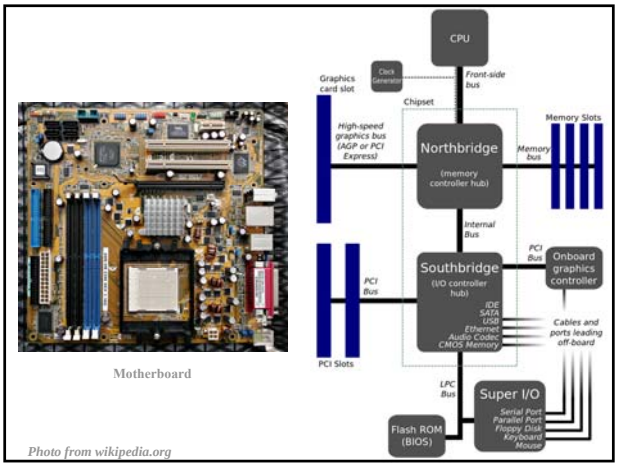
Operating System

Dr. GuoJun LIU

Harbin Institute of Technology

http://guojun.hit.edu.cn/os/

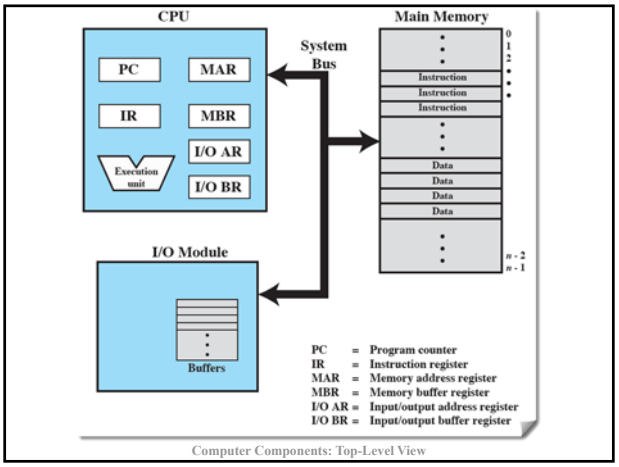


Motherboard

Photo from wikipedia.org

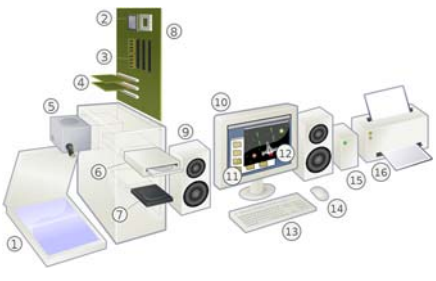
My Summary of OS

… …



Computer Components: Top-Level View

A modern PC and peripherals

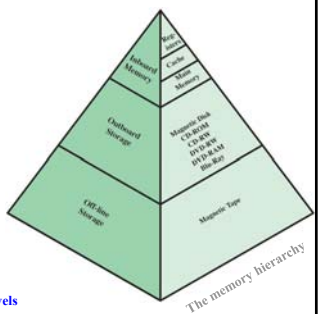


An exploded view of a modern personal computer and peripherals

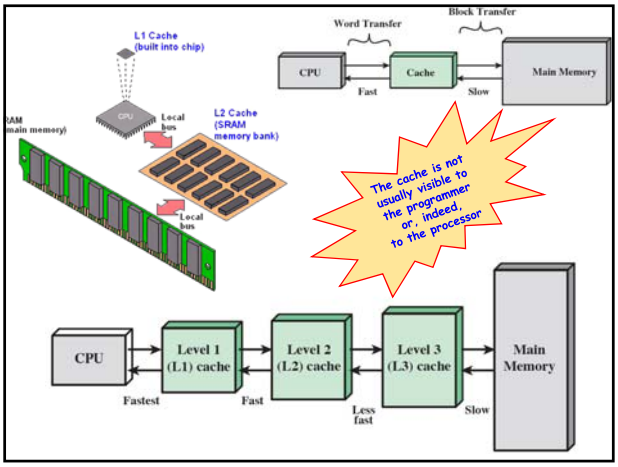
1. Scanner
2. CPU
3. Memory (RAM)
4. Expansion cards (graphics cards, etc.)
5. Power supply
6. Optical disc drive
7. Storage (Hard disk)
8. Motherboard
9. Speakers
10. Monitor
11. System software
12. Application software
13. Keyboard
14. Mouse
15. External hard disk
16. Printer

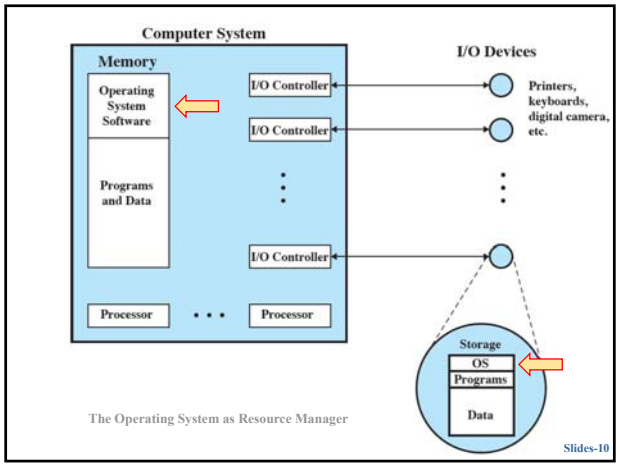
The Memory Hierarchy

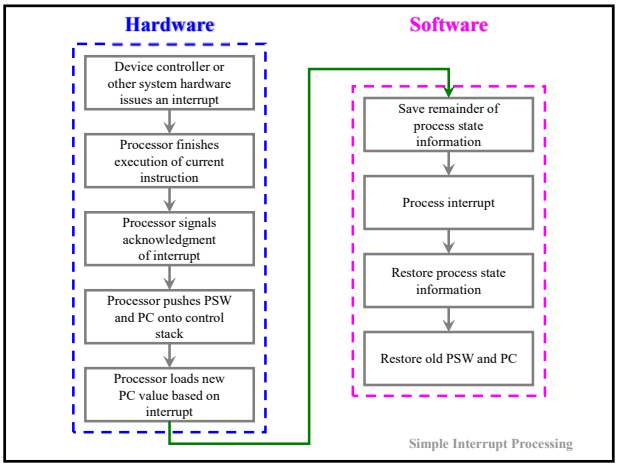
- **Going down the hierarchy:**
  - a. decreasing cost per bit
  - b. increasing capacity
  - c. increasing access time
  - d. decreasing frequency of access to the memory by the processor

