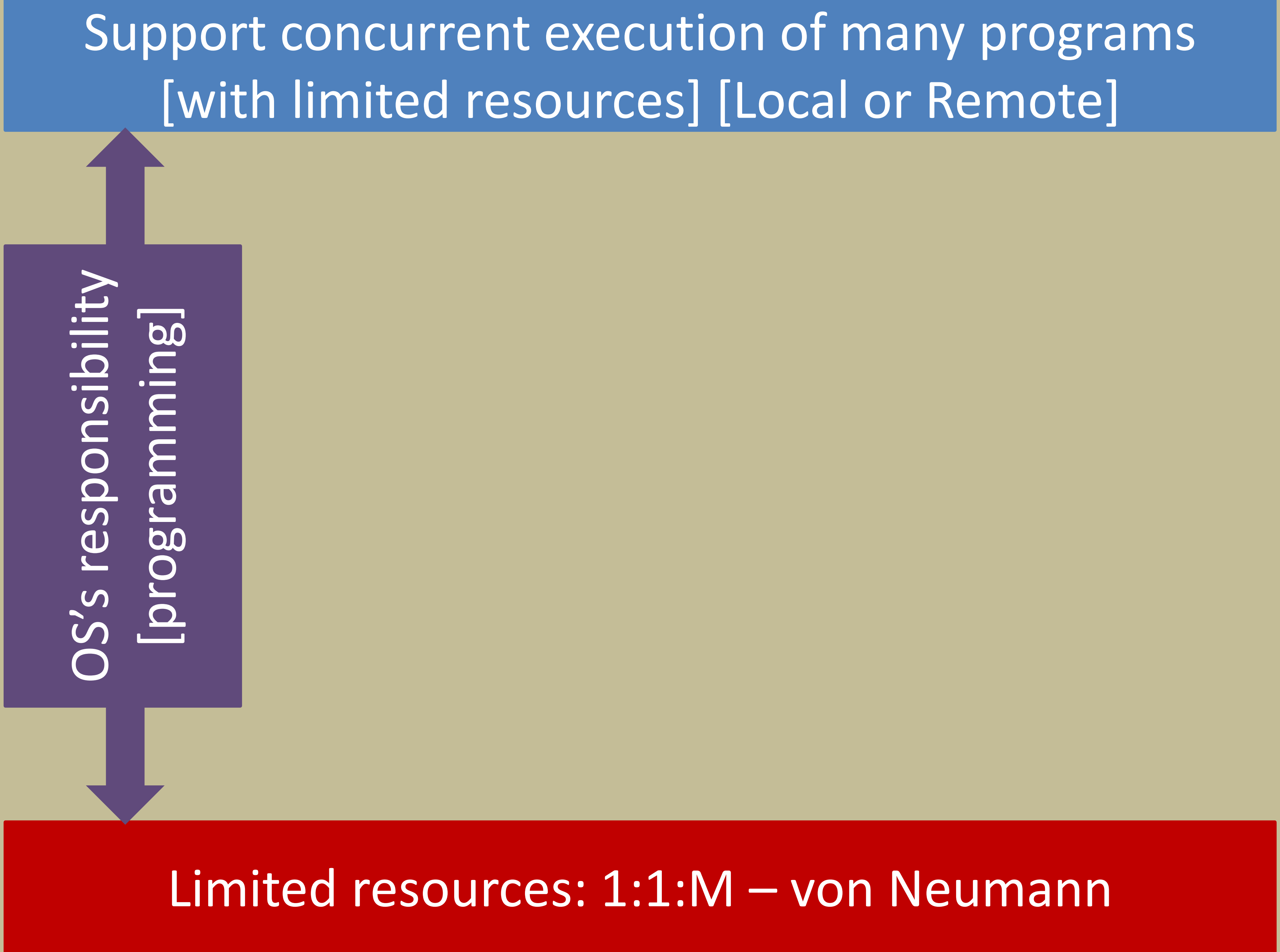
Operating system

Review: OS’s big picture now

Review – Concepts

- Problems and Solution ideas to understand Modern OS
- Load OS
- Details
  - Execution of a program
    - CPU, File → Mapping 1 + 2
  - Overcome concurrency
    - Synchronization, deadlock



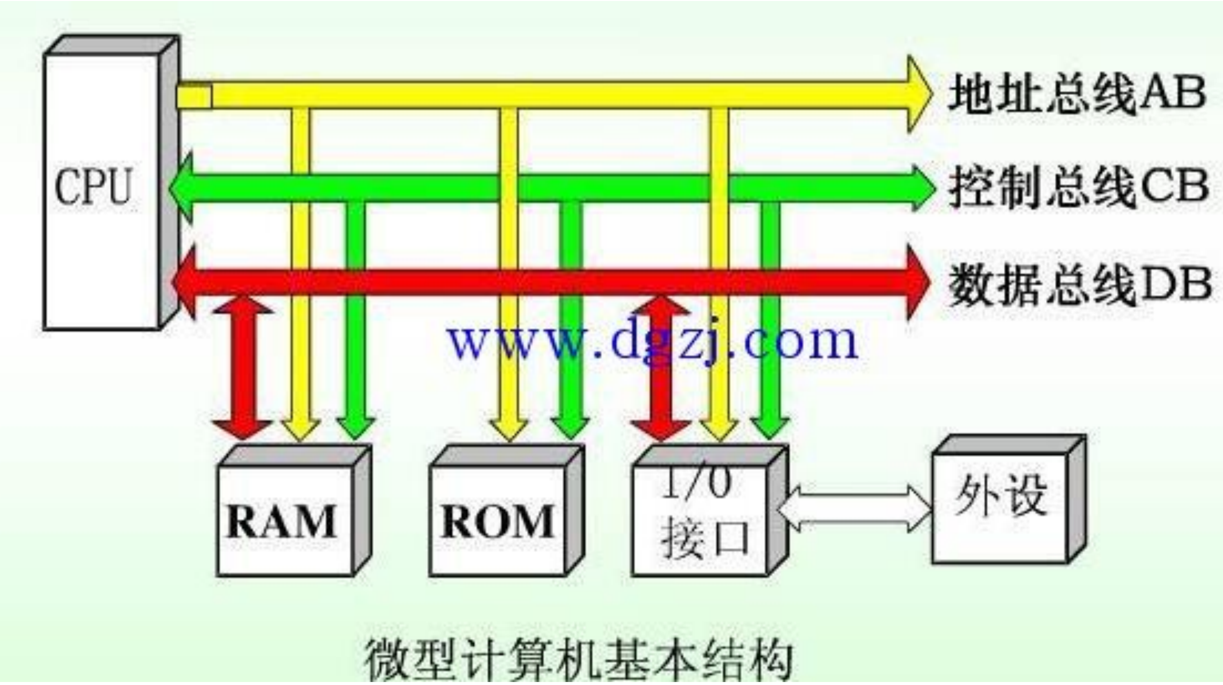
Support concurrent execution of many programs [with limited resources] [Local or Remote]

OS’s responsibility [programming]

Limited resources: 1:1:M – von Neumann

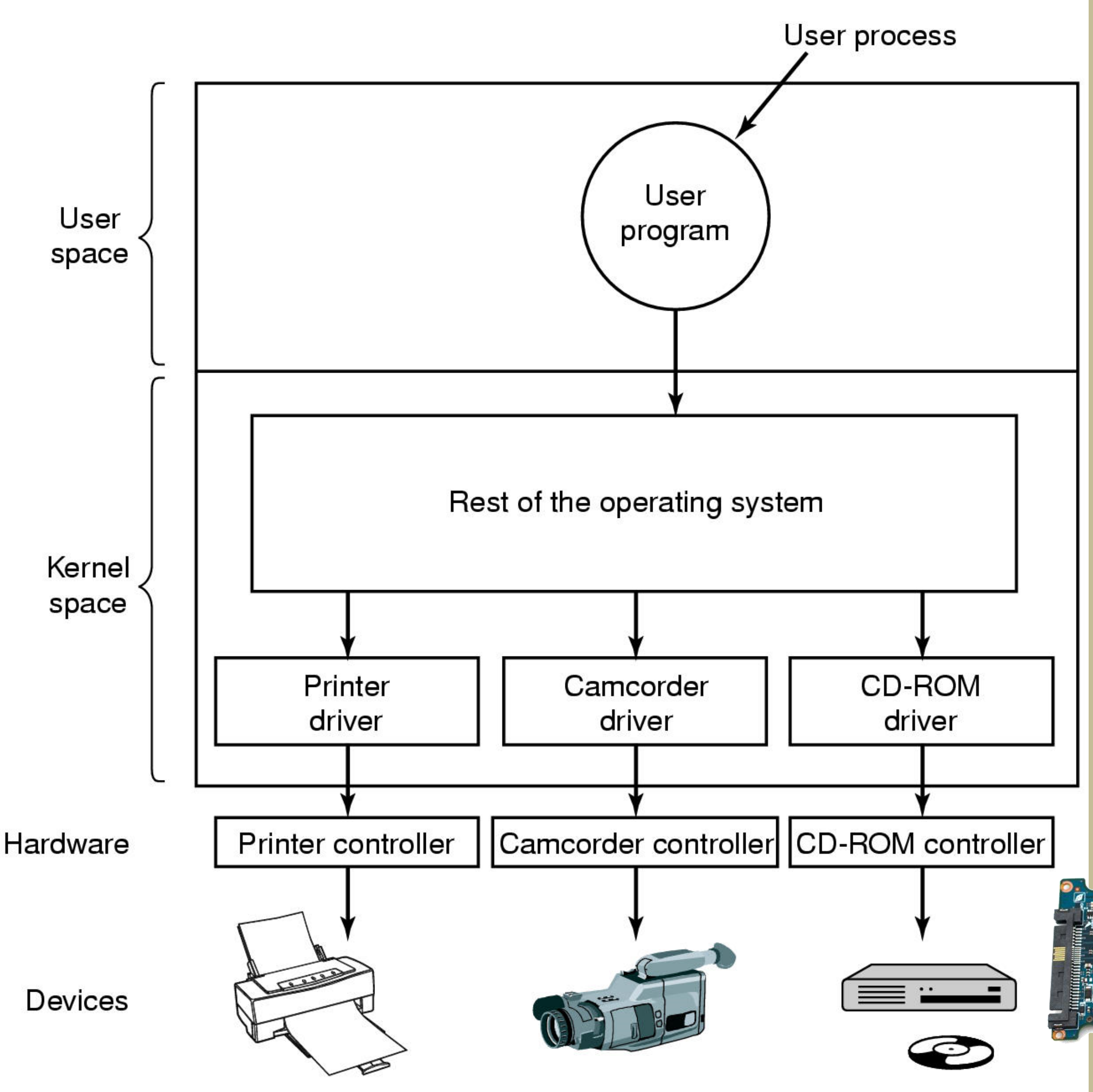
Limited resources – 1 CPU, 2 Spaces

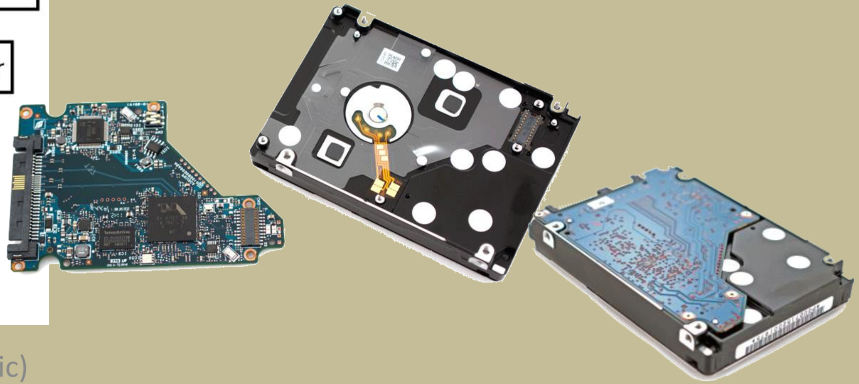
- Namely, we have 1 CPU, limited linear addressed MM space (including ROM – Read Only Memory), limited linear addressed sector space (after **LLF** – Low Level Formatting by HDD manufacturers), keyboard, display, CDROM …
  - Connected through buses



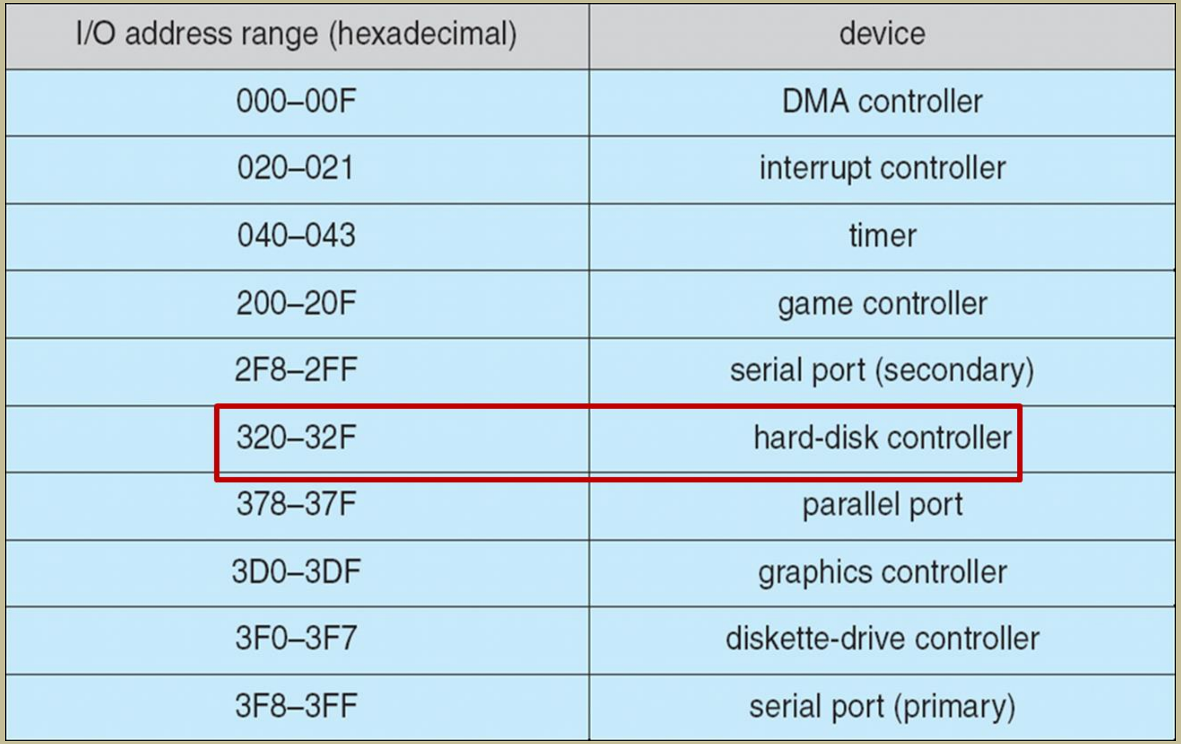
- Some programs
  - POST (Power On Self Testing), BIOS (Basic Input/Output System), Bootloader
  - By using instructions defined in **ISA** (Instruction-Set Architecture)

They share similar connection architecture





- The device communicates with the computer via a communication point called a **port**. [端口号]
- Exchange data with CPU via registers
  - By writing into these registers
    - OS can command the device to deliver or accept data, to switch the device on or off
  - By reading from the registers
    - OS can learn the status of the device



