Chapter 1: Operating-System Introduction

肖 卿 俊

办公室：江宁区无线谷6号楼226办公室

电邮：csqjxiao@seu.edu.cn
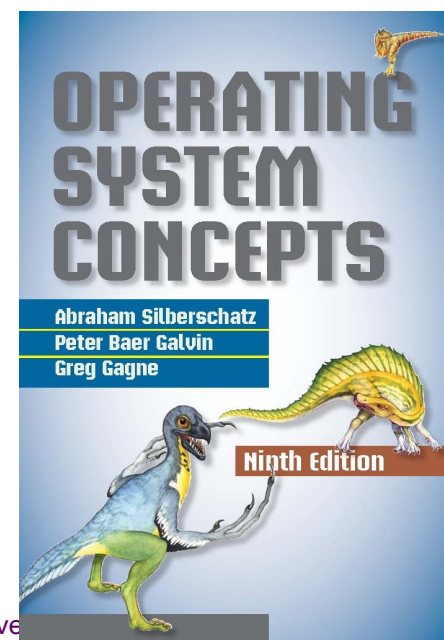
主页： https://csqjxiao.github.io/PersonalPage

电话：025-52091022

Course Summary

■ This course is to introduce the fundamental concepts that modern operating systems use to manage the physical computer. Students will understand the core of the operating system, known as the kernel.

◆ Prerequisites: C/C++, Computer Organization，计算机组成原理

◆ 学时：64学时理论课+16学时实验课

◆ Textbook：西尔伯莎茨编著的《操作系统概念》（第七版或第九版 影印版），高教出版社
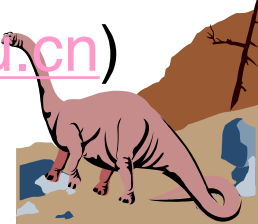
Course Requirements

- **Grading Policy (考核方式)**
  - ◆ 闭卷考试
  - ◆ 成绩评定:由下述三部分组成
    - ✓ 期末考试成绩
    - ✓ 期中考试成绩
    - ✓ 平时成绩（作业、出勤等）

- Course Schedule:
  - ◆ [1-16周] 周一(1-2) 教四-103，08:00~09:35（理论）
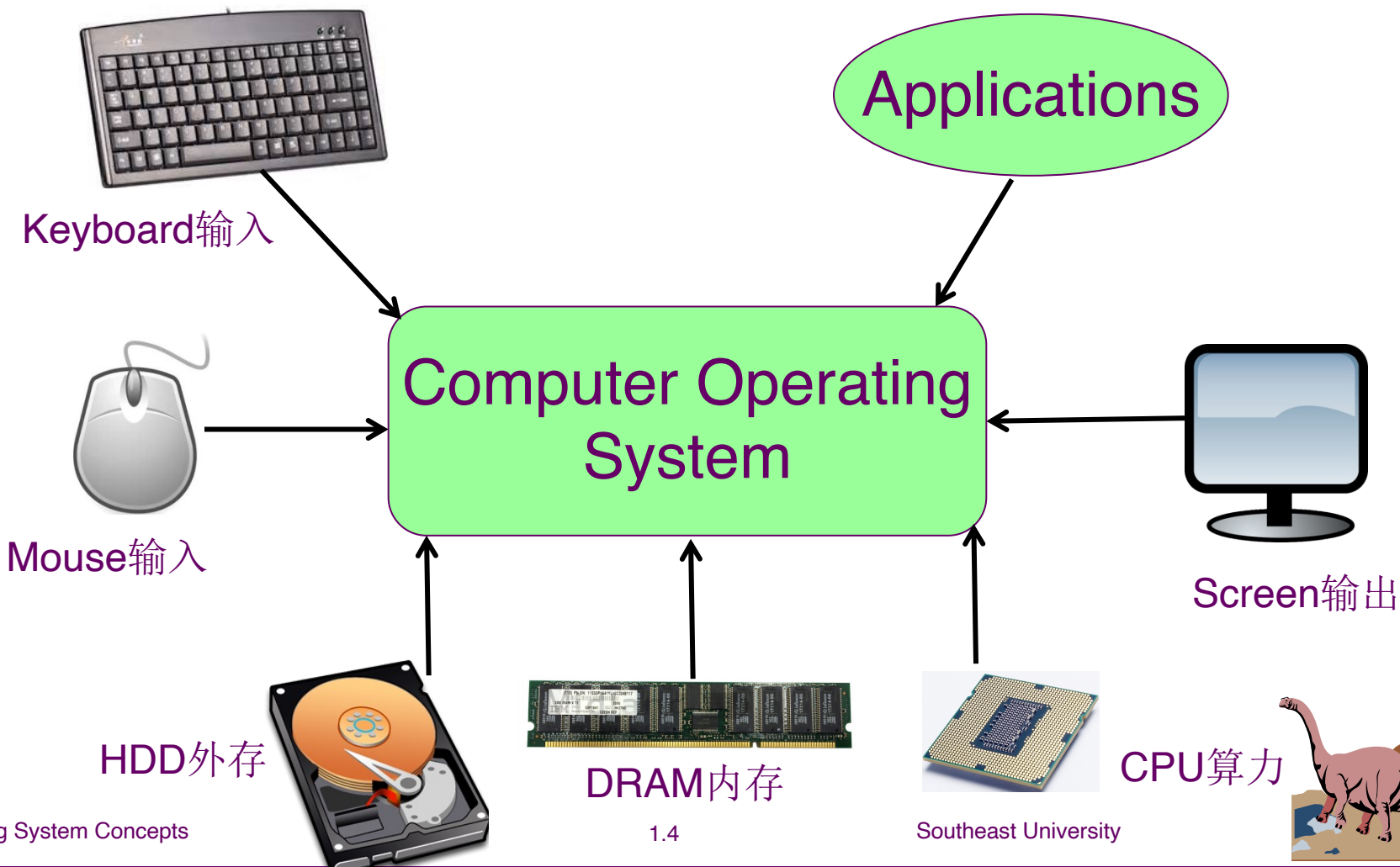   周三(3-4) 教四-103，09:50~12:15（理论+实验）

- Office Hours:
  - ◆ Tuesday: 12:00~13:00; Thursday:12:00~13:00
  - (E-mail discussions are welcome: csqjxiao@seu.edu.cn)

Course Outline (1/3)
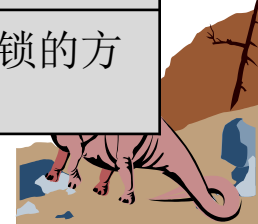
■ 操作系统是硬件系统上覆盖的第一层软件，帮助用户管理系统的计算、内存、外存等硬件资源

Keyboard输入

Applications

Mouse输入

Computer Operating System

Screen输出

HDD外存

DRAM内存

CPU算力

■ 操作系统是硬件系统上覆盖的第一层软件，帮助用户管理系统的计算、内存、外存等硬件资源

Chapter	教学要求
1. Introduction	掌握操作系统的一般情况, 了解本课程的学习方法
2. Operating System Structures	介绍操作系统的组成 使学生树立操作系统的整体概念 (a top-down view)
3. Process Management	操作系统的最重要的部分, 要求学生牢固掌握进程概念, 能够借助进程概念编写并发程序
4. Threads	要求学生掌握创建和管理线程的方法 熟练使用多线程技术编程
5. CPU Scheduling	要求学生理解并能编写一个进程调度的程序
6. Process Synchronization	要求学生理解同步概念, 掌握同步的编程方法
7. Deadlocks	要求学生理解死锁现象, 了解预防、避免、检测、解除死锁的方法

算力管理

Course Outline (3/3)

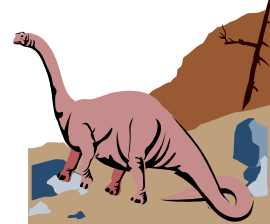| Chapter	教学要求
8. Memory Management	掌握固定分区,可变分区,段式,页式存储管理原理与算法
9. Virtual Memory	虚拟存储是操作系统另一个重要概念。要求学生正确理解虚存的概念。 熟练掌握请求页式的地址变换过程以及常用的页面置换算法
10. File Systems	掌握文件系统的基本概念，熟练使用文件系统的操作方法
11. I/O Systems	掌握I/O系统的基本概念，理解设备驱动程序。 弄清从用户请求到设备完成用户请求的全过程
12. Mass-Storage Structure	了解几种磁盘调度算法

内存管理

外存管理

Chapter 1: Introduction (Outline)

- What Operating Systems Do
- Computer-System Organization (Hardware)
- Computer-System Architecture (Hardware)
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security
- Distributed Systems
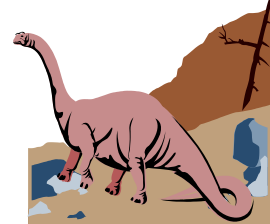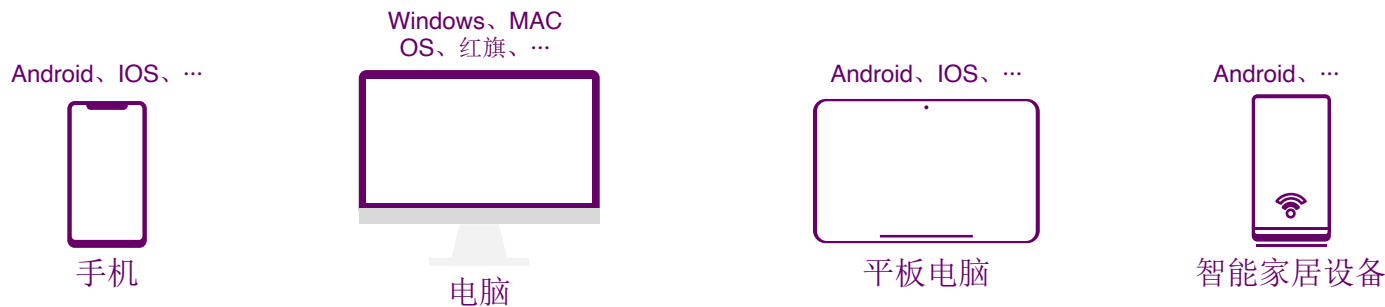- Special-Purpose Systems
- Computing Environments

Southeast University

常见的计算机硬件形态

操作系统介绍

- 操作系统是一组管理计算机硬件与软件资源的程序，同时也是计算机系统的内核与基石。

- 操作系统身负诸如管理与配置处理器、内存、外存等硬件资源，决定资源供需的优先次序、控制输入与输出设备、操作网络与管理文件系统等基本事务

- 操作系统也提供一个让使用者与系统交互的操作接口。小到智能穿戴设备、电视盒子，大到电脑、服务器、路由交换设备，都有自己的操作系统。

Android、IOS、…

手机

Windows、MAC
OS、红旗、…

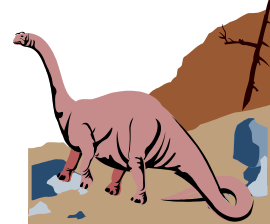电脑

Android、IOS、…

平板电脑

Android、…

智能家居设备

Question about OS

An operating system must manage system hardware resources. Which of the following do you think is an OS? (multiple choices)

- ☐ Firefox by Mozilla

- ☐ Mac OS by Apple Inc.

