上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

SJTU

L1-2. 课程简介

宋卓然

上海交通大学计算机系

songzhuoran@sjtu.edu.cn

从 Hello World 说起

```c
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
    return 0;
}
```

```
bash$ gcc hello.c -o hello

# 运行一个hello world程序
bash$./hello
Hello World!

# 同时启动两个hello world程序
bash$ ./hello & ./hello
[1] 144
Hello World!
Hello World!
[1]+ Done        ./hello
```

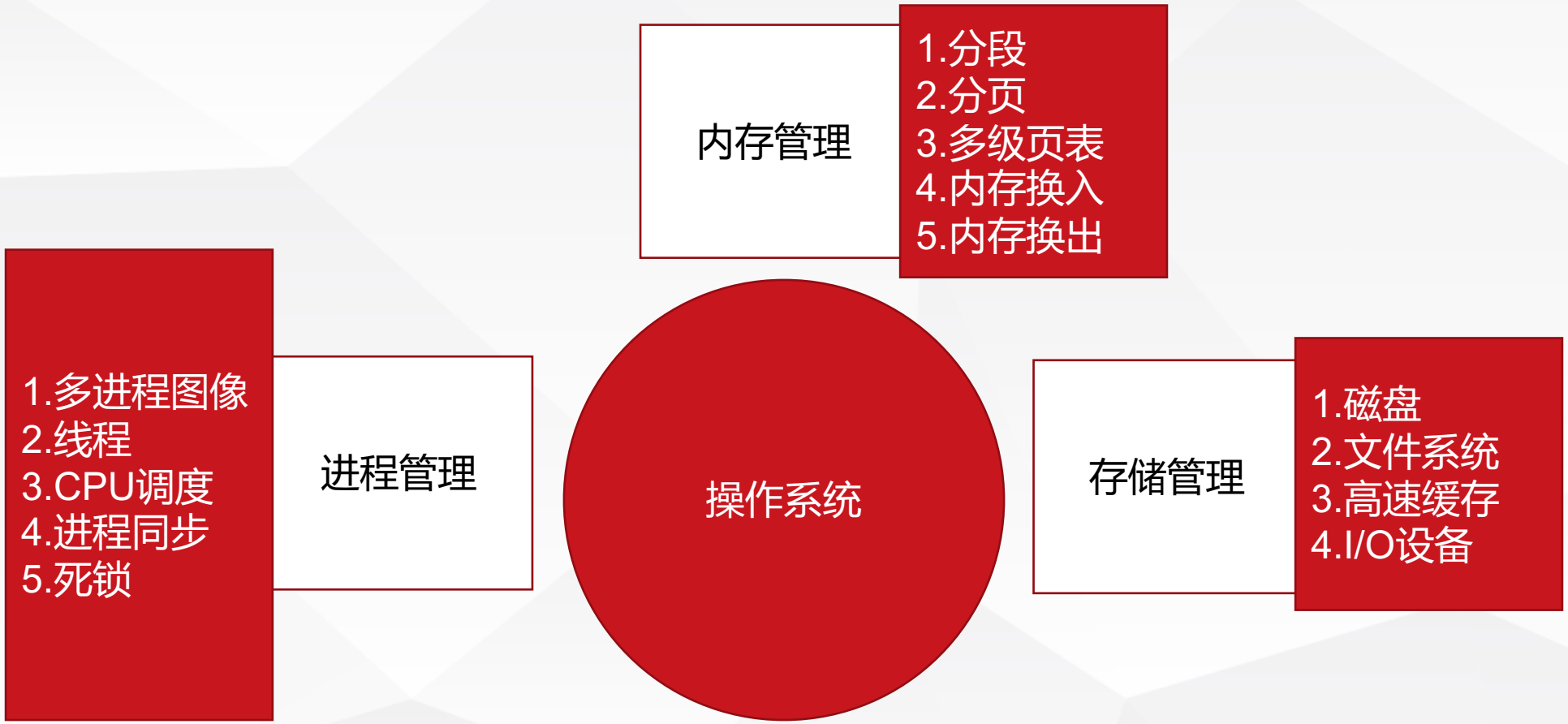
运行Hello时，操作系统的作用是？

操作系统考虑的一些问题

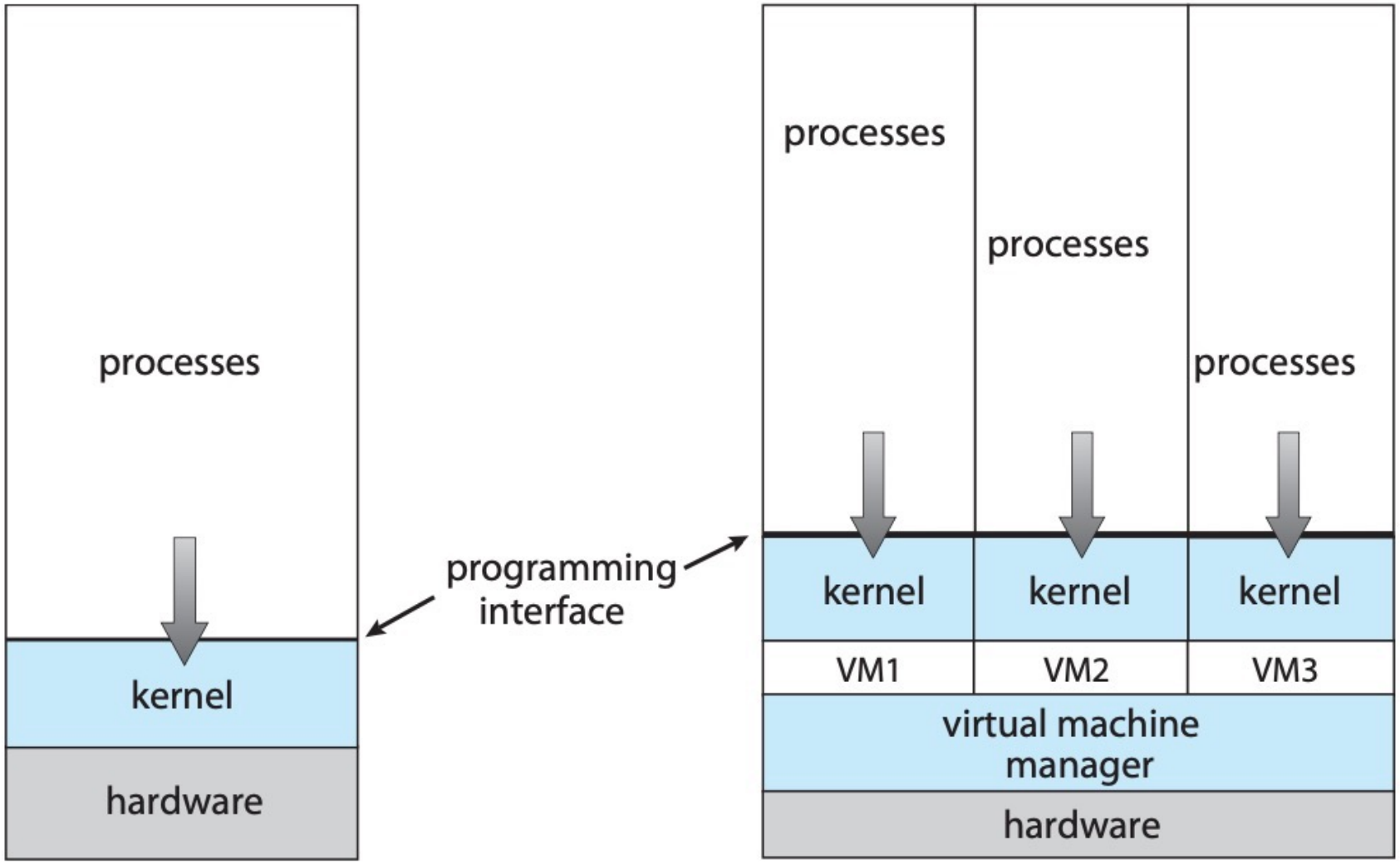
- hello 这个可执行文件存储在什么位置?是如何存储的?
- hello 这个可执行文件是如何加载到 CPU 中运行?
- hello 这个可执行文件是如何将"Hello World!"这行字输出到屏幕?
- 两个hello 程序同时运行的过程中如何在一个 CPU 中运行?

操作系统需要：1、服务应用；2、管理应用

操作系统课程总览


内存管理
1.分段
2.分页
3.多级页表
4.内存换入
5.内存换出
1.多进程图像
2.线程
3.CPU调度
4.进程同步
5.死锁
进程管理
操作系统
存储管理
1.磁盘
2.文件系统
3.高速缓存
4.I/O设备

单进程 vs 多进程


processes
kernel
hardware
programming interface
processes
processes
processes
kernel
kernel
kernel
VM1
VM2
VM3
virtual machine manager
hardware
(a)
(b)

CPU调度

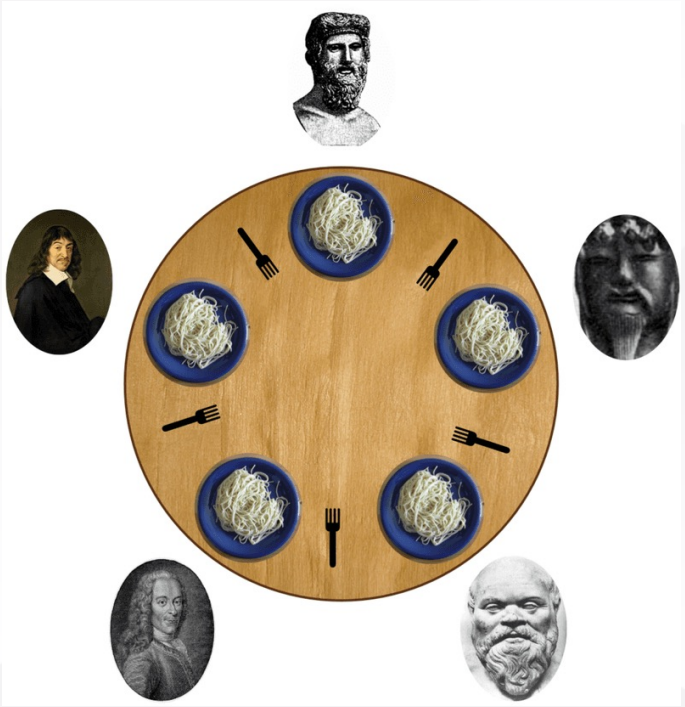
- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-First (SJF) Scheduling
- Priority Scheduling
- Round-Robin Scheduling（时间片）
- Multilevel Queue Scheduling
- Multilevel Feedback Queue Scheduling

进程同步

- 哲学家就餐问题

假设有五位哲学家围坐在一张圆形餐桌旁，做以下两件事情之一：吃饭，或者思考。吃东西的时候，他们就停止思考，思考的时候也停止吃东西。假设哲学家必须用两只餐叉吃东西。他们只能使用自己左右手边的那两只餐叉。

