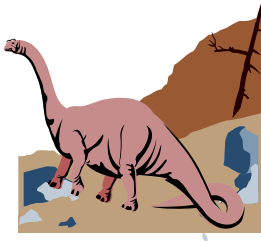
操作系统原理

肖 卿 俊
办公室：九龙湖校区计算机楼212室
电邮：csqjxiao@seu.edu.cn
主页：https://csqjxiao.github.io/PersonalPage
电话：025-52091022



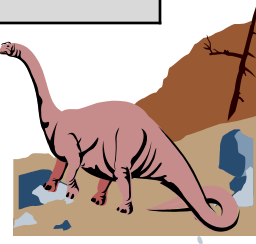
Course Summary

- This course is intended 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
- 学时：64学时
- Textbook：操作系统概念（第七版或第九版影印版） 西尔伯莎茨编著 高教出版社

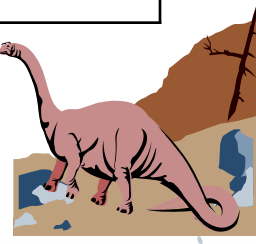
Course Outline (1/2)

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



Course Outline (2/2)

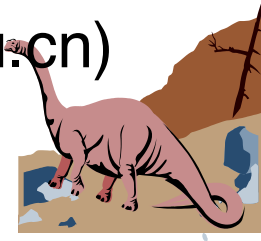
Chapter	教学要求
8. Memory Management	掌握固定分区, 可变分区, 段式, 页式存储管理原理与算法
9. Virtual Memory	虚拟存储是操作系统另一个重要概念。要求学生正确理解虚存的概念。 熟练掌握请求页式的地址变换过程以及常用的页面置换算法
10. File Systems	掌握文件系统的基本概念，熟练使用文件系统的操作方法
11. I/O Systems	掌握I/O系统的基本概念，理解设备驱动程序。 弄清从用户请求到设备完成用户请求的全过程
12. Mass-Storage Structure	了解几种磁盘调度算法



Course Requirements

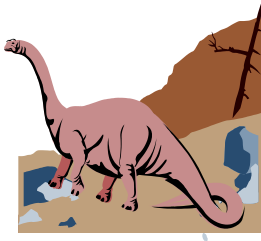
- **Grading Policy (考核方式)**
  - 闭卷考试
  - 成绩评定:由下述三部分组成
    - ✓期末考试成绩
    - ✓期中考试成绩
    - ✓平时成绩（作业、出勤等）
- Course Schedule:
  - [1-16周]　周二(6-7) 教三-300，14:00~15:35
    周五(3-4) 教三-300，09:50~11:25
- Office Hours:
  - Monday: 10:00~13:45;　Thursday:12:00~13:00

(E-mail discussions are welcome: csqjxiao@seu.edu.cn)

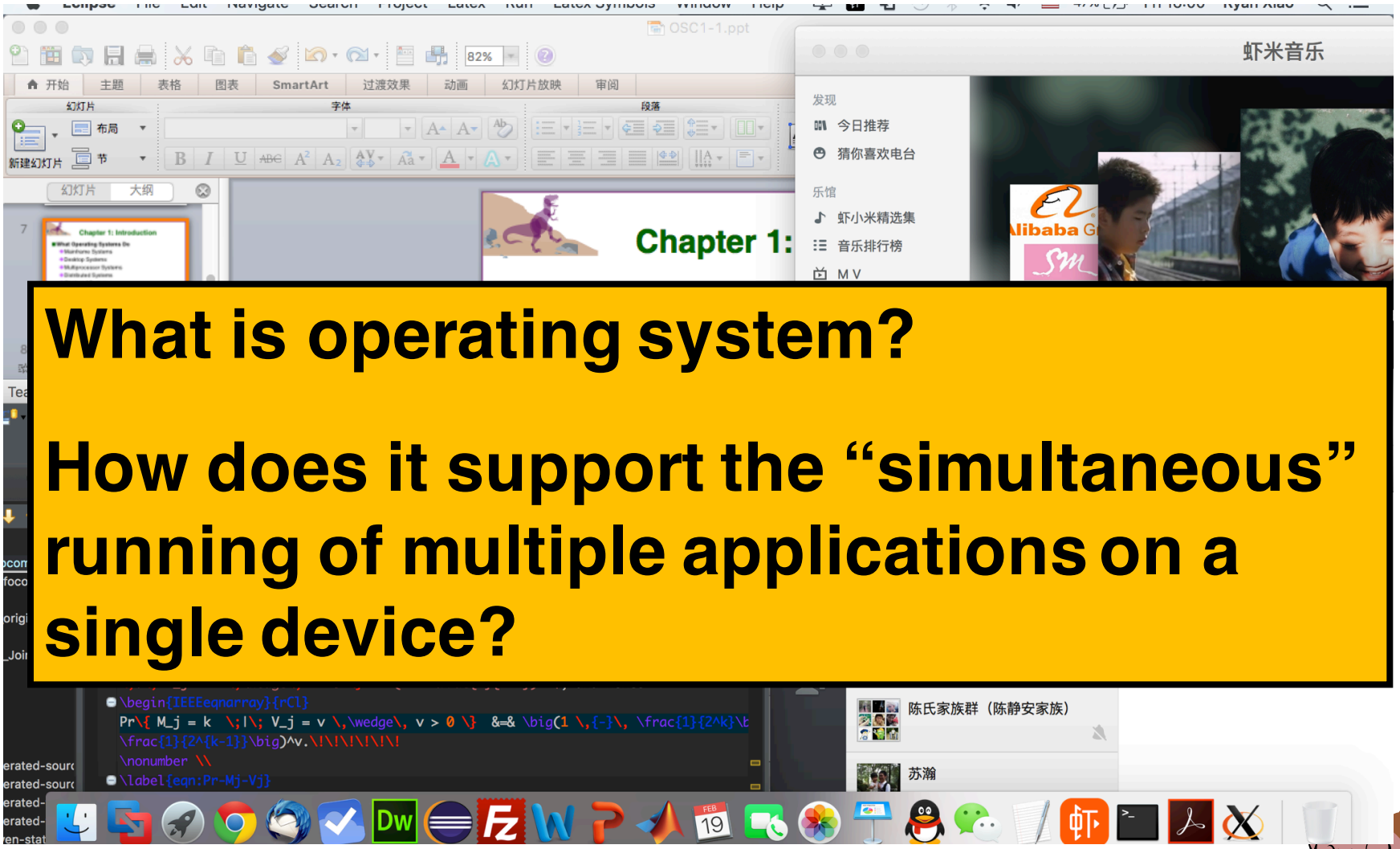


Chapter 1: Introduction (Outline)

- 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



Multiple applications can be run simultaneously on a single device



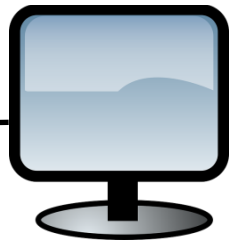
What is operating system?

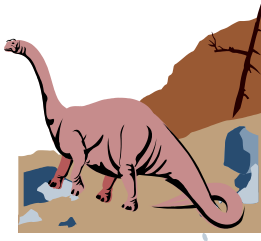
How does it support the "simultaneous" running of multiple applications on a single device?

Computer Operating System manages all the HW resources

Applications

Computer Operating System

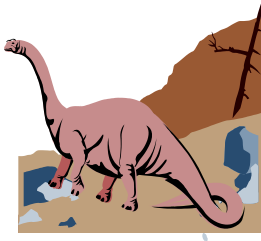




Question about OS

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



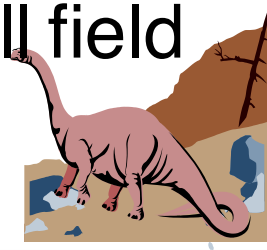
The Powerful Concept of "Abstraction"

- 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

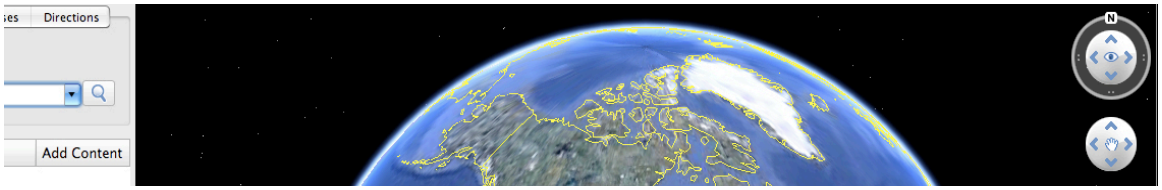
Question about Abstraction

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

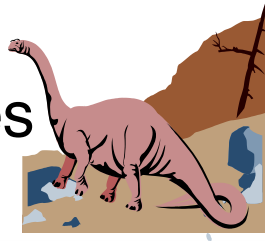


Google Earth

Applications
System Software (**OS**, compiler, etc.)
Instruction set architecture
Computer organization (data path + control)
Sequential + Comb. Logic elements
Logic gates (and, or, not, …)
Transistors

