Functions of operating system

- File management
- Device management
- Memory management
- Process management

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Functions of operating system

- 文件管理
  - File management system: FAT32 NTFS EXT3 EXT4 ZFS Btrfs
  - 控制文件的访问
  - 管理文件的创建、删除和修改
  - 给文件命名
  - 管理文件的存储

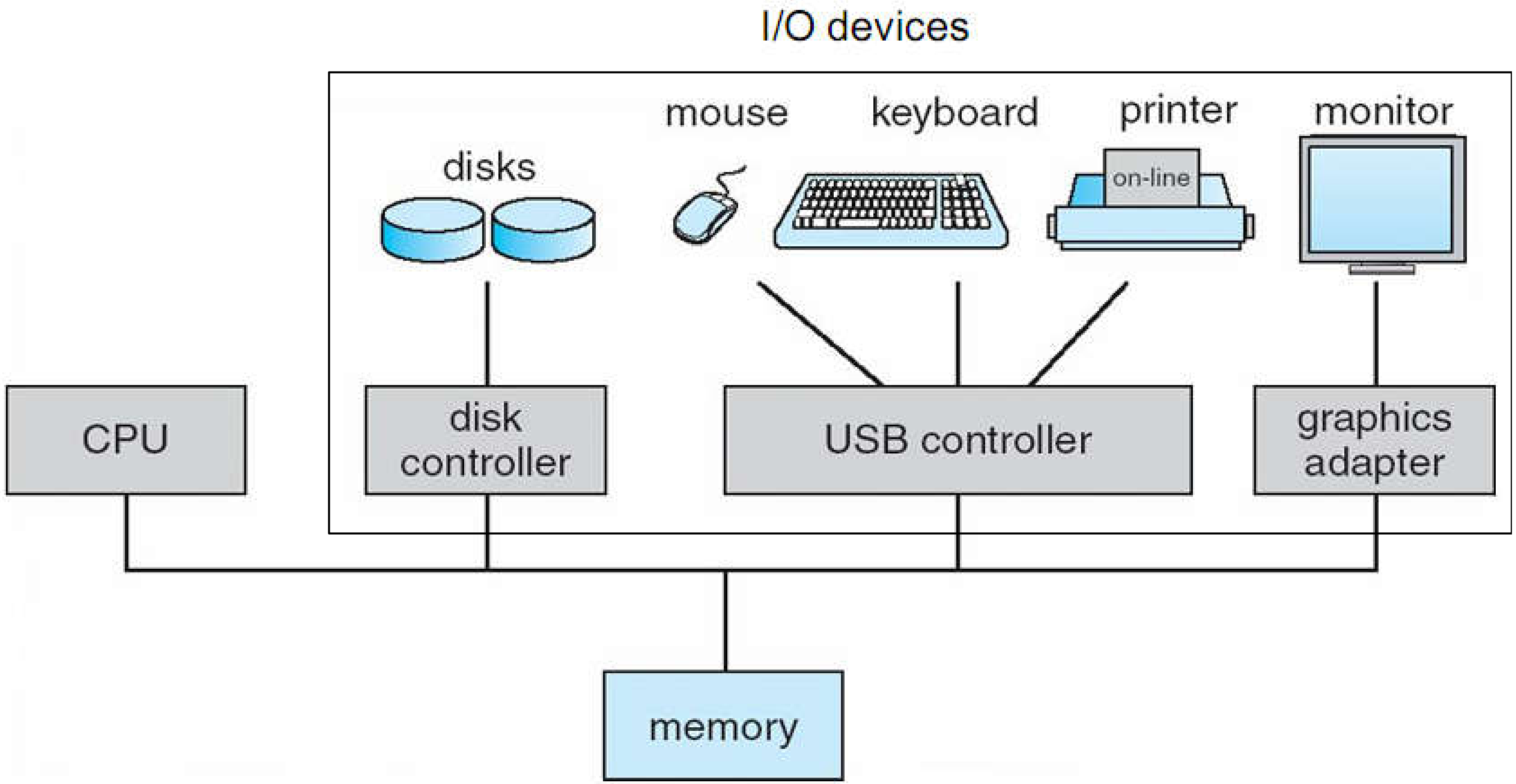
Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Functions of operating system

- 设备管理
  - 访问输入输出设备
  - 输入输出设备存在着数量和速度限制
  - 提高设备使用效率

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

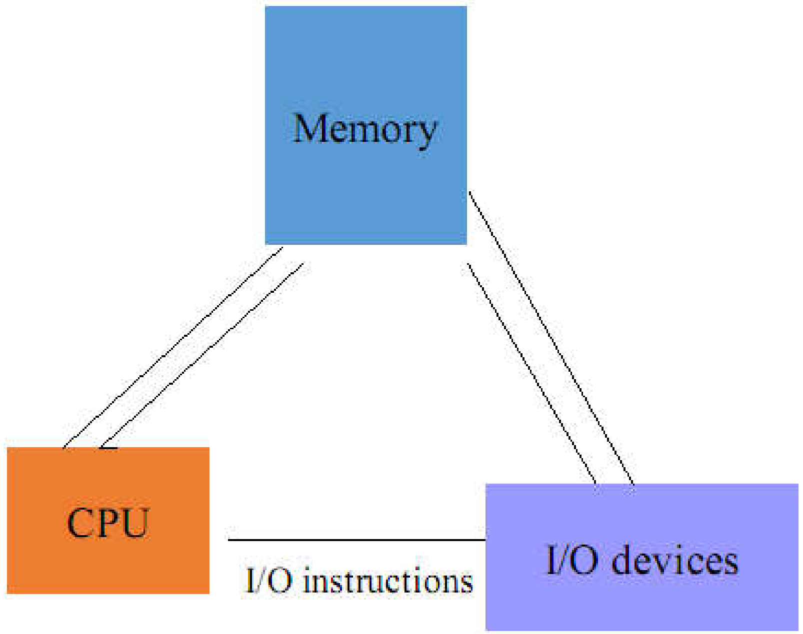
Computer System Organization



Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Speeding up I/O: Direct Memory Access (DMA)

- Data moved directly between I/O devices and memory
- CPU can work on other tasks



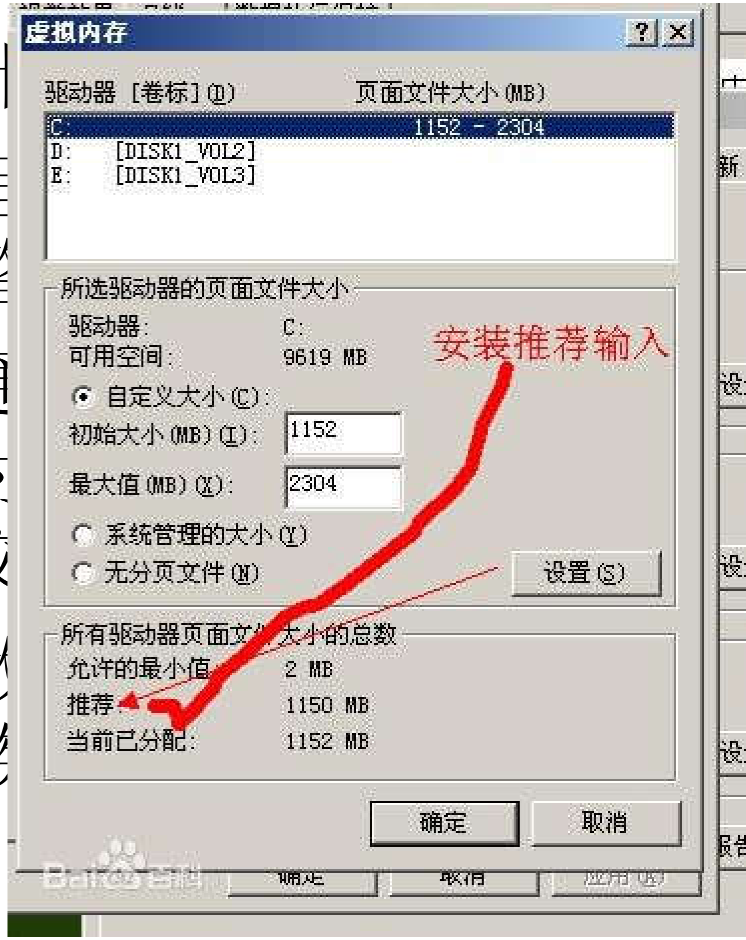
Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Functions of operating system