哲学家从来不交谈，这就很危险，可能产生**死锁**，每个哲学家都拿着左手的餐叉，永远都在等右边的餐叉（或者相反）。



避免死锁

- 银行家算法

3种资源：A（10）、B（5）、C（7）

Snapshot at time T_0:

	Max	Allocation	Need	Available
	A B C	A B C	A B C	A B C
P_0	7 5 3	0 1 0	7 4 3	3 3 2
P_1	3 2 2	2 0 0	1 2 2	
P_2	9 0 2	3 0 2	6 0 0	
P_3	2 2 2	2 1 1	0 1 1	
P_4	4 3 3	0 0 2	4 3 1	

< P1, P3, P0, P2, P4> 是安全的调度方案

存储结构

- CPU从内存（main memory）加载指令执行、读写数据
  - 动态随机访问内存（dynamic random access memory，DRAM）
  - 只读内存（ROM）：游戏盒
  - 电可擦可编程只读内存（EEPROM）：擦写次数有限，工厂安装的程序
- 冯诺依曼架构与非冯诺依曼架构
- 易失性与非易失性存储器

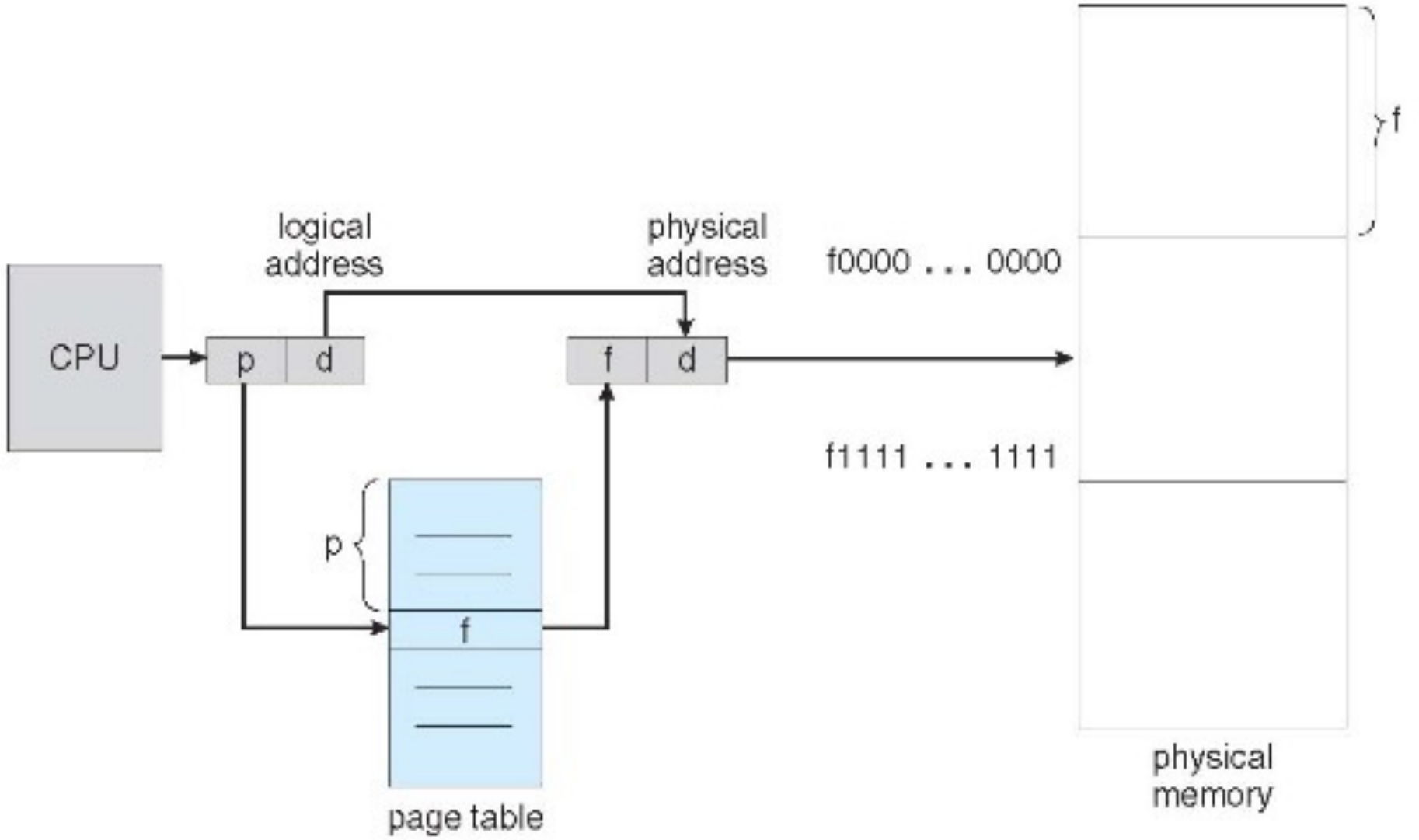
高速缓存

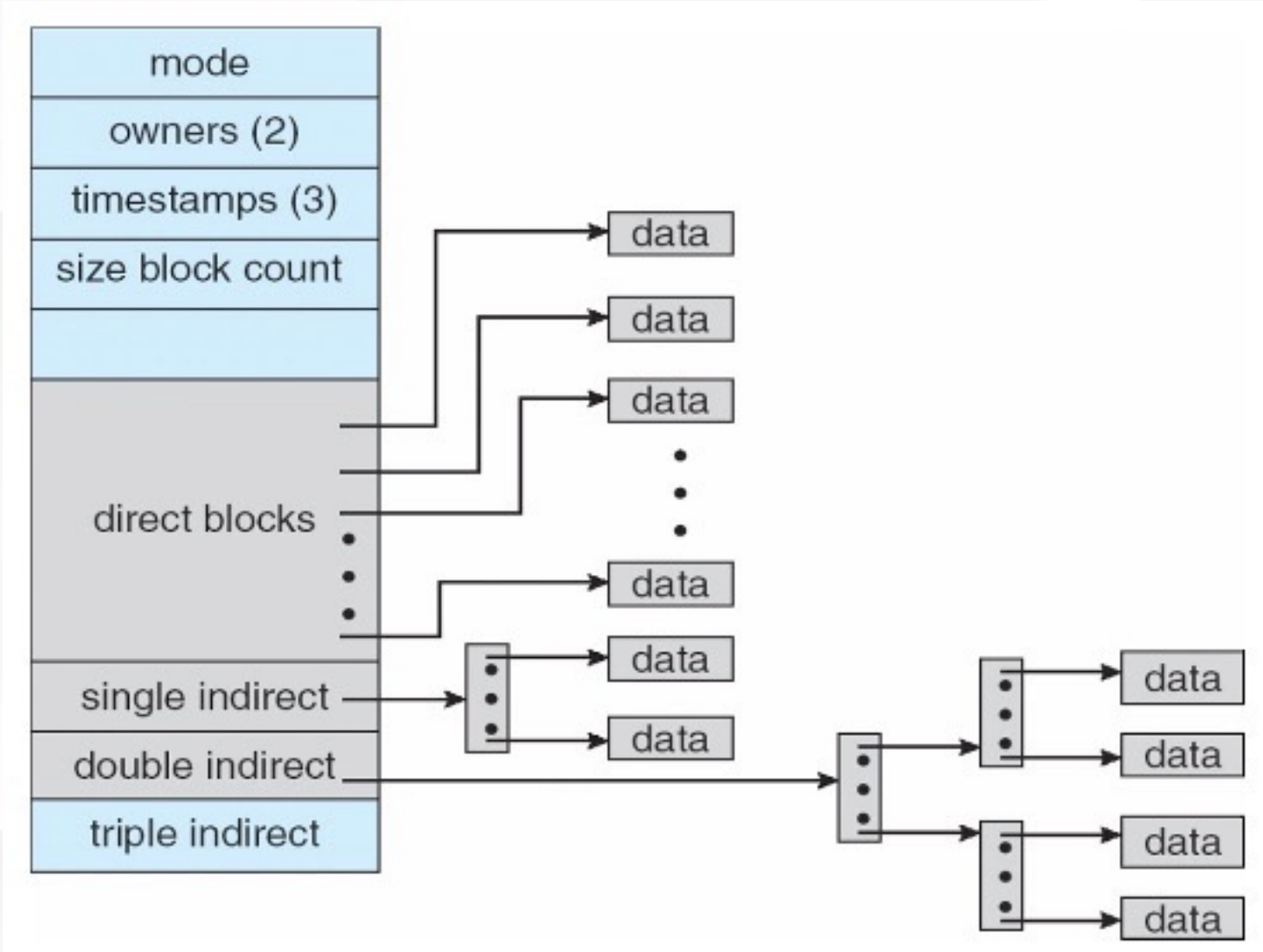
级别	1	2	3	4	5
名称	寄存器	高速缓存	内存	固态盘	硬盘
典型尺寸	< 1KB	< 16MB	< 64GB	< 1TB	< 10TB
实现技术	具有多个端口CMOS的定制内存	片上或片外CMOS SRAM	CMOS SRAM	闪存	硬盘
访问时间（ns）	0.25 ~ 0.5	0.5 ~ 25	80 ~ 250	25 000 ~ 50 000	5 000 000
带宽（MB/sec）	20 000 ~ 100 000	5 000 ~ 10 000	1 000 ~ 5 000	500	20 ~ 150
由谁管理	编译器	硬件	操作系统	操作系统	操作系统
由谁备份	高速缓存	内存	硬盘	硬盘	硬盘或磁带

存储定义与符号

- 基本单位：位或比特（bit）---0/1
- 字节（byte）：8bit
- 字（word）：一个或多个字节
- 千字节（kilobyte，KB）：1024个字节
- 兆字节（megabyte，MB）：1024^2个字节

