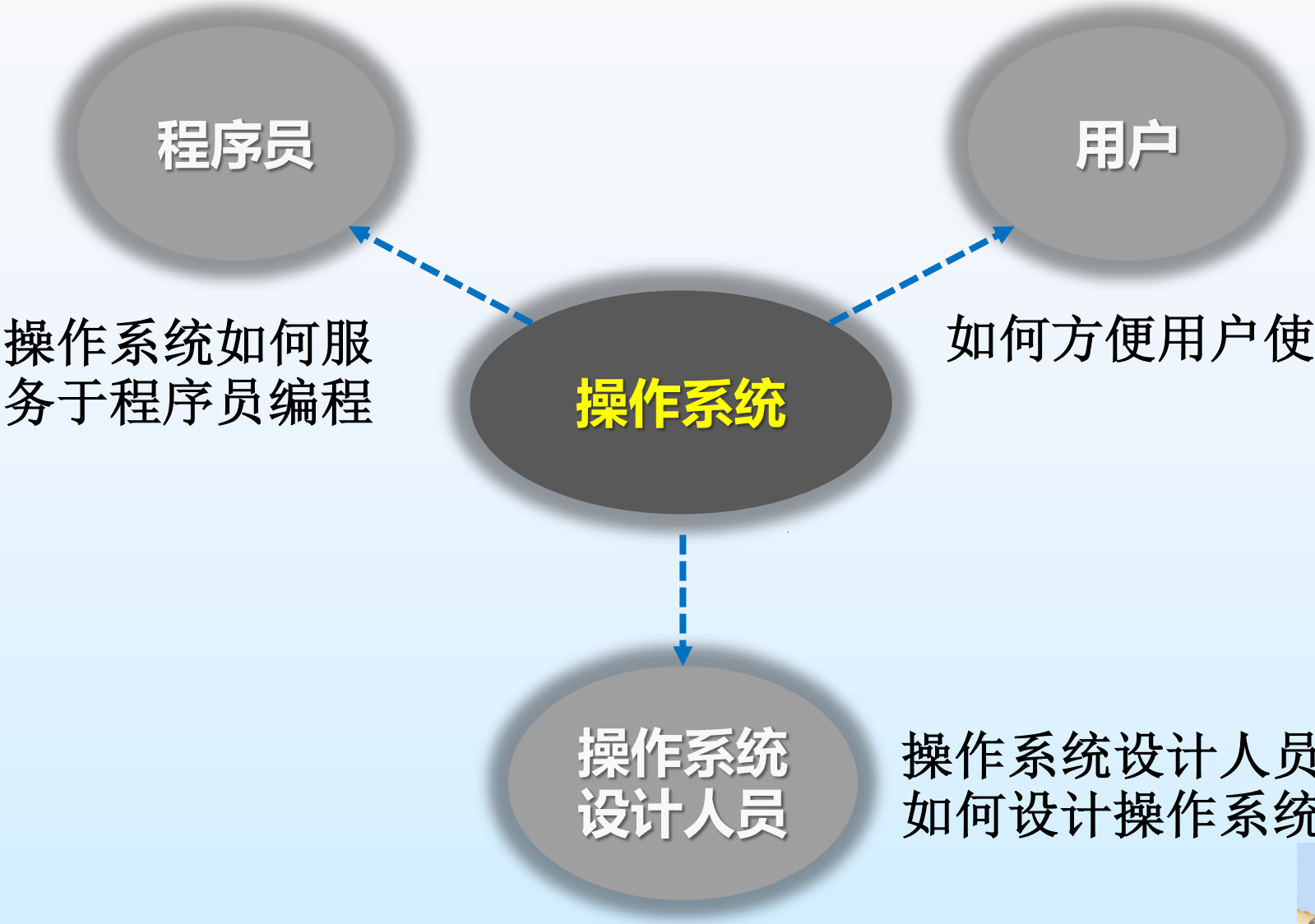
第2章 操作系统结构


程序员
用户
操作系统如何服务于程序员编程
操作系统
如何方便用户使用
操作系统设计人员
操作系统设计人员如何设计操作系统

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

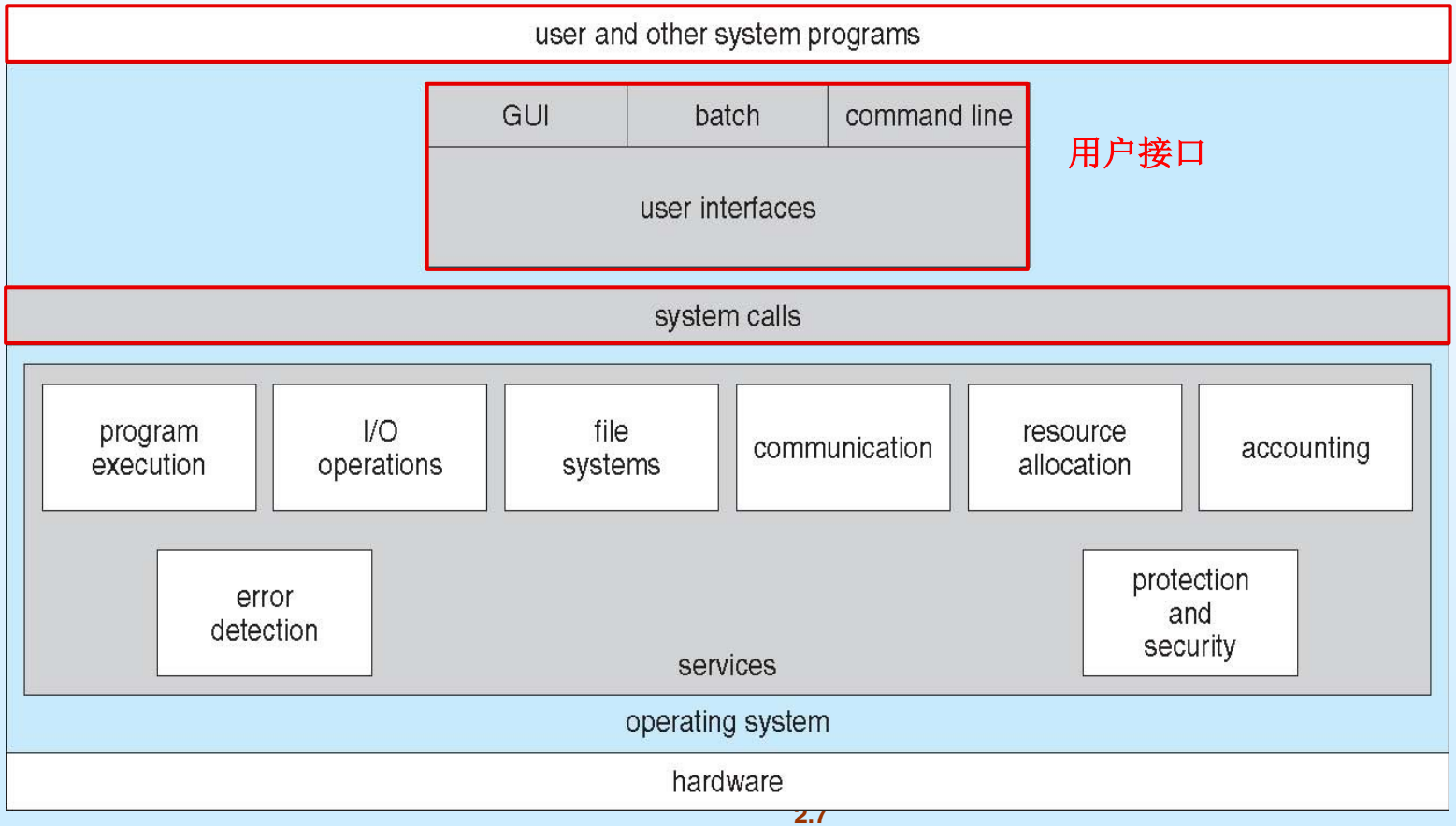
- 操作系统以服务形式向程序和用户提供一个环境执行程序
- 系统服务提供对用户有用函数:基本服务
  - **用户界面** –形式有命令行界面(CLI)、图形用户界面(GUI)等
  - **程序执行** – 将程序调入内存并运行
  - **I/O 操作** - I/O可能涉及文件或设备.
  - **文件系统操作** - 程序需要读写文件和目录，创建删除文件，检索
  - **通信** – 进程间可能需要交换信息
  - **错误检测** – OS 需要知道可能出现的错误