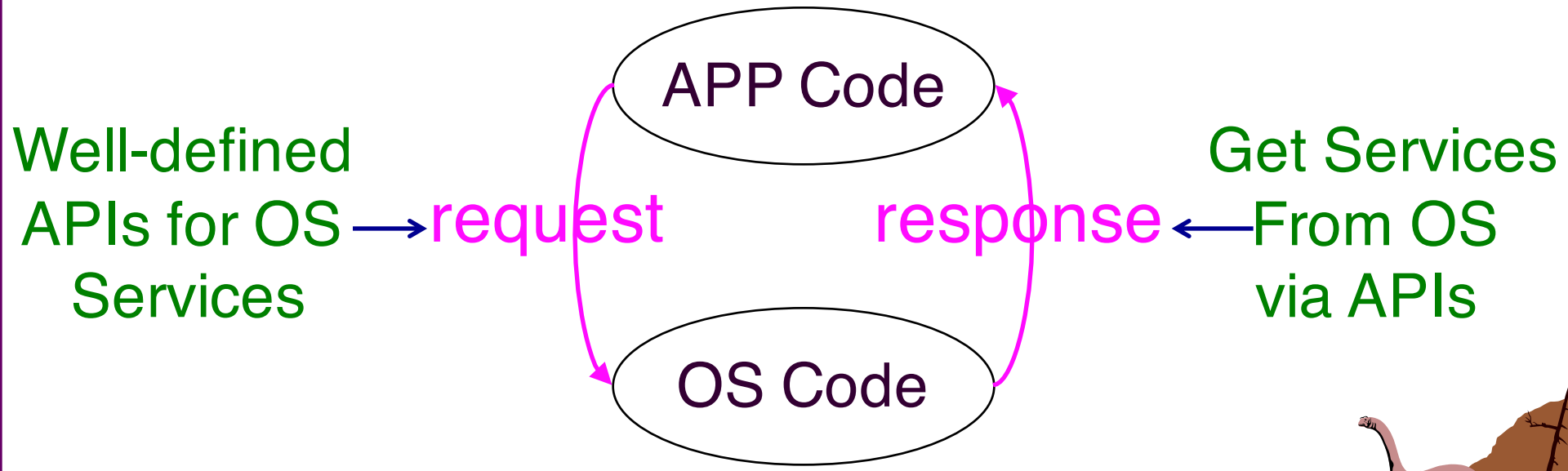
Series of Abstractions

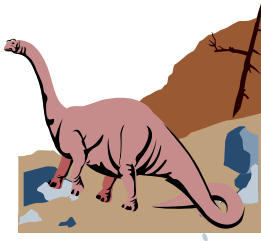
Electrons + Holes



What is an Operating System?

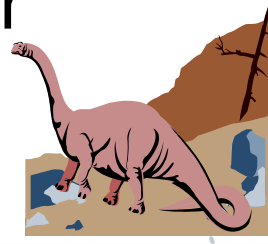
- OS, in a nutshell, provides
  - Protected access to hardware resources
  - Arbitrate among competing request





What is an Operating System?

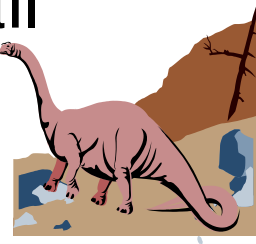
- A program that acts as an intermediary between the application programs and the computer hardware.
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient way
- From the computer's point of view:
  - A resource allocator, a control program, or more concisely, a "kernel"



Question about OS functionalities

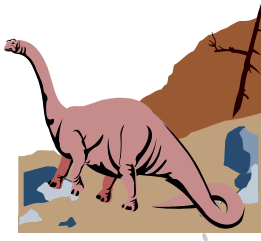
Please select the choices that apply to the functionalities provided by an OS

- ☐ 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



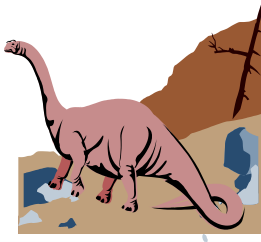
Virtualization

- 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.



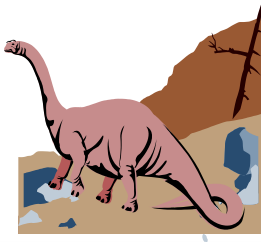
Virtualizing the CPU

```
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>
```

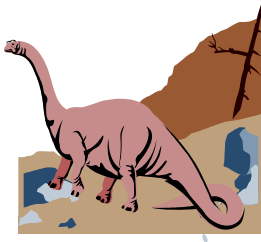


Virtualizing the CPU

```
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
...
```

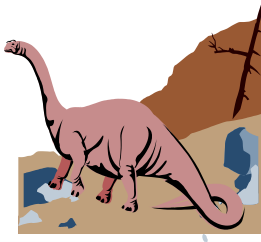
It may appear that each program is running with one dedicated processor core.

Reason: multi-processor? Not really.



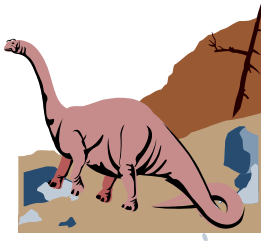
Virtualizing the Memory

```
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 }
```



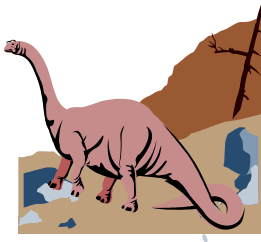
Virtualizing the Memory

```
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
```



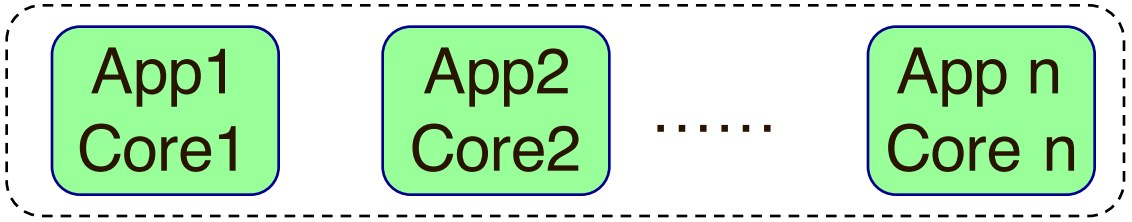
Virtualizing the Memory

```
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
```

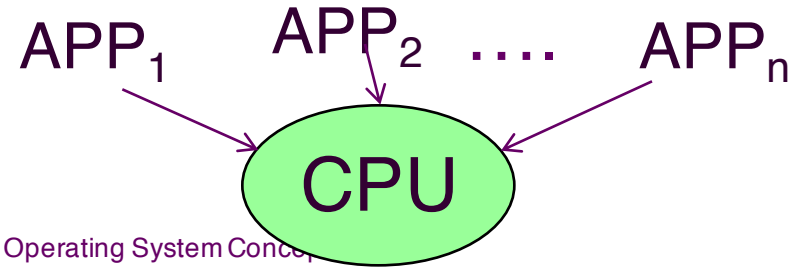


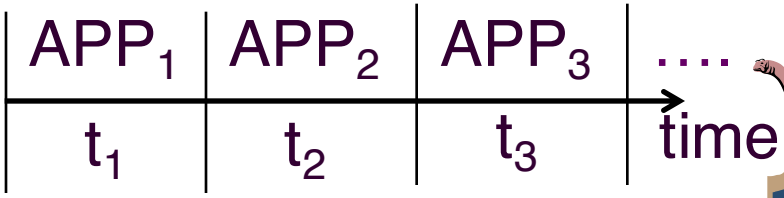
Program Concurrency

- Computer seemingly runs several program in parallel: email, browser, music player, …
- If there is only one CPU, how is it possible?
  - Multiple cores in a CPU



  - Only one program at a time. When one program finishes, start the next one.
  - Concurrent running of multiple applications

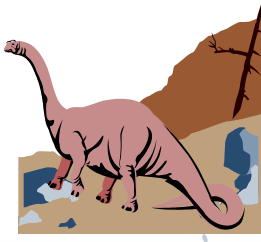




Example of Concurrency

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "common.h"
4
5 volatile int counter = 0;
6 int loops;
7
8 void *worker(void *arg) {
9     int i;
10    for (i = 0; i < loops; i++) {
11        counter++;
12    }
13    return NULL;
14 }
15
```

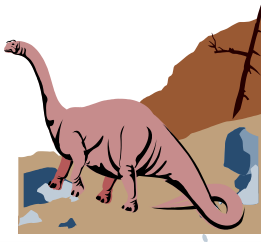
```
16 int
17 main(int argc, char *argv[])
18 {
19    if (argc != 2) {
20        fprintf(stderr, "usage: threads <value>\n");
21        exit(1);
22    }
23    loops = atoi(argv[1]);
24    pthread_t p1, p2;
25    printf("Initial value : %d\n", counter);
26
27    Pthread_create(&p1, NULL, worker, NULL);
28    Pthread_create(&p2, NULL, worker, NULL);
29    Pthread_join(p1, NULL);
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
```

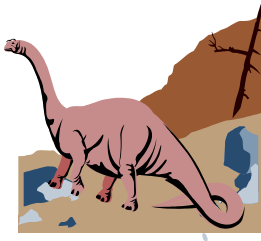
```
prompt> ./thread 100000
Initial value : 0
Final value : 143012 // huh??
prompt> ./thread 100000
Initial value : 0
Final value : 137298 // what the??
```



Persistence

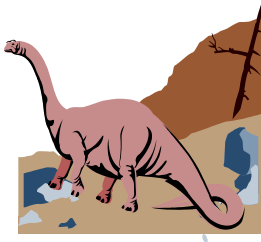
- How to store data (e.g., outputs of various programs) persistently in a computer.

- 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



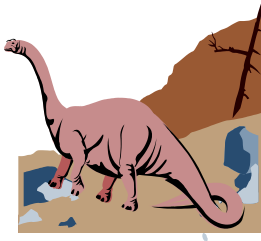
Example of Persistence

