

《计算机导论》

— 第四部分：操作系统



四川大學
SICHUAN UNIVERSITY

操作系统

操作系统需要做什么？
操作系统是什么时候开始工作的？
如何运行一个程序？
如何同时运行多个任务？



一、操作系统概述

- 典型操作系统介绍 (unix/windows为例)
- 基本特征 (分时/共享/虚拟/异步)
- 类型 (网络/实时/虚拟)

二、操作系统基本工作原理

- 一道程序的执行：内存加载，分配计算资源
- 核心功能 (管理资源：进程/中断/内存/文件)
- 多道程序的执行：轮转机制，分时机制

三、操作系统内核

- 进程管理
 - 进程控制块PCB
 - 状态与队列调度
 - 线程：并行与并发
- 存储管理
 - 内存虚拟化
 - 层次化管理
 - 地址管理机制
- 中断管理
 - 中断管理机制
 - 中断类型与中断向量
- 文件管理
 - 文件系统(结构与索引)
 - 存储 (格式与路径)

难点与重点

- 时空虚拟化概念
 - 轮转机制（时间换空间）
 - 内存虚拟化（虚拟空间）
- 进程与线程
- 中断（内部/外部，同步/异步）
- 内存管理-地址变换



一、操作系统概述

- 典型操作系统介绍（unix/windows为例）
- 基本特征（分时/共享/虚拟/异步）
- 类型（网络/实时/虚拟）

1. 经典操作系统

Unix

Windows (DOS)

Andoird

FreeRTOS



四川大學
SICHUAN UNIVERSITY

主机系统—Unix

- Unix
 - Bell Lab, DEC PDP-7, 1969
 - Ken Thompson, Dennis Ritchie, Doug McIlroy, Joe Ossana
 - 1970 Brian Kernighan dubbed the system “Unix”
 - Rewritten in C in 1973, announced in 1974
 - BSD (UC, Berkeley), System V(Bell lab)
 - Solaris (Sun Microsystem)
- Posix standard
- Ken Thompson, Dennis Ritchie
 - ACM Turing Award, 1983



主流操作系统分类

Windows (Dos)

Linux (Unix)

MacOs (Unix)

Android (Linux)

分时操作系统

多任务操作系统

实时操作系统

分布式操作系统



四川大學
SICHUAN UNIVERSITY

2. 操作系统基本工作原理

- 一道程序的执行：内存加载，分配计算资源
- 核心功能（管理资源：进程/中断/内存/文件）
- 多道程序的执行：轮转机制，分时机制

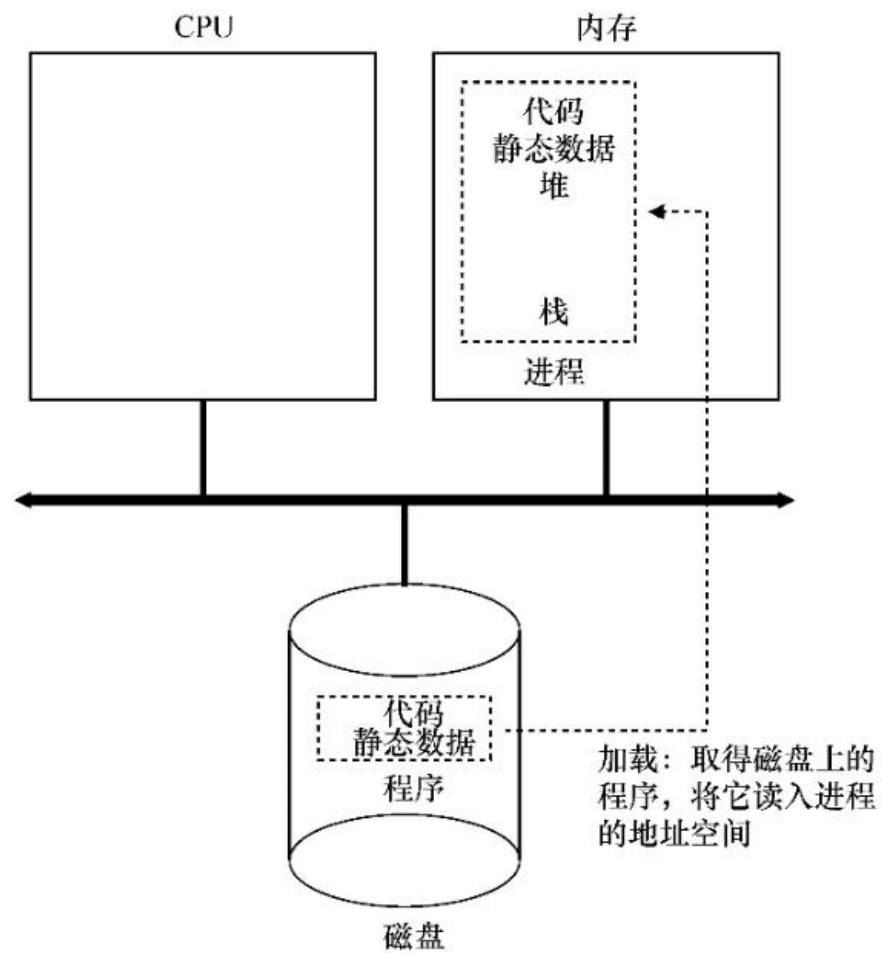


- 操作系统也是一道程序
- 目的是代替手工操作

手工操作:程序直接写在卡片上,由读卡机读入计算机,从而替代了插件板的方式,但是仍然存在许多人工操作。

问题: 人机无法交互,效率低下。

1. 操作系统如何运行一道程序

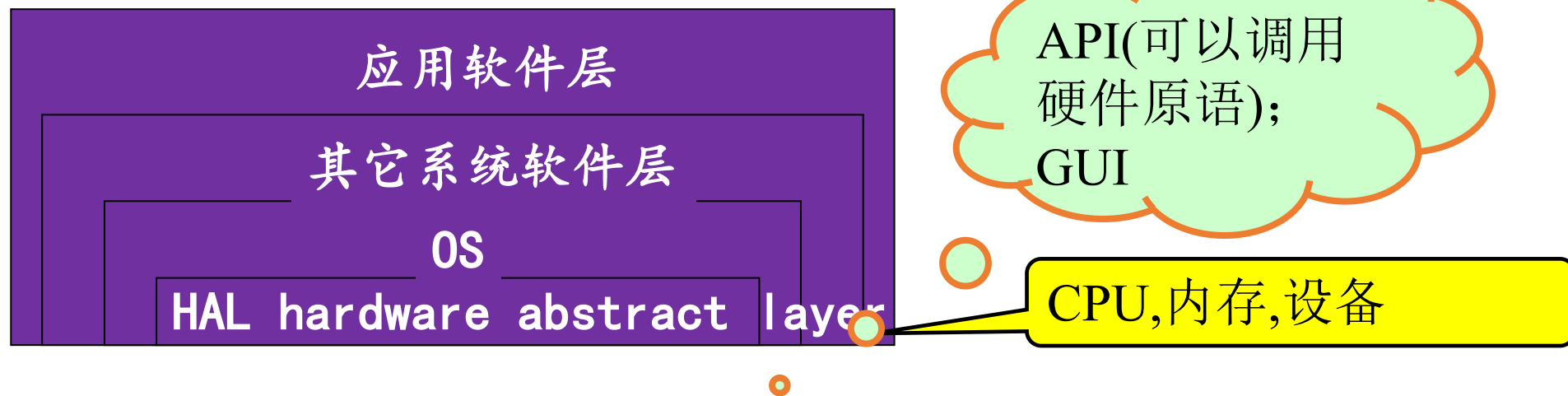


OS内核做了哪些工作?

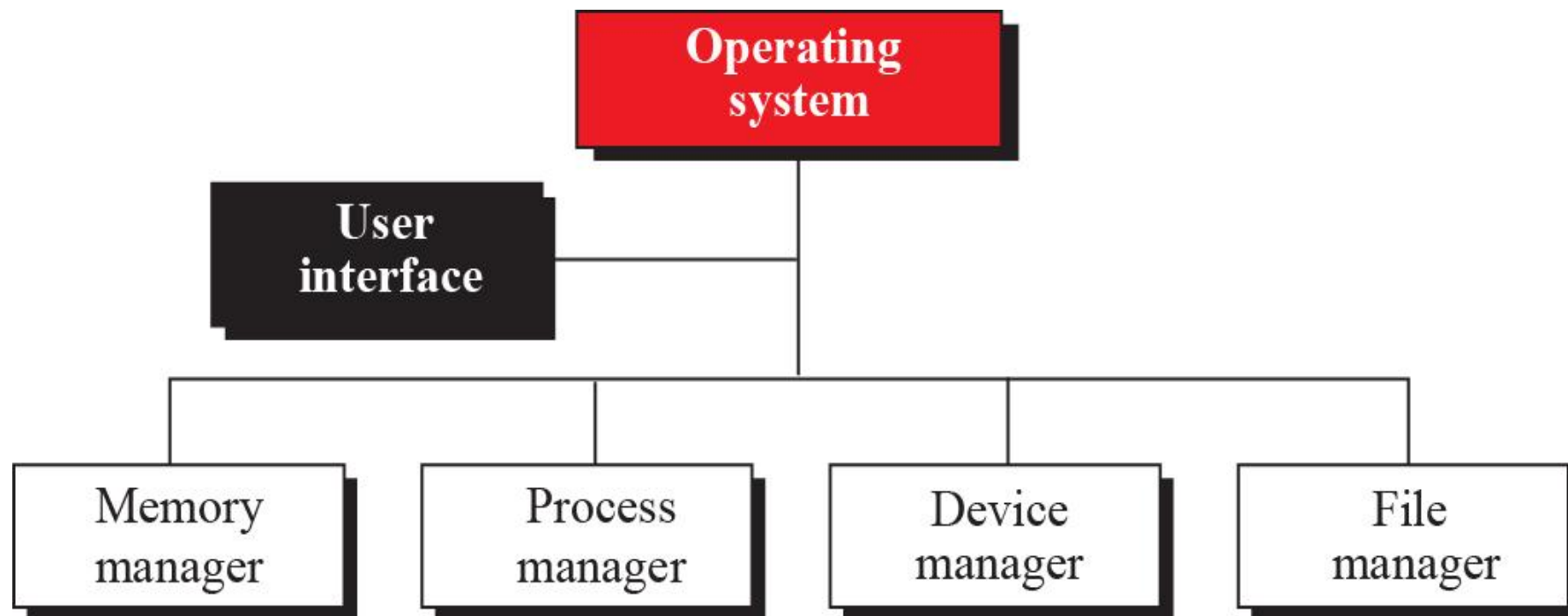
- 用fork()生成进程 OS:进程管理
- 用execve()启动进程管理 OS: 进程管理
- 用mmap()为程序分配存储资源 (MEM,寄存器等) OS:存储管理: 寄存器/虚拟存储
- 从磁盘调入需要执行的指令 (用户/系统 OS:存储管理: 地址映射
- 开始执行, 调用系统函数printf(), 产生中断, 调用核心中断write()
OS:中断管理
- 完成进程, 在显示器上显示结果, 并清理内存
OS:设备管理



