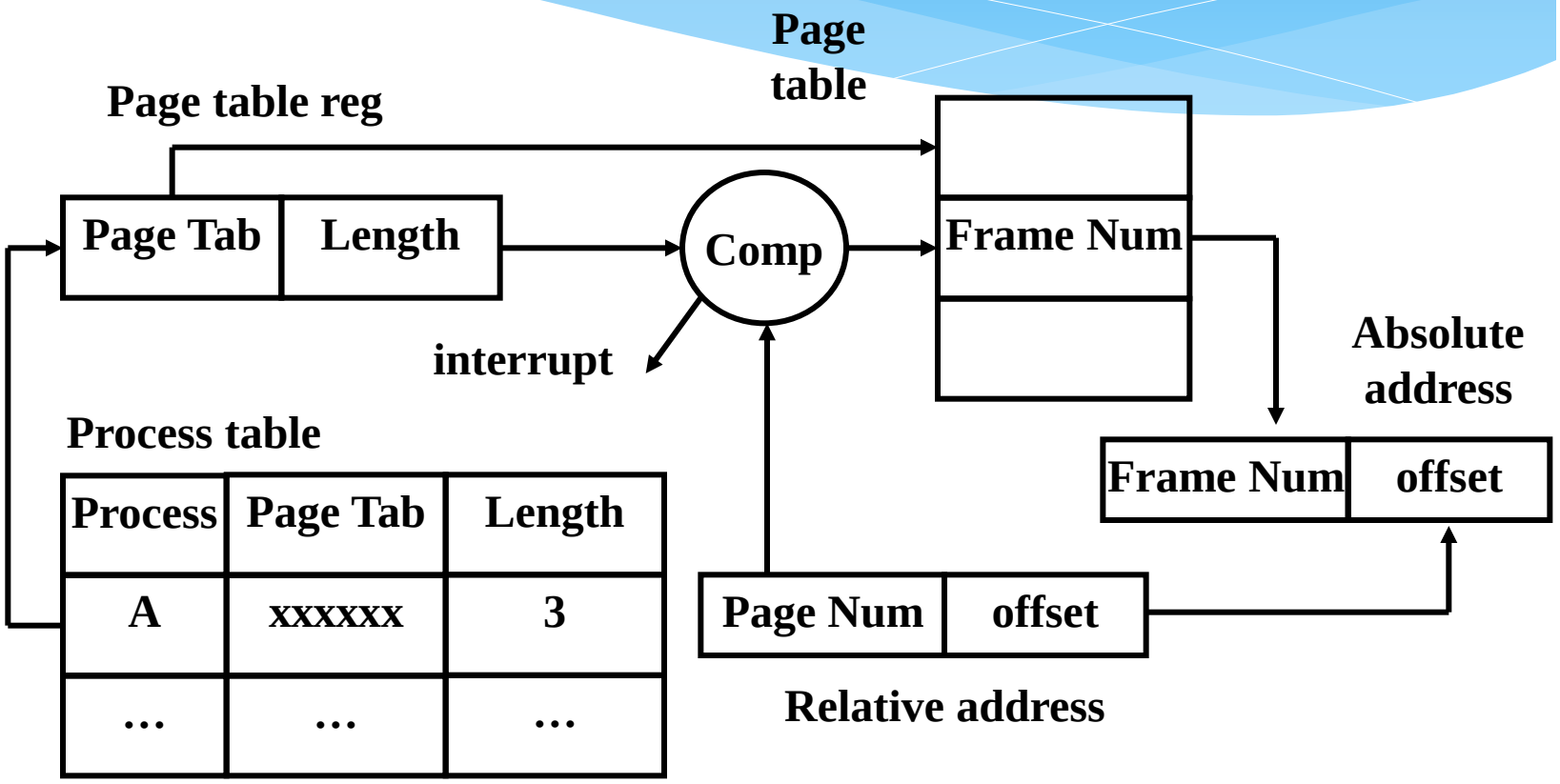
指定进程页到空闲页框


Main memory
0 A.0
1 A.1
2 A.2
3 A.3
4 B.0
5 B.1
6 B.2
7 C.0
8 C.1
9 C.2
10 C.3
11
12
13
14
(d) Load process C
Main memory
0 A.0
1 A.1
2 A.2
3 A.3
4
5
6
7 C.0
8 C.1
9 C.2
10 C.3
11
12
13
14
(e) Swap out B
Main memory
0 A.0
1 A.1
2 A.2
3 A.3
4 D.0
5 D.1
6 D.2
7 C.0
8 C.1
9 C.2
10 C.3
11 D.3
12 D.4
13
14
(f) Load process D

页表


0 0
1 1
2 2
3 3
Process A page table
0 —
1 —
2 —
Process B page table
0 7
1 8
2 9
3 10
Process C page table
0 4
1 5
2 6
3 11
4 12
Process D page table
13
14
Free frame list

分页的重定位

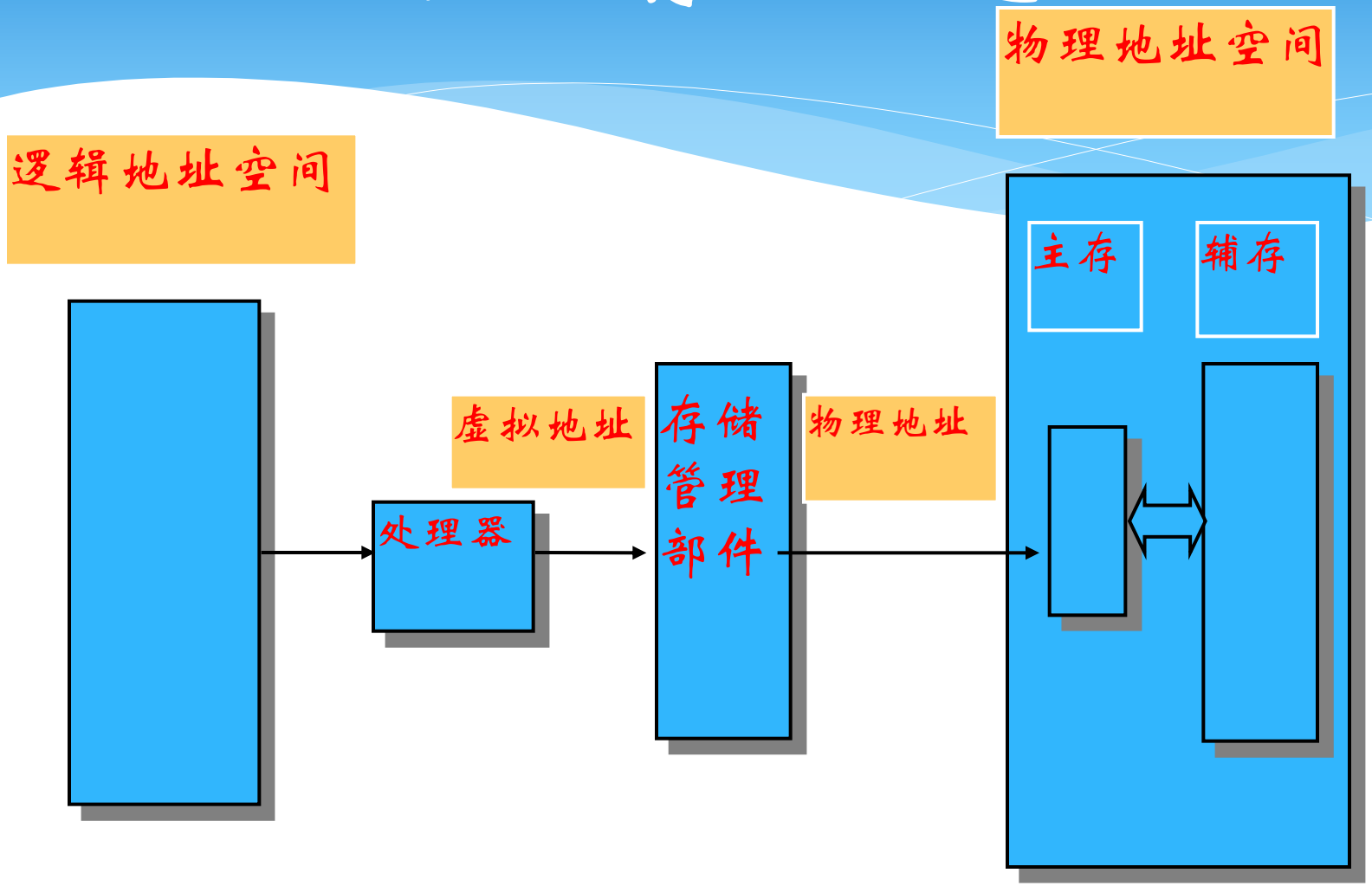

Page table
Page table reg
Page Tab
Length
Comp
Frame Num
interrupt
Absolute address
Process table
Process
Page Tab
Length
A
xxxxxx
3
…
…
…
Frame Num
offset
Page Num
offset
Relative address

南京大学
NANJING UNIVERSITY

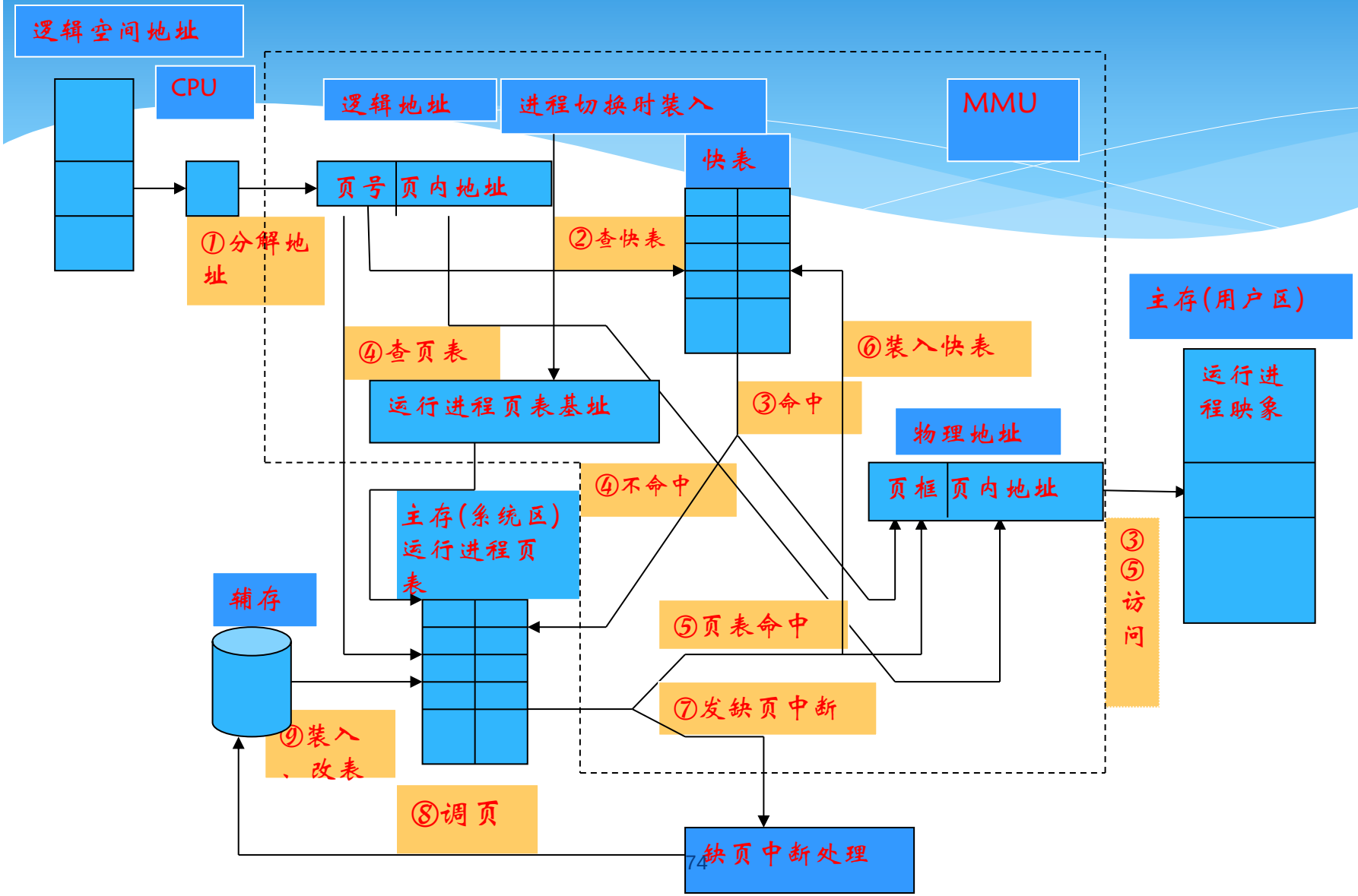
虚拟存储管理
本主题教学目标

1. 了解虚拟存储管理的基本特点(程序局部性原理)
2. 掌握页式虚拟存储管理
3. 掌握段式虚拟存储管理
4. 掌握页面调度策略与算法
5. 了解Pentium的地址转换机构

虚拟存储概念

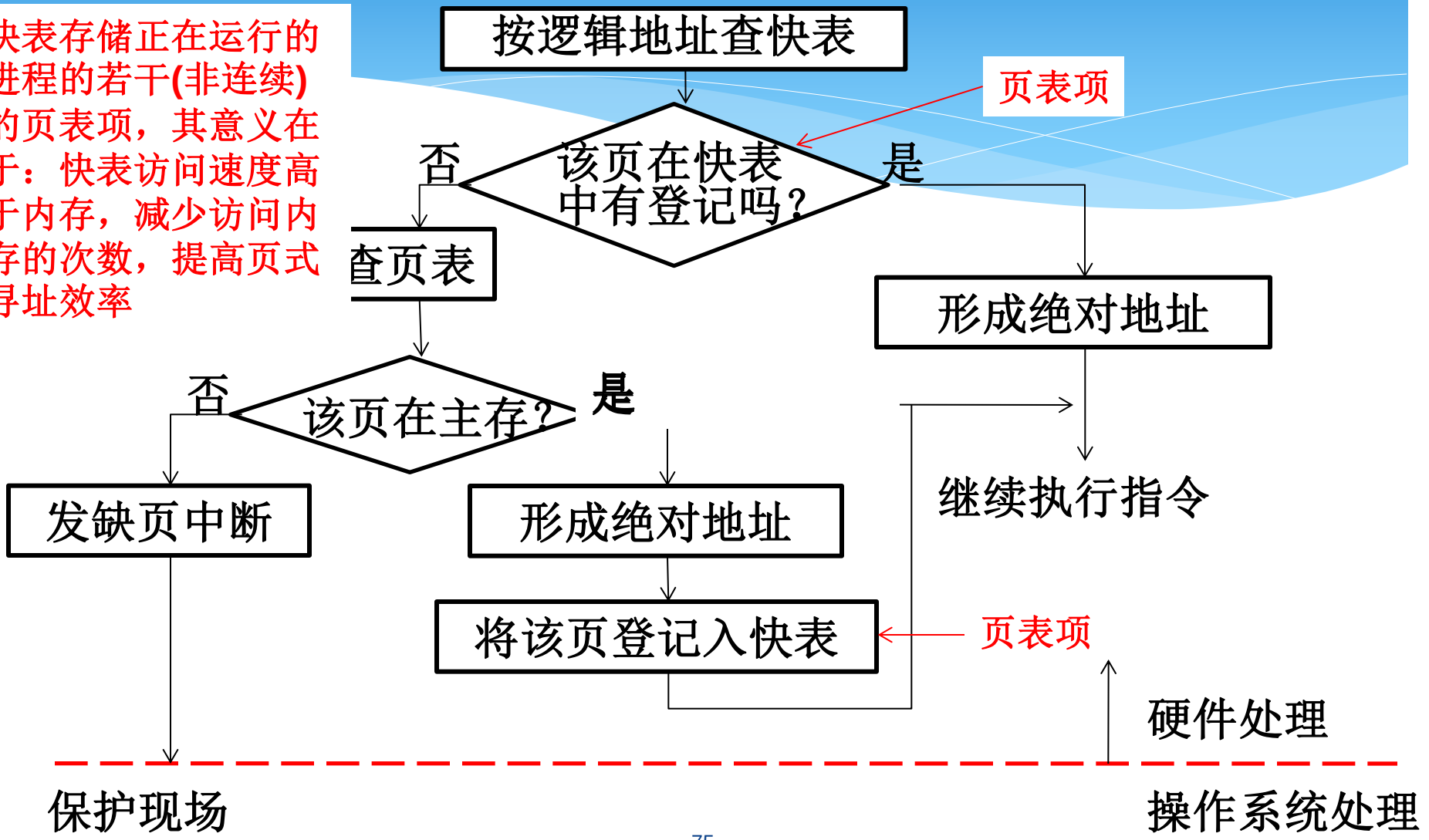


请求分页虚存地址转换过程(1)

