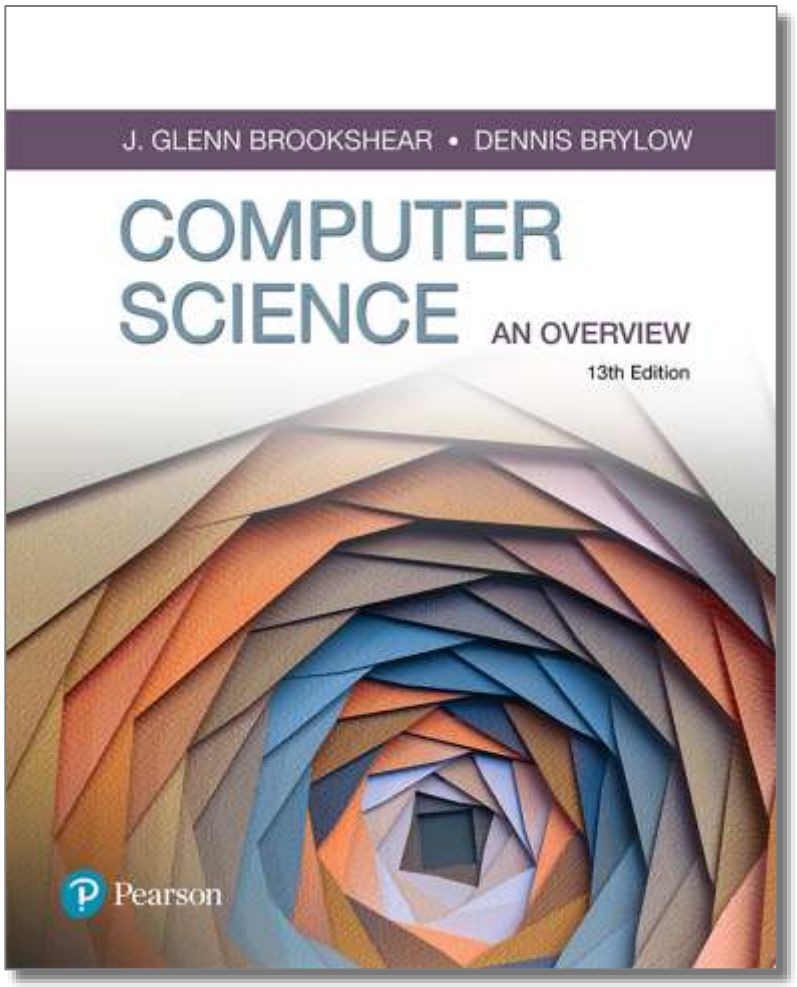
Computer Science An Overview

13th Edition



Chapter 3

Operating Systems

- Who can get the microphone?

We Need a Chairperson!

- The chairperson decides who gets the microphone to speak next
- Two ways to schedule:
  - Let each speaker talk until he/she finishes
  - "Interrupts" the speaker to get back the microphone and turn to another speaker → time sharing

Suppose the computer runs only one program …

Full control of everything, e.g.
CPU

Manage everything

In a single-processor computer, CPU is shared by all programs

Suppose the computer runs many programs …

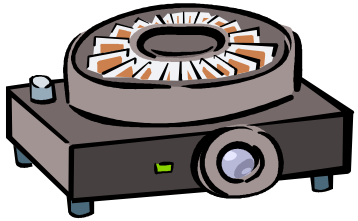
- How do they get executed?
- How do they get the most important resource – the CPU?

Chairperson Can Do More

- Which portion of blackboard a speaker can write?
  → memory management

Chairperson is OS

- Who can use 幻灯机 (projector)?
  - Device driver and management



How many operating systems do you know?

What is the role of the operating system?

What is an Operating System?

Technical Definition: An OS is a collection of system software that coordinates between the hardware, provides a platform for software to run on and provides the user with an interface for command inputs.

The governmental systems like Postal system, Railway System are analogous to Operating Systems.

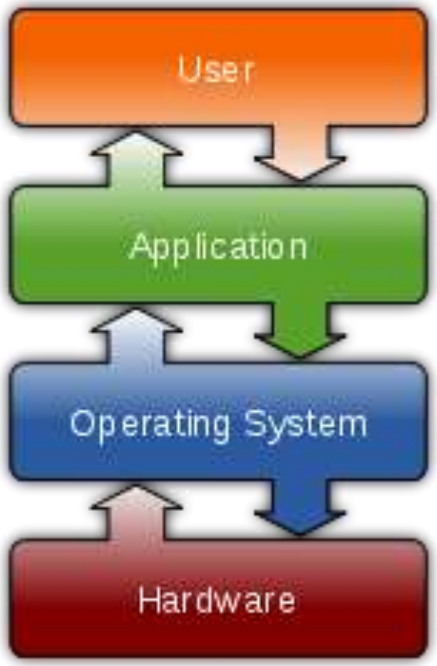
Chapter 3: Operating Systems

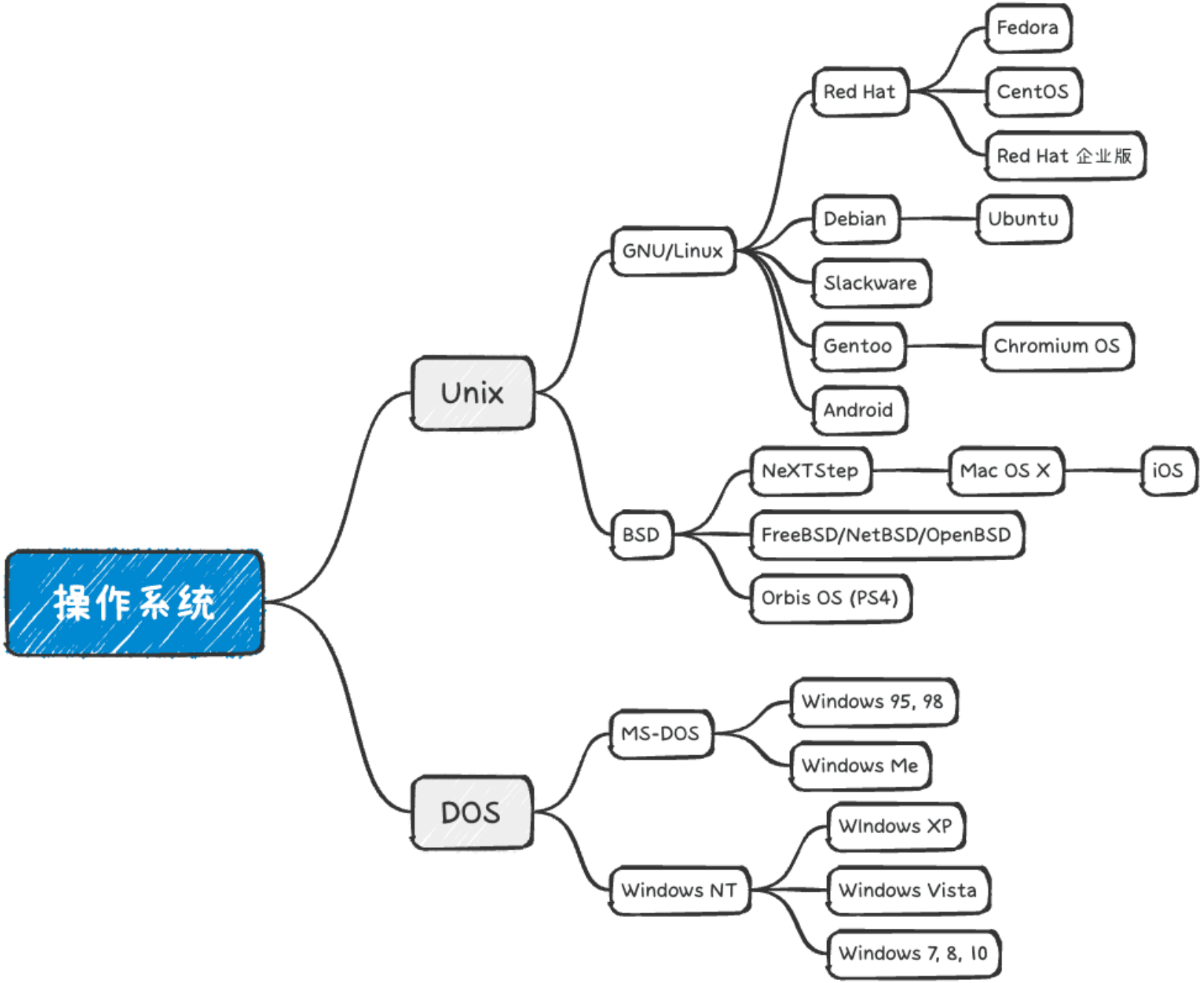
- 3.1 The History of Operating Systems
- 3.2 Operating System Architecture
- 3.3 Coordinating the Machine's Activities
- 3.4 Handling Competition Among Processes
- 3.5 Security