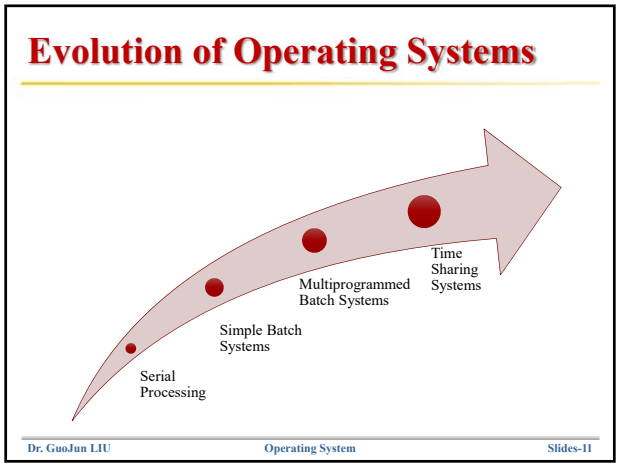


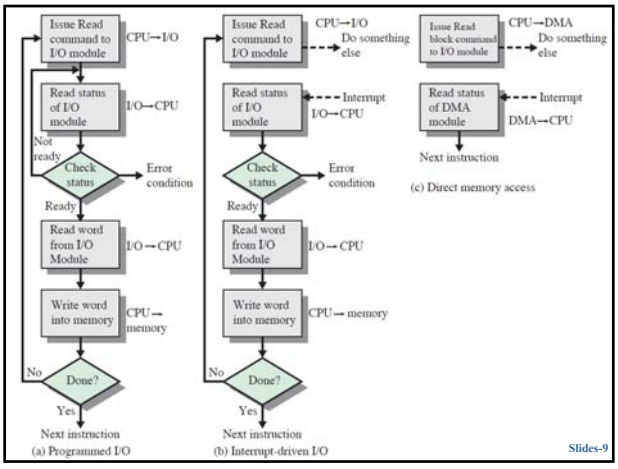
Decreasing frequency of access at lower levels

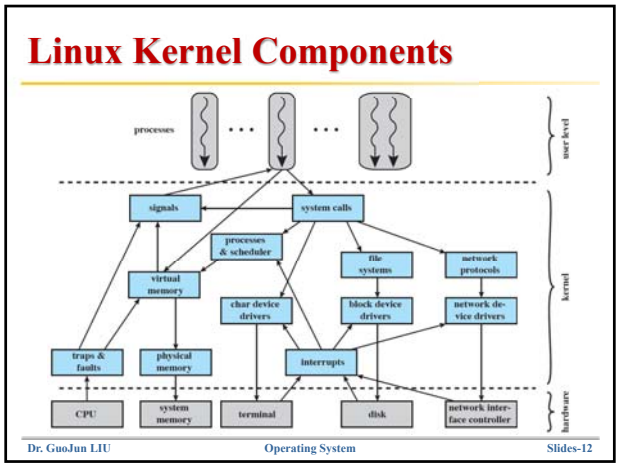

L1 Cache
(built into chip)
Word Transfer
Block Transfer
CPU
Cache
Main Memory
Fast
Slow
Local bus
L2 Cache
(SRAM
memory bank)
The cache is not usually visible to the programmer or, indeed, to the processor
CPU
Level 1 (L1) cache
Level 2 (L2) cache
Level 3 (L3) cache
Main Memory
Fastest
Fast
Less fast
Slow

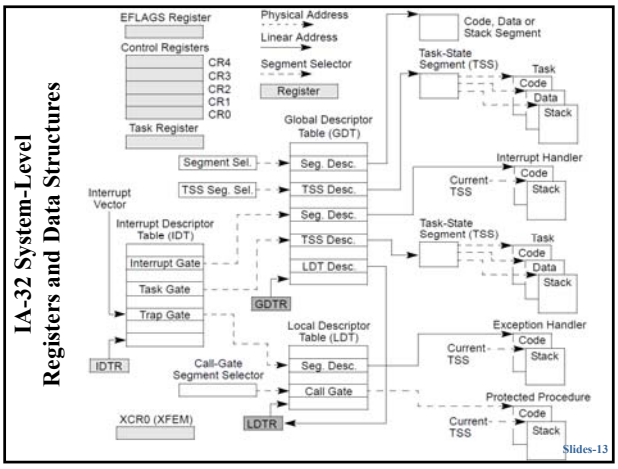

Computer System
Memory
Operating System Software
Programs and Data
I/O Controller
I/O Controller
I/O Controller
Processor
Processor
I/O Devices
Printers, keyboards, digital camera, etc.
Storage
OS
Programs
Data
The Operating System as Resource Manager
Slides-10

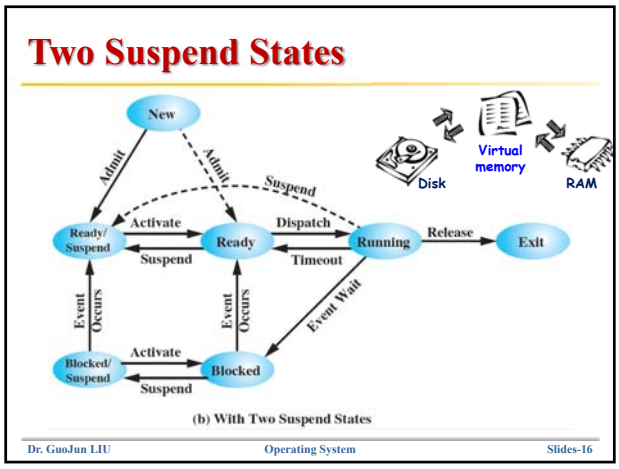

Hardware
Software
Device controller or other system hardware issues an interrupt
Processor finishes execution of current instruction
Processor signals acknowledgment of interrupt
Processor pushes PSW and PC onto control stack
Processor loads new PC value based on interrupt
Save remainder of process state information
Process interrupt
Restore process state information
Restore old PSW and PC
Simple Interrupt Processing