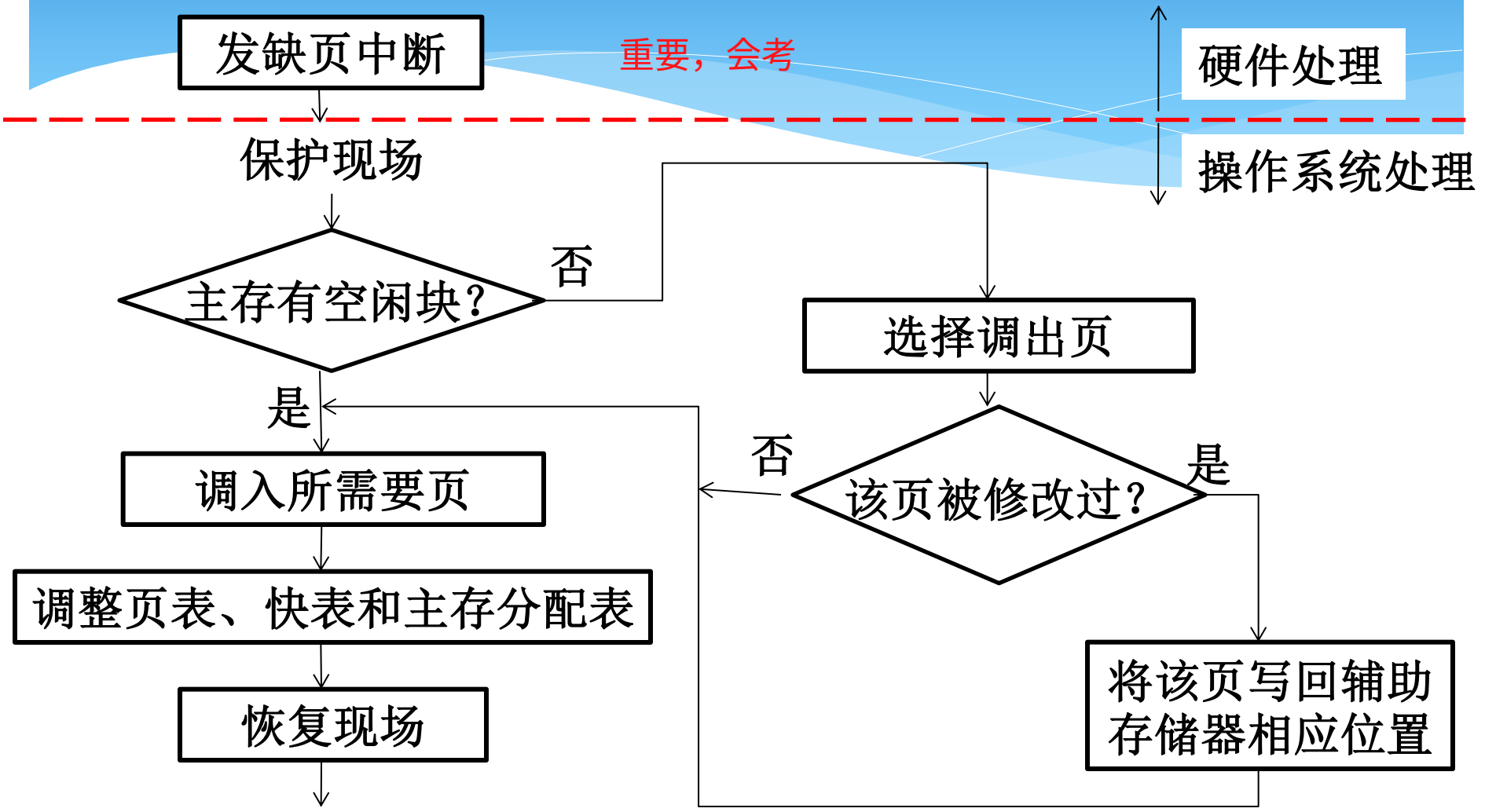


缺页中断的处理流程

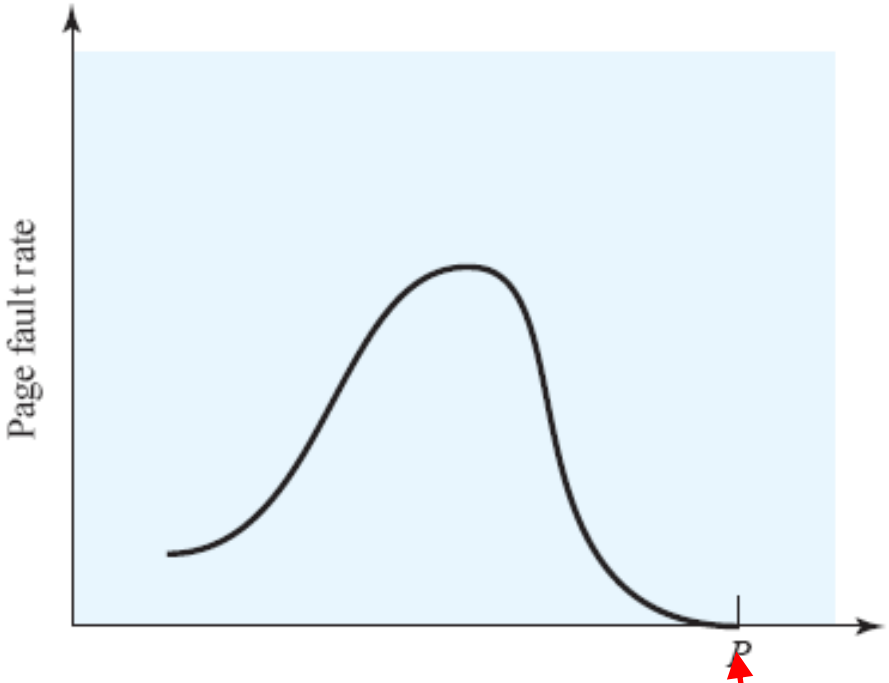
快表存储正在运行的进程的若干(非连续)的页表项，其意义在于：快表访问速度高于内存，减少访问内存的次数，提高页式寻址效率



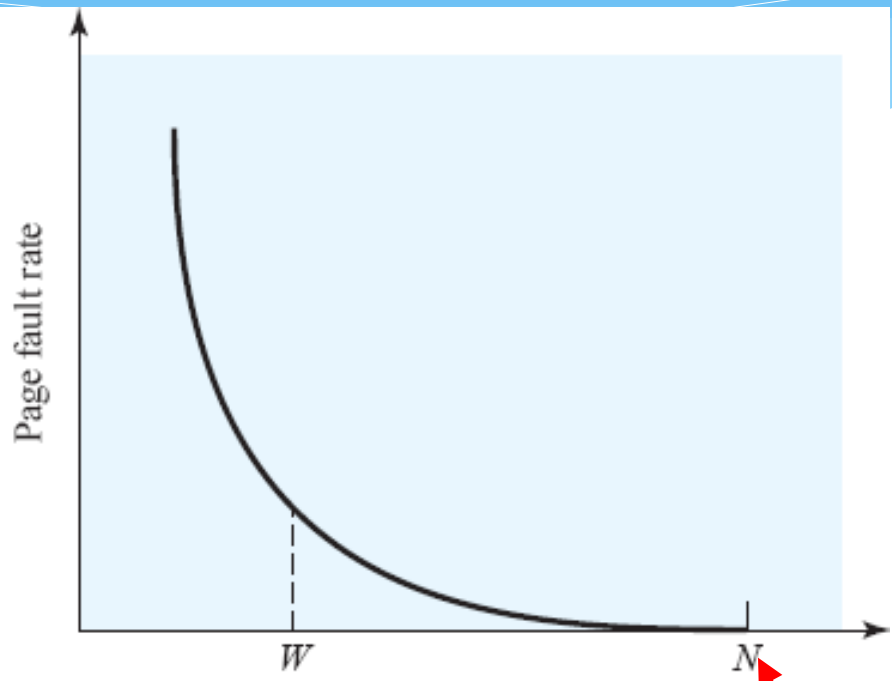
缺页中断的处理流程


发缺页中断
重要，会考
硬件处理
操作系统处理
保护现场
主存有空闲块？
否
是
选择调出页
该页被修改过？
否
是
调入所需要页
调整页表、快表和主存分配表
恢复现场
将该页写回辅助存储器相应位置

页尺寸



(a) Page size



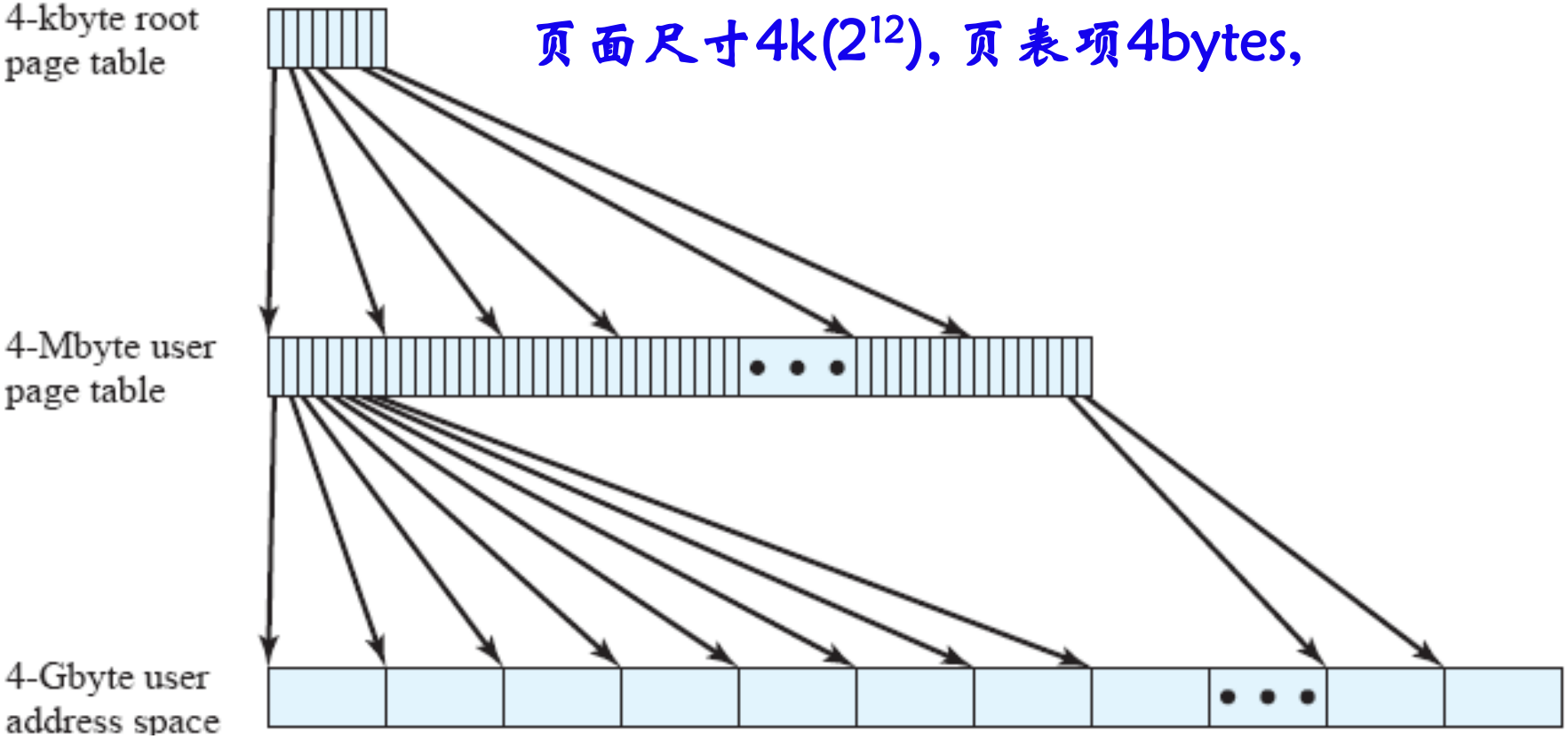
(b) Number of page frames allocated

退化为固定分区

极限为页面全装载

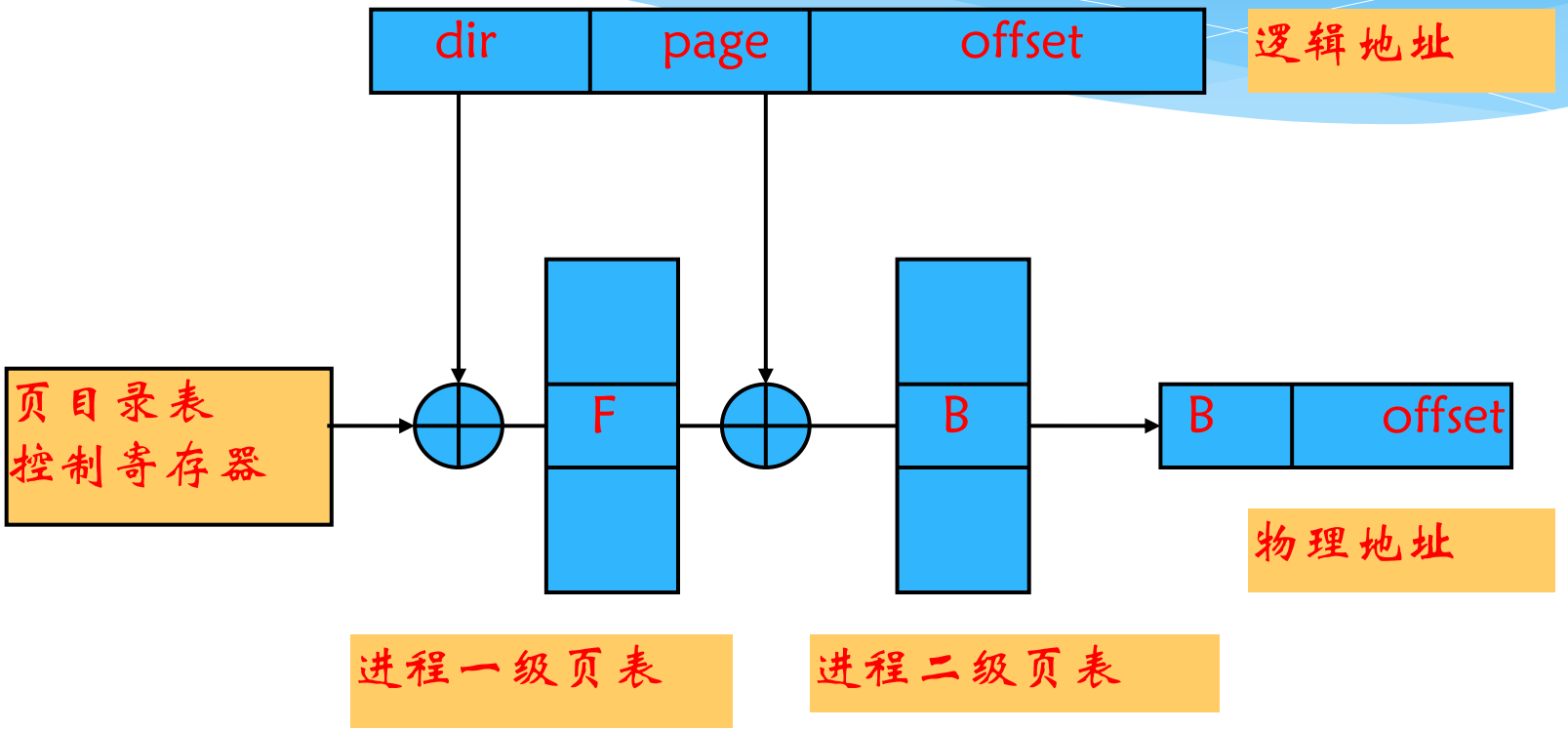
- P = size of entire process
- W = working set size
- N = total number of pages in process

两级页表(32位地址)