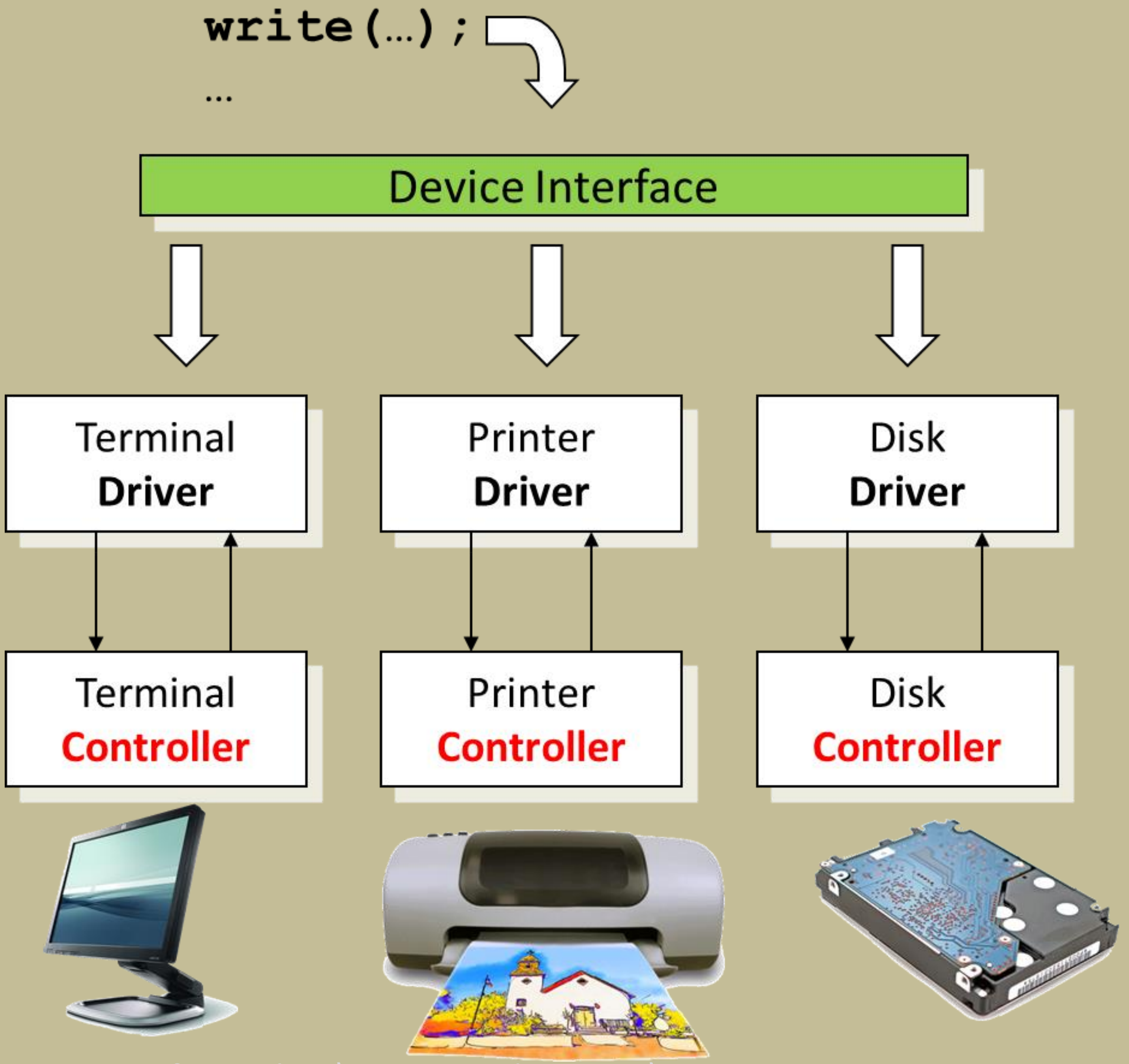
I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

PPTs from others\epcc.sjtu.edu.cn_~os\OS16.ppt

We also need device Drivers [驱动程序]

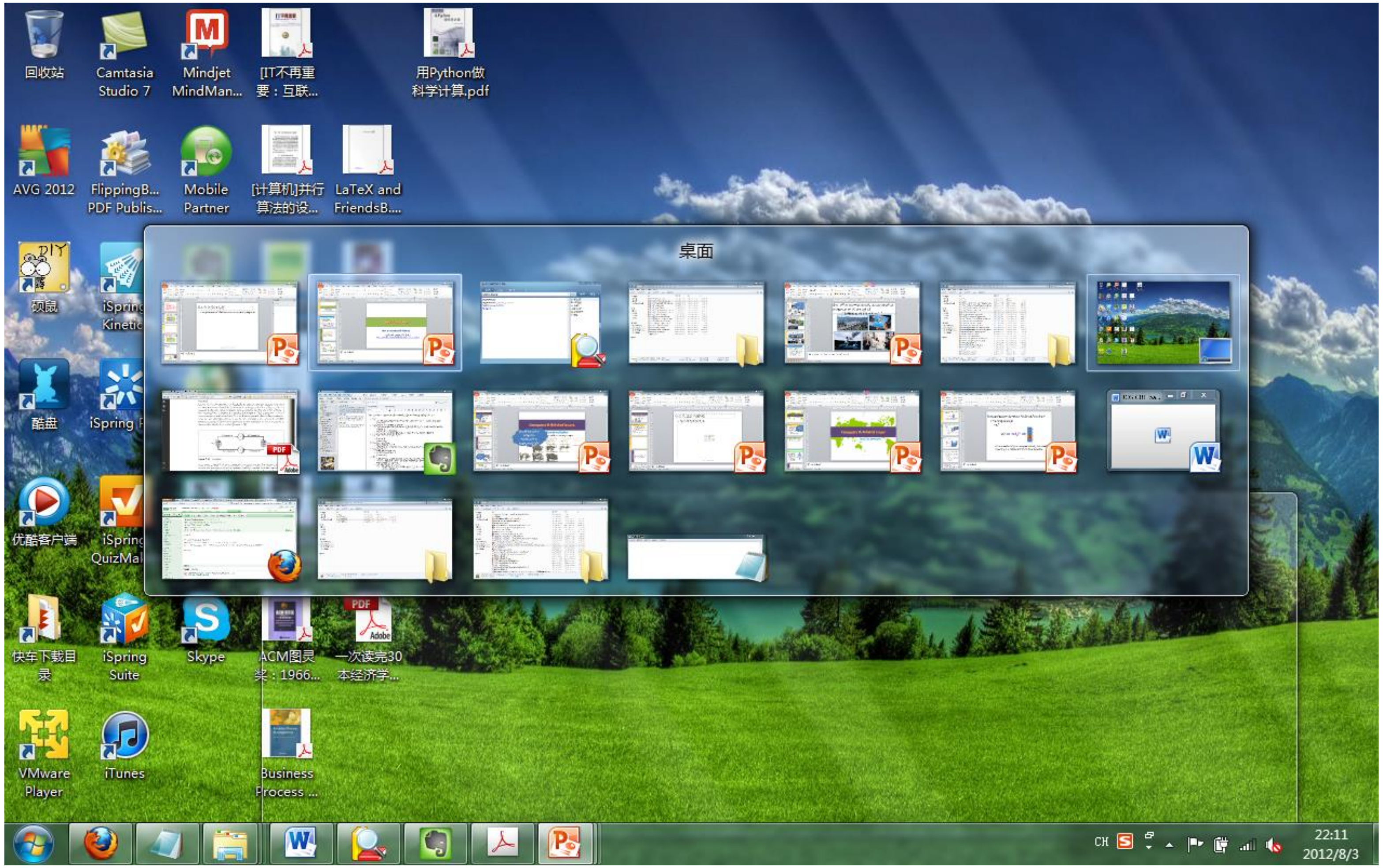
- The software that talks to the device controllers
  - Device specific
  - Tailored to individual device characteristics
  - Written by device manufacturers
  - Part of the OS Kernel

- Know about the details of the devices
  - Disk driver knows about sectors, tracks, cylinders, heads, arm motions, motor drives
  - Mouse driver knows about button pressed



PPTs from others\epcc.sjtu.edu.cn_~os\OS17.ppt

Goal of modern OS – Concurrent execution of programs



- Challenges are
  - Multiplex CPU
  - Small MM vs. Large Program
  - Data Inconsistency
  - Deadlock
  - Organize files

- Ideas are
  - CPU switching <> 1 CPU

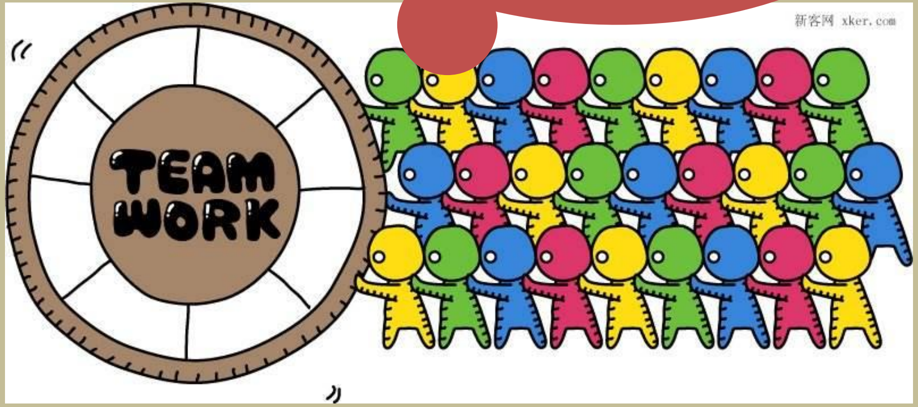
Introduction to Computer Science
Part I Introduction
Some things about this course

- Ideas are
  - Cut program and MM into segments <> Small MM vs. Large Program

Concurrency **OF COURSE** benefits users! (Not free?- only if well controlled)

- Concurrent processes executing in the operating system allows for the processes to cooperate (both mu
with other processes
  - The simplest example of how th
processes are using the sa
- Reasons for cooperating pro
  - Several processes may need to access
stored in a file)
  - Information sharing
  - Computation speed-up
  - Modularity
  - Convenience

This is of course not FREE! The cooperation leads to complexity – deadlock and data inconsistency in later chapters



IPC: Inter-Process Communication

- Cooperating processes require an inter-process communication (IPC) mechanism that will allow them to exchange data and information.
- There are two fundamental models of inter-process communication:
  - **Shared memory**
  - **Message passing**
    - message passing interfaces, mailboxes and message queues
    - sockets, STREAMS, pipes

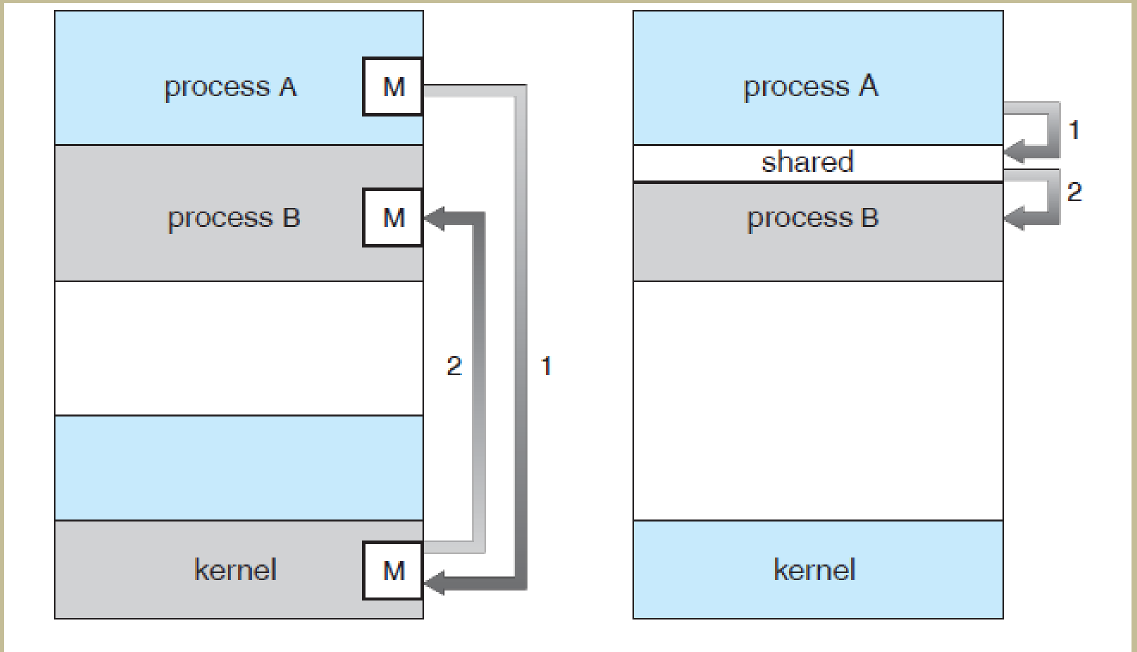
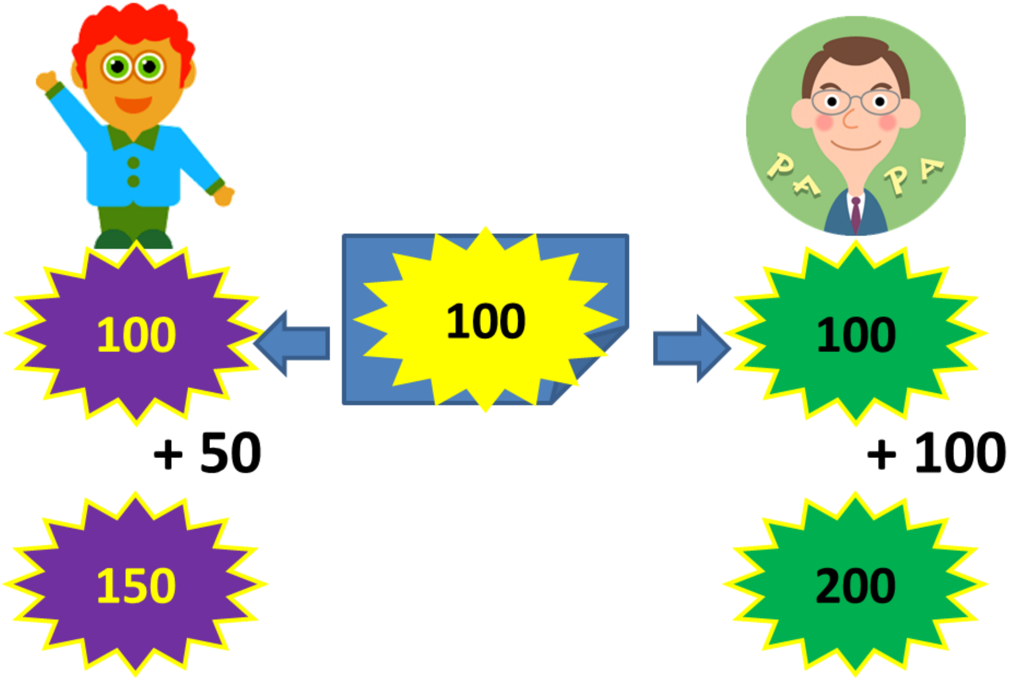


Figure 3.13 Communications models. (a) Message passing. (b) Shared memory.