- 内存管理
  - 单道程序
    - 程序需全载入内存
    - 一个程序运行时，其他程序不能运行
  - 多道程序
    - 同一时刻可载入多个程序
    - 模式：非交换、交换
    - 非交换：程序在运行期间始终驻留在内存
    - 交换：运行过程中，程序可以在内存和硬盘间多次交换数据
  - 虚拟内存

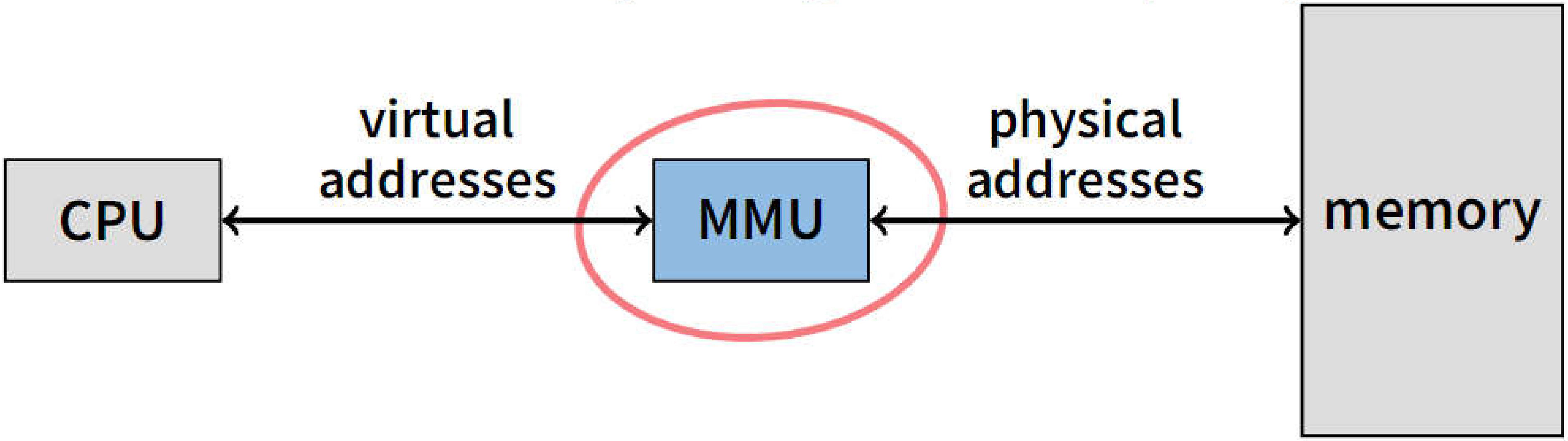
Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

- 虚拟内存是计 一种技术
- 它使得应用程 可用的内存（ 一个连续完整
- 实际上，它通 理内存碎片， 还有部分暂时 器上，在需要 时进行数据交
- 目前，大多数 以内存，如 Windows家族 ux的“交换空 间”等

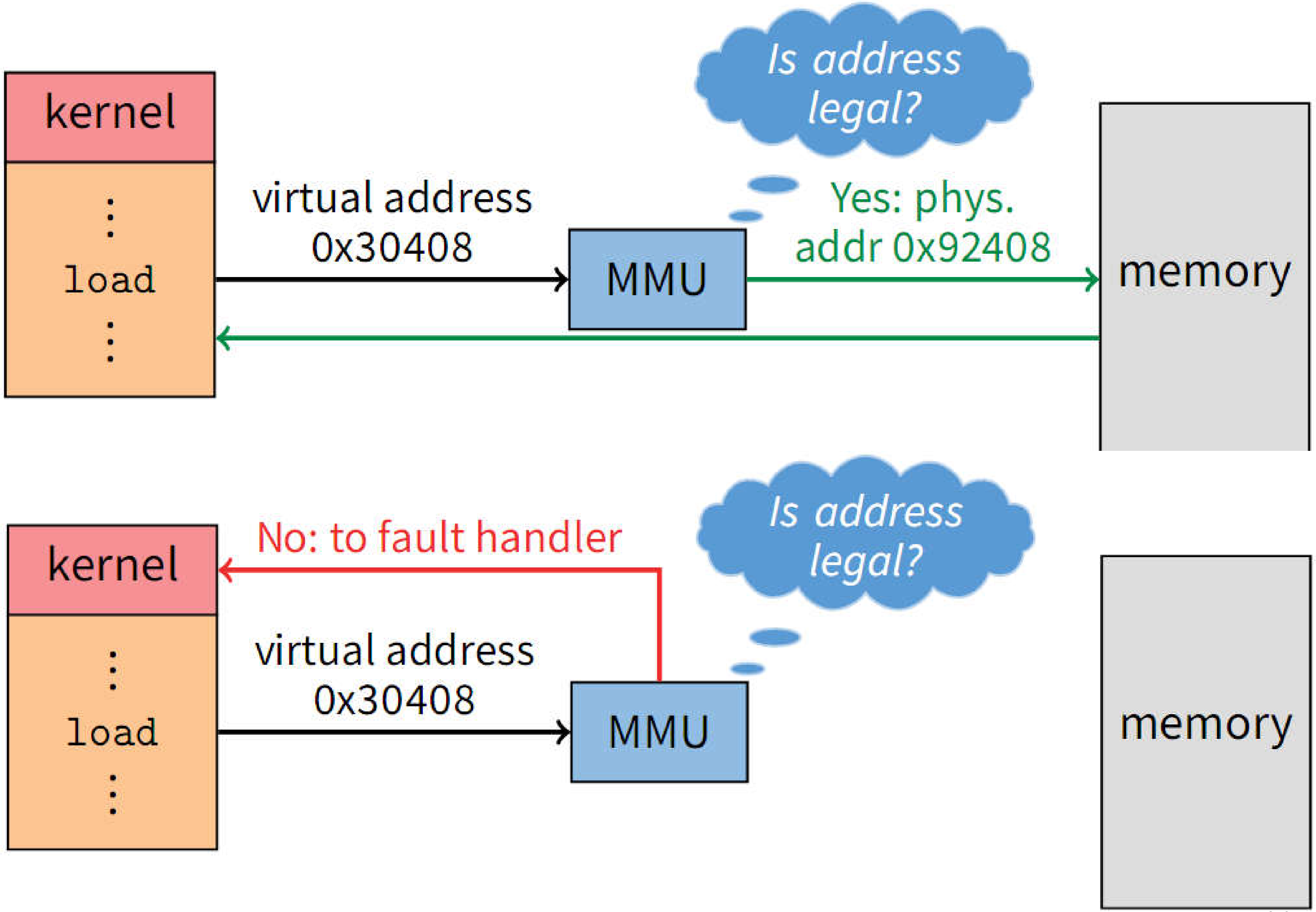


Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

- Programs load/store to virtual addresses
- Actual memory uses physical addresses
- VM Hardware is Memory Management Unit (MMU)