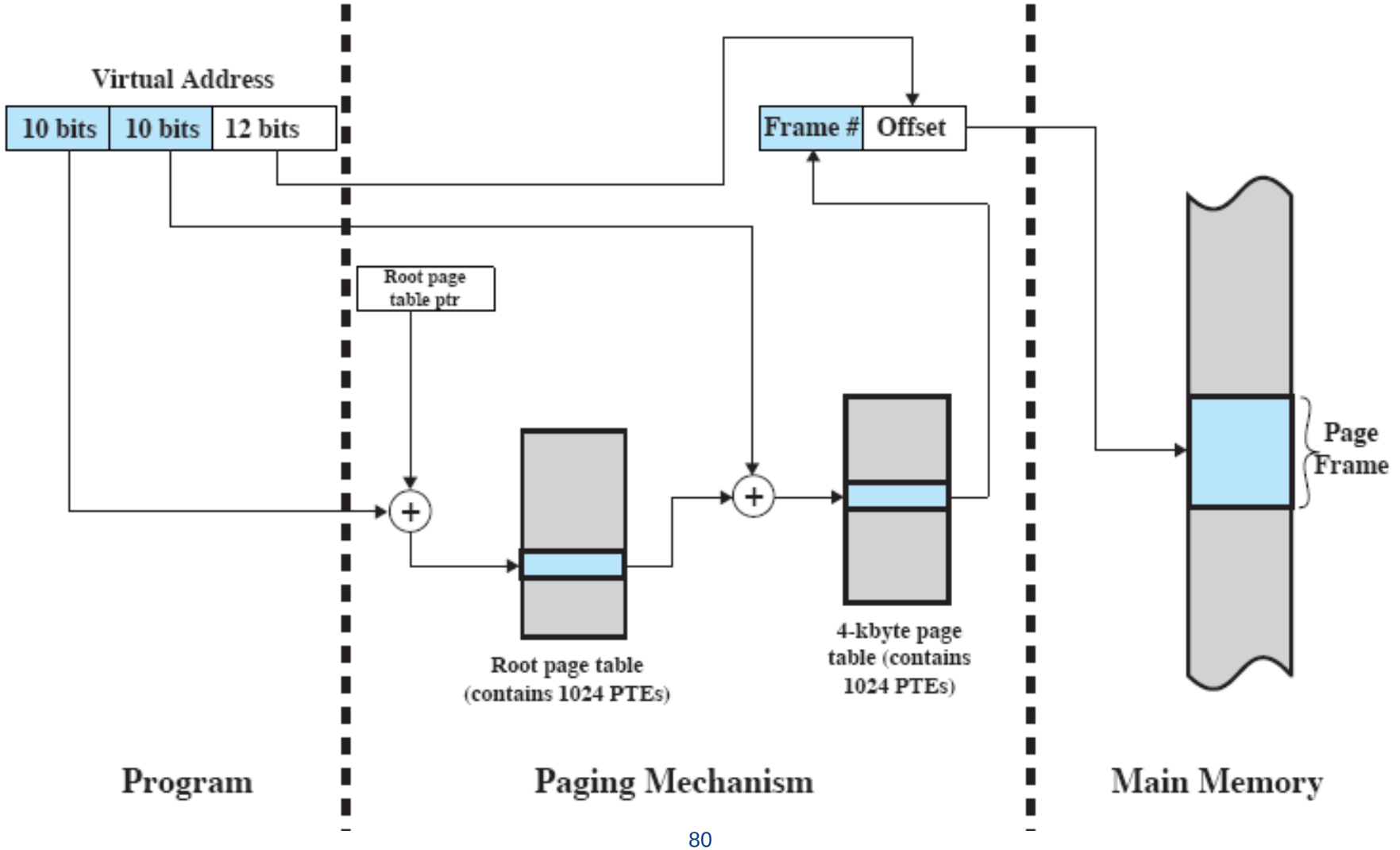


页面尺寸4k(2^{12})，页表项4bytes，

多级页表地址转换过程


dir
page
offset
逻辑地址
页目录表
控制寄存器
F
B
B
offset
物理地址
进程一级页表
进程二级页表

Two-Level Scheme for 32-bit Address



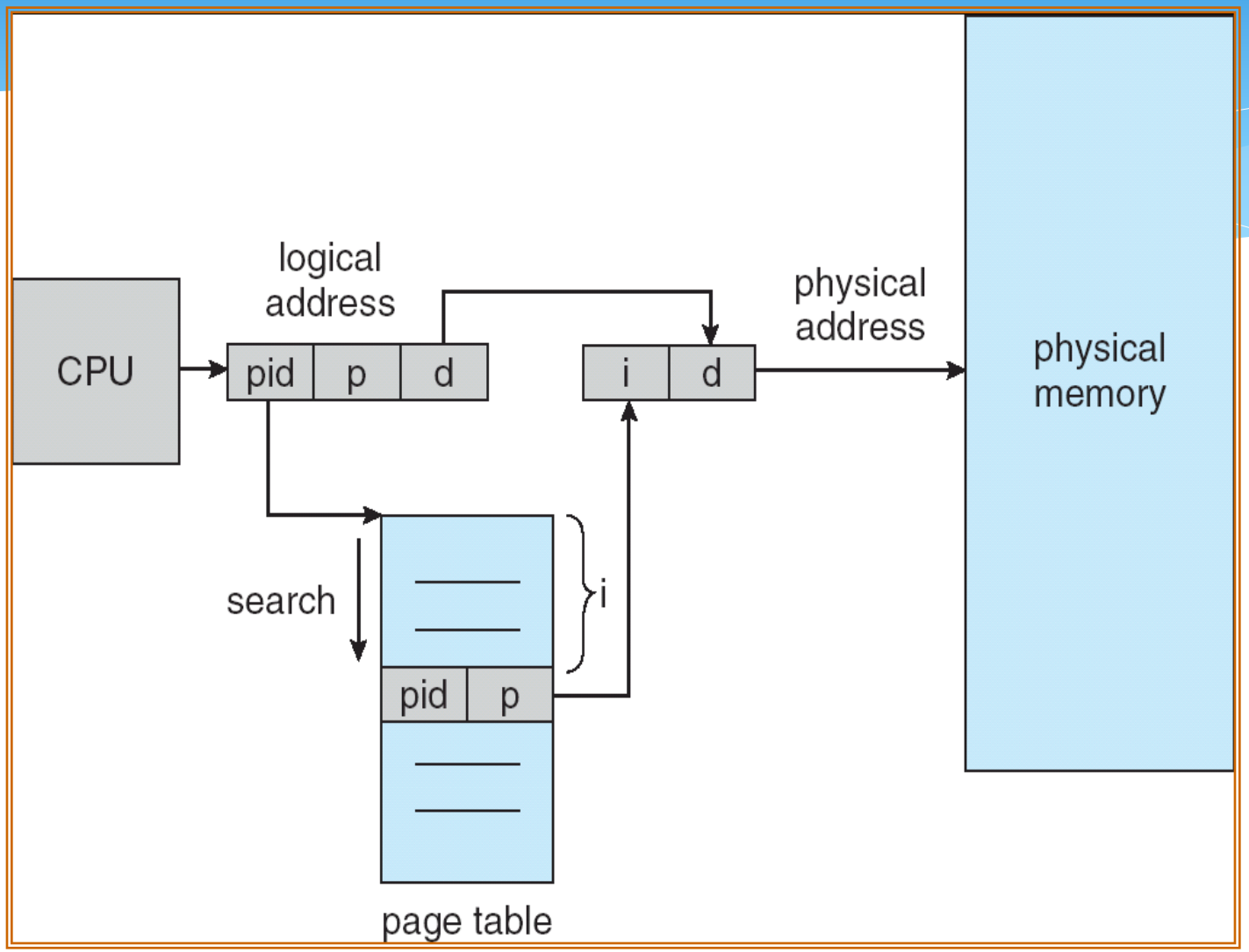
反置页表

* 页表设计的一个重要缺陷是页表的大小与虚拟地址空间的大小成正比
* 在反向页表方法中，虚拟地址的页号部分使用一个简单散列函数映射到哈希表中。哈希表包含一个指向反向表的指针，而反向表中含有页表项。
* 通过这个结构，哈希表和反向表中只有一项对应于一个实存页(面向实存)，而不是虚拟页(面向虚存)。
* 因此，不论由多少进程、支持多少虚拟页，页表都只需要实存中的一个固定部分。
* PowerPC, UltraSPARC, and IA-64

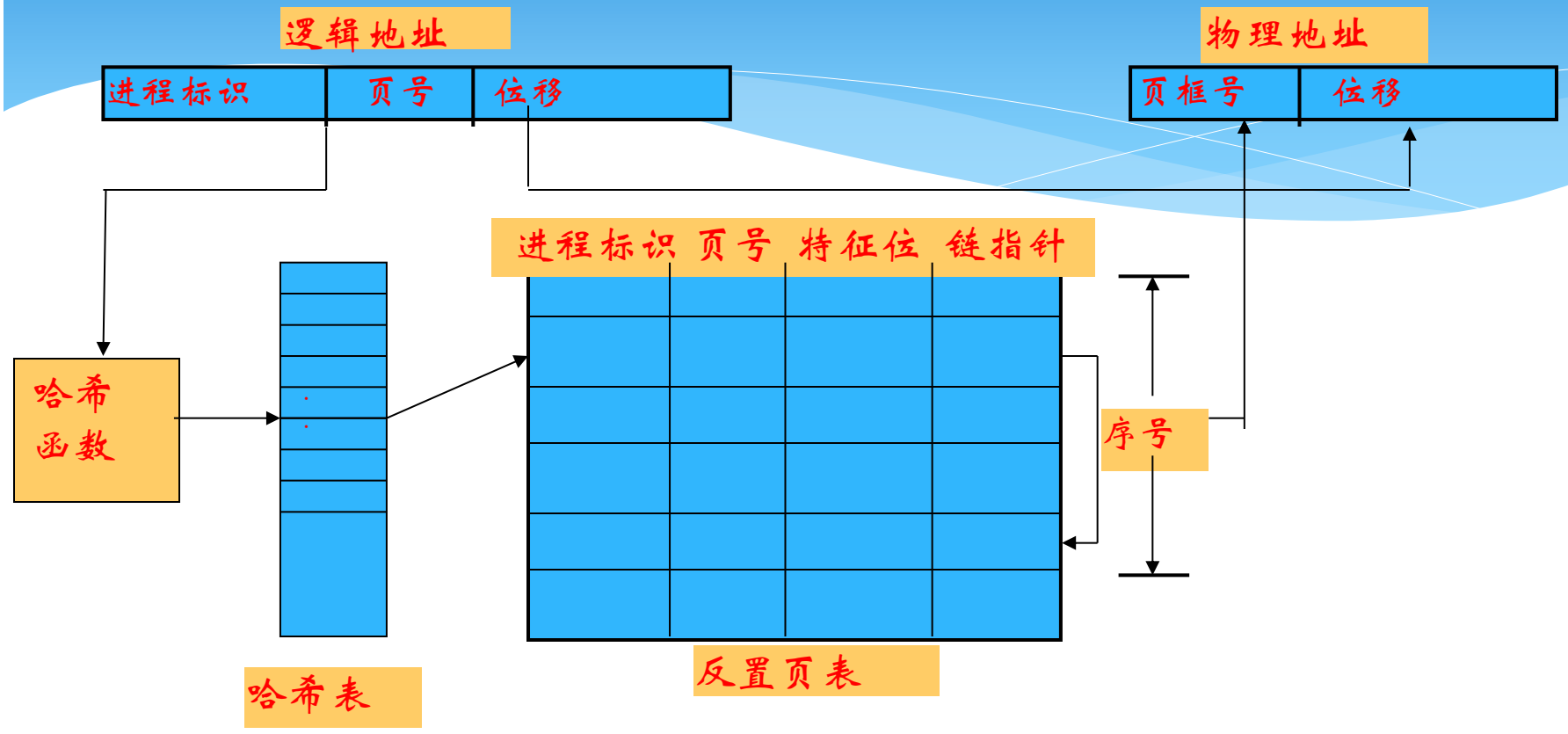
反置页表

* **页号**：虚拟地址页号部分。
* **进程标志符**：使用该页的进程。页号和进程标志符结合起来标志一个特定的进程的虚拟地址空间的一页。
* **控制位**：该域包含一些标记，比如有效、访问和修改，以及保护和锁定的信息。
* **链指针**：如果某个项没有链项，则该域为空(允许用一个单独的位来表示)。

反置页表的结构


CPU
logical address
pid
p
d
i
d
physical address
physical memory
search
i
pid
p
page table