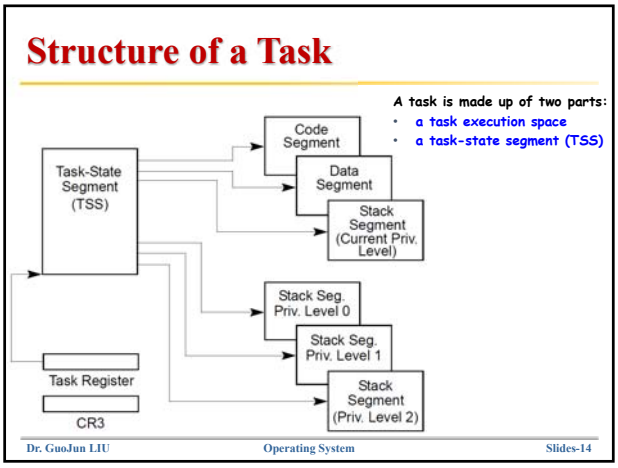

Evolution of Operating Systems
Time Sharing Systems
Multiprogrammed Batch Systems
Simple Batch Systems
Serial Processing
Dr. GuoJun LIU
Operating System
Slides-11

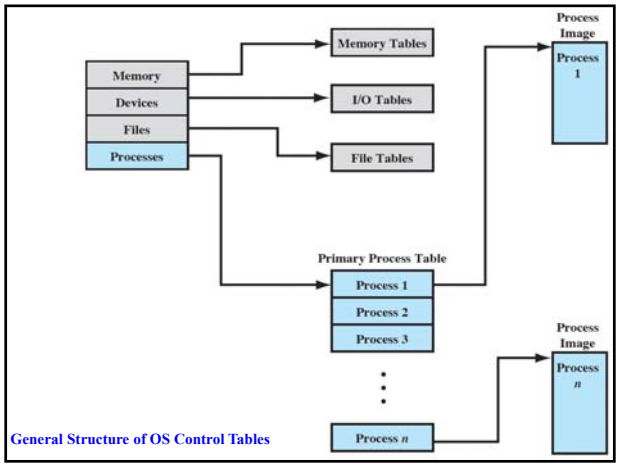

Issue Read command to I/O module
CPU→I/O
Read status of I/O module
I/O→CPU
Not ready
Check status
Error condition
Ready
Read word from I/O Module
I/O→CPU
Write word into memory
CPU→memory
No
Done?
Yes
Next instruction
(a) Programmed I/O
Issue Read command to I/O module
CPU→I/O
Do something else
Read status of I/O module
Interrupt
I/O→CPU
Check status
Error condition
Ready
Read word from I/O Module
I/O→CPU
Write word into memory
CPU→memory
No
Done?
Yes
Next instruction
(b) Interrupt-driven I/O
Issue Read block command to I/O module
CPU→DMA
Do something else
Read status of DMA module
Interrupt
DMA→CPU
Next instruction
(c) Direct memory access
Slides-9

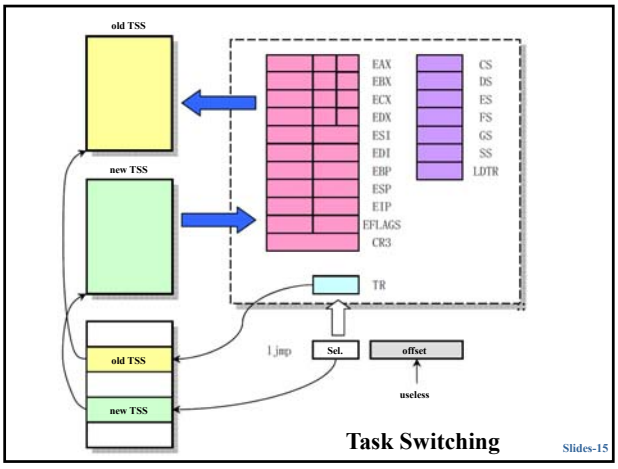

Linux Kernel Components
processes
signals
system calls
processes & scheduler
virtual memory
file systems
network protocols
char device drivers
block device drivers
network device drivers
traps & faults
physical memory
interrupts
CPU
system memory
terminal
disk
network interface controller
user level
kernel
hardware
Dr. GuoJun LIU
Operating System
Slides-12

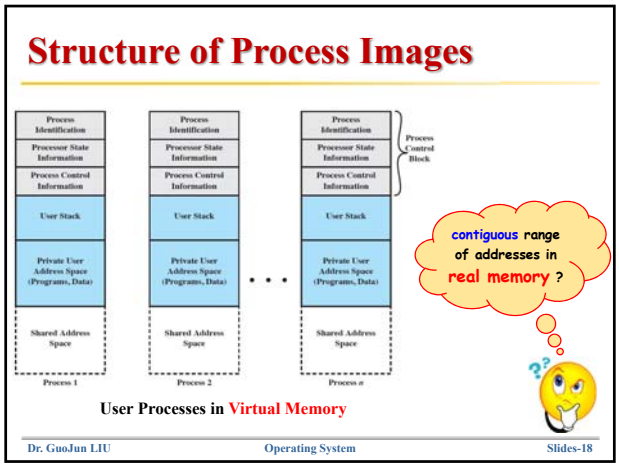

IA-32 System-Level
Registers and Data Structures
EFLAGS Register
Control Registers
CR4
CR3
CR2
CR1
CR0
Task Register
Physical Address
Linear Address
Segment Selector
Register
Global Descriptor Table (GDT)
Segment Sel.
TSS Seg. Sel.
Interrupt Vector
Interrupt Descriptor Table (IDT)
Interrupt Gate
Task Gate
Trap Gate
IDTR
GDTR
Seg. Desc.
TSS Desc.
Seg. Desc.
TSS Desc.
LDT Desc.
Local Descriptor Table (LDT)
Call-Gate Segment Selector
Seg. Desc.
Call Gate
LDTR
XCR0 (XFEM)
Code, Data or Stack Segment
Task-State Segment (TSS)
Task
Code
Data
Stack
Interrupt Handler
Current-TSS
Exception Handler
Protected Procedure
Slides-13


