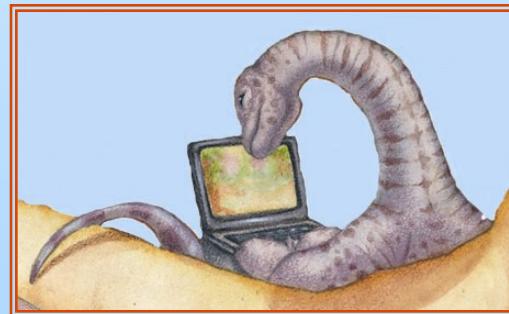
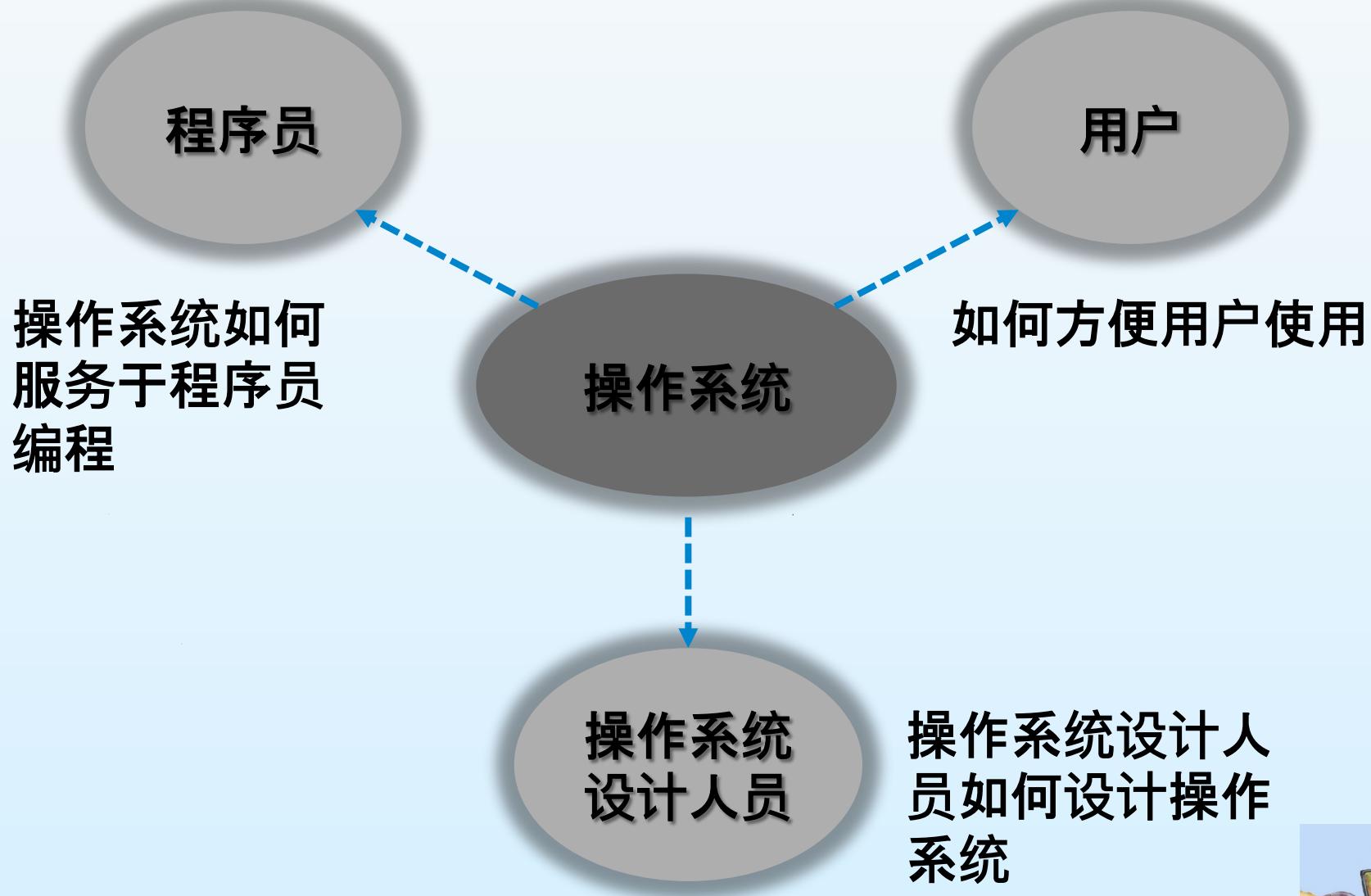


第2章 操作系统结构







目录

- 1、操作系统服务
- 2、操作系统程序接口-系统调用
- 3、操作系统用户界面
- 4、系统程序
- 5、操作系统结构
- 6、虚拟机（VM）

1、操作系统服务



提供用户功能
确保系统本身高效运行



操作系统服务

- 操作系统以服务形式向程序和用户提供一个环境执行程序
- 系统服务提供对用户有用函数:**基本服务**
 - 用户界面 - 形式有命令行界面(CLI)、图形用户界面(GUI)等
 - 程序执行 - 将程序调入内存并运行
 - I/O 操作 - I/O可能涉及文件或设备.
 - 文件系统操作 - 程序需要读写文件和目录，创建删除文件，检索
 - 通信 - 进程间可能需要交换信息
 - 错误检测 - OS 需要知道可能出现的错误





操作系统服务

■ 通过共享计算机资源来提高效率：增值**服务**

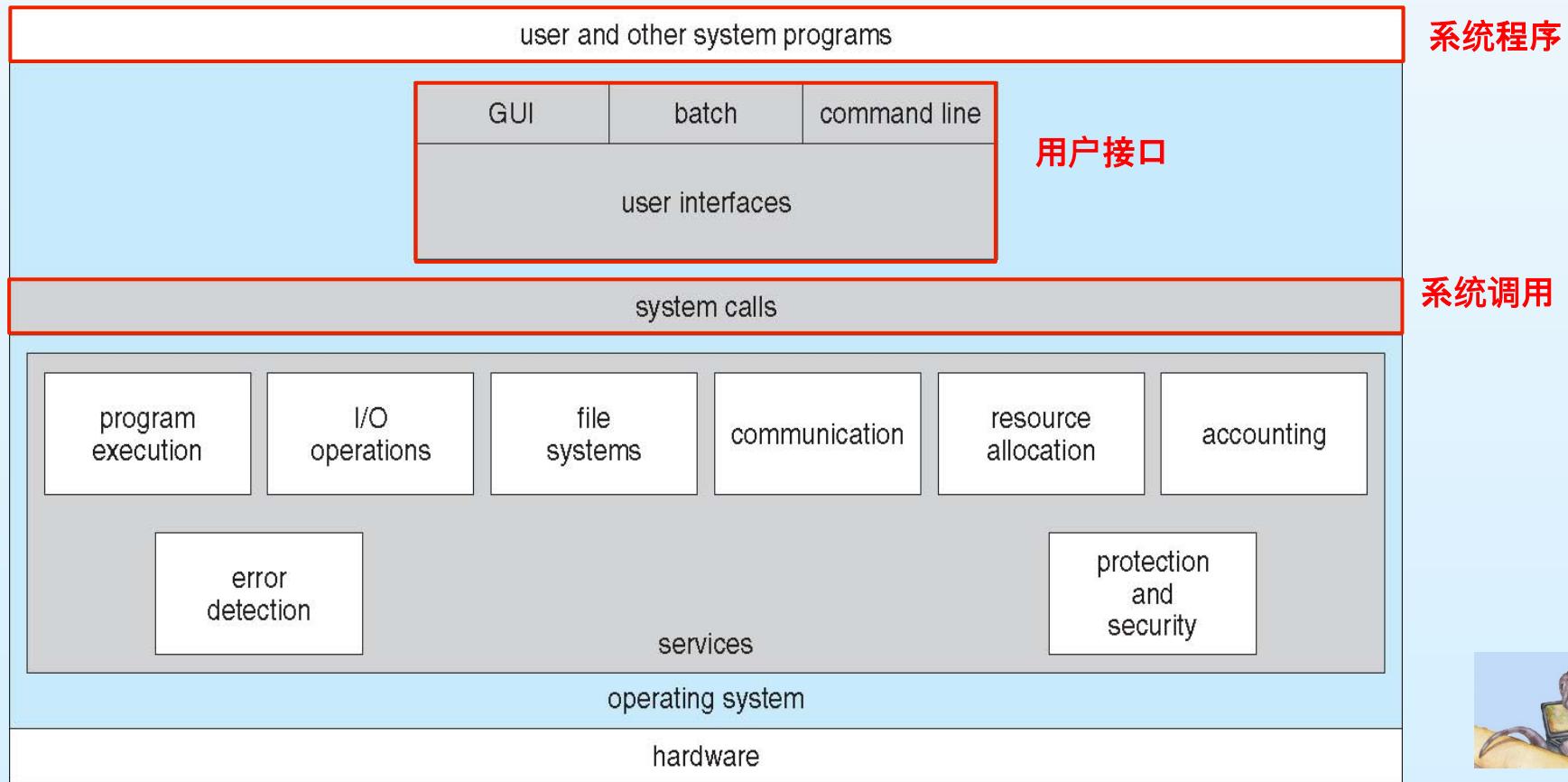
- 资源分配 - 多个作业并发运行时，系统为它们分配资源
- 统计 - 需要记录哪些用户使用了多少和什么类型的资源
- 保护和安全 - 用户可能需要控制信息的使用
 - ▶ 保护：确保所有对系统资源的访问受控
 - ▶ 安全：不受外界侵犯，延伸到外部I/O设备不受非法访问