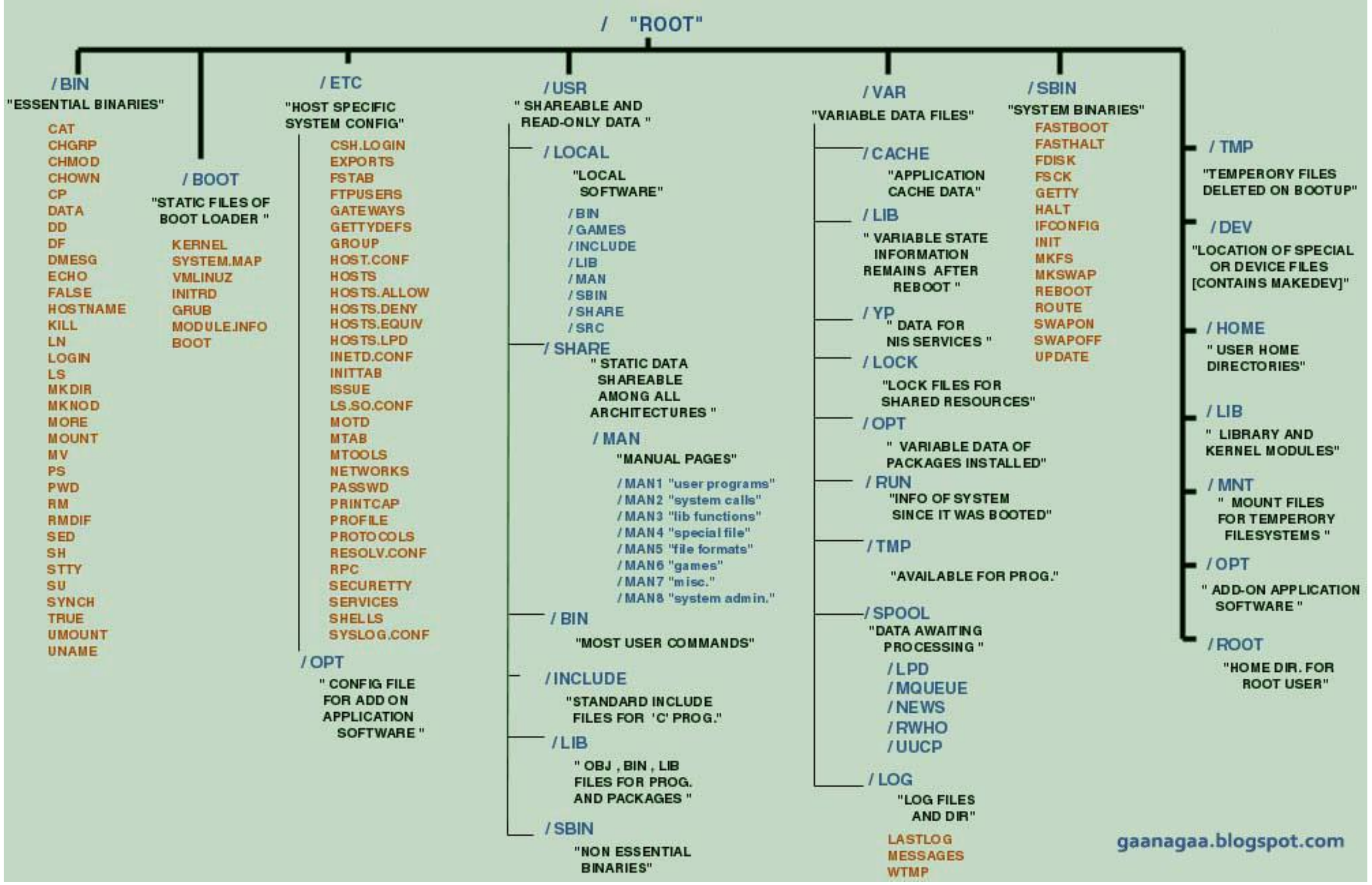
- Ideas are
  - Data Inconsistency <> Lock Mechanism

Depositing [存款]



- Ideas are
  - “君子不立危墙”和“鸵鸟策略” <> Deadlock

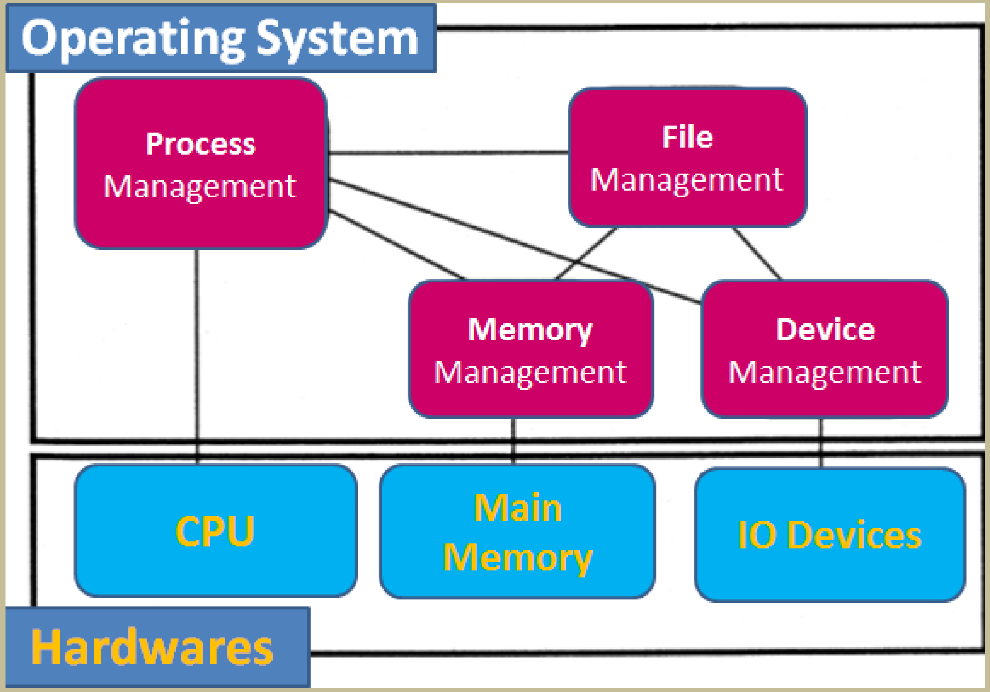
- Ideas are
  - – Structure <> Organize files

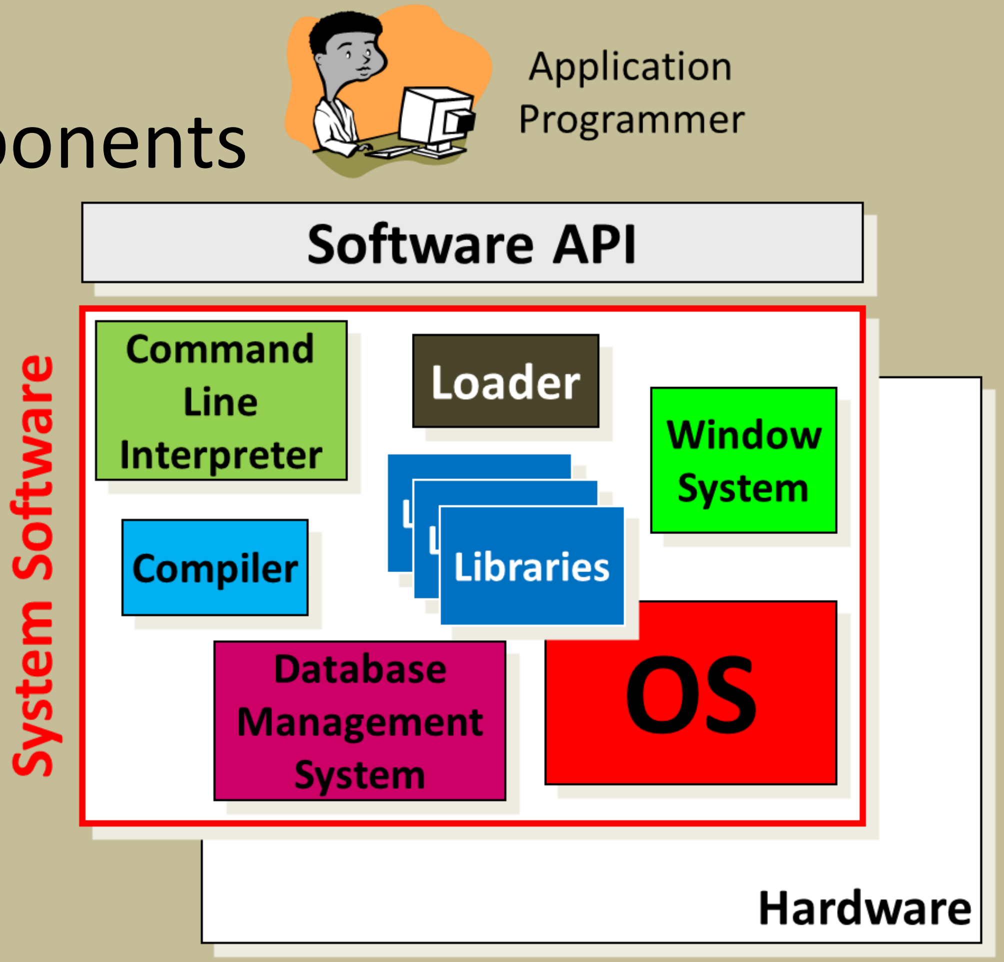


By OS now

- The software/program which contains a collection of many routines (functions, programs) to support the **automatic execution** of many **cooperated and concurrent programs**
- Many subtasks should be considered
  - **EMM**
    - **E**xecution: how is your program run?
    - **M**apping **2**: locate the program files (instructions and data) in Hdisk
    - **M**apping **1**: copy the selected program files (instructions and data) from Hdisk into appropriate regions in MM
  - **GSD**: **G**UI, **S**ecurity, **D**istributed

- We can distill 4 kinds of resources / concepts
  - **CPU, Main Memory, Hard Disk, File**
- Modern OS has 4 fundamental components





Review – Concepts

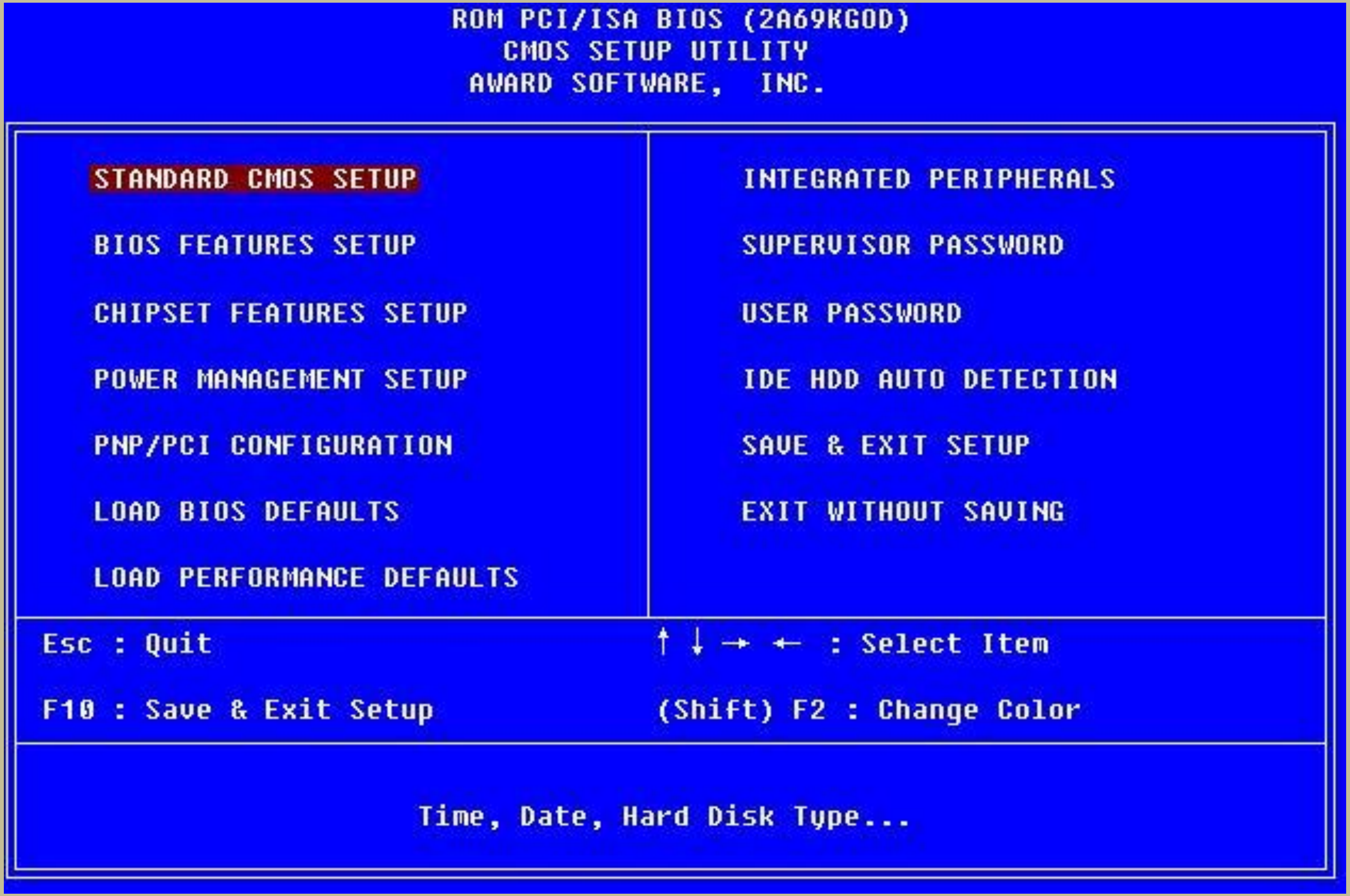
- Problems and Solution ideas to understand Modern OS
- Load OS
- Details
  - Execution of a program
    - CPU, File → Mapping 1 + 2
  - Overcome concurrency
    - Synchronization, deadlock

Before you use your OS

- When you turn on the power of your computer, the ***first program*** that runs is usually a set of instructions kept in the computer's read-only memory (**ROM**).
  - This code examines the system hardware to make sure everything is functioning properly.
    - 3 subprograms: **POST**, **BIOS**, **Boot loader**
      - Sometimes, loader is believed as part of BIOS in some textbooks
    - This **power-on self test** (**POST**) checks the CPU, memory, and **basic input-output systems** (**BIOS**) for errors and stores the result in a special memory location (**CMOS**).
    - Once the **POST** has successfully completed, the software loaded in **ROM** (sometimes called the **BIOS** or **firmware** [固件]) will begin to activate the computer's disk drives.

Complementary Metal Oxide Semiconductor
（互补金属氧化物半导体）

You could configure BIOS to set BOOT sequence



```
                    ROM PCI/ISA BIOS (2A69KGOD)
                        CMOS SETUP UTILITY
                       AWARD SOFTWARE, INC.

   STANDARD CMOS SETUP                 INTEGRATED PERIPHERALS

   BIOS FEATURES SETUP                 SUPERVISOR PASSWORD

   CHIPSET FEATURES SETUP              USER PASSWORD

   POWER MANAGEMENT SETUP              IDE HDD AUTO DETECTION

   PNP/PCI CONFIGURATION               SAVE & EXIT SETUP

   LOAD BIOS DEFAULTS                  EXIT WITHOUT SAVING

   LOAD PERFORMANCE DEFAULTS

 Esc : Quit                          ↑ ↓ → ← : Select Item

 F10 : Save & Exit Setup             (Shift) F2 : Change Color

                   Time, Date, Hard Disk Type...
```

Before you use your OS

- The **bootstrap loader** is a small program that has a single function： **It loads the operating system into memory and allows it to begin operation**.