Two Suspend States
New
Admit
Admit
Suspend
Ready/ Suspend
Activate
Ready
Dispatch
Running
Release
Exit
Suspend
Timeout
Event Occurs
Event Occurs
Event Wait
Blocked/ Suspend
Activate
Blocked
Suspend
Disk
Virtual memory
RAM
(b) With Two Suspend States
Dr. GuoJun LIU
Operating System
Slides-16

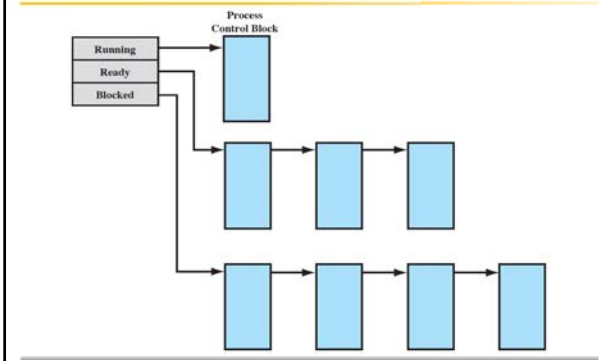

Structure of a Task
A task is made up of two parts:
a task execution space
a task-state segment (TSS)
Task-State Segment (TSS)
Code Segment
Data Segment
Stack Segment (Current Priv. Level)
Stack Seg. Priv. Level 0
Stack Seg. Priv. Level 1
Stack Segment (Priv. Level 2)
Task Register
CR3
Dr. GuoJun LIU
Operating System
Slides-14


Memory
Devices
Files
Processes
Memory Tables
I/O Tables
File Tables
Process Image
Process 1
Primary Process Table
Process 1
Process 2
Process 3
Process n
Process Image
Process n
General Structure of OS Control Tables

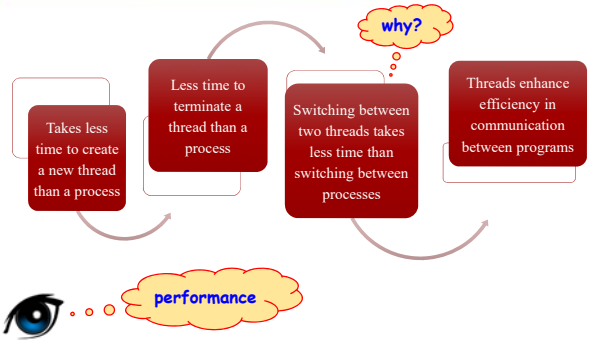

old TSS
new TSS
EAX
EBX
ECX
EDX
ESI
EDI
EBP
ESP
EIP
EFLAGS
CR3
CS
DS
ES
FS
GS
SS
LDTR
TR
old TSS
new TSS
1.jmp
Sel.
offset
useless
Task Switching
Slides-15


Structure of Process Images
Process Identification
Processor State Information
Process Control Information
User Stack
Private User Address Space (Programs, Data)
Shared Address Space
Process Control Block
Process 1
Process 2
Process n
contiguous range of addresses in real memory ?
User Processes in Virtual Memory
Dr. GuoJun LIU
Operating System
Slides-18

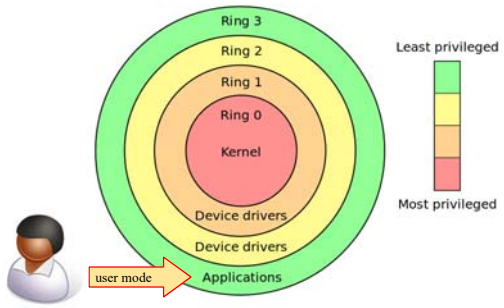
Process List Structures



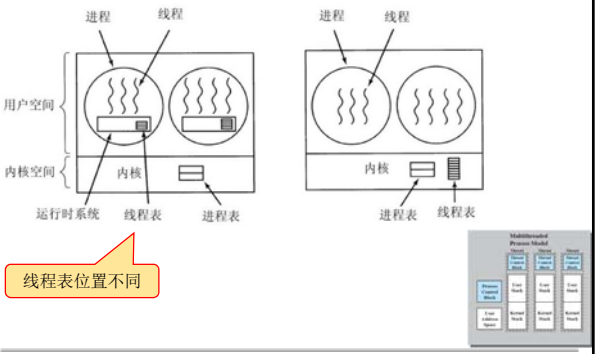
Key Benefits of Threads



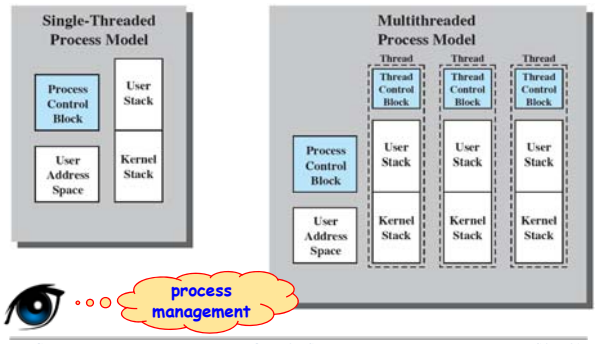
Modes of Execution



ULT vs. KLT



Threads vs. Processes



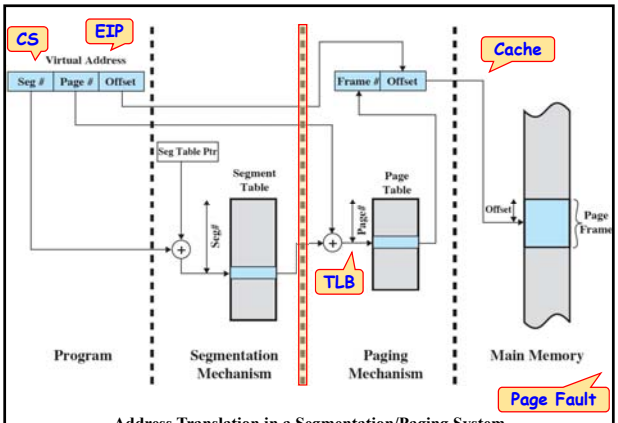
A Solution to the Infinite-Buffer Producer/Consumer Problem Using Semaphores

```
/* program  producerconsumer */
semaphore n = 0, s = 1;
void producer()
{
     while (true) {
          produce();
          semWait(s);
          append();
          semSignal(s);
          semSignal(n);
     }
}
void consumer()
{
     while (true) {
          semWait(n);
          semWait(s);
          take();
          semSignal(s);
          consume();
     }
}
void main()
{
     parbegin (producer, consumer);
}
```

Which one produces a serious flaw ?