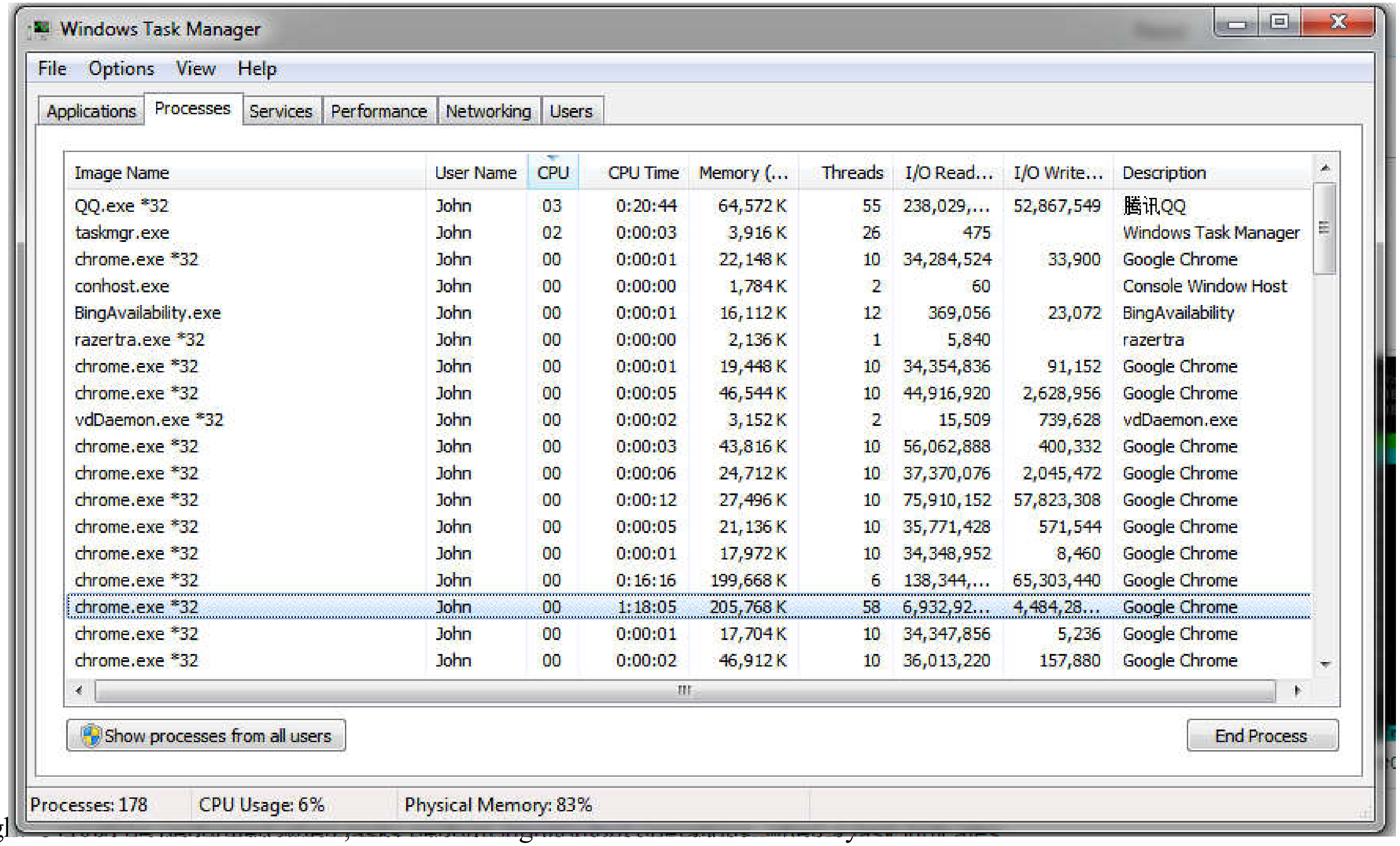


Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley


kernel
load
virtual address
0x30408
MMU
Is address legal?
Yes: phys. addr 0x92408
memory
kernel
load
No: to fault handler
virtual address
0x30408
MMU
Is address legal?
memory