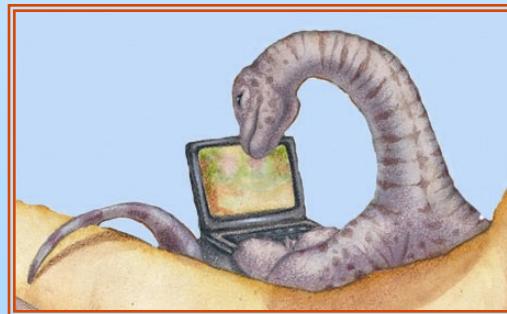


操作系统服务

- 系统调用 (System Call)
- 用户接口 (User Interface)
- 系统程序 (System Program)



2、系统调用





系统调用

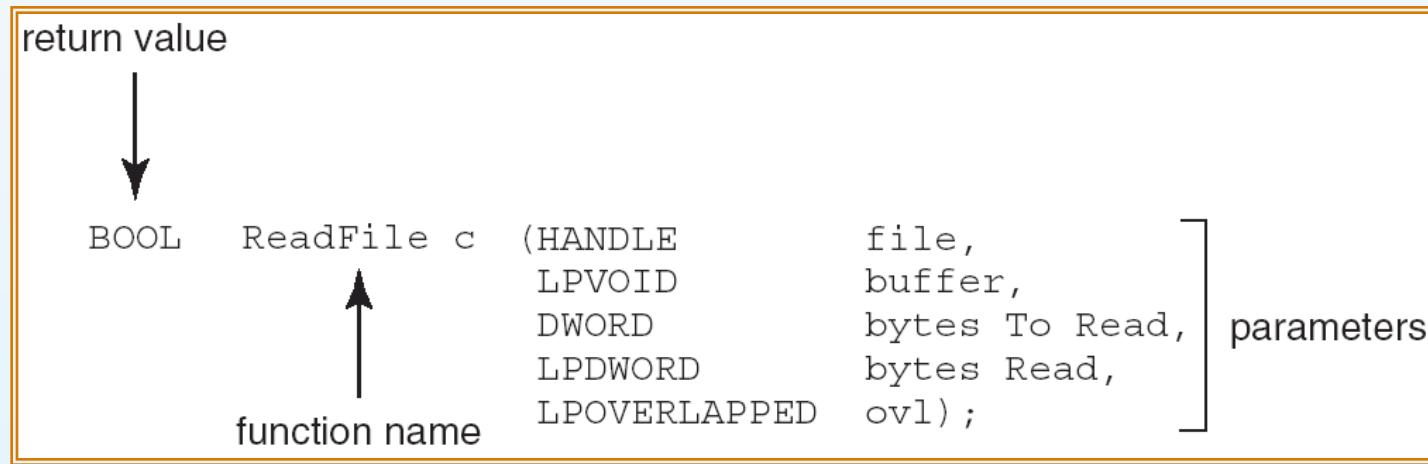
- 系统调用：用户程序可以通过操作系统提供的服务接口调用系统功能。
- 系统调用是操作系统提供给编程人员的唯一接口
- 系统调用在程序中无处不在，各种文件操作、屏幕输出、设备访问都需要调用系统调用
- 操作系统服务的编程接口-面向程序
- 通常用高级语言编写 (C or C++)
- 程序通过应用程序接口(API)访问，而不是直接使用系统调用
- 三种常用 APIs：
 - Windows 的Win32 API
 - POSIX系统 (包括几乎所有版本的UNIX, Linux, 和Mac OS X) 的POSIX API
 - Java 虚拟机 (JVM) 的Java API





标准 API的例子

■ Win32 API中ReadFile()方法，从文件中读取内容



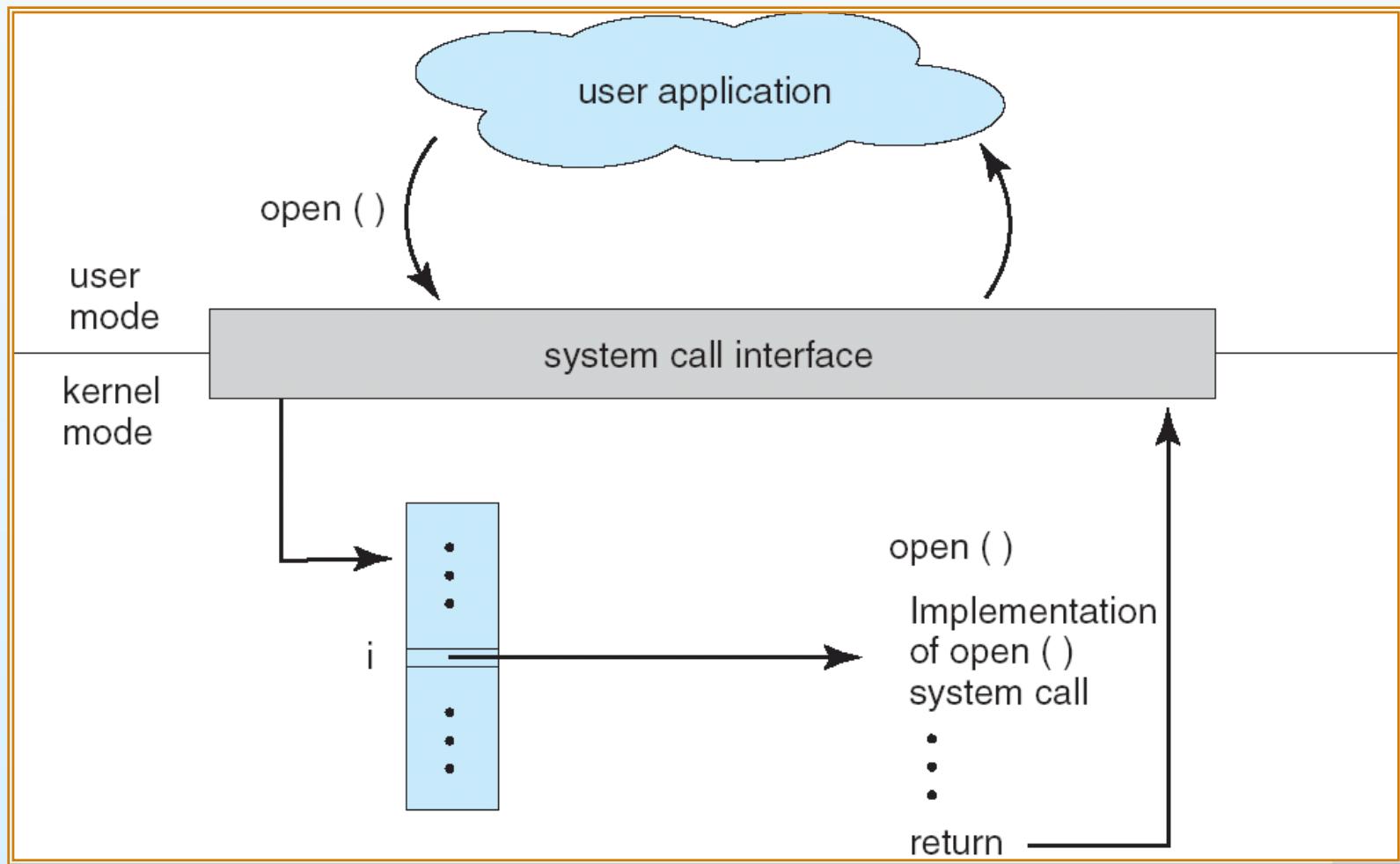
■ ReadFile()函数的参数描述如下：

- `HANDLE file`—所要读取的文件
- `LPVOID buffer`—读进写出的数据缓冲
- `DWORD bytesToRead`—将要读入缓冲区的字节数
- `LPDWORD bytesRead`—上次读操作读的字节数
- `LPOVERLAPPED ovl`—指示是否使用重叠 I/O





API – 系统调用 – OS 之间的关系





思考

系统调用只能在核心态运行

- A. 正确
- B. 错误



Solaris 10 dtrace 动态跟踪工具

```
# ./all.d `pgrep xclock` XEventsQueued
dtrace: script './all.d' matched 52377 probes
CPU FUNCTION
0 -> XEventsQueued U
0 -> _XEventsQueued U
0 -> _X11TransBytesReadable U
0 <- _X11TransBytesReadable U
0 -> _X11TransSocketBytesReadable U
0 <- _X11TransSocketBytesreadable U
0 -> ioctl U
0 -> ioctl K
0 -> getf K
0 -> set_active_fd K
0 <- set_active_fd K
0 <- getf K
0 -> get_udatamodel K
0 <- get_udatamodel K
...
0 -> releaseef K
0 -> clear_active_fd K
0 <- clear_active_fd K
0 -> cv_broadcast K
0 <- cv_broadcast K
0 <- releaseef K
0 <- ioctl K
0 -> ioctl U
0 <- _XEventsQueued U
0 <- XEventsQueued U
```