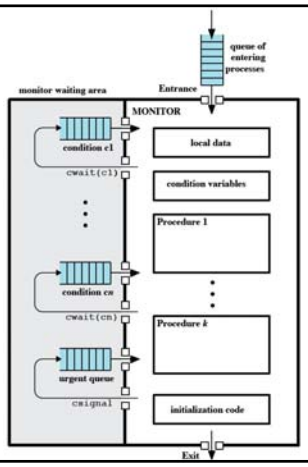
信号量总结

- 信号量总是成对出现的
- 互斥信号量在一个进程内，同步信号量分散在不同的进程
- 同步信号量 semWait 操作在互斥信号量之前，semSignal 顺序不重要
- 不要把同步信号量加在互斥信号量之间，否则容易死锁
- 要求会写伪代码

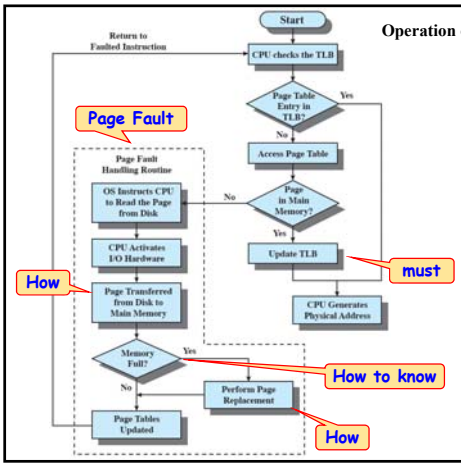


Address Translation in a Segmentation/Paging System

Structure of a Monitor



Operation of Paging and TLB



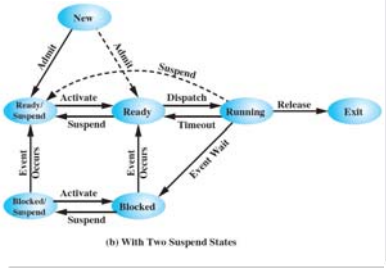
Deadlock

(a) 遇到危险的鸵鸟

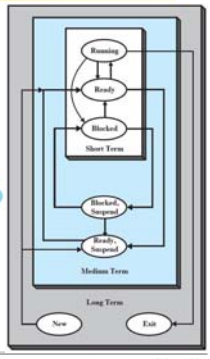
(b) 假装什么都没看见

图 6.1 鸵鸟算法

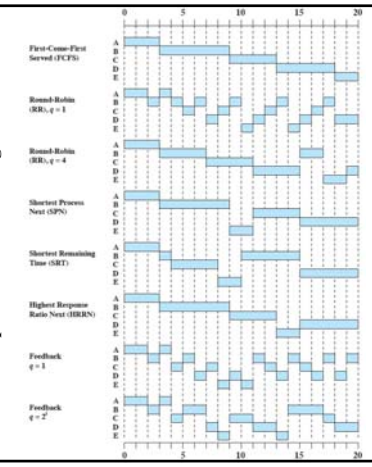
Nesting of Scheduling Functions



(b) With Two Suspend States

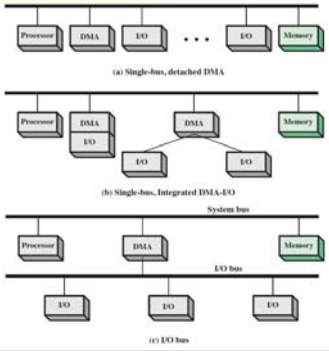


A Comparison of Scheduling Policies



Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

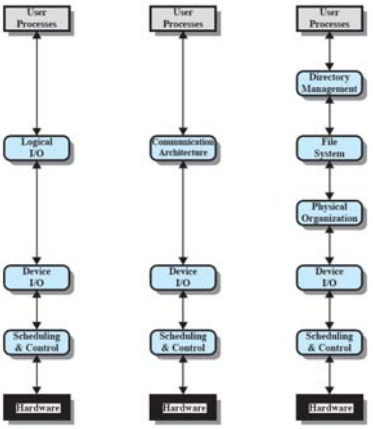
Alternative DMA Configurations



A Comparison of Scheduling Policies

Process	A	B	C	D	E	
Arrival Time	0	2	4	6	8	
Service Time (Ts)	3	6	4	5	2	Mean
FCFS						
Finish Time	3	9	13	18	20	
Turnaround Time (Tr)	3	7	9	12	12	8.60
Tr/Ts	1.00	1.17	2.25	2.40	6.00	2.56
RR q = 1						
Finish Time	4	18	17	20	15	
Turnaround Time (Tr)	4	16	13	14	7	10.80
Tr/Ts	1.33	2.67	3.25	2.80	3.50	2.71
RR q = 4						
Finish Time	3	17	11	20	19	
Turnaround Time (Tr)	3	15	7	14	11	10.00
Tr/Ts	1.00	2.5	1.75	2.80	5.50	2.71
SPN						
Finish Time	3	9	15	20	11	
Turnaround Time (Tr)	3	7	11	14	3	7.60
Tr/Ts	1.00	1.17	2.75	2.80	1.50	1.84
SRT						
Finish Time	3	15	8	20	10	
Turnaround Time (Tr)	3	13	4	14	2	7.20
Tr/Ts	1.00	2.17	1.00	2.80	1.00	1.59
HRRN						
Finish Time	3	9	13	20	15	
Turnaround Time (Tr)	3	7	9	14	7	8.00
Tr/Ts	1.00	1.17	2.25	2.80	3.5	2.14
FB q = 1						
Finish Time	4	20	16	19	11	
Turnaround Time (Tr)	4	18	12	13	3	10.00
Tr/Ts	1.33	3.00	3.00	2.60	1.5	2.29