2. 操作系统是什么？



任务（进程）管理、内存管理、文件管理、设备管理



操作系统是什么？

- 自动执行的程序
- 目的是协调控制对多个任务进行自动执行
- 它包含一系列内核任务API
- 对软件资源（进程，内存，中断，文件）与硬件资源进行管理



3. 操作系统如何运行多道程序

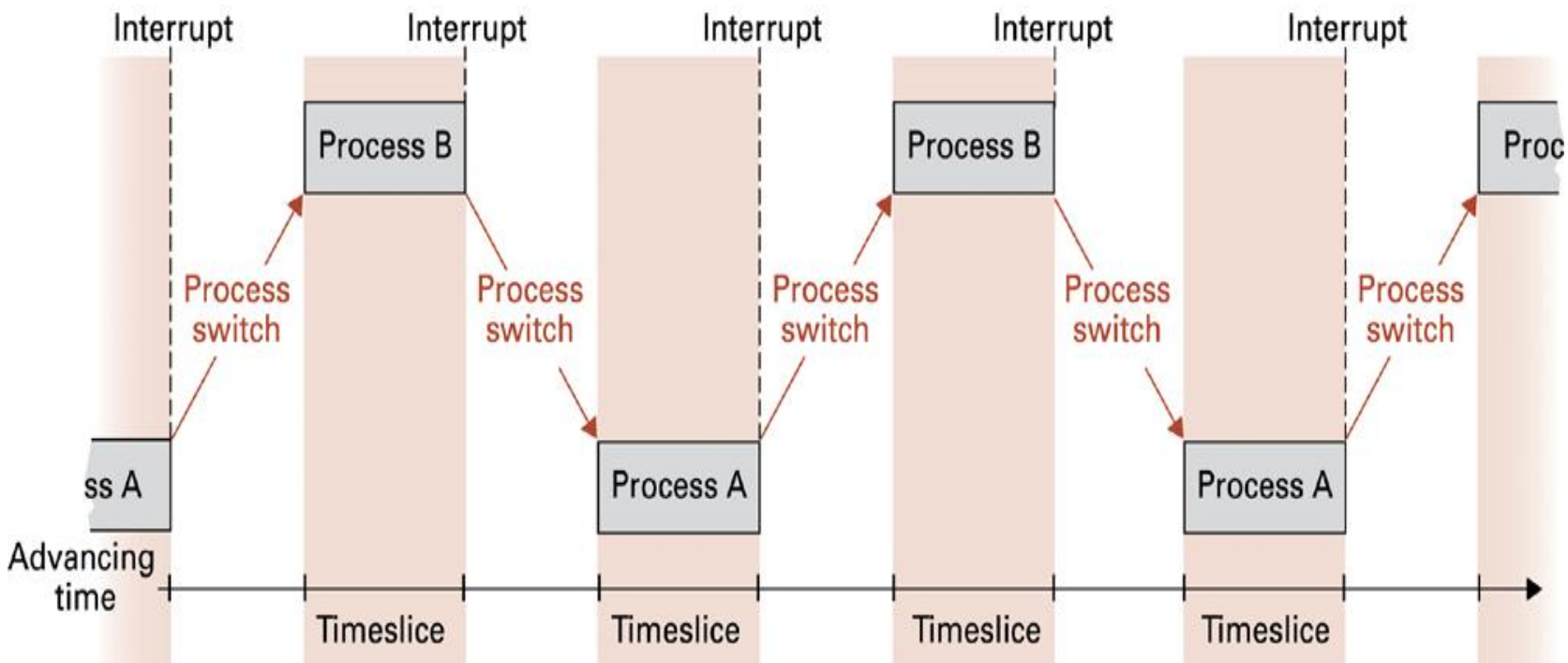
轮转机制

时空虚拟化

轮转机制：分时复用（time sharing）

- 内存是否只有一道程序？
 - CPU同时执行多道程序（分时复用）

时间片轮转（实时系统）- Round Robin-大家都一样



调度程序将时间划分为多个时间片
时间片的结束通过**中断**（中断）发出信号。

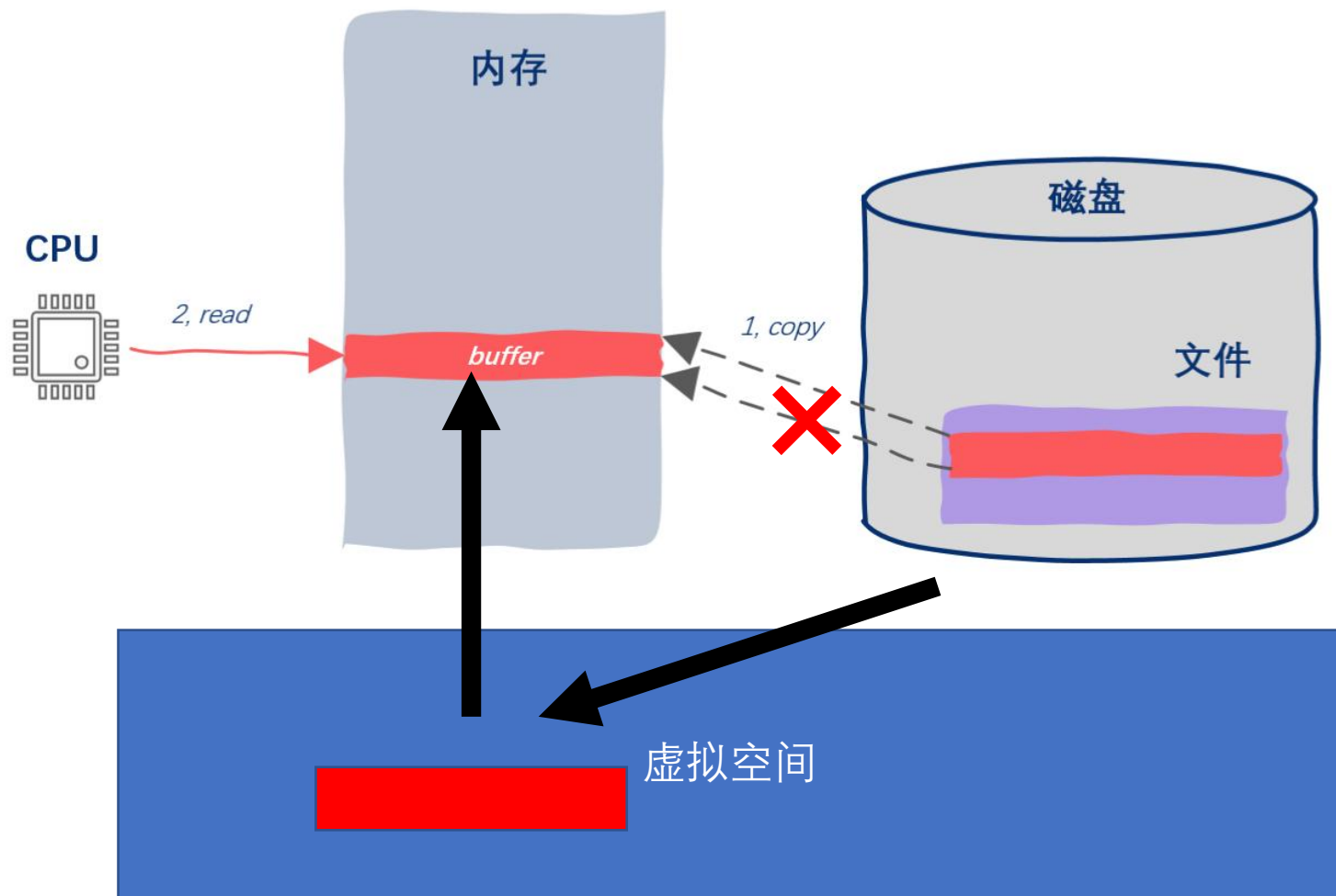
- 调度程序中断一个时间片用尽的进程，并将时间片分配给另一个进程（进程切换）。

时间片的结束通过**中断**（中断）发出信号。

- 调度程序中断一个时间片用尽的进程，并将时间片分配给另一个进程（进程切换）。

- 调度程序中断一个时间片用尽的进程，并将时间片分配给另一个进程（进程切换）。

2) 空间虚拟化：虚拟内存：映射到虚拟空间- 实体空间（内存）创新：空间转换

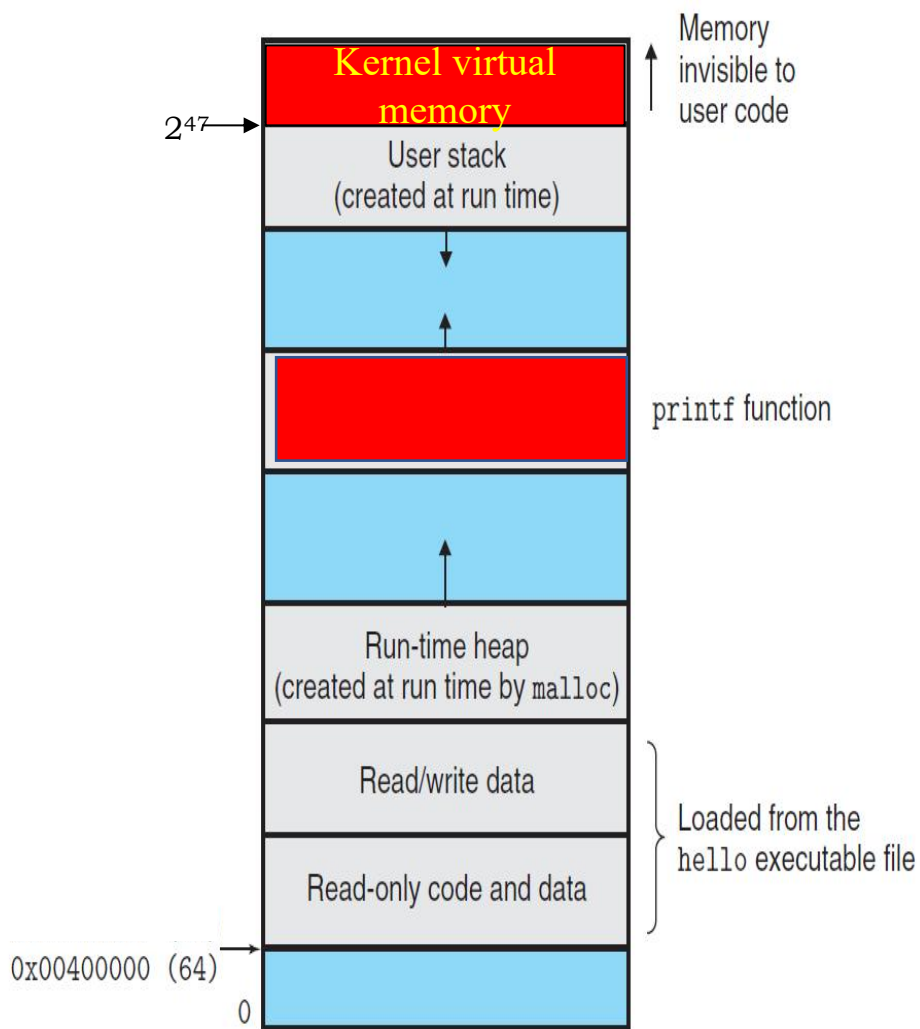


3.加载操作系统

- X86 OS:
- 执行 BIOS 中的上电自检程序检测所有硬件是否功能正常
- MBR 主引导扇区中读取 512 bytes 的 OS 引导程序 (BootLoader)，将其加载到 0x7c00 这个内存位置
- BIOS 执行 BootLoader，把操作系统放入系统内存