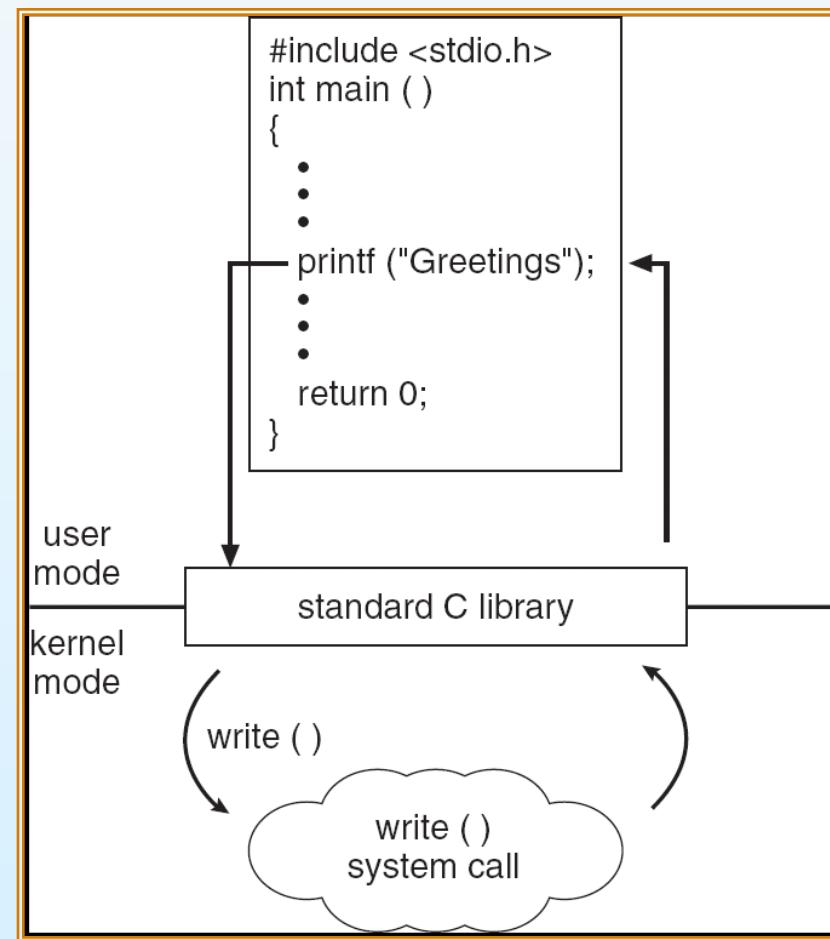
U:用户模式
K:核心模式





标准的C库例子

- C 程序库调用 `printf()`, 其调用了`write()`系统调用



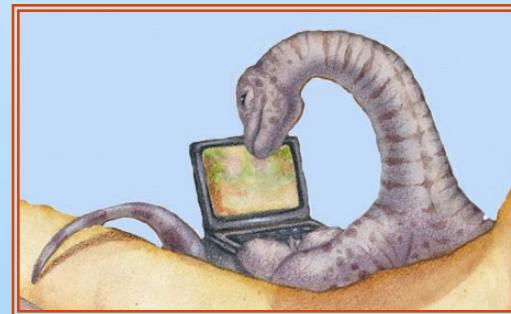


Windows 和 Unix 系统调用例子

	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() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()



3、操作系统用户界面





■ 用户接口

- 命令行接口
- 图形接口





CLI (Command-Line Interface)

CLI 允许用户直接输入操作系统完成的命令

- 主要作用是获取并执行用户指定的命令
- 字符模式
- 用户直接输入命令，执行内置的或外置的命令
- 运行结果通过字符显示在屏幕上

- 内核或系统程序实现
- 有时有多种实现方式 – 外壳(**shells**)

- 优点：简单、健壮、效率高
- 缺点：使用不方便，界面不美观





Bourne Shell

```
Default
New Info Close Execute Bookmarks
Default Default
PBG-Mac-Pro:~ pbgs w
15:24 up 56 mins, 2 users, load averages: 1.51 1.53 1.65
USER TTY FROM LOGIN@ IDLE WHAT
pbgs console - 14:34 50 -
pbgs s000 - 15:05 - w
PBG-Mac-Pro:~ pbgs iostat 5
      disk0          disk1          disk10         cpu    load average
      KB/t   tps   MB/s      KB/t   tps   MB/s      KB/t   tps   MB/s us sy id 1m 5m 15m
33.75 343 11.30 64.31 14 0.88 39.67 0 0.02 11 5 84 1.51 1.53 1.65
  5.27 320 1.65 0.00 0 0.00 0.00 0 0.00 4 2 94 1.39 1.51 1.65
  4.28 329 1.37 0.00 0 0.00 0.00 0 0.00 5 3 92 1.44 1.51 1.65
^C
PBG-Mac-Pro:~ pbgs ls
Applications           Music           WebEx
Applications (Parallels) Pando Packages config.log
Desktop                Pictures          getsmartdata.txt
Documents              Public            imp
Downloads              Sites             log
Dropbox                Thumbs.db        panda-dist
Library                Virtual Machines prob.txt
Movies                 Volumes           scripts
PBG-Mac-Pro:~ pbgs pwd
/Users/pbg
PBG-Mac-Pro:~ pbgs ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=2.257 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=1.262 ms
^C
--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 1.262/1.760/2.257/0.498 ms
PBG-Mac-Pro:~ pbgs 
```





MS-DOS

```
C:\Windows\system32\cmd.exe
C:\Users\pfli>dir
驱动器 C 中的卷没有标签。
卷的序列号是 7088-C6D2

C:\Users\pfli 的目录

2015/07/09  13:44    <DIR>
2015/07/09  13:44    <DIR>          .
2015/07/09  13:44    <DIR>          ..
2015/01/23  17:28    <DIR>          21E247D45E274BEAAA4D19A81203FE2A.TMP
2015/07/09  13:51    <DIR>          Contacts
2015/07/01  20:11    <DIR>          Desktop
2015/07/12  09:47    <DIR>          Documents
2015/05/17  09:35    <DIR>          Downloads
2015/01/23  17:28    <DIR>          Favorites
2015/01/23  17:28    <DIR>          Links
2015/01/23  17:28    <DIR>          Music
2015/01/23  17:28    <DIR>          Pictures
2015/01/23  17:28    <DIR>          Saved Games
2015/01/23  17:28    <DIR>          Searches
2015/01/23  17:28    <DIR>          Videos
                                0 个文件          0 字节
                                14 个目录 66,474,819,584 可用字节

C:\Users\pfli>
```





GUI(Graphical User Interface)

■ 用户界面友好的桌面接口

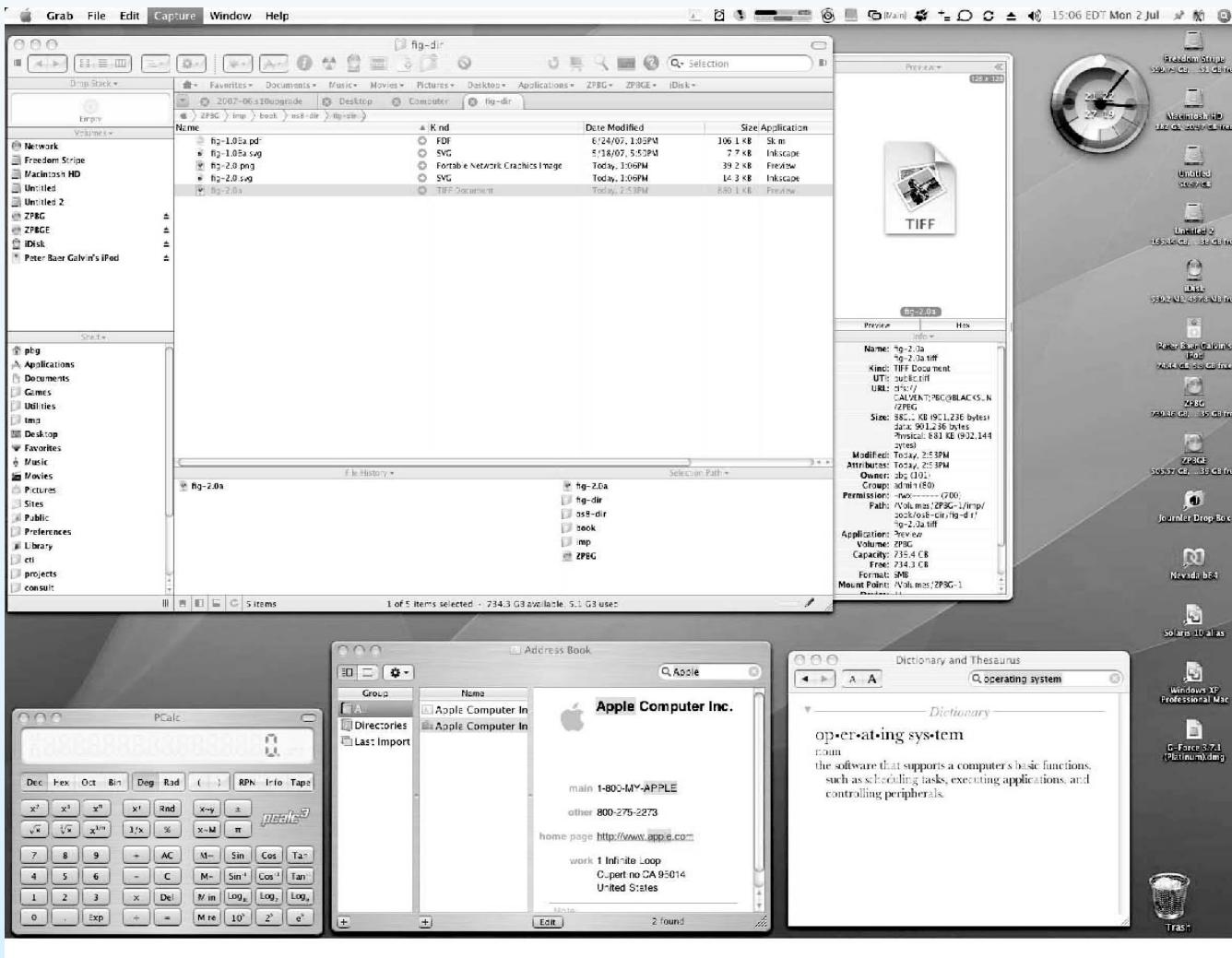
- 通常使用鼠标、键盘和监视器
- 常用元素：图标、窗口、滚动条等
- 图标代表文件、程序、系统功能等
- 不同对象上鼠标按钮导致不同的动作
- 用户命令以**鼠标操作**为主
- GUI首次出现在 Xerox PARC