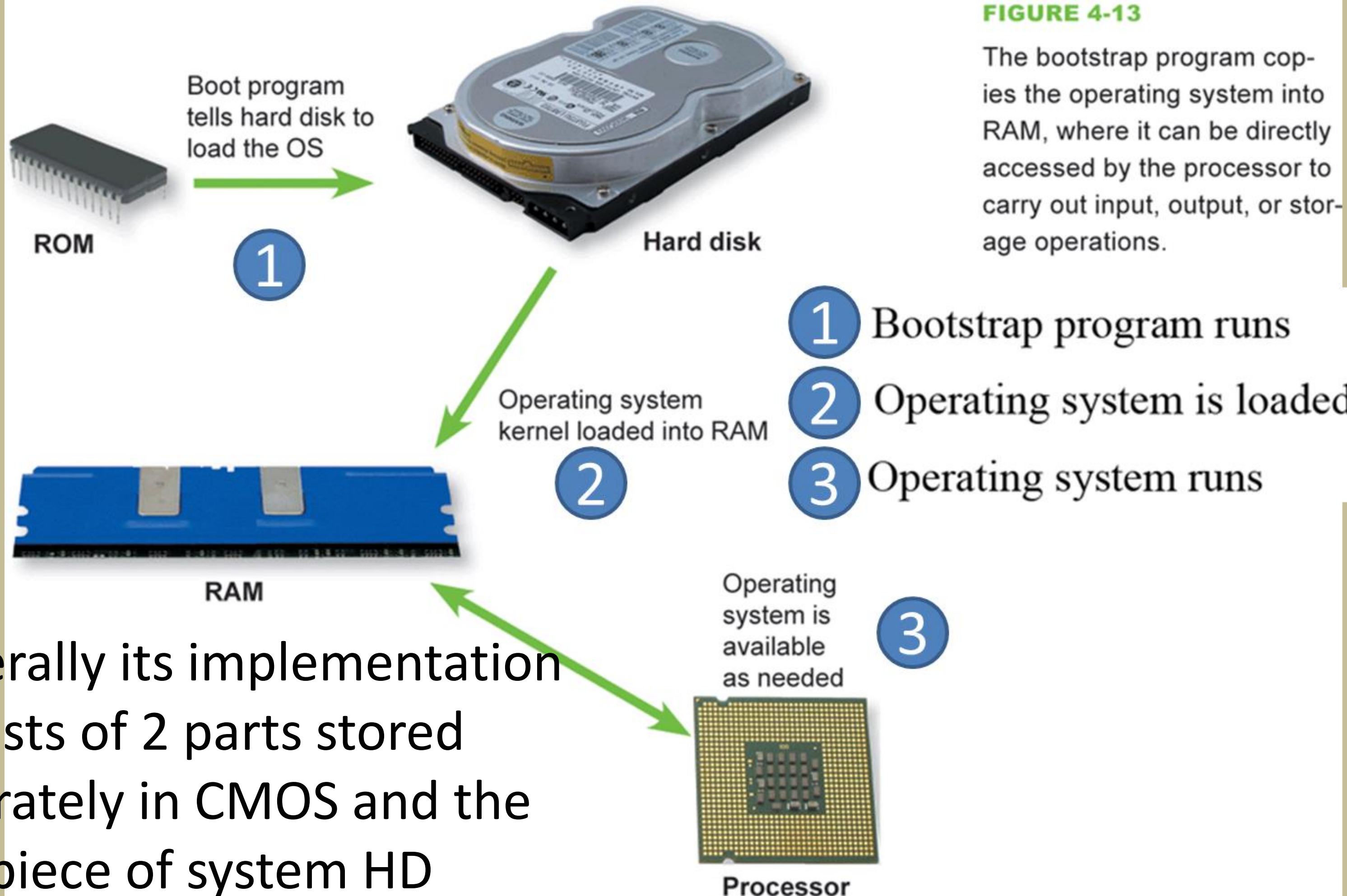

Boot program tells hard disk to load the OS
ROM
1
Hard disk
Operating system kernel loaded into RAM
2
RAM
Operating system is available as needed
3
Processor

FIGURE 4-13

The bootstrap program copies the operating system into RAM, where it can be directly accessed by the processor to carry out input, output, or storage operations.

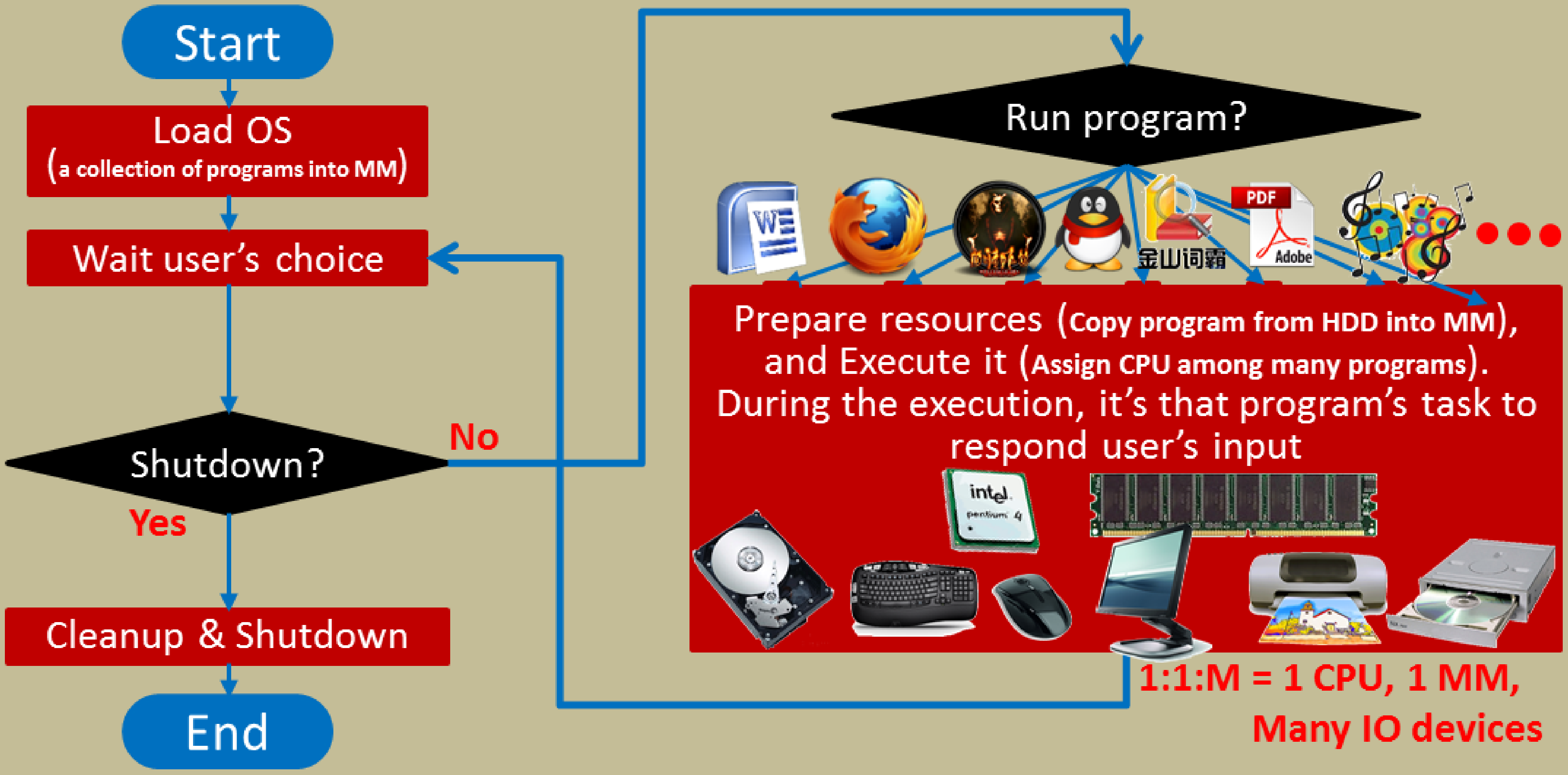
1 Bootstrap program runs

2 Operating system is loaded

3 Operating system runs

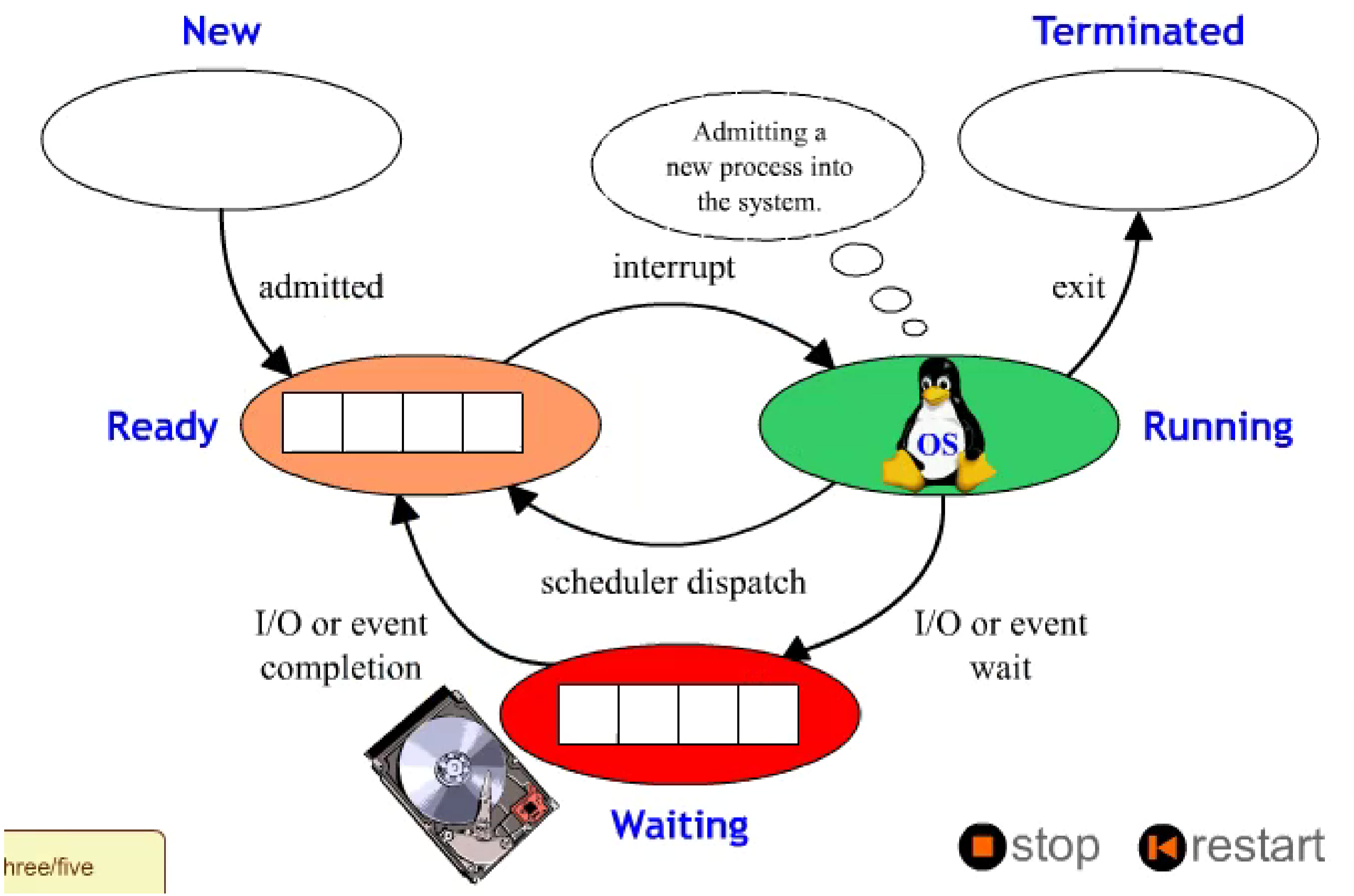
Generally its implementation consists of 2 parts stored separately in CMOS and the first piece of system HD

- Then it’s OS!


Start
Load OS
(a collection of programs into MM)
Wait user’s choice
Shutdown?
No
Yes
Cleanup & Shutdown
End
Run program?
金山词霸
PDF
Adobe
Prepare resources (Copy program from HDD into MM), and Execute it (Assign CPU among many programs). During the execution, it’s that program’s task to respond user’s input
intel pentium 4

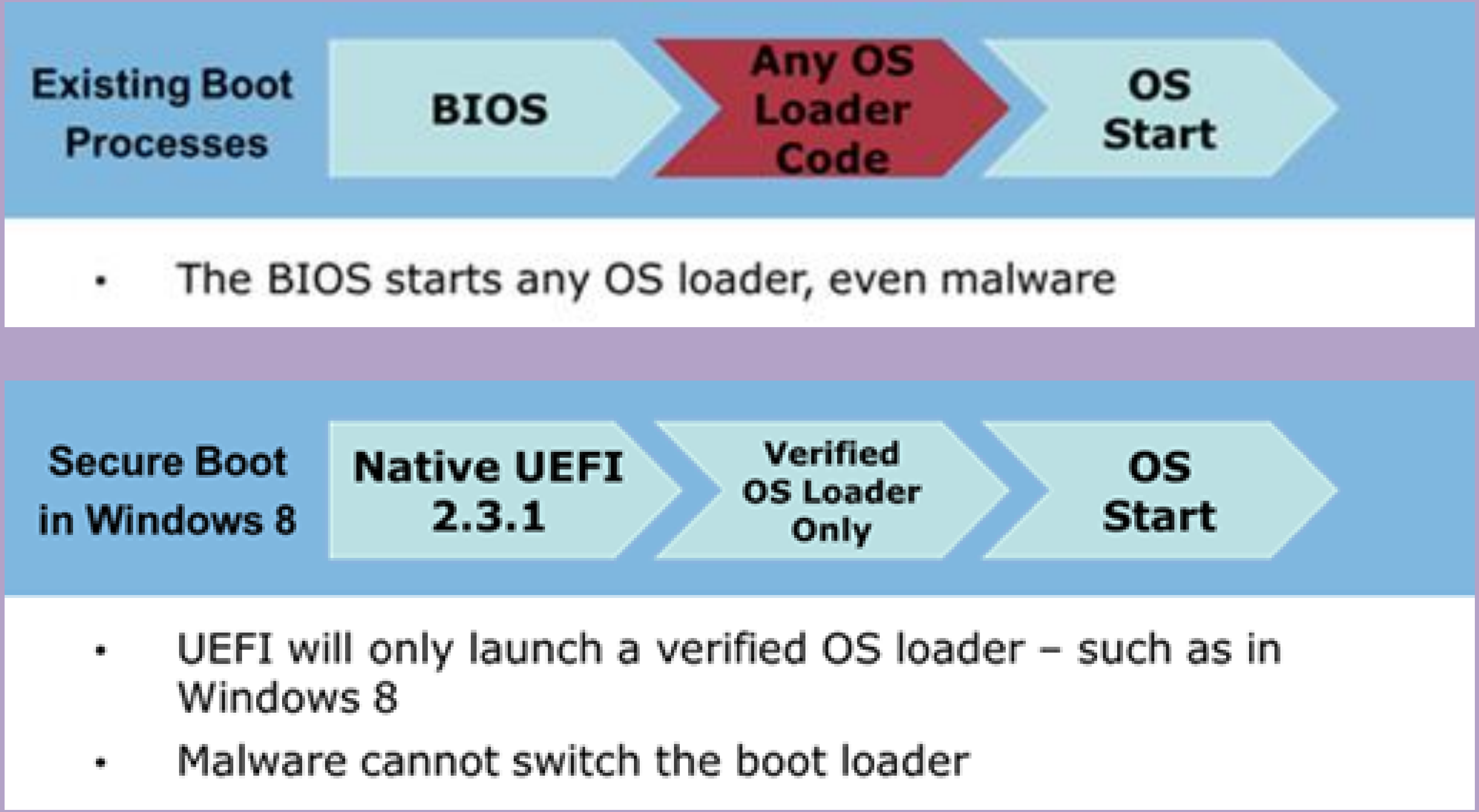
1:1:M = 1 CPU, 1 MM, Many IO devices

“The execution of a program” - It’s alive



(Unified Extensible Firmware Interface: 统一可扩展固件接口)