A Model of I/O Organization



Typical I/O Device Data Rates

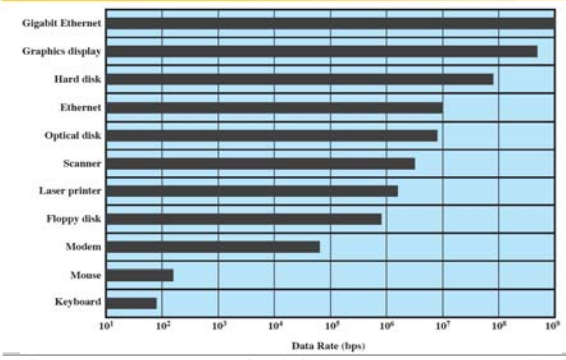
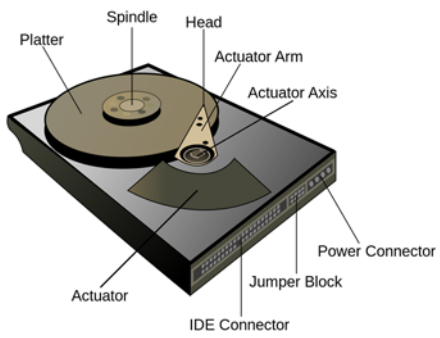
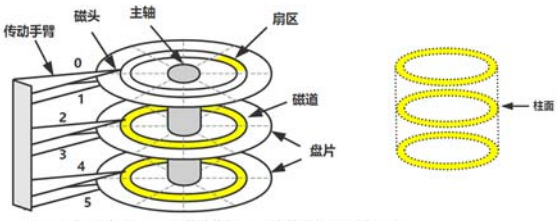


Diagram labeling the major components of HDD





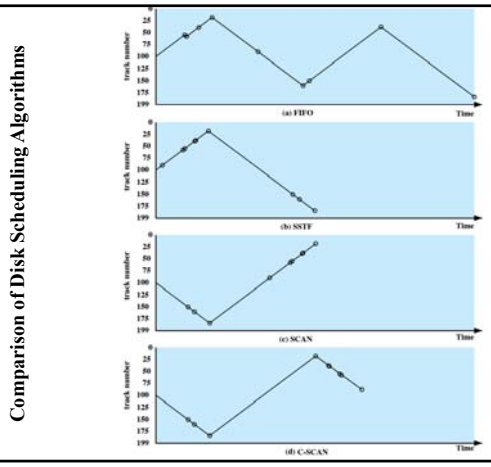
9.3.3.4 扇区号与柱面号、当前磁道扇区号和当前磁头号的对应关系

假定硬盘的每磁道扇区数是 track_secs，硬盘磁头总数是 dev_heads，磁道总数是 tracks。对于指定的硬盘顺序扇区号是 sector，对应的当前柱面号是 cyl，在当前磁道上的扇区号是 sec，当前磁头号是 head。那么若想从指定顺序扇区号 sector 换算成对应的当前柱面号、当前磁道上扇区号以及当前磁头号，则可以使用以下步骤：

- sector / track_secs = 整数是 tracks，余数是 sec；
- tracks / dev_heads = 整数是 cyl，余数是 head；
- 在当前磁道上扇区号从 1 算起，于是需要把 sec 增 1。

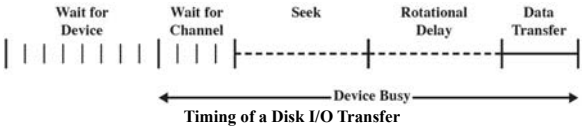
若想从指定的当前 cyl、sec 和 head 换算成从硬盘开始算起的顺序扇区号，则过程正好与上述相反。换算公式和上面给出的完全一样，即：

```
sector = (cyl * dev_heads + head) * track_secs + sec -1
```

Disk Performance Parameters

- The actual details of disk I/O operation depend on
  - computer system
  - operating system
  - nature of the I/O channel and disk controller hardware



Timing of a Disk I/O Transfer

T = transfer time,
b = number of bytes to be transferred,
N = number of bytes on a track, and
r = rotation speed, in revolutions per second.

$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

Comparison of Disk Scheduling Algorithms

(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) SCAN (starting at track 100, in the direction of increasing track number)		(d) C-SCAN (starting at track 100, in the direction of increasing track number)	
Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length	55.3	Average seek length	27.5	Average seek length	27.8	Average seek length	35.8

举例：一个典型的磁盘

- 平均寻道时间 4 ms
- 转速 7500 rpm
- 每个磁道有 500 个扇区、每个扇区有 512 个字节
- 假设读取一个包含2500 个扇区，大小1.28M的文件
- 估计传送需要的总时间

$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

Average seek	4 ms
Rotational delay	4 ms
Read 500 sectors	8 ms
	16 ms

Total time = 16 + (4 × 12) = 64 ms = 0.064 seconds

Average seek	4 ms
Rotational delay	4 ms
Read 1 sector	0.016 ms
	8.016 ms

Total time = 2,500 × 8.016 = 20,040 ms = 20.04 seconds

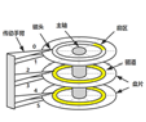
硬盘分区表

表 9-10 硬盘主引导扇区 MBR 的结构

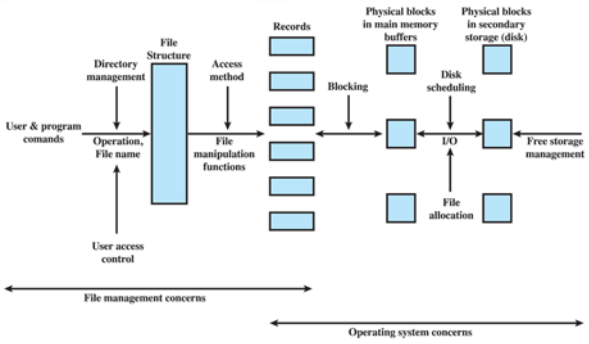
偏移位置	名称	长度（字节）	说明
0x000	MBR 代码	446	引导程序代码和数据。
0x1BE	分区表项 1	16	第 1 个分区表项，共 16 字节。
0x1CE	分区表项 2	16	第 2 个分区表项，共 16 字节。
0x1DE	分区表项 3	16	第 3 个分区表项，共 16 字节。
0x1EE	分区表项 4	16	第 4 个分区表项，共 16 字节。
0x1FE	引导标志	2	有效引导扇区的标志，值分别是 0x55, 0xAA。