- 优点：操作方便、界面直观、漂亮
- 缺点：效率差、不够健壮





The Mac OS X GUI





第一个图形界面——Xerox Alto

Ready:
Select file names with the mouse
Red-Copy, Yel-Copy/Rename, Blue-Delete
Click 'Start' to execute file name commands

Start

Quit

Clear

Type

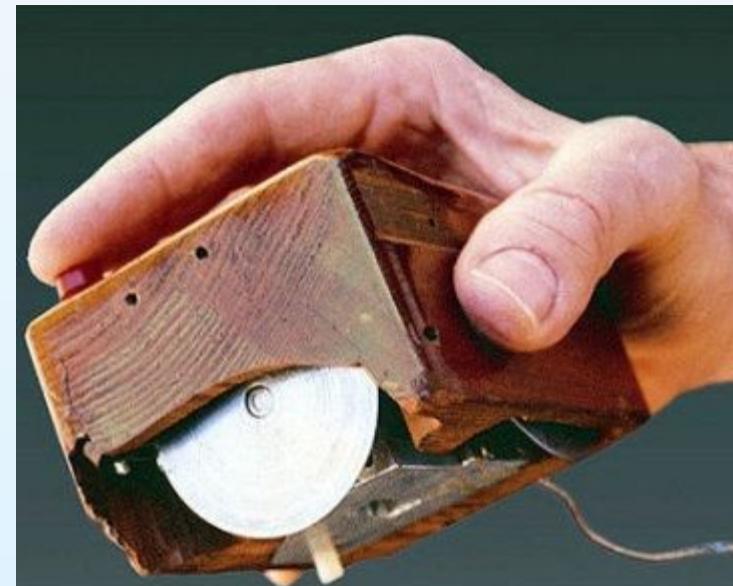
--

Pages: 832	Log
Files listed: 60	
Files selected: 0	Delete: 0
Copy/Rename: 0	Copy: 0

Pages: 0	Log
Files listed: 0	
Files selected: 0	Delete: 0
Copy/Rename: 0	Copy: 0

DP0: <SysDir.> *.*

```
~~ BEGINNING ~~
1012-AstroRoids.Boot.
Anonymous.1.
BattleShip.er.
BattleShip.RUN.
BlackJack.RUN.
BuildKal.cm.
CalcSources.dm.
Calculator.RUN.
Chess.log.
Chess.run.
Com.Cm.
CompileKal.cm.
CRTTEST.RUN.
DMT.boot.
EdsBuild.run.
empress.run.
Executive.Run.
Fly.run.
galaxian.boot.
Garbage.$.
Go9.run.
GoFont.AL.
Invaders.Run.
junk.
junk.press.
Kal.bcpl.
Kal.cm.
KalA.asm.
KalMc.mu.
Kinetic4.RUN.
LoadKal.cm.
MasterMind.RUN.
maze.run.
Mesa.Typescript.
Missile.run.
NEPTUNE.RUN.
othello.run.
Pinball-easy.run.
POLYGONS.RUN.
```





人机接口

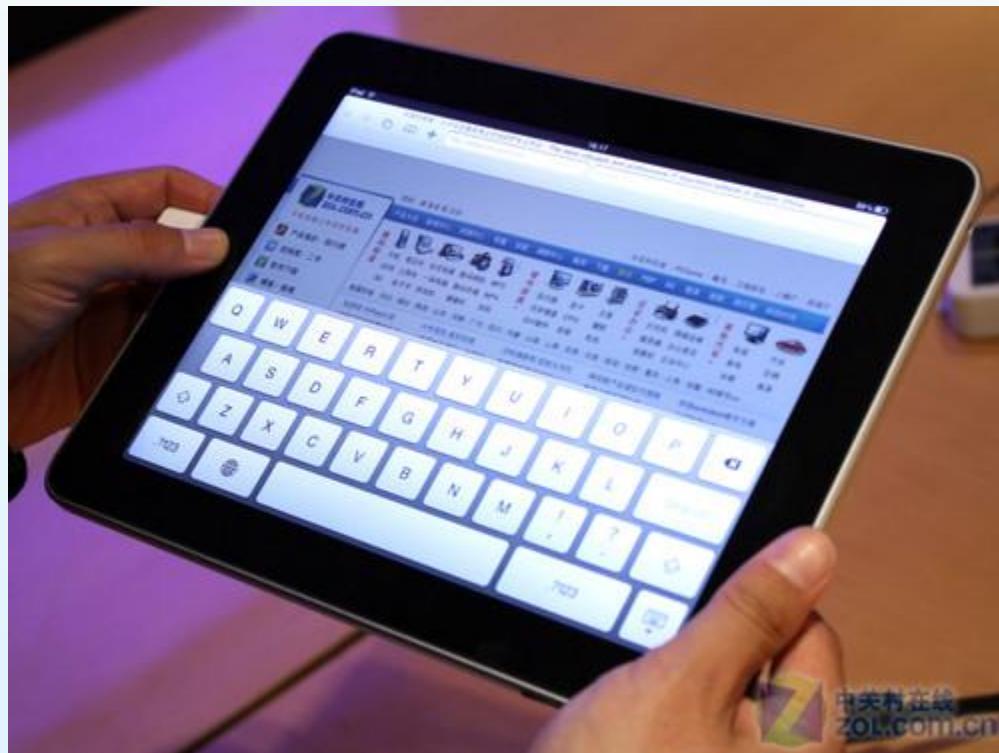
■ 许多系统同时包含CLI和GUI界面

- Microsoft Windows 使用带有命令行的图形界面
- Linux (Gnome, KDE) 和 Shell
- Apple Mac OS X 采用 “Aqua” 界面, UNIX 使用命令行界面并有多个 shell 可用
- Solaris 是 CLI 界面带有可选的图形界面 (Java 桌面, KDE)





触摸屏GUI



- 触摸屏GUI把键盘、鼠标和显示器进行三合一，统一为触摸屏
- 触摸屏GUI有多点触控、大按钮等特点



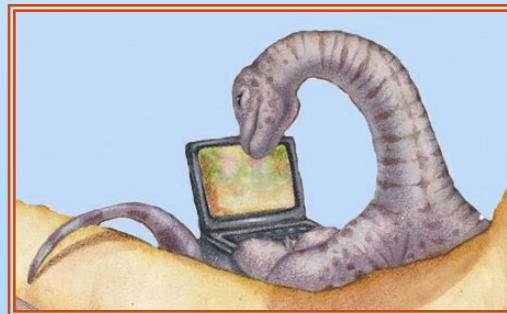


未来的人机接口

- i-Air Touch 技术
- 语音操控
- 体感操控
-



4、系统程序





系统程序

- 一般认为用于管理、维护操作系统的程序是系统程序。
- 提供一个方便的环境，以开发程序和执行程序
- 为用户使用操作系统服务

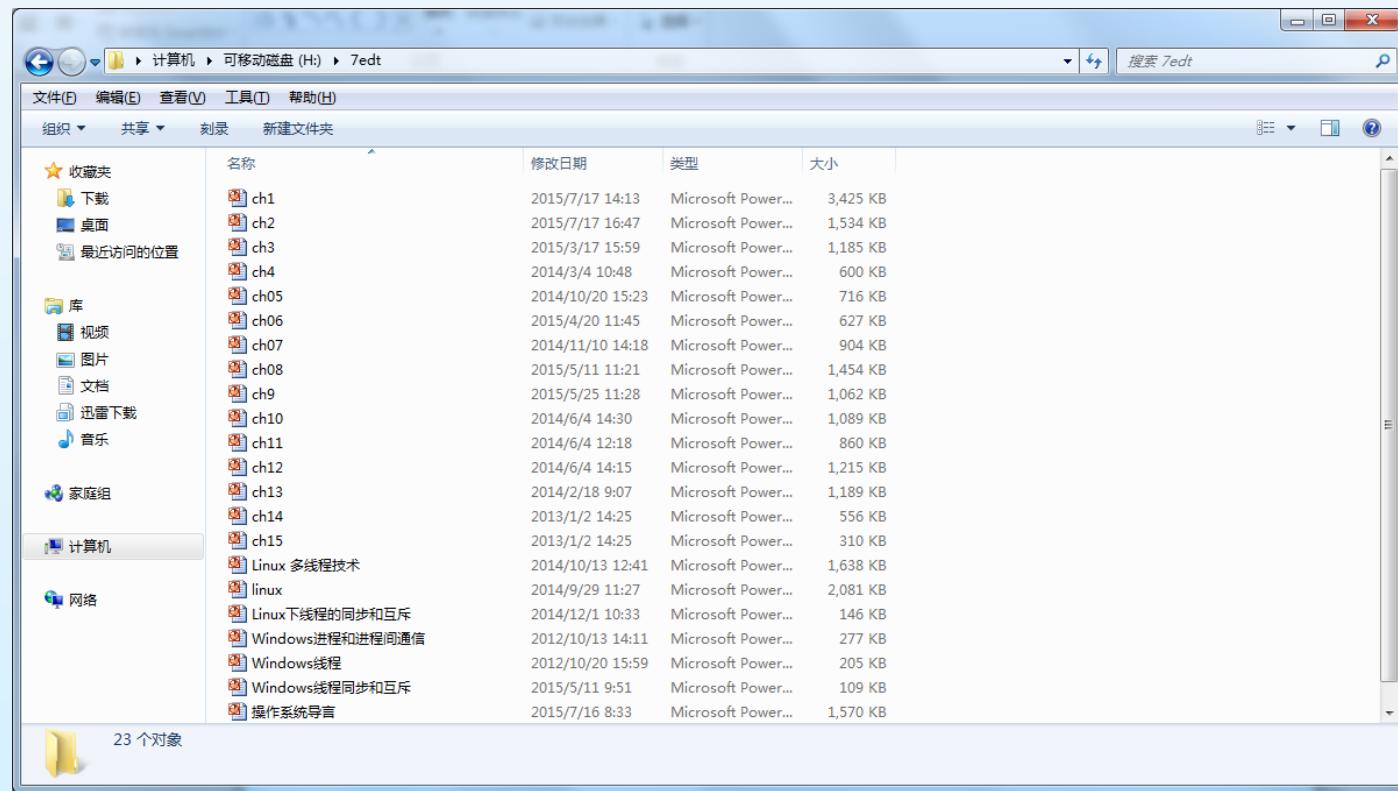
- 功能：
 - 文件管理
 - 状态信息
 - 文件处理
 - 程序语言支持
 - 程序装入和执行
 - 通信