```
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 }
```



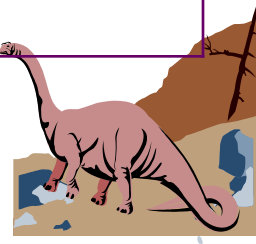
What does an Operating System Do?

- Discussion
  - An OS takes 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.



What does an Operating System Do? (cont.)

Fundamental Concepts / Functional Concepts	Virtualization	Concurrency	Persistence
Process Management	Process, CPU Scheduling	Threads, Dead Locks, Thread Synchronization	
Memory Management	Virtual Memory, Memory Management		
Storage Management	File System		Mass Storage Structure



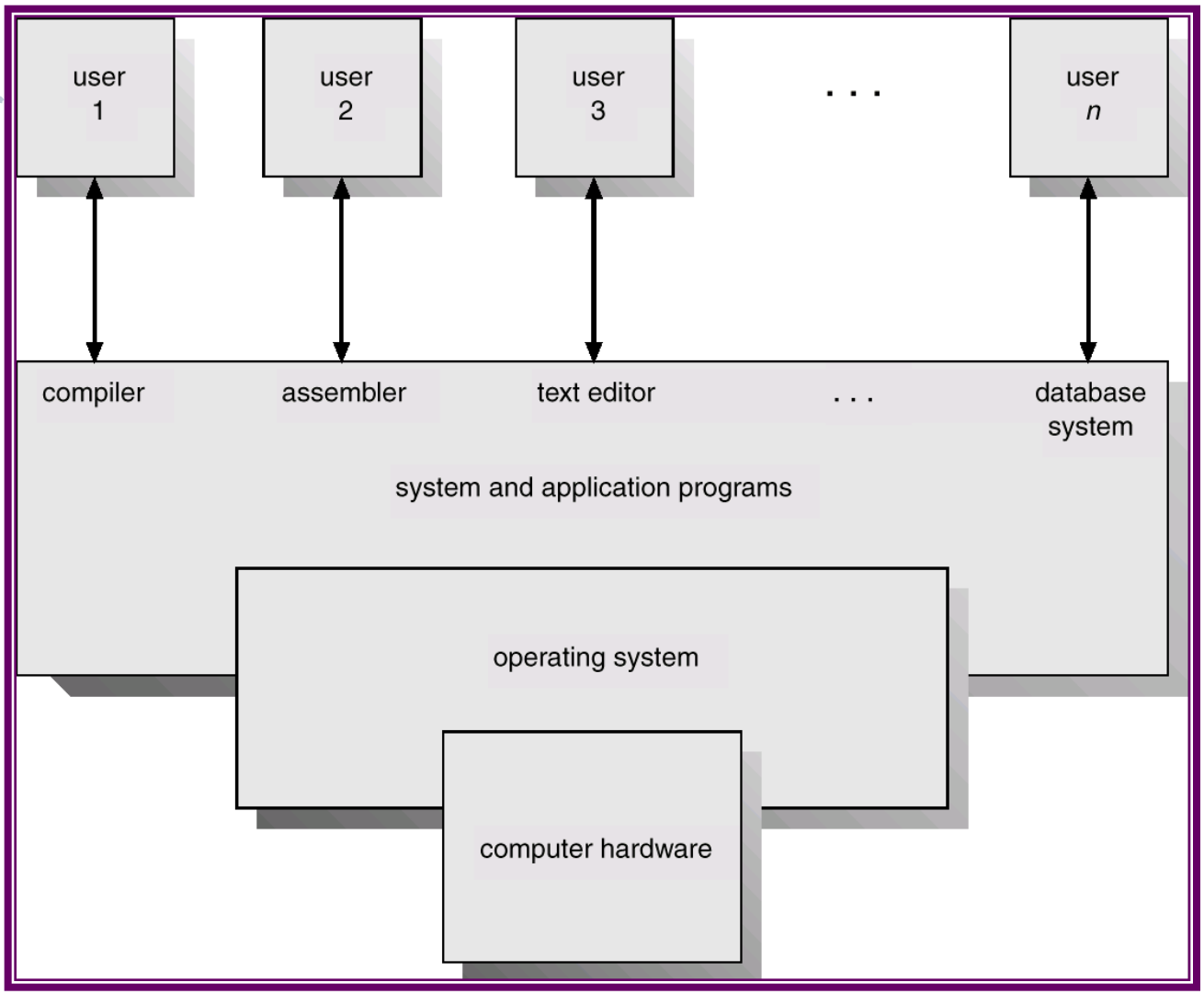
Chapter 1: Introduction

- **What Operating Systems Do**
  - Real-Time Systems
  - Handheld Systems
  - Desktop Systems
  - Multiprocessor Systems
  - Mainframe Systems
  - Clustered System
  - Distributed Systems
  - Computing Environments (traditional, CS, P2P, Web)

Computer System Components

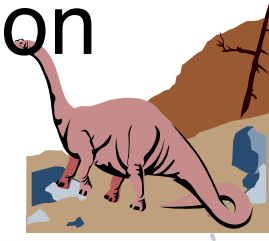
1. Hardware – provides basic computing resources (CPU, memory, I/O devices).
2. Operating system – controls and coordinates the use of the hardware among the various application programs for the various users.
3. Applications programs – (compilers, database systems, video games, business programs) are used to solve user problems.
4. Users (people, machines, other computers).

Abstract View of System Components



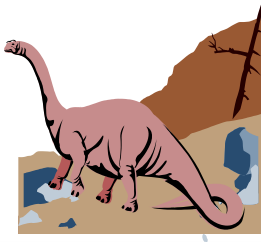
OS from User View

- 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 (1/2)

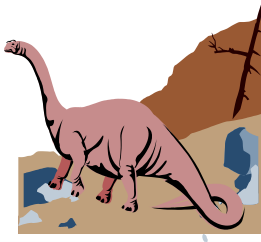
- *A Resource allocator*: the operating system allocates and reclaims the system hardware resources to and from user programs
- *A Control Program*: the operating system controls the execution of user programs to prevent errors and improper use of the computer



OS from System View (2/2)

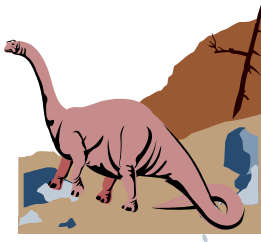
- There is *no* universal definition of what is an operating system.

- 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



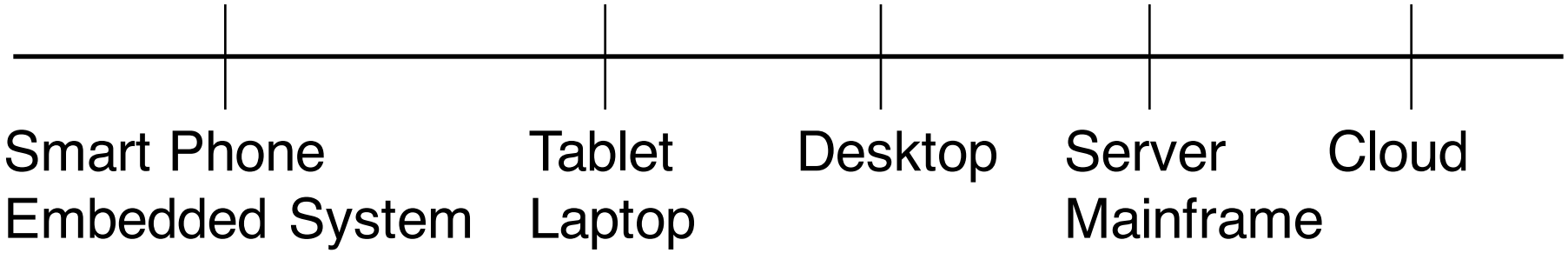
Goals of an Operating System

- 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