Process进程

- What is a process



Windows Task Manager

File Options View Help

Applications | Processes | Services | Performance | Networking | Users

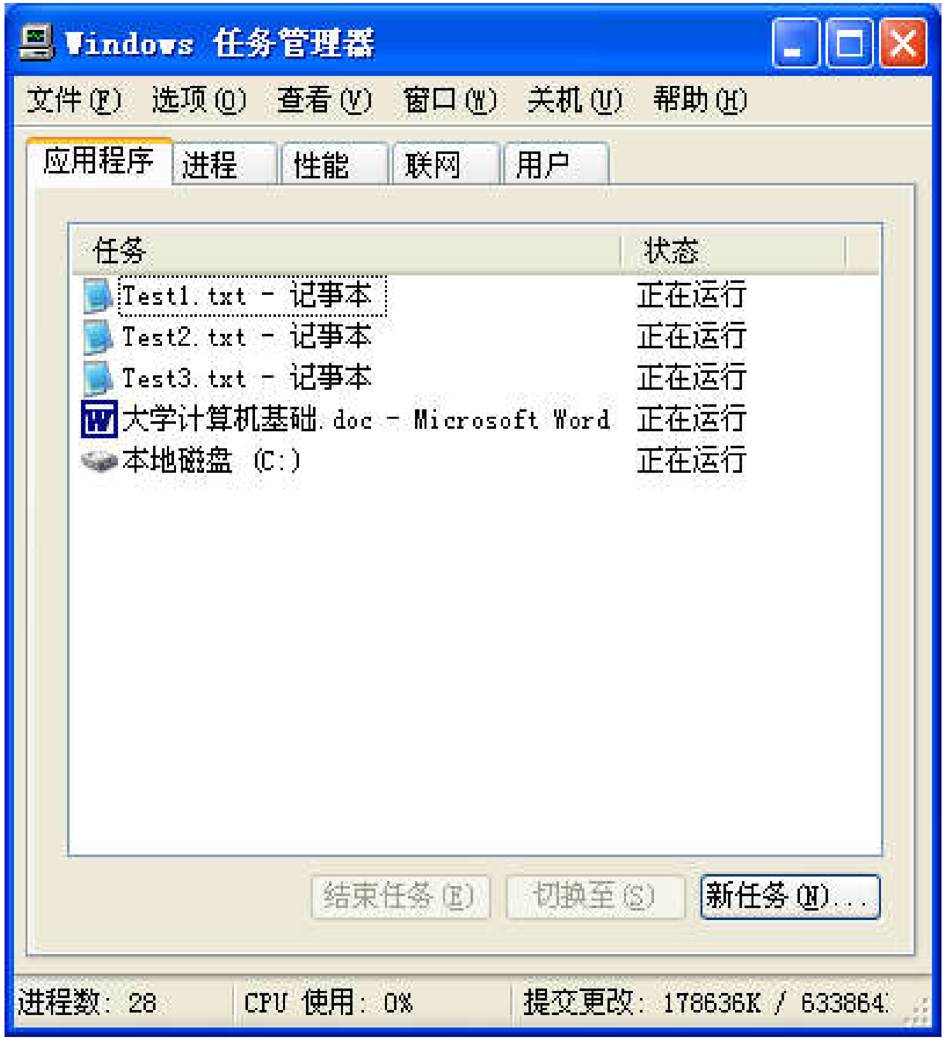
Image Name	User Name	CPU	CPU Time	Memory (...	Threads	I/O Read...	I/O Write...	Description
QQ.exe *32	John	03	0:20:44	64,572 K	55	238,029,...	52,867,549	腾讯QQ
taskmgr.exe	John	02	0:00:03	3,916 K	26	475		Windows Task Manager
chrome.exe *32	John	00	0:00:01	22,148 K	10	34,284,524	33,900	Google Chrome
conhost.exe	John	00	0:00:00	1,784 K	2	60		Console Window Host
BingAvailability.exe	John	00	0:00:01	16,112 K	12	369,056	23,072	BingAvailability
razertra.exe *32	John	00	0:00:00	2,136 K	1	5,840		razertra
chrome.exe *32	John	00	0:00:01	19,448 K	10	34,354,836	91,152	Google Chrome
chrome.exe *32	John	00	0:00:05	46,544 K	10	44,916,920	2,628,956	Google Chrome
vdDaemon.exe *32	John	00	0:00:02	3,152 K	2	15,509	739,628	vdDaemon.exe
chrome.exe *32	John	00	0:00:03	43,816 K	10	56,062,888	400,332	Google Chrome
chrome.exe *32	John	00	0:00:06	24,712 K	10	37,370,076	2,045,472	Google Chrome
chrome.exe *32	John	00	0:00:12	27,496 K	10	75,910,152	57,823,308	Google Chrome
chrome.exe *32	John	00	0:00:05	21,136 K	10	35,771,428	571,544	Google Chrome
chrome.exe *32	John	00	0:00:01	17,972 K	10	34,348,952	8,460	Google Chrome
chrome.exe *32	John	00	0:16:16	199,668 K	6	138,344,...	65,303,440	Google Chrome
chrome.exe *32	John	00	1:18:05	205,768 K	58	6,932,92...	4,484,28...	Google Chrome
chrome.exe *32	John	00	0:00:01	17,704 K	10	34,347,856	5,236	Google Chrome
chrome.exe *32	John	00	0:00:02	46,912 K	10	36,013,220	157,880	Google Chrome

Show processes from all users | End Process

Processes: 178 | CPU Usage: 6% | Physical Memory: 83%

Copyrigl

Processes



Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Process

- Each process represents a task
- Jobs vs Tasks
  - A collection of tasks that is used to perform a computation is known as a *job* (MSDN)
- Is a process the same as a program?

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

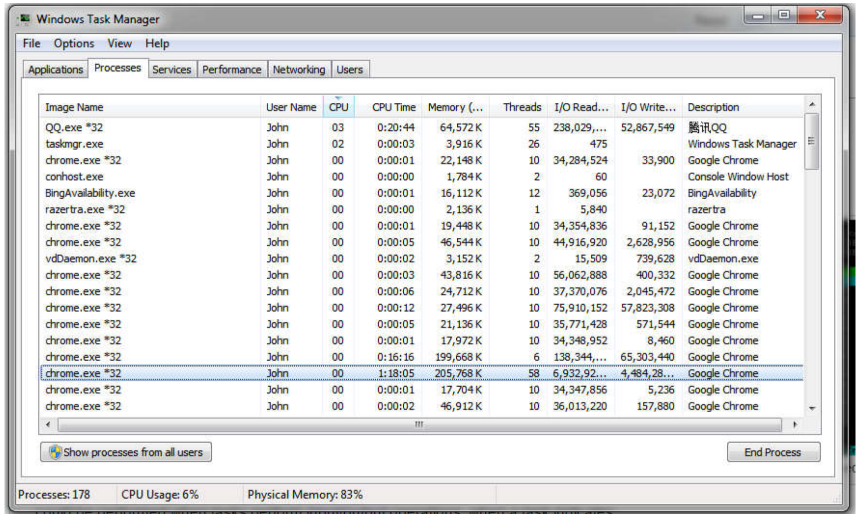
A Program vs. a Process

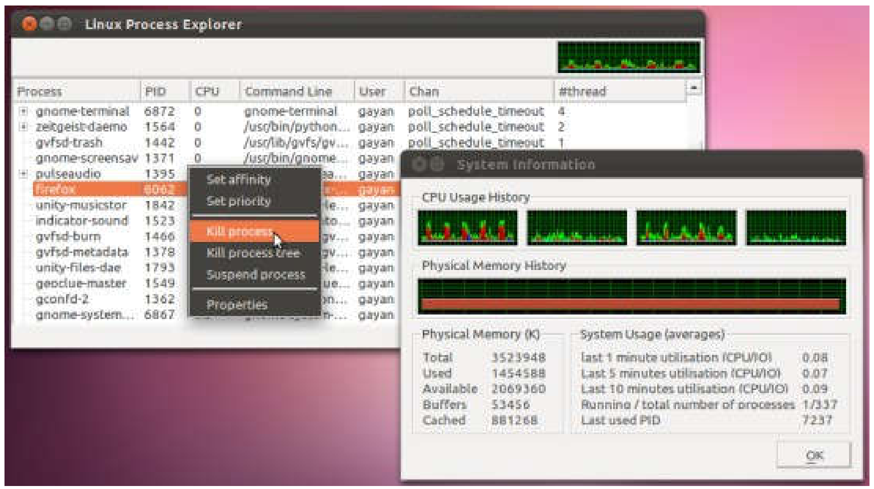
- Program: a set of instructions, e.g., notepad.c, notepad.exe
- Process: activity of executing a program
- A program can be run multiple times, each instance/activity called a process
- **Process State**: Current status of the activity
  - Program counter
  - General purpose registers
  - Related portion of main memory

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Process table

- A table of all the processes maintained by the operating system.
- Each entry is a process and its descriptions (PCB).





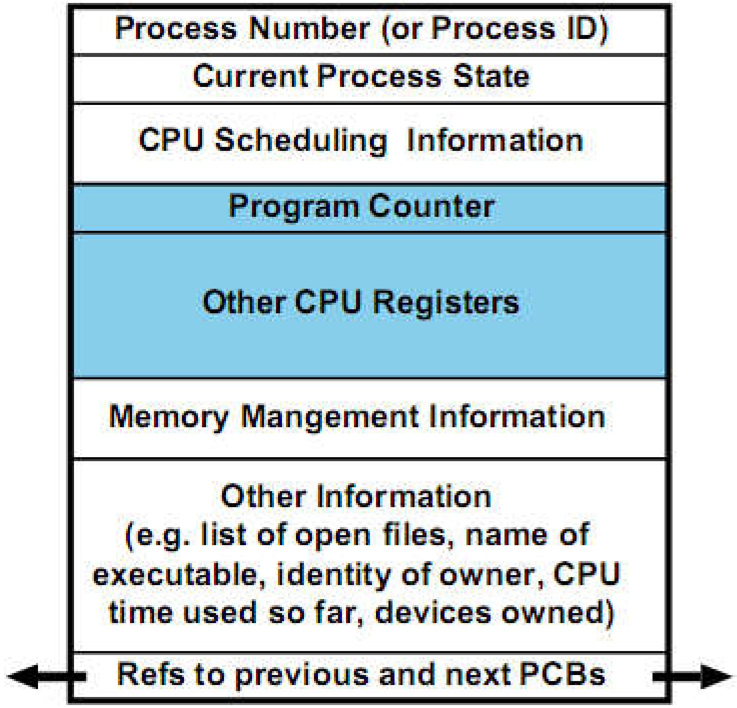
Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Process control block (PCB)

- "the manifestation of a process in an operating system"
  - Process identification
  - Processor state data (the status of a process, saved registers, program counter etc.)
  - Process control data (scheduling state, privileges etc.)