系统程序功能

■ 文件管理 – 创建、删除、复制、重命名、打印、转储、列出和操作文件和目录





系统程序功能

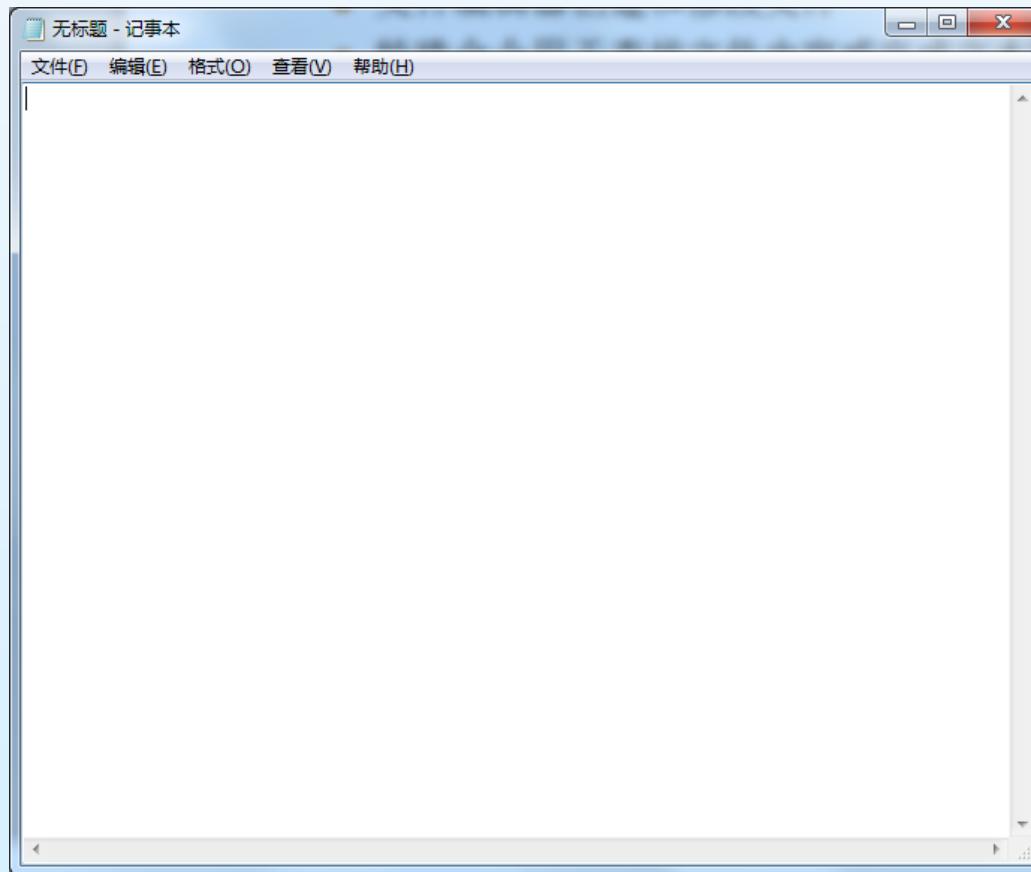
- 状态信息 – 日期、时间、可用内存、磁盘空间和用户数等
- 性能、登录和调试信息
- 注册表 – 用于存储和检索配置信息





系统程序功能

- 文件处理
- 文件编辑器创建和修改文件
- 特殊命令用于查找文件内容或完成文本转换





系统程序功能

■ 程序语言支持

- 常用程序设计语言的编译程序、汇编程序、调试程序和解释程序



```
zsxinlu@zsxinlu-P43SJ: ~/code
zsxinlu@zsxinlu-P43SJ: ~$ /home/zsxinlu/code
bash: /home/zsxinlu/code: Is a directory
zsxinlu@zsxinlu-P43SJ: ~$ cd /home/zsxinlu/code
zsxinlu@zsxinlu-P43SJ: ~/code$ g++ -o helloworld helloworld.C
helloworld.C:3:14: error: two or more data types in declaration of 'main'
zsxinlu@zsxinlu-P43SJ: ~/code$
```



系统程序功能

■ 程序装入和执行

- 绝对加载程序、重定位加载程序、链接编辑器和覆盖式加载程序，还有高级语言或机器语言的调试程序

```
C:\> C:\WINDOWS\system32\cmd.exe - debug
-t
AX=0001 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B52 ES=0B52 SS=0B52 CS=1000 IP=000A NV UP EI PL NZ NA PO NC
1000:000A B90200 MOU CX,0002
-u 1000:0
1000:0000 0000 ADD [BX+SI],AL
1000:0002 62 DB 62
1000:0003 61 DB 61
1000:0004 69 DB 69
1000:0005 64 DB 64
1000:0006 75B8 JNZ FFC0
1000:0008 0100 ADD [BX+SI],AX
1000:000A B90200 MOU CX,0002
1000:000D 01C8 ADD AX,CX
1000:000F 0000 ADD [BX+SI],AL
1000:0011 0000 ADD [BX+SI],AL
1000:0013 0000 ADD [BX+SI],AL
1000:0015 0000 ADD [BX+SI],AL
1000:0017 0000 ADD [BX+SI],AL
1000:0019 0000 ADD [BX+SI],AL
1000:001B 0000 ADD [BX+SI],AL
1000:001D 0000 ADD [BX+SI],AL
1000:001F 0000 ADD [BX+SI],AL
```





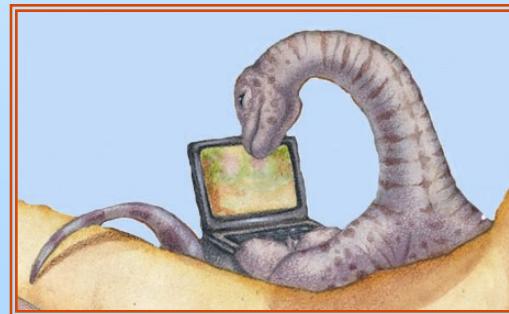
系统程序功能

■ 通信

- 提供在进程、用户和计算机系统之间创建虚拟链接的机制



5、操作系统结构





操作系统结构类别

- 简单结构 (Simple structure)
- 层次结构 (Layered)
- 微内核结构 (Microkernel)
- 模块结构 (Modules)
- 混合结构 (Hybird)

通用操作系统是个庞然大物





简单结构

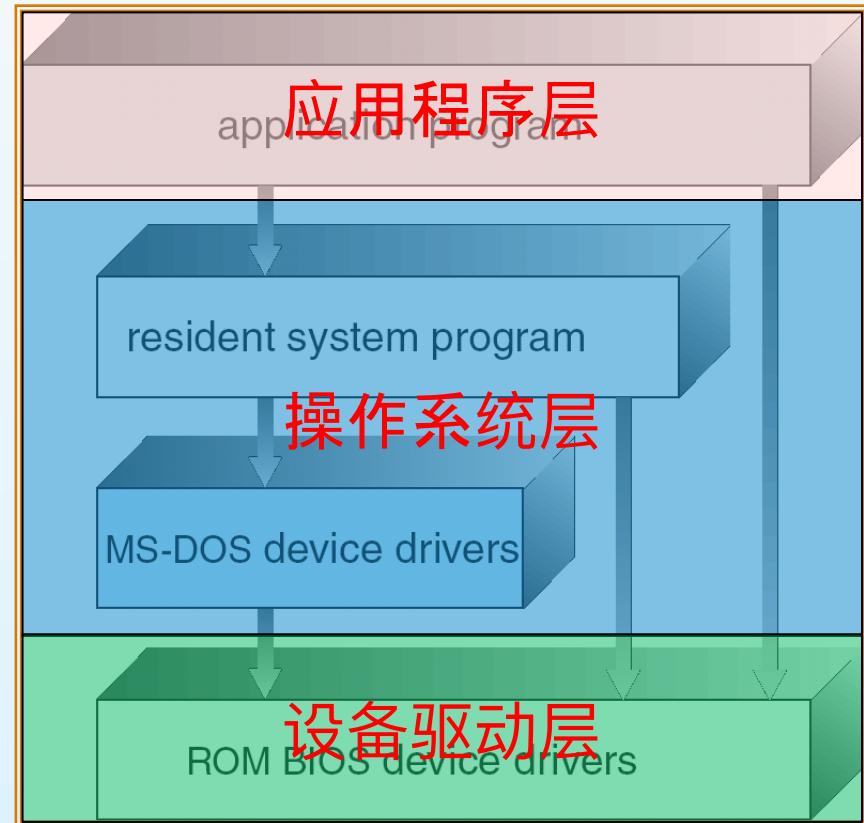
- 也就是没有结构
- 早期操作系统（规模小，简单，功能有限）
- 几个人，甚至一个人设计实现
- 系统较为混乱、不易维护和更新、不适合大规模系统开发