Security risk is everywhere – UEFI



- The BIOS starts any OS loader, even malware

- UEFI will only launch a verified OS loader – such as in Windows 8
- Malware cannot switch the boot loader

http://answers.microsoft.com/en-us/windows/forum/windows_7-security/uefi-secure-boot-in-windows-81/65d74e19-9572-4a91-85aa-57fa783f0759

Review – Concepts

- Problems and Solution ideas to understand Modern OS
- Load OS
- Details
  - Execution of a program
    - CPU, File → Mapping 1 + 2
  - Overcome concurrency
    - Synchronization, deadlock

PCB is the **very** concept/data structure

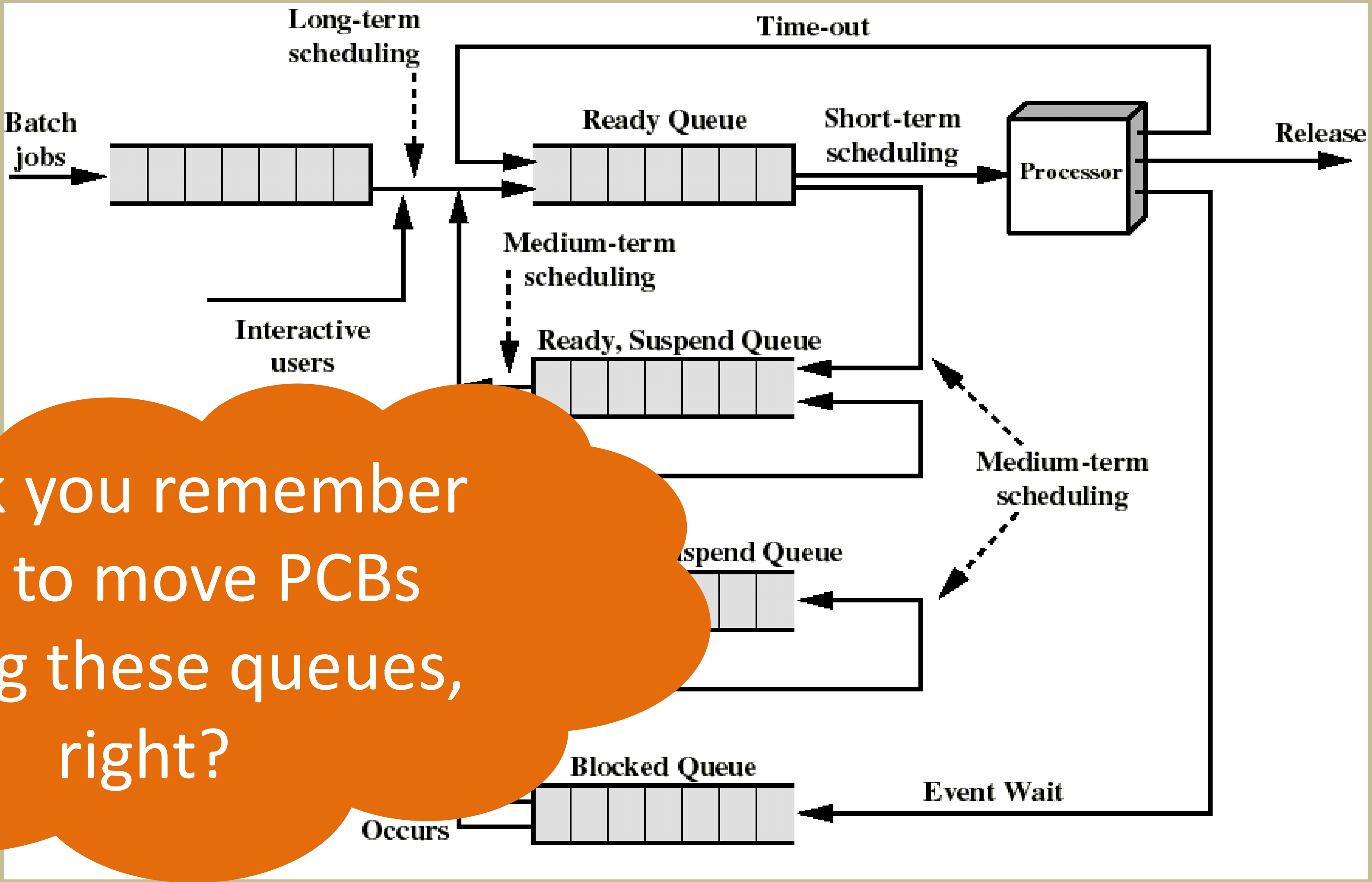
- You have learned how to record those kinds of information in **programming**.
  - Design the **data structure**!

- **PCB** (Process Control Block) is the one used/named data structure
  1. Process **location** information
  2. Process **identification** information
  3. Process **state** information
  4. Process **control** information

Supplement:

Data structures to manage those state-related processes

- **Queue**! ← You have learned it



I think you remember how to move PCBs among these queues, right?

Three kinds of schedulers

1. Long-term scheduler (jobs scheduler) – selects which programs/processes should be brought into the **ready queue**.
2. Medium-term scheduler (emergency scheduler) – selects which job/process should be **swapped** out if system is loaded.
3. Short-term scheduler (CPU scheduler) – selects which process should be **executed** next and allocates CPU.

PPTs from others\From Ariel J. Frank\OS381\os3-2.ppt

Scheduling Algorithms

- **First Come First Serve Scheduling** [先来先服务] (Non-preemptive)
- **Shortest Job First Scheduling** [最短任务先服务]
  - SRTF (Shortest Remaining Time First Scheduling)/SRJF
- **Priority Scheduling** [优先权]
- **Round-Robin Scheduling** [时间片轮转]
- Multilevel Queue Scheduling [多层次队列]
  - Multilevel Feedback-Queue Scheduling [多层次反馈队列]
- Lottery Scheduling [抽彩]
- HRRN (High Response Rate Next)

PPTs from others\www.cs.gsu.edu_~cscbecx\csc4320 Chapter 5-1.ppt

Does adding RAM always reduce misses?

- Yes for LRU and MIN
  - Memory content of X pages $\subseteq$ X + 1 pages

- **No for FIFO**
  - Due to modulo math
  - ***Belady's anomaly***: getting more page faults by increasing the memory size

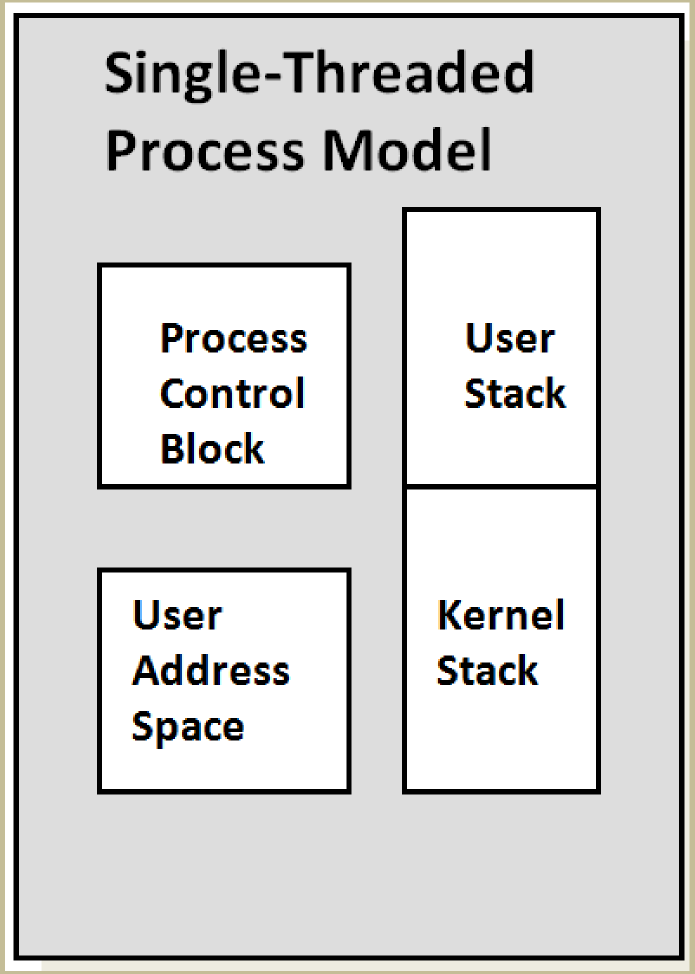
Possibility of **Thrashing**

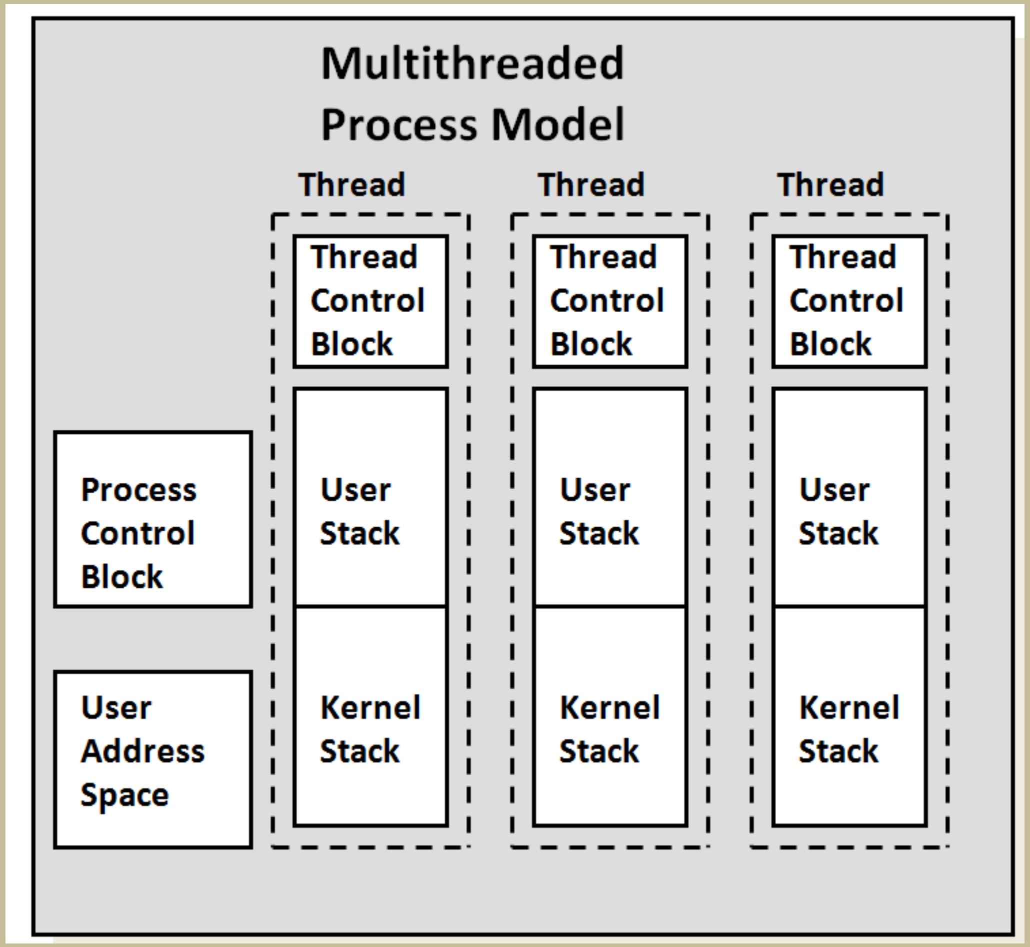
- If a process does not have “enough” pages, the page-fault rate is very high. This leads to:
  - low CPU utilization.
  - operating system thinks that it needs to increase the degree of multiprogramming.
  - another process added to the system.
  - This just increases the load on physical memory.

- **Thrashing** = a process is busy swapping pages in and out.

PPTs from others\From Ariel J. Frank\OS381\os8-2_vir.ppt

Single Threaded and Multithreaded Process Models





Thread Control Block (<u>TCB</u>) contains a register image, thread priority and thread state information

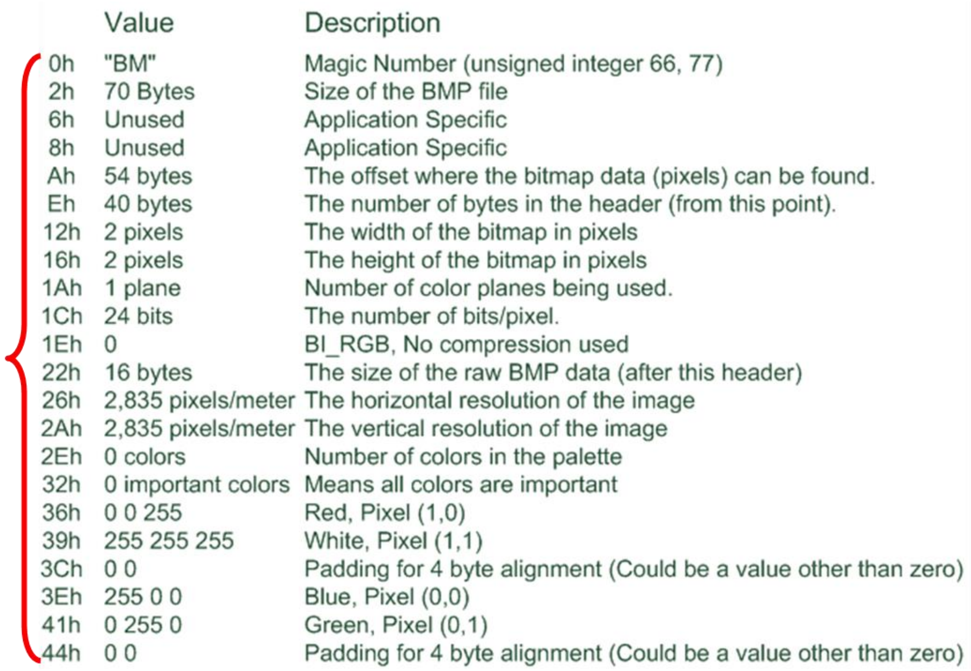
PPTs from others\OperatingSystems\Processes.ppt

Review – Concepts

- Problems and Solution ideas to understand Modern OS
- Load OS
- Details
  - Execution of a program
    - CPU, File → Mapping 1 + 2
  - Overcome concurrency
    - Synchronization, deadlock