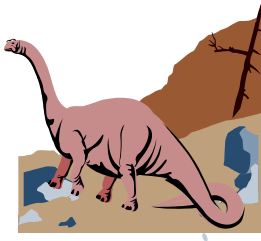


Question about Computer Organization

- Hardware Continuum

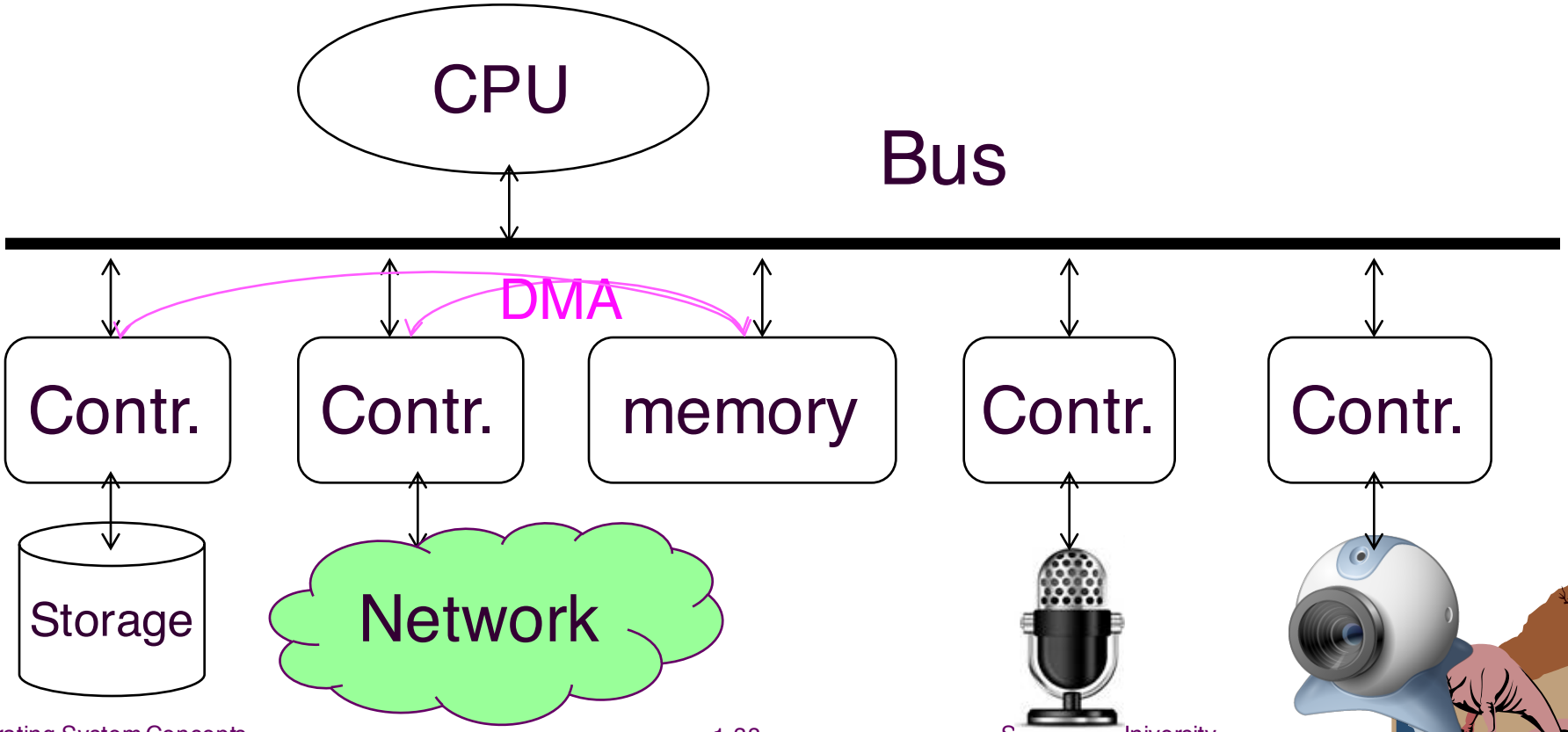


- Is the internal organization of the computer system in this continuum vastly different?
  - ☐ Yes
  - ☐ No



Hardware Resources in a Computer System

- Internal organization
  - Largely the same for all manifestations



Organization with I/O Bus

北桥？南桥？

Frame buffer

CPU

memory

System Bus

Higher Speed

Bridge

DMA

I/O Bus

Lower Speed

Contr.

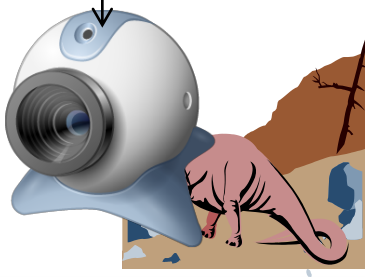
Contr.

Contr.

Contr.

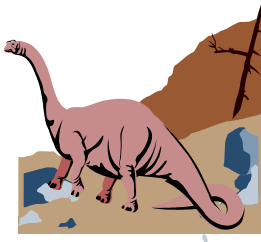
Storage

Network



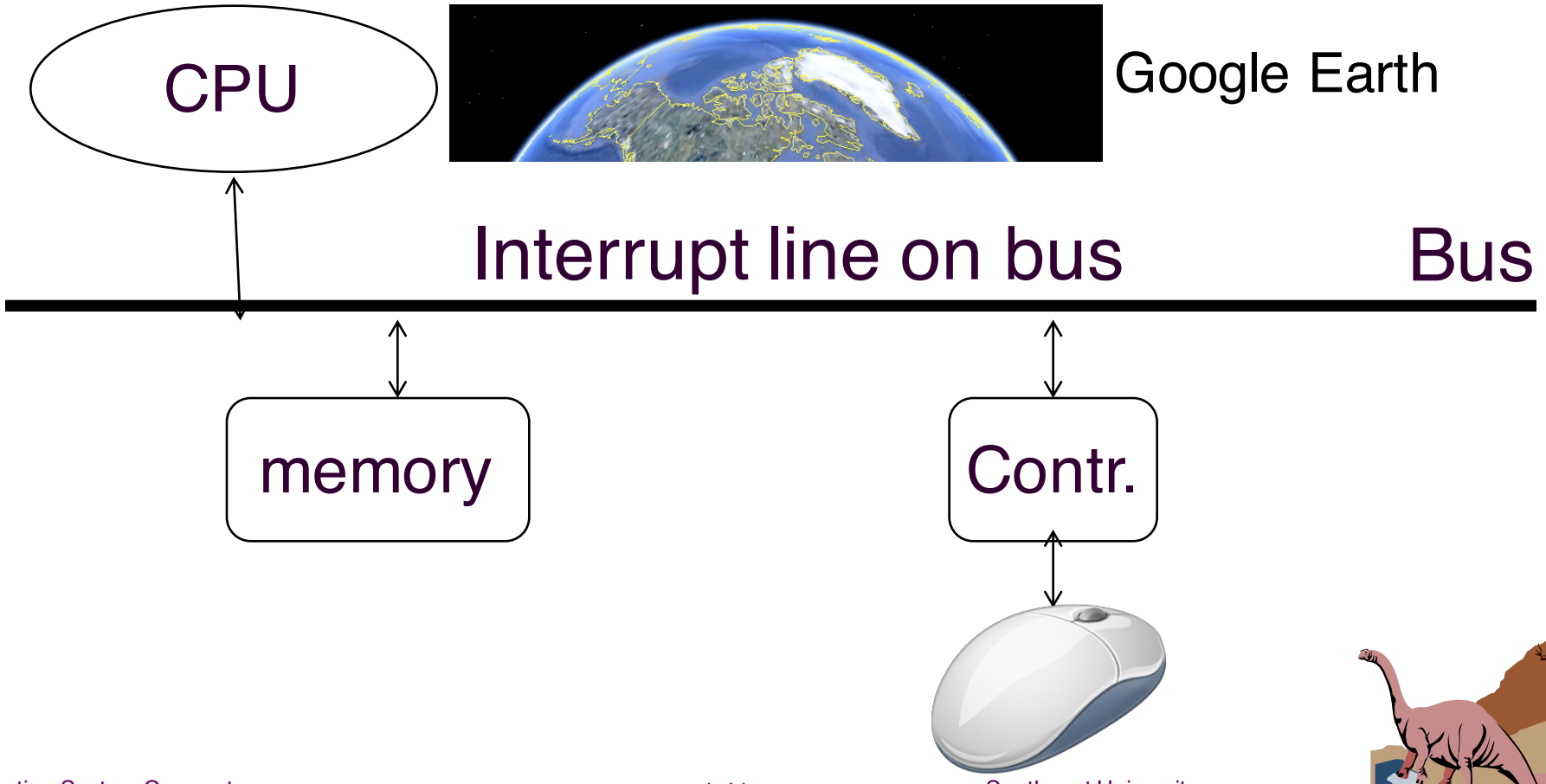
Question about HW/SW interaction

- What happen when you click the mouse?
  - Mouse clicks correspond to specific CPU instructions
  - Mouse click starts up a specific program to read spatial coordinates of mouse
  - It results in a CPU interrupt



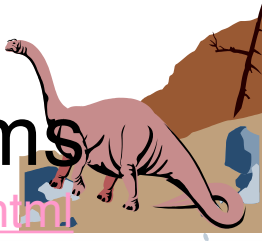
An example of HW/SW Interaction

- What happens when you click your mouse?



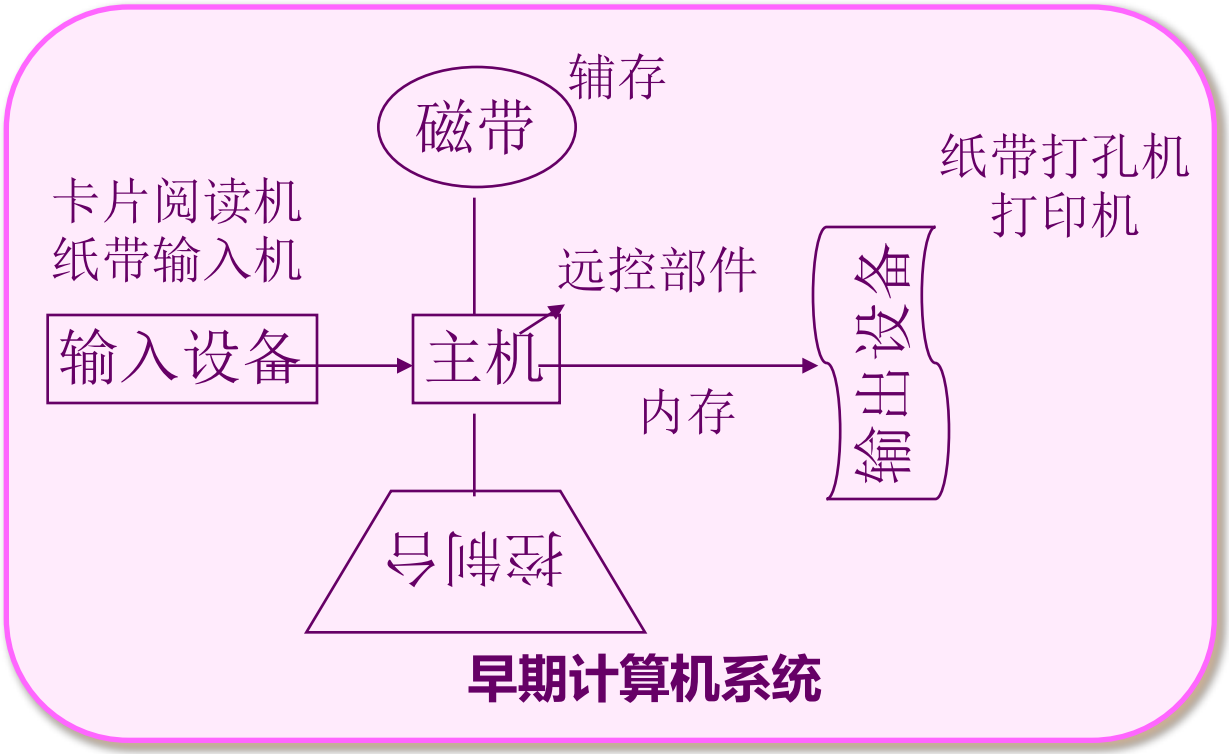
Brief History of Operating System

- 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

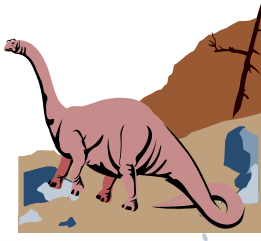


Simple Batch Systems

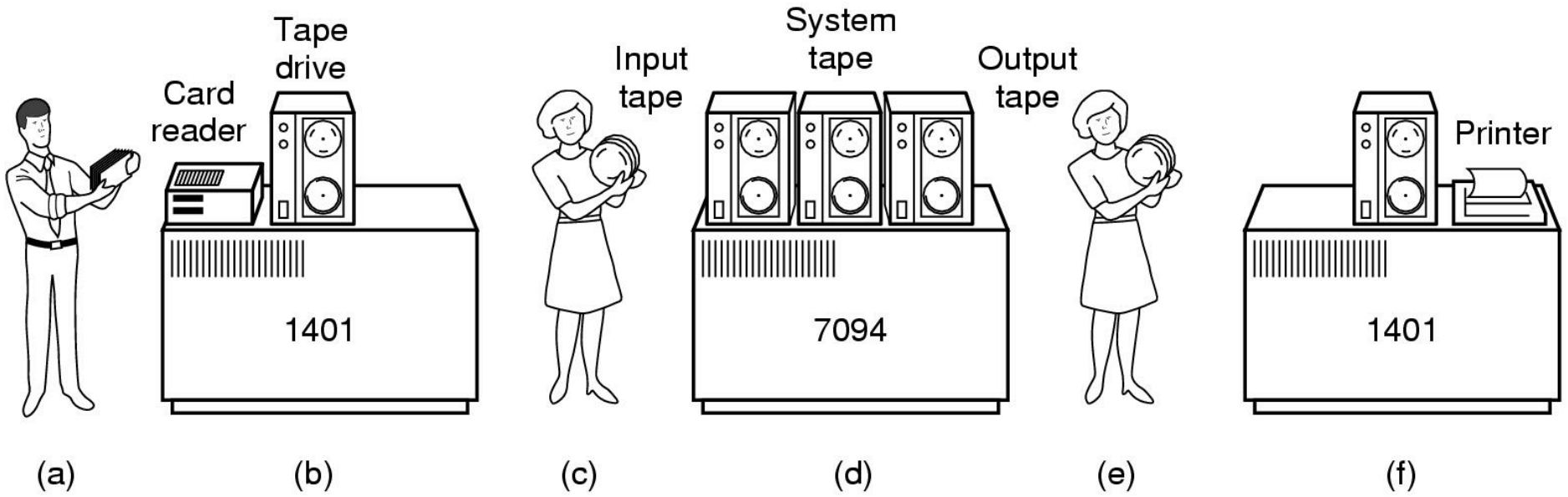
- Execute a list of scientific computation jobs



- **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**

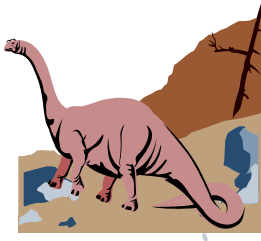


An Example of Batch Systems



- 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

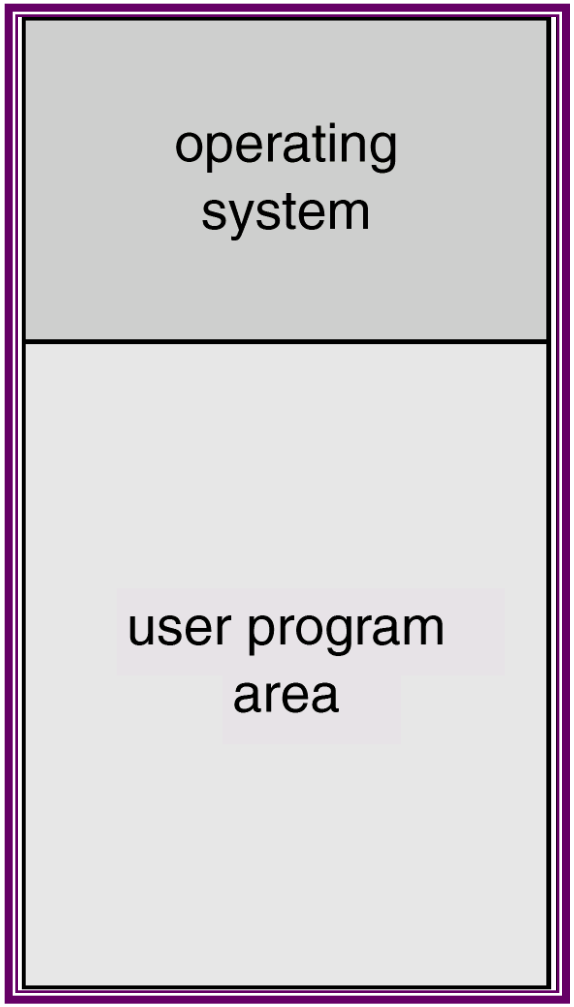
https://en.wikipedia.org/wiki/Timeline_of_operating_systems

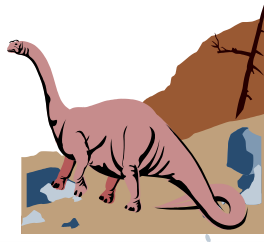


Features of Simple Batch Systems

- Automatic job sequencing – automatically transfers control from one job to another.
- First rudimentary operating system：Resident monitor (驻留监督程序)
  - initial control in monitor
  - control transfers to job
  - when job completes control transfers back to monitor
- To speed up processing, the operator will group all the jobs with similar needs, and run them together.