操作系统结构设计

- 简单操作系统: MS-DOS(1981-1994)
- 不划分模块的单体内核
- 没有保护机制, 容易被攻击破坏
- 很难扩展
- 计算能力有限, 存储有限, 内存640K
- 受硬件限制操作系统很难突破
- MS-DOS-以最小的空间提供最多的功能
- 尽管MS-DOS有某种结构, 其接口和功能层没有划分清楚





简单结构-早期UNIX

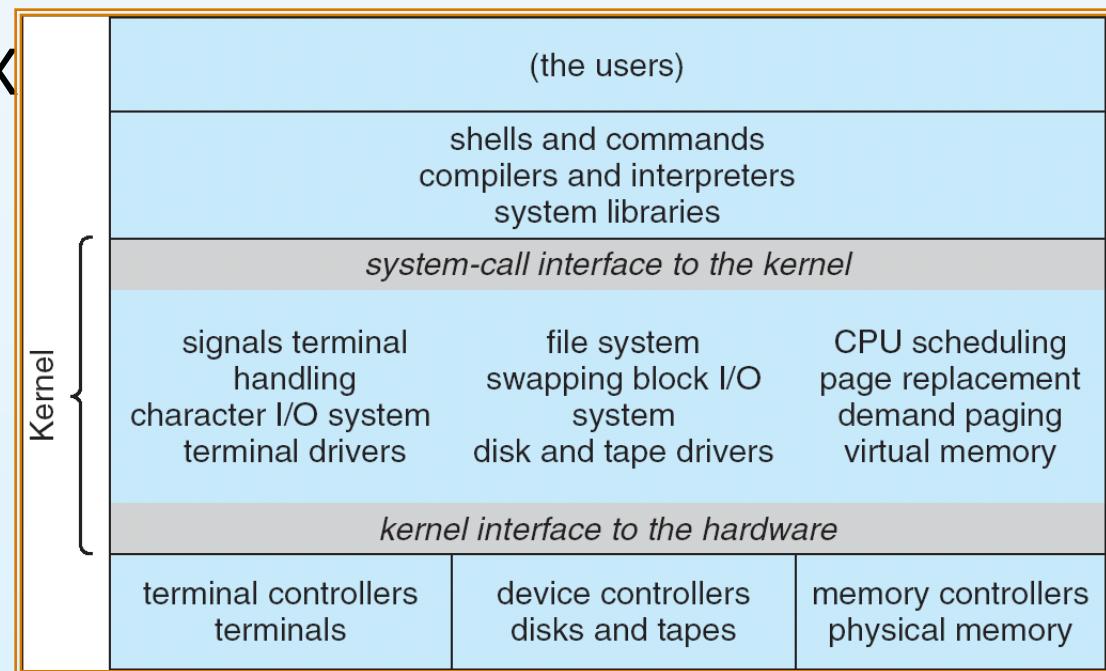
■ UNIX – 受到硬件功能的限制，原始的UNIX操作系统UNIX OS 由两个独立部分组成：

- 系统程序

- 内核

- ▶ 接口和驱动程序，硬件上和系统调用接口下的所有部分

- ▶ 文件系统、CPU调度、内存管理和其他操作系统功能





操作系统源代码规模

■ 1) Windows

- Windows 95: 1500万行代码
- Windows 98: 1800万行代码
- Windows XP: 3500万行代码
- Windows Vista: 5000万行代码
- Windows 10: 0.5T 代码、400 万文件、50 万文件夹

■ 2) Linux

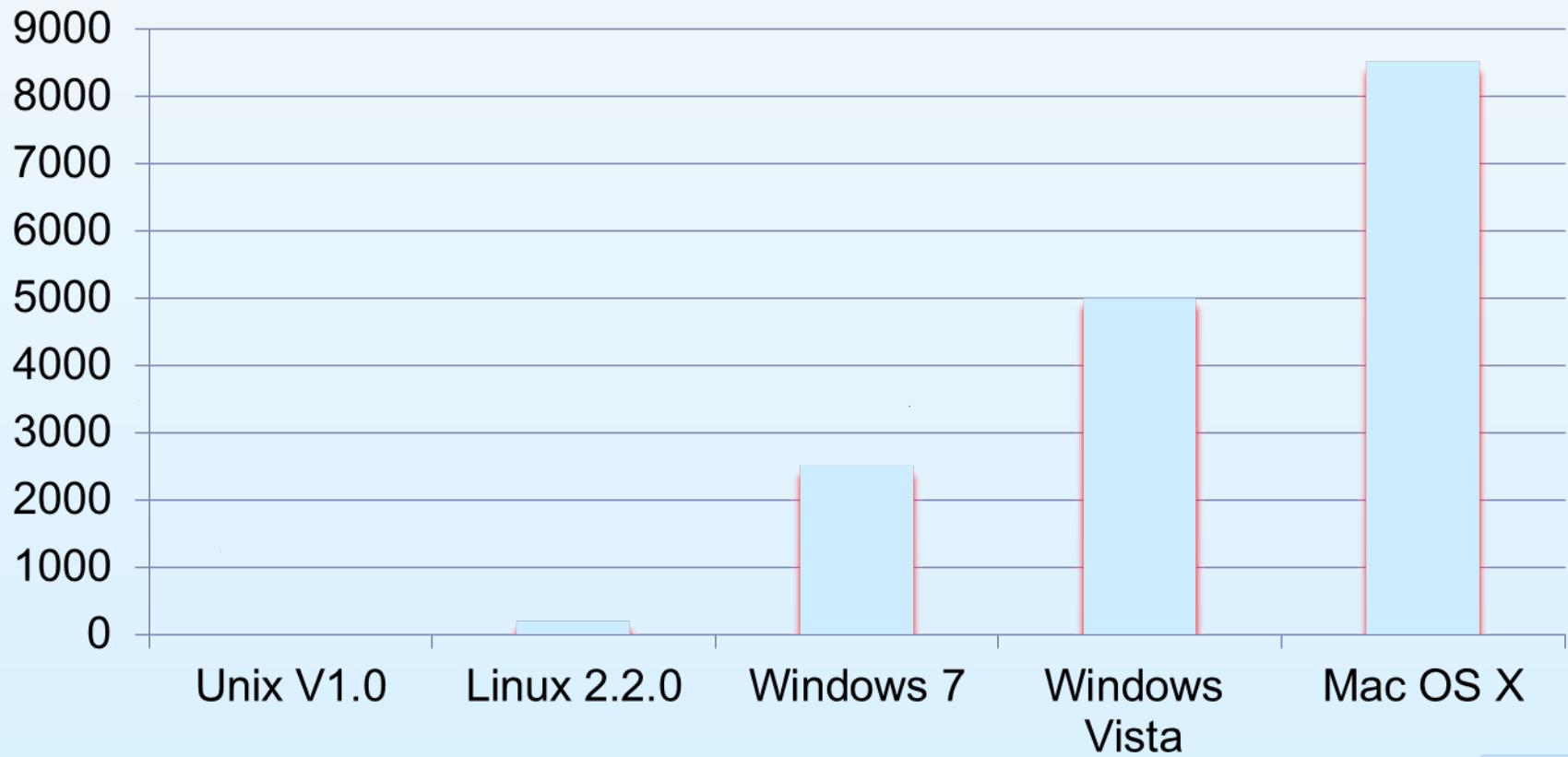
- Linux内核，早期几十万，后来100多万。
- Linux 2.6.23版本发布时，它的源代码总行数大概为550万行左右。
- Linux 2.6.27之后的内核源代码已经超过1000万行。
- Mac OS X 8500万行