操作系统加载-操作系统是电脑执行的第一道程序



1. 加电
2. BIOS自检，对外设硬件检验
3. 加载操作系统引导程序 (BootLoader)
4. 引导程序加载操作系统
5. 操作系统初始化CPU
 - ① 设置heap 堆地址
 - ② 建立中断向量表
 - ③ 定义全局时钟树寄存器，内部的多个时钟频率
6. 执行指令
7. 关机断电。

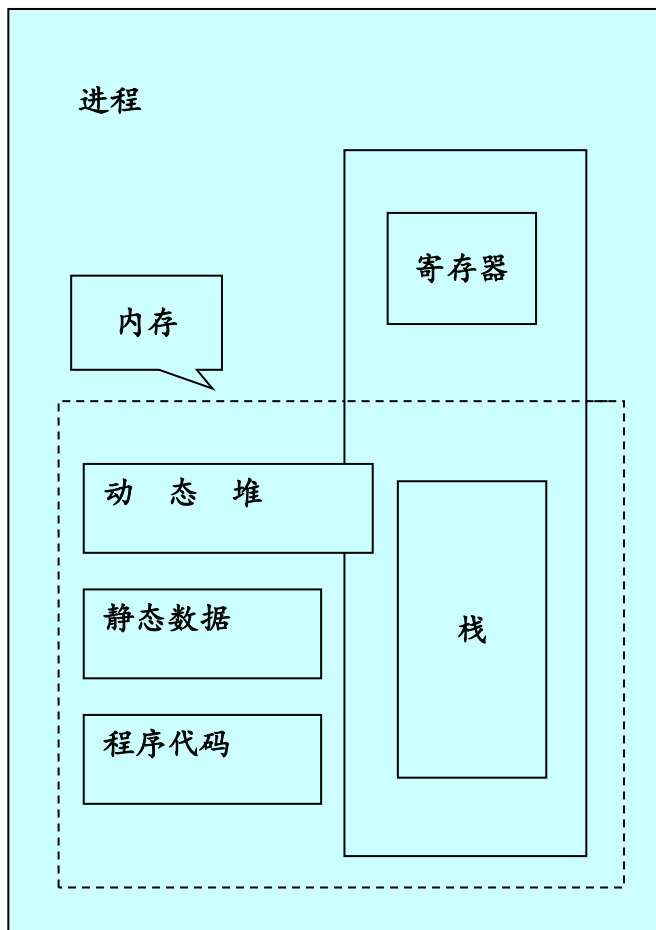


三、操作系统内核

1. 进程管理
2. 内存管理
3. 中断管理
4. 文件管理

1. 进程管理

1) 进程控制块PCB:



进程：是具有一定独立功能的程序关于一个数据集合的一次运行活动。
是**一次程序执行过程**
是操作系统进行**资源分配的基本单位**。

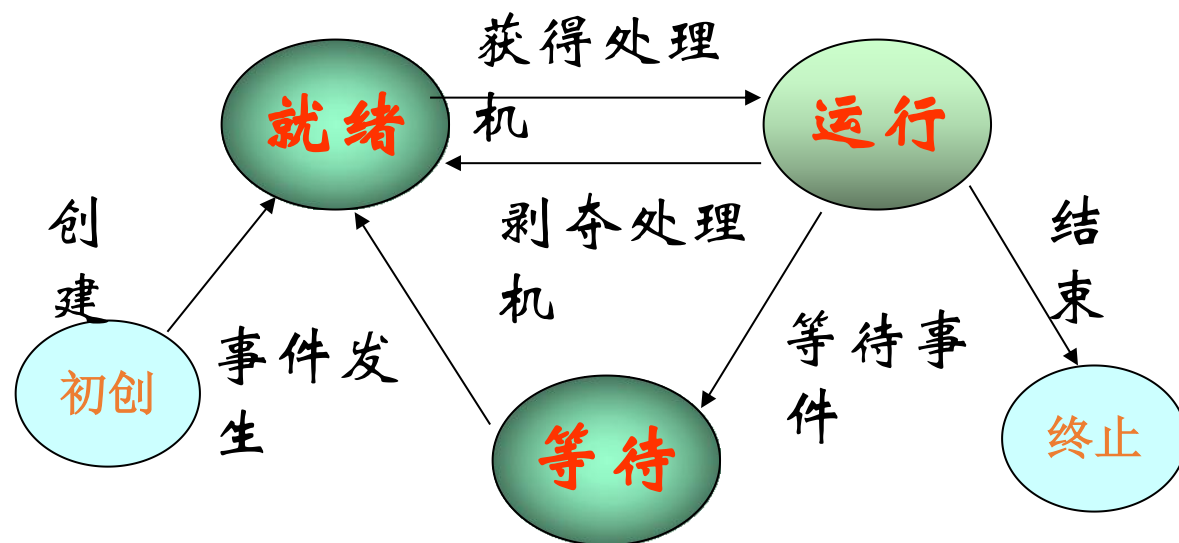
对计算机硬件资源进行请求并使用
动态：并发：可与其他进程同时执行。



2) 队列管理

- 进程状态
- 队列与调度算法

进程状态与进程状态转换图



系统开销：运行操作系统程序完成系统管理工作所花费的时间和空间

上下文切换开销：物理环境

PCB+程序

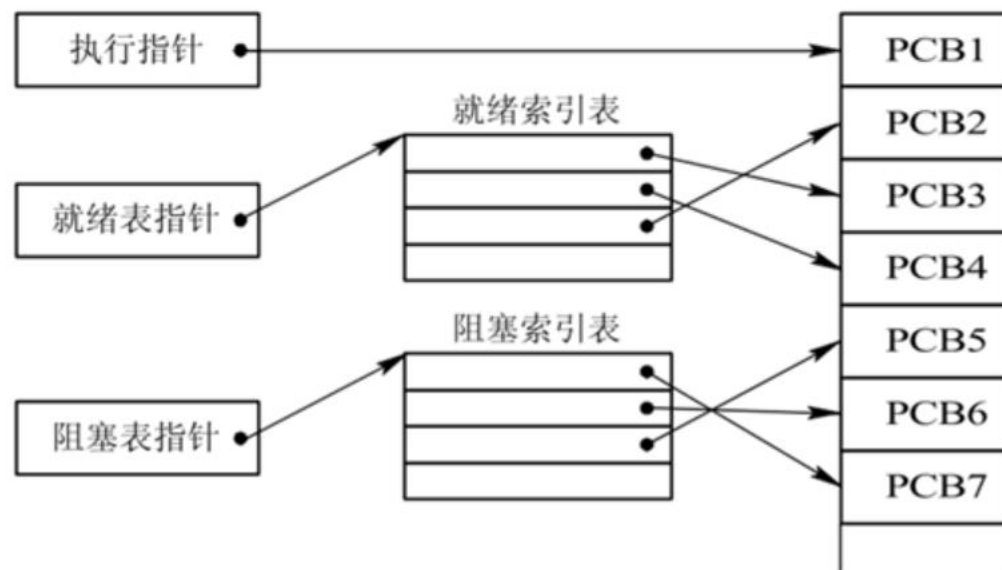
系统环境：地址空间，系统栈，打开文件表，...



四川大学

SICHUAN UNIVERSITY

进程PCB队列



1. 就绪队列：系统一个或若干个（根据调度算法确定）
2. 等待队列：每个等待事件一个
3. 运行队列：每个处理机一个

进程调度与调度算法

Scheduler : 将新进程添加到流程表，并从流程表中删除已完成的流程

Dispatcher : 控制对过程表中的过程分配时间片 - 调度程序是内核的组成部分，可确保计划的进程实际执行。

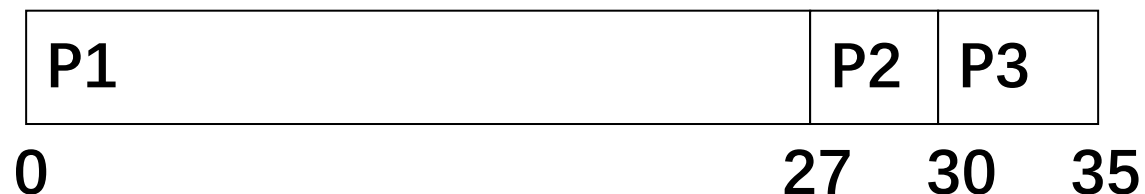


调度算法-FCFS (First Come First Serve)-先到优先

调度最先进入就绪队列的作业。

不利于短作业，造成了短作业等待时间过长。

process	Burst time
P1	27
P2	3
P3	5



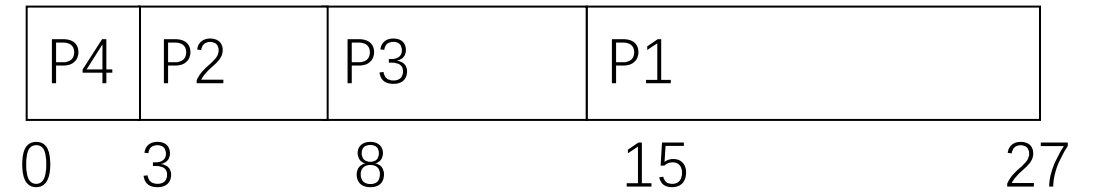
平均等待时间：
 $(0+27+30) / 3 = 19 \text{ (ms)}$



四川大學
SICHUAN UNIVERSITY

短作业优先 (SJF) shortest job first-短的优先

Process	Burst time
P1	12
P2	5
P3	7
P4	3



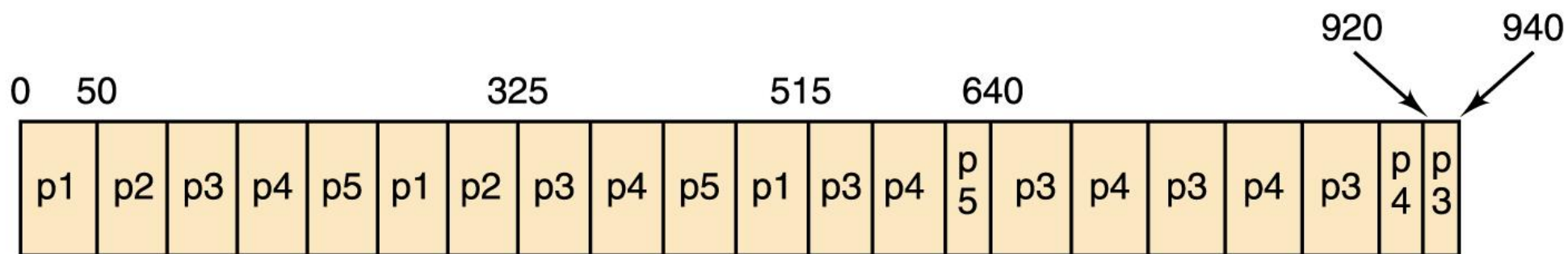
平均等待时间:

$$(0+3+8+15) / 4 = 6.5 \text{ (ms)}$$



时间片轮转（**实时系统**）- Round Robin-大家都一样

- 效率和时间片（**Time slice**）有关
- 每次进程切换要保存上下文，进程切换频繁，会花过多时间。



平均周转时间：

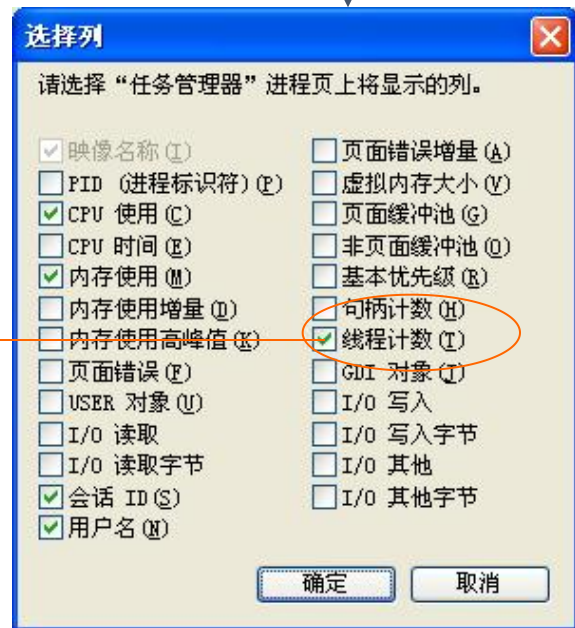
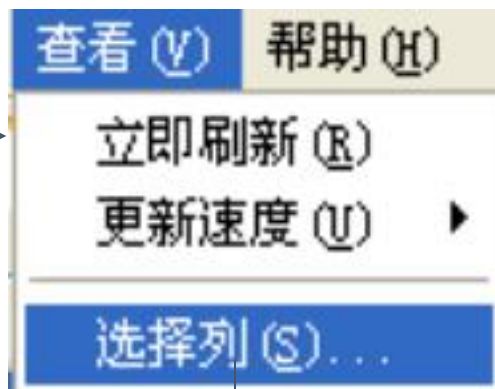
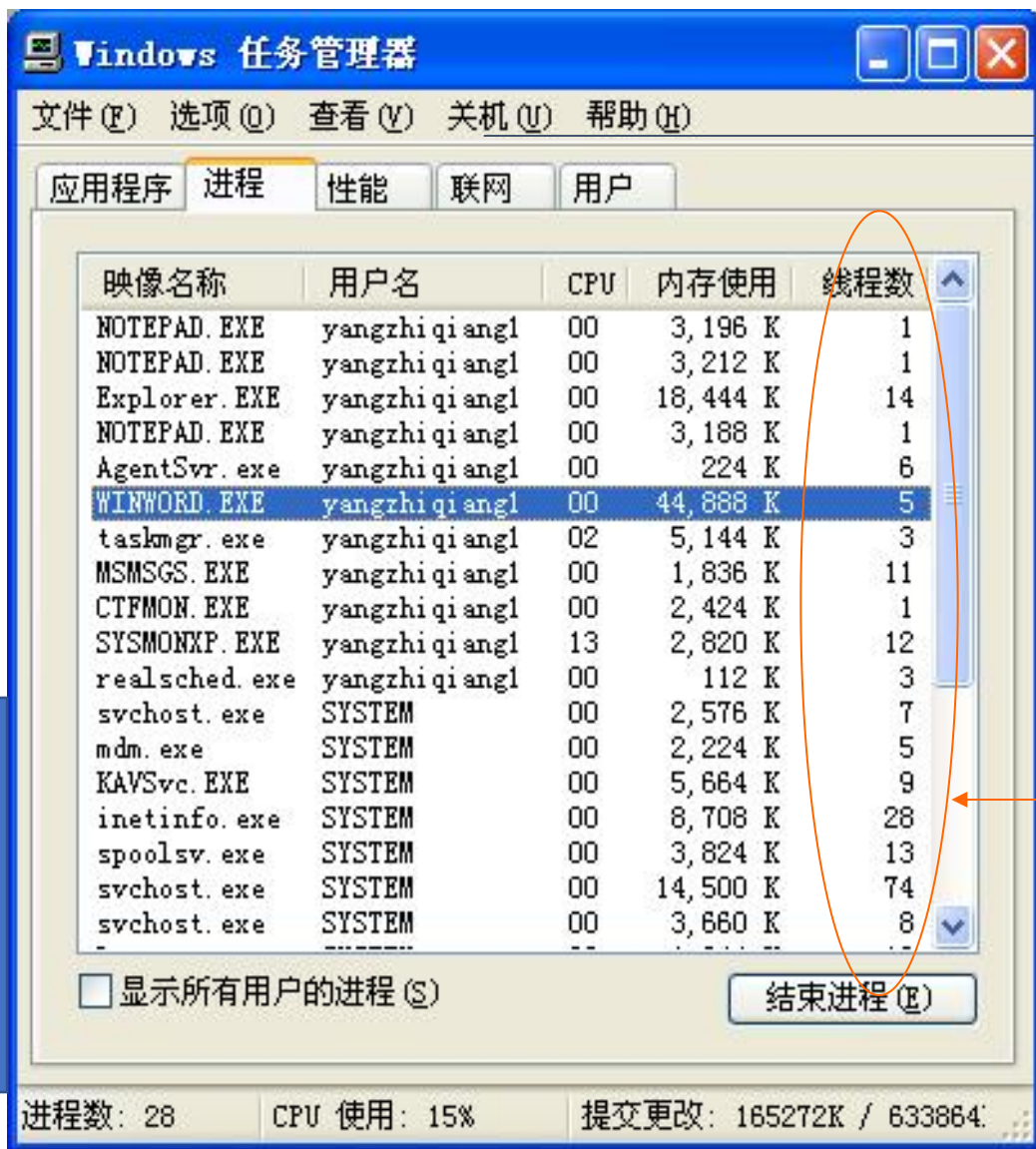
$$(515 + 325 + 940 + 920 + 640) / 5 = 668 \text{ (ms)}$$



3) 进程与线程

- 线程
- 并行与并发

查看进程任务：多线程

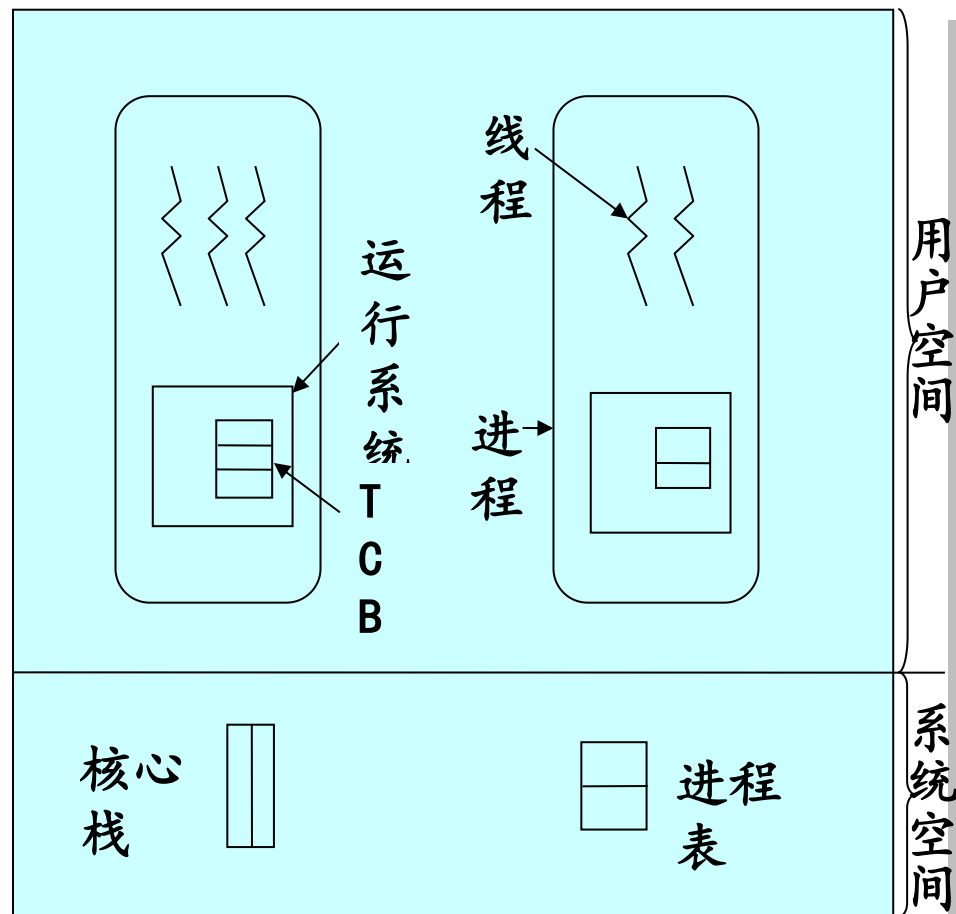


Svchost:
运行系统
调用DLL

四川大学

SICHUAN UNIVERSITY

进程与线程



- 进程中一个相对独立的执行流。
- 进程 / 线程
 - 进程是资源分配单位
 - 线程是最小执行单位
- 线程优点
 - 切换速度快
 - 系统开销小
 - 通讯容易（共享数据空间）



进程与线程

- 进程是资源提供者;
- 线程共享进程空间, 是任务的执行者;
- 进程的系统开销比较大, 线程则小;
- 进程之间通信比较麻烦, 线程 (同一进程内) 方便。