- ☐ Windows by Microsoft

- ☐ Android by Google

- ☐ Linux by open source community

The Powerful Concept of "Abstraction"

- Abstraction is a well-understood interface that hides all the details within a subsystem

- With a good abstraction, we can treat a subsystem as a "black box". We don't have to understand the inner details within the box

- Later we shall explain how the operating system works as a layer of abstraction within a computer system architecture
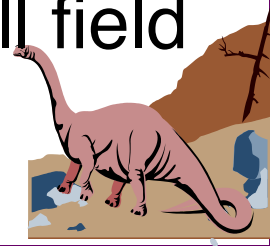
Question about Abstraction

Which of the following is **NOT** an abstraction that hides all the details? (multiple choices)

- ☐ Door knob
- ☐ Instruction set architecture
- ☐ Exact number of pins coming out of a chip
- ☐ And logic gates
- ☐ Transistor
- ☐ INT data type in a programming language, e.g., C
- ☐ Specific location of base pads in a baseball field

Question: Name as many abstractions between the two extremes

Google Earth

Applications

System Software (**OS**, compiler, etc.)

Instruction set architecture (SW/HW separation)

Computer organization (data path + control)

Sequential + Comb. Logic elements
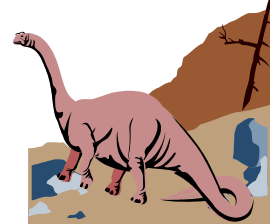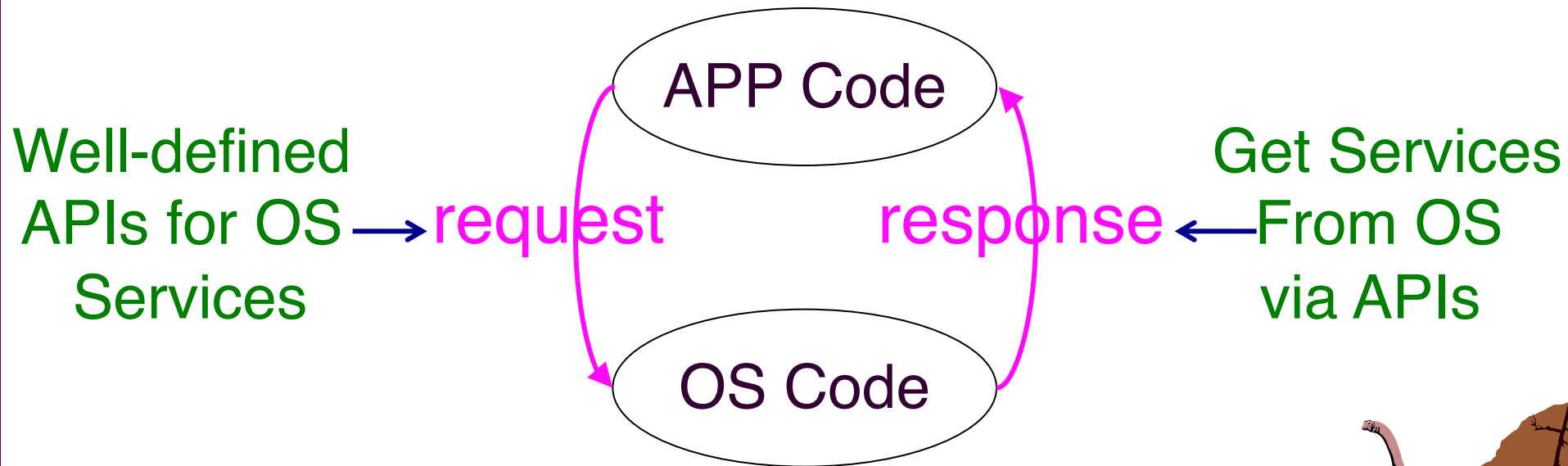
Logic gates (and, or, not, …)

Transistors

Series of Abstractions

Electrons + Holes
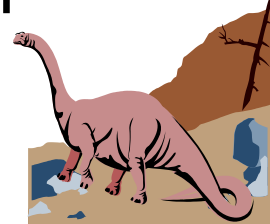
What is an Operating System?

■ OS, in a nutshell, provides
  ◆ Protected access to hardware resources
  ◆ Arbitrate among competing request

APP Code

Well-defined APIs for OS Services → request

response ← Get Services From OS via APIs

OS Code

What is an Operating System?

■ A program that acts as an intermediary between the application programs and the computer hardware.

  ◆ Execute user programs and make solving user problems easier

  ◆ Make the computer system convenient to use

  ◆ Use the computer hardware in an efficient way

■ From the computer's point of view:

  ◆ A resource allocator, a control program, or more concisely, a "kernel"

UNIX、Linux系统的发展

■操作系统内核历史上出现过大量的发展分支。目前生态好、大量使用的只有Windows、Linux、MacOS等。本课程主要学习Linux。

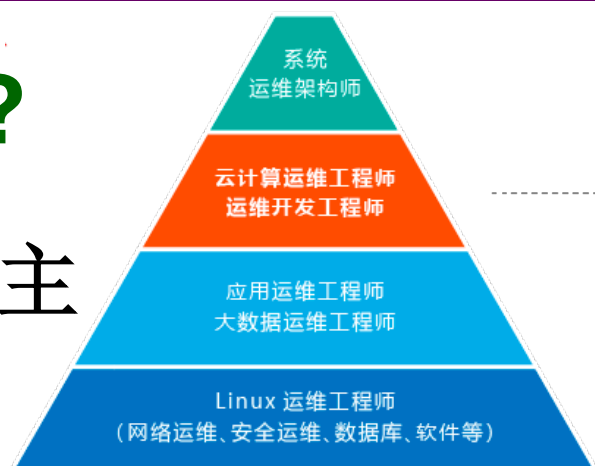|		
Fedora	deepin	RedHat
Kaili Linux	Oracle Linux	CentOS
DVL	Arch Linux	**Debian**
OpenSuse	中标麒麟	Ubuntu

Why Learn Linux?

■ **实验以Linux内核的操作系统为主**

■ **为什么要学习Linux和云计算？**

 ◆ **行业发展需求**：业界主流操作系统、软件开发环境、互联网的企业级应用都是基于Linux操作系统

 ◆ **企业人才需求**：**系统运维工程师**职业发展路线清晰，待遇优厚；我国Linux市场的就业岗位达百万

■ **未来的趋势：** 大数据、人工智能、5G等新兴产业都需要云计算所提供的强大算力以及成熟的PaaS平台能力。因此Linux与云计算等新技术的结合将会是未来传统运维工程师进阶的重要方向。
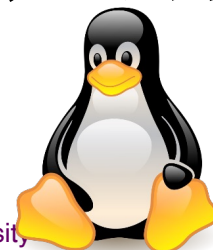
参考文献：为什么要学习Linux和云计算
https://edu.sdnlab.com/linux-cloud-train

Linux操作系统的组成

■ Linux操作系统构成
  ◆ Linux内核
  ◆ 应用程序

■ Linux内核项目介绍
  ◆ 主要作者：Linus Torvalds（芬兰）。
  ◆ 1991年10月，发布Linux 0.02版（第一个公开版）
  ◆ 1994年3月，Linux 1.0版发布。
  ◆ Linux内核的标志——企鹅Tux，取自芬兰的吉祥物

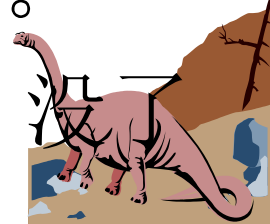# 不同Linux发行版：CentOS (Community Enterprise OS)简介

- 来自于Red Hat Enterprise Linux依照开放源代码规定发行的源代码所编译而成。

- CentOS不包含封闭源代码软件。CentOS是RHEL源码再编译的产物，而且在RHEL的基础上修正了不少已知的Bug。

- 相对于其他Linux发行版，CentOS稳定性优秀。

- CentOS在2014年加入红帽后，继续坚持提供免费操作系统的使用和更新。目前CentOS操作系统的使用领域非常广泛，适合服务器使用。
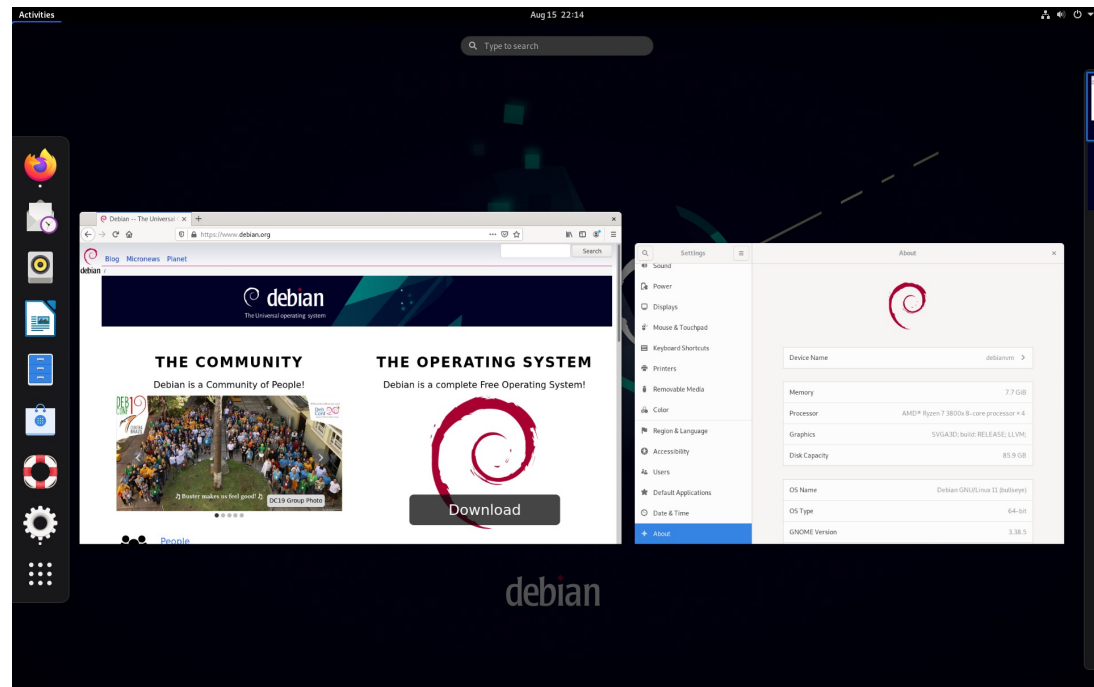
- CentOS8于2021年结束，"免费"RHEL 没了