- A program – Logical space as the File

Continuous space

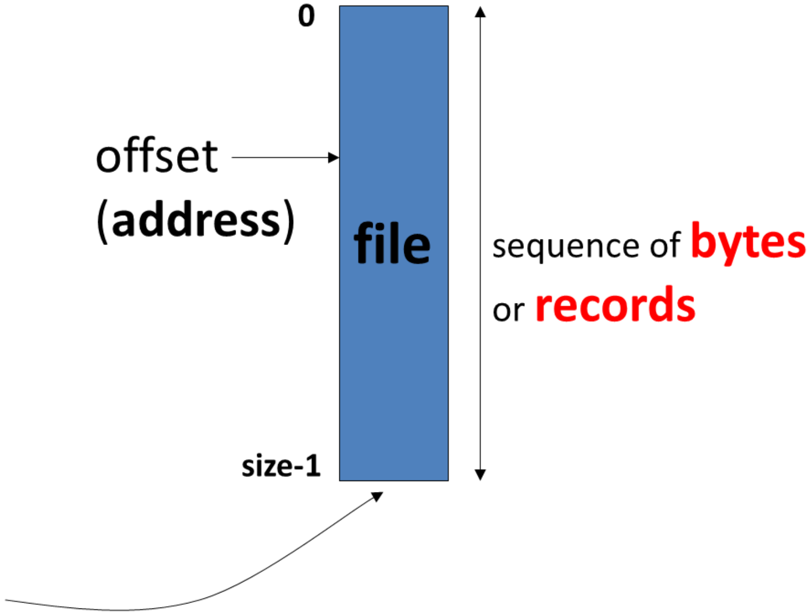


	Value	Description
0h	"BM"	Magic Number (unsigned integer 66, 77)
2h	70 Bytes	Size of the BMP file
6h	Unused	Application Specific
8h	Unused	Application Specific
Ah	54 bytes	The offset where the bitmap data (pixels) can be found.
Eh	40 bytes	The number of bytes in the header (from this point).
12h	2 pixels	The width of the bitmap in pixels
16h	2 pixels	The height of the bitmap in pixels
1Ah	1 plane	Number of color planes being used.
1Ch	24 bits	The number of bits/pixel.
1Eh	0	BI_RGB, No compression used
22h	16 bytes	The size of the raw BMP data (after this header)
26h	2,835 pixels/meter	The horizontal resolution of the image
2Ah	2,835 pixels/meter	The vertical resolution of the image
2Eh	0 colors	Number of colors in the palette
32h	0 important colors	Means all colors are important
36h	0 0 255	Red, Pixel (1,0)
39h	255 255 255	White, Pixel (1,1)
3Ch	0 0	Padding for 4 byte alignment (Could be a value other than zero)
3Eh	255 0 0	Blue, Pixel (0,0)
41h	0 255 0	Green, Pixel (0,1)
44h	0 0	Padding for 4 byte alignment (Could be a value other than zero)

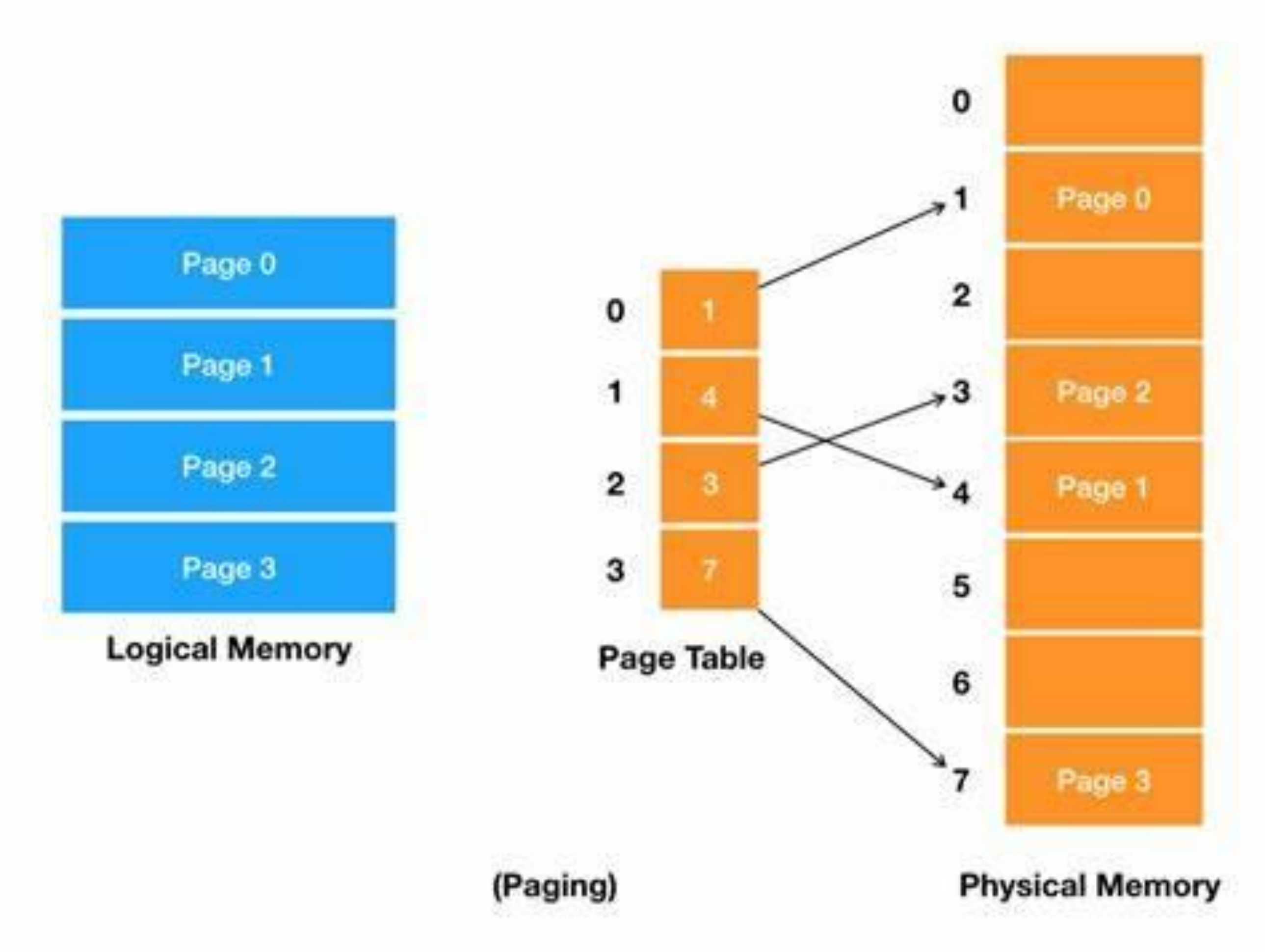
00.Computer & Related\PPTs.2009\Part III Operating System.pptx

- Contiguous logical address space (a storage)
- Content:
  - Data
    - numeric
    - character
    - binary
  - Program

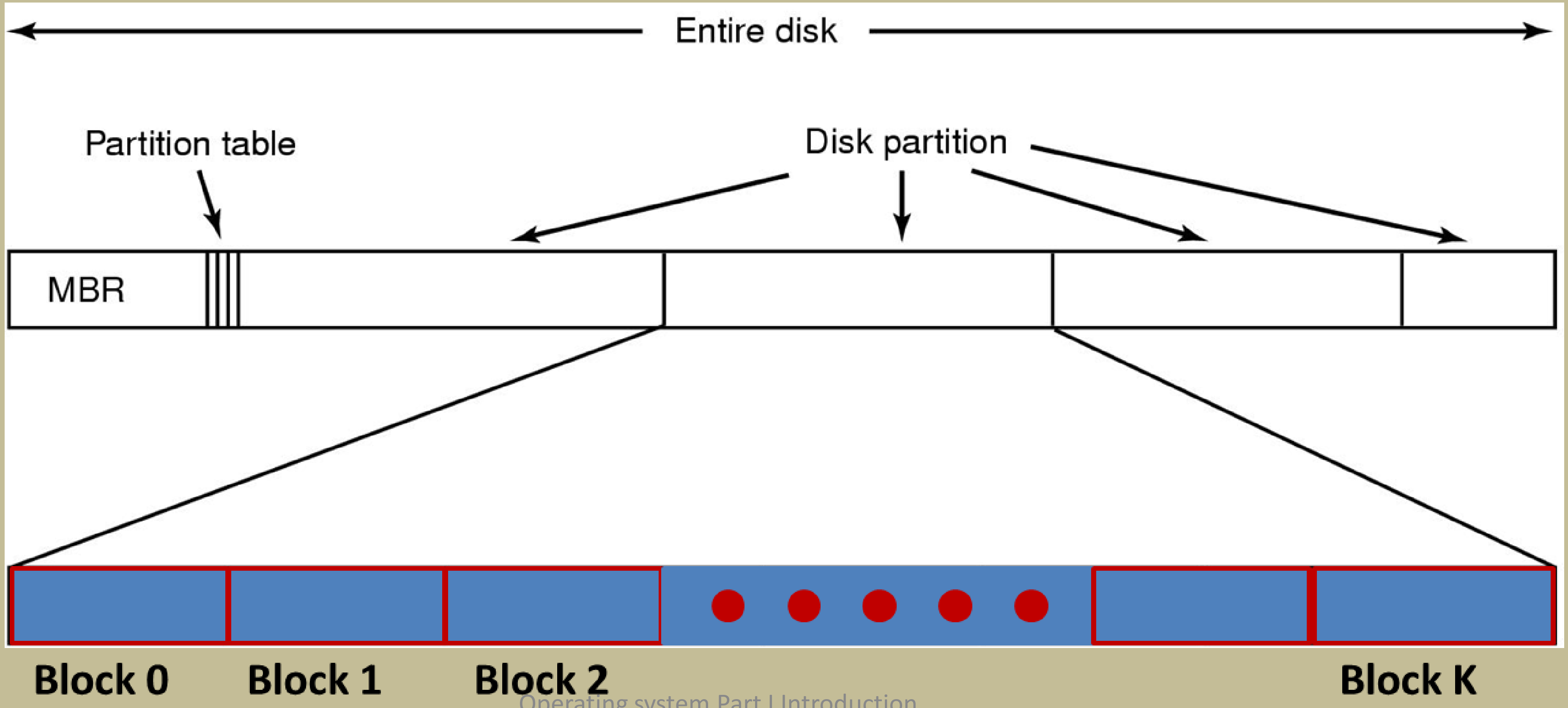
User's (processes') view of a file



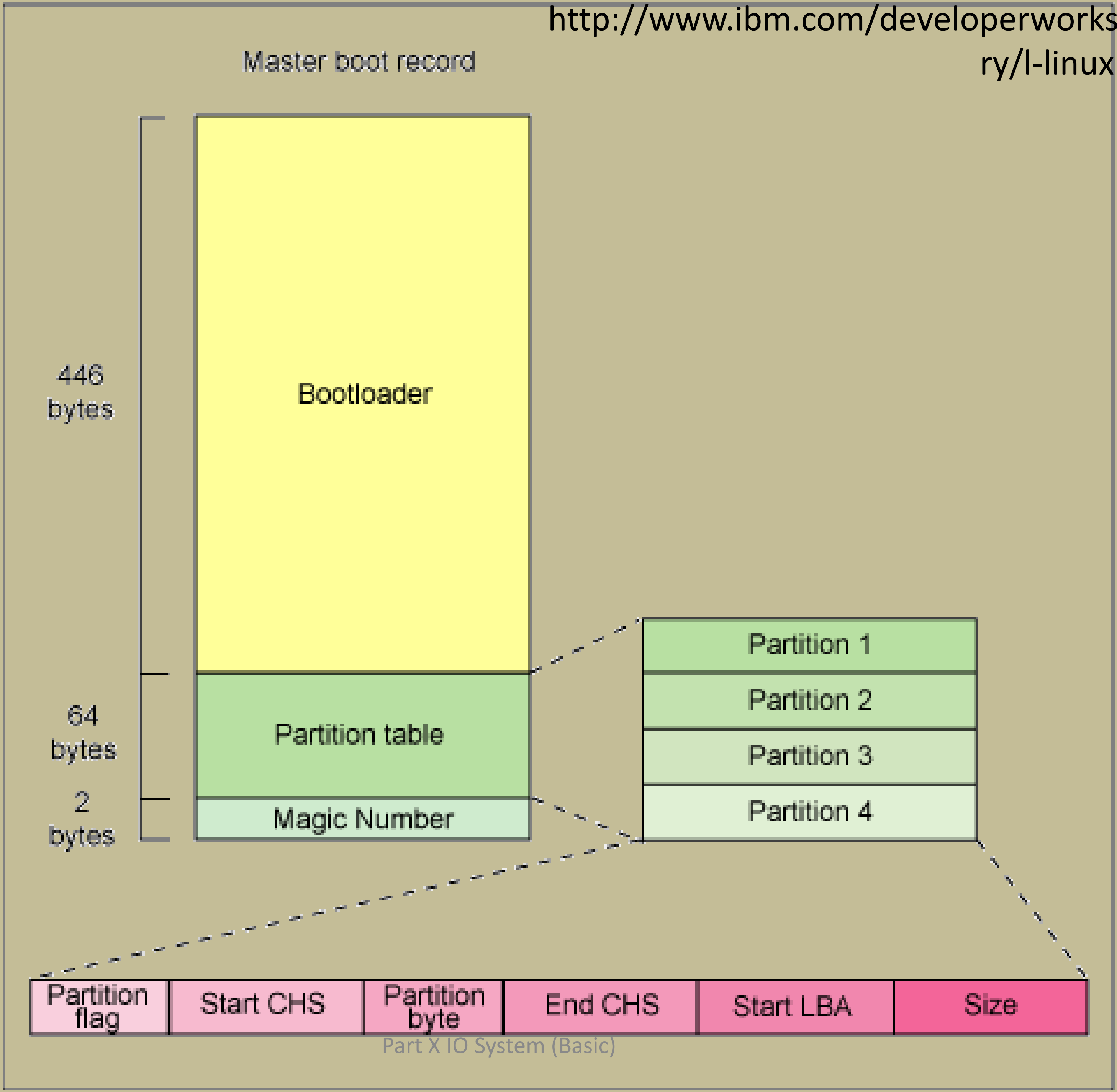
- Linear addressed block/frame space – because **Paging** is the basis for current OSs
  - – **MM**, HDD, CDROM, Fdisk, …
    - Registers, Caches, ROM, **RAM**, Video buffer, …
  - – Virtual **M**emory concept



- Linear addressed block/frame space – because **Paging** is the basis for current OSs
  - – MM, **HDD, CDROM, Fdisk**, …