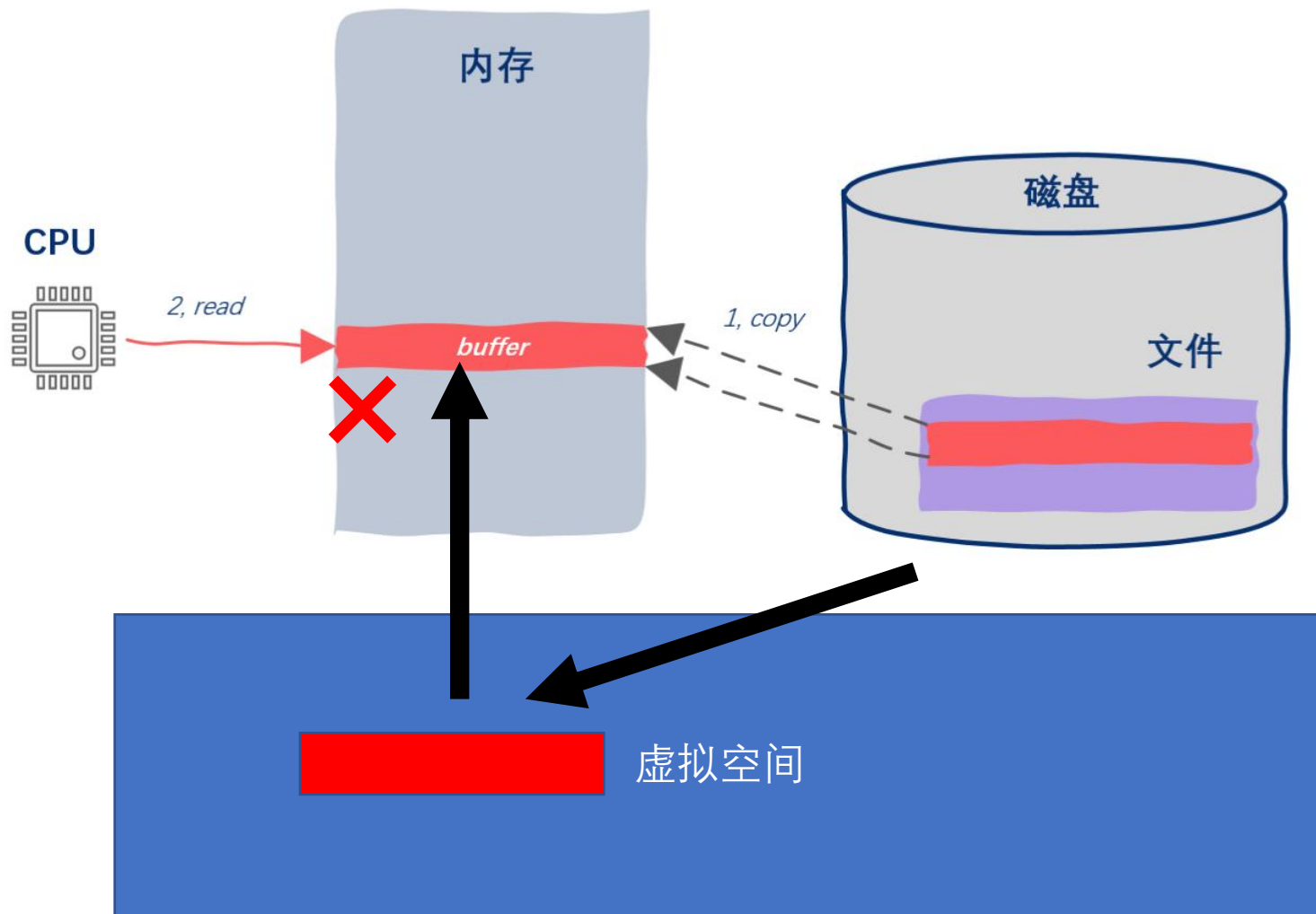
2、存储管理

- 内存虚拟化
- 层次化管理：层次结构
- 存储系统地址管理机制
 - 虚实转换
 - 段页管理
 - 替换算法



1) 虚拟内存:映射到虚拟空间-实体空间（内存）创新：空间转换

虚拟内存是计算机系统内存管理的一种技术。它使得应用程序认为它拥有连续可用的内存（一个连续完整的地址空间），而实际上，它通常是被分隔成多个物理内存碎片，还有部分暂时存储在外部磁盘存储器上，在需要时进行数据交换。



2) 存储系统的层次结构
局部性原理
两种存储系统



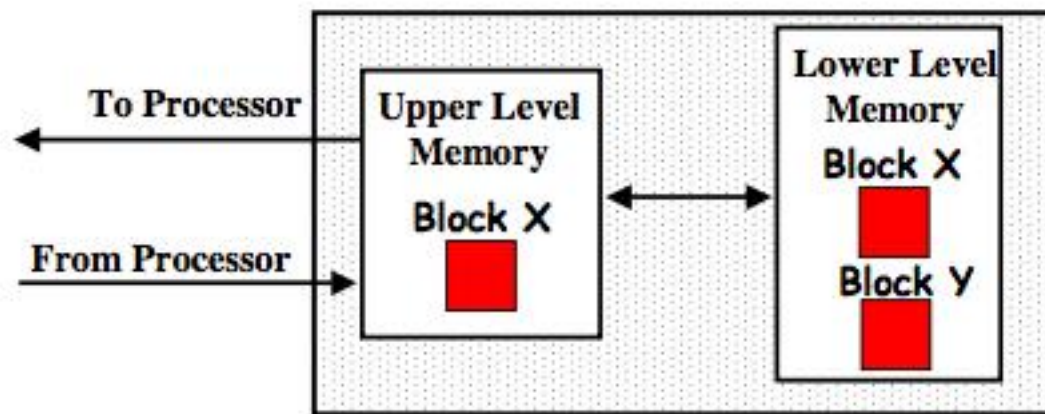
存储系统存在的依据与原理

局部性原理——依据



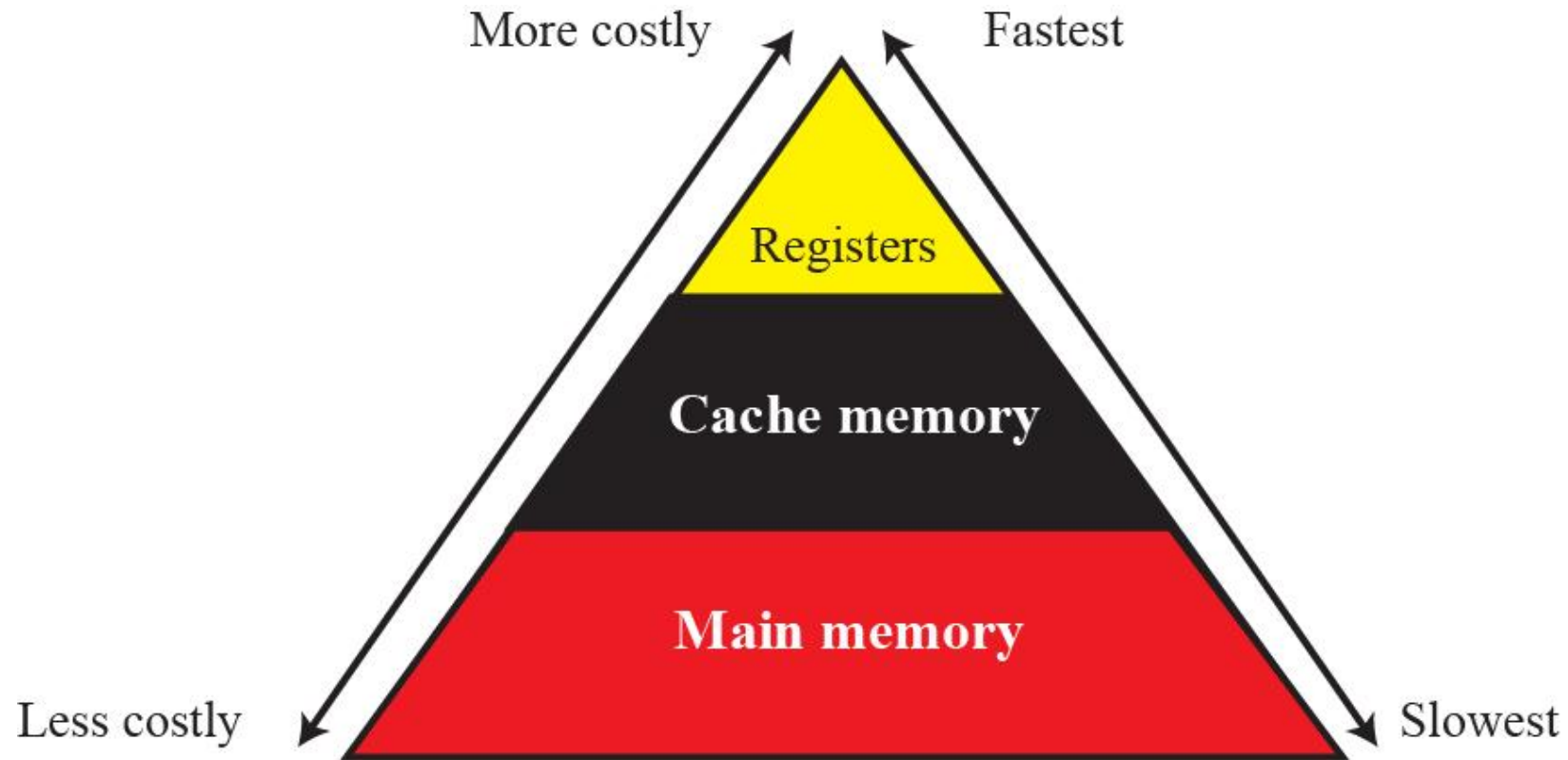
程序执行时所访问的存储器地址分布不是随机的，而是**相对簇集**；这种簇集包括指令和数据两部分。

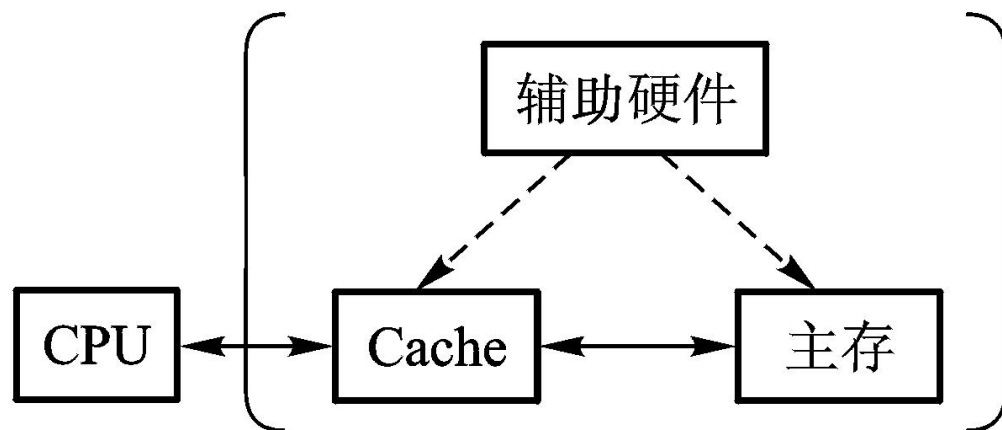
Hit & miss



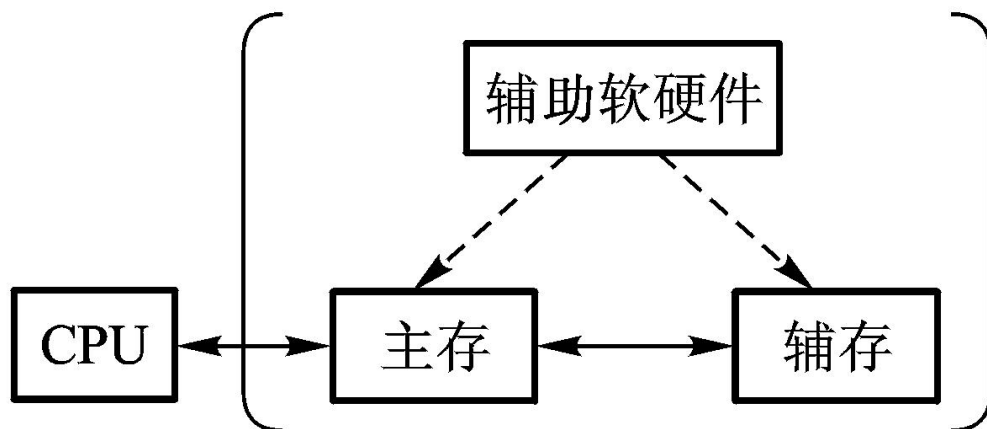
- 存储是多层次的，离Pu越近速度越快
- 临近两层的沟通机制很重要。

Memory hierarchy(存储层次)





(a) “Cache-主存”层次



(b) “主存-辅存”层次

两种存储系统:

- **Cache存储系统**: 由Cache和主存储器构成

主要目的: 提高存储器速度

- **虚拟存储系统**: 由主存储器和磁盘存储器构成

主要目的: 扩大存储器容量

Cache 存储系统与虚拟存储系统的比较

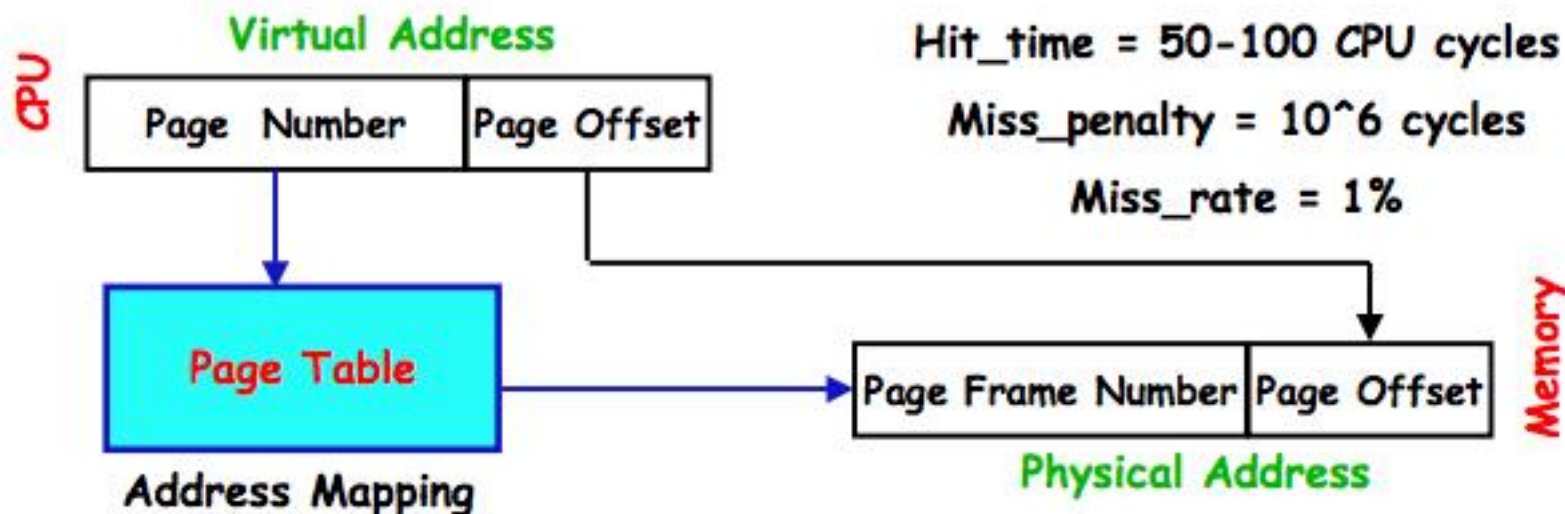
存储系统	Cache	虚拟存储器
要达到的目标	提高速度	扩大容量
实现方法	全部硬件	软件为主, 硬件为辅
两级存储器速度比	3~10 倍	10^5 倍
页 (块) 大小	1~16 字	1KB~16KB
等效存储容量	主存储器	虚拟存储器
透明性	对系统和应用程序员	仅对应用程序员
不命中时处理方式	等待主存储器	任务切换

3) 存储系统地址管理机制

地址变换：虚拟地址与物理地址
段页管理



虚拟地址-实地址



64位机寻址空间有多大？

64位系统寻址空间理论上限是 $2^{64}=16\text{EiB}$ 大小，通常而言，操作系统虚拟地址空间无需这么大的空间，于是目前通常设定64位系统的虚拟地址寻址空间大小最大是48位，也就是 $2^{48}=256\text{TB}$.

虚拟地址

CPU	CPU Address Bus Size	Maximum RAM
8086	20 bit	1MB
8088	20 bit	1MB
80286	24 bit	16MB
80386SX	24 bit	16MB
80386DX	32 bit	4GB
80486SX	32 bit	4GB
80486DX	32 bit	4GB
Pentium I	32 bit	4GB
K6	32 bit	4GB
Duron	32 bit	4GB
Athlon	32 bit	4GB
Athlon XP	32 bit	4GB

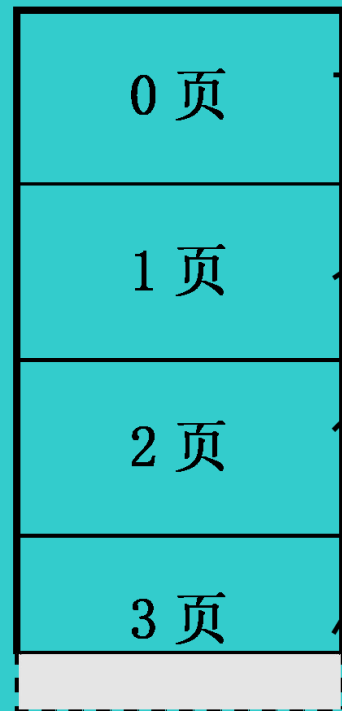
Celeron	36 bit	64GB
Pentium Pro	36 bit	64GB
Pentium II	36 bit	64GB
Pentium III	36 bit	64GB
Pentium 4	36 bit	64GB
Athlon	40 bit	1TB
Athlon-64	40 bit	1TB
Athlon-64 FX	40 bit	1TB
Opteron	40 bit	1TB
Itanium	44 bit	16TB
Itanium 2	44 bit	16TB

段页管理

- 页表
- 段表
- 段页表



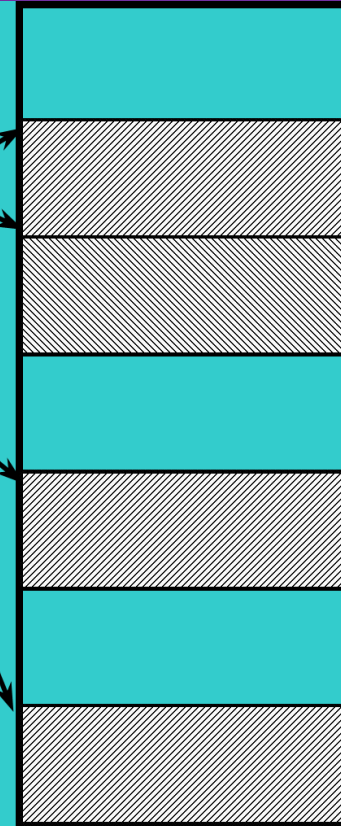
页表



用户程序

页号	主存页号
0	
1	
2	
3	

页 表



主存储器

页式虚拟存储器的地址映象

页式管理优缺点

主要优点：

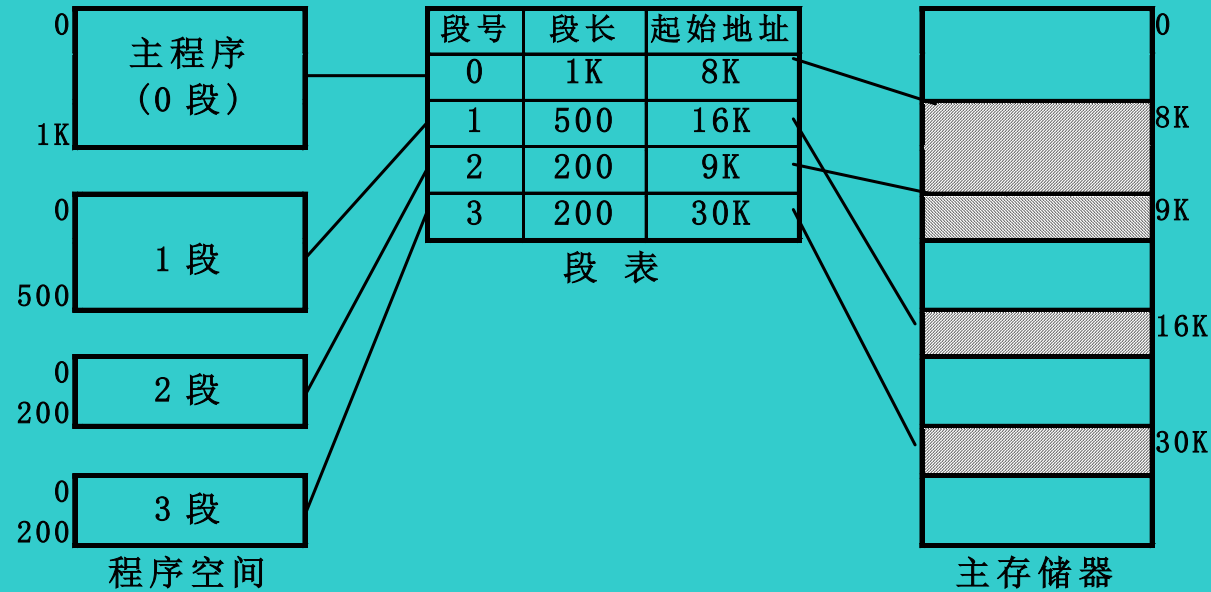
- (1) 主存储器的利用率比较高
- (2) 页表相对比较简单
- (3) 地址变换的速度比较快
- (4) 对磁盘的管理比较容易

主要缺点：

- (1) 程序的模块化性能不好
- (2) 进程变量保护不好。不好统一设置禁止读写某函数或是进程变量
- (3) 页表很长，需要占用很大的存储空间

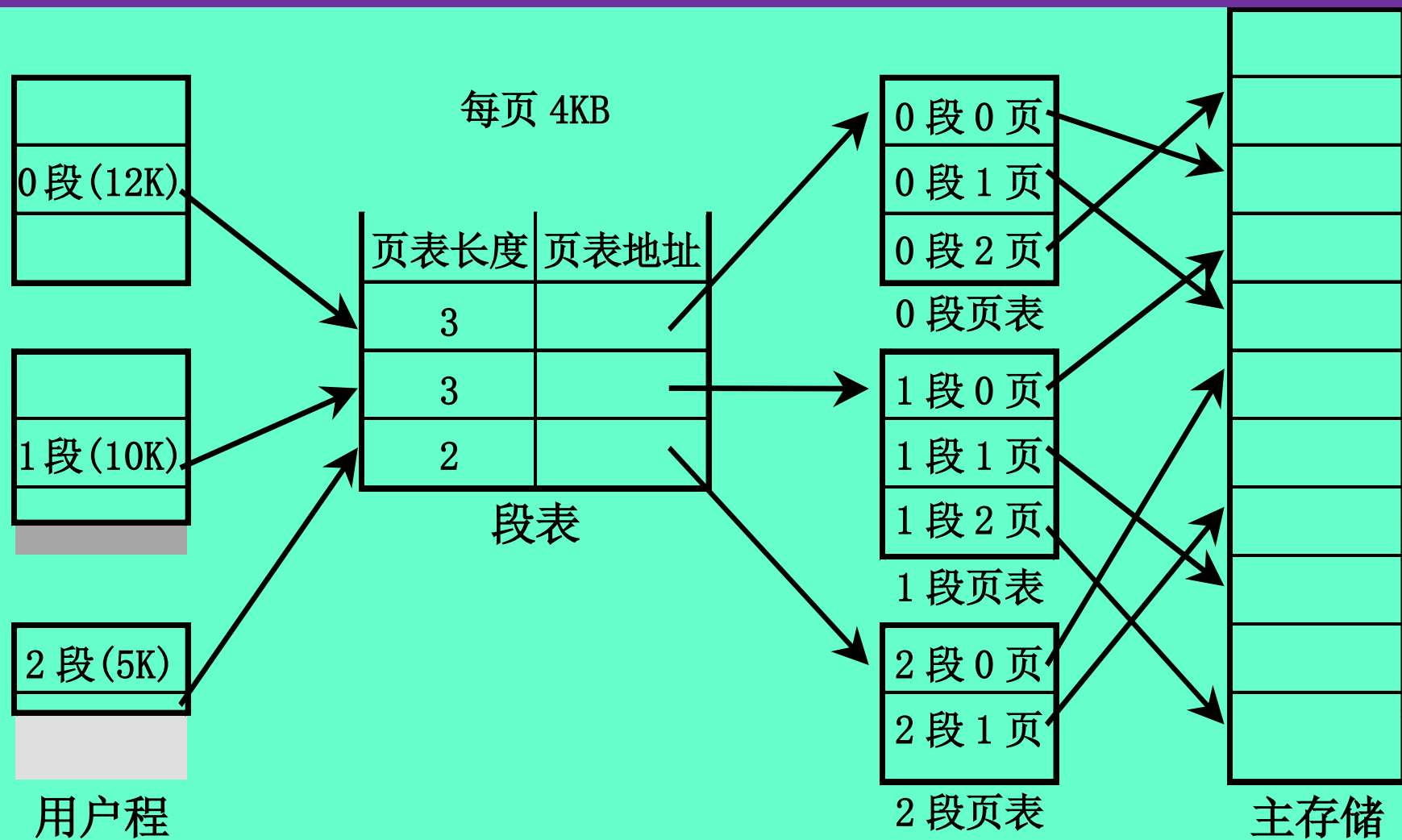


段式管理



地址映象方法：每个程序段都从0地址开始编址，长度可长可短，可以在程序执行过程中动态改变程序段的长度。

段页式管理



段页式虚拟存储器的地址映象

段式管理优缺点

- (1) 程序的模块化性能好。
- (2) 便于程序和数据的**共享**。
- (3) 程序的**动态链接**和调度比较容易。
- (4) 便于实现**信息保护**。