内存管理

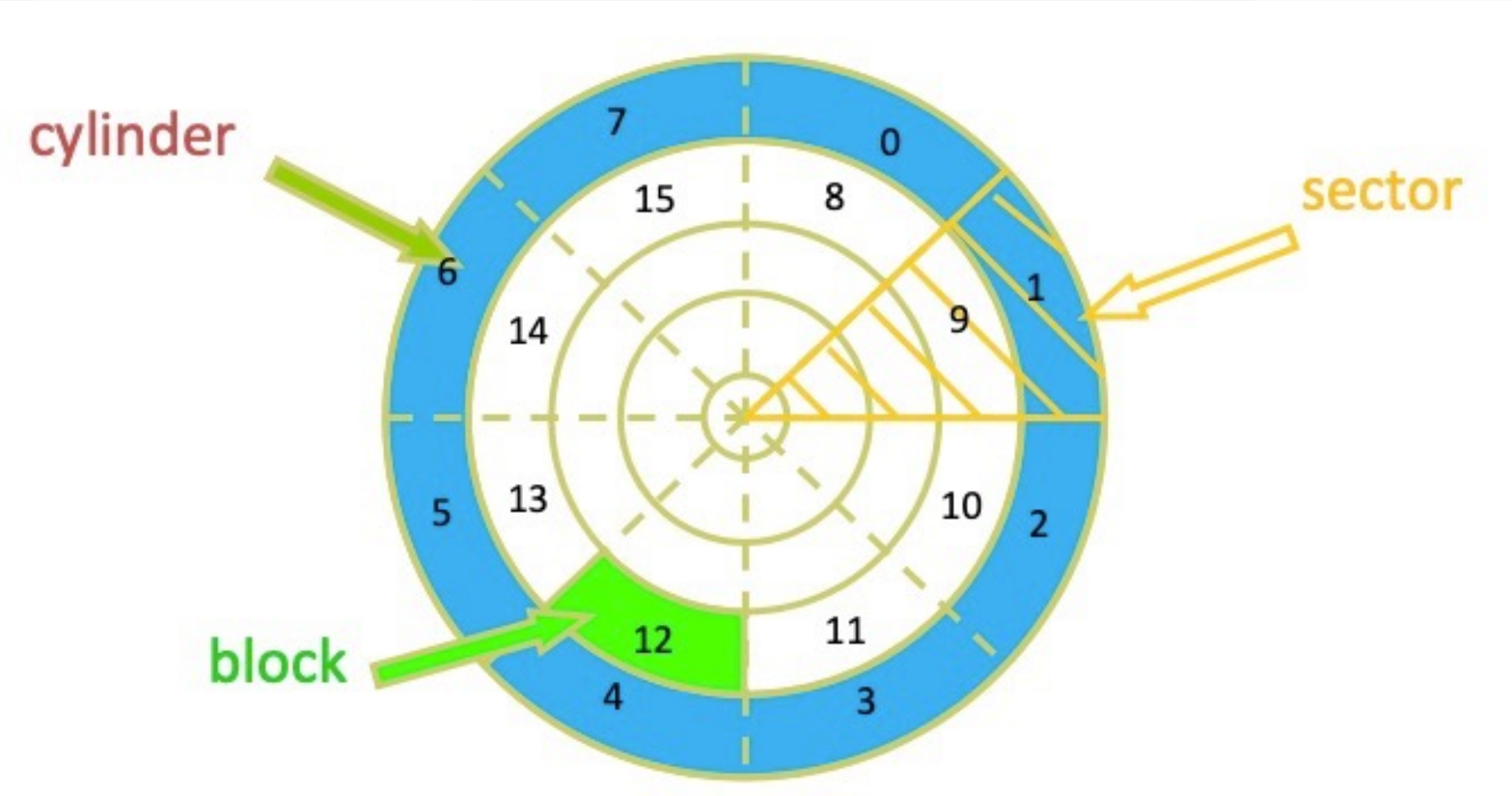

CPU
logical address
p
d
physical address
f
d
f0000 . . . 0000
f1111 . . . 1111
f
p
f
page table
physical memory


mode
owners (2)
timestamps (3)
size block count
direct blocks
single indirect
double indirect
triple indirect
data
data
data
data
data
data
data
data
data
data

文件系统

- 存储程序、数据
  - 字符、数值、二进制
- 操作系统负责文件管理
  - 创建、删除文件
  - 创建、删除目录
  - 提供文件和目录的操作原语
  - 映射文件到外存

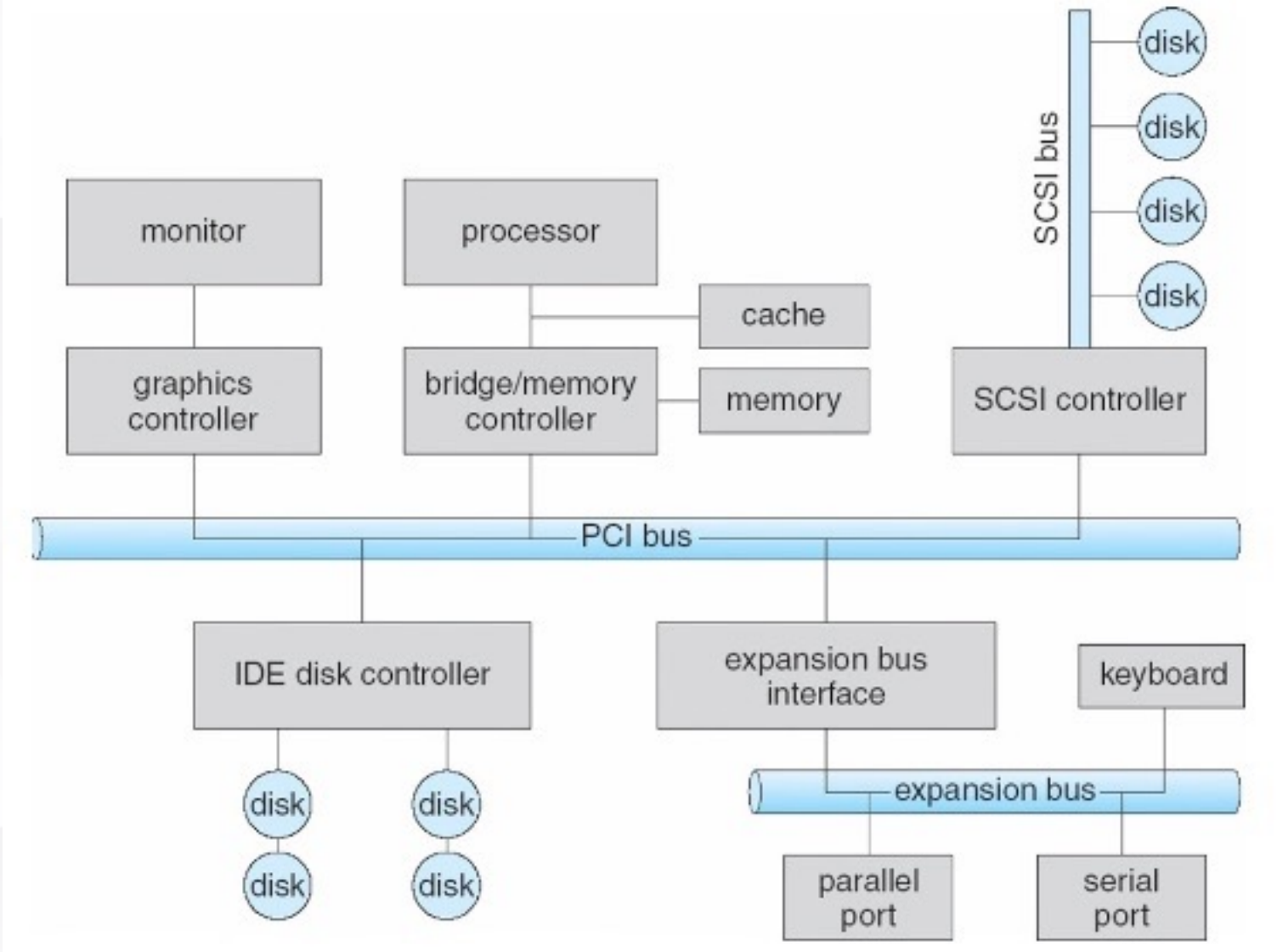
磁盘



磁盘

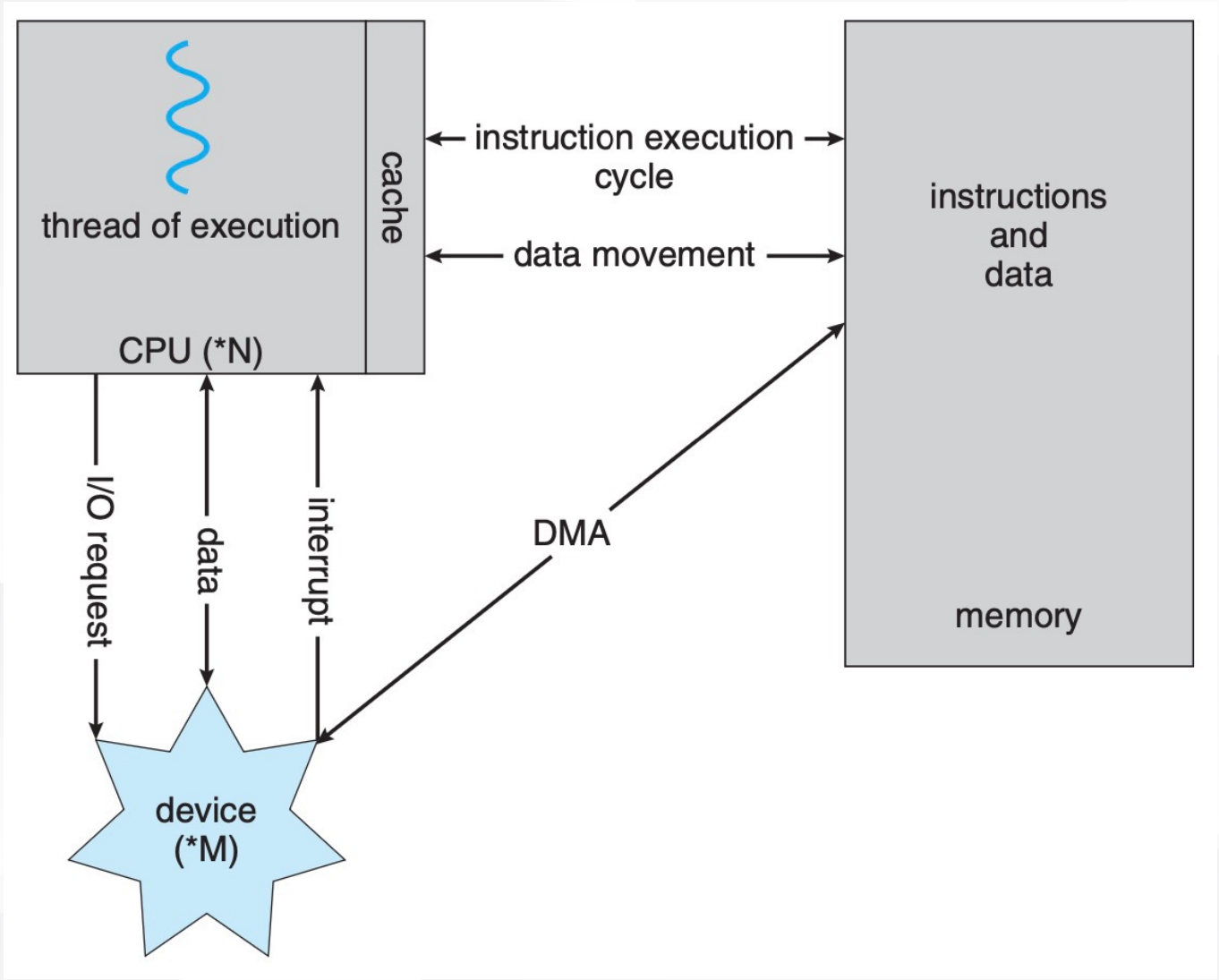
- 磁头、柱面、磁道、扇区
  - 将磁头移动到指定的磁道上
  - 磁道旋转，转到相应的扇区
  - 旋转，磁生电，读取数据
- 机械行为
- 操作系统负责磁盘管理
  - 空闲空间管理
  - 存储空间分配
  - 磁盘调度