不同**Linux**发行版：**Debian**简介

■ Debian是迄今为止最遵循GNU规范的Linux系统。提供了接近十万种不同的开源软件支持，具有认可度和使用率，对于各类内核架构支持性良好，稳定性、安全性强，更有免费的技术支持。
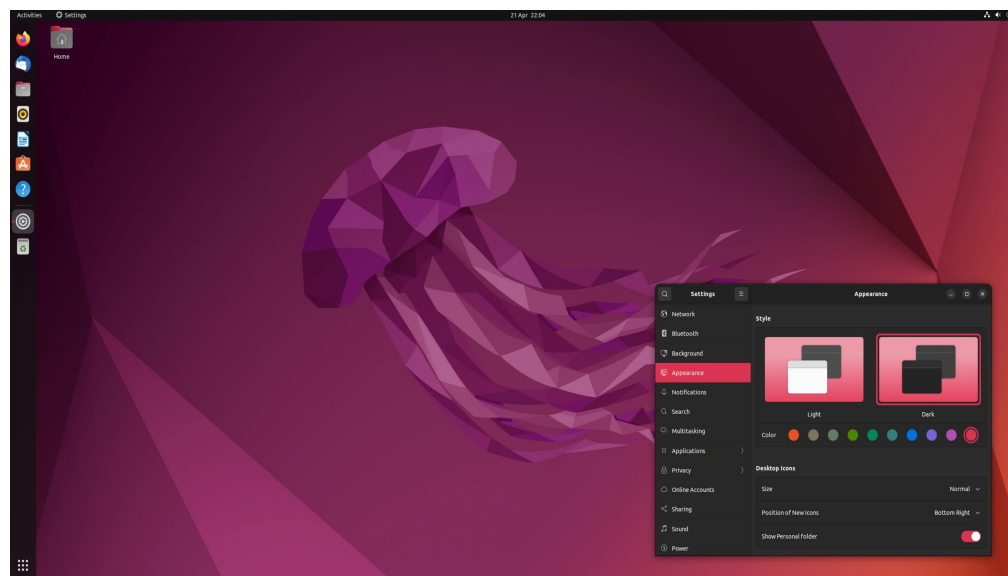
■ 基于其优秀的稳定性和资源开销小等特点，Debian在虚拟专用服务器中应用比较多。

不同Linux发行版：Ubuntu简介

- Ubuntu是Debian的一款衍生版，是一个以桌面应用为主的开源免费的操作系统

- Ubuntu有桌面版、服务版，在云计算、甚至一些运行Ubuntu Linux的移动设备上很常见

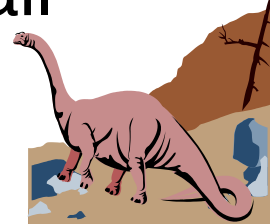- Ubuntu具有庞大的社区力量，用户可以方便地获得帮助。

- 我们的实验主要采用Ubuntu。
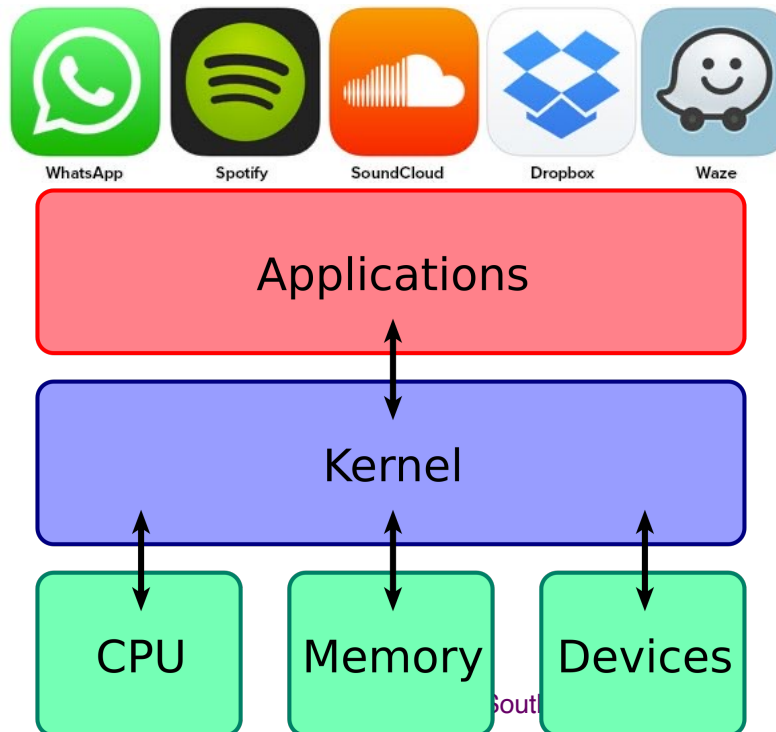
Question about OS functionalities

Please select the choices that apply to the functionalities provided by an OS

☐ OS is a resource manager

☐ OS provides a consistent interface to the hardware resource (CPU, memory, I/O, …)

☐ OS schedules applications on the CPU

☐ OS stores personal information such as credit card numbers, social security number, email addresses, and so on
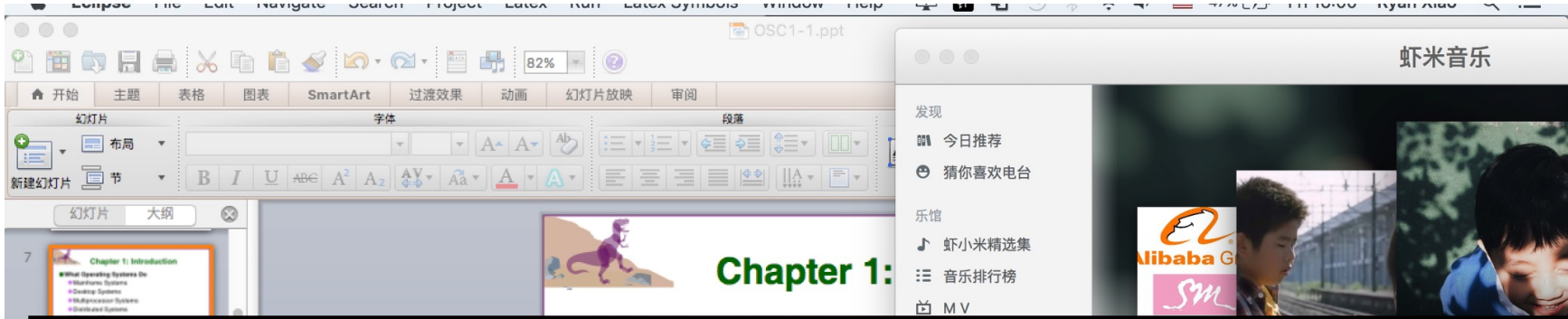
What does an Operating System Do?

■ An OS takes over physical resources, such as a CPU, memory, or disk, and virtualizes them. It handles tough and tricky issues related to concurrency. And it stores files persistently, thus making them safe over the long-term.

All because there are multi-programs running in a computer.

How does an operating system support the "simultaneous" running of multiple applications on a single device?
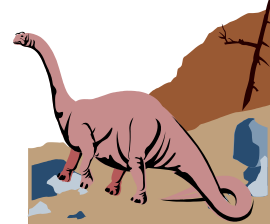
Virtualization

■ Many programs are run concurrently in a computer.
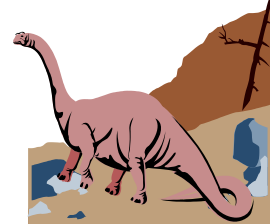
◆ Virtualization

■ What is virtualization?

◆ The OS takes a physical resource (such as the processor, or memory, or a disk) and transforms it into a more general, powerful, and easy-to-use virtual form of itself.

◆ It creates an illusion that each program exclusively owns the physical resource.

Virtualizing the CPU

```c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/time.h>
4 #include <assert.h>
5 #include "common.h"
6
7  int
8  main(int argc, char *argv[])
9  {
10     if (argc != 2) {
11         fprintf(stderr, "usage: cpu <string>\n");
12         exit(1);
13     }
14     char *str = argv[1];
15     while (1) {
16         Spin(1);      /* busy waiting for one second */
17         printf("%s\n", str);
18     }
19     return 0;
20 }
```