- (1) 地址变换所花费的**时间比较长**，做两次加法运算。
- (2) 主存储器的**利用率**往往比较低。
- (3) 对辅存（磁盘存储器）的**管理**比较困难。



页替换算法：LRU FIFO



四川大學
SICHUAN UNIVERSITY

页替换算法：LRU OPT FIFO



四川大學
SICHUAN UNIVERSITY

- **FIFO, First In First Out, 先入先出算法**, 是最简单的页面替换策略, 它按照页面进入内存的顺序来决定替换哪个页面。即优先淘汰最早进入内存的页面, 不论这些页面之后是否被频繁访问。但是FIFO可能会导致“Belady异常”, 即随着分配给进程的物理块数增加, 缺页次数反而增加。
- **OPT, Optimal Page Replacement, 最佳页面替换算法**, 是一种基于全局信息做出决策的页面替换算法, 每次选择未来最长时间不再被访问的页面进行替换, 可以达到最低的缺页率。
- **LRU, Least Recently Used, 最近最少使用算法**, 如果一个数据最近被访问过, 那么将来被访问的可能性也较大。因此, 它选择最近最长时间未被访问的页面进行替换。LRU的性能和效率接近OPT, 但是对于频繁访问的页面更新开销较大。



LRU vs. FIFO

page size: 4 words; 3 page frames: a,b,c; 10 pages: 0,1,2, ..., 9

Word Trace: 0,1,2,3, 4,5,6,7, 8, 16,17, 9,10,11, 12, 28,29,30, 8,9,10, 4,5, 12, 4,5
 ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
 Page Trace: 0 1 2 4 2 3 7 2 1 3 1

	PF	0	1	2	4	2	3	7	2	1	3	1	Hit Rate
LRU	a	0	0	0	4	4	4	7	7	7	3	3	
	b		1	1	1	1	3	3	3	1	1	1	
	c			2	2	2	2	2	2	2	2	2	
	Faults	*	*	*	*		*	*		*	*		
OPT	a	0	0	0	4	4	3	7	7	7	3	3	
	b		1	1	1	1	1	1	1	1	1	1	
	c			2	2	2	2	2	2	2	2	2	
	Faults	*	*	*	*		*	*			*		
FIFO	a	0	0	0	4	4	4	4	2	2	2	2	
	b		1	1	1	1	3	3	3	1	1	1	
	c			2	2	2	2	7	7	7	3	3	
	Faults	*	*	*	*		*	*	*	*	*		

3: 中断管理

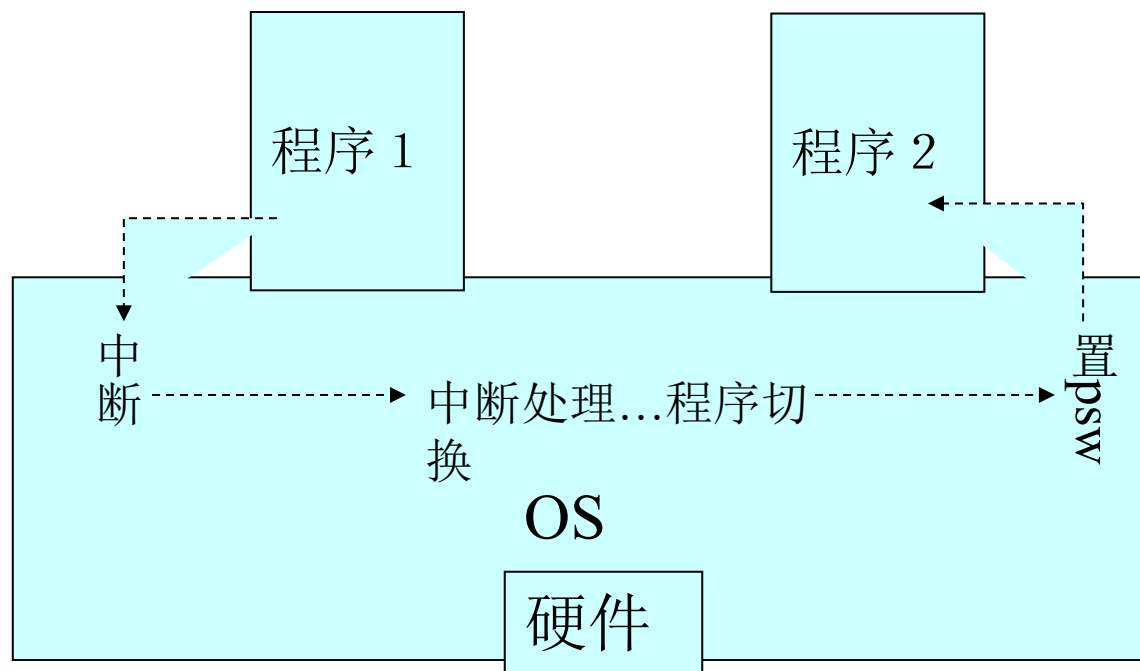
中断管理机制

中断类型与中断向量



四川大學
SICHUAN UNIVERSITY

1) 中断管理机制：本质上是调用一个内部函数处理中断



中断系统：

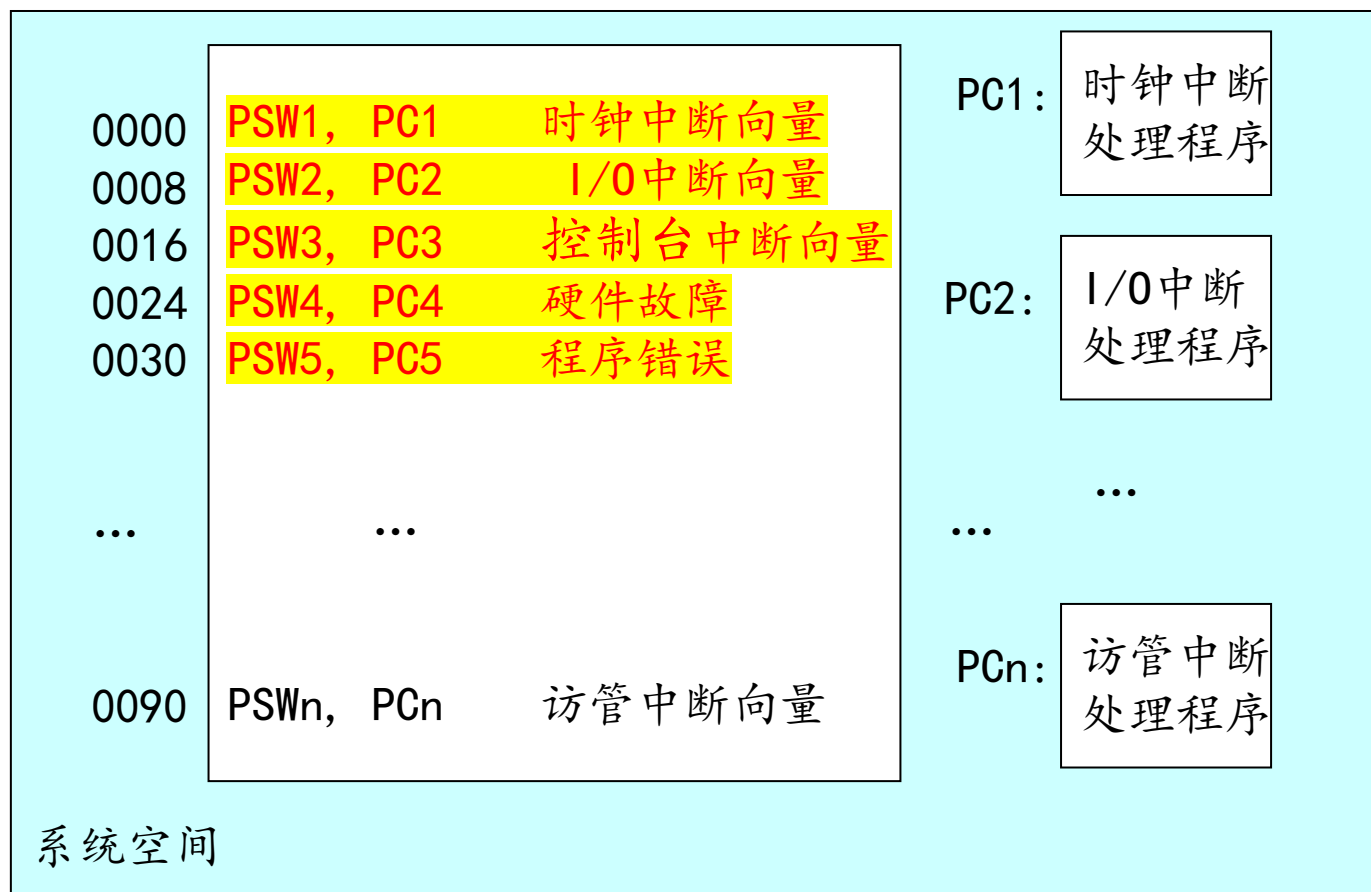
保存上下文 -> 修改栈指针 ->
恢复上下文

Fork-execve

调度下一个剧目



2) 中断类型与中断向量



- 每类中断事件有一个中断向量，
- 中断向量的存放位置是由硬件规定的，
- 中断向量的内容是OS在系统初始化时设置好的。

中断类型

强迫性中断

运行程序不期望的

时钟中断、IO中断、控制台中断

硬件故障中断 (power failure、内存校验错)
程序性中断 (越界, 越权、非法指令)

自愿性中断

运行程序期望的

系统调用:

`fd=open(fname, mode)`

访管指令: 准备参数



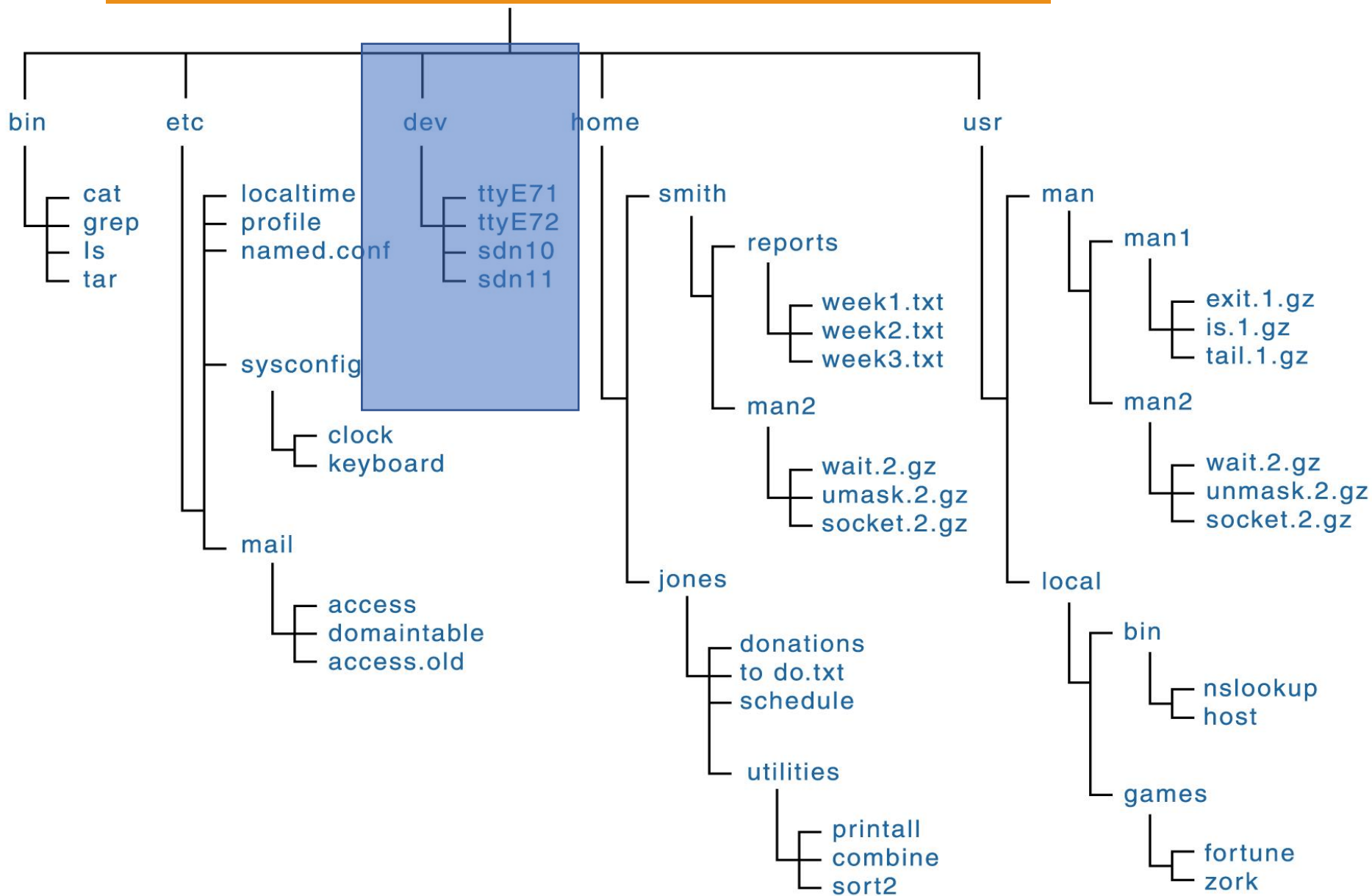
四川大学

SICHUAN UNIVERSITY

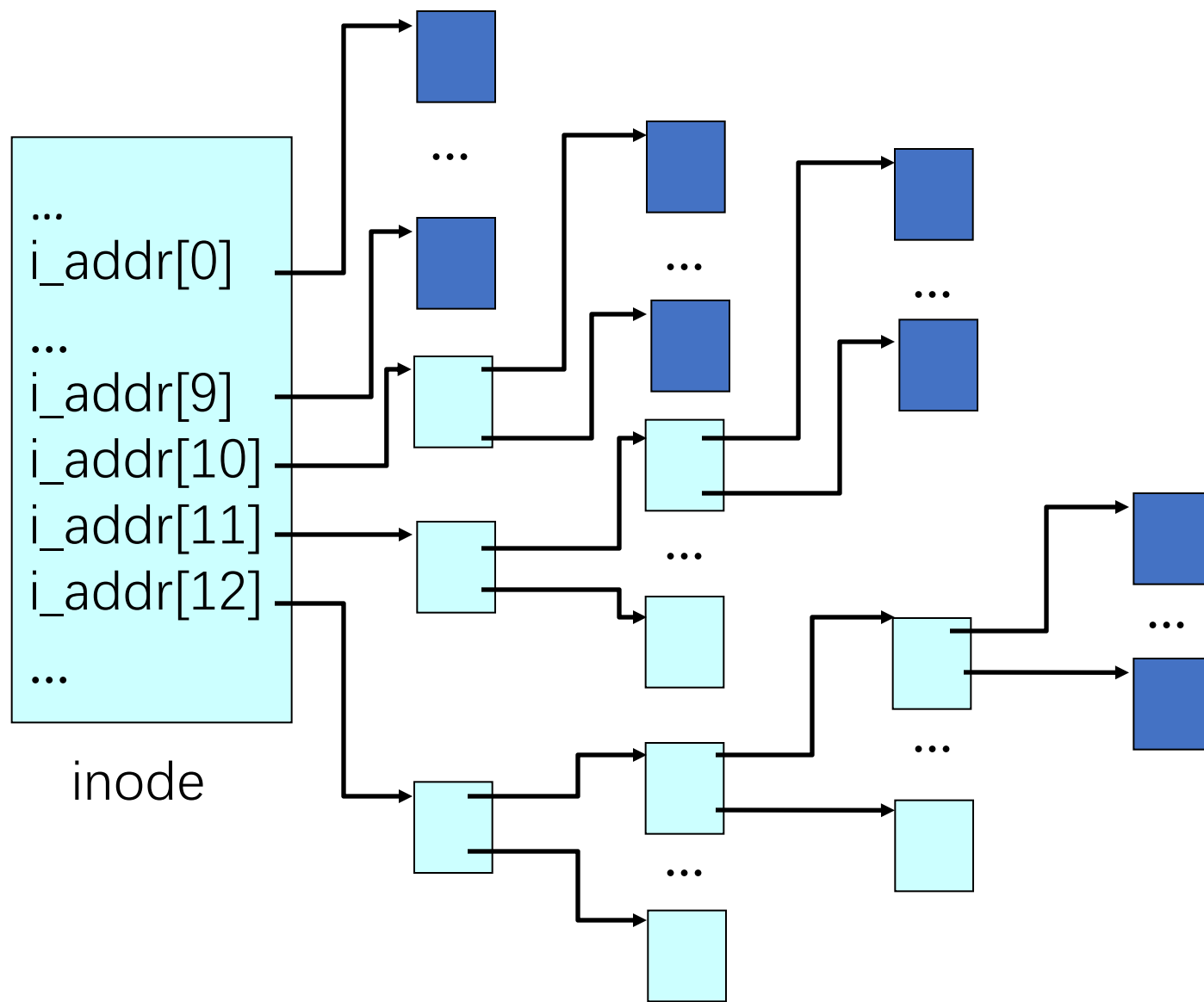
4. 文件系统管理

- 文件系统(结构与索引)
- 存储 (格式与路径)

文件系统：unix 目录树（设备文件）

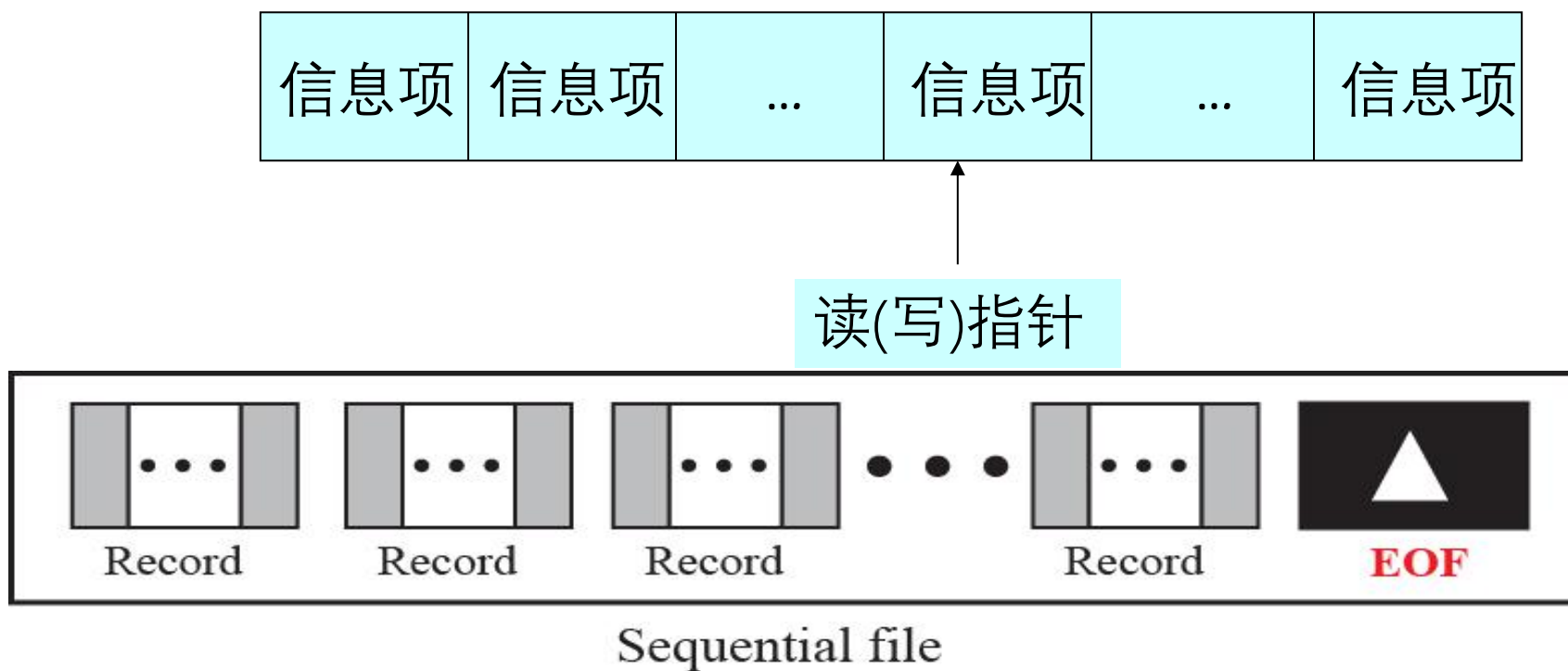


UNIX文件物理结构(索引+链接)



顺序存取

直接访问 通过指定逻辑记录号将文件中的数据概念上划分为编号的逻辑记录并直接访问的技术



2) 存储格式与路径

文本文件与二进制文件

- Text file(文本文件): 包含ASCII或Unicode字符集中的字符的文件
- Binary file (二进制文件): 包含特定格式数据的文件, 需要对其位进行特殊解释

01000000_01000001_01000010_01000011按ASCII “ABCD”



路径

Absolute paths (绝对路径)

- C:\Program Files\MS Office\WinWord.exe
- C:\My Documents\letters\applications\vaTech.doc
- C:\Windows\System\QuickTime
- 目前的绝对路径-C:\My Documents\letters
- 相对路径
 - cancelMag.doc
 - applications\calState.doc