I/O系统


monitor
processor
cache
graphics controller
bridge/memory controller
memory
SCSI bus
disk
disk
disk
disk
SCSI controller
PCI bus
IDE disk controller
expansion bus interface
keyboard
disk
disk
disk
disk
expansion bus
parallel port
serial port

I/O系统

- 设备驱动程序加载设备控制器的寄存器
- 设备控制器采取操作（读、写、打印）
- 控制器向内存传输数据，一旦传输完成，发出中断，通知设备驱动程序
- 设备驱动程序返回控制到CPU
- 对于大量数据移动，采用直接内存访问（direct memory access，DMA）
  - 无需CPU干预



计算机系统的体系结构

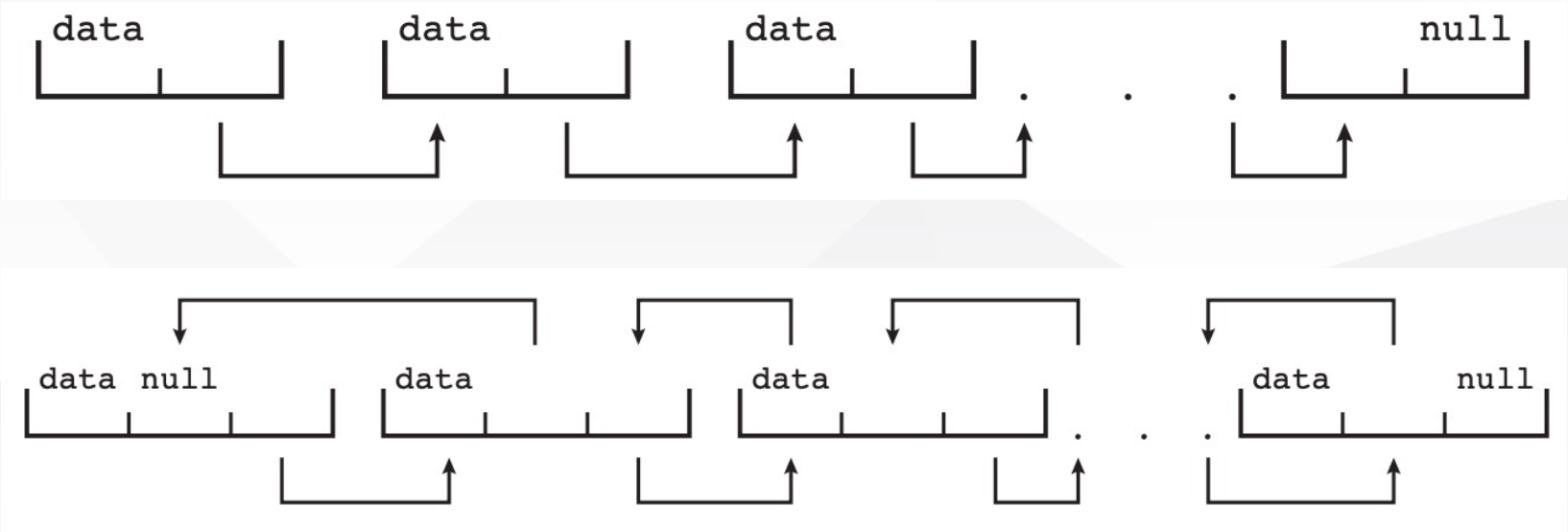
- 单处理器系统
  - 只有一个主CPU
- 多处理器系统
  - 多个CPU，共享总线、时钟、内存、外设
- 多处理器系统的优点
  - 增加吞吐量
  - 规模经济
  - 增加可靠性

定时器

- 防止用户程序陷入死循环、执行时间过长
  - 固定
  - 可变
- 直到计数器值为0，发出中断

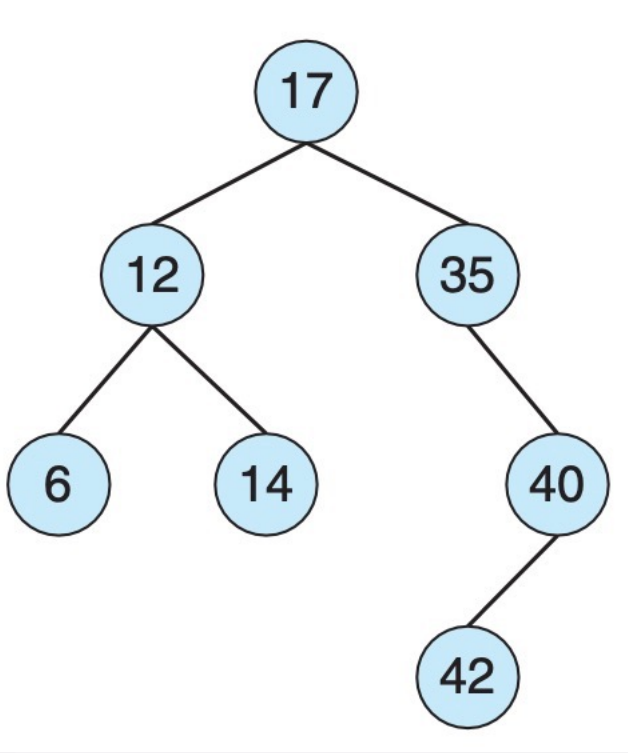
内核数据结构

- 列表：一组数据表示成序列
- 链表
  - 单向链表
  - 双向链表



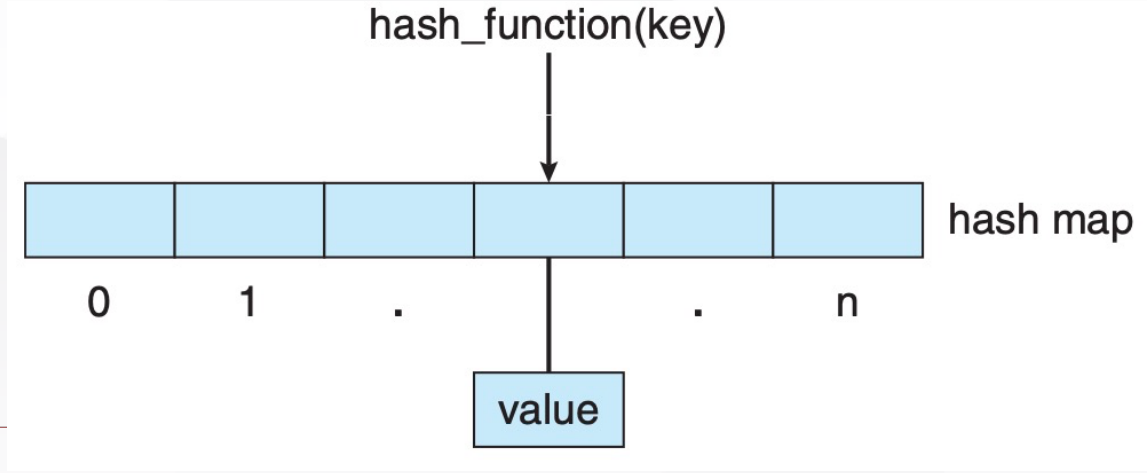
内核数据结构

- 堆栈：后进先出
  - 压入
  - 弹出
- 队列：先进先出
- 树
  - 二叉树：一个父节点最多有两个子节点
  - 二叉查找树
  - 平衡二叉查找树（CPU调度）

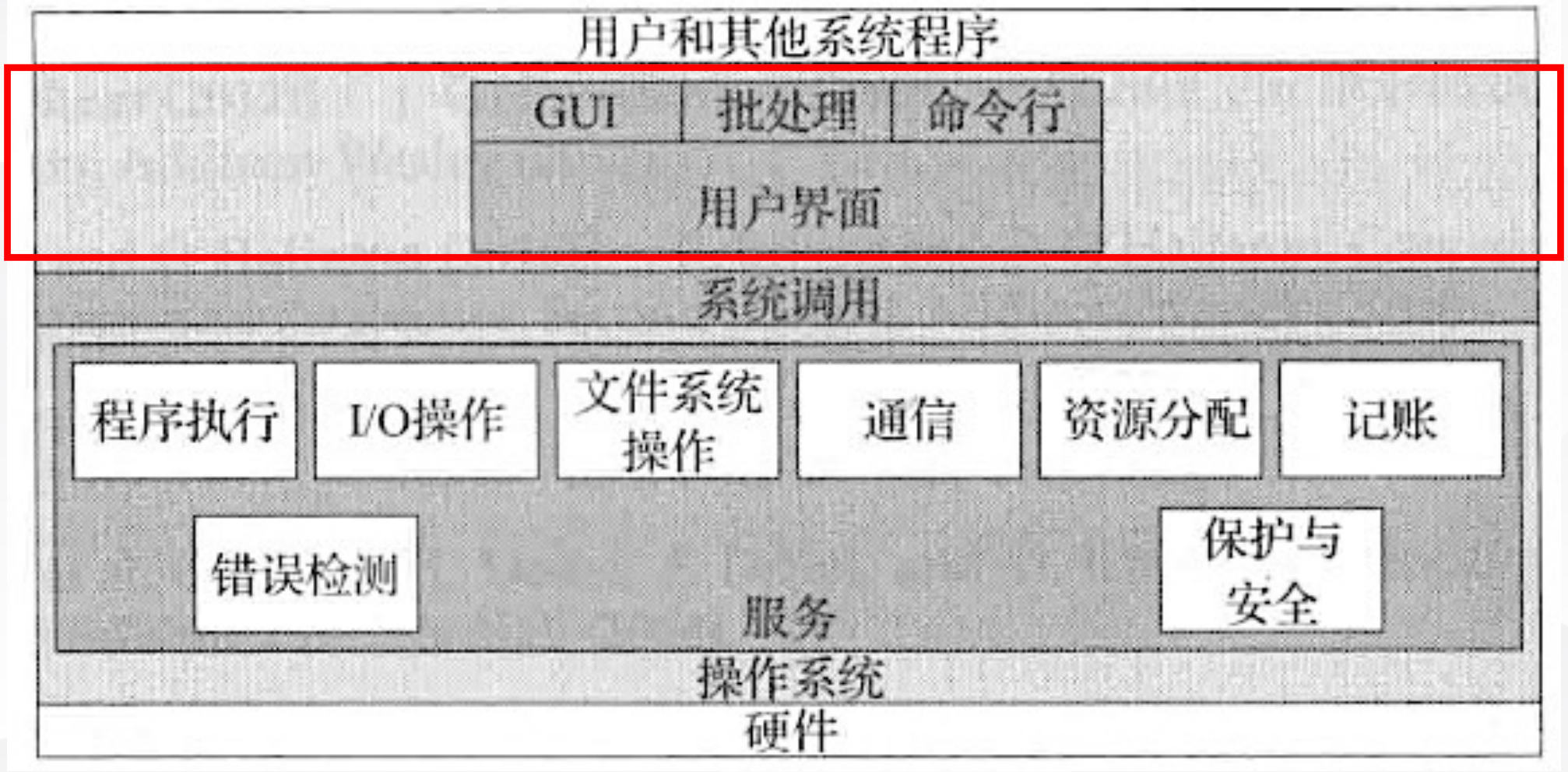


内核数据结构

- 哈希函数
  - 将一个数据作为输入，对此进行数值运算，返回一个数值
  - y=H(key)
  - H(key)=key MOD p 除留取余
- 高效：O(1)复杂度
- 哈希碰撞