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Process Control Block



Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

The secret of concurrent execution

- What do you have to do when switching from one ongoing task to another?
  - Simply stop the current task and turn to another
  - Record the status of the current task and suspend it and turn to another
- Context switch

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

What is a context?

- Snapshot of current status of a process (PCB)
  - A process identifier, or PID
  - Register values, Program Counter value
  - The memory space, I/O, files for the process
  - Can be saved and resumed as if the process is not interrupted
- Another meaning: execution *state* of the process
  - Ready: ready for execution
  - Waiting: waiting for some I/O
  - Complete: finished process

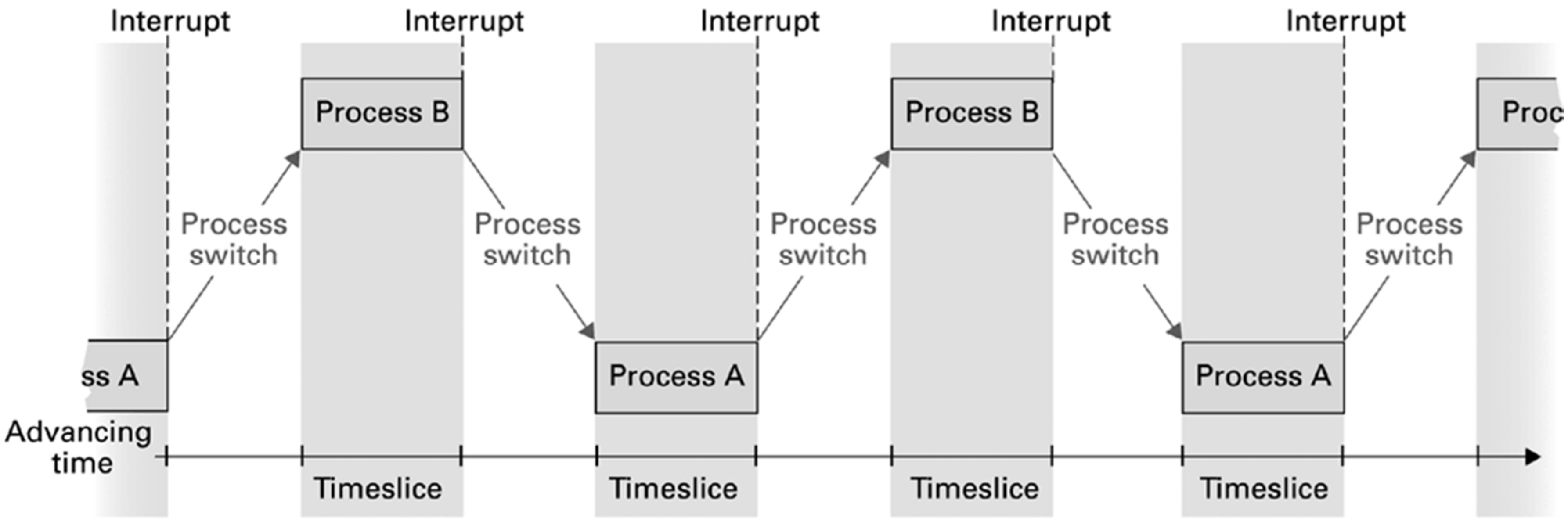
Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Context switch

- The process of storing and restoring the state (context) of a process so that execution can be resumed from the same point at a later time.
- This enables multiple processes to share a single CPU and is an essential feature of a multitasking operating system.

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Figure 3.6 **Time-sharing between process A and process B**



Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Who is responsible for context switching?

- **Process management**
  - **Scheduler (调度):** Adds new processes to the process table and removes completed processes from the process table
  - **Dispatcher (分派):** Controls the allocation of time slices to the processes in the process table

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

When to switch?

- **Interrupt** a signal to the processor emitted by hardware or software indicating an event that needs immediate attention.
- The *interrupt handler* (part of dispatcher) starts after the interrupt to perform context switch
- Modern architectures are interrupt driven.
  – Software interrupt (I/O)
  – Hardware interrupt (press a key)

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Scheduler

- Determines which processes should be considered for execution based on some priorities or concerns
  - Using process table for administration
- Process table
  - Ready or waiting
  - Priority
  - Non-scheduling information: memory pages, etc.

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

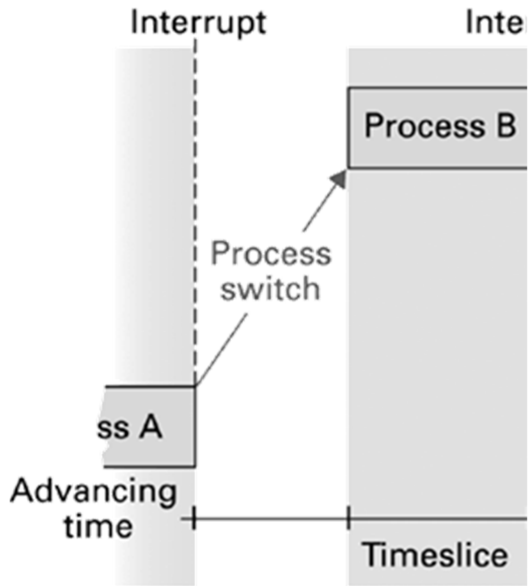
Dispatcher

- Gives time slices to a process that is ready
- Executes a context switch when the running process's time slice is over
  - *Time slice*: a time segment for each execution
  - *Interrupt*: the signal generated by a hardware timer to indicate the end of a time slice.
  - The *interrupt handler* (part of dispatcher) starts after the interrupt to perform context switch