Virtualizing the CPU
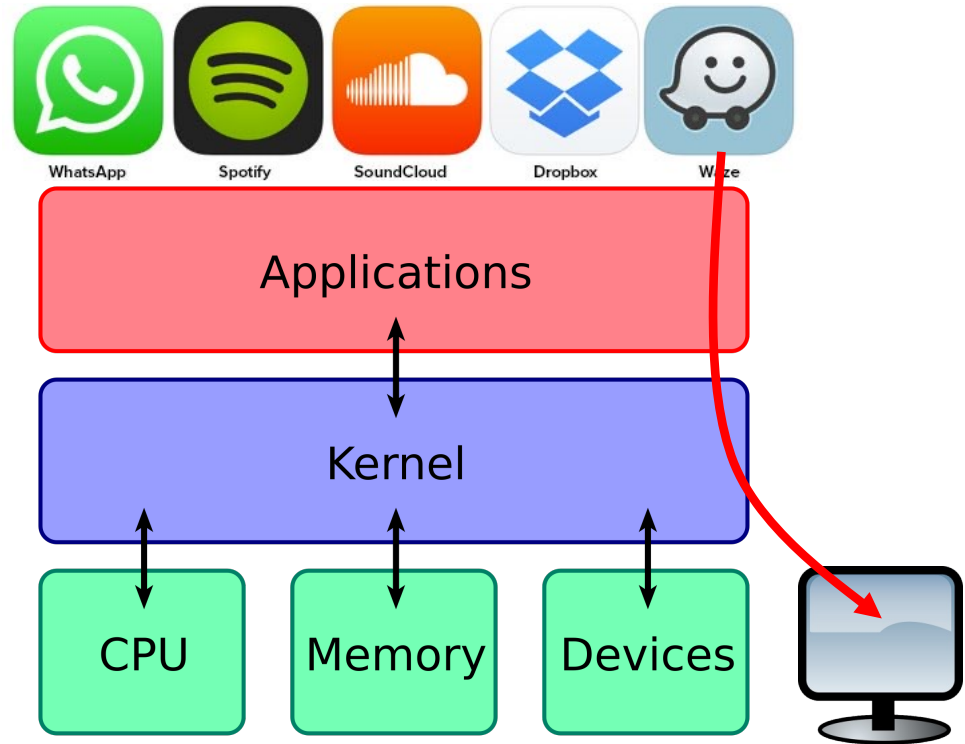
```
prompt> gcc -o cpu cpu.c -Wall
prompt> ./cpu A
A
A
A
A
^C
prompt>
```
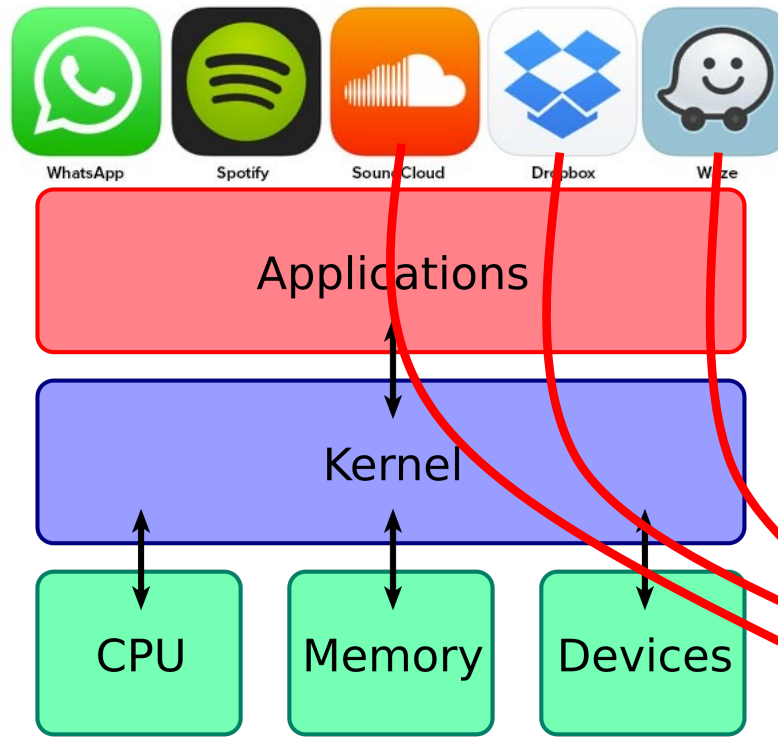
Applications

Kernel

CPU   Memory   Devices

Screen输出

Virtualizing the CPU

prompt> ./cpu A & ./cpu B & ./cpu C
& ./cpu D
[1] 7353
[2] 7354
[3] 7355
[4] 7356
A
B
D
C
A
B
D
C
A
C
B
D
...

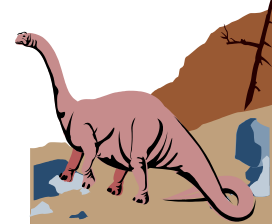It may appear that each program is running
with one dedicated processor core.
Reason: multi-processor? Not really.

Applications

Kernel

CPU   Memory   Devices

Screen输出

Virtualizing the Memory

```
1 #include <unistd.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "common.h"
5
6 int
7 main(int argc, char *argv[])
8 {
9      int *p = malloc(sizeof(int)); // a1
10     assert(p != NULL);
11     printf("(%d) memory address of p: %08x\n", getpid(), (unsigned) p); // a2
12     *p = atoi(argv[1]);      // assign value to addr stored in p
13     while (1) {
14         Spin(1);
15         *p = *p + 1;
16         printf("(%d) p: %d\n", getpid(), *p); // a4
17     }
18     return 0;
19 }
```

Virtualizing the Memory

```
prompt> ./mem 0
(2134) memory address of p: 00200000
(2134) p: 1
(2134) p: 2
(2134) p: 3
(2134) p: 4
(2134) p: 5
^C
```
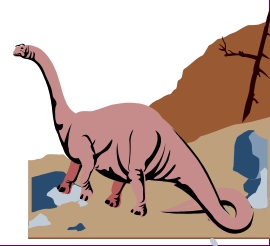
Virtualizing the Memory

```
prompt> ./mem 0 & ./mem 1
[1] 21846
[2] 21847
(pid:21846) addr of p:        7fff5fbffab8
(pid:21846) addr stored in p: 100100350
(pid:21847) addr of p:        7fff5fbffab8
(pid:21847) addr stored in p: 100100350
(pid:21847) value of p: 2
(pid:21846) value of p: 1
(pid:21847) value of p: 3
(pid:21846) value of p: 2
(pid:21847) value of p: 4
(pid:21846) value of p: 3
(pid:21846) value of p: 4
(pid:21847) value of p: 5
(pid:21846) value of p: 5
(pid:21847) value of p: 6
(pid:21846) value of p: 6
(pid:21847) value of p: 7
```
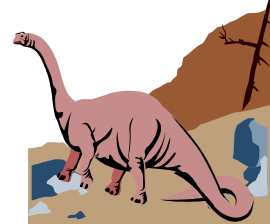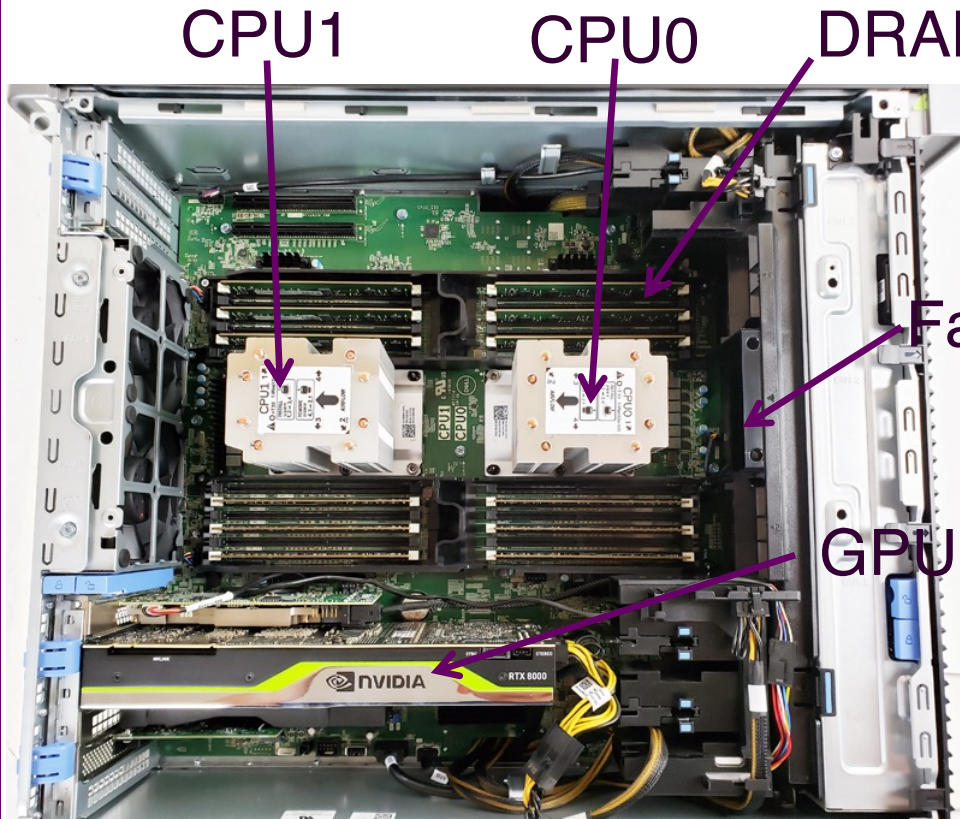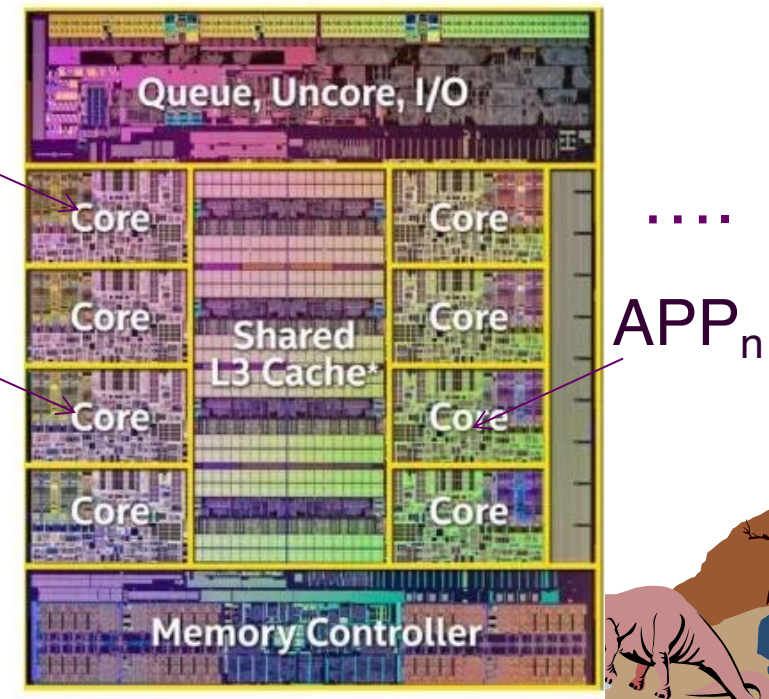
Program Concurrency

■ Computer seemingly runs several program in parallel: email, browser, music player, …

■ One machine can have multiple CPUs, and one CPU can have multiple cores.