- 操作系统种类:
  - 智能卡OS、实时OS 、传感器节点OS 、嵌入式OS 、个人计算机OS 、网络OS和大型机OS

Different operating systems

- Unix, Solaris, Chrome OS
- Windows, Dos, Mac OS, Linux
- 手机：iOS, Android, HarmonyOS
- 嵌入式**实时**操作系**统**:µC/OS-II、嵌入式Linux、Windows Embedded、VxWorks




操作系统
Unix
GNU/Linux
Red Hat
Fedora
CentOS
Red Hat 企业版
Debian
Ubuntu
Slackware
Gentoo
Chromium OS
Android
BSD
NeXTStep
Mac OS X
iOS
FreeBSD/NetBSD/OpenBSD
Orbis OS (PS4)
DOS
MS-DOS
Windows 95, 98
Windows Me
Windows NT
WIndows XP
Windows Vista
Windows 7, 8, 10

桌面OS

- 桌面操作系统主要用于个人计算机上。个人计算机从硬件架构上来说主要分为PC机与Mac机，从软件上可主要分为两大类，分别为类Unix操作系统和Windows操作系统
- 类Unix操作系统：Mac OS X，Linux发行版（如Debian，Ubuntu，openSUSE，Red Hat，）；
- Windows: Windows 98，Windows 2000，Windows XP，Windows Vista，Windows 7，Windows 10，Windows11

服务器OS

- 服务器操作系统一般指的是安装在大型计算机上的操作系统，比如Web服务器、应用服务器和数据库服务器等
- Unix系列：SUNSolaris，FreeBSD，OS X Server 等；
- Linux系列：Red Hat Linux，Debian，UbuntuServer等；
- Windows系列：Windows NT Server，Windows Server 2008，Windows server 2012，windows server technical
- 在服务器方面Linux、UNIX和Windows Server占据了市场的大部分份额。在超级计算机方面，Linux取代Unix成为了第一大操作系统

DOS (Disk Operation System)

single-user single-task

Windows (from 1985-present)

Mac OS (from 1984-present)

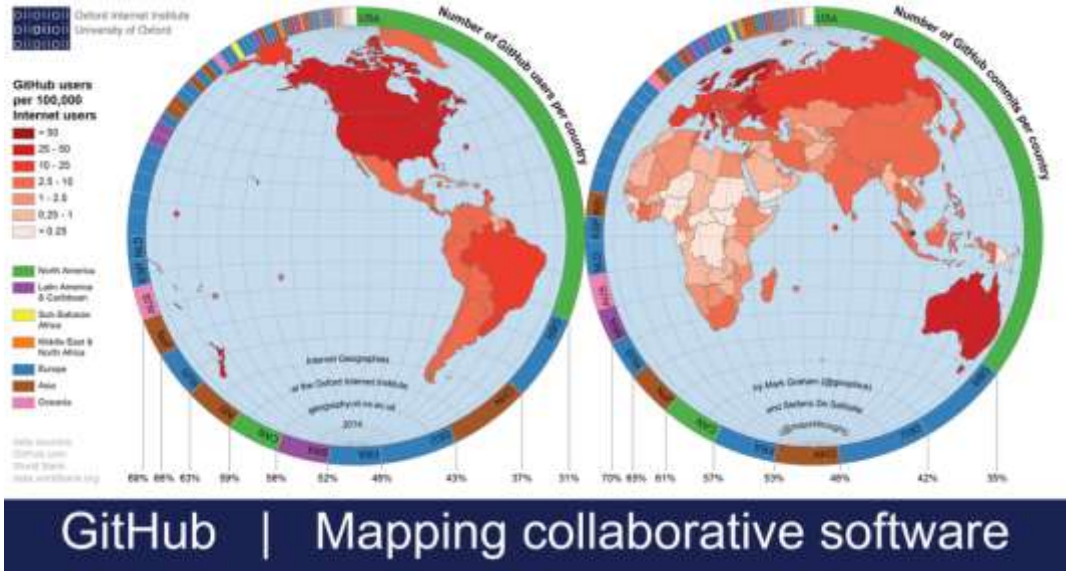
Linux

- First released on 5 October 1991 by Linus Torvalds

GNU

- A free software, mass collaboration project, announced on 27 September 1983, by Richard Stallman at MIT.
- 旨在开发一个类似 Unix， 且为自由软件的完整操作系统GNU 系统
- GNU是一个自由软件操作系统，所用的典型内核是Linux，Linux成为常见的GNU计划软件运行平台
- 改变了计算机技术的发展方向，为软件开发者们提供了一个开放的平台

- Git: 分布式版本控制系统
- Github：全球最大的社交编程及代码托管网站，为开源项目免费提供Git管理



Functions of Operating Systems

- Oversee operation of computer
- Store and retrieve files
- Provide the user interface to request execution of programs
- Coordinate the execution of programs

3.1 History of Operating Systems

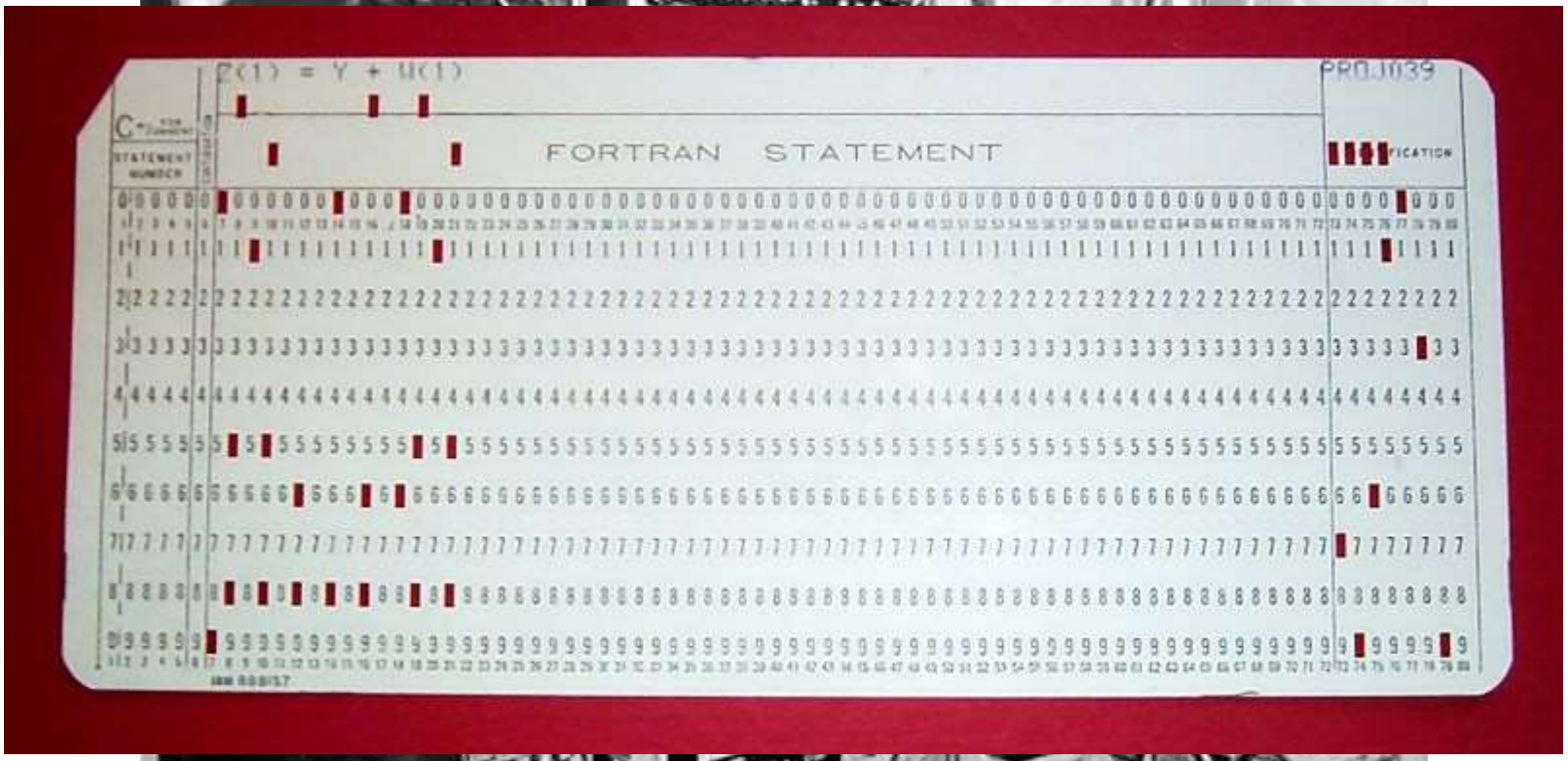
- Each program is called a “job”
- Early computers required significant setup time
- Each “job” required its own setup
- Operating Systems began as systems for simplifying setup and transitions between jobs