反置页表



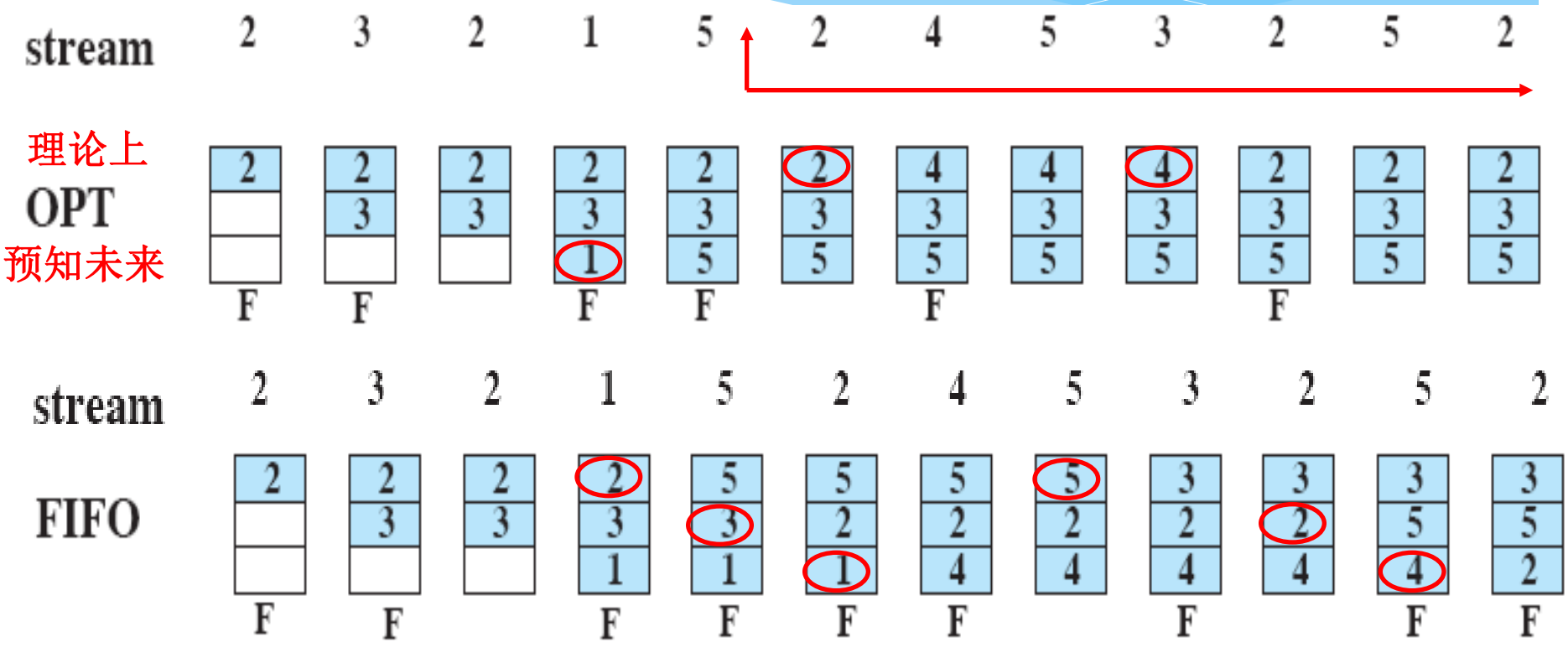
反置页表及其地址转换

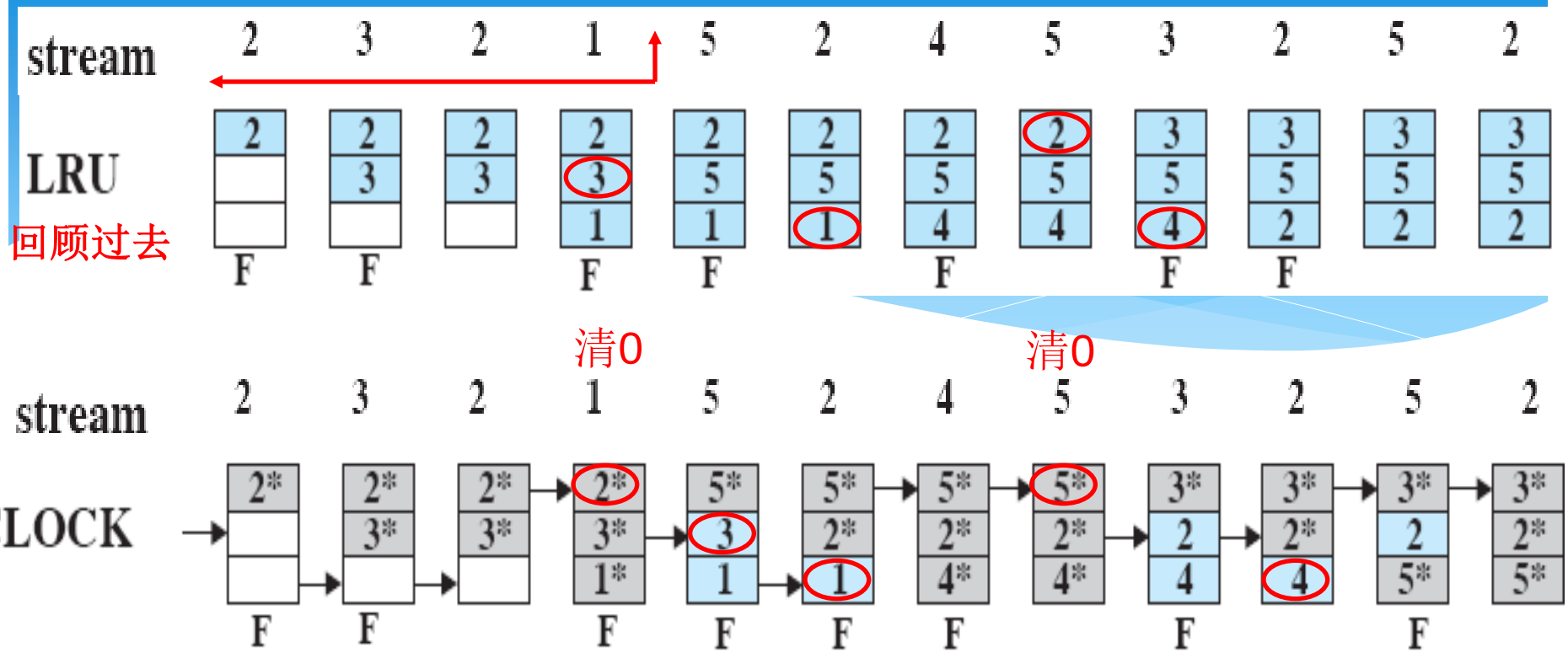
虚拟分页的操作系统软件

* (1) 读取策略
* (2) 放置策略
* (3) 替换策略
* (4) 驻留集管理
* (5) 清除策略

南京大学
NANJING UNIVERSITY

示例:页面替换算法





星号表示相应的使用位等于1，箭头表示指针的当前位置。

当一页被替换时，指向下一帧。 虽然早就进来，但是最近使用过，所以不急着替换

当需要替换一页时，扫描缓冲区，查找使用位被置为0的一帧。

每当遇到一个使用位为1的帧时，就将该位重新置为0；

如果在这个过程开始时，所有帧的使用位均为0，选择遇到的第一个帧替换；

如果所有帧的使用位为1，则指针在缓冲区中完整地循环一周，把所有使用位都置为0，并且停留在最初的位置上，替换该帧中的页。

Belady's Anomaly (Belady异常)

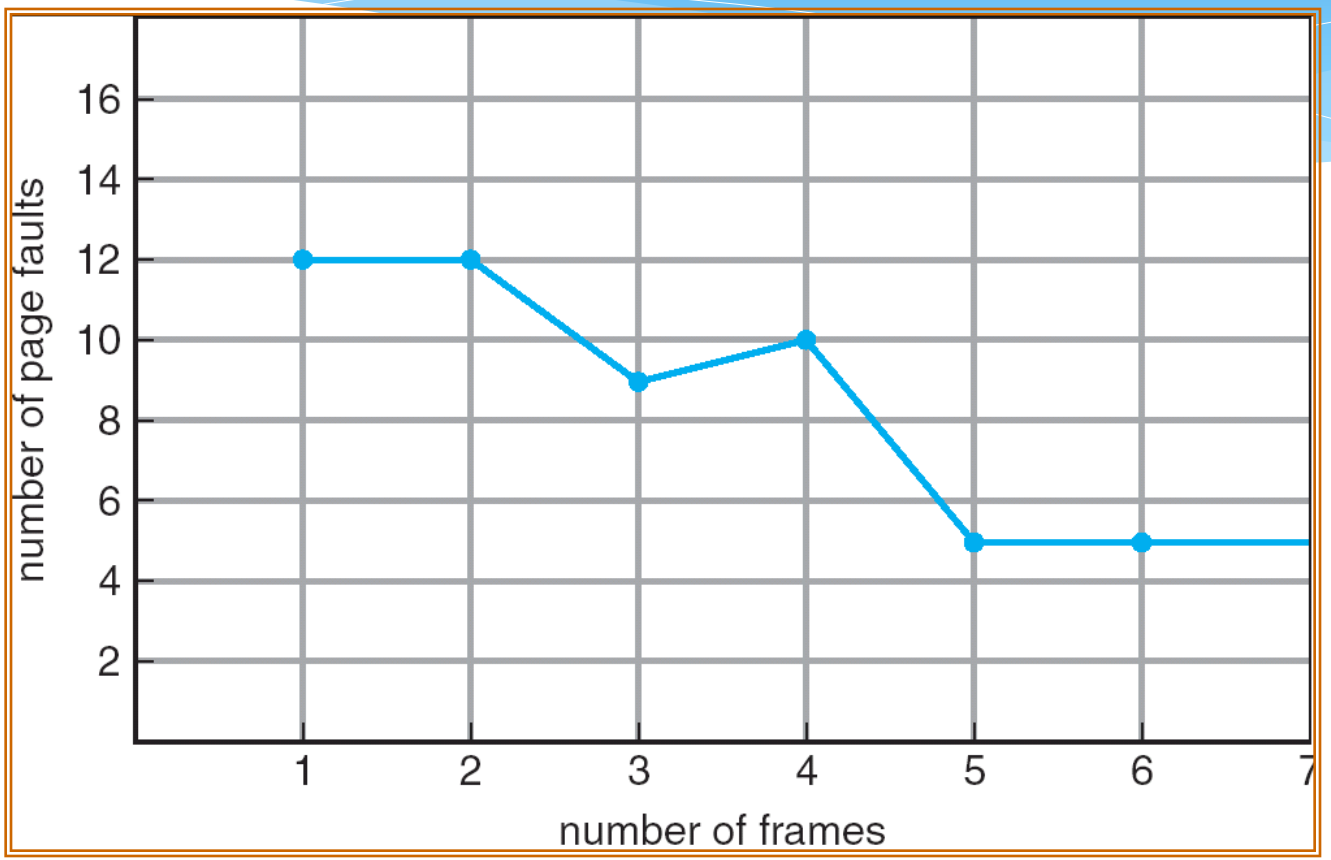
Belady 异常

FIFO	1	2	3	4	1	2	5	1	2	3	4	5

more frames ⇒ more page faults

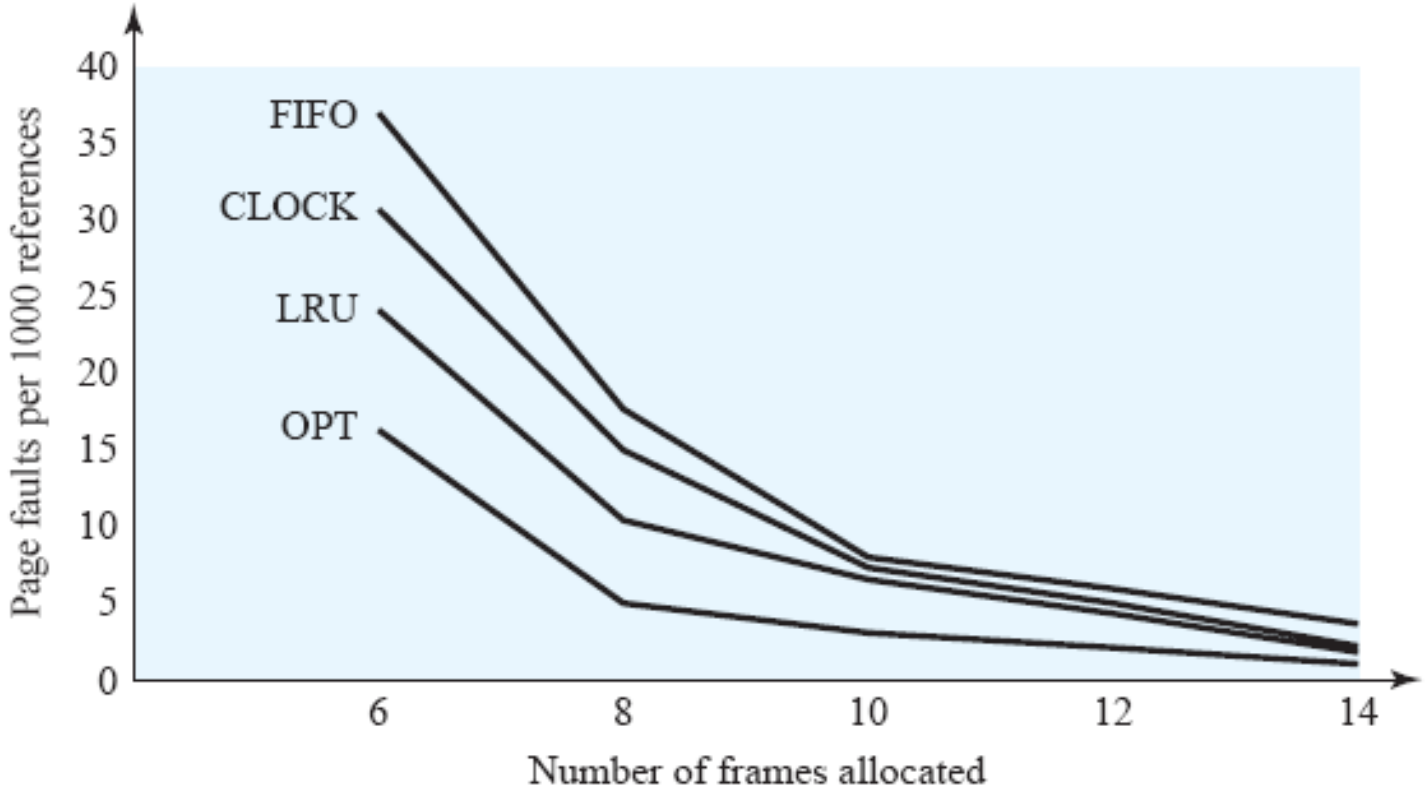
对应《操作系统教程(第5版)》pp.225 “Belady异常”

FIFO Illustrating Belady's Anomaly (FIFO算法的Belady异常)



对应《操作系统教程(第5版)》pp.265 “Belady异常”

基本替换算法的比较



这是一个整体的性能比较，对于个案，有可能Clock优于LRU

南京大学
NANJING UNIVERSITY

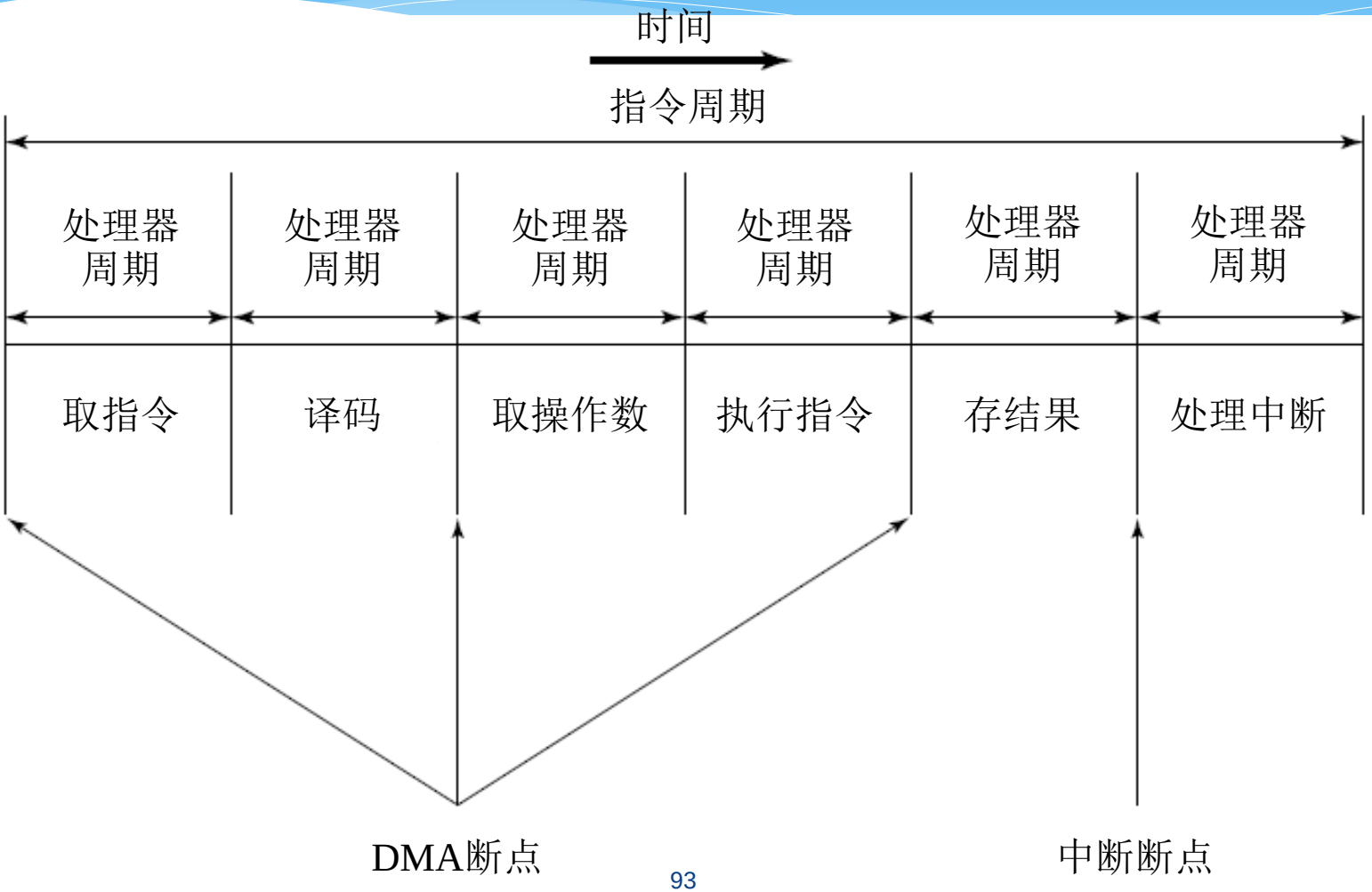
第四章 设备管理
本主题教学目标

1. 复习、了解I/O设备、I/O控制方式
2. 掌握I/O缓冲区的设计
3. 掌握磁盘调度
4. 掌握I/O软件系统的设计与实现
5. 掌握虚拟设备

I/O控制方式

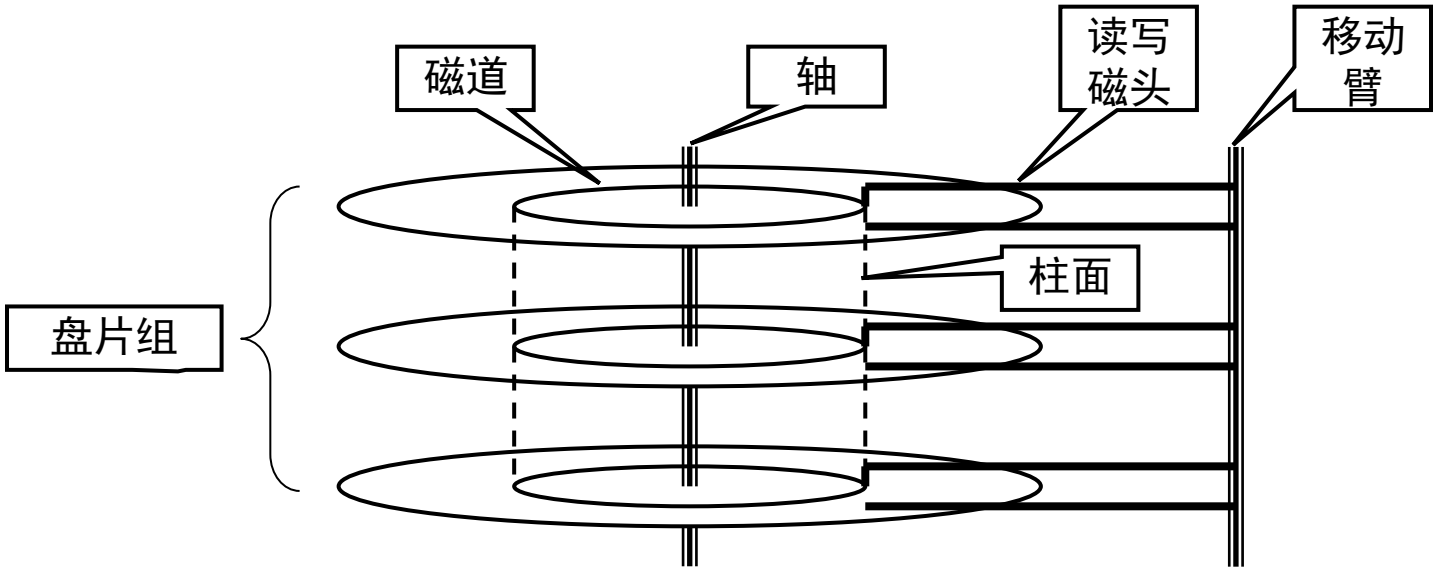
* 程序控制I/O (轮询方式)
  * 处理器代表给I/O模块发送一个I/O命令，该进程进入忙式等待(busy-waiting)，等待操作的完成，然后才可以继续操作
* 中断驱动 I/O
  * 处理器代表进程向I/O模块发出一个I/O命令，然后继续执行后续指令，当I/O模块完成工作后，处理器被该模块中断
  * 如果该进程不需要等待I/O完成，则后续指令可以仍是该进程中的指令，否则，该进程在这个中断上挂起，处理器执行其他工作
* 直接存储器访问(DMA)
  * 一个DMA模块控制主存和I/O模块之间的数据交换。为传送一块数据，处理器给DMA模块发请求，只有当整个数据块传送结束后，处理器才被中断