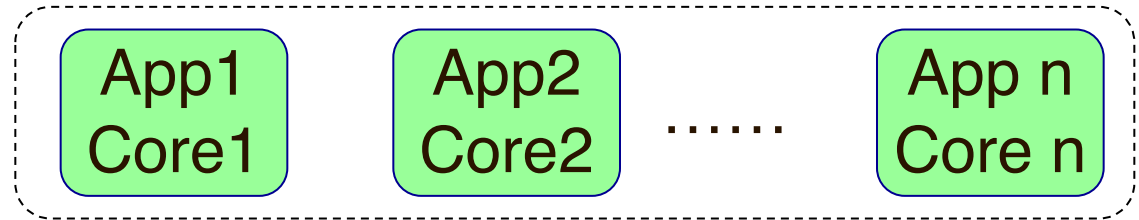CPU1 CPU0 DRAM
Fans
GPU

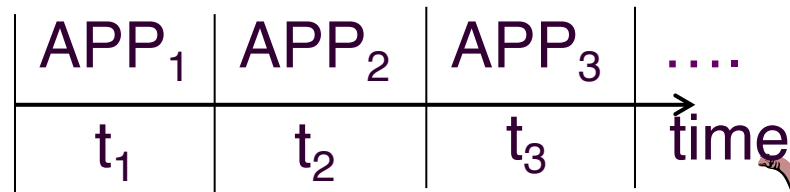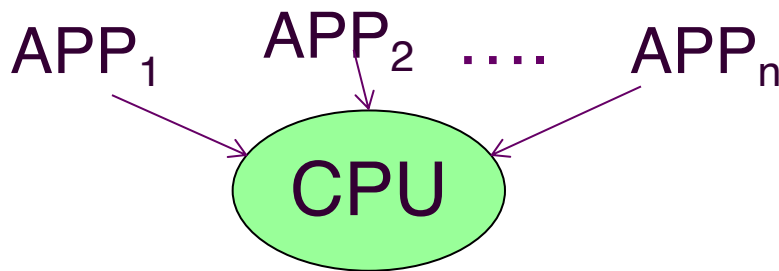APP$_1$
APP$_2$
APP$_n$
….

英特尔CPU内核图

Queue, Uncore, I/O
Core
Core
Core
Core
Shared L3 Cache*
Core
Core
Core
Memory Controller

Program Concurrency

■ Suppose there is only one CPU chip. How is it possible?

☐ Multiple cores in a CPU

| App1 Core1 | App2 Core2 | ...... | App n Core n |

☐ Only one program at a time. When one program finishes, start the next one.

☐ Concurrent running of multiple applications

APP_1 APP_2 .... APP_n

CPU

APP_1	APP_2	APP_3	....
t_1	t_2	t_3	time

Example of Concurrency

```
1 #include <stdio.h>                    16 int
2 #include <stdlib.h>                   17 main(int argc, char *argv[])
3 #include "common.h"                   18 {
4                                        19     if (argc != 2) {
5 volatile int counter = 0;             20         fprintf(stderr, "usage: threads <value>\n");
6 int loops;                            21         exit(1);
7                                        22     }
8 void *worker(void *arg) {             23     loops = atoi(argv[1]);
9     int i;                            24     pthread_t p1, p2;
10    for (i = 0; i < loops; i++) {     25     printf("Initial value : %d\n", counter);
11        counter++;                    26
12    }                                 27     Pthread_create(&p1, NULL, worker, NULL);
13    return NULL;                      28     Pthread_create(&p2, NULL, worker, NULL);
14 }                                    29     Pthread_join(p1, NULL);
15                                      30     Pthread_join(p2, NULL);
                                        31     printf("Final value : %d\n", counter);
                                        32     return 0;
                                        33 }
```

Example of Concurrency

prompt> gcc -o thread thread.c -Wall -pthread

prompt> ./thread 1000

Initial value : 0

Final value : 2000

prompt> ./thread 100000
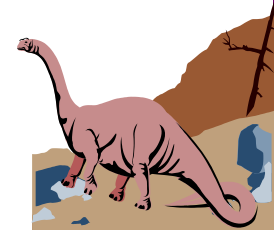
Initial value : 0

Final value : 143012 // huh??

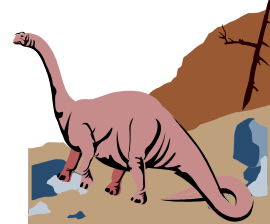prompt> ./thread 100000

Initial value : 0
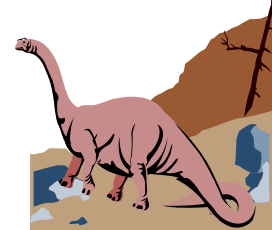
Final value : 137298 // what the??

Persistence

■ How to store data (e.g., outputs of various programs) persistently in a computer.

■ Persistence

◆ In system memory, data can be easily lost, as devices such as DRAM store values in a volatile manner; when power goes away or the system crashes, any data in memory is lost.

◆ Hardware: hard drive, solid-state drives (SSDs), etc.

◆ Software in the operating system: the file system

Example of Persistence

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <assert.h>
4 #include <fcntl.h>
5 #include <sys/types.h>
6
7 int
8 main(int argc, char *argv[])
9 {
10     int fd = open("/tmp/file", O_WRONLY | O_CREAT | O_TRUNC, S_IRWXU);
11     assert(fd > -1);
12     int rc = write(fd, "hello world\n", 13);
13     assert(rc == 13);
14     close(fd);
15     return 0;
16 }
```
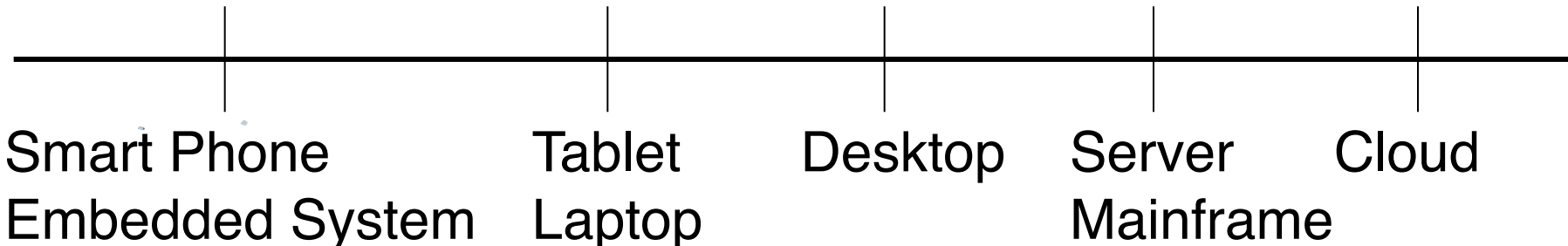
Supported Platforms and Operating Environments

- **Various types of operating systems can be designed to function on different computing platforms**
  - ◆ Real-Time Systems – Embedded devices, automotive systems, industrial control systems, medical devices.
  - ◆ Handheld Systems – Smartphones, tablets, PDAs.
  - ◆ Desktop Systems – Personal computers, laptops.
  - ◆ Multiprocessor Systems – High-performance computing (HPC), scientific computing.
  - ◆ Mainframe Systems – Enterprise servers.
  - ◆ Clustered System – Data centers, parallel processing.
  - ◆ Distributed Systems – Cloud computing.
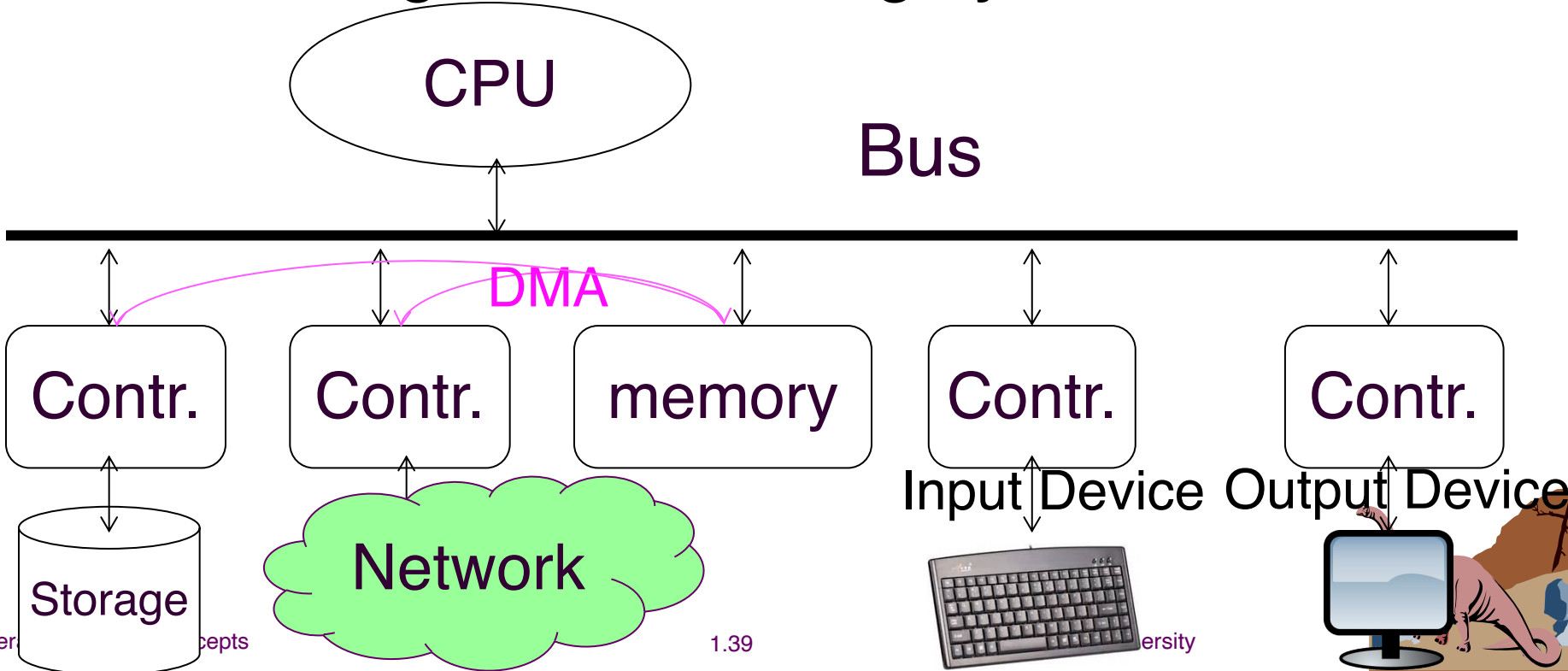 - ◆ Computing Environments – traditional, CS, P2P, Web

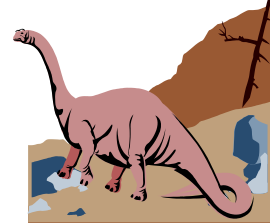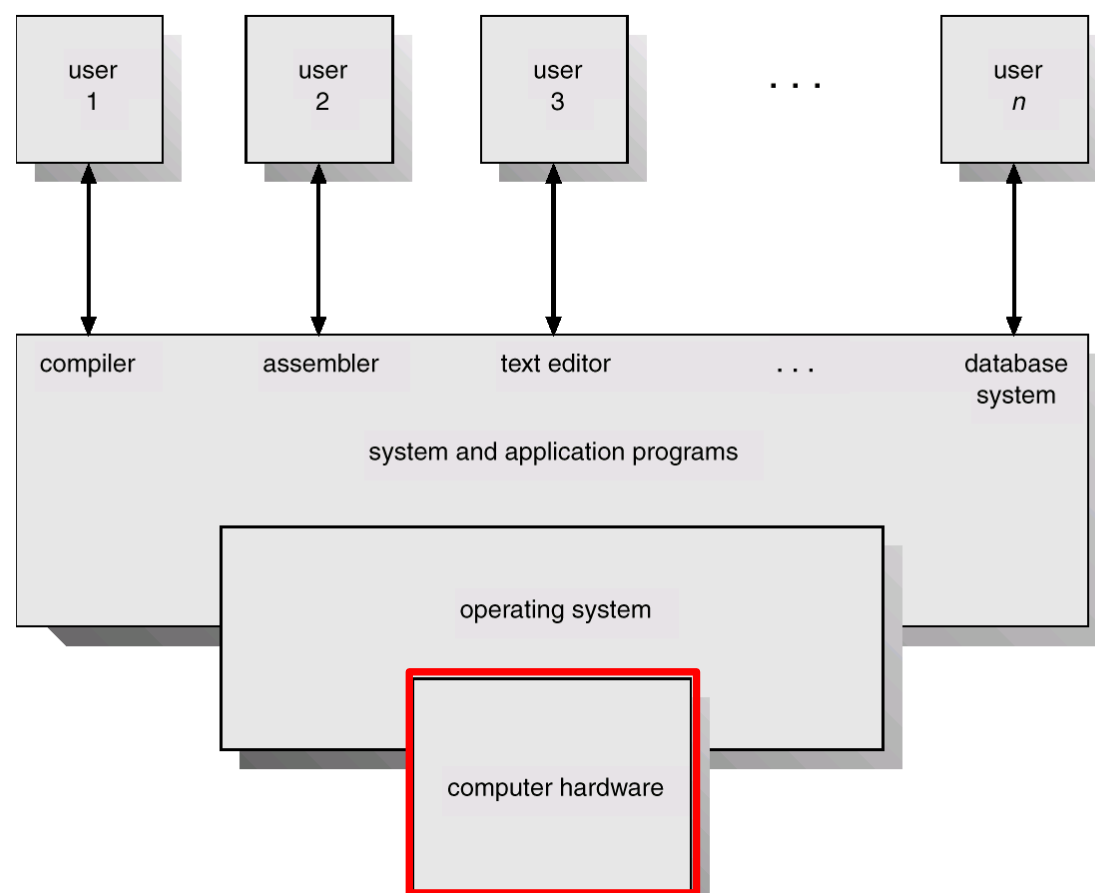# Similar Computer Hardware Organization

■ Hardware Continuum