3.1 History of Operating Systems

- Batch processing (job queue)
- Interactive processing (real time)
- Time-sharing (one machine, many users)
- Multitasking (one user, many tasks)
- Multiprocessor machines (load balancing)
- Embedded Systems (specific devices)

The birth of OS

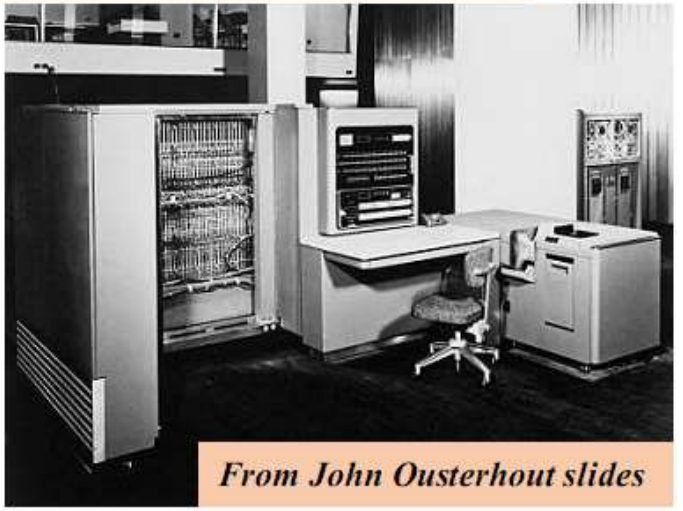
- Before OS was invented



Wikipedia

Early Systems - Bare Machine (1950s)

Hardware – *expensive* ; Human – *cheap*



From John Ousterhout slides

- Structure
  - Large machines run from console
  - Single user system
    - Programmer/User as operator
  - Paper tape or punched cards
- Early software
  - Assemblers, compilers, linkers, loaders, device drivers, libraries of common subroutines.
- Secure execution
- Inefficient use of expensive resources
  - Low CPU utilization, high setup time.

Jobs

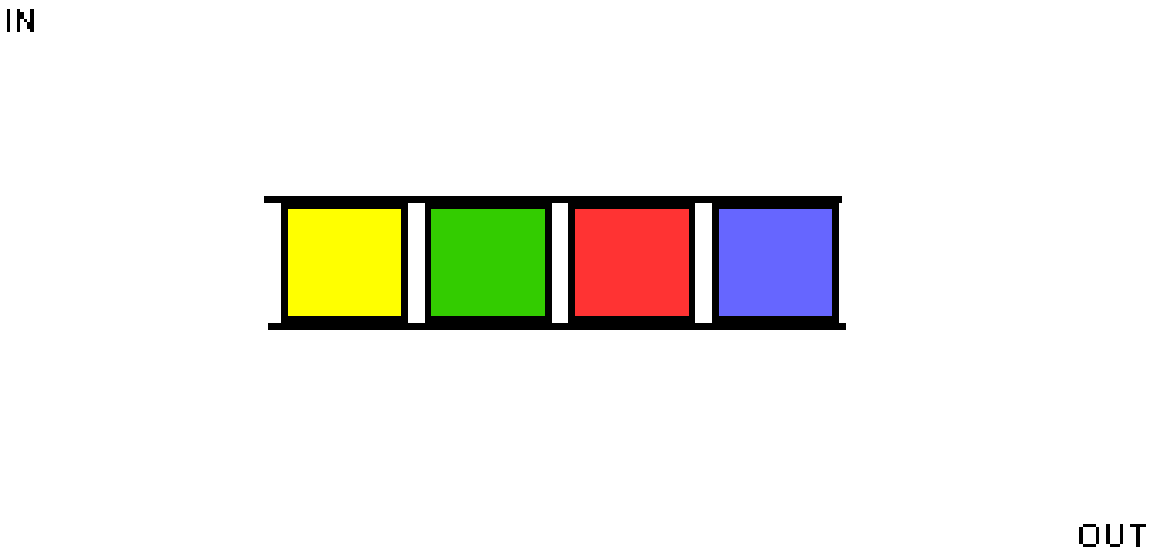
- Jobs queue

Significant set-up time to run programs -> Need for Batch systems

http://getlevelten.com/blog/ian-whitcomb/whats-wrong-project-application-queue

Queue

- FIFO

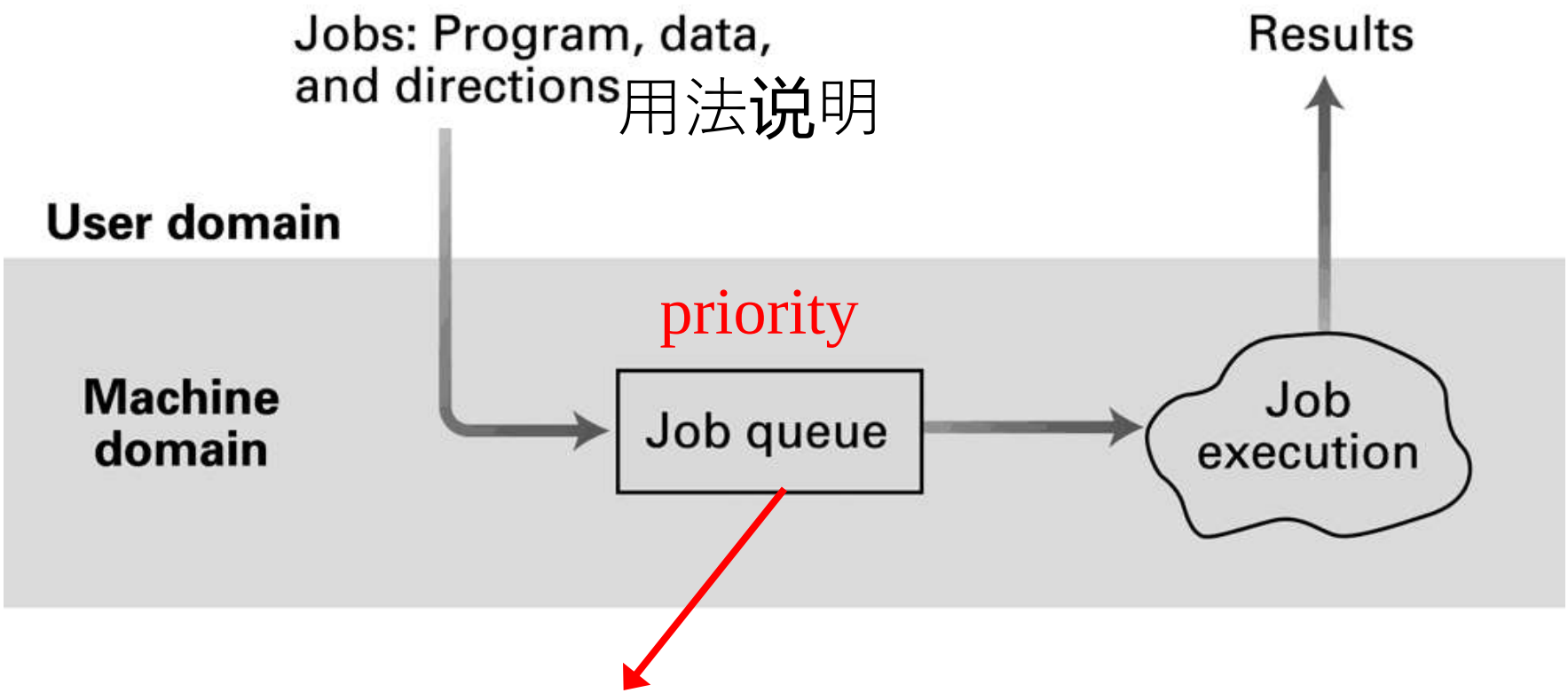


- 用户在计算机上的所有工作都要人工干预，如程序的装入、运行、结果的输出等。
- 随着计算机硬件的发展，人机矛盾（速度和资源利用）越来越大，必须寻求新的解决办法。
- 手工操作阶段有两个突出的缺点：
  - ①用户独占全机，虽然不会出现因资源已被其他用户占用而等待的现象，但资源利用率低。
  - ②CPU等待手工操作，CPU的利用不充分。 唯一的解决办法就是用高速的机器代替相对较慢的手工操作来对作业进行控制。