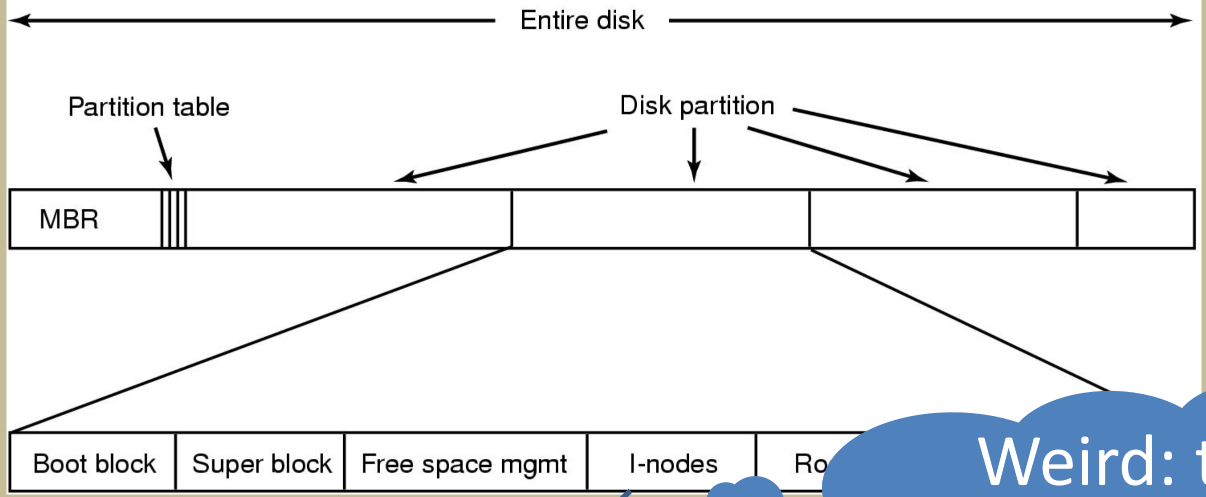


http://www.ibm.com/developerworks/library/l-linuxboot/

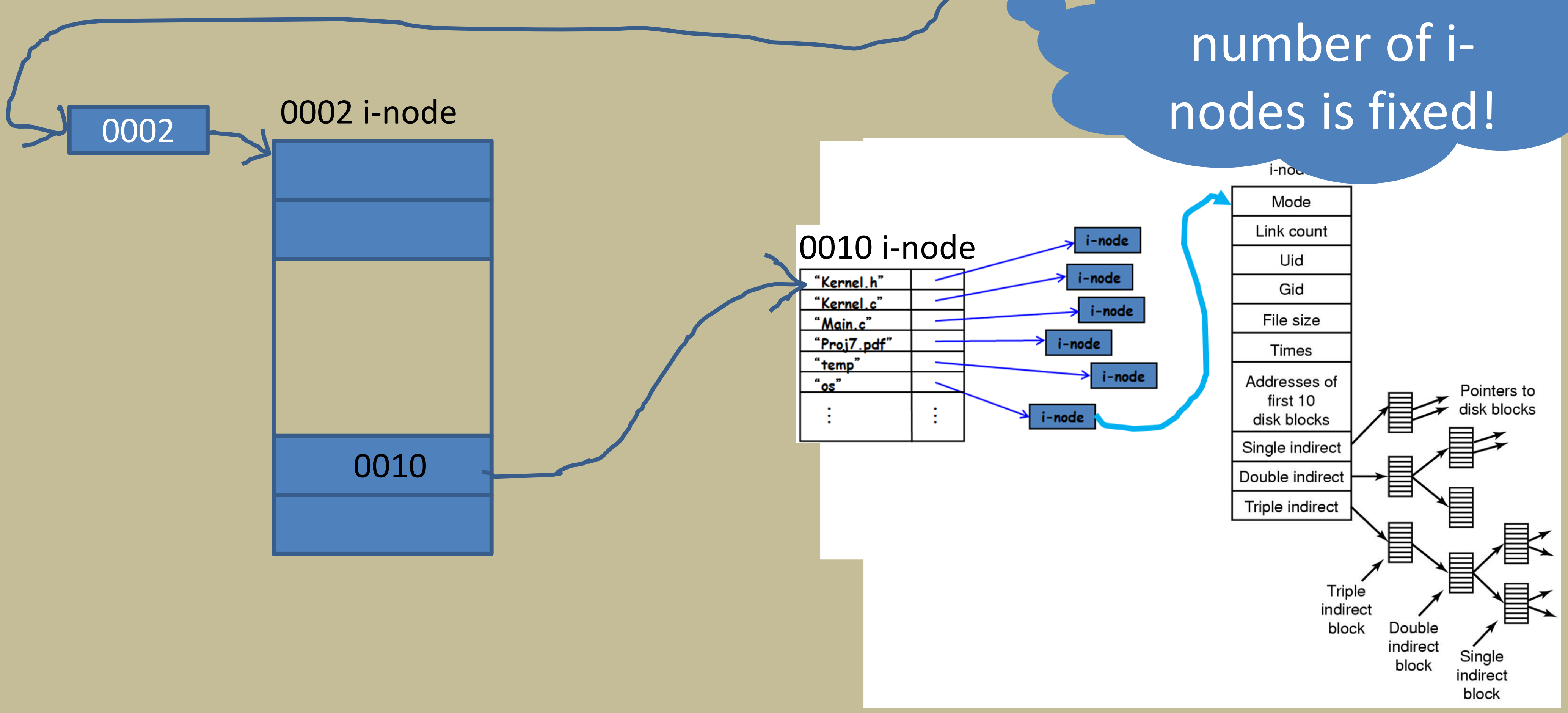


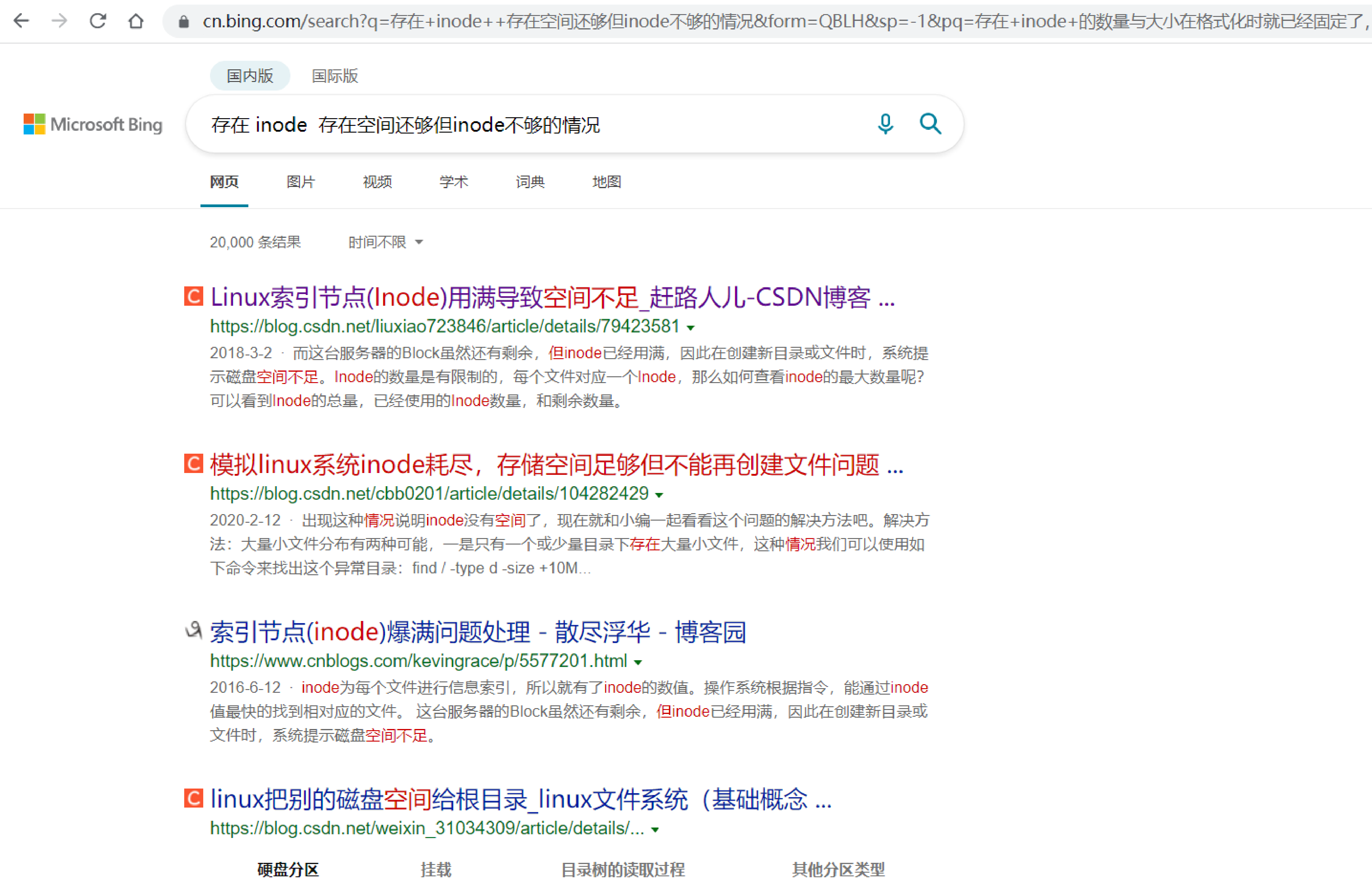
Winter

The UNIX I-node



Weird: the number of i-nodes is fixed!





cn.bing.com/search?q=存在+inode++存在空间还够但inode不够的情况&form=QBLH&sp=-1&pq=存在+inode+的数量与大小在格式化时就已经固定了,

存在 inode 存在空间还够但inode不够的情况

20,000 条结果

Linux索引节点(Inode)用满导致空间不足_赶路人儿-CSDN博客 ...

https://blog.csdn.net/liuxiao723846/article/details/79423581

2018-3-2 · 而这台服务器的Block虽然还有剩余，但inode已经用满，因此在创建新目录或文件时，系统提示磁盘空间不足。Inode的数量是有限制的，每个文件对应一个Inode，那么如何查看inode的最大数量呢?可以看到Inode的总量，已经使用的Inode数量，和剩余数量。

模拟linux系统inode耗尽，存储空间足够但不能再创建文件问题 ...

https://blog.csdn.net/cbb0201/article/details/104282429

2020-2-12 · 出现这种情况说明inode没有空间了，现在就和小编一起看看这个问题的解决方法吧。解决方法：大量小文件分布有两种可能，一是只有一个或少量目录下存在大量小文件，这种情况我们可以使用如下命令来找出这个异常目录：find / -type d -size +10M...

索引节点(inode)爆满问题处理 - 散尽浮华 - 博客园

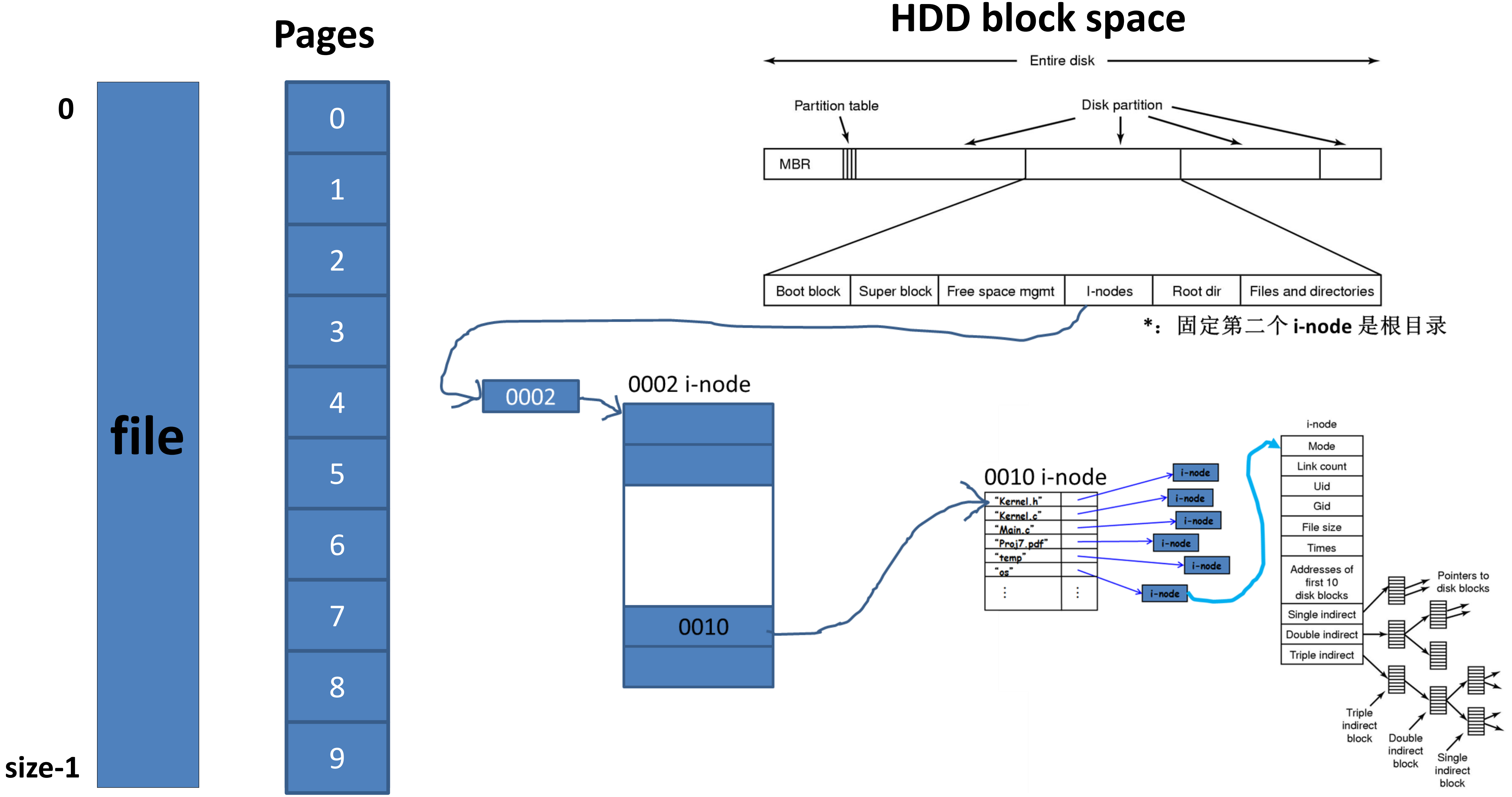
https://www.cnblogs.com/kevingrace/p/5577201.html

2016-6-12 · inode为每个文件进行信息索引，所以就有了inode的数值。操作系统根据指令，能通过inode值最快的找到相对应的文件。 这台服务器的Block虽然还有剩余，但inode已经用满，因此在创建新目录或文件时，系统提示磁盘空间不足。

linux把别的磁盘空间给根目录_linux文件系统（基础概念 ...

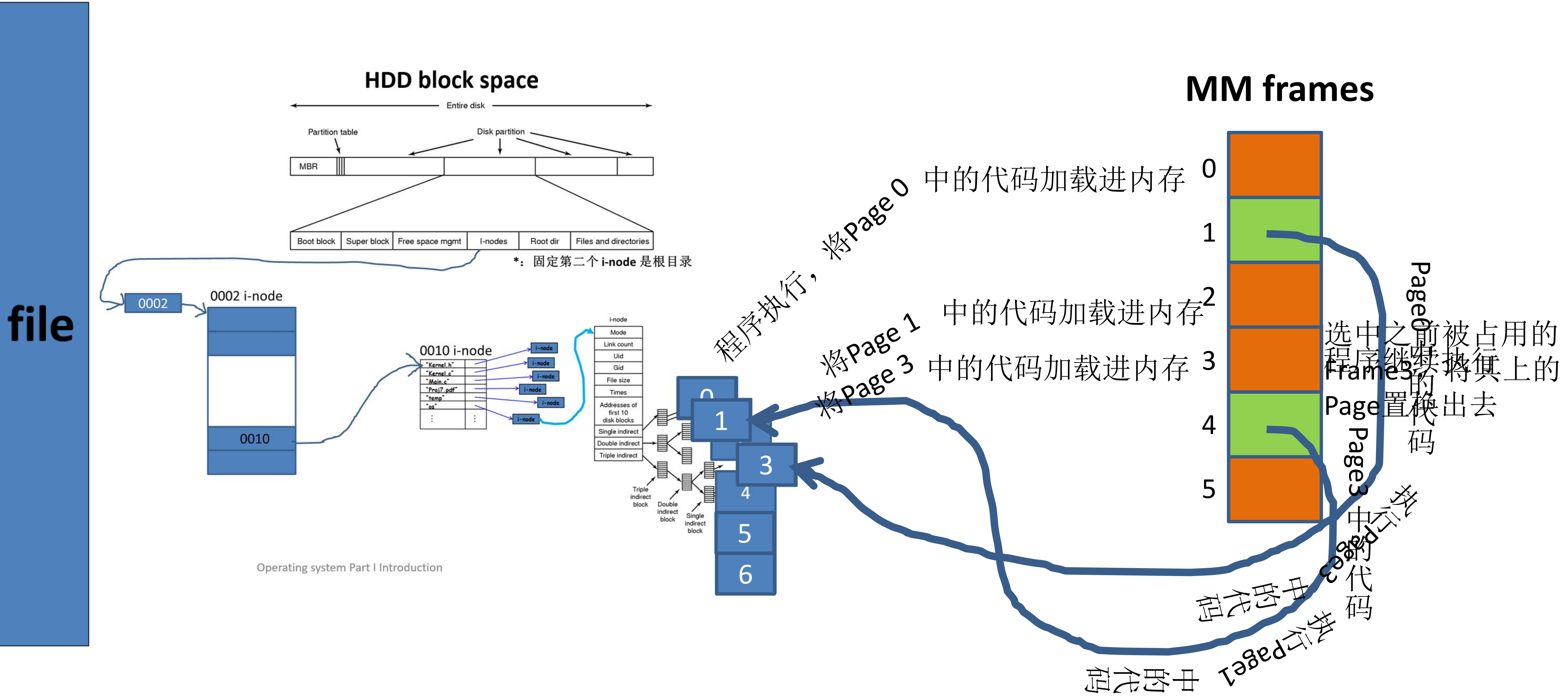
https://blog.csdn.net/weixin_31034309/article/details/...