```
|_____|_____|_____|_____|_____|
```

Smart Phone   Tablet   Desktop   Server   Cloud
Embedded System   Laptop   Mainframe

■ Internal organization: Largely the same

CPU

Bus

DMA

| Contr. | Contr. | memory | Contr. | Contr. |

Input Device Output Device

Storage

Network

Computer System Components (1)

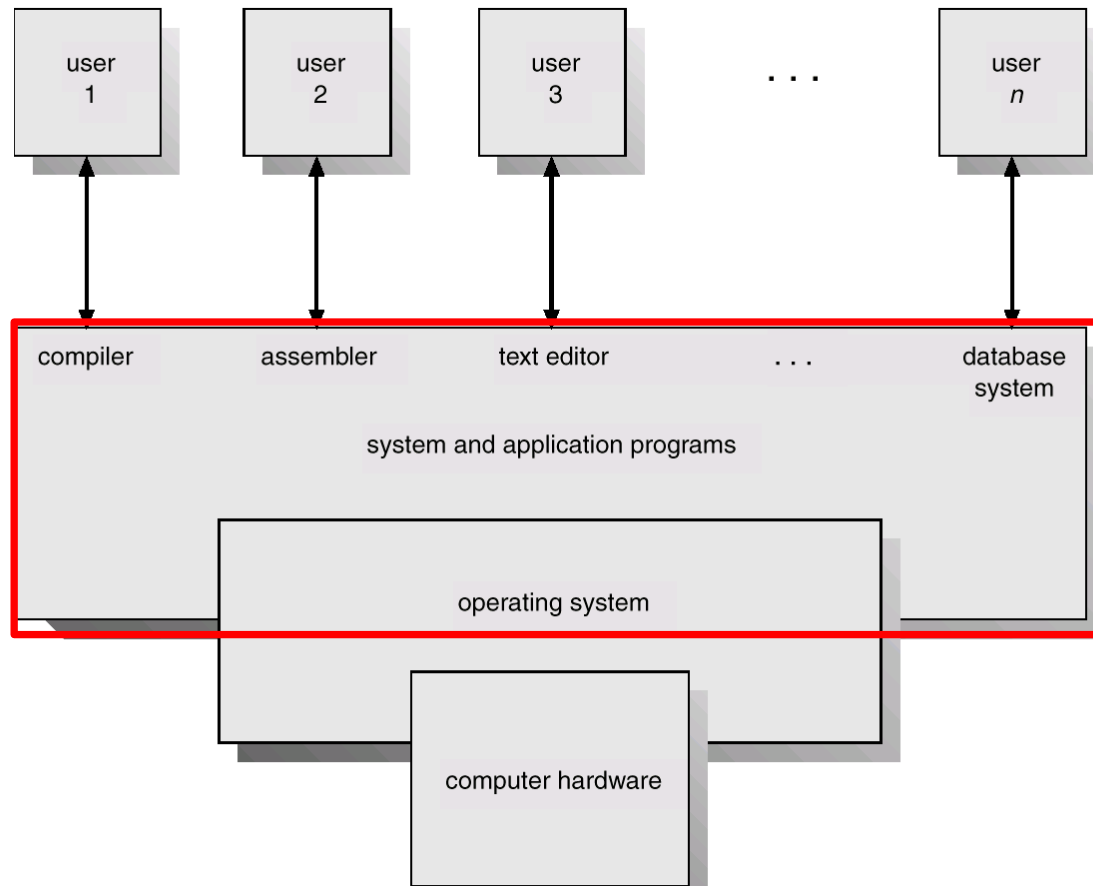1. Hardware – provides basic computing resources (CPU, memory, I/O devices).

Computer System Components (2)

2. Operating system – controls and coordinates the use of the hardware among the various application programs for the various users.
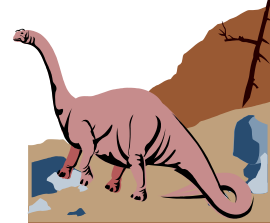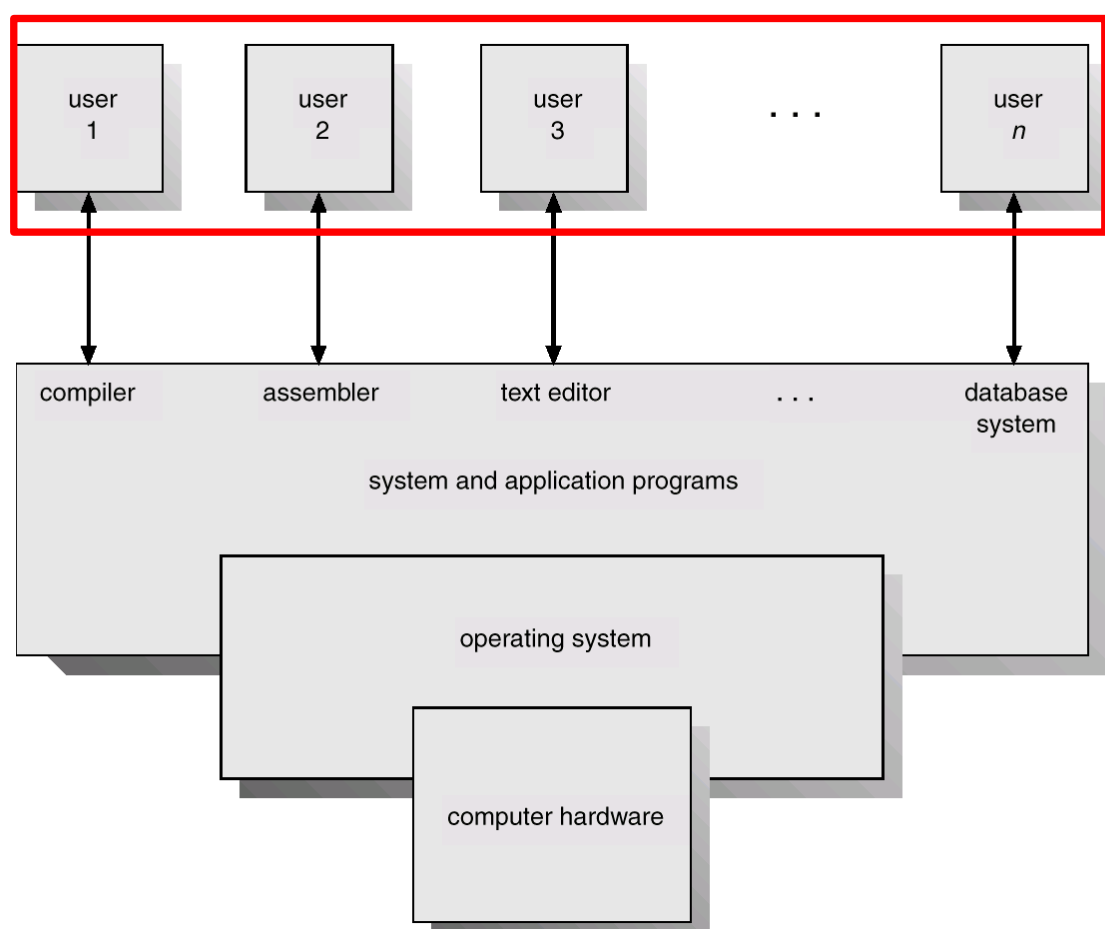
3. Applications programs – (compilers, database systems, video games, business programs) are used to solve user problems.

Computer System Components (4)

4. Users (people, machines, other computers).

OS from User View

■ PC Users: an operating system is a program that provides an **easy-to-use interface** for using the hardware

■ Mainframe/Minicomputer Users: an operating system is a program that helps **maximize the system resource utilization**

■ Workstation Users: an operating system is a program designed to **compromise** between individual usability and resource utilization

OS from System View

- *A Resource allocator*: the operating system allocates and reclaims the system hardware resources to and from user programs

- *A Control Program*: the operating system controls execution of user programs to prevent errors and improper use of computer

- *No* universal definition of what is operating system. But a *common definition* is that the operating system is the one program running at all times on the computer (*i.e.*, the *kernel*). All other programs are application programs
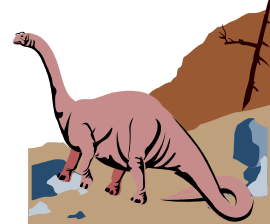
Goals of an Operating System

■ Primary Goal of Operating System

◆ *Efficient* operation of the computer system

◆ In some systems, it is *convenience*

■ In the past, efficiency is far more important than convenience, due to the scarcity of hardware resources

Brief History of Operating System

- Simple Batch System (单道批处理)
  - ◆ Its major task was to transfer the control automatically from one job to the next.

- Multiprogrammed Batch System (多道批处理)