操作系统服务级结构

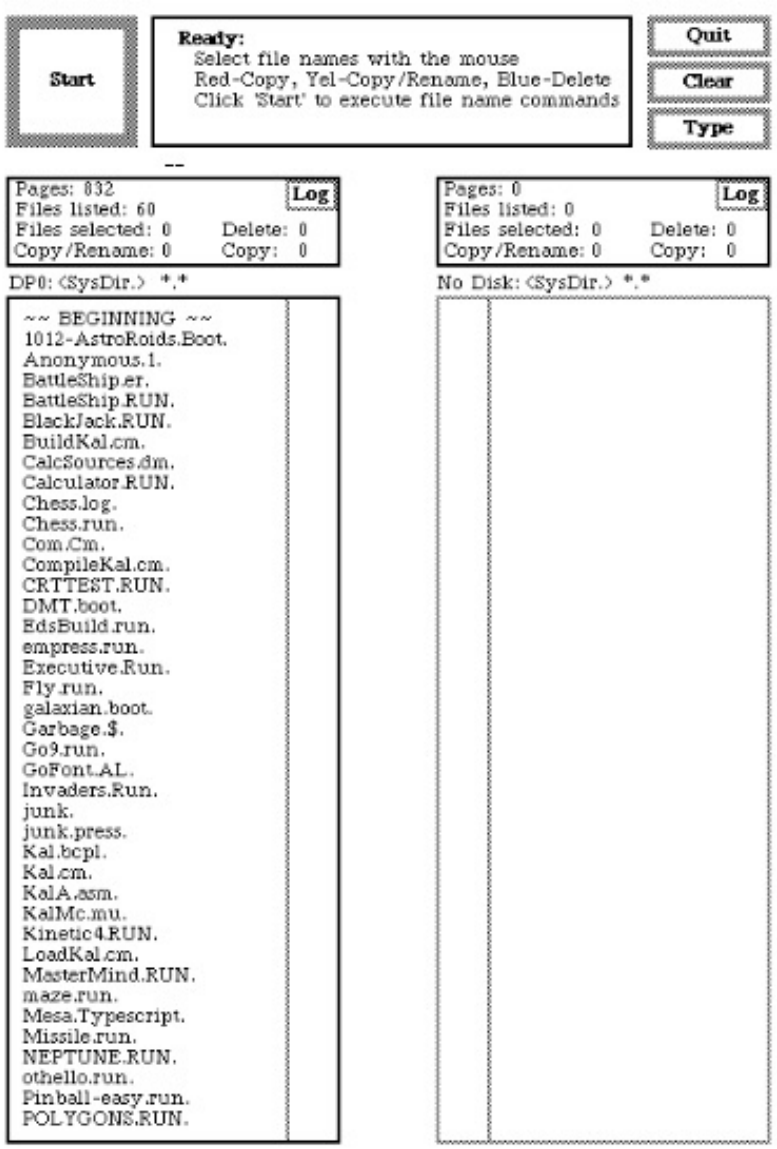


操作系统为程序和用户提供程序和服务执行提供环境

用户界面

- 命令行界面
  - 采用文本命令，键入
  - 批处理界面，命令及控制命令的执行编成文件以便执行
  - 图形用户界面，视窗系统，引入I/O设备：鼠标、键盘、触摸屏
    - 桌面
    - 图标
    - 文件
    - 文件夹

First GUI (1973)



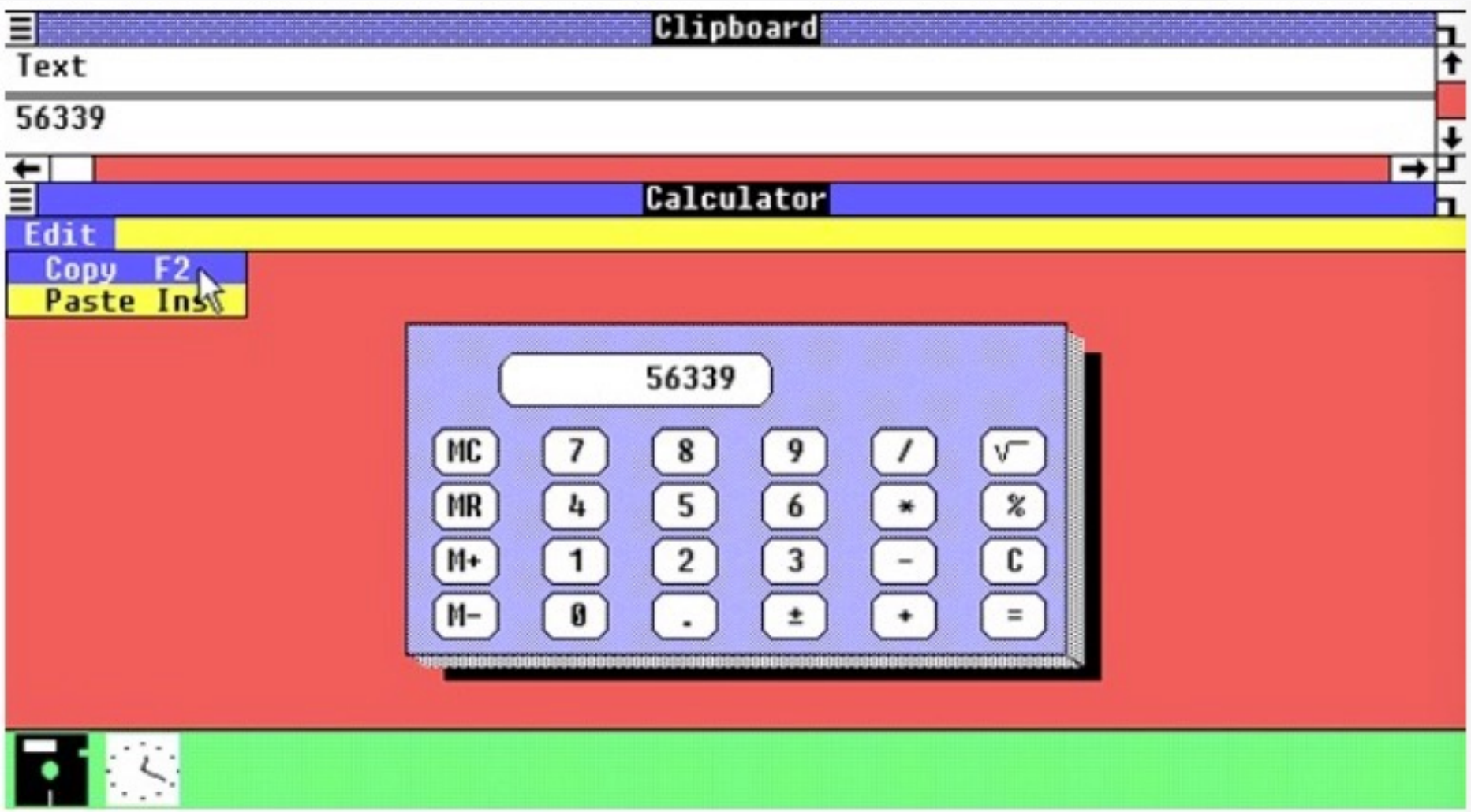
The first appeared on the Xerox Alto computer in 1973.

Mac OS System 1.0 (1984)

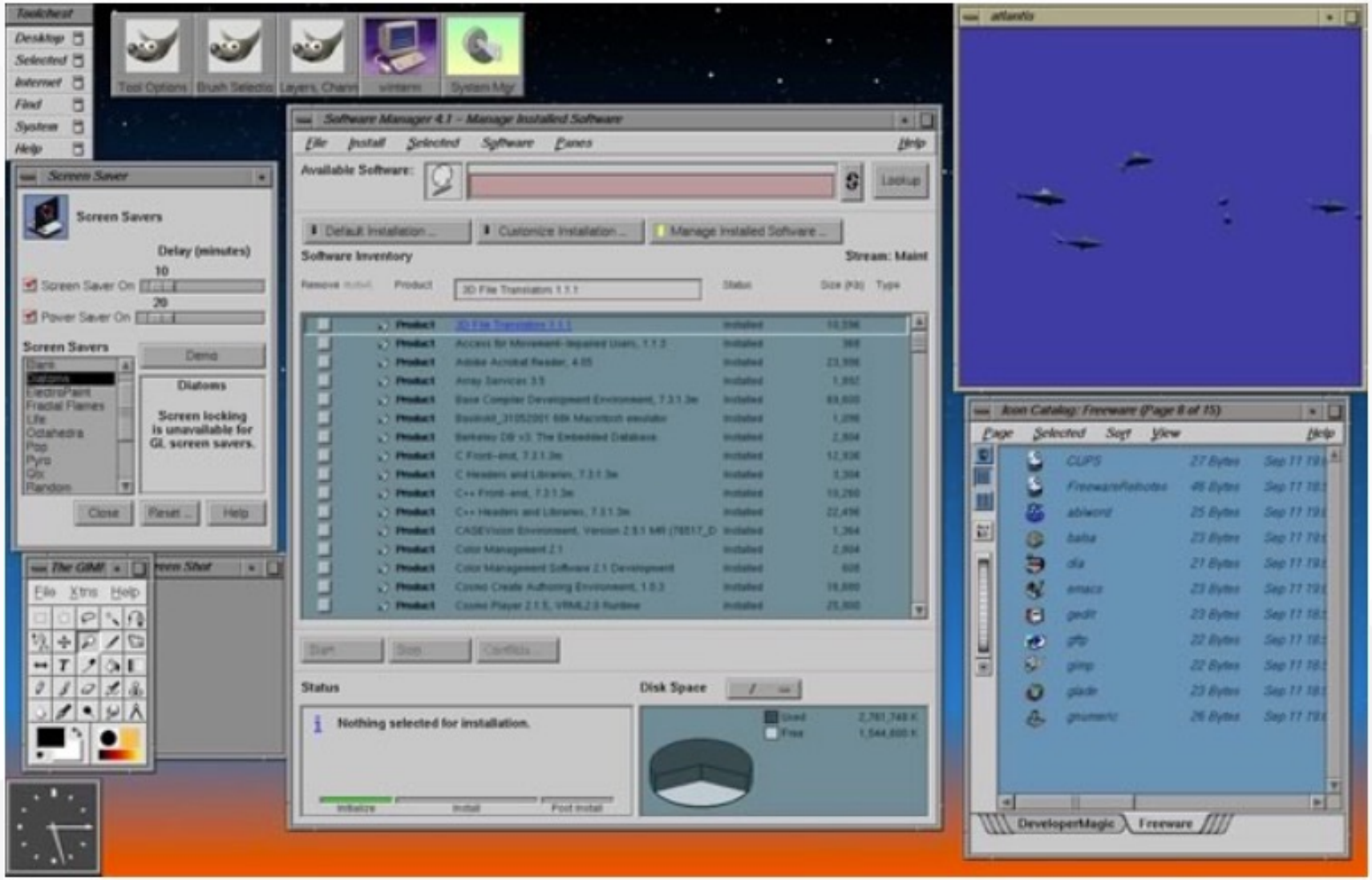
Amiga Workbench 1.0 (1985)