Memory Layout for a Simple Batch System

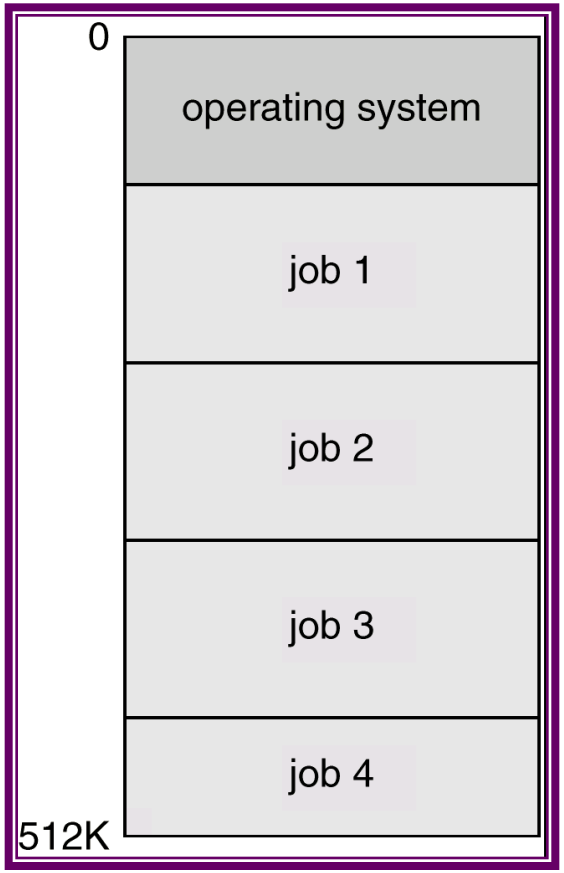




Multiprogramming Systems

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 more busy as follows:

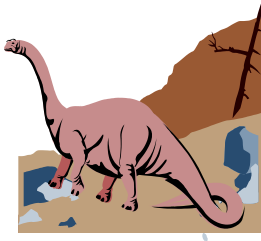
- **Keeps serval jobs in memory simultaneously**
- **Picks and begins to execute one**
- **If that job needs to wait, CPU is switched to another one**

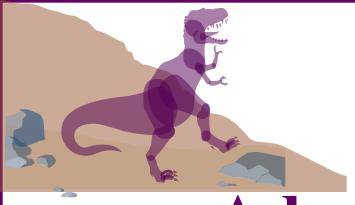


It's the first instance where the operating system must make decisions for the users, which essentially leads to job scheduling

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



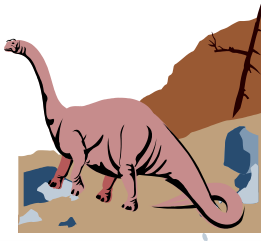


Advantages and disadvantages of multiprogramming:

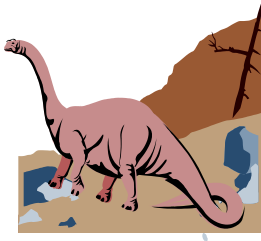
- CPU 利用率大大提高

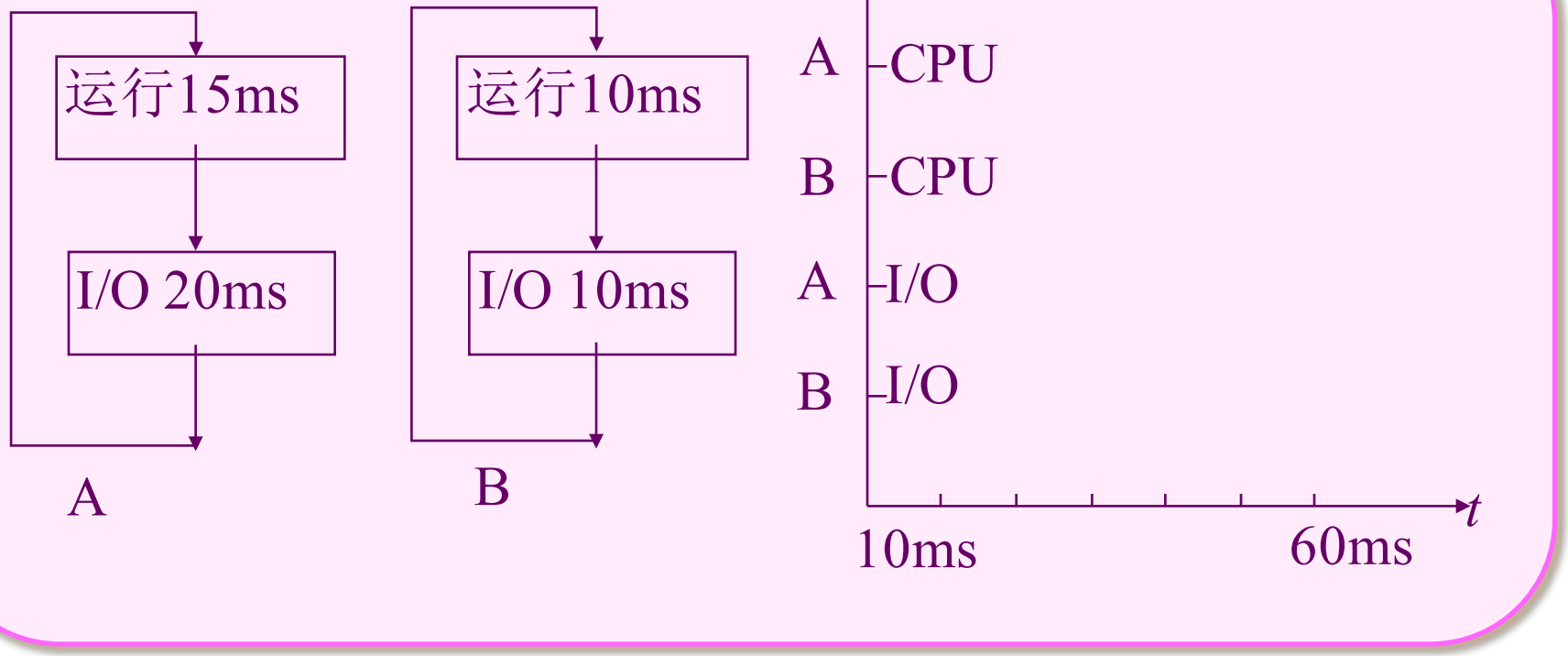
用户无控制权，无交互性，延迟大

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。

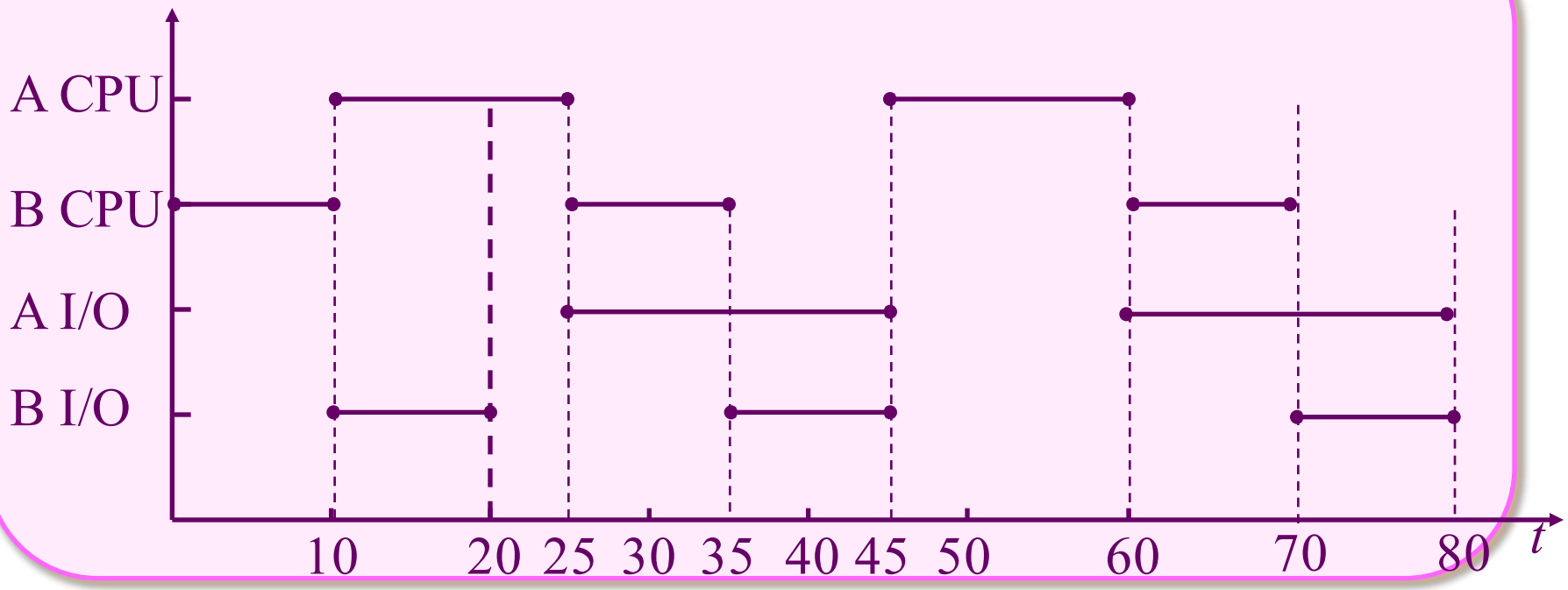


例： 有两道程序A、B，按下图以多道程序方式运行，要求在右图画出它们的运行轨迹，并计算在60ms内，CPU的利用率，假设起始时首先运行B，并允许忽略监督程序切换A、B的时间。




运行15ms
I/O 20ms
A
运行10ms
I/O 10ms
B
A
CPU
B
CPU
A
I/O
B
I/O
t
10ms
60ms

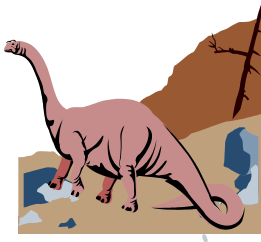
解：



$$P_{\text{CPU}} = \frac{60-(45-35)}{60} \times 100\% = \frac{50}{60} \times 100\% = 83.3\%$$

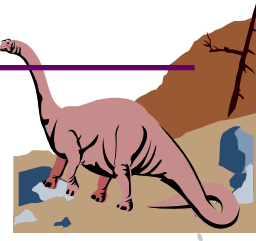
若在单道程序系统中， CPU的利用率是多少？

为了说明多道程序的优点，不妨参考R Turner 提出的例子：某计算机系统，有256KB的主存(不包含操作系统)，一个磁盘，一个终端和一台打印机。同时提交的三个作业分别命名为JOB1、JOB2、JOB3。各作业运行时间分别为5min、15min和10min。它们对资源的使用情况如下表所示：

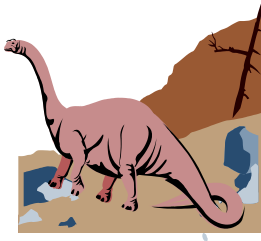


三个作业的执行要求

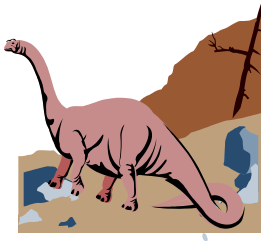
作业名	JOB1	JOB2	JOB3
作业类型	CPU型	I/O型	I/O型
所需主存/KB	50	100	80
所需磁盘	不用	不用	需要
所需终端	不用	需要	不用
所需打印机	不用	不用	需要
运行时间/min	5	15	10



假定JOB1主要使用CPU处理数据，JOB2主要使用终端进行作业的输入，JOB3运行时主要使用磁盘和打印机，后两作业都只需要较少的CPU时间。对于简单批处理情况，这些作业将按顺序执行。JOB1运行5min完成，JOB2在等待5min后，运行15min完成，JOB3在等待20min后开始执行。三个作业全部完成需要30min(这三个作业是一批)。



采用多道程序设计技术，可让这三个作业并行运行。由于它们运行中几乎不同时使用同一资源，所以三个作业可同时运行。JOB1在进行数据处理的同时，JOB2在终端上进行作业输入，JOB3在使用磁盘和打印机。因此，JOB1只需5min完成，JOB2需15min完成，JOB3需10min完成。这样三个作业全部完成的时间只需15min，显然系统处理效率明显提高。

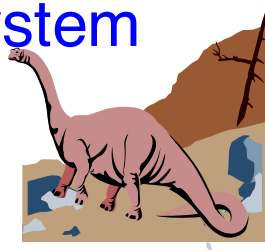


多道程序与单道程序的平均资源利用率