Batch processing批处理

操作系统开始出现

job queue



FIFO: first in first out

Batch Systems (1960's)

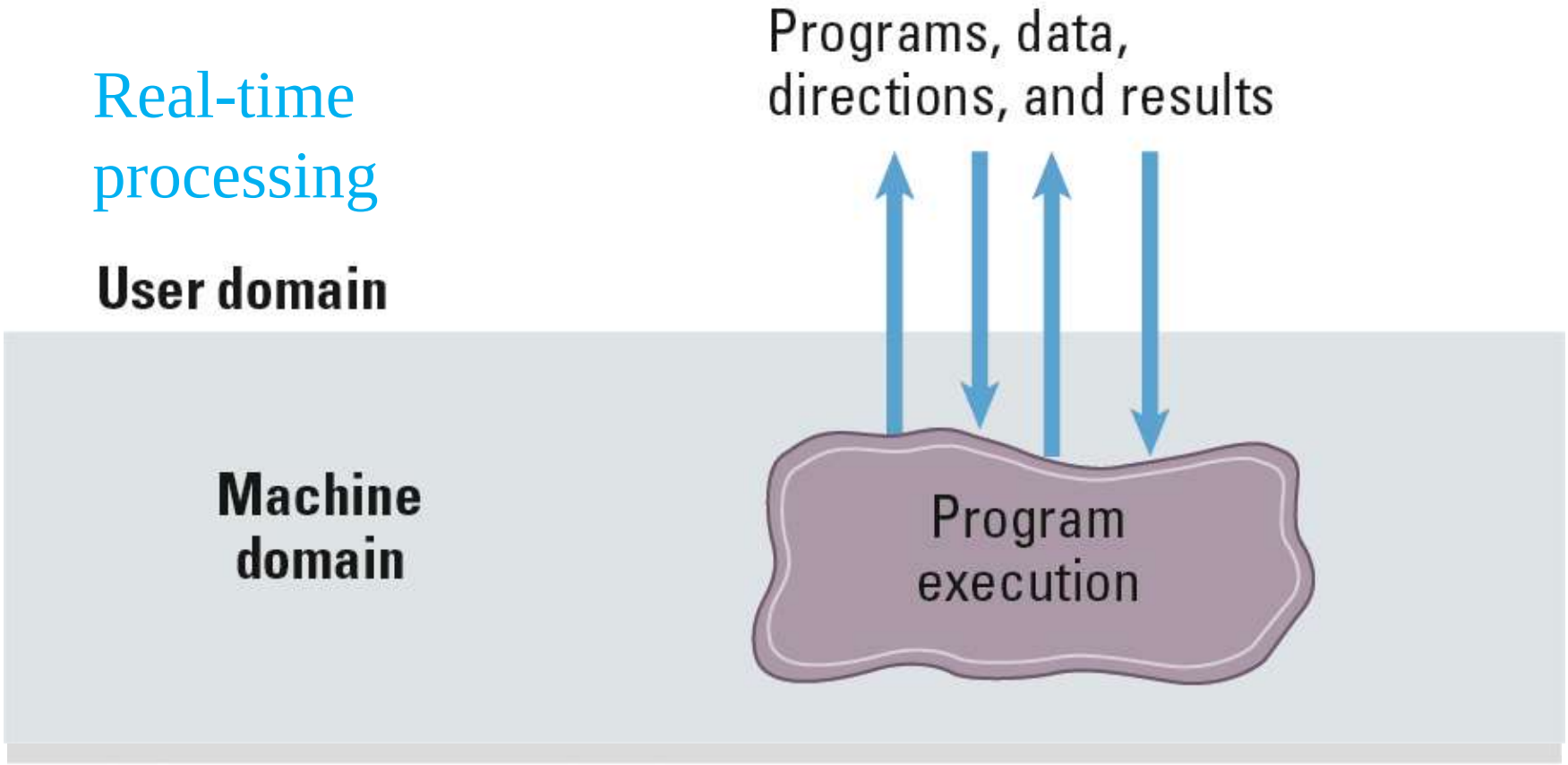
- Reduce setup time by batching jobs with similar requirements.
- Hire an operator
  - User is NOT the operator
- Automatic job sequencing
  - Forms a rudimentary OS.
  - Resident Monitor
    - Holds initial control, control transfers to job and then back to monitor.
  - Problem
    - Need to distinguish job from job and data from program.
    - Special cards indicate what to do.
    - User program prevented from performing I/O

Computer terminal

Wikipedia

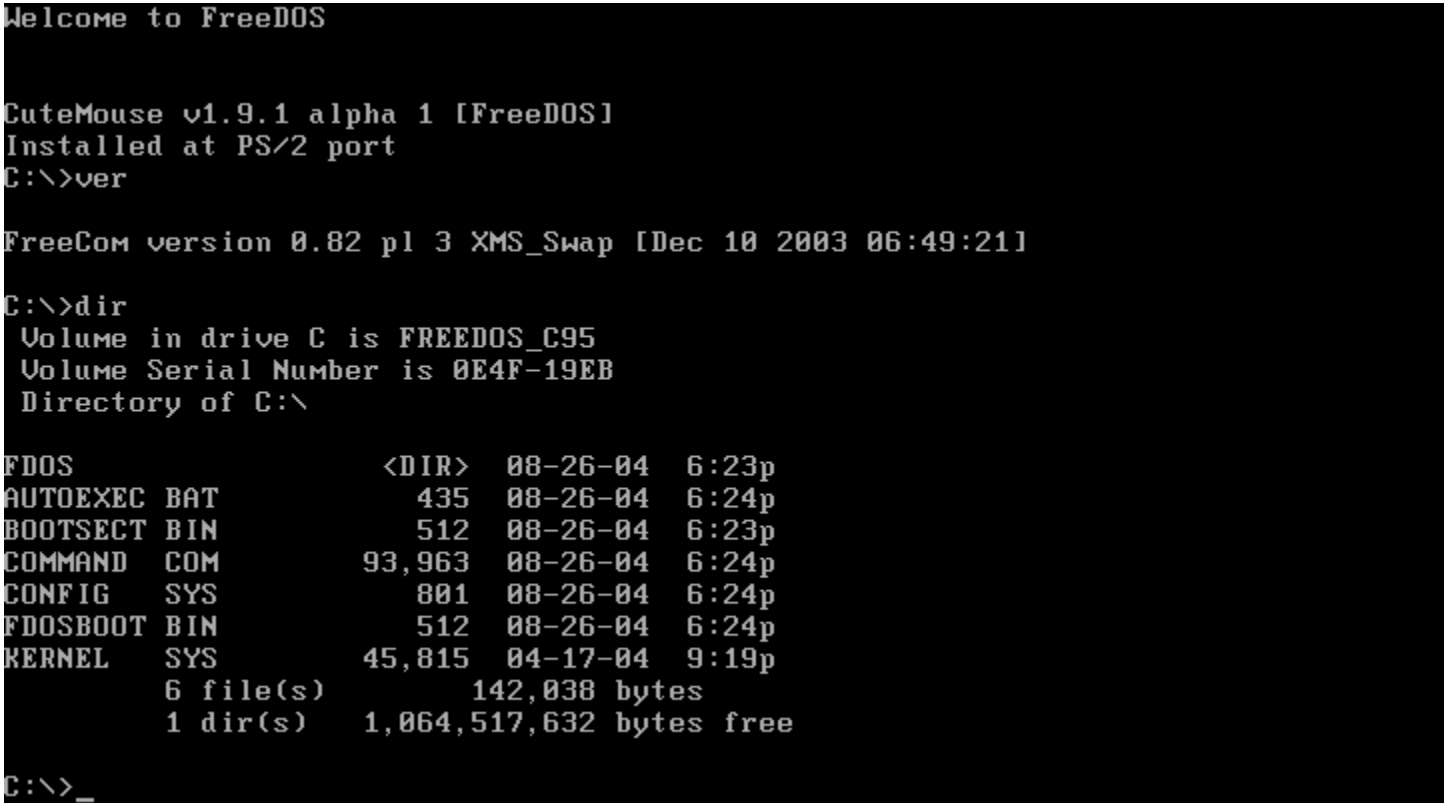
Interactive processing real time

Real-time processing



Text editing, music/movie playing, …

DOS (Disk Operation System)