HDD block space



*：固定第二个 i-node 是根目录

when on-demand paging



Summary – Main Memory Management

- Old ways
  - Partitioning (Fixed or Variable)
  - Overlay
  - Buddy’s algorithm
- **Virtual Memory (VM)**
  - Paging
    - Paging-based VM (On-demand paging)
      - How to support the transparency of using space larger than the physical memory space
    - Page replacement algorithms
  - Segmenting
    - Segmentation-based VM (On-demand Segmenting)
      - How to support the transparency of using space larger than the physical memory space
  - Segment-page scheme (Hybrid)

Summary of Disk scheduling algorithms

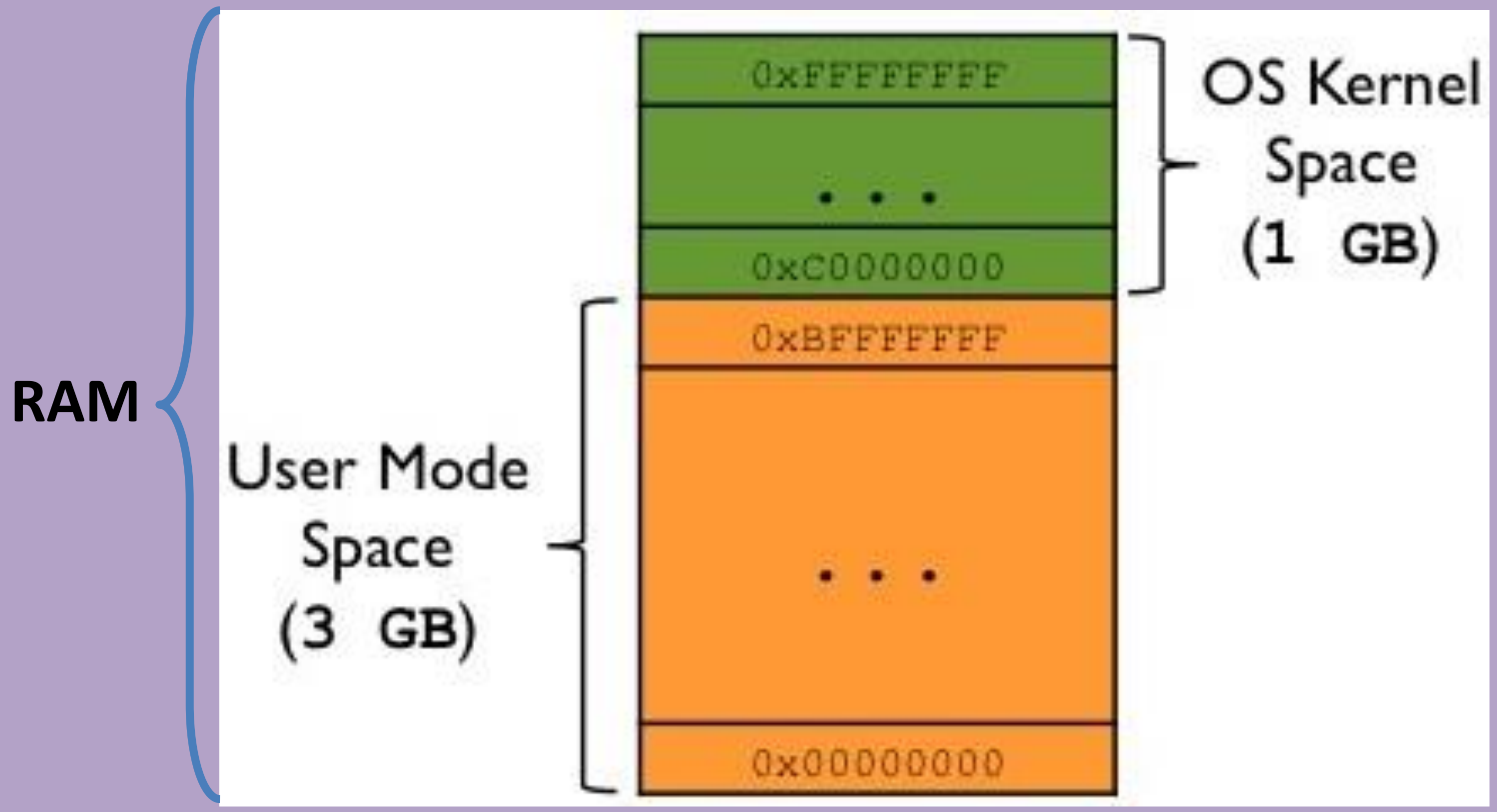
- Algorithms
  - FCFS, SSTF, Scan, C-Scan, Look, C-Look

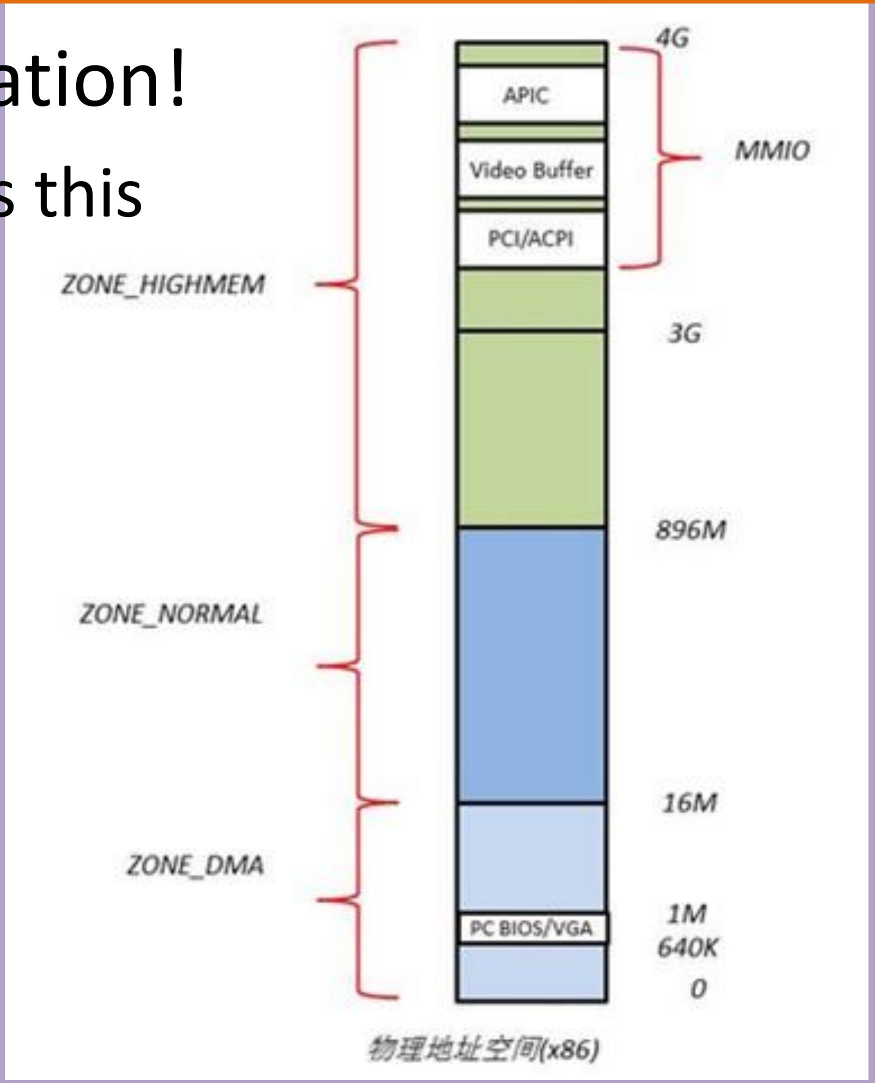
Go until Direction	Go until the last cylinder	Go until the last request
Service both directions	**Scan**	**Look**
Service in only one direction	**C-Scan**	**C-Look**

- Question
  - a request queue (0-199).
    - 98, 183, 37, 122, 14, 124, 65, 67
    - After visiting 40, current Head pointer is at 53

Attention!

- MMM is more complicated in real application!
  - – Try yourself ☺ - like to check how Linux does this





Review – Concepts

- Problems and Solution ideas to understand Modern OS
- Load OS
- Details
  - Execution of a program
    - CPU, File → Mapping 1 + 2
  - Overcome concurrency
    - Synchronization, deadlock

Summarized as **Mutual Exclusion** [互斥] problem

http://en.wikipedia.org/wiki/Mutual_exclusion

- Mutual exclusion, in computer science, refers to the problem of **ensuring that no two processes or threads** (henceforth referred to only as processes) **can be in their code to access the shared data among those processes at the same time**
  - It was formally defined by **Edsger Dijkstra** in 1965.

Died at
August 6, 2002
(aged 72)

Turing award 1972

The key of synchronization (access to the shared data)

TO CONTROL THE EXECUTION OF CRITICAL SECTIONS AMONG CONCURRENT PROCESSES/THREADS

⇨ Ensuring **<u>Mutual Exclusion</u>**

Every time, no more than one process is accessing the shared data

Rules for robust synchronization

- Of course, **Mutual exclusion** should be guaranteed (**consistency**) [互斥]
  - No more than one process/thread in critical section at a time
- Progress (**deadlock-free**) [有空让进]
  - If several simultaneous requests, must allow one to proceed
  - Must not depend on threads outside critical section
- Bounded (**starvation-free**) [有限等待]
  - Must eventually allow each waiting thread to enter

PPTs.2012\PPTs from others\pages.cs.wisc.edu_~remzi_Classes_537_Fall2005\lecture4.ppt

Types of solutions to CS problem

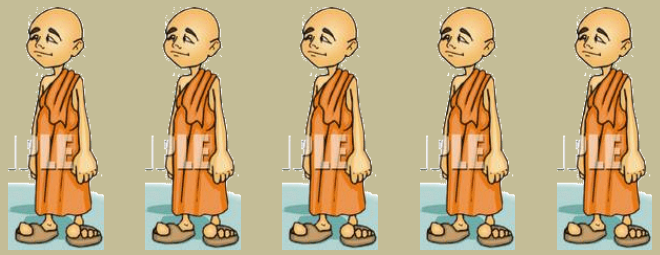
- **Software** solutions –
  - algorithms who's correctness does not rely on any other assumptions.
- **Hardware** solutions –
  - rely on some special machine instructions.
- **Operating System** solutions –
  - provide some functions and data structures to the programmer through system/library calls.
- **Programming Language** solutions –
  - Linguistic constructs provided as part of a language.

General rules to cope with CS problem using semaphores

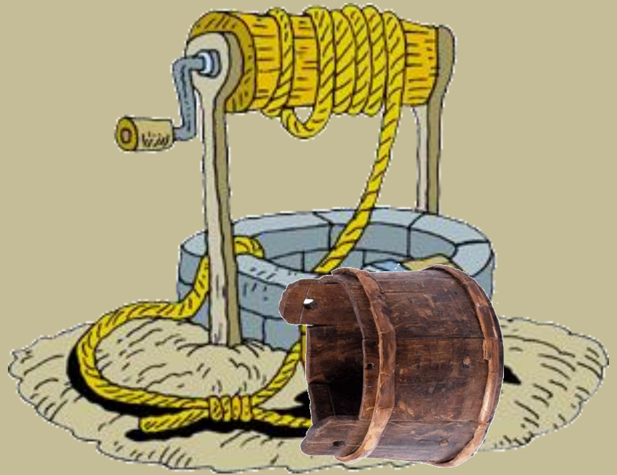
1. Find the types of actors
   - To determine the **processes**
2. Recognize the shared resources between actors → initial values of semaphores
3. Conclude the constraints based on the situations when actors use those shared resources
   - **ME or SCH**?
     - To determine semaphores and their initial values
     - To determine the code (nested for ME, and scattered for SCH)
4. Use semaphores to finish those processes

A story: monks drink water

- Many monks
  - Some are **old**
  - Some are **young**



- One well
  - Only one bucket in well every time



- One Vat
  - Can contain 10 buckets of water
  - One bucket to put water into and fetch water from the vat

- Three buckets in total

Another problem

- The unisex bathroom problem
  - there should **never be** more than 10 people in the bathroom at once
  - there should **never be** both males and females in the bathroom at once
  - there is **no** starvation and **no** deadlock

CLASSIC SYNCHRONIZATION MODELS

- Producer-Consumer model
- Readers-Writers Problem
- The Barbershop Problem
- Dining philosopher problem

Review – Concepts

- Problems and Solution ideas to understand Modern OS
- Load OS
- Details
  - Execution of a program
    - CPU, File → Mapping 1 + 2
  - Overcome concurrency
    - Synchronization, deadlock

Methods for Handling Deadlocks

- Staying Safe (君子不立危墙)
  - Providing **enough resources**
  - **Preventing** Deadlocks
  - **Avoiding** Deadlocks
- Living Dangerously (鸵鸟策略)
  - Let the deadlock happen, then **detect** it and **recover** from it.
  - **Ignore** the risks

Providing enough resources - A useful equation!

- Given:
  - Here are **3** processes: A, B, C. Each of them requires **5** system resources.
- Question:
  - How many resources should the system at least have so that the system is safe?
- Rule:
  - If the number of system resources satisfies the following equation, then the system is safe!

$$\sum (P_{\max} - 1) + 1 \leq R_{Total}$$

Staying Safe - Deadlock Prevention (预防)

- Do not allow one of the four conditions to occur.
  - **Mutual Exclusion** [互斥]
    - Only one process may use a resource at a time
  - **Hold and Wait** [持有和等待]
    - A process may hold allocated resources while awaiting assignment of others
  - **No Preemption** [非抢占]
    - No resource can be forcibly removed form a process holding it
  - **Circular Wait** [循环等待]
    - A closed chain of processes exists, such that each process holds at least one resource needed by the next process in the chain

Staying Safe - Deadlock **Avoidance** (避免)

- Deadlock prevention → low device utilization and reduced system throughput.
- Deadlock avoidance
  - Given the complete sequence of requests and releases for each process, we can decide for each request whether or not the process should wait.
  - For every request, the system
    - considers the resources currently **available**, the resources currently **allocated**, and the **future (Needed)** requests and releases of each process, and
    - decides whether the current request can be satisfied or must wait to avoid a possible future deadlock.

PPTs from others\OS PPT in English\ch08.ppt

Living Dangerously -

- the Ostrich[鸵鸟] or **Head-in-the-Sand** algorithm

D.J.[ˈɔstritʃ]

- **Of course**, Try to reduce chance of deadlock as far as reasonable
- **And**, accept that deadlocks will occur occasionally
  - example: kernel table sizes - max number of pages, open files etc.
- **Because**, maybe
  - MTBF versus deadlock probability ?
  - cost of any other strategy may be too high
    - overheads and efficiency

MTBF : mean-time between "failures"

Most Operating systems do this!!

PPTs from others\www.dcs.ed.ac.uk_teaching_cs3_osslides\deadlock.ppt

When deadlock happened - Detect & Recover

- **Check** for deadlock (periodically or sporadically[偶发地,零星地]), then **recover**

- Differentiate between
  - Serially reusable resources: A unit must be allocated before being released
  - Consumable resources: Never release acquired resources; resource count is the number currently available

PPTs from others\cms.dt.uh.edu\chap10.ppt