	单　道	多道 (三道作业)
CPU利用率		
主存利用率		
磁盘利用率		
打印机利用率		
全部作业完成时间/min		
吞吐量/(作业 $\cdot h^{-1}$)		
平均周转时间/min		

Time-Sharing Systems-Interactive Computing (1/2)

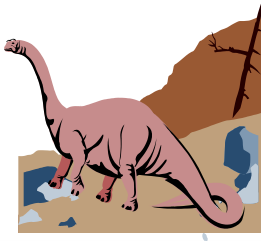
- Interactive computer system provides direct communication between user and the system
  - The user gives instructions to the system directly, waits for immediately results. Response time should be short.
  - When the operating system finishes the execution of one command, it seeks the next “control 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 (2/2)

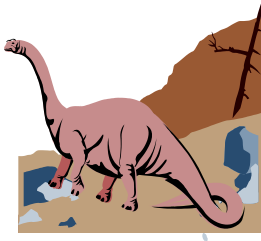
- The CPU is multiplexed among several jobs that are kept in memory and on disk
- To obtain a reasonable response time, a job needs to be swapped in and out of memory to the disk (virtual memory).
- Disk management must be provided
- Sophisticated CPU-scheduling schemes are required for concurrent execution
- 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）



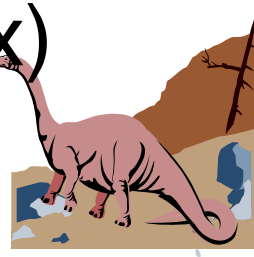
Desktop Systems (1/2)

- *Personal computers* – computer system dedicated to a single user.
- I/O devices – keyboards, mouse, display screens, small printers.
- User convenience and responsiveness.
- Less protection, since its for personal use.



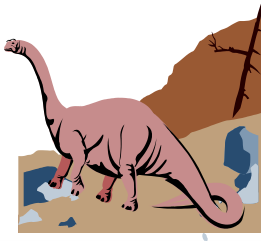
Desktop Systems (2/2)

- Can adopt technology developed for larger operating system.
- Ideas are similar to other OS for multiprogramming and time-sharing systems.
- Individuals have sole use of computer and do not need advanced CPU utilization of protection features.
- May run several different types of operating systems (Windows, MacOS, UNIX, Linux)



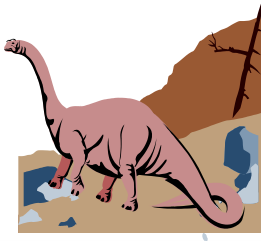
Parallel Systems

- Multiprocessor systems with more than one CPU in close communication.
- *Tightly coupled system (紧耦合系统)* – processors share the bus, a clock, and sometimes memory and peripheral devices.
- Communication usually takes place through the shared memory.



Parallel Systems (Cont.)

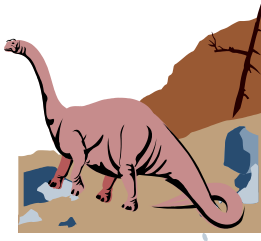
- Advantages of parallel system:
  - Increased throughput: Gets more jobs done
  - Economy of scale: Because of resource sharing, multiprocessor systems is cheaper than multiple single processor systems
  - Increased reliability: the failure of one processor will not halt the whole system
    - ✓ Graceful degradation
    - ✓ Fault-tolerant systems



Symmetric vs. Asymmetric Multiprocessing of Parallel Systems

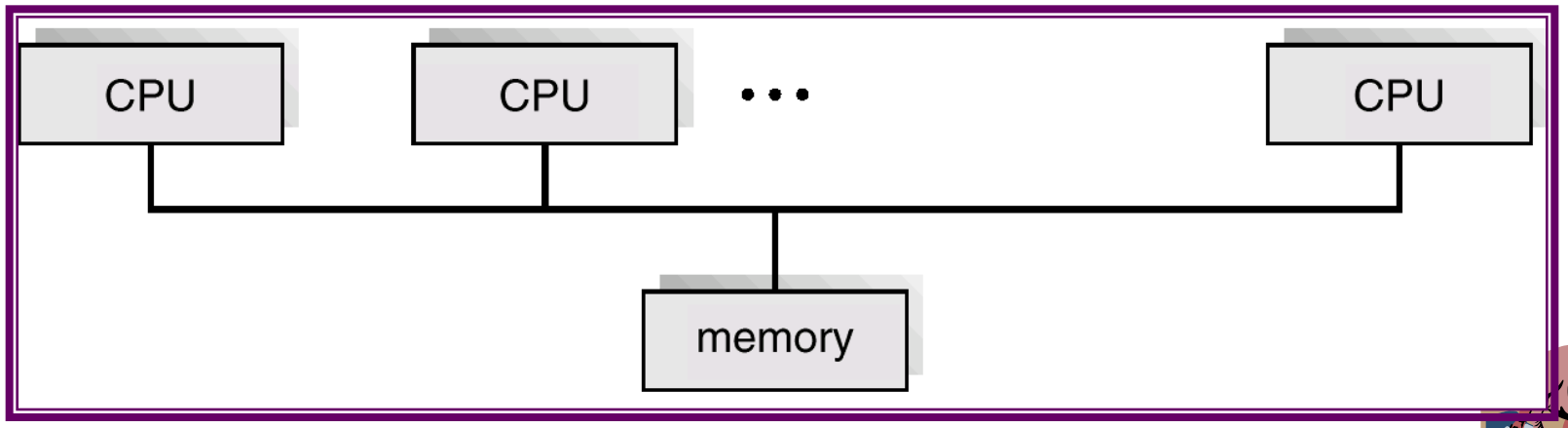
- Symmetric Multi-processors - common identical copy of OS on all processors.

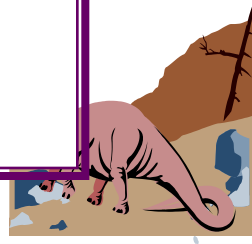
- Asymmetric Multi-processors - each processor is supplied a specific task, typical in master-slave configurations.



Symmetric Multi-Processing (SMP)

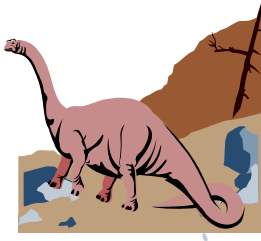
- Many processes can run at once without performance deterioration.
- Must carefully control I/O to ensure that the data reach the appropriate processor
- May result in inefficiencies
- Most modern operating systems support SMP





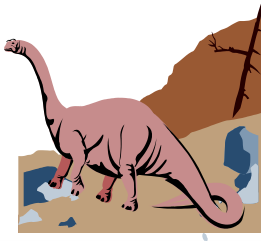
Asymmetric Multi-Processing

- Each processor is assigned a specific task.
- A master processor controls the system. The other processors either look to the master for instructions or have predefined tasks.
- This defines a master-slave relationship
- master processor schedules and allocated work to slave processors.
- More common in extremely large systems



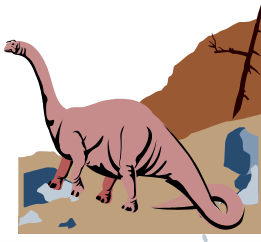
Distributed Systems

- Distribute the computation among several physical processors.
- *Loosely coupled system (松耦合系统)* – each processor has its own local memory; processors communicate with one another through various communications lines, such as high-speed buses, telephone lines, Ethernet, or optical fibers.



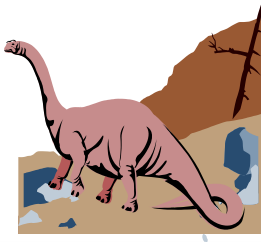
Distributed Systems (cont.)

- A distributed system is a collection of physically *separate*, possible *heterogeneous* computer systems that are *networked* to provide the users with access to the various resources that the system maintains
- Advantages of distributed systems.
  - Resources sharing
  - Computation speed up through load sharing