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）



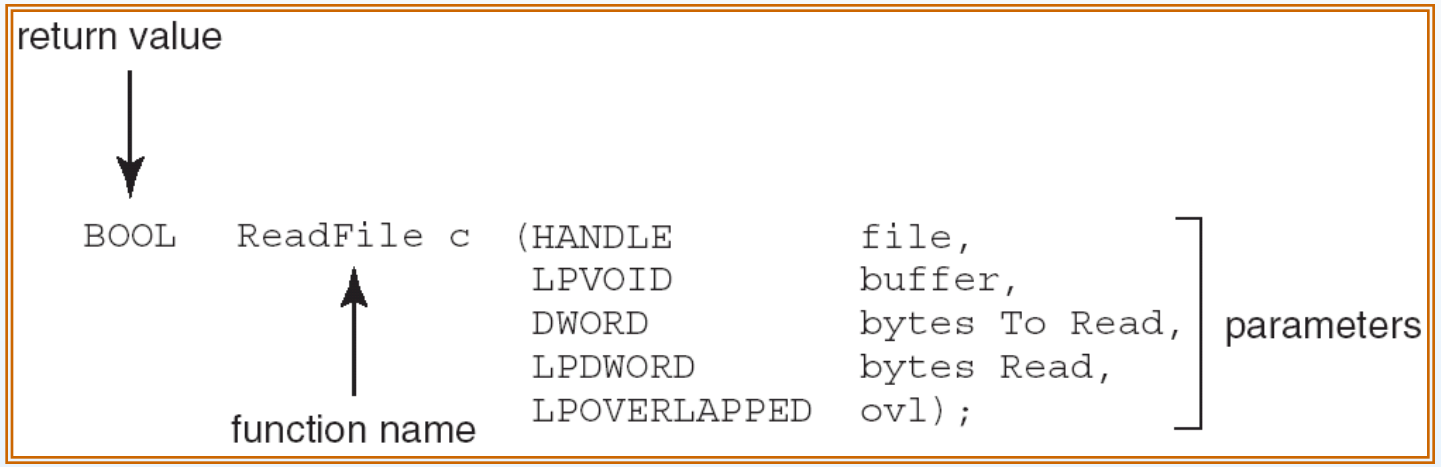
2、系统调用

系统调用

1. 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。
2. 系统调用是操作系统提供给编程人员的唯一接口
3. 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
4. 操作系统服务的编程接口-**面向程序**
5. 通常用高级语言编写 (C or C++)
6. 程序通过应用程序接口(API)访问，而不是直接使用系统调用
7. 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