DMA


时间
指令周期
处理器周期
处理器周期
处理器周期
处理器周期
处理器周期
处理器周期
取指令
译码
取操作数
执行指令
存结果
处理中断
DMA断点
中断断点

磁盘

* 磁盘结构
  * 一个磁道分为固定多个扇区
  * 磁盘物理块地址:(柱面号,磁头号,扇区号)
  * 磁盘读写方式:移臂 → 旋转 → 读写



磁盘调度策略

* 扫描 (SCAN)
  * 要求磁头臂仅仅沿一个方向移动，并在途中满足所有为完成的请求，直到它到达这个方向上的最后一个磁道，或者在这个方向上没有其他请求为止，后一种改进有时候称为LOOK策略
  * 接着反转服务方向，沿着相反方向扫描，同样按顺序完成所有请求

虚拟设备

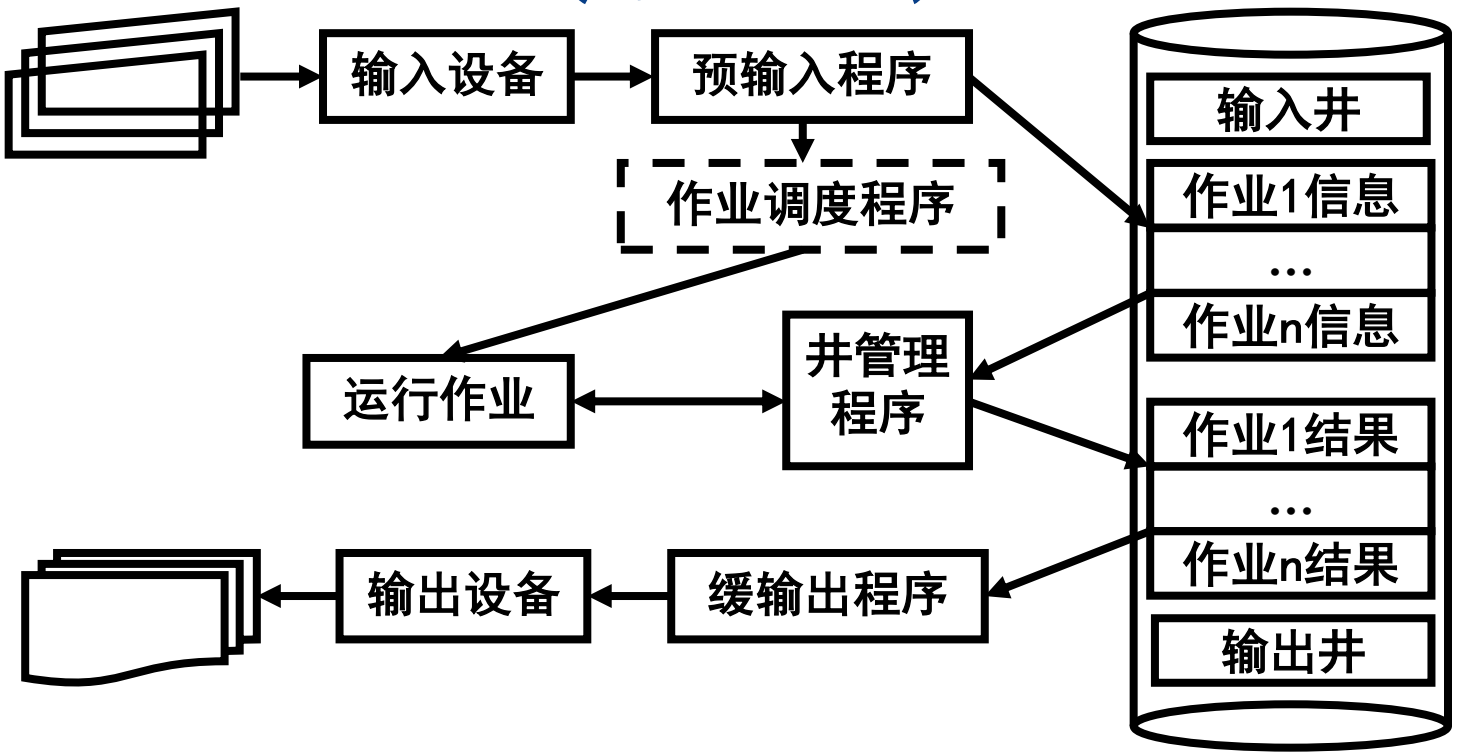
* 虚拟设备
  * 使用一类物理设备模拟另一类物理设备的技术
  * 通常是使用共享型外围设备模拟独占型外围设备
* 脱机同时外围设备操作

SPOOLing

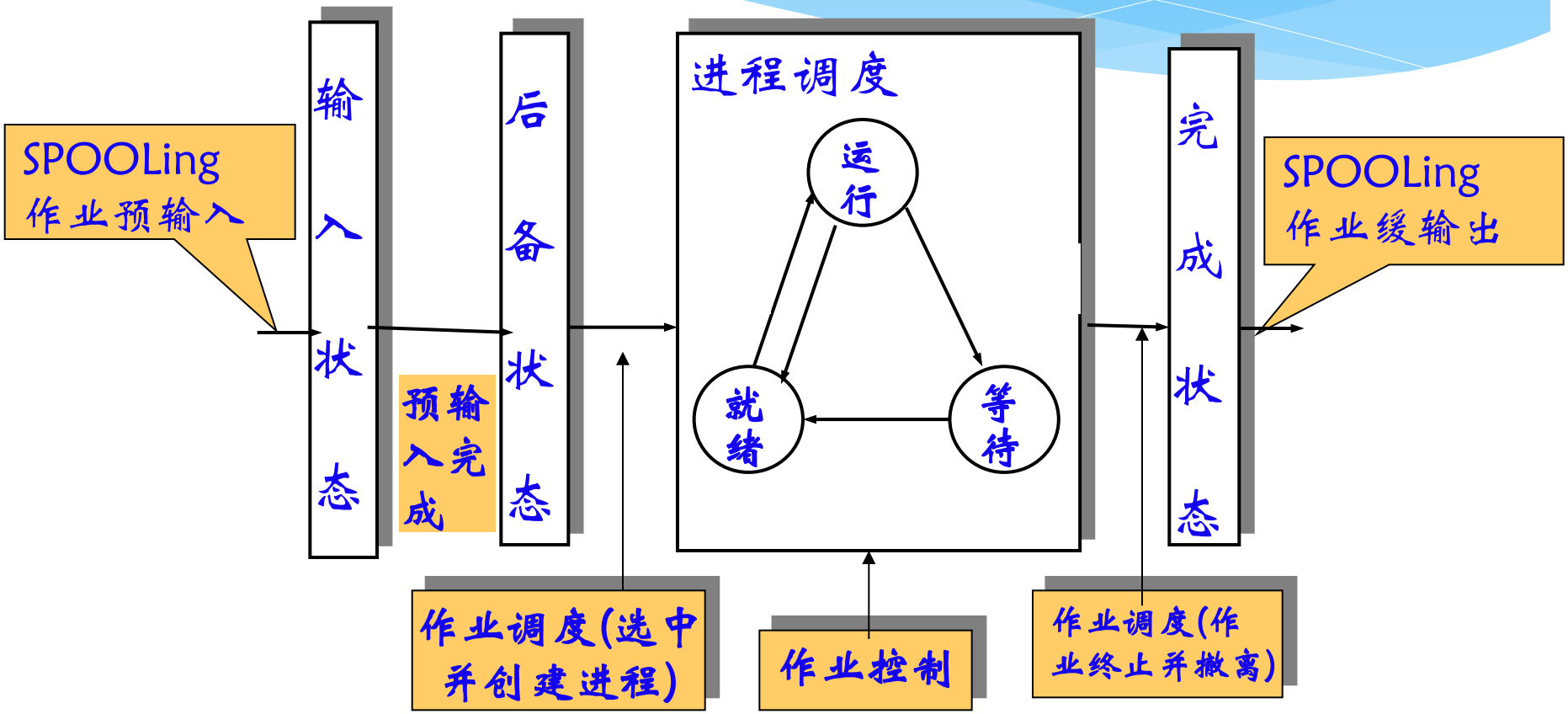
* “井”是用作缓冲的存储区域，采用井的技术能调节供求之间的矛盾，消除人工干预带来的损失。
* “预输入程序”
  * 操作系统将一批作业从输入设备上预先输入到磁盘的输入缓冲区中暂时保存，这称为“预输入”，此后，由作业调度程序调度作业执行，作业使用数据时不必再启动输入设备，只要从磁盘的输入缓冲区中读入
* “缓输出程序”
  * 作业执行中不必直接启动输出设备，只要将作业的输出数据暂时保存到磁盘的输出缓冲区，当作业执行完毕后，由操作系统组织信息成批输出
* “井管理程序”

SPOOLing

* 联机同时外围设备操作(斯普林系统)