操作系统规模

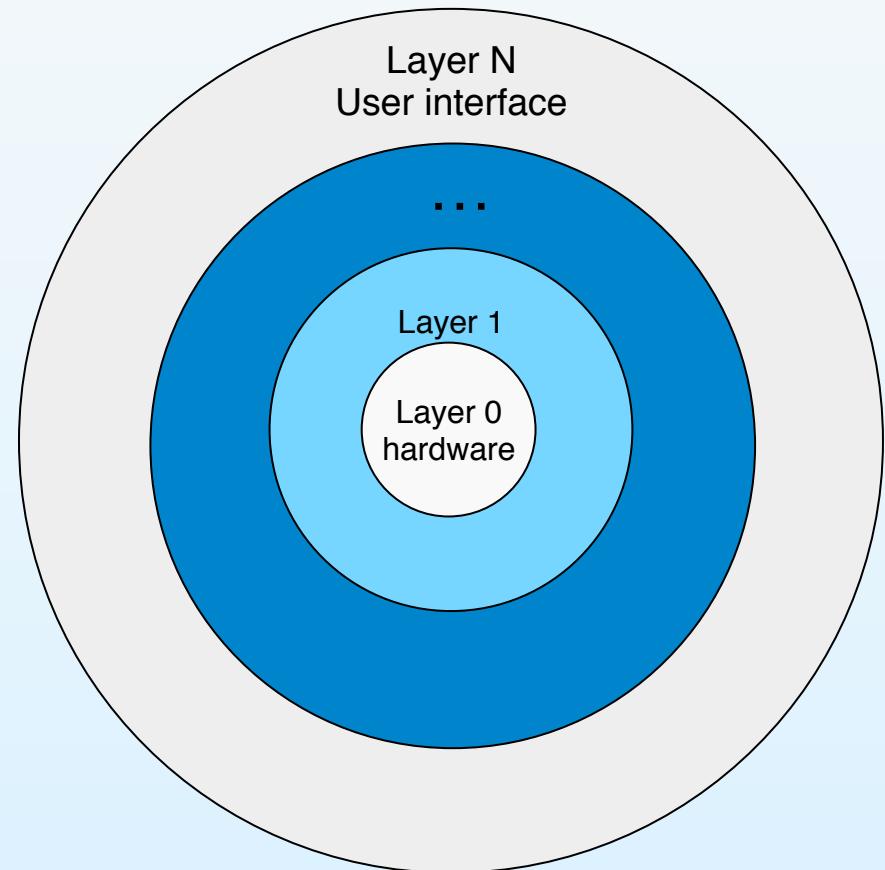
源代码（万行）





操作系统结构设计

- 操作系统划分为若干层
- 在低层上构建高层
- 底层（0层）为硬件
- 最高层（N层）为用户层
- 每层只使用低层次的功能和服务





■ 在层次结构中，第n层能够调用第()层提供的服务。

- A. n-1
- B. n+1
- C. 0~n-1
- D. 以上都不对





层次结构

■ 优点

- 简化了系统设计和实现
- 便于调试和升级维护

■ 缺点

- 层定义困难
- 效率差

■ 例子-THE

- 1968年由E. W. Dijkstra开发
- 第一次提出了层次式结构设计方法
- 运行在荷兰的Electrologica X8（32K内存）上的一个简单批处理系统





层次结构-THE

第五层 用户程序

第四层 输入/输出管理

第三层 操作员控制台

操作系統

第二层 储存管理

第一层 CPU调度与信号

第零层 硬件设施



Dijkstra

■ 艾兹格·W·迪科斯彻(Edsger Wybe Dijkstra)

- 提出信号量和**PV**原语—第六章
 - 解决了“哲学家聚餐”问题—第六章
 - 最短路径算法(**SPF**)和银行家算法的创造者-第七章
 - 结构程序设计之父
 - **THE**操作系统的设计者和开发者—本章
- 与D. E. Knuth并称为我们这个时代最伟大的计算机科学家





iOS

■ 基于Mac OS X, 增加部分功能性部件

- Cocoa Touch: 提供Objective-C API 用于开发Apps
- Media services : 图像、视频和声音等服务
- Core services: 提供云计算、数据库等服务
- Core OS: Mac OS X 内核

Cocoa Touch

Media Services

Core Services

Core OS





微内核

- 问题：内核越来越大，越来越难管理
- 内核微型化：从核内移出尽可能多功能到用户空间
- 发生在用户模块间的通信使用消息传递形式
- 优点：
 - 便于扩充微内核
 - 便于移植操作系统到新架构系统上
 - 更稳定 (更少的代码运行在核心态)
 - 更安全
- 缺点：
 - 用户空间和内核空间通信的系统开销增加
- 解决方法：
 - 提出消息传递机制





微内核系统

■ 第一个微内核系统：CMU的Mach

- <http://www.cs.cmu.edu/afs/cs/project/mach/public/www/mach.html>

■ Tru64 Unix使用Mach内核

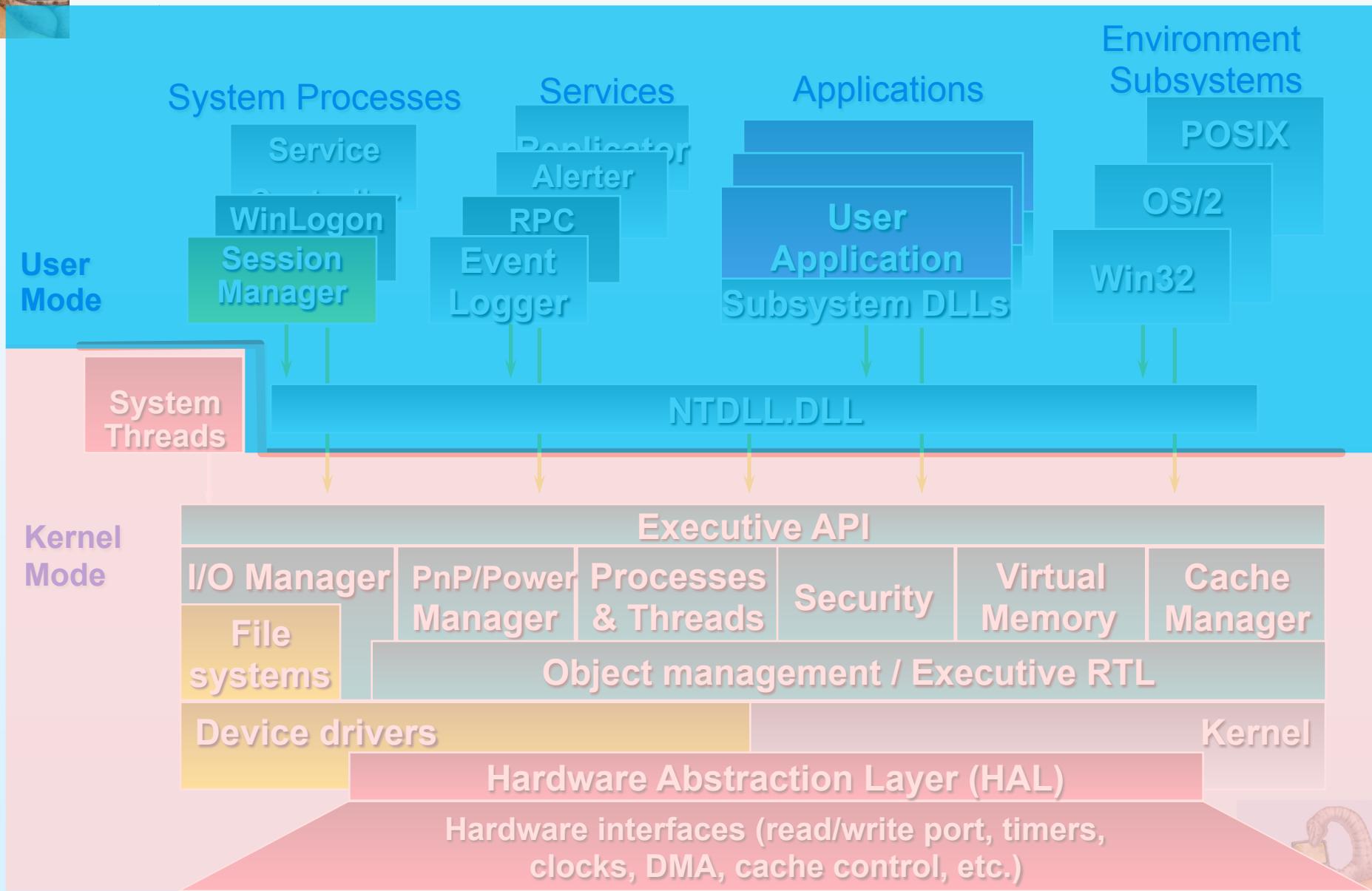
■ QNX-基于微内核的实时操作系统

■ Windows NT, 2000, 2003等





Windows 2000 微内核结构





模块结构

■ 大部分现代操作系统采用模块结构

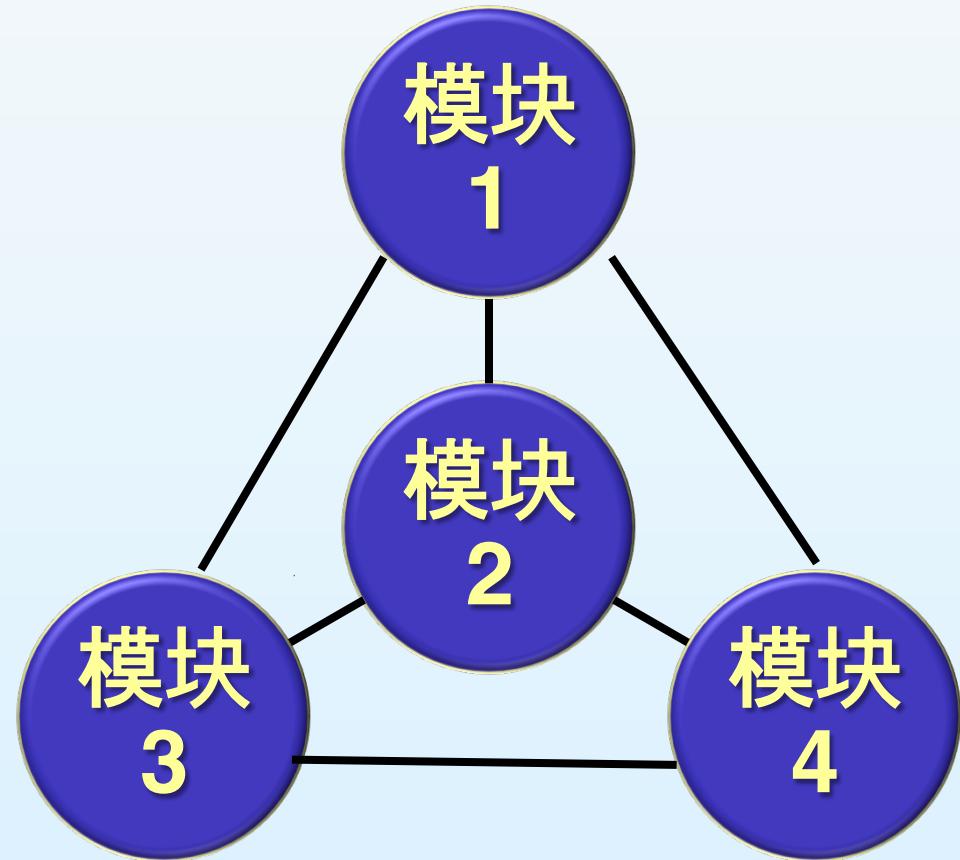
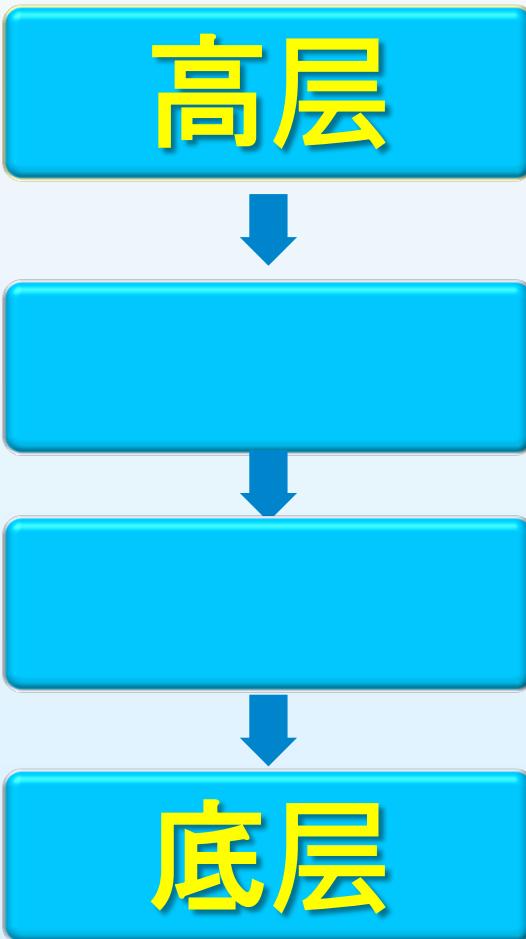
- 使用面向对象方法
- 每个核心部件分开为多个模块
- 每个与其他组件（模块）的会话被称为接口
- 每个组件（模块）在需要时被加载到内核