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

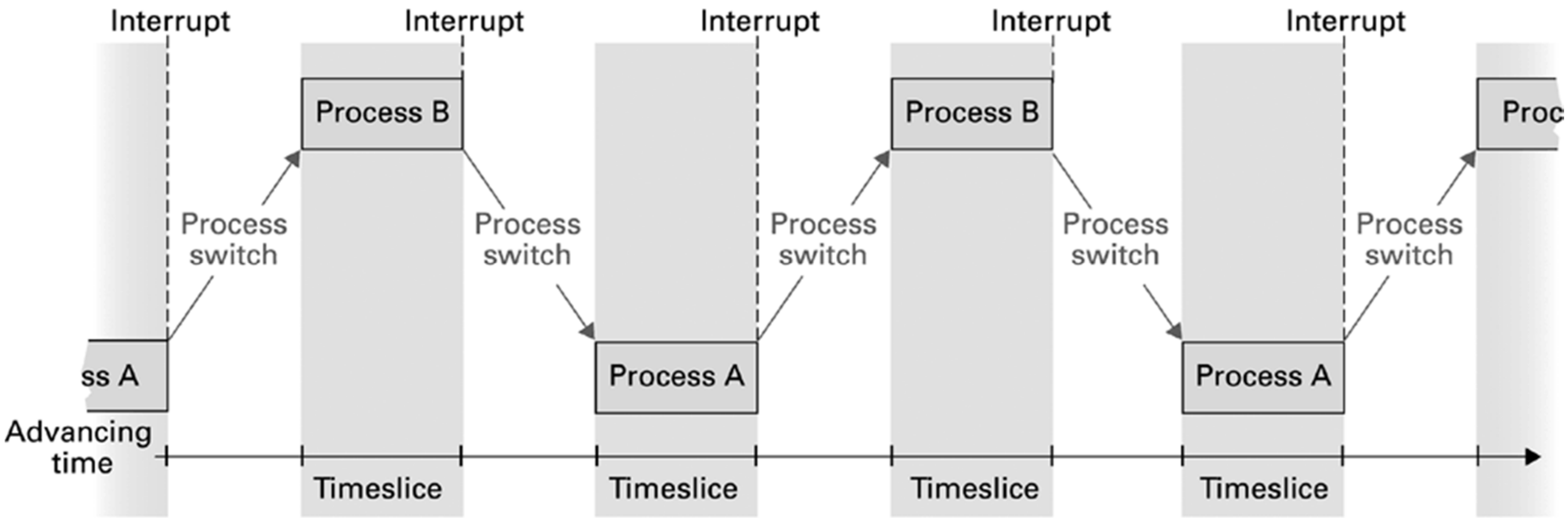
Context Switch (process switch)

1. Get an interrupt from timer
2. Go to the interrupt handler
   a. Save the context of process A
   b. Find a process ready to run (Assume that is process B)
   c. Load the context of process B
3. Start (continue) process B



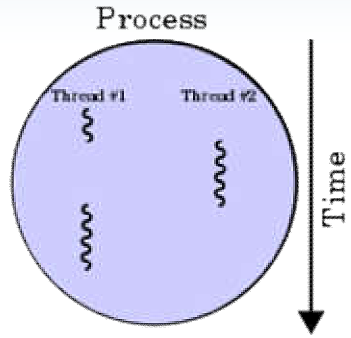
Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Figure 3.6 Time-sharing between process A and process B



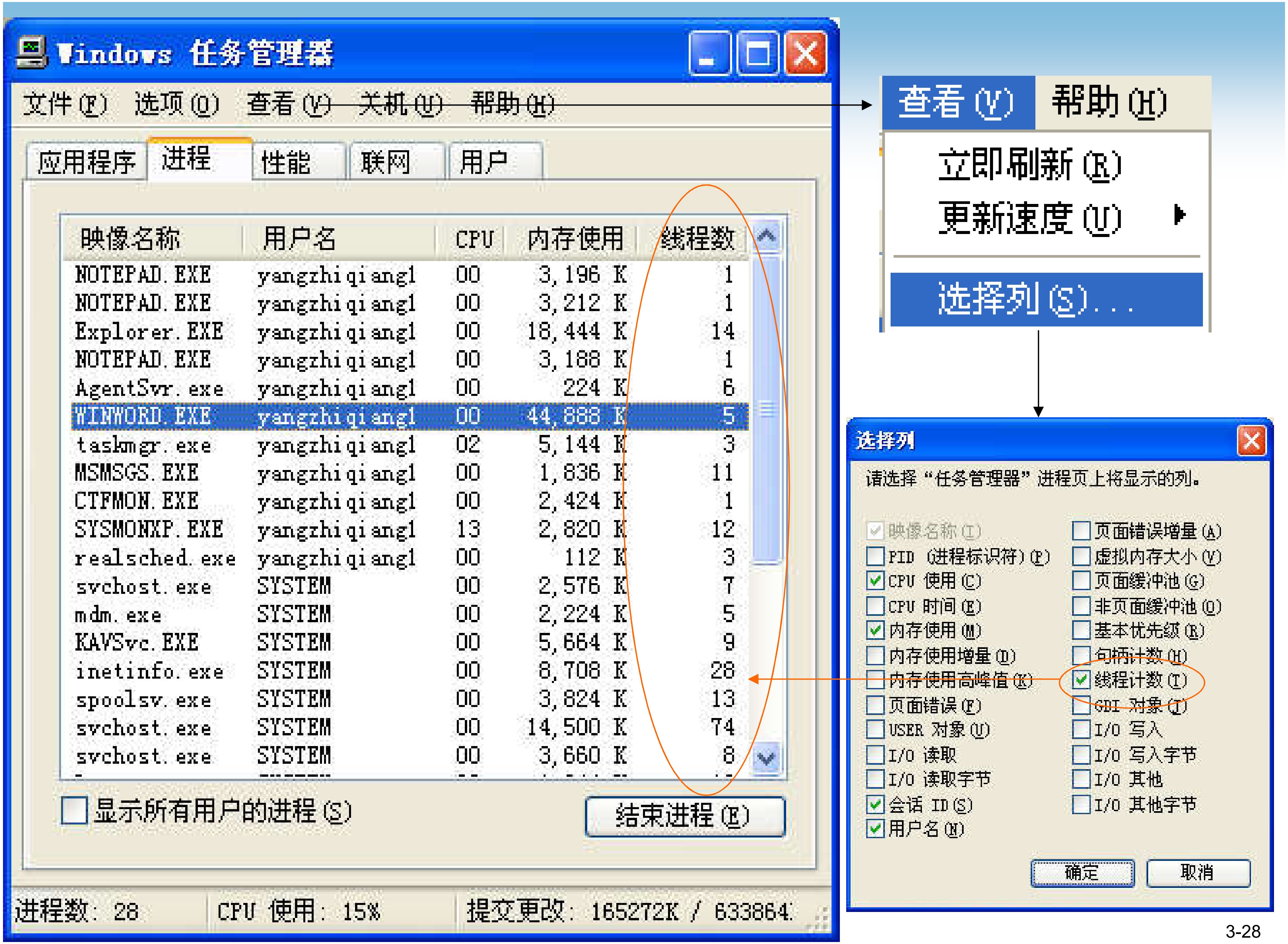
Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Thread