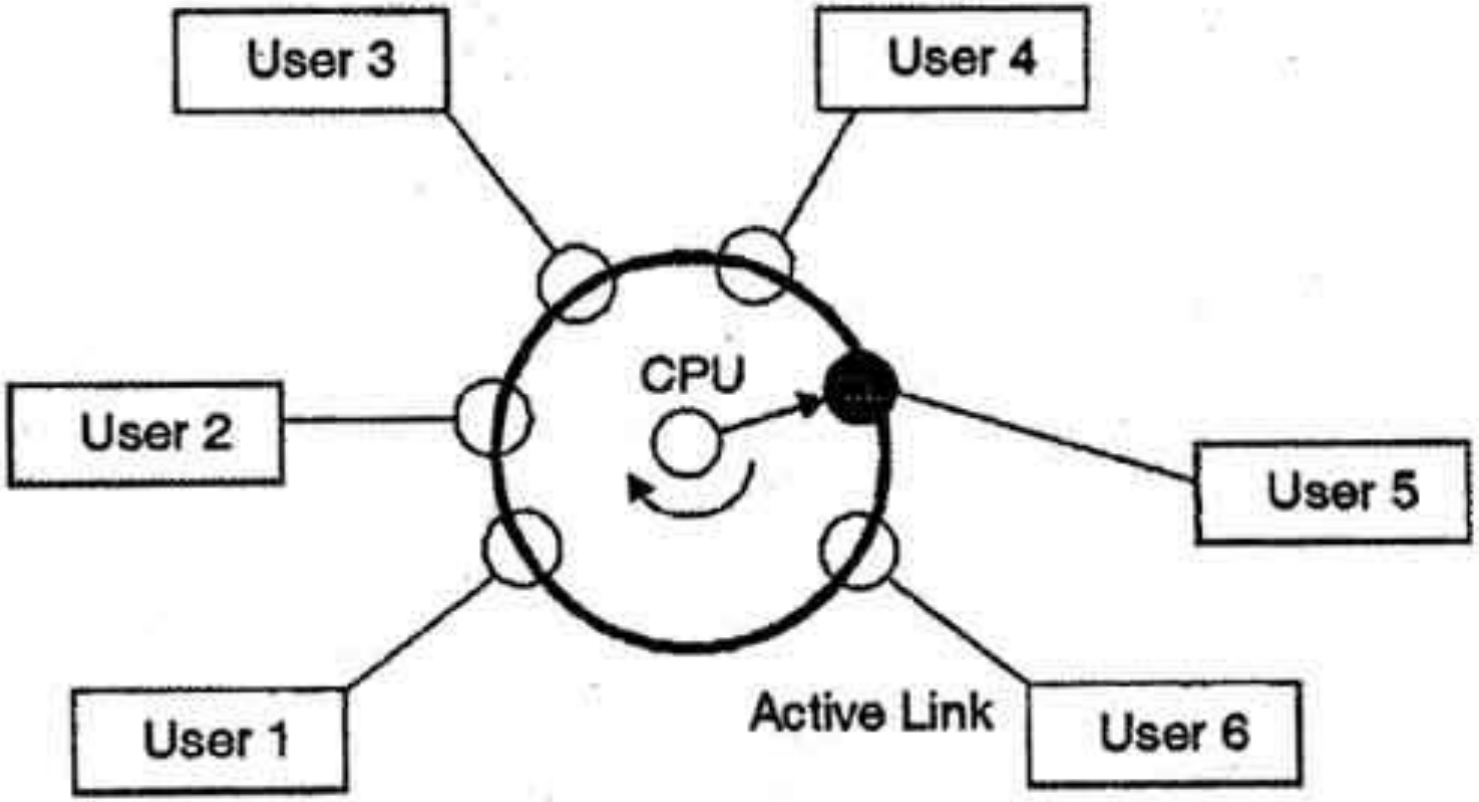


Time-sharing system （分时系统）

one machine, many users

- CPU switches between jobs frequently to allow user interaction

多个用户共享使用一个计算机



DINESH THAKUR

Multi-tasking one user, many tasks

一个用户同时执行多个任务

Multiprogramming

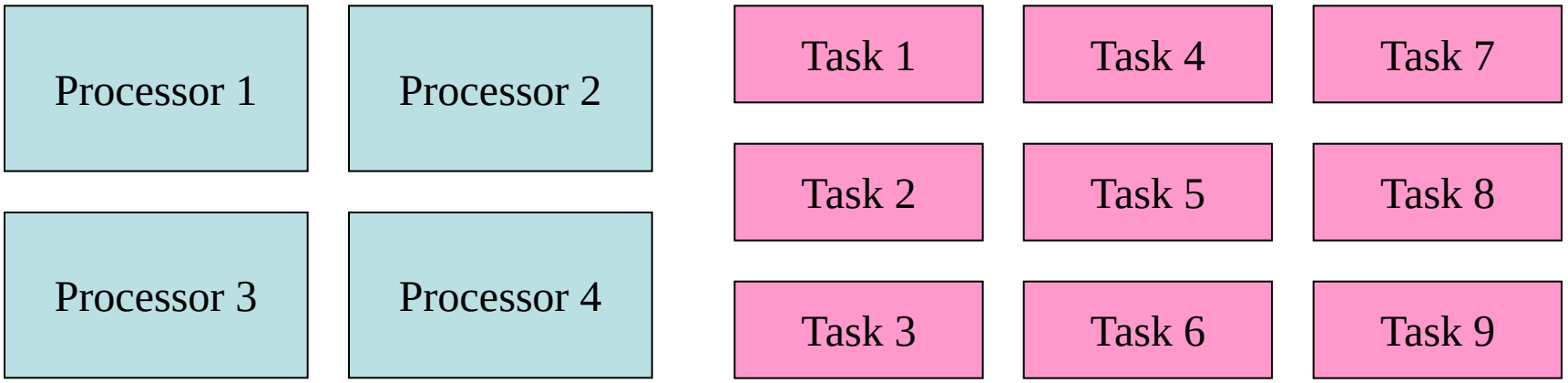
- Multiprogramming（多道程序）techniques are used in single-user as well as multiuser systems
- For single-user, multiprogramming is called multitasking (多任务)
- Multitasking: 一个用户同时执行多个任务
- Time-sharing:多个用户共享使用一个计算机

- 多道程序系统是在计算机内存中同时存放几道相互独立的程序，使它们在管理程序控制之下，相互穿插的运行。 两个或两个以上程序在计算机系统中同处于开始和结束之间的状态。这就称为多道程序技术运行的特征：多道、宏观上并行、微观上串行。各道程序轮流使用CPU，交替执行，现代计算机系统都采用了多道程序设计技术。

- 分时操作系统是使一台计算机同时为几个、几十个甚至几百个用户服务的一种操作系统。把计算机与许多终端用户连接起来，分时操作系统将系统处理机时间与内存空间按一定的时间间隔，轮流地切换给各终端用户的程序使用。由于时间间隔很短，每个用户的感觉就像他独占计算机一样。

- 分时操作系统是给不同用户提供程序的使用，而多道程序系统则是不同程序间的穿插运行。总之，分时操作系统主要是针对于多用户来说的，而多道程序系统主要是针对于多程序来说的

Multiprocessor Machines load balancing



- How to assign tasks to processors?
  - Load balance problem
- How to use processors to handle one task?
  - Parallelization, scaling problem