■ 类似于分层方法，但更灵活

■ 优点：

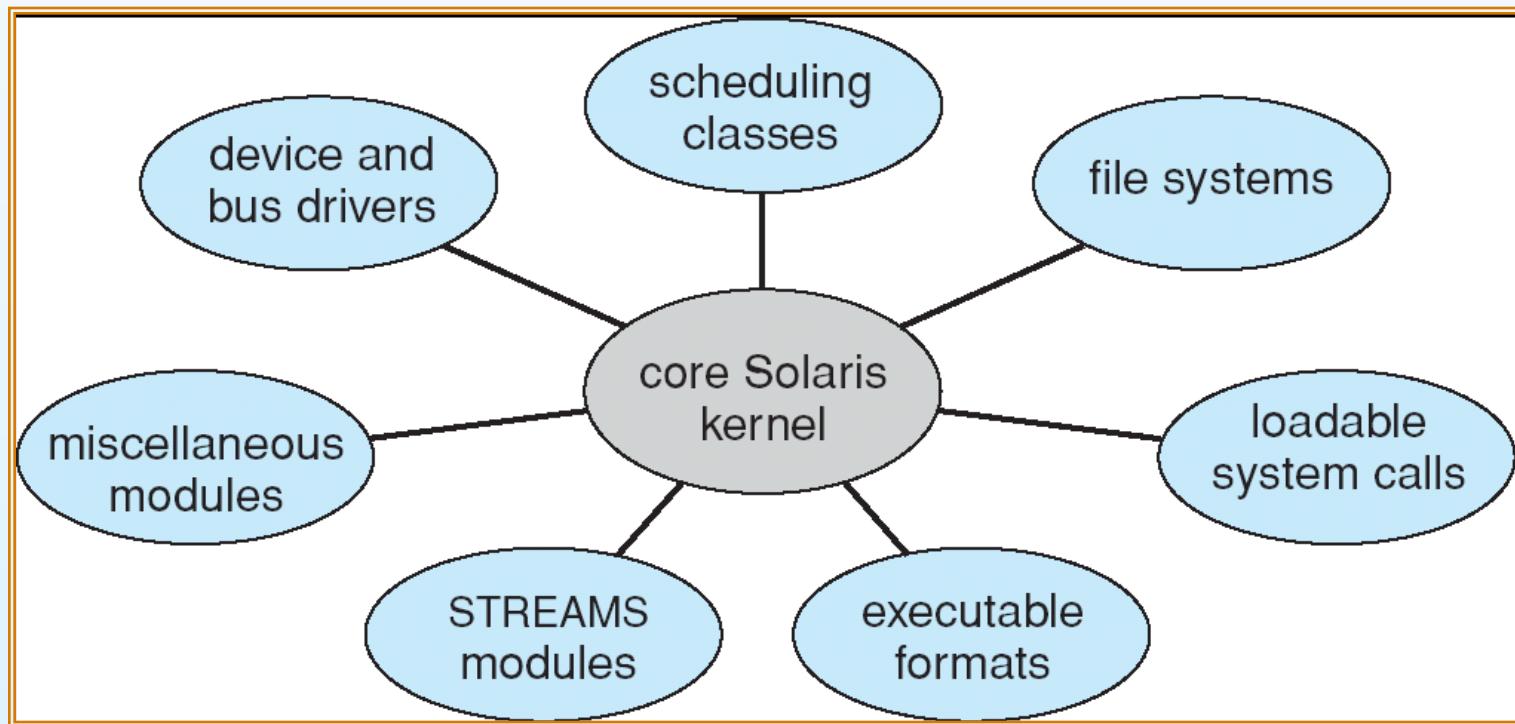
- 模块可以在需要时加载到内核
 - ▶ 类似于硬件的即插即用技术
- 用户可以自己设计模块并装入系统







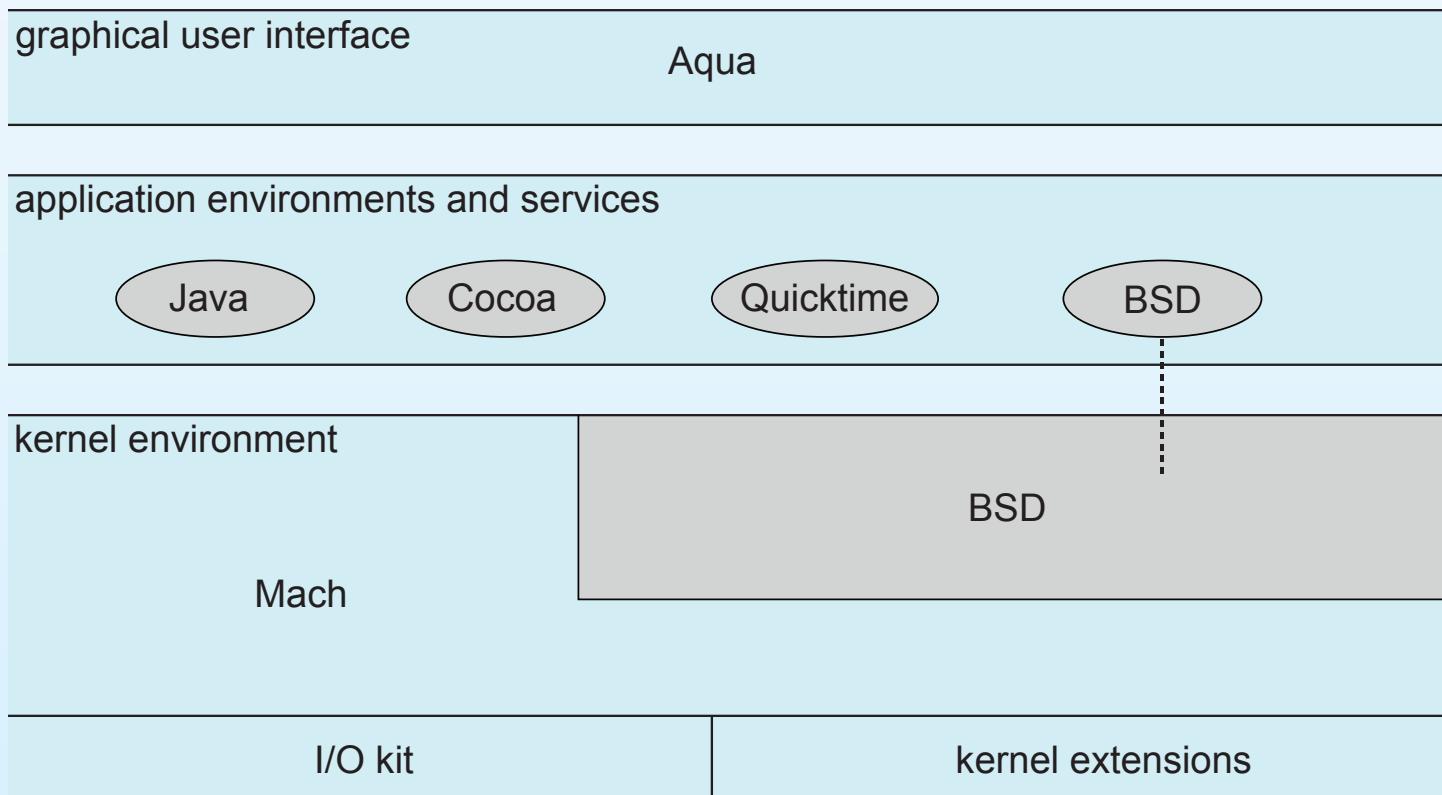
Solaris 模块





混合结构--Mac OS X

- 许多现代操作系统不是采用一种单一结构，通过采用多种结构获取性能、安全、使用等方面的需求





Android

■ 基于修改的Linux内核