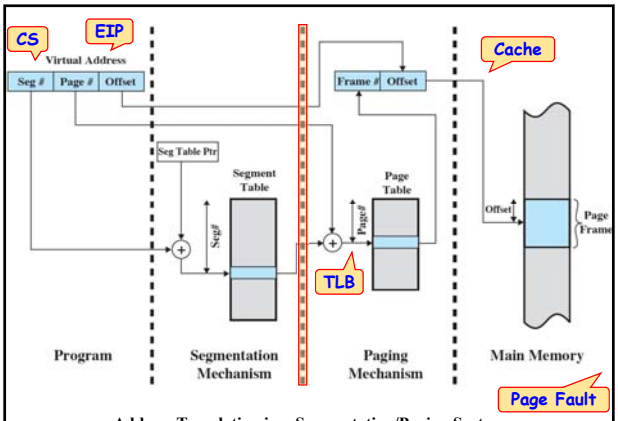
表 9-11 硬盘分区表项结构

位置	名称	大小	说明
0x00	boot_ind	字节	引导标志。4 个分区中同时只能有一个分区是可引导的。0x00-不从该分区引导操作系统；0x80-从该分区引导操作系统。
0x01	head	字节	分区起始处的磁头号。磁头号范围为 0~255。
0x02	sector	字节	分区起始的当前柱面中扇区号(位 0-5) 和柱面号高 2 位(位 6-7)。
0x03	cyl	字节	分区起始处的柱面号低 8 位。柱面号范围为 0~1023。
0x04	sys_ind	字节	分区类型字节。0x0b-DOS; 0x80-Old Minix; 0x83-Linux ...
0x05	end_head	字节	分区结束处的磁头号。磁头号范围为 0~255。
0x06	end_sector	字节	分区结束的当前柱面中扇区号(位 0-5) 和柱面号高 2 位(位 6-7)。
0x07	end_cyl	字节	分区结束的柱面号低 8 位。柱面号范围为 0~1023。
0x08-0x0b	start_sect	长字	分区起始的物理扇区号。它以整个硬盘的扇区号顺序从 0 计起。
0x0c-0x0f	nr_sects	长字	分区占用的扇区数。