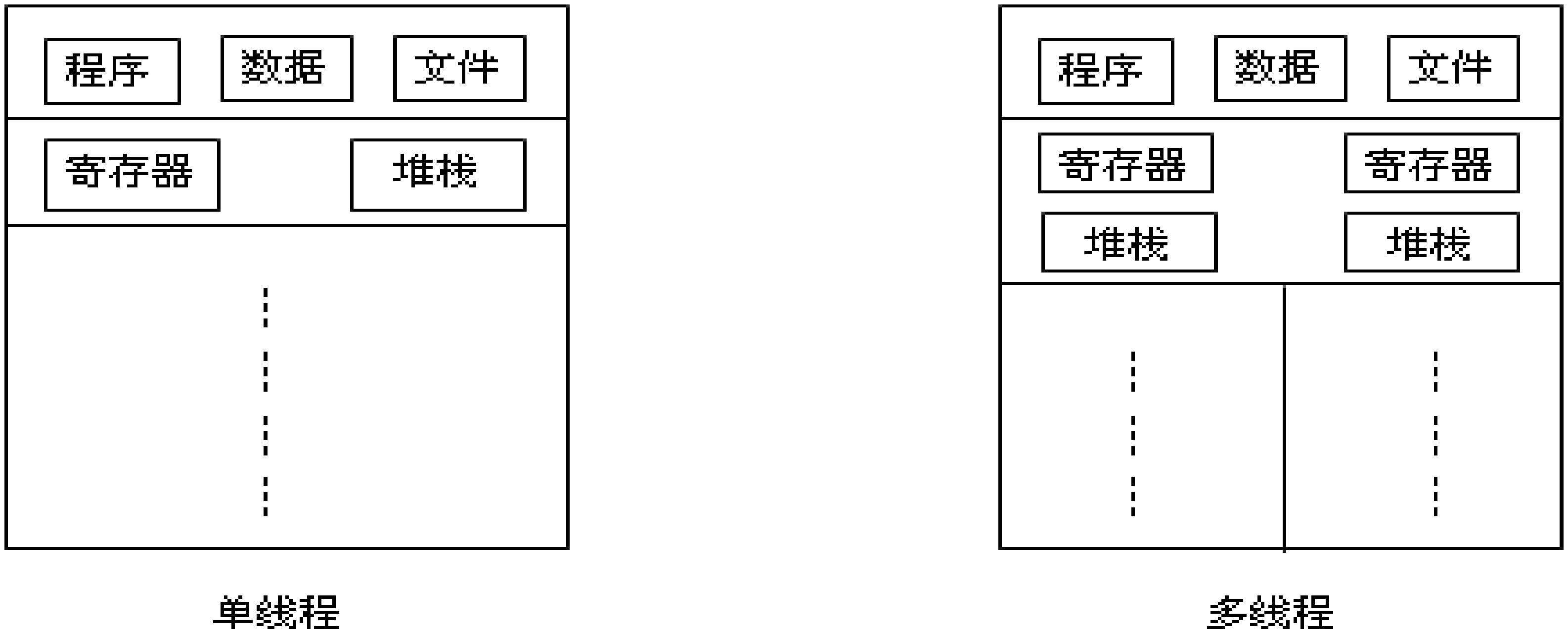
- A task existing within a process that allows multiple independent instances to be executed concurrently
  - Multiple threads share resources such as memory, program code, …
  - Each thread has its own program counter, registers, and stack (local memory)
- The context switch of threads is much faster than that of processes

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



Windows 任务管理器
文件(F) 选项(O) 查看(V) 关机(U) 帮助(H)
应用程序 进程 性能 联网 用户
映像名称 用户名 CPU 内存使用 线程数
NOTEPAD.EXE yangzhiqiang1 00 3,196 K 1
NOTEPAD.EXE yangzhiqiang1 00 3,212 K 1
Explorer.EXE yangzhiqiang1 00 18,444 K 14
NOTEPAD.EXE yangzhiqiang1 00 3,188 K 1
AgentSvr.exe yangzhiqiang1 00 224 K 6
WINWORD.EXE yangzhiqiang1 00 44,888 K 5
taskmgr.exe yangzhiqiang1 02 5,144 K 3
MSMSGS.EXE yangzhiqiang1 00 1,836 K 11
CTFMON.EXE yangzhiqiang1 00 2,424 K 1
SYSMONXP.EXE yangzhiqiang1 13 2,820 K 12
realsched.exe yangzhiqiang1 00 112 K 3
svchost.exe SYSTEM 00 2,576 K 7
mdm.exe SYSTEM 00 2,224 K 5
KAVSvc.EXE SYSTEM 00 5,664 K 9
inetinfo.exe SYSTEM 00 8,708 K 28
spoolsv.exe SYSTEM 00 3,824 K 13
svchost.exe SYSTEM 00 14,500 K 74
svchost.exe SYSTEM 00 3,660 K 8
显示所有用户的进程(S)
结束进程(E)
进程数: 28 CPU 使用: 15% 提交更改: 165272K / 633864
查看(V) 帮助(H)
立即刷新(R)
更新速度(U)
选择列(S)...
选择列
请选择“任务管理器”进程页上将显示的列。
映像名称(I)
PID (进程标识符)(P)
CPU 使用(C)
CPU 时间(E)
内存使用(M)
内存使用增量(D)
内存使用高峰值(K)
页面错误(F)
USER 对象(U)
I/O 读取
I/O 读取字节
会话 ID(S)
用户名(N)
页面错误增量(A)
虚拟内存大小(V)
页面缓冲池(G)
非页面缓冲池(O)
基本优先级(B)
句柄计数(H)
线程计数(T)
GDI 对象(J)
I/O 写入
I/O 写入字节
I/O 其他
I/O 其他字节
确定
取消

A Thread vs. a Process



Share more

High Concurrence

Fast Switch

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Exercises

- Suppose an OS allocates time slices in 10 millisecond units and the time required for a context switch is negligible. How many processes can obtain a time slice in one second?

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Competition for Resources

- What are *resources*?
  - CPU, memory, files, peripheral devices, …
- In a multitasking system, resources are shared by processes
  - Some resources should not be employed by more than one process simultaneously
  - E.g., printer

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Handling Competition for Resources

- **Semaphore:** A “control flag”
- **Critical Region:** A group of instructions that should be executed by only one process at a time
- **Mutual exclusion:** Requirement for proper implementation of a critical region

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

First Algorithm

- Use a flag (a global memory address)
  - flag=1: the critical region is occupied
  - flag=0: no process is in the critical region
- Problem:

Process A

Context switch to A

```
if (flag == 0) {
    flag = 1;
    /*critical region*/
}
```

Process B

Context switch to B

```
if (flag == 0) {
    flag = 1;
    /*critical region*/
}
```

  - Both processes get into the critical region

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Solutions

- Testing&setting the flag must be completed without interruption

1. Use disable_Interrupt() to prevent context switch during the flag test and set process.

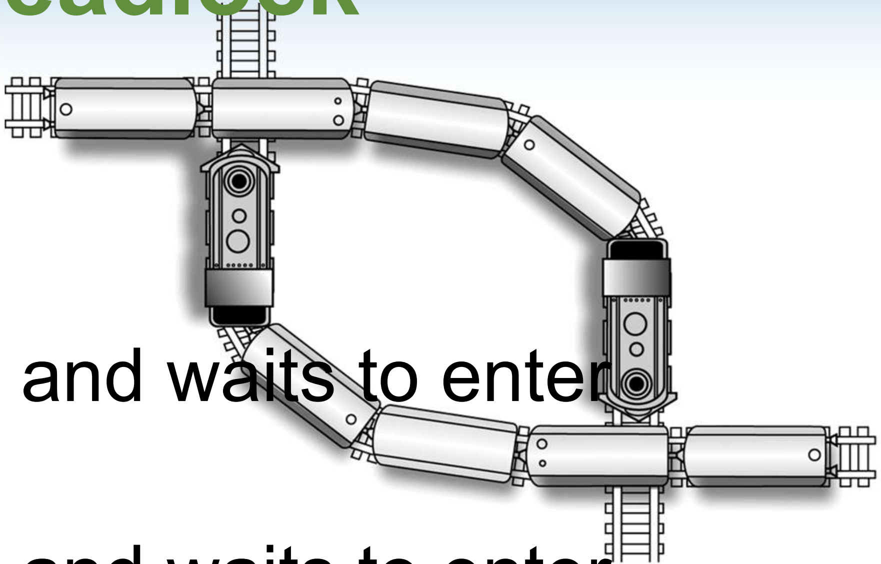
```
Diable_Interrupt();
if (flag == 0) {
    flag = 1;
    Enable_Interrupt();
    / *critical region*/
}
Enable_Interrupt();
```