  - Increased reliability
  - Communications

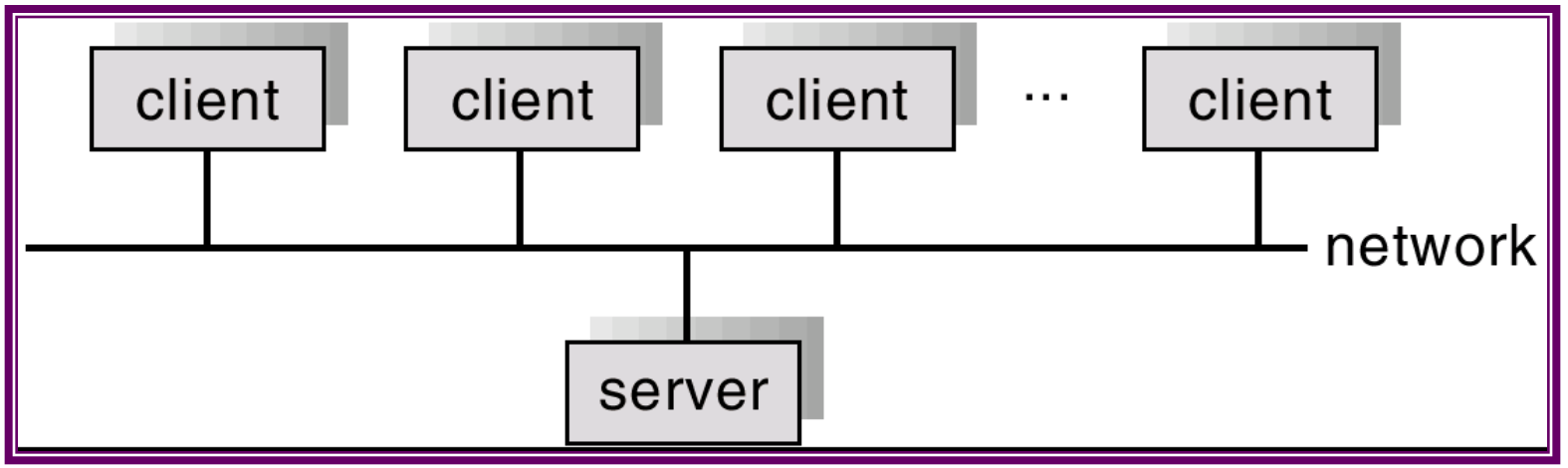


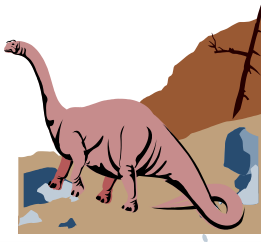
Distributed Systems (cont.)

- Requires networking infrastructure.
- Local area networks (LAN) or Wide area networks (WAN)
- May be either client-server or peer-to-peer systems.



General Structure of Client-Server



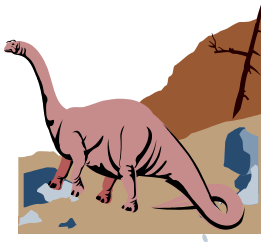


Clustered Systems

- A clustered system has two or more individual systems shared storage and closely linked via LAN networking
- The key of clustered system is high availability.
- Asymmetric Clustering:
  - One machine is in hot-standby mode while others are running applications.
  - The hot-standby machine (*i.e.*, does nothing but) monitors other machines and becomes active if one server fails.
- Symmetric Clustering:
  - Two or more hosts are running applications and monitor each other.
  - This is more efficient as it uses all available hardware

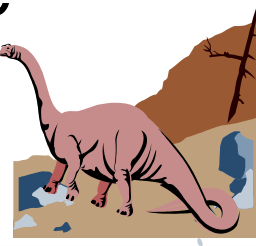
Real-Time Systems

- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.



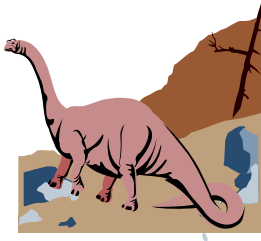
Real-Time Systems (Cont.)

- A real-time operating system has well-defined, fixed time constraints.
- Processing *must* be done within the defined constraints, or the system will fail.
- Real-Time systems may be either *hard* or *soft* real-time.
  - *Hard Real-Time Systems* guarantee that critical tasks be completed on time.
  - *Soft Real-Time Systems* prioritize critical tasks. That is, a critical task get priority over other tasks, and retains that priority until it completes.
- Embedded systems almost always run real-time operating systems

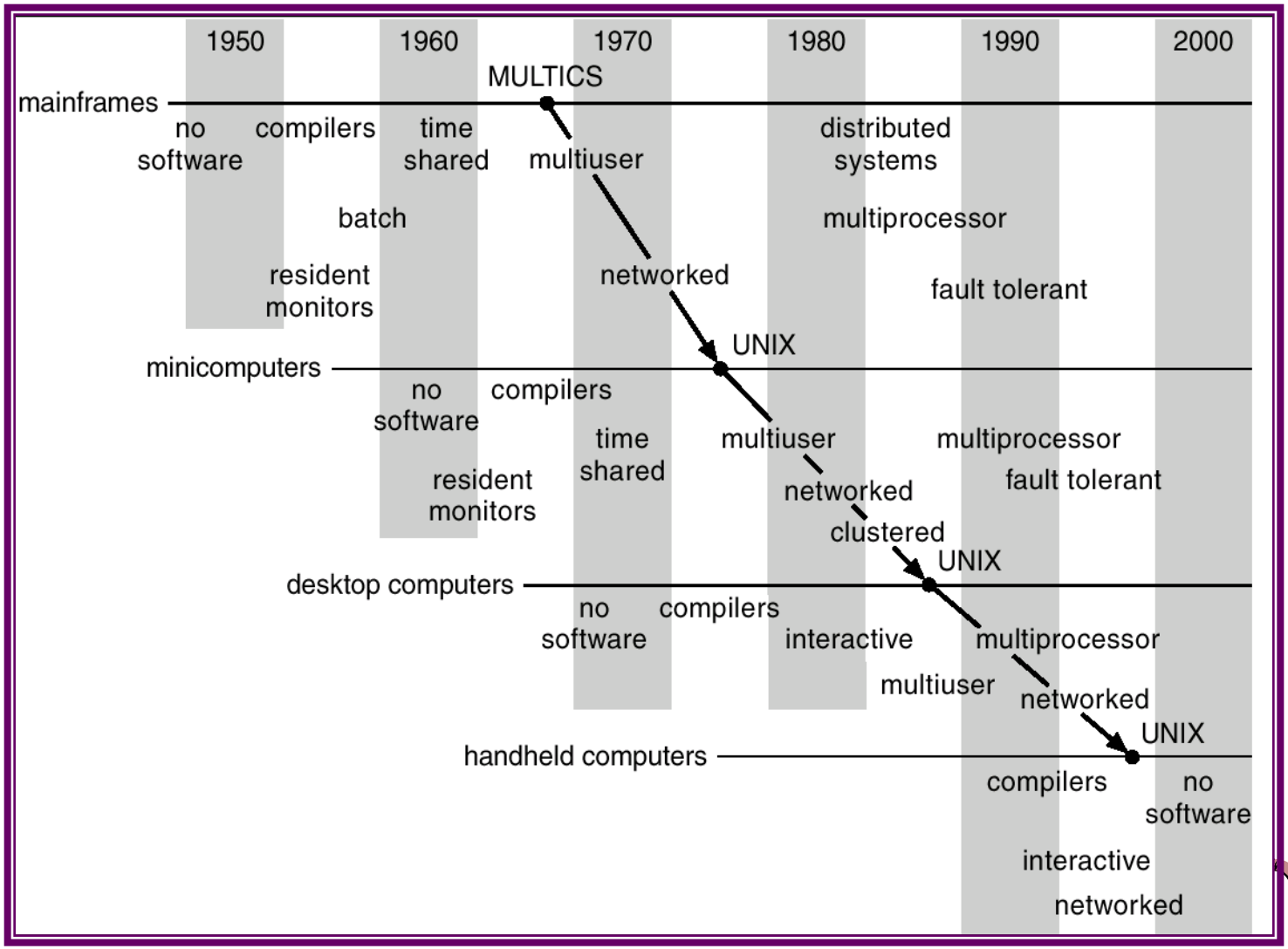


Handheld Systems

- Personal Digital Assistants (PDAs)
- Cellular telephones
- Smart phones
- Issues:
  - Limited memory
  - Slow processors
  - Small display screens.



Migration of Operating-System Concepts and Features



Computing Environments

- Traditional computing
- Web-Based Computing
- Embedded Computing
- Pervasive Computing
- Cloud Computing
- ……