第一个带颜色图形渲染的GUI

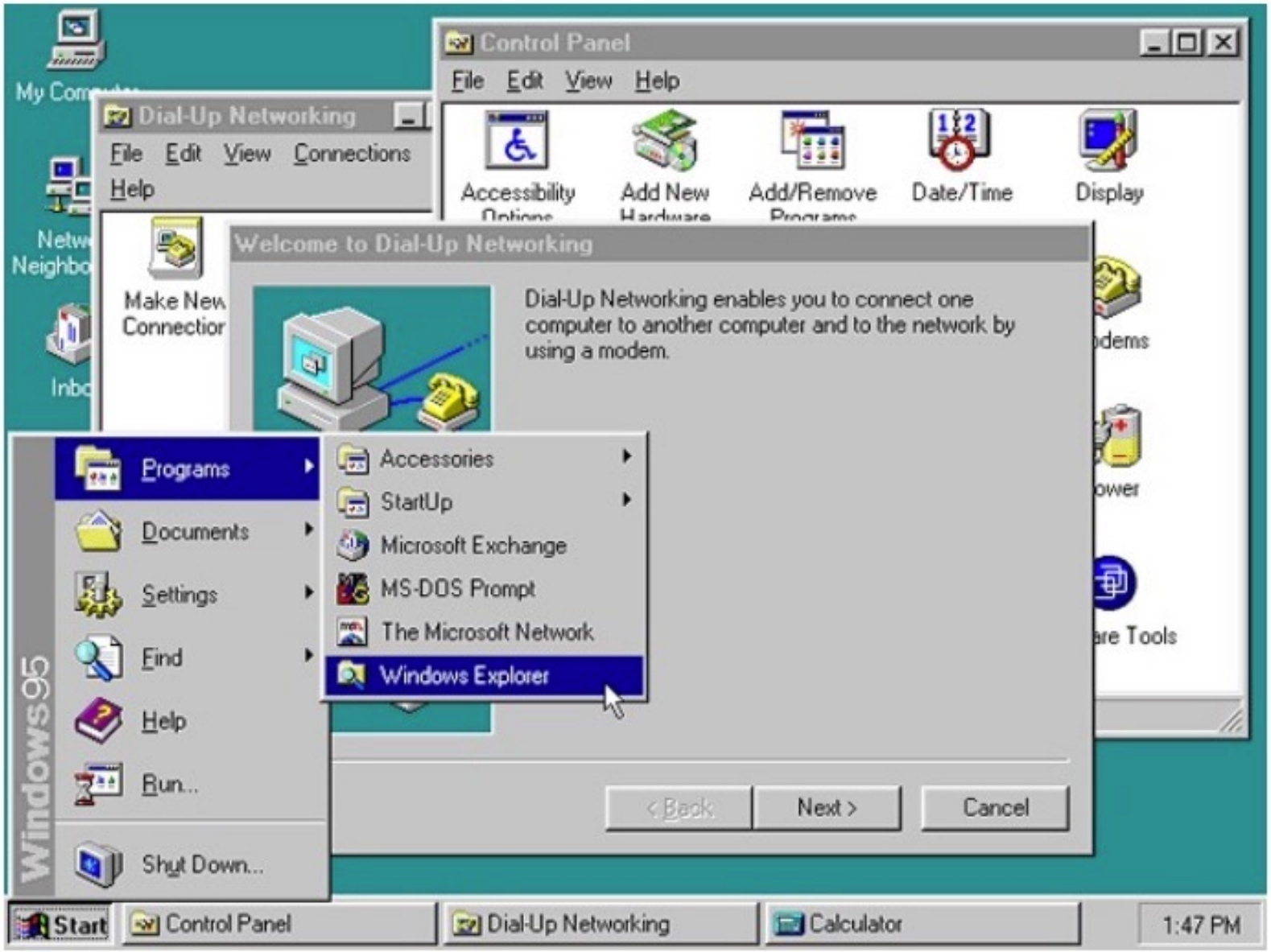
Windows 1.0x (1985)



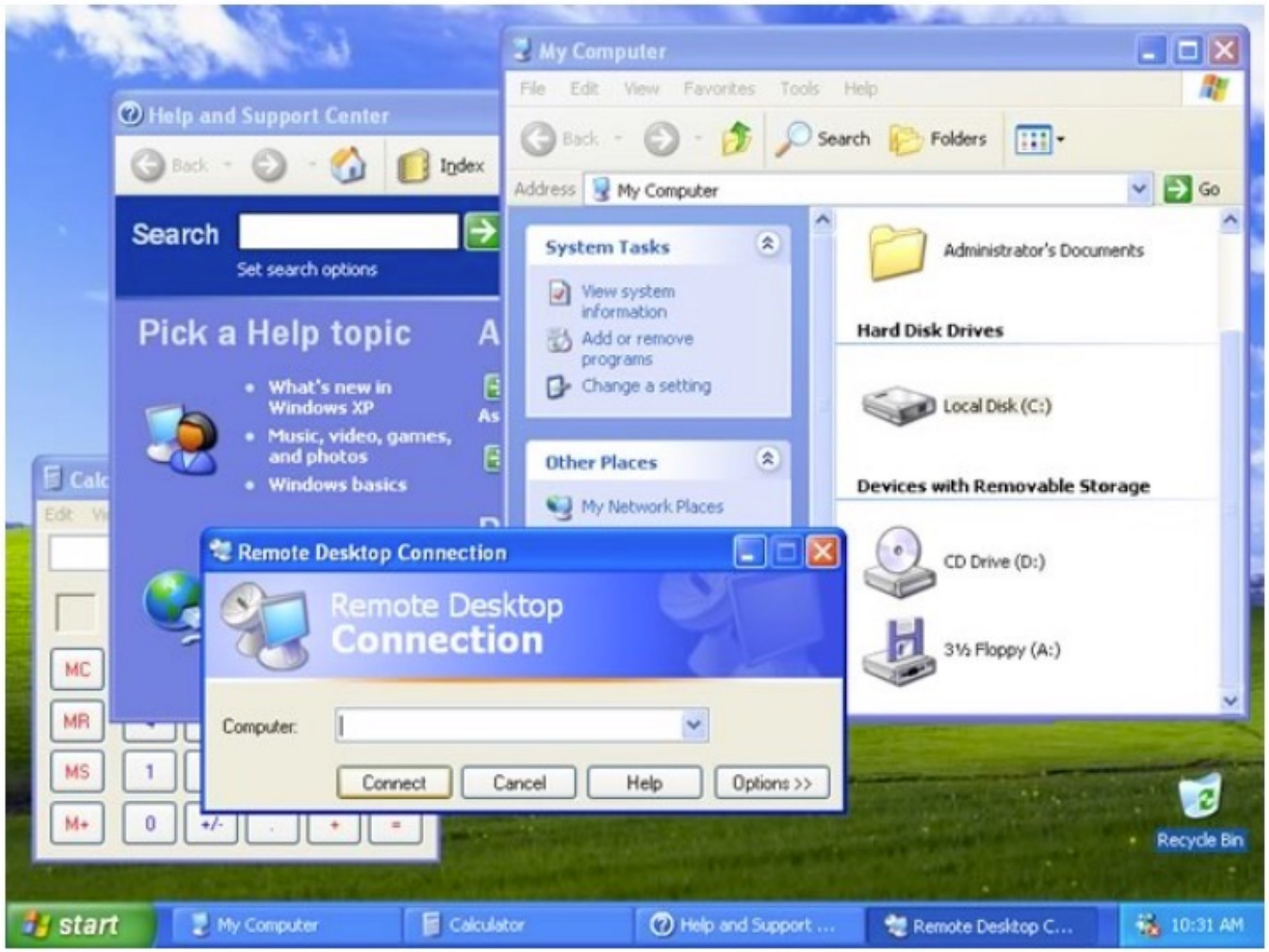
IRIX 3 (released in 1986, first release 1984)



Windows 95 (1995)