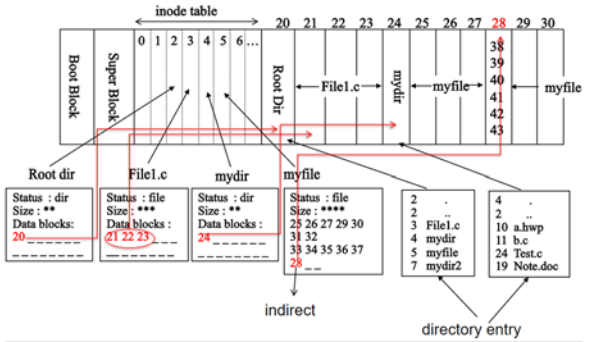
Elements of File Management



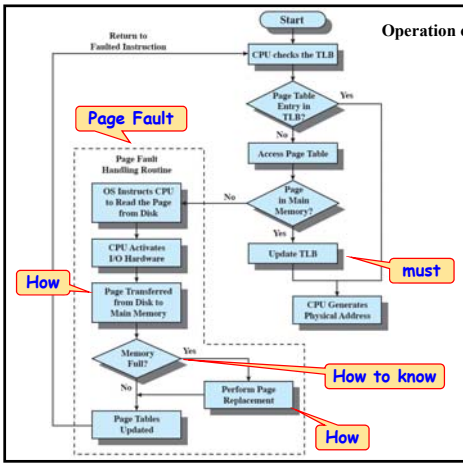


Address Translation in a Segmentation/Paging System

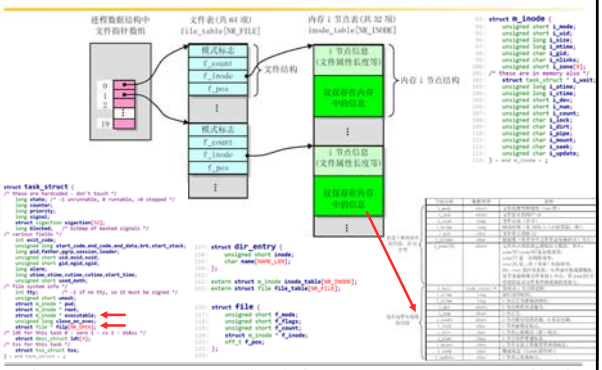
Demo



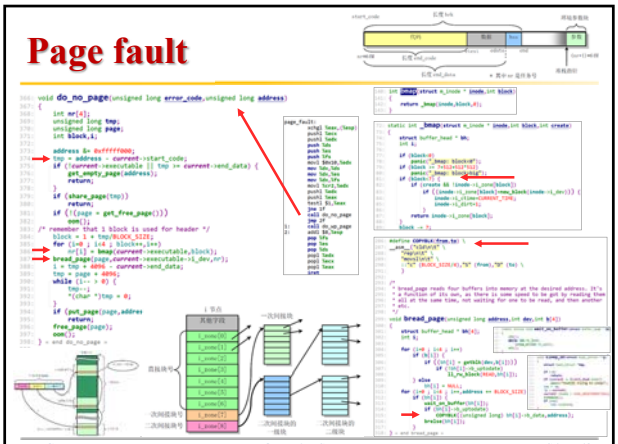
Operation of Paging and TLB



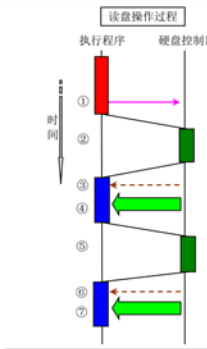
进程打开文件使用的内核数据结构



Page fault

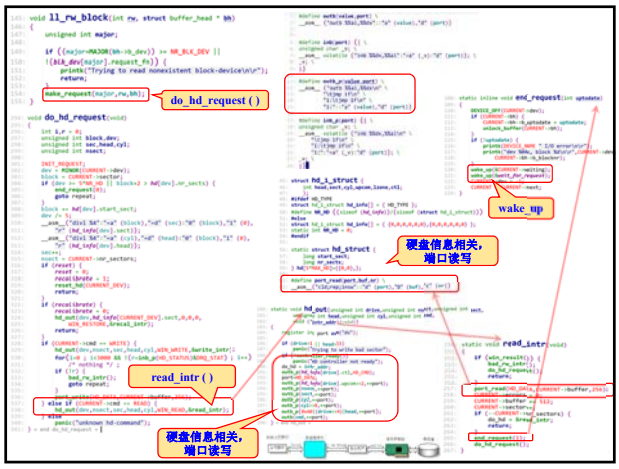


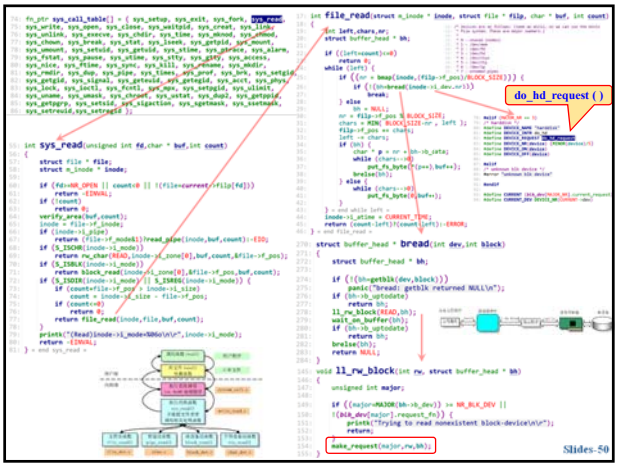
读硬盘数据操作的时序关系

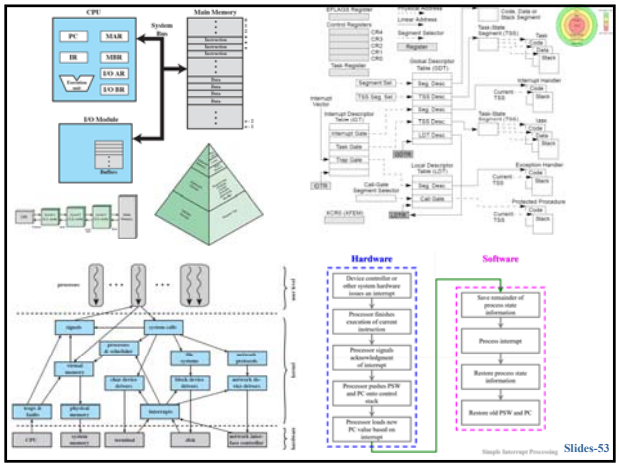


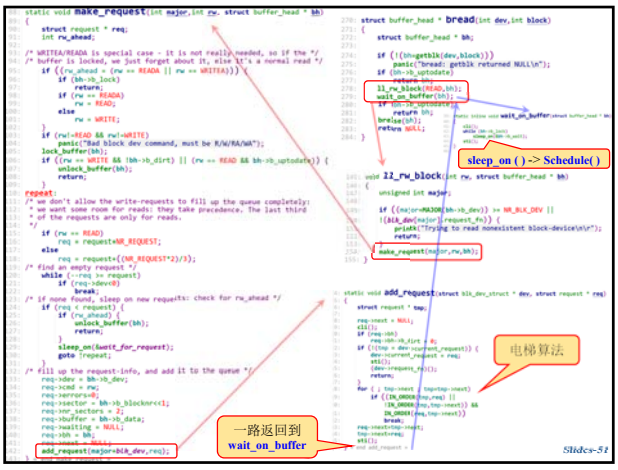
处理请求项 → 写命令
中断处理过程 ←- - 读状态
控制器处理 ← 传读数据

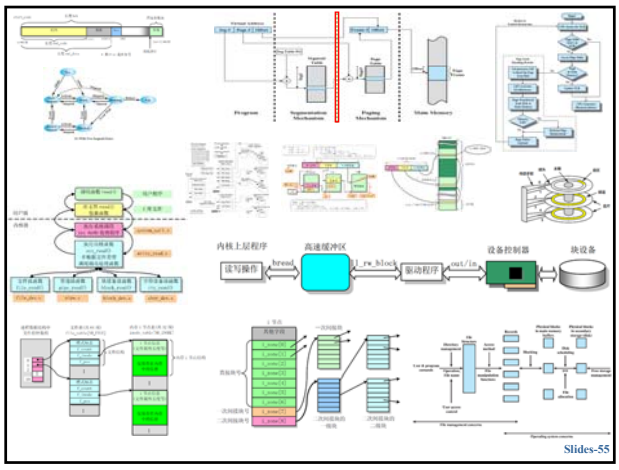
① 函数 do_hd_request()中用 hd_out()发送读命令，然后退出返回调用程序执行，同时等待控制器发中断；
② 控制器接到读命令后就执行从硬盘读数据到控制器缓冲区中的操作。操作完后就发中断；
③ 在中断处理程序中执行 read_intr()。读取控制器状态；
④ 并从控制器缓冲区读取 1 个扇区数据到请求项缓冲块中；
⑤ 控制器随后再执行从硬盘读下一扇区数据到控制器缓冲区中的操作。操作完后就发中断；
⑥ 在中断处理程序中执行 read_intr()。读取控制器状态；
⑦ 并从控制器缓冲区读取 1 扇区数据到请求项缓冲块中，若数据已全部读出则处理本请求项结束事宜，并执行其他硬盘请求操作。











漫谈 OS

- **虚拟**
  - 虚拟内存、虚拟文件系统
- **并行**
  - 流水、进程、线程
- **硬件加速**
  - Cache，TLB，缓冲，磁盘缓冲
- **抽象**
  - 层次模型
- **主奴机制 -- operating**
  - 软硬结合
  - 模式
- **并发、共享**
  - 加锁，信号量
- **中断**
- **栈**
  - 各种巧妙用法
- **内存映射**
  - Map、变换
- **调度**
  - Trade-off
- **视角**
  - Hardware、OS、User