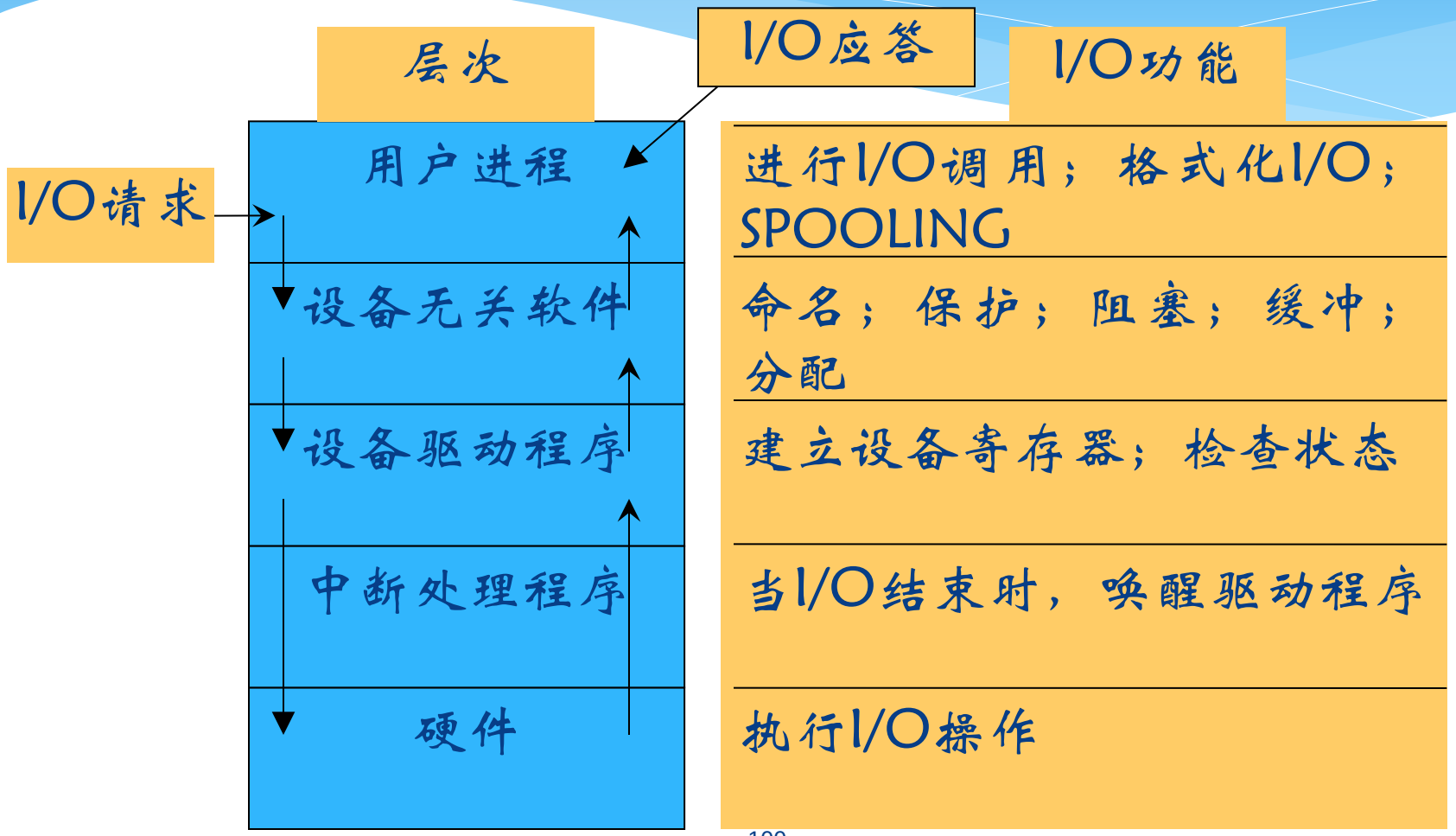


作业调度与进程调度的关系



99

I/O系统各层软件及其功能

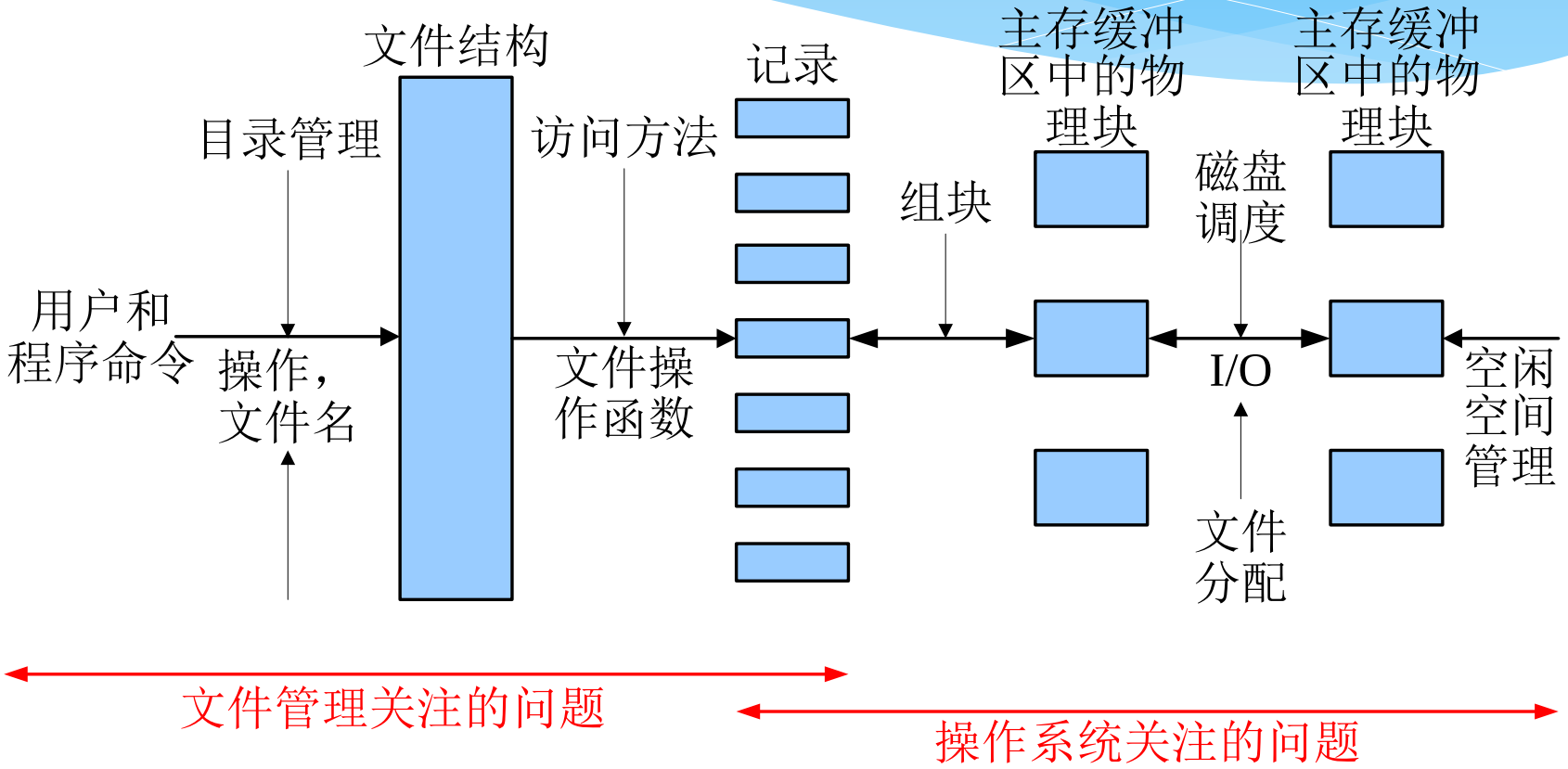


南京大学
NANJING UNIVERSITY

第五章 文件管理
本主题教学目标

1. 掌握文件和文件系统的概念
2. 了解文件系统的实现层次
3. 掌握文件目录
4. 掌握文件的组织与定位
5. 掌握文件记录成组技术
6. 掌握文件共享技术

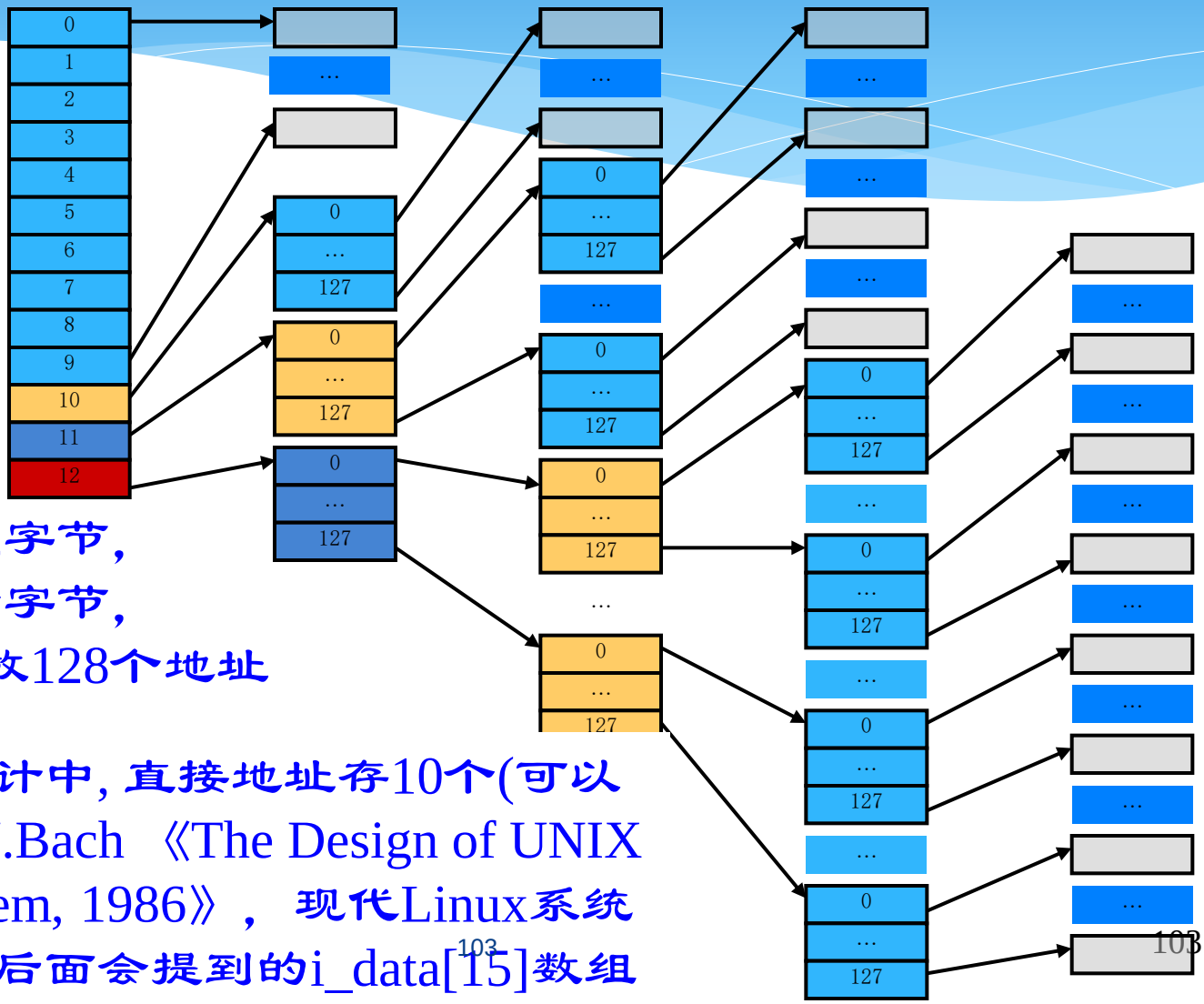
文件管理的要素


文件结构
记录
主存缓冲区中的物理块
主存缓冲区中的物理块
目录管理
访问方法
组块
磁盘调度
用户和程序命令
操作，文件名
文件操作函数
I/O
空闲空间管理
文件分配
文件管理关注的问题
操作系统关注的问题

102

索引文件(4)

UNIX/Linux多重索引结构



每个物理块512字节，

每个块地址4个字节，

每个块可以存放128个地址

Unix最初的设计中，直接地址存10个(可以参阅 Maurice J.Bach 《The Design of UNIX Operating System, 1986》，现代Linux系统改为12个，即后面会提到的i_data[15]数组

103

* 在UNIX系统中，每个i节点中分别含有10个直接地址的索引和一、二、三级间接索引。若每个盘块放128个盘块地址，则一个1MB的文件分别占用多少间接盘块？20MB的文件呢？设每个盘块有512B。
* 答：

直接块容量=10×512B/1024=5KB

一次间接容量=128×512B/1024=64KB

} 69KB

二次间接容量=128×128×512B/1024=64KB×128=8192KB

三次间接容量
=128×128×128×512B/1024=64KB×128=8192KB×128=1048576KB

1MB为1024KB，1024KB-69KB=955KB，955×1024B/512B=1910块，1MB的文件分别占用1910个二次间接盘块。

20×1024KB-69-8192=12219KB，12219×1024B/512=24438块，20MB的文件分别占用24438个三次间接盘块和128*128=16384个二次间接盘块。

文件系统内部结构

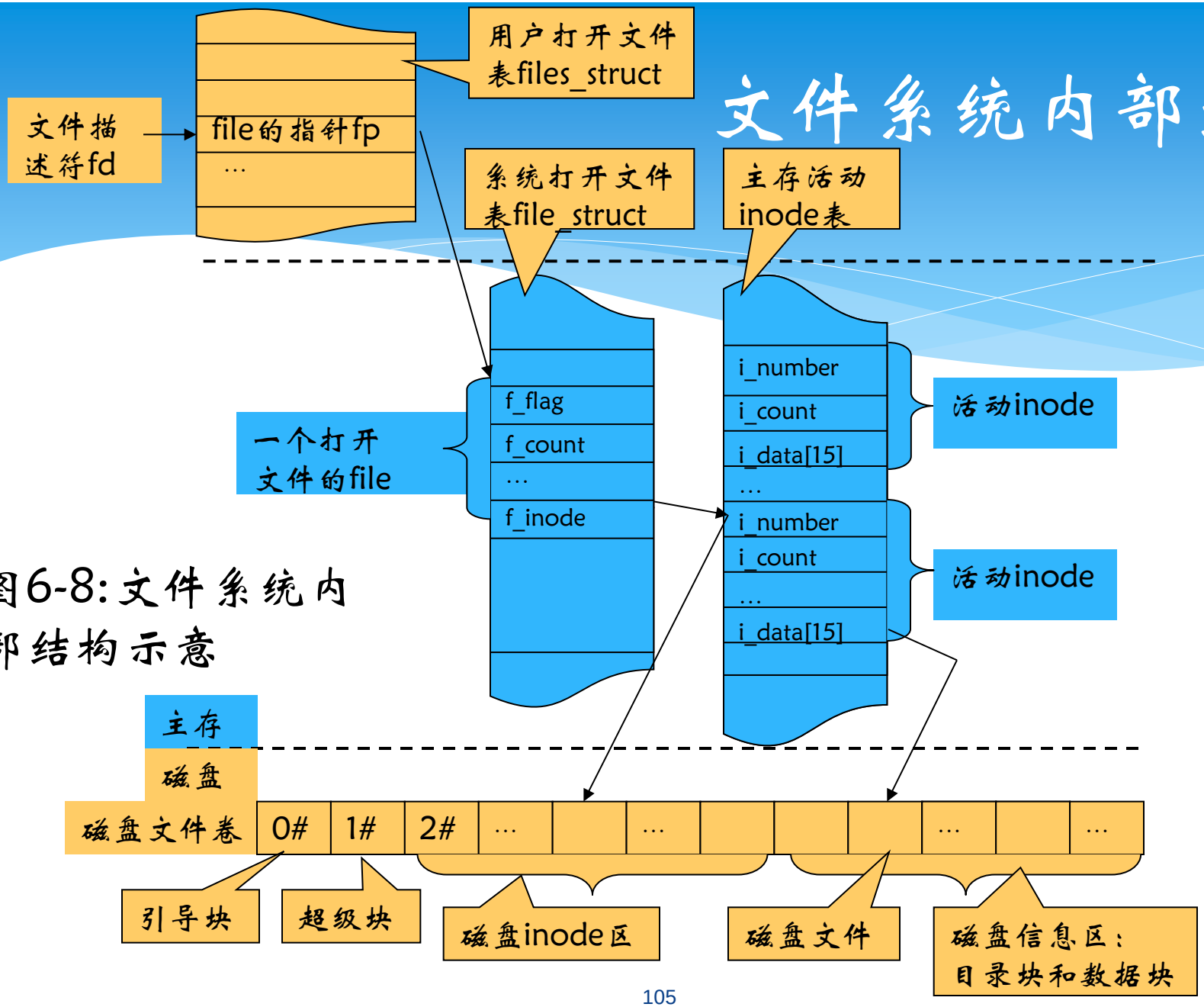
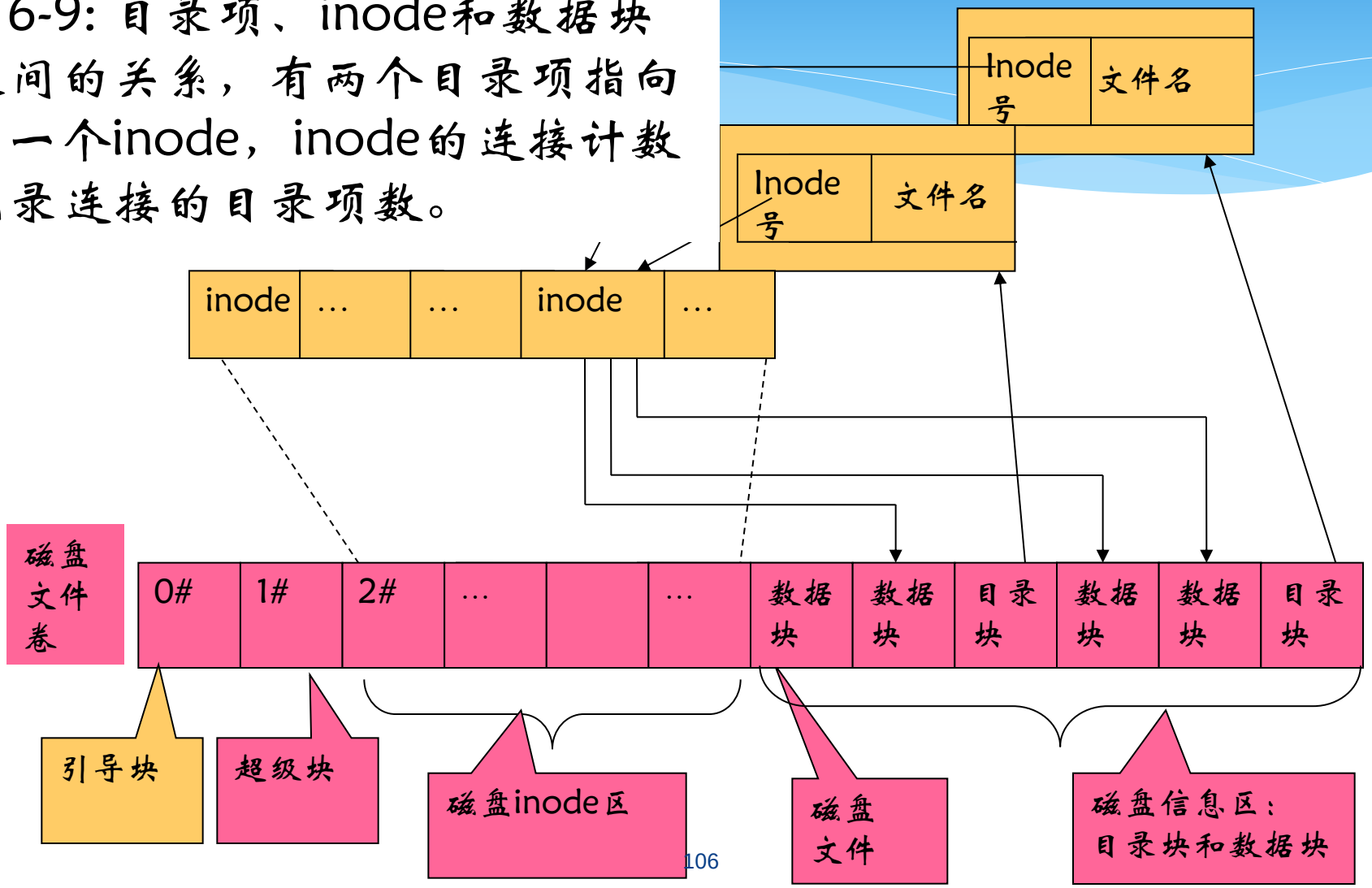


图6-8:文件系统内部结构示意

目录项、inode和数据块的关系

图6-9：目录项、inode和数据块之间的关系，有两个目录项指向同一个inode，inode的连接计数记录连接的目录项数。



文件系统调用 (1)
(1)文件的创建

系统调用C语言格式为：

```
int fd, mode;
char *filenamep;
fd = create (filenamep, mode);
```

文件系统调用 (2)

文件创建执行过程

① 为新文件分配索引节点和活动索引节点，并把索引节点编号与文件分量名组成新目录项，记到目录中。

② 在新文件所对应的活动索引节点中置初值，如置存取权限i_mode，连接计数i_nlink等。

③ 分配用户打开文件表项和系统打开文件表项，置表项初值，读写位移f_offset清“0”。

④把各表项及文件对应的活动索引节点用指针连接起来，把文件描述字返回给调用者。

文件系统调用 (3)
(2) 文件的删除

* 删除把指定文件从所在的目录文件中除去。
* 如果没有连接用户(i_link 为“1”)，还要把文件占用的存储空间释放。删除系统调用形式为：unlink (filenamep)。
* 在执行删除时，必须要求用户对该文件具有“写”操作权。

文件系统调用 (4)
(3) 文件的打开(1)

调用方式为：

```
int fd, mode;
char * filenamep;
fd = open (filenamep, mode);
```

文件系统调用 (5)
文件打开执行过程(2)

① 检索目录，把它的外存索引节点复制到活动索引节点表。

② 根据参数mode核对权限，如果非法，则这次打开失败。

③ 当“打开”合法时，为文件分配用户打开文件表项和系统打开文件表项，并为表项设置初值。通过指针建立这些表项与活动索引节点间的联系。把文件描述字，即用户打开文件表中相应文件表项的序号返回给调用者。

文件系统调用 (6)
(4) 文件的关闭 (1)

调用方式为：

```
int fd;
close (fd);
```

文件系统调用 (7)
文件的关闭(2)