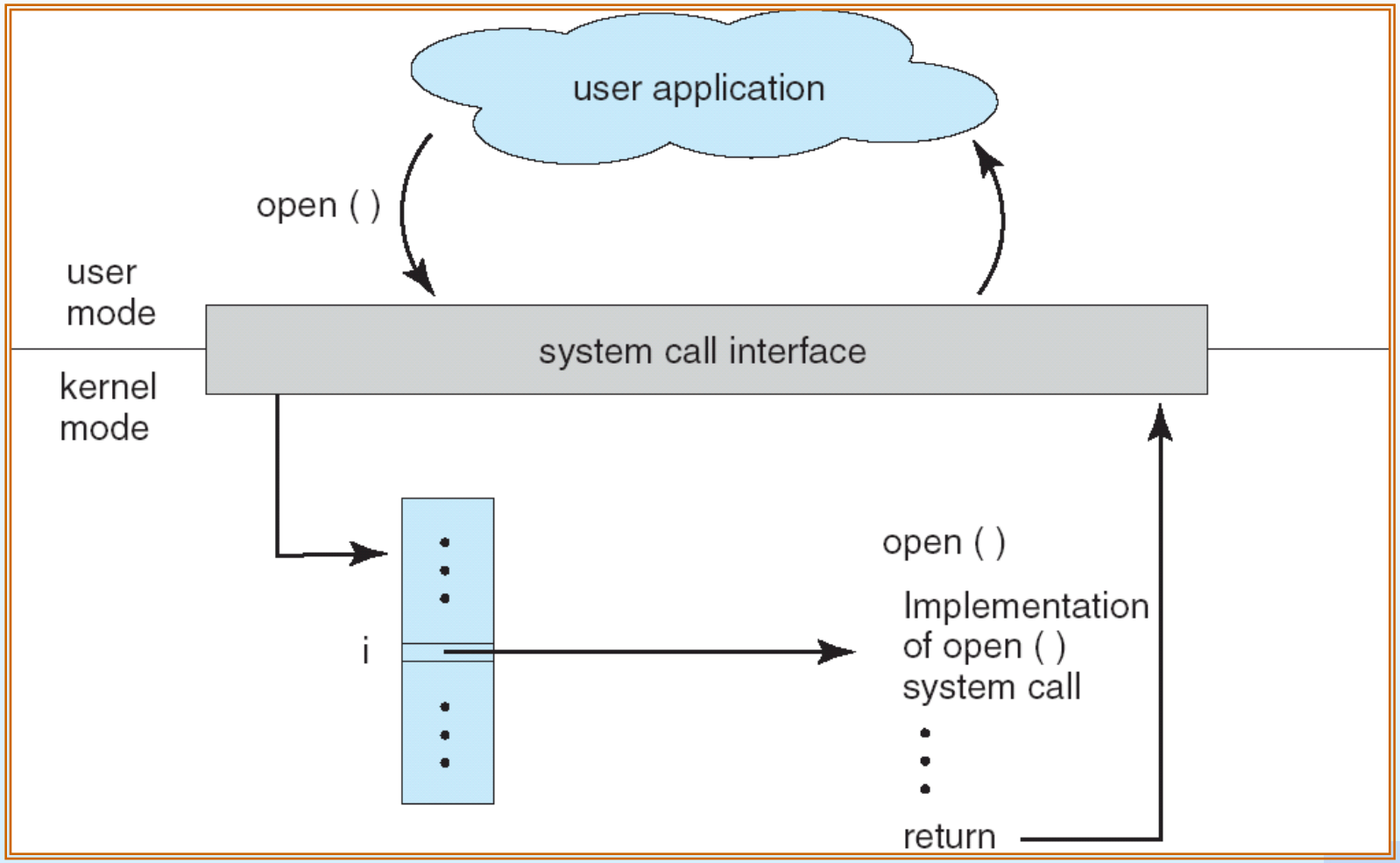
标准 API的例子

- 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



系统调用表

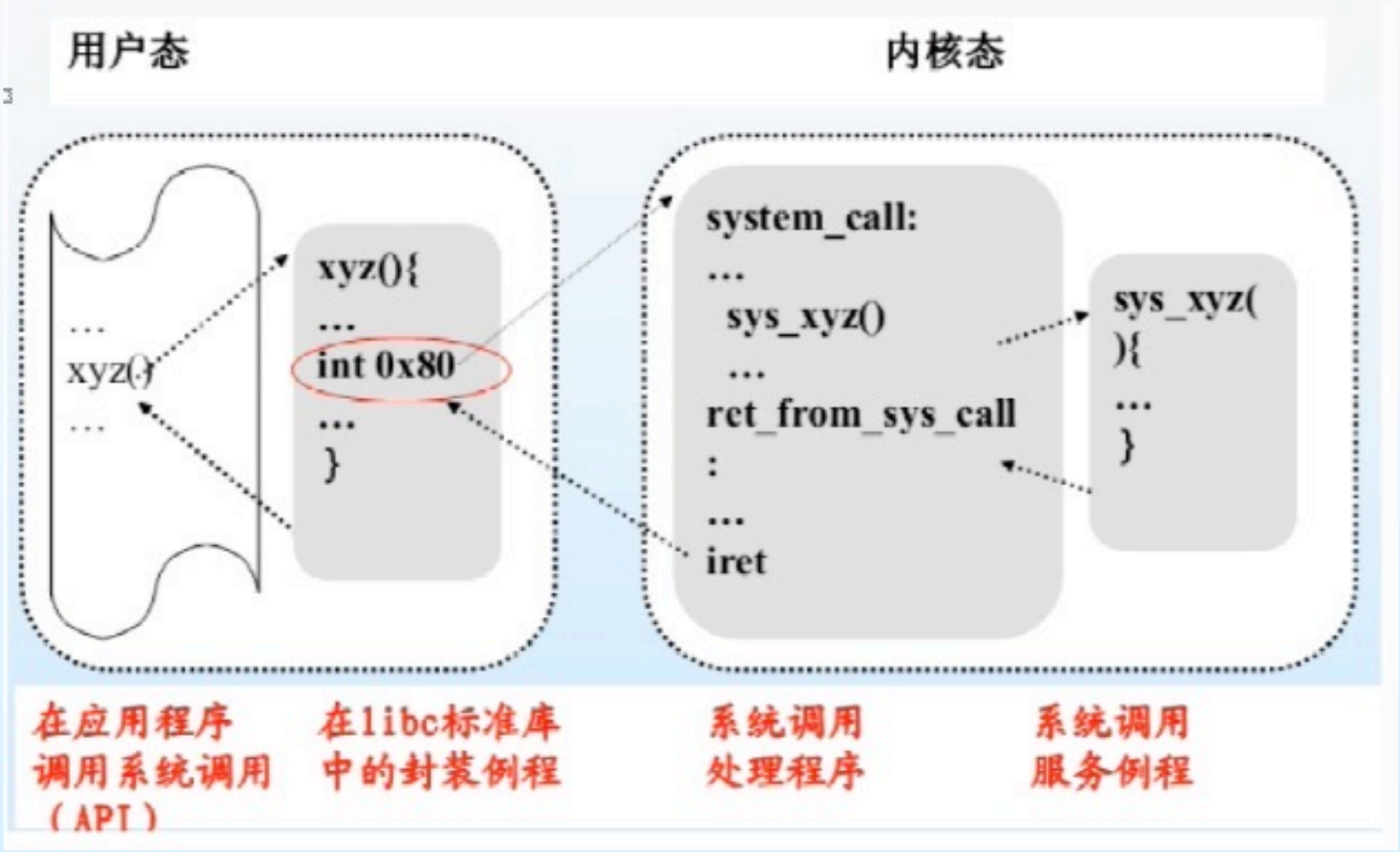
```
1 /*
2 * linux/arch/arm/kernel/calls.S
3 *
4 * Copyright (C) 1995-2005 Russell King
5 *
6 * This program is free software; you can redistribute it and/or modify