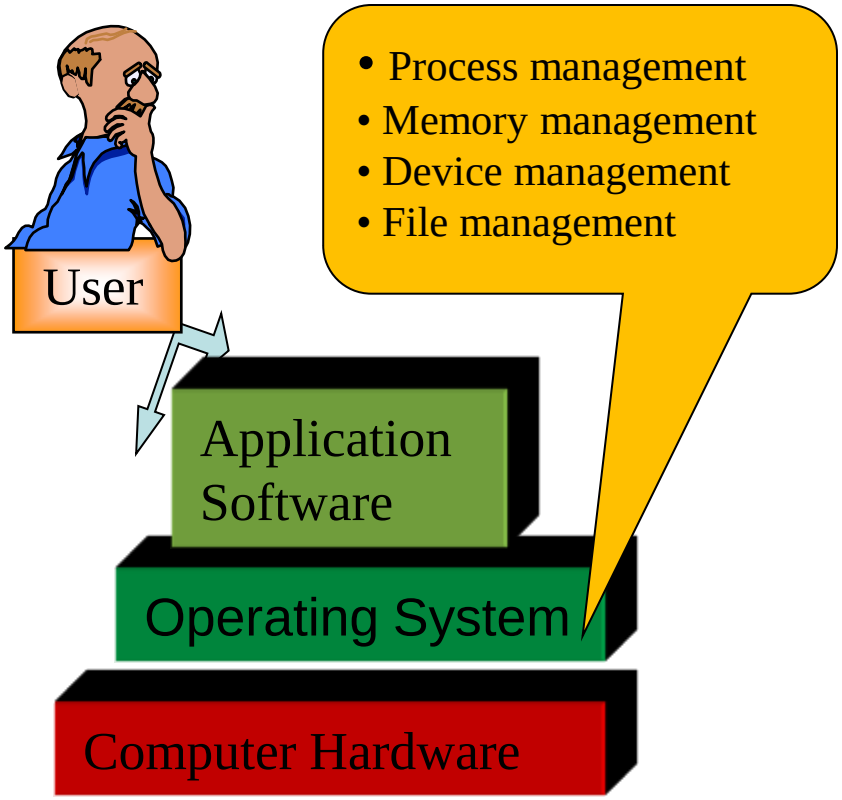
Computer network OS

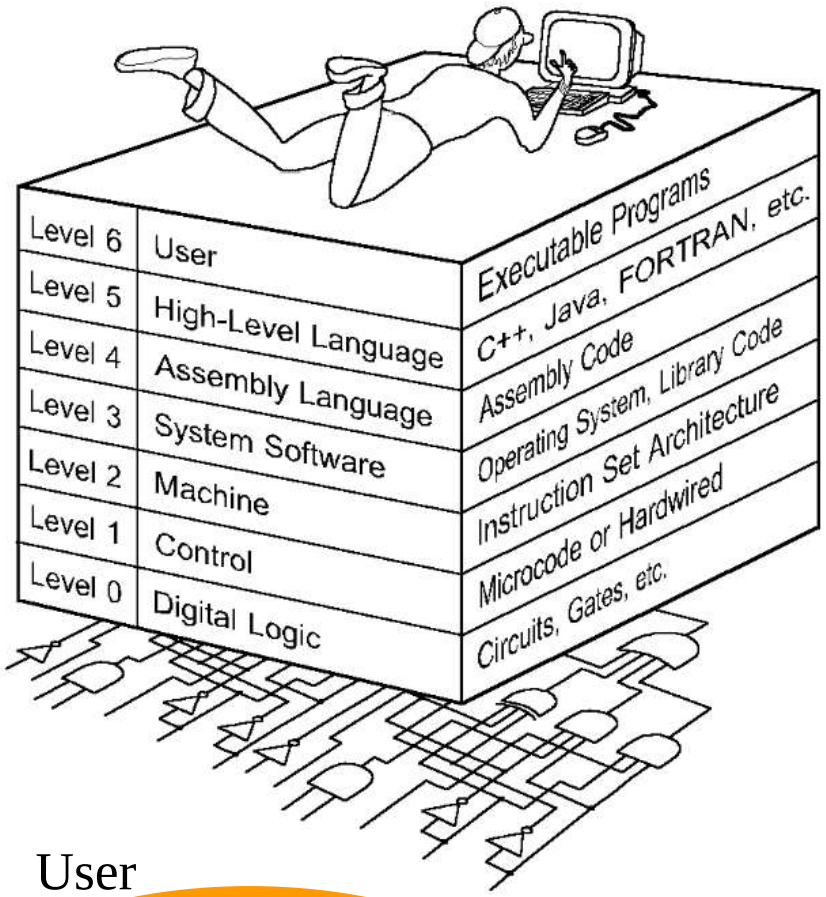
- 网络操作系统把计算机网络中的各台计算机有机地结合起来，提供一种统一、经济而有效的使用各台计算机的方法，实现各台计算机之间数据的互相传送。
- 网络操作系统最主要的特点是网络中各种资源的共享及各台计算机之间的通信。

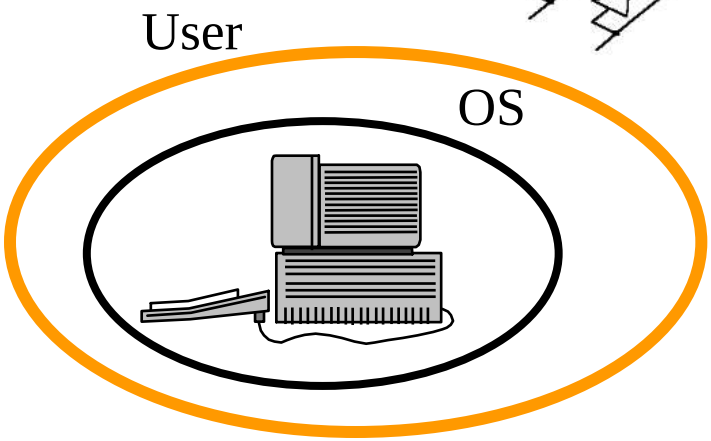
OS for Embedded Systems

- Embedded systems, PDA, mp3 player, cell phone, GPS,…
  - Limited storage, limited power,
  - Usually has real time requirement
- Turn key system: store all programs and data in a persistent memory
  - No BIOS and program loader

Microwave

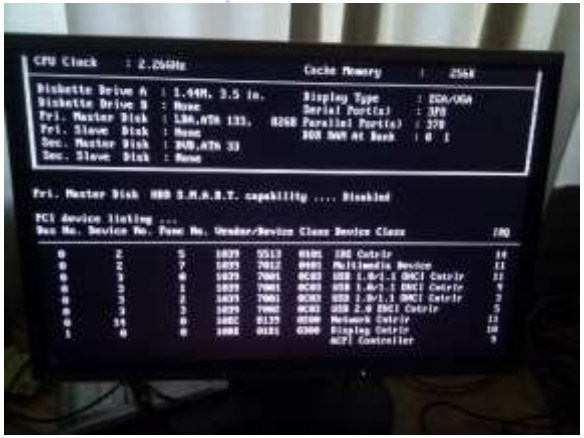






How does a computer start executing?

计算机启动过程



1. **电源开启**：用户按下电源按钮，电源开始向计算机的各个组件供电。
2. **电源自检（Power-On Self Test, POST）**：计算机启动时，BIOS（Basic Input/Output System）或UEFI（Unified Extensible Firmware Interface）固件会执行一系列的自检程序，检查硬件是否正常工作。
3. **BIOS/UEFI设置**：自检完成后，BIOS/UEFI会加载其设置，这些设置包括启动顺序、系统时间、硬件配置等。
4. **启动顺序**：根据BIOS/UEFI设置中的启动顺序，计算机会尝试从不同的设备（如硬盘、SSD、USB设备、光盘驱动器等）启动。
5. **加载引导扇区**：计算机找到启动设备后，会读取该设备的引导扇区（Boot Sector），引导扇区包含启动操作系统所需的代码。
6. **操作系统加载**：引导扇区代码会加载操作系统的内核和必要的系统文件。
7. **初始化操作系统**：操作系统内核开始运行，进行系统初始化，包括加载驱动程序、启动系统服务和应用程序。
8. **用户界面**：操作系统加载用户界面，如Windows的桌面或macOS的Dock，等待用户输入。
9. **用户登录**：如果操作系统需要，用户会输入用户名和密码进行登录。
10. **启动完成**：用户登录后，操作系统完全启动，用户可以开始使用计算机。

这个过程可能会因为不同的硬件配置、操作系统或BIOS/UEFI版本而略有不同。

BIOS（Basic Input/Output System）和UEFI（Unified Extensible Firmware Interface）都是计算机启动时运行的固件