  - ◆ Several jobs are available on disk, OS can select which job to run.

- Time-sharing or Multi-tasking Systems (分时)
  - ◆ Multiple jobs are executed by the CPU, but the switching is so frequent that the user may interact with the program while it is running
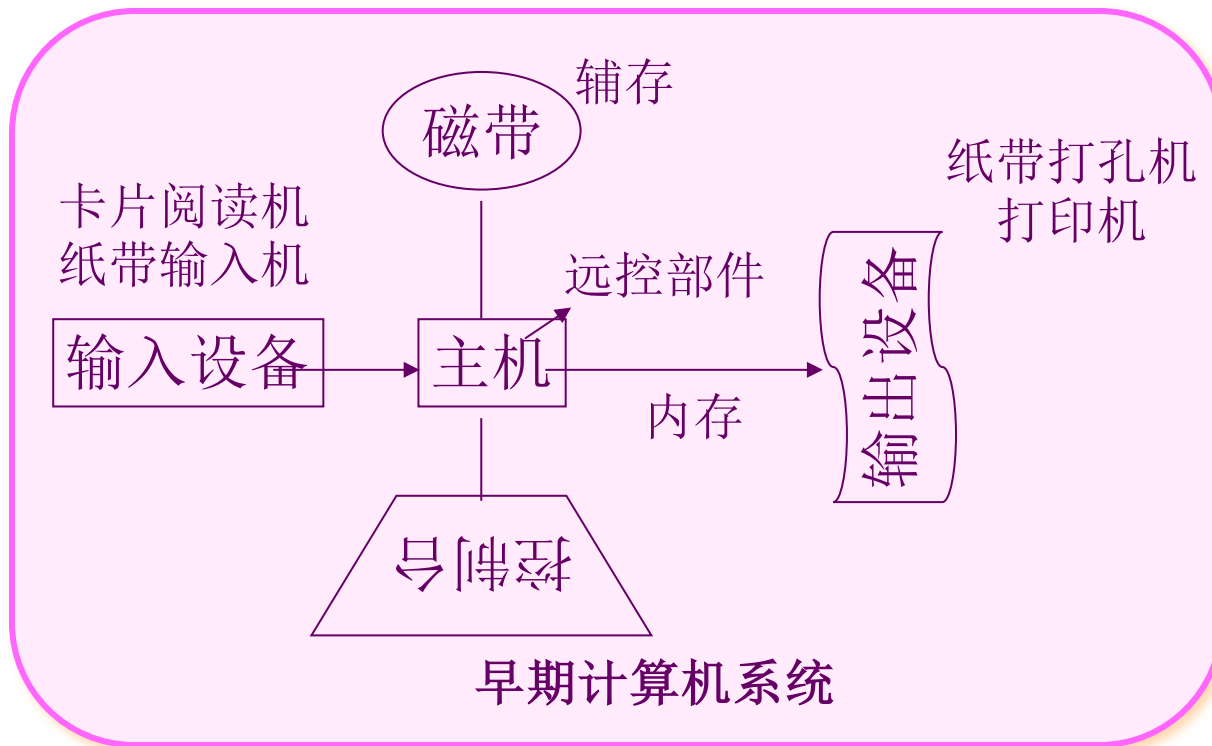
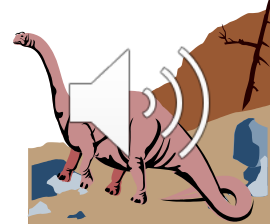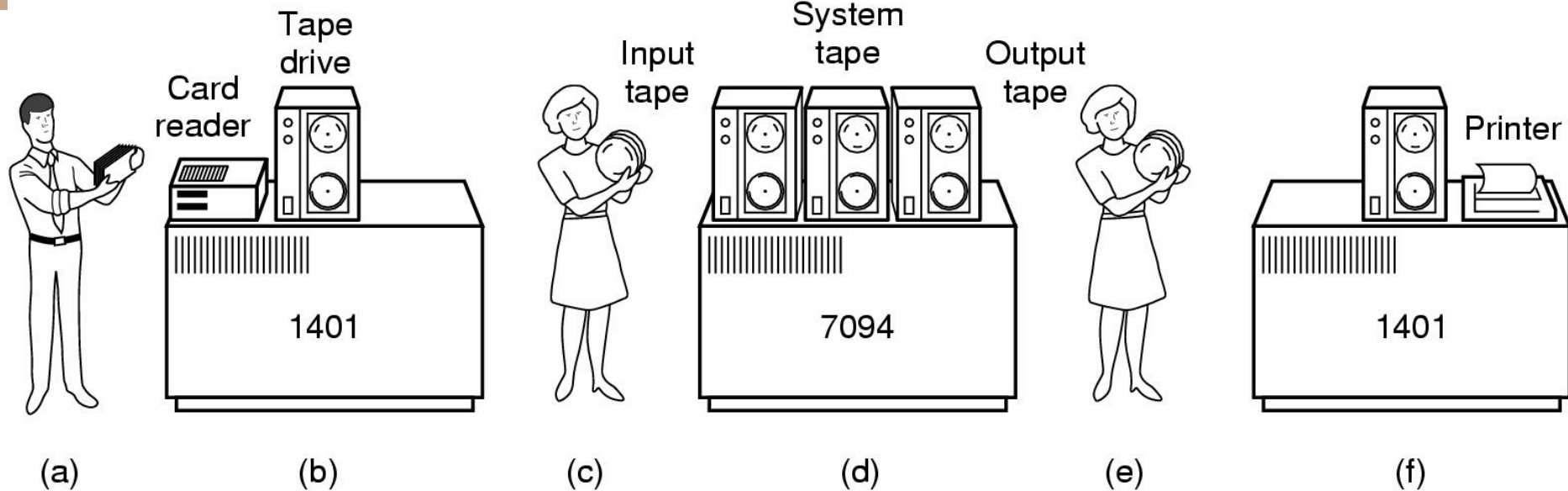- Parallel Systems and Distributed Systems

http://people.scs.carleton.ca/~maheshwa/courses/300/11/node4.html

Simple Batch Systems

■ Execute a list of scientific computation jobs

早期计算机系统

■ 用 户 独 占 全 机 资源

■ 程 序 运 行 前 准 备 时 间 过 长

■ 人 机 速 度 不 匹 配

An Example of Batch Systems

Tape drive — Card reader — 1401 (a)(b)

Input tape — System tape — Output tape — 7094 (c)(d)(e)

Printer — 1401 (f)

◆ control card tells what the job is supposed to do, e.g. a PASCAL program or a Fortran program etc.

◆ bring cards to 1401

◆ read cards to tape

◆ put tape on 7094 which does computing

◆ put tape on 1401 which prints output

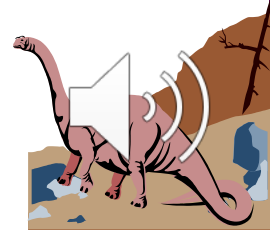https://en.wikipedia.org/wiki/Timeline_of_operating_systems
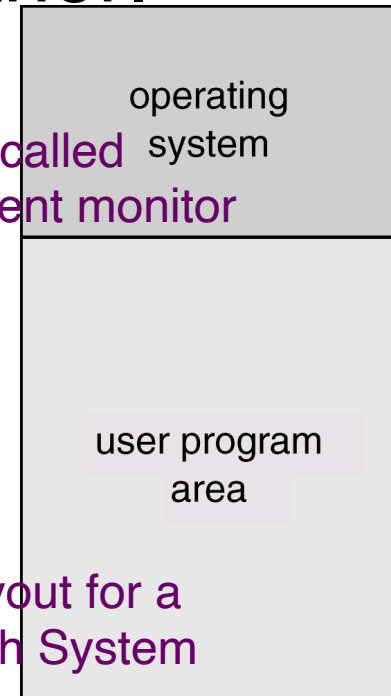
Features of Simple Batch Systems

■ Automatic job sequencing – automatically transfers control from one job to another.

■ First rudimentary operating system：Resident monitor (驻留监督程序)

Also called resident monitor

◆ initial control in monitor

◆ control transfers to job

◆ when job completes control transfers back to monitor

Memory Layout for a Simple Batch System

operating system
user program area

■ To speed up processing, the operator will group all the jobs with similar needs, and run them together.
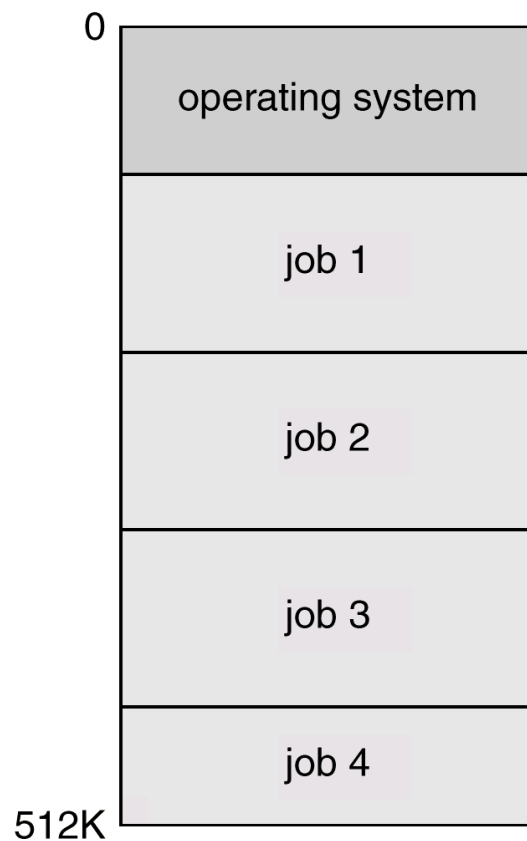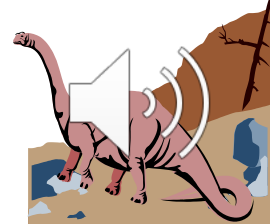
Multiprogramming Systems

Its idea is to ensure that CPU is always busy. A single job may not be able to be keep CPU busy. Multiprogramming allows to keep the CPU busier as follows:

■**Keeps serval jobs in memory simultaneously**

■**Picks and begins to execute one**

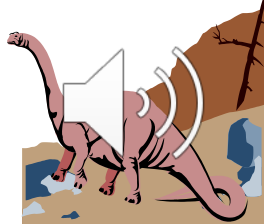■**If that job needs to wait, CPU is switched to another one**

0

operating system
job 1
job 2
job 3
job 4

512K

It's the first instance where the operating system must make decisions for the users, which essentially leads to job scheduling