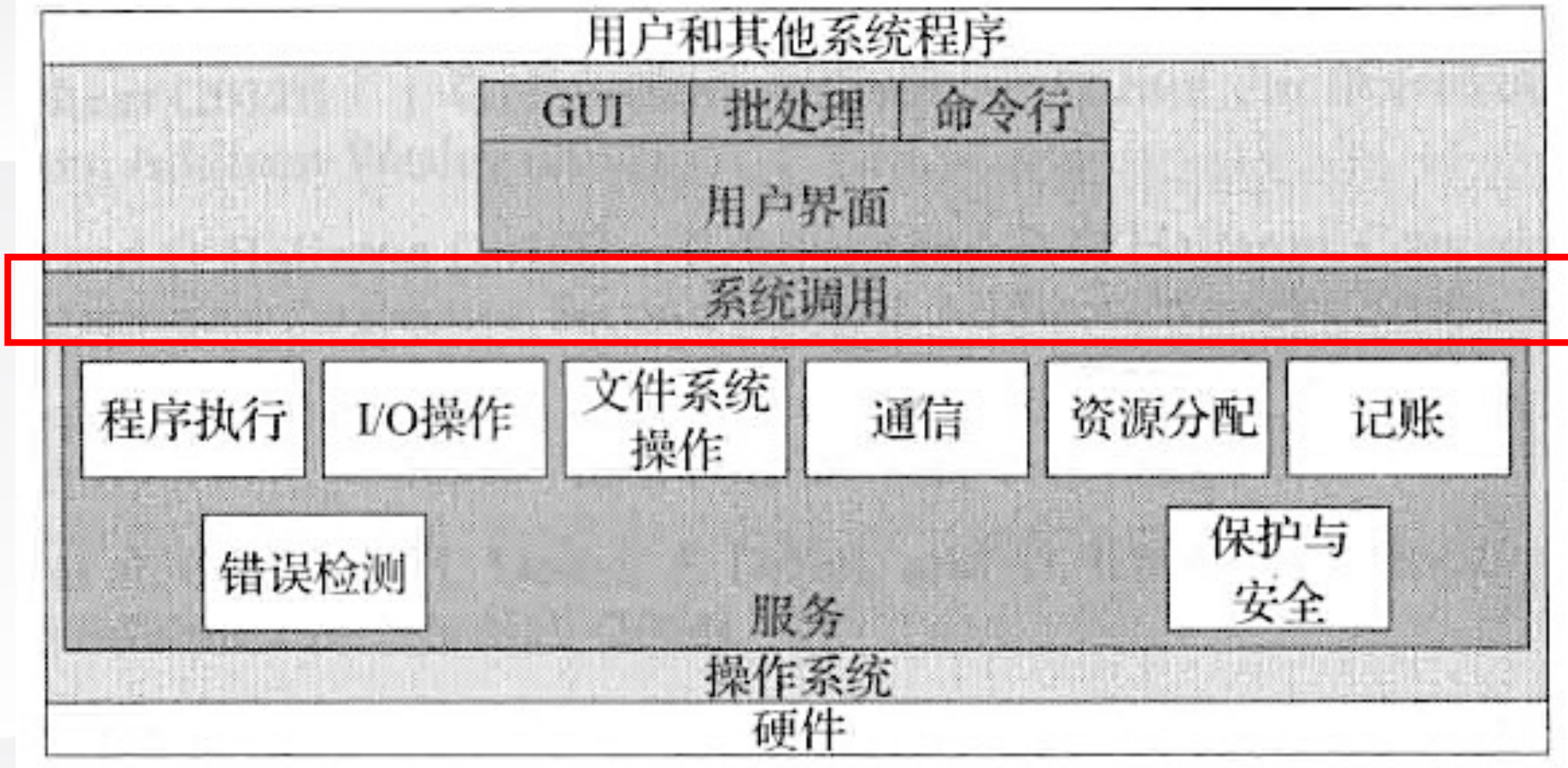
Windows XP (released in 2001)



Mac OS X Leopard (released in 2007)

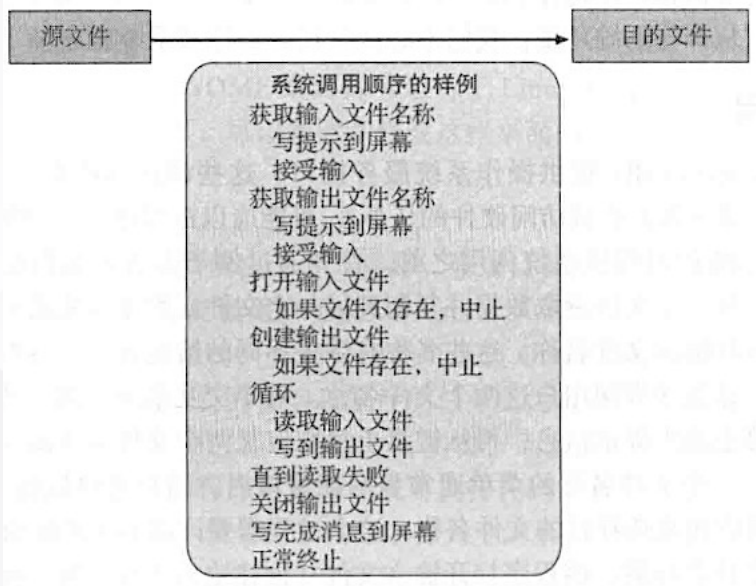
Windows 10 (July 2015)

操作系统服务级结构


用户和其他系统程序
GUI
批处理
命令行
用户界面
系统调用
程序执行
I/O操作
文件系统操作
通信
资源分配
记账
错误检测
保护与安全
服务
操作系统
硬件

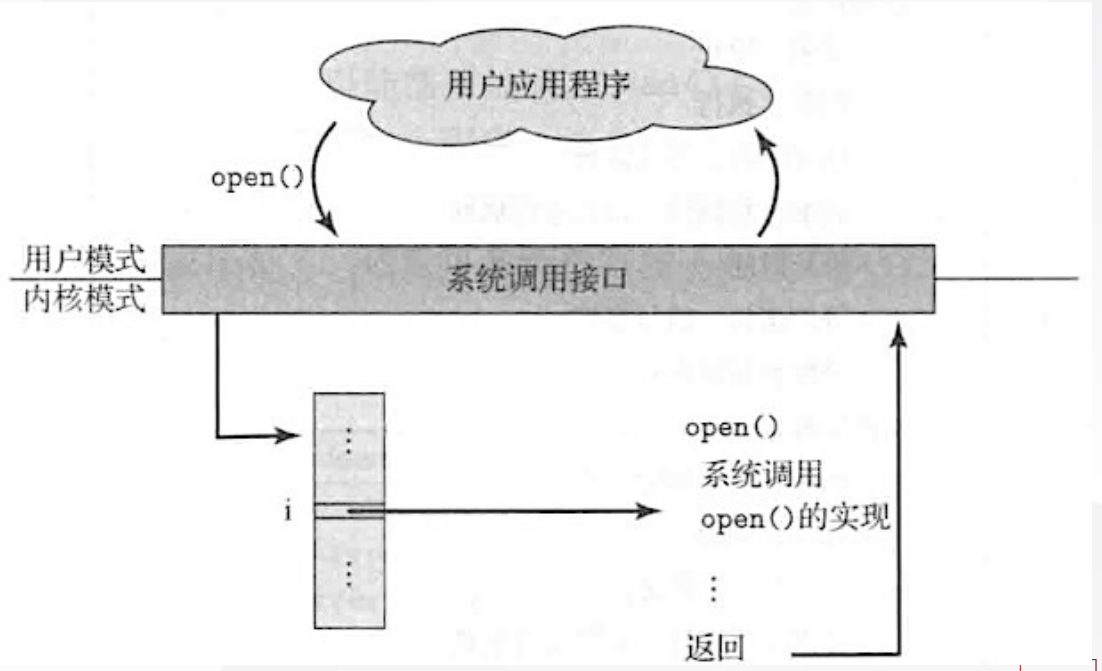
系统调用

- 系统调用(systemcall) 提供操作系统服务接口。这些调用通常以C或C++ 编写，当然，对某些底层任务(如需直接访问硬件的任务)，可能应以汇编语言指令编写。
- 例子：通过系统调用将一个文件的内容复制给另一个文件



系统调用-与操作系统的关系

- 调用者无需知道如何实现系统调用，而只需使用Application Program Interface (API)。通过API，操作系统将大多数细节隐藏起来， 且可由运行时库来管理
- 由API调用系统调用

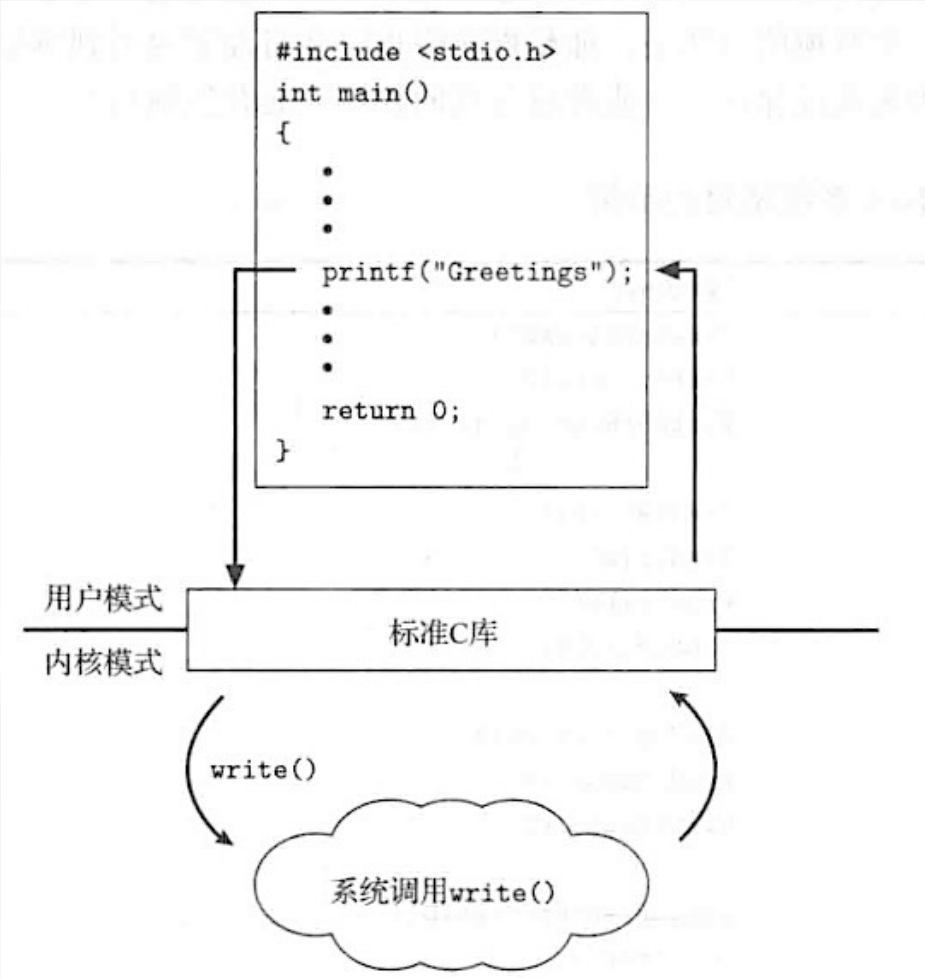


API

- 三种常见的API
  - Win32 API for Windows
  - POSIX API for POSIX-based systems (UNIX, Linux, and Mac OS X)
  - Java API for the Java virtual machine (JVM)
- 为什么使用API而不是直接使用系统调用？
  - 灵活的可编程性
  - 系统调用根据硬件专门设计，更加难用、注重细节