① 根据fd找到用户打开文件表项，再找到系统打开文件表项。释放用户打开文件表项。

② 把对应系统打开文件表项中的f_count减“1”，如果非“0”，说明还有进程共享这一表项，不用释放直接返回；否则释放表项。

③ 把活动索引节点中的i_count减“1”，若不为“0”，表明还有用户进程正在使用该文件，不用释放而直接返回，否则在把该活动索引节点中的内容复制回文件卷上的相应索引节点中后，释放该活动索引节点。

文件系统调用(8)
文件的关闭(3)

④ f_count和i_count分别反映进程动态地共享一个文件的两种方式，

a) f_count反映不同进程通过同一个系统打开文件表项共享一个文件的情况；

b) i_count反映不同进程通过不同系统打开文件表项共享一个文件的情况。

c) 通过两种方式，进程之间既可用相同的位移指针f_offset，也可用不同位移指针f_offset共享同一个文件。

文件系统调用 (9)
(5)读文件(1)

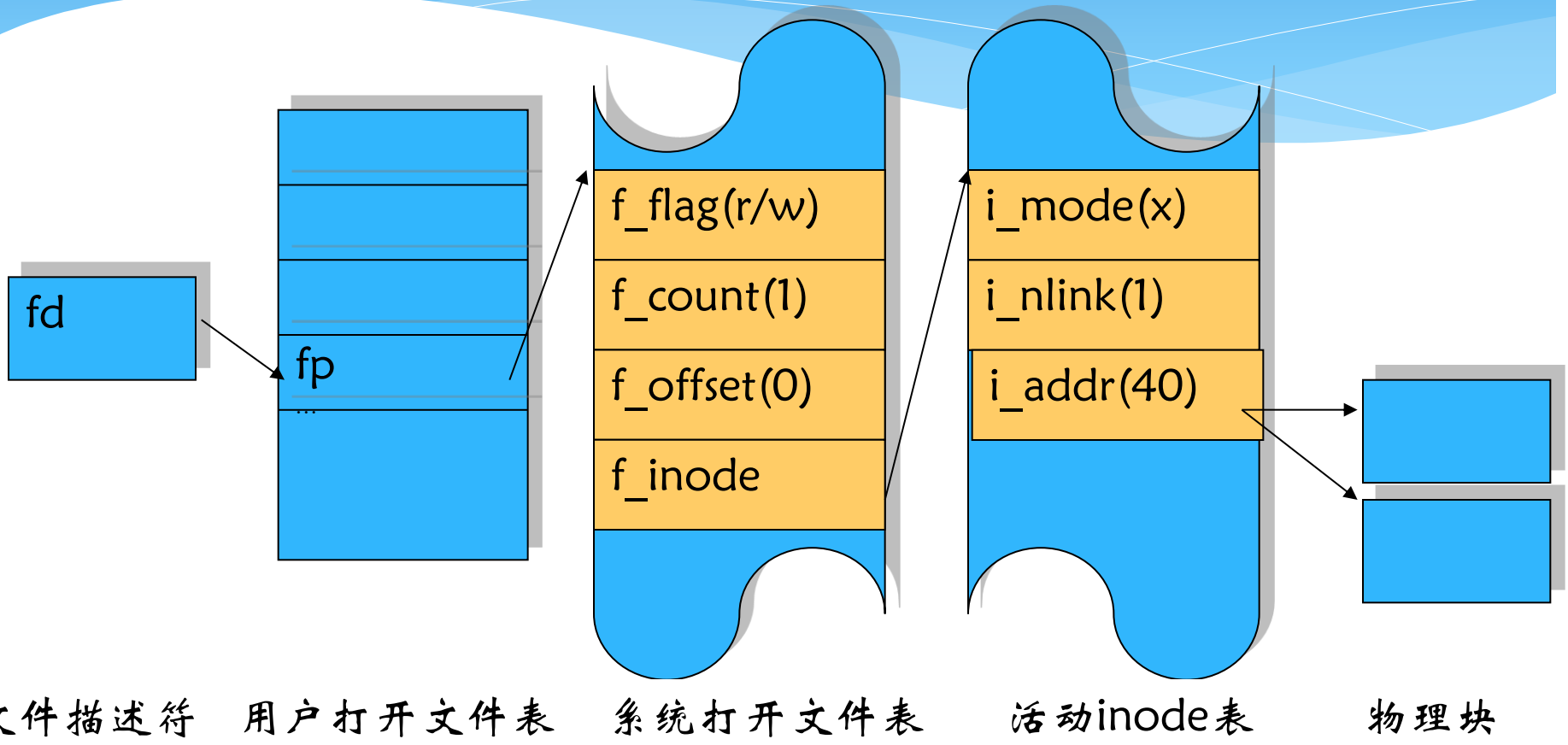
调用的形式为：

```
int nr, fd, count;
char buf [ ];
nr = read (fd, buf, count);
```

文件系统调用 (10)
读文件(2)

* 系统根据f_flag中的信息，检查读操作合法性，
* 再根据当前位移量f_offset值，要求读出的字节数，及活动索引节点中i_addr指出的文件物理块存放地址，把相应的物理块读到缓冲区中，然后再送到bufp指向的用户主存区中。

文件系统调用 (11)
读文件(3)



读操作时文件数据结构的关系

文件系统调用 (12)
(6)写文件

调用的形式为：

nw = write (fd, buf, count);

buf是信息传送的源地址，即把buf所指向的用户主存区中的信息，写入到文件中。

文件系统调用 (13)
(7)文件的随机存取(1)

* 在文件初次“打开”时，文件的位移量f_offset清空为零，以后的文件读写操作总是根据offset的当前值，顺序地读写文件。为了支持文件的随机访问，提供系统调用lseek，它允许用户在读、写文件前，事先改变f_offset的指向

系统调用的形式为：

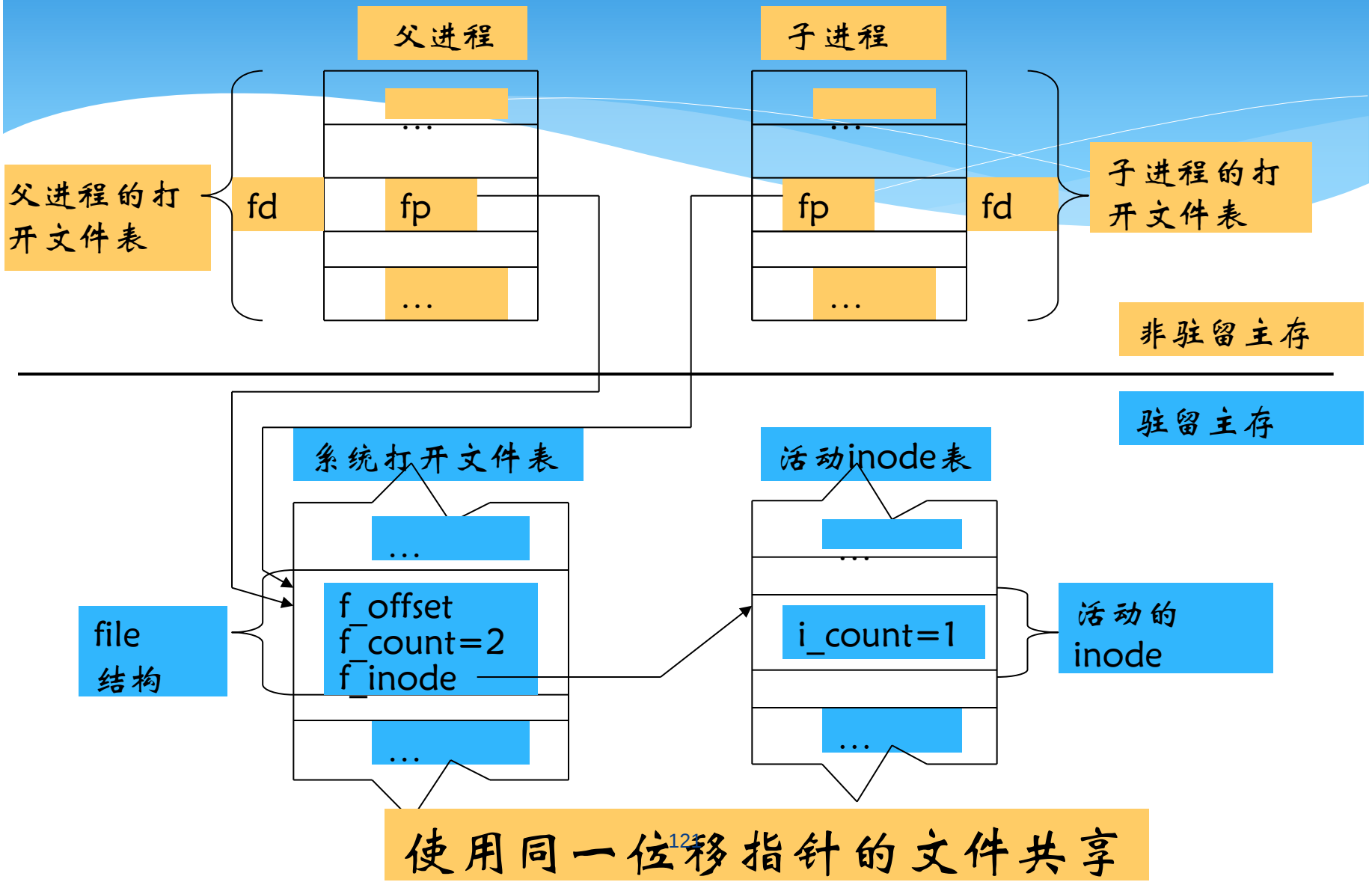
long lseek;
long offset;
int whence, fd;
lseek (fd, offset, whence);

文件系统调用 (14)
文件的随机存取(2)

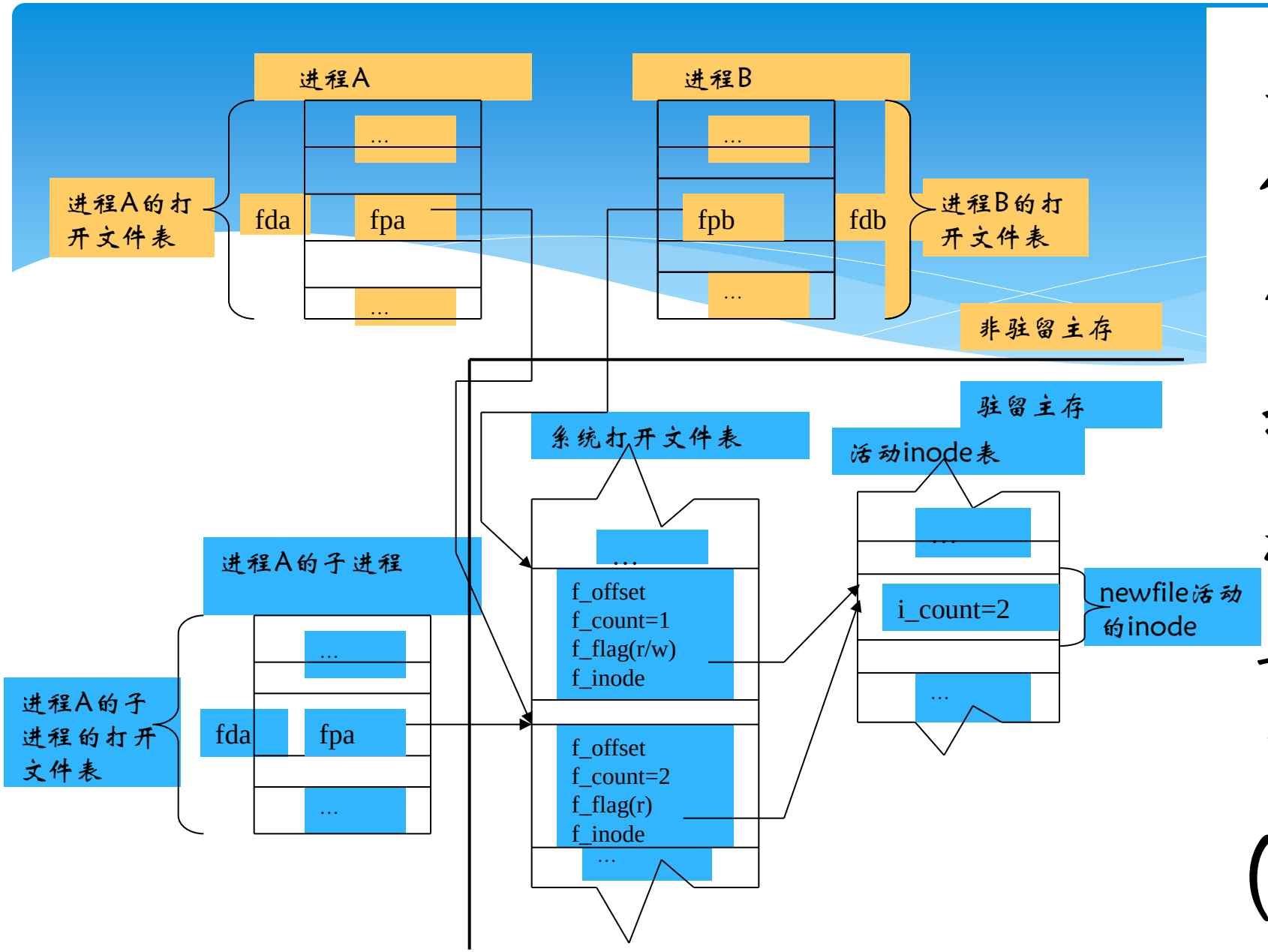
文件描述字fd必须指向一个用读或写方式打开的文件，

* 当whence是“0”时，则f_offset被置为offset，
* 当whence是“1”时，则f_offset被置为文件当前位置加上offset。

文件的动态共享(3)

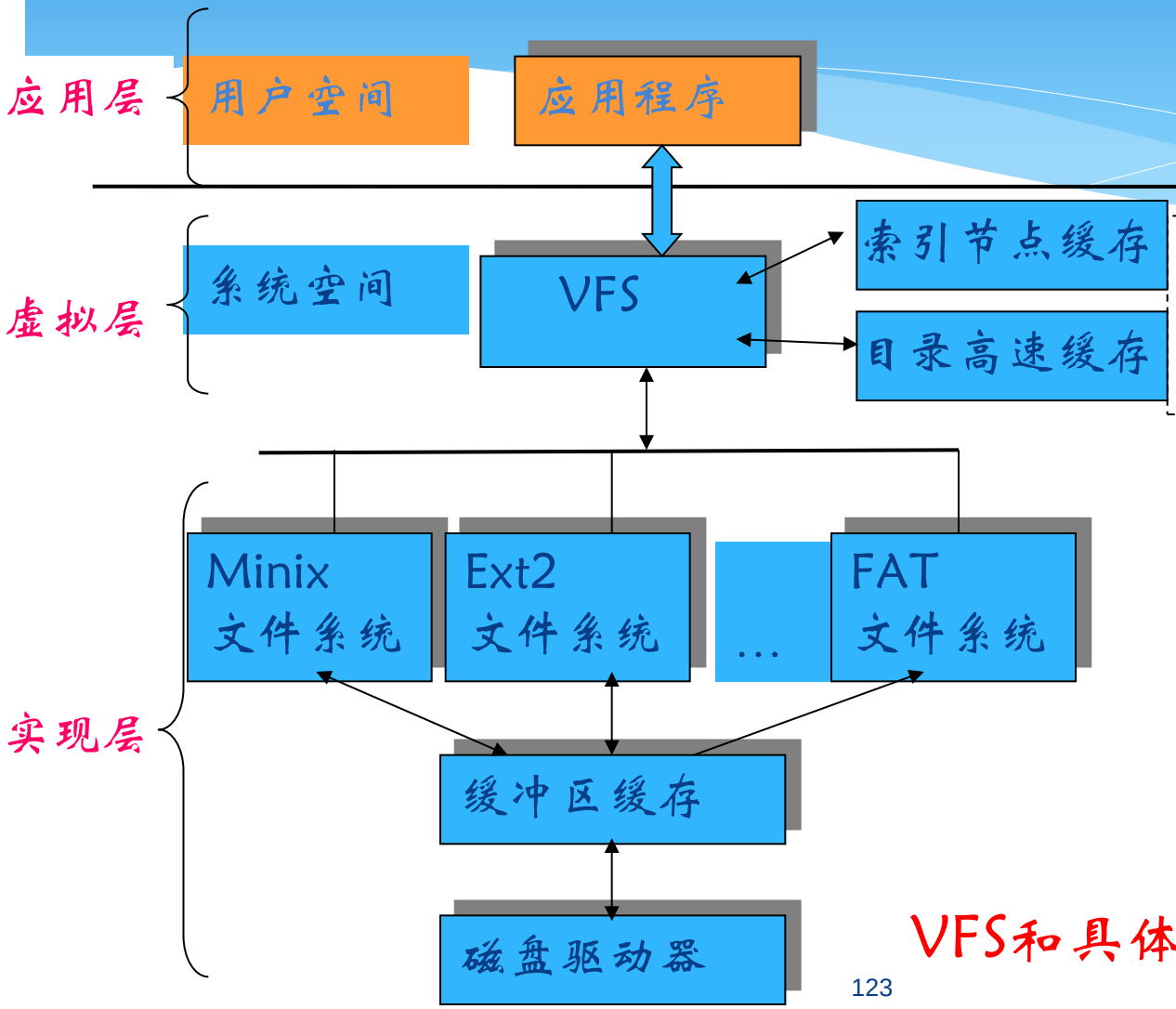

父进程
子进程
父进程的打开文件表
fd
fp
fp
fd
子进程的打开文件表
非驻留主存
驻留主存
系统打开文件表
活动inode表
file结构
f_offset
f_count=2
f_inode
i_count=1
活动的inode
使用同一位移指针的文件共享