2. A machine instruction called “test-and-set” which cannot be interrupted

- Semaphore: a properly implemented flag

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Another Problem: Deadlock



- Example:
  - A is in critical region 1, and waits to enter critical region 2
  - B is in critical region 2, and waits to enter critical region 1

Process A

```
if (test_set(flag1)) {
    /*critical region 1*/
   while(!test_set(flag2));
   /*critical region 2*/
  }
```

Context switch to A

Process B

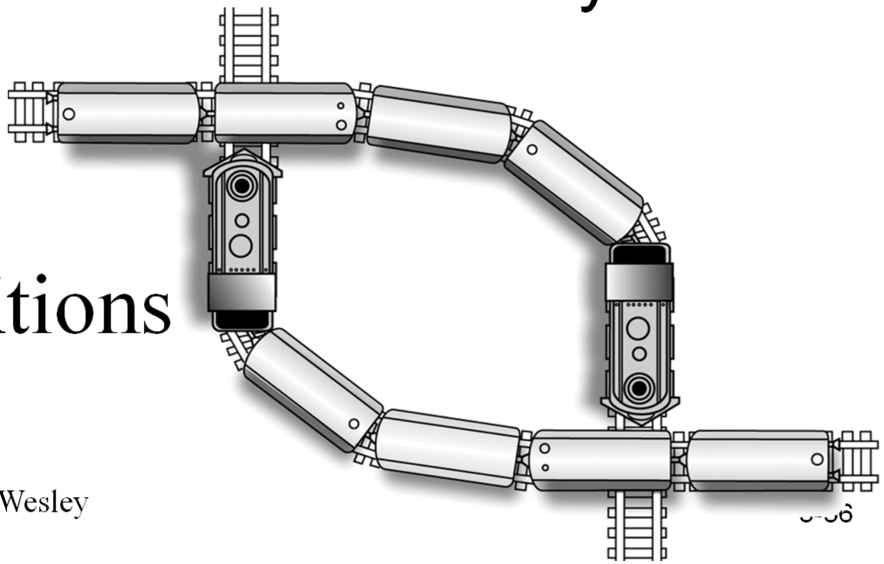
```
if (test_set(flag2)) {
    /*critical region 2*/
   while (!test_set(flag1));
     /*critical region 1*/
  }
```

Context switch to B

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Deadlock

- Processes block each other from continuing
- Conditions required for deadlock
  1. Competition for non-sharable resources
  2. Resources requested on a partial basis
  3. An allocated resource can not be forcibly retrieved



Remove any one of the conditions can resolve the deadlock.

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Solutions

Which condition is removed?

1. Kill one of the process
2. Processes need to request all the required resources at one time
3. Spooling
   - For example, stores the data to be printed and waits the printer available
4. Divide a file into pieces so that it can be altered by different processes

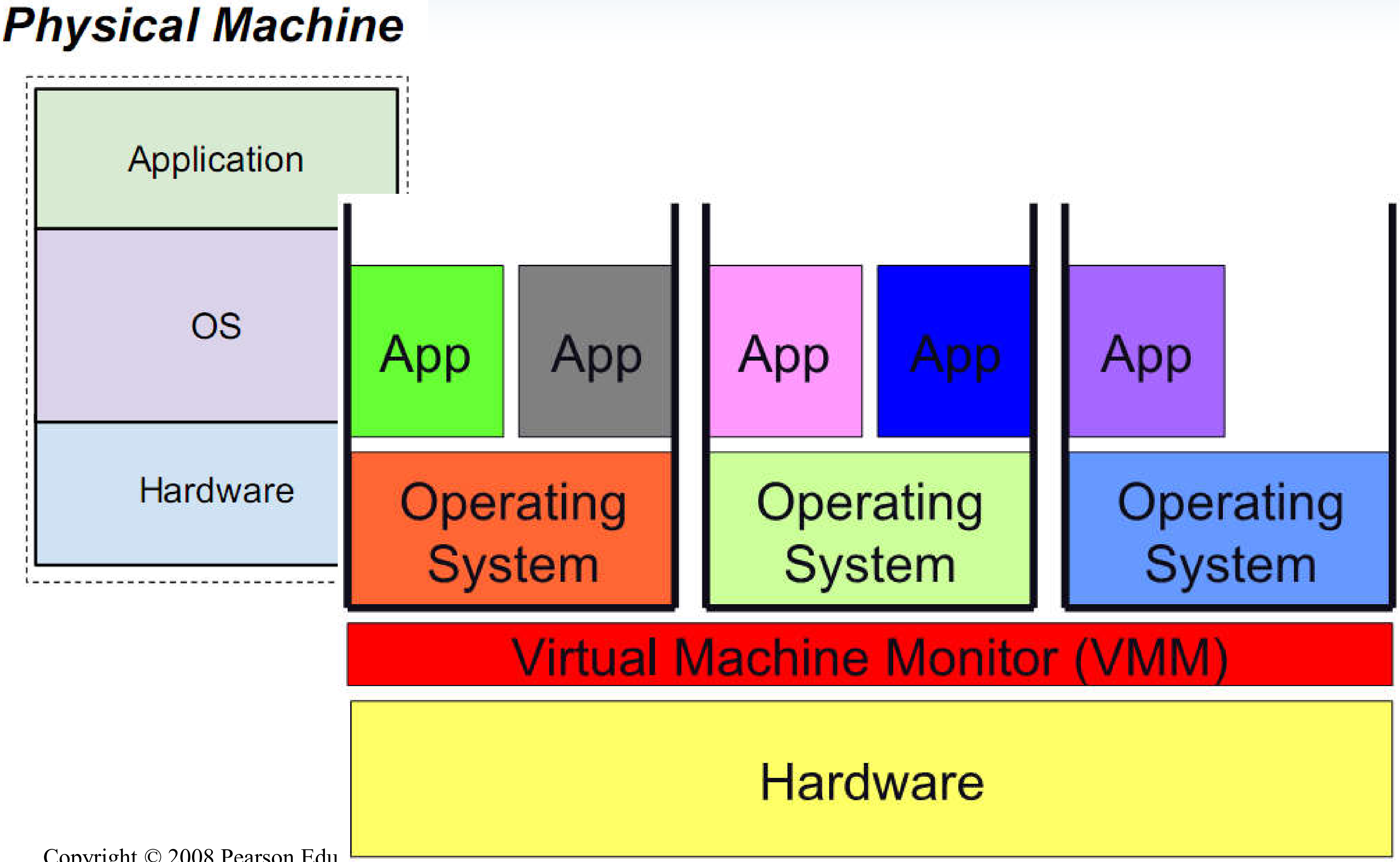
Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Exercises

- There is a bridge that only allows one car to pass. When two cars meet in the middle, it causes “deadlock”. The following solutions remove which conditions
  - Do not let a car onto the bridge until the bridge is empty.
  - If cars meet, make one of them back up.
  - Add a second lane to the bridge.
- What’s the drawback of solution 1?

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Virtual machine虚拟机



Copyright © 2008 Pearson Edu