标准C语言-系统调用的例子

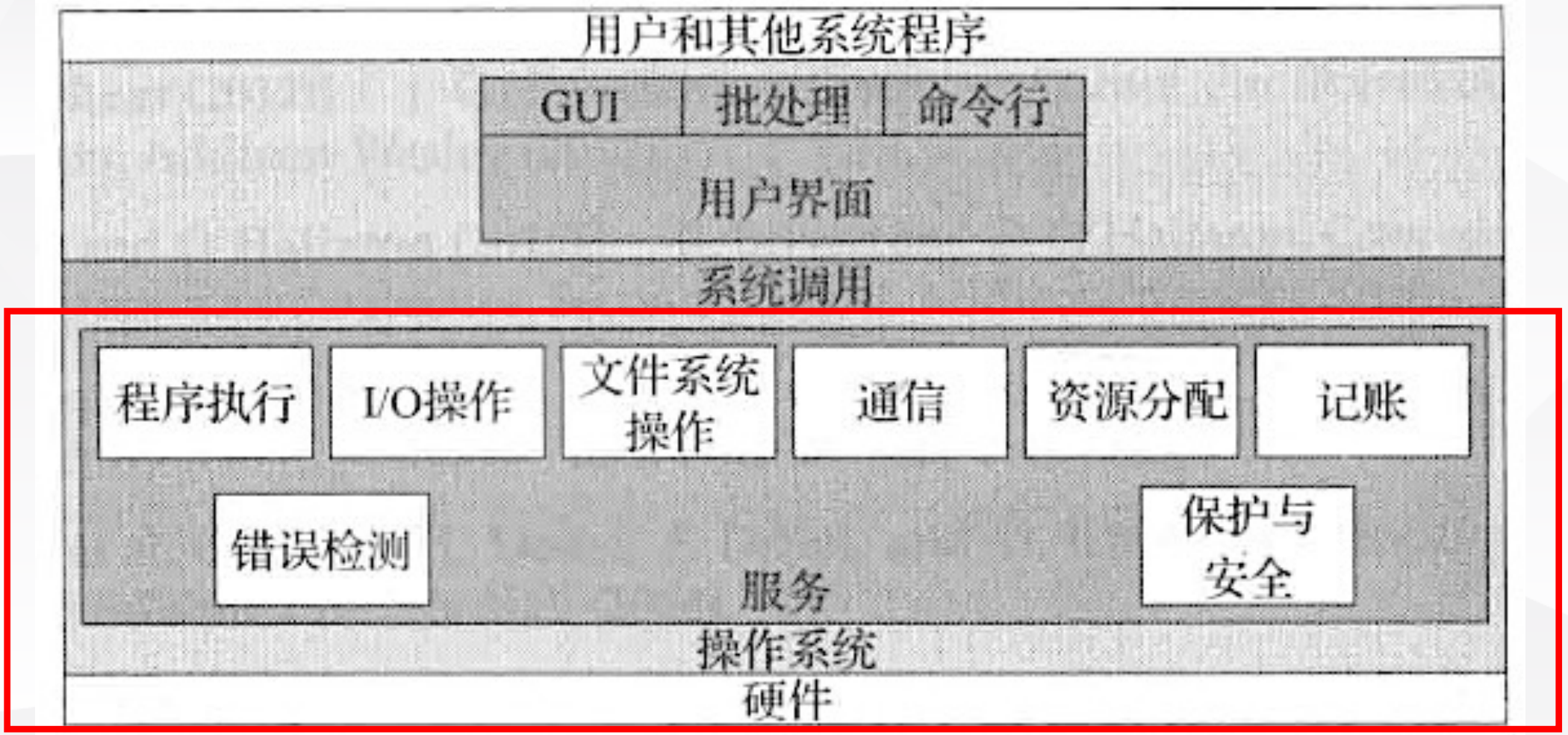
- 使用C语言中的printf()库，会经过操作系统唤起write()系统调用



Windows和Unix系统中常见的系统调用

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

操作系统服务级结构



操作系统为用户提供服务

操作系统服务

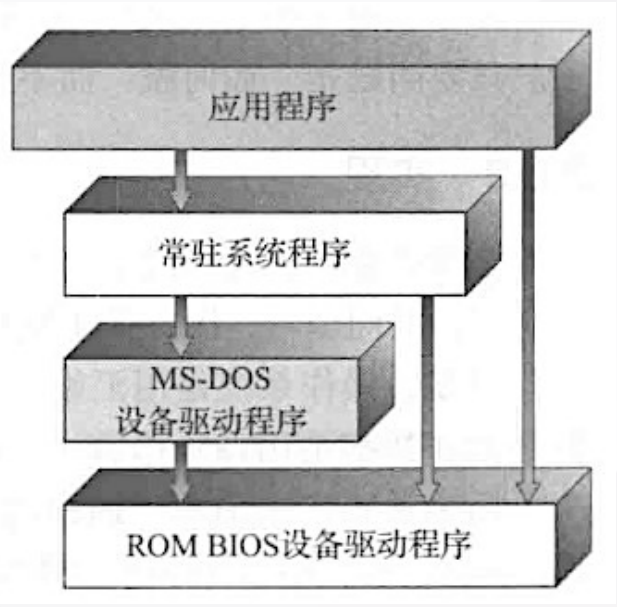
- 用户接口
  - 基本所有操作系统都具有用户接口：Graphics User Interface (GUI), Command-Line (CLI), Batch
- 程序执行
  - 系统必须能够将程序加载到内存中并运行该程序
- I/O行为
  - 正在运行的程序可能需要 I/O，这可能涉及文件或 I/O 设备
- 文件系统管理
  - 程序需要读写文件和目录，创建和删除它们，搜索它们，列出文件信息，权限管理。

操作系统服务

- 通信：进程可以在同一台计算机上（共享内存）或通过网络在计算机之间交换信息
- 错误检测
  - 错误可能发生在内存、CPU、I/O设备，操作系统需要进行合理的动作来确保正确、一致的计算
  - 调试设施可以大大提高用户和程序员高效使用系统的能力
- 资源分配：当多个用户或多个作业同时运行时，必须为每个用户或作业分配资源
- 保护与安全：提供控制访问计算机的系统资源的机制

操作系统的结构-简单结构

- MS-DOS（单任务系统）
  - 没有模块化，且MS-DOS的应用程序可以直接访问基本的I/O程序、并直接写到显示器和磁盘驱动。没有很好地区分功能的接口和层次。
    - 应用程序可以访问I/O程序、直接写入磁盘
    - 容易出错！

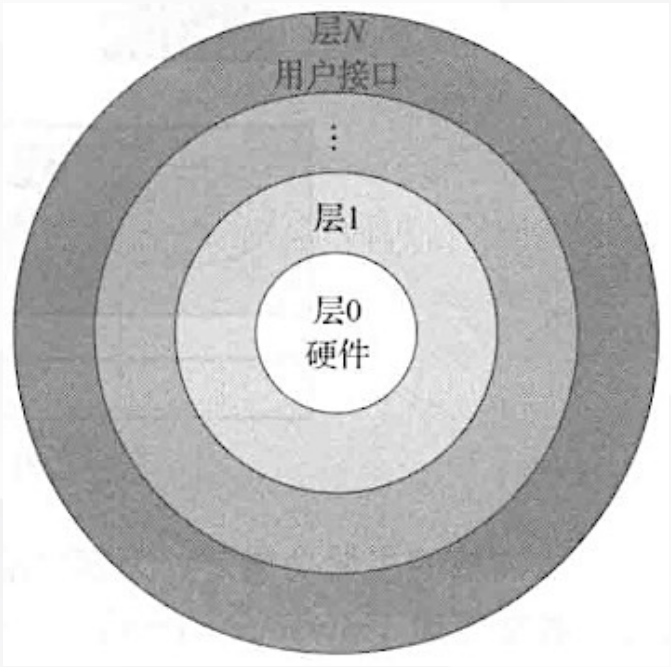


操作系统的结构-UNIX系统结构

- Unix
  - 采用有限结构，由两个独立部分组成：内核+系统程序。
  - 内核包含大量功能，难以实现。但优势在于系统调用接口与内核通信的开销极小。
  - 现代的Linux、Windows依然采用这种简单的单片结构

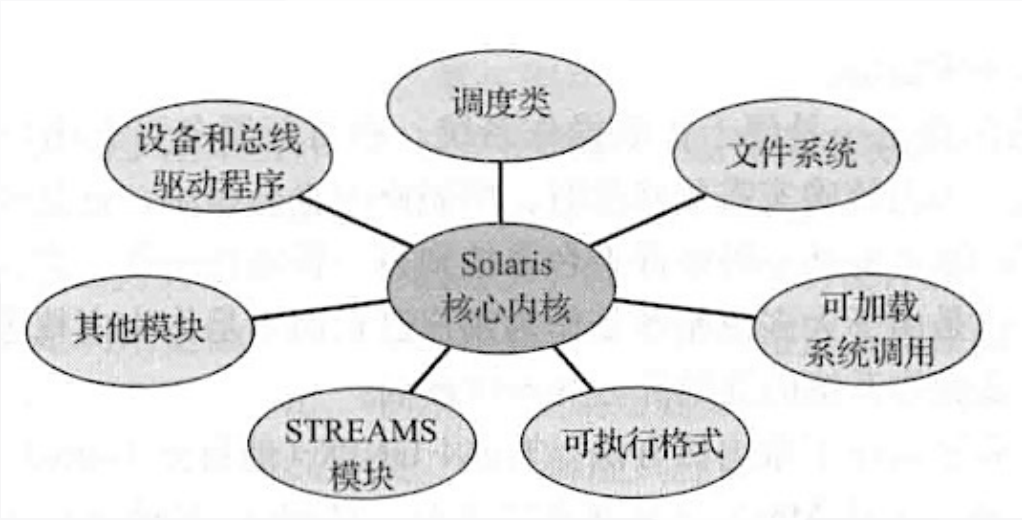
操作系统的结构-分层方法

- 操作系统被分为若干层，最底层为硬件，最上层为用户接口
- 每一层可以调用更底层的操作
- 优点：调试，第一层调试好了再调试第二层，简化系统设计
- 缺点：难以合理定义各层，哪些层用于管理内存；效率较低，需要逐层深入



操作系统的结构-模块

- 可加载的内核模块
  - 内核提供核心服务，而其他服务可在内核运行时动态实现。 动态链接服务优于直接添加新功能到内核，这是因为对于每次更改，后者都要重新编译内核。例如，可将CPU调度器与内存管理的算法直接建立在内核中，而通过可加载模块，可支持不同文件系统
- 更高的灵活性、更高效：任何模块可以调用其他任何块
- Linux、Solaris



课堂习题

1. 回顾前面的Hello world程序，操作系统的作用有哪些？

2. 请描述一种机制以加强内存保护，防止一个程序修改与其他程序相关的内存

3. 当前系统中有不同层次的缓存，有的缓存是为了单个处理核专用的，而有的缓存是为所有处理核共用的。为什么这么设计缓存系统?