1. BIOS：

- **存储位置**：传统的BIOS通常存储在主板上的一个芯片中，这个芯片通常是ROM（只读存储器）、EPROM（可擦写可编程只读存储器）或者EEPROM（电可擦除可编程只读存储器）。
- **存储类型**：由于BIOS需要在系统启动时被读取，而这些操作通常在断电状态下也能进行，所以它被存储在非易失性存储器中。

2. UEFI：

- **存储位置**：UEFI固件通常存储在主板上的一个专用芯片中，这个芯片可以是Flash内存，具体来说，通常是NOR Flash。与BIOS类似，UEFI也被存储在非易失性存储器中，以便在断电时保持固件的完整性。
- **存储类型**：UEFI固件存储在Flash存储器中，这种存储器允许固件被更新或刷新，因为UEFI支持固件的更新，以获得新功能或安全更新。

BIOS（Basic Input/Output System）和UEFI（Unified Extensible Firmware Interface）的区别：

1. **设计年代**：
   - **BIOS**：最早于1980年代设计，用于传统计算机的启动和硬件管理。
   - **UEFI**：设计于1990年代末，为了解决BIOS的一些限制，如只能使用16位代码和640KB内存限制。

2. **启动方式**：
   - **BIOS**：使用MBR（Master Boot Record）分区方案，最大只支持2.2TB的硬盘。
   - **UEFI**：使用GPT（GUID Partition Table）分区方案，支持更大的硬盘容量。

3. **安全性**：
   - **BIOS**：安全性较低，容易受到恶意软件的攻击。
   - **UEFI**：提供了安全启动（Secure Boot）功能，可以防止未授权的操作系统或启动程序加载。

4. **用户界面**：
   - **BIOS**：通常有基于文本的界面，操作较为复杂。
   - **UEFI**：提供了图形用户界面（GUI），操作更直观、更易于使用。

5. 速度:

- **BIOS**：启动速度较慢，因为它需要在启动过程中执行更多的自检。
- **UEFI**：启动速度通常更快，因为它可以更快地执行自检和启动过程。

6. 兼容性:

- **BIOS**：与旧的硬件和操作系统兼容。
- **UEFI**：设计时考虑了向后兼容性，但某些旧的硬件或软件可能不完全兼容。

7. 可扩展性:

- **BIOS**：由于设计较早，可扩展性有限。
- **UEFI**：设计时就考虑了可扩展性，可以更容易地添加新的功能和驱动程序。

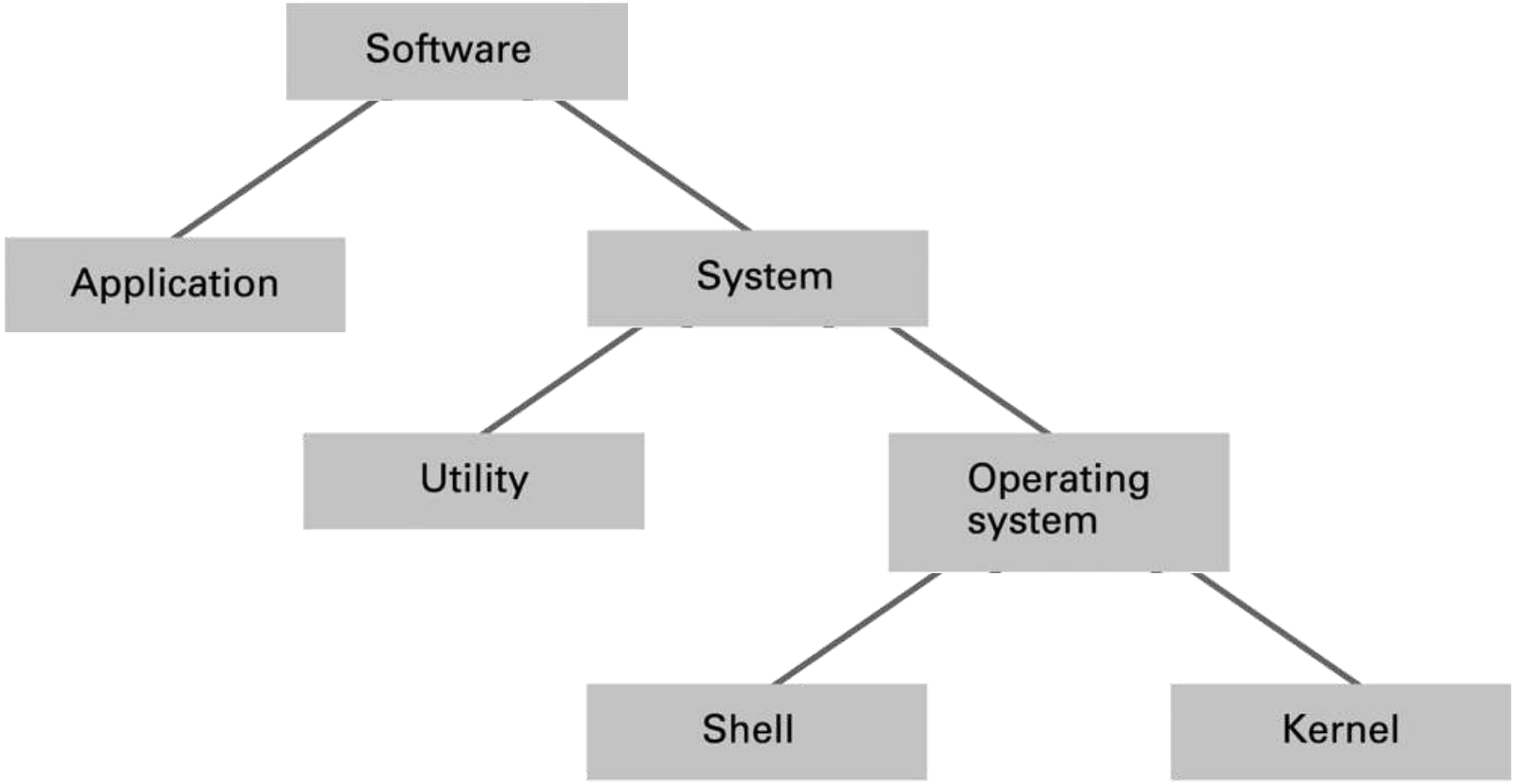
8. 内存访问:

- **BIOS**：只能使用16位代码，限制了对内存的访问。
- **UEFI**：可以使用32位或64位代码，可以更有效地访问更大的内存。

9. 操作系统支持:

- **BIOS**：主要与旧的操作系统兼容。
- **UEFI**：与现代操作系统兼容，如Windows 8及以后的版本，以及最新的Linux发行版。

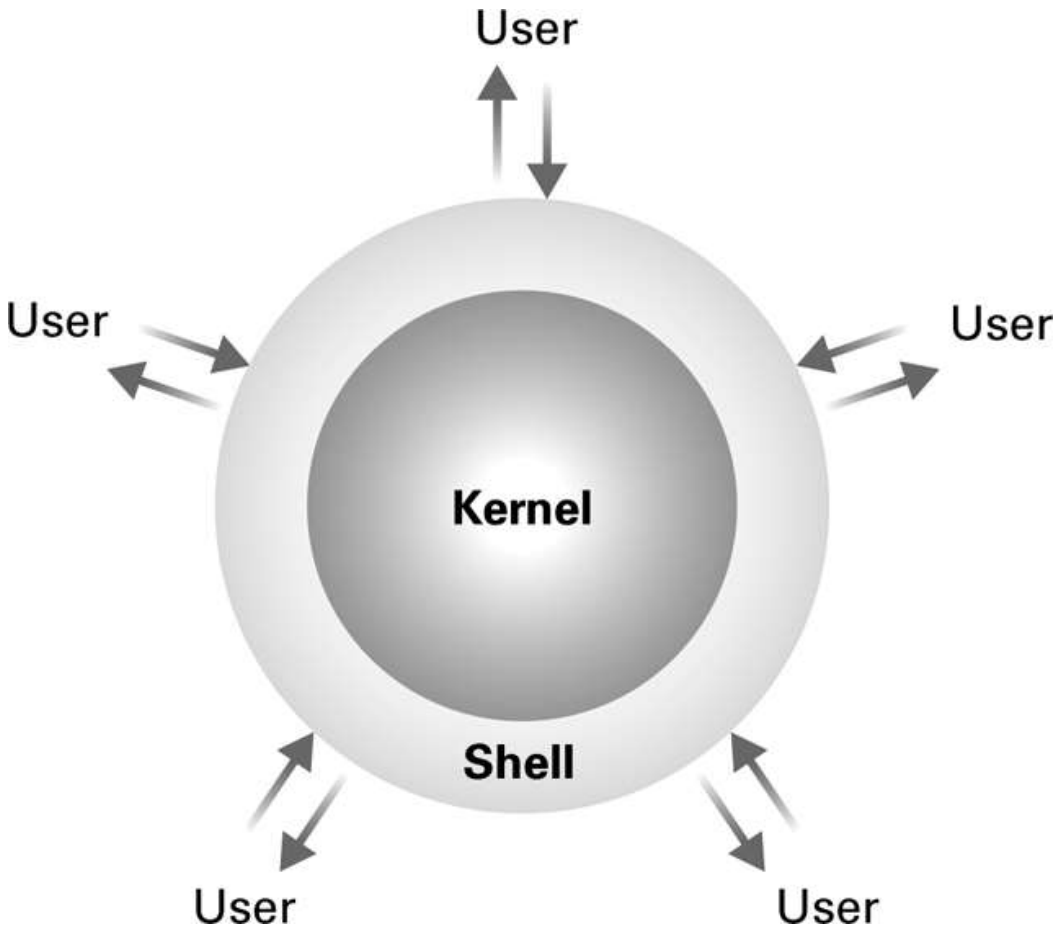
Software classification



Types of Software

- Application software
  - Performs specific tasks for users
- System software
  - Provides infrastructure for application software
  - Consists of operating system and utility software

Figure 3.4 The shell as an interface between users and the operating system



Operating System Components

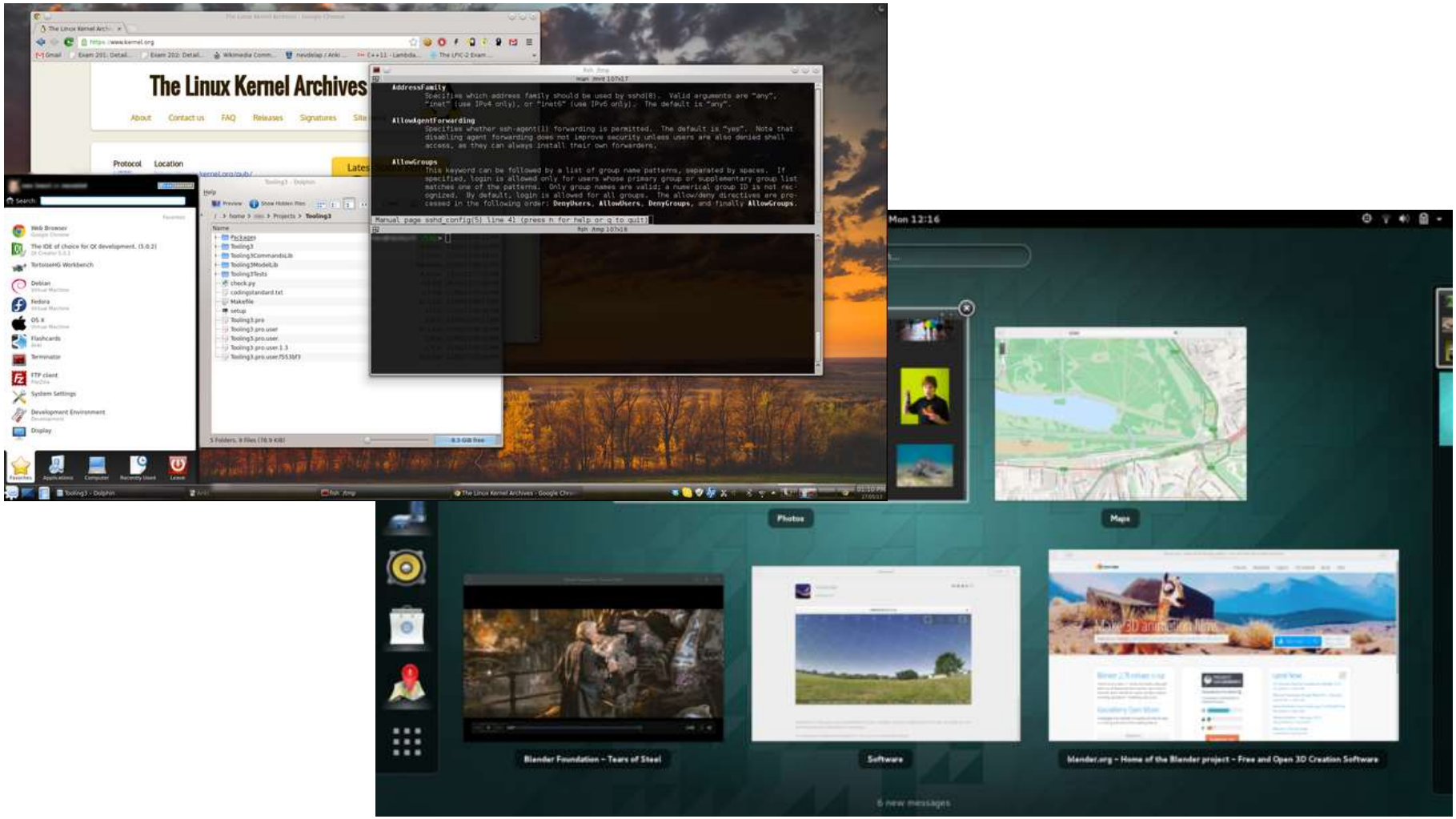
- **Shell:** Communicates with users
  - Text based
  - Graphical user interface (GUI)
- **Kernel:** Performs basic required functions
  - File manager
  - Device drivers
  - Memory manager
  - Scheduler and dispatcher

Shell: CLI and GUI

```
chealer@vinci:/usr/share/doc/bash$ export LC_ALL=C
chealer@vinci:/usr/share/doc/bash$ cd ~chealer/
chealer@vinci:~$ ls
Cloutier  Ido         Musique  logs      skolo  sources
Desktop   Mes images  boston   ncix.png  smb4k  vieux
chealer@vinci:~$ #Why is there color when calling ls without arguments?
chealer@vinci:~$ which ls
/bin/ls
chealer@vinci:~$ $(!!)
$(which ls)
Cloutier  Ido         Musique  logs      skolo  sources
Desktop   Mes images  boston   ncix.png  smb4k  vieux
chealer@vinci:~$ type ls #"ls" doesn't just run /bin/ls
ls is aliased to `ls --color=auto'
chealer@vinci:~$ echo $PS1
${debian_chroot:+($debian_chroot)}\u@\h:\w\$
chealer@vinci:~$ sh
sh-3.1$ echo $PS1
\s-\v\$
sh-3.1$ echo $BASH_VERSION
3.1.17(1)-release
sh-3.1$ ls
Cloutier  Ido         Musique  logs      skolo  sources
Desktop   Mes images  boston   ncix.png  smb4k  vieux
sh-3.1$ echo $SHELLOPTS # ls isn't an alias in POSIX mode
braceexpand:emacs:hashall:histexpand:history:interactive-comments:monitor:posix
sh-3.1$ kill
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill
-l [sigspec]
sh-3.1$ /bin/kill &> killerror # collect stdout and stderr of $ /bin/kill; in ki
llerror
sh-3.1$ wc -l !$
wc -l killerror
7 killerror
sh-3.1$ type kill # kill doesn't just run /bin/kill, even in POSIX mode.
kill is a shell builtin
sh-3.1$ !$ -n 9 $$ # OK, kill self
kill -n 9 $$ # OK, kill self
Killed
chealer@vinci:~$
```

KDE and GNOME

Unix 和Linux 的桌面环境系统



Components of OS

- For **user**: shell, privilege control (security)
- For **data**: file manager
- For **hardware**: device manager, memory manager, and boot manager
- For **software**:
  - Where to store: file manger, registry
  - How to execute: scheduler, process manager

How about Procedure?

- OS needs to define a set of rules or working flows for users and hardware/ software developers.
  - For example, you need to *double click* an *icon* to open a program or a file.
  - Design a simple yet useful procedure for a complicated system is not an easy job.
  - This is for books like “How to use computers?” to talk about

Functions of operating system

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