OS Features Needed for Multiprogramming

■ I/O routine must be supplied by OS.

■ Memory management – the system must allocate the memory to several jobs.

■ CPU scheduling – the system must choose among several jobs ready to run.

■ Allocation of devices.

■ Job scheduling – the system must choose among jobs ready to be brought into memory
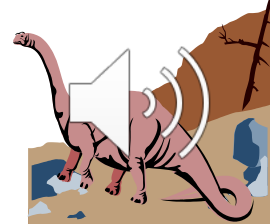
Advantages and disadvantages of multiprogramming:
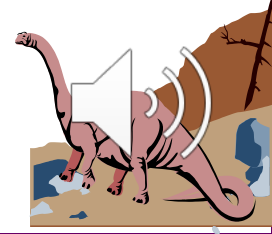
· CPU 利用率大大提高
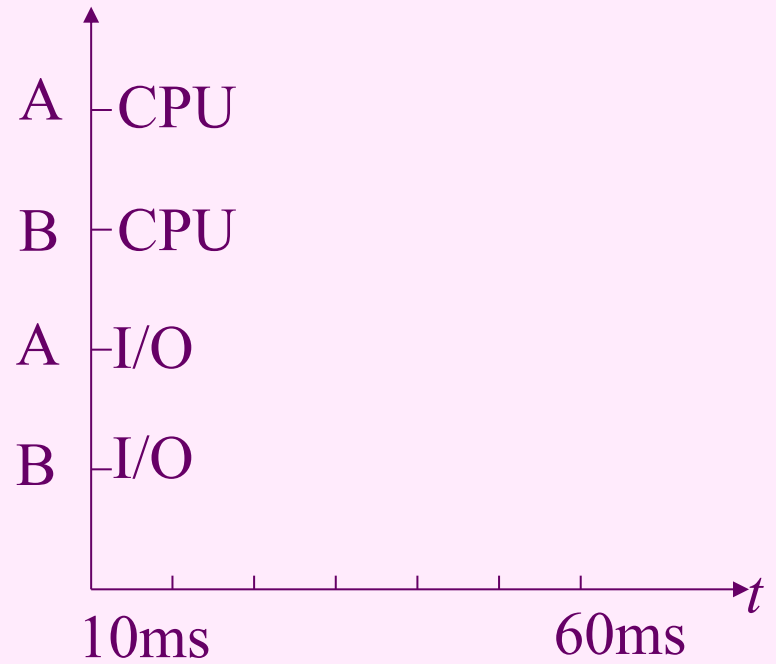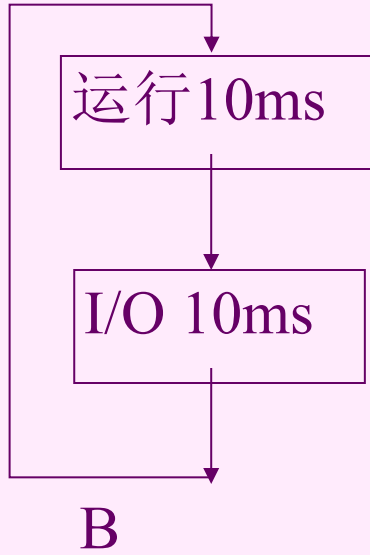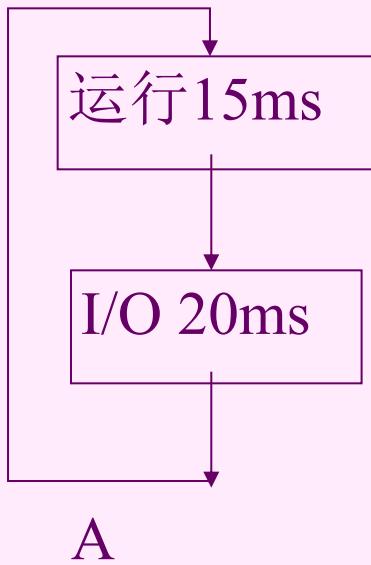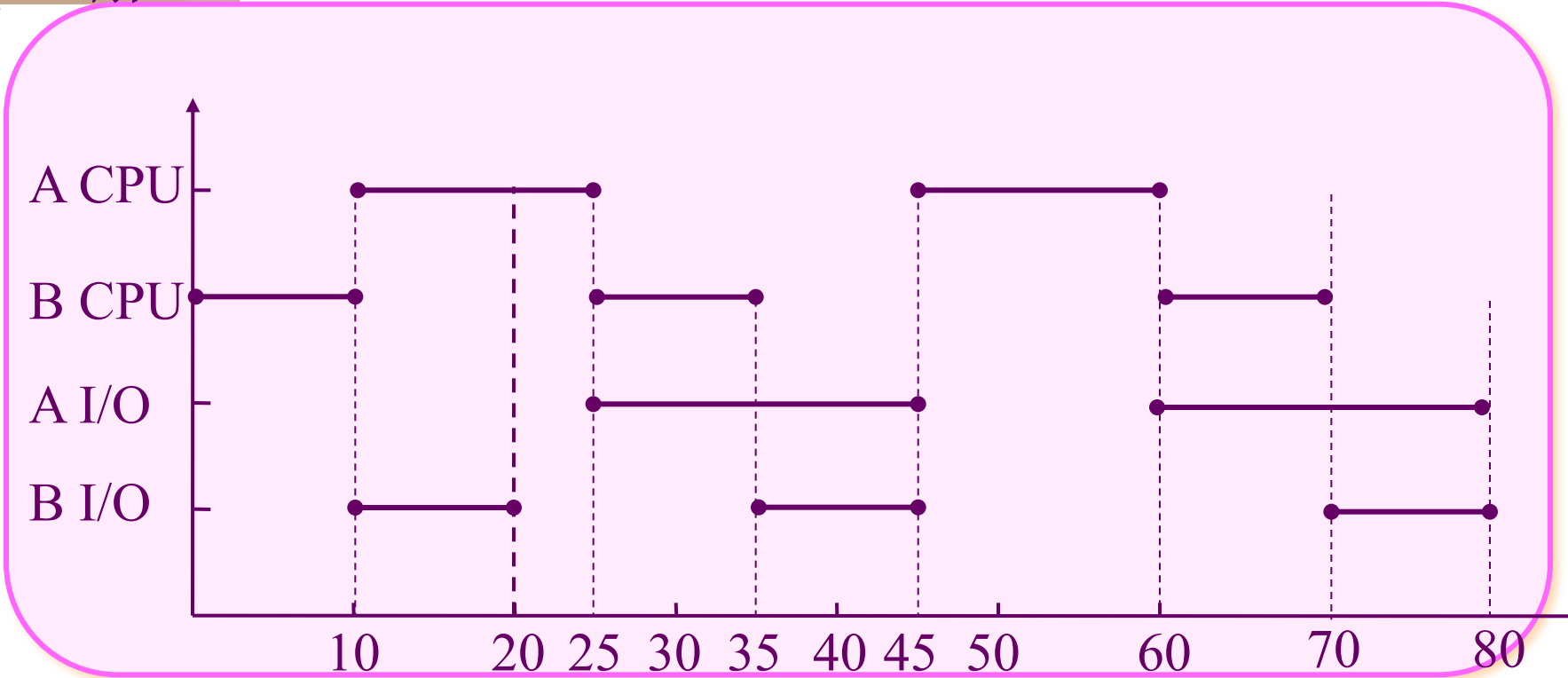
用户无控制权，无交互性，延迟大

引入多道程序设计技术的根本目的：
提高CPU的利用率，充分发挥并行性，
这包括：程序之间，设备之间，设备与CPU之间
均并行工作。

例： 有两道程序A、B，按下图以多道程序方式运行，要求在右图画出它们的运行轨迹，并计算在60ms内，CPU的利用率，假设起始时首先运行B，并允许忽略监督程序切换A、B的时间。

运行15ms

I/O 20ms

A

运行10ms

I/O 10ms

B
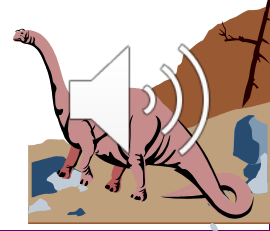
A — CPU

B — CPU

A — I/O

B — I/O

10ms   60ms t

解：

$$P_{\mathrm{CPU}} = \frac{60-(45-35)}{60}\times100\% = \frac{50}{60}\times100\% = 83.3\%$$

若在单道程序系统中，**CPU**的利用率是多少？

Time-Sharing Systems-Interactive Computing

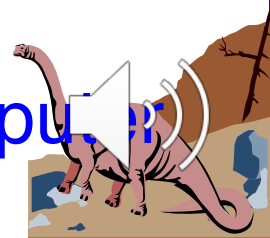- Interactive computer system provides direct communication between user and the system
  - ◆ The user gives instructions to the system directly, waits for immediately results. Response time should be short.
  - ◆ When the operating system finishes the execution of one command, it seeks the next "control statement" from the user's keyboard.
- Time-sharing operating system allows many users to share the computer simultaneously
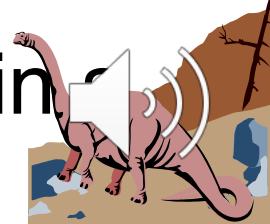 - ◆ Switches rapidly from one user to the next
  - ◆ Gives users the impression: the entire computer system is dedicated for my use.

Time-Sharing Systems-Interactive Computing (cont.)

■ The CPU is multiplexed among several jobs that are kept in memory and on disk

■ To obtain a reasonable response time, a job needs to be swapped in and out of memory to the disk (virtual memory).

■ Disk management must be provided

■ Sophisticated CPU-scheduling schemes are required for concurrent execution

■ Job synchronization mechanisms are needed

■ It may ensure that jobs do not get stuck in a deadlock

作业：从技术发展的角度，简要介绍批处理系统、多道程序系统和分时系统各自的技术特性。（在该技术出现以前系统存在哪些问题，该技术是如何解决这些问题的）