文件的动态共享(5)



使用不同位移指针的文件共享

Linux虚拟文件系统



标准文件类系统调用，open()、read()、write()、close()等

系统调用对应的内核函数，sys_open()、sys_read()、sys_write()、sys_close()

具体文件系统对应的文件操作函数，open()、read()、write()、close()等

VFS和具体文件系统的关系

第六章 并发程序设计
本主题教学目标

1. 了解程序的并发性与并发程序设计
2. 掌握临界区互斥及其解决方案
3. 熟练使用PV进行程序设计
4. 掌握Hoare管程
5. 掌握消息传递

进程的交互：竞争与协作

* 进程之间存在两种基本关系：竞争关系和协作关系
* **第一种是竞争关系**，一个进程的执行可能影响到同其竞争资源的其他进程，如果两个进程要访问同一资源，那么，一个进程通过操作系统分配得到该资源，另一个将不得不等待
* **第二种是协作关系**，某些进程为完成同一任务需要分工协作，由于合作的每一个进程都是独立地以不可预知的速度推进，这就需要相互协作的进程在某些协调点上协调各自的工作。当合作进程中的一个到达协调点后，在尚未得到其伙伴进程发来的消息或信号之前应阻塞自己，直到其他合作进程发来协调信号或消息后方被唤醒并继续执行

125

竞争关系：进程的互斥

* 进程的互斥(mutual exclusion)是解决进程间竞争关系(间接制约关系)的手段。进程互斥指若干个进程要使用同一共享资源时，任何时刻最多允许一个进程去使用，其他要使用该资源的进程必须等待，直到占有资源的进程释放该资源

126

南京大学
NANJING UNIVERSITY

协作关系：进程的同步

* 进程的同步(Synchronization)是解决进程间协作关系(直接制约关系)的手段。进程同步指两个以上进程基于某个条件来协调它们的活动。一个进程的执行依赖于另一个协作进程的消息或信号，当一个进程没有得到来自于另一个进程的消息或信号时则需等待，直到消息或信号到达才被唤醒

127

进程的交互：竞争与协作

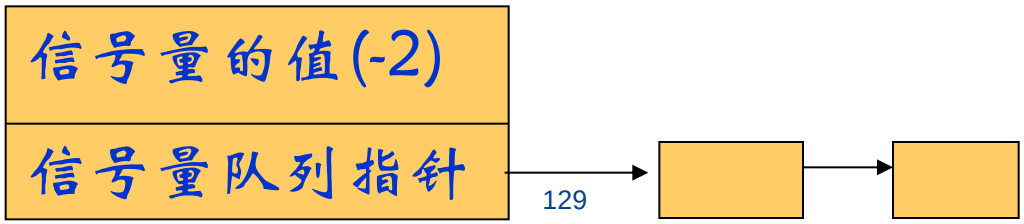
* 进程互斥关系是一种特殊的进程同步关系，即逐次使用互斥共享资源，是对进程使用资源次序上的一种协调

128

信号量与PV操作

设s为一个记录型数据结构，一个分量为整型量value，另一个为信号量队列queue，P和V操作原语定义：

* P(s)：将信号量s减去1，若结果小于0，则调用P(s)的进程被置成等待信号量s的状态
* V(s)：将信号量s加1，若结果不大于0，则释放(唤醒)一个等待信号量s的进程，使其转换为就绪态

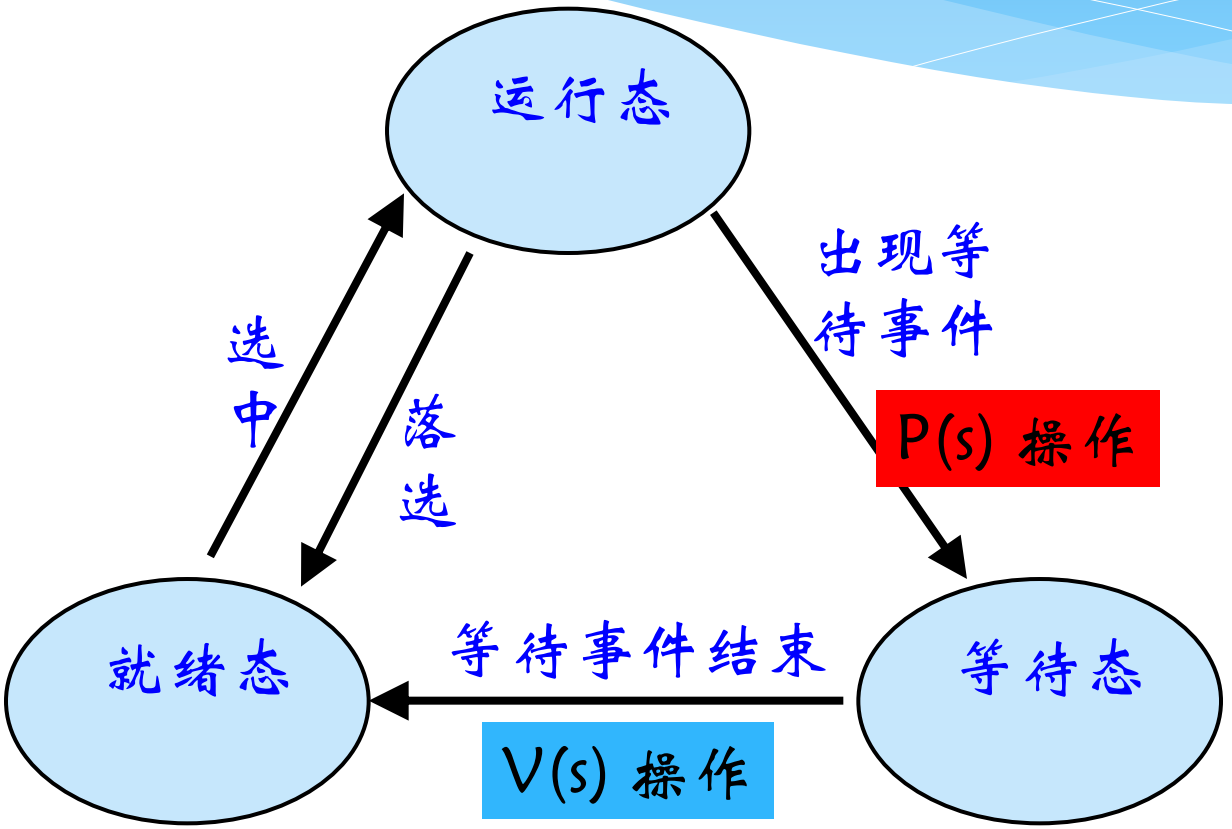


信号量与PV操作

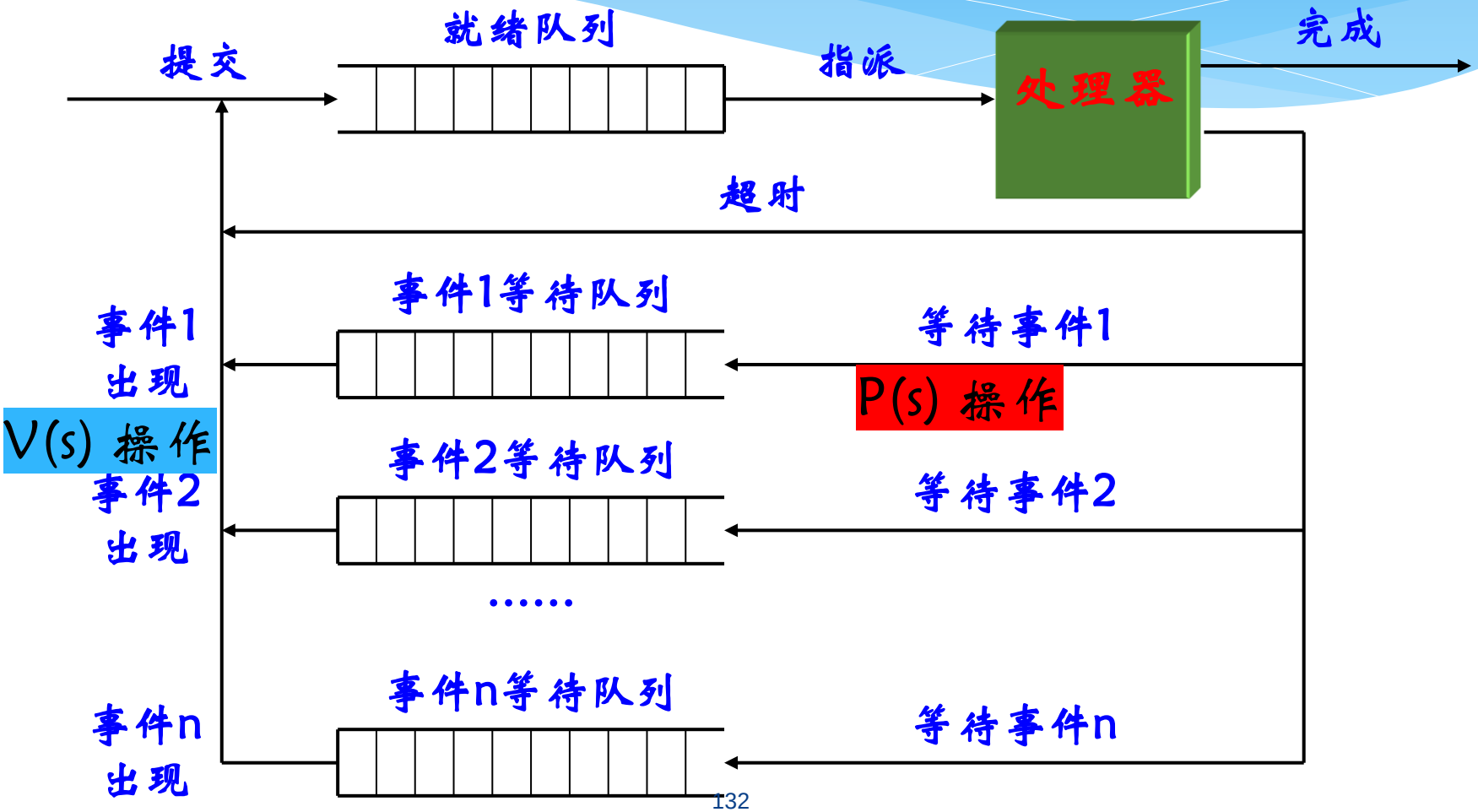
```
struct  semaphore
  { int count; QueueType queue; }
void P(semaphore s)  // also named wait
{
  s.count - -;
  if (s.count < 0)  { place this process in s.queue; block this process }
}

void V(semaphore s)  // also named signal
{
    s.count ++;
    if (s.count <= 0)
       { remove a process from s.queue; convert it to ready state; }
}
```

PV操作与进程状态转换模型



PV操作与进程状态队列模型



信号量

* 推论1：若信号量s为正值，则该值等于在封锁进程之前对信号量s可施行的P操作次数、亦等于s所代表的实际还可以使用的物理资源数
* 推论2：若信号量s为负值，则其绝对值等于登记排列在该信号量s队列之中等待的进程个数、亦即恰好等于对信号量s实施P操作而被封锁起来并进入信号量s队列的进程数
* 推论3：通常，P操作意味着请求一个资源，V操作意味着释放一个资源。在一定条件下，P操作代表阻塞进程操作，而V操作代表唤醒被阻塞进程的操作

南京大学
NANJING UNIVERSITY

经典问题求解
(信号量与PV操作)

* 互斥问题
  * (1) 飞机票问题
  * (2) 哲学家就餐问题
* 同步问题
  * (1)生产者-消费者问题
  * (2) 苹果-桔子问题

南京大学
NANJING UNIVERSITY

直接通信

* 对称直接寻址，发送进程和接收进程必须命名对方以便通信，原语send() 和 receive()定义如下：
  * send(P, messsage) 发送消息到进程P
  * receive(Q, message) 接收来自进程Q的消息
* 非对称直接寻址，只要发送者命名接收者，而接收者不需要命名发送者，send()和 receive()定义如下：
  * send(P, messsage) 发送消息到进程P
  * receive(id, message) 接收来自任何进程的消息，变量id置成与其通信的进程名称

南京大学
NANJING UNIVERSITY

间接通信

* 消息不是直接从发送者发送到接收者，而是发送到由临时保存这些消息的队列组成的一个共享数据结构，这些队列通常成为信箱(mailbox)
* 一个进程给合适的信箱发送消息，另一进程从信箱中获得消息
* 间接通信的send()和 receive()定义如下：
  * send(A,message)：把一封信件(消息)传送到信箱A
  * receive(A,message)：从信箱A接收一封信件(消息)

南京大学
NANJING UNIVERSITY

死锁

12.1 死锁产生

12.2 死锁防止

12.3 死锁避免

12.4 死锁检测和解除

死锁防止(1)

系统形成死锁的四个必要条件

* 互斥条件(mutual exclusion)：系统中存在临界资源，进程应互斥地使用这些资源
* 占有和等待条件(hold and wait)：进程请求资源得不到满足而等待时，不释放已占有的资源
* 不剥夺条件(no preemption)：已被占用的资源只能由属主释放，不允许被其它进程剥夺
* 循环等待条件(circular wait)：存在循环等待链，其中，每个进程都在链中等待下一个进程所持有的资源，造成这组进程永远等待

实例说明系统所处的安全或不安全状态(1)

* 如果系统中共有五个进程和A、B、C三类资源
* A类资源共有10个,B类资源共有5个,C类资源共有7个
* 在时刻T_0,系统目前资源分配情况如下：

实例说明系统所处的安全或不安全状态(2)

process	Allocation			Claim			Available		
	A	B	C	A	B	C	A	B	C
* P_0	0	1	0	7	5	3	3	3	2
* P_1	2	0	0	3	2	2			
* P_2	3	0	2	9	0	2			
* P_3	2	1	1	2	2	2			
* P_4	0	0	2	4	3	3			

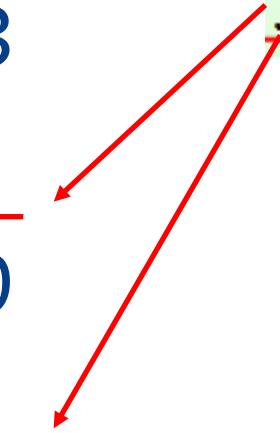
资源总数 [10 5 7]

实例说明系统所处的安全或不安全状态(3)

每个进程目前还需资源为Cki-Aki

process	C_{ki}-A_{ki}		
	A	B	C
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Available		
A	B	C
3	3	2



简化进程-资源分配图检测系统是否处于死锁状态(1)

(1)如果进程-资源分配图中无环路，则此时系统没有发生死锁

(2)如果进程-资源分配图中有环路，且每个资源类中仅有一个资源，则系统中发生了死锁，此时，环路是系统发生死锁的充要条件，环路中的进程便为死锁进程

(3)如果进程-资源分配图中有环路，且涉及的资源类中有多个资源，则环路的存在只是产生死锁的必要条件而不是充分条件

简化进程-资源分配图检测系统是否处于死锁状态(2)

* 如果能在进程-资源分配图中消去此进程的所有请求边和分配边，成为孤立结点。经一系列简化，使所有进程成为孤立结点，则该图是可完全简化的；否则则称该图是不可完全简化的
* 系统为死锁状态的充分条件是：当且仅当该状态的进程-资源分配图是不可完全简化的。该充分条件称为死锁定理

王国维-人间词话

第一境界：昨夜西风凋碧树。独上高楼，望尽天涯路。

第二境界：衣带渐宽终不悔，为伊消得人憔悴。

第三境界：众里寻他千百度。蓦然回首，那人却在灯火阑珊处。

祝大家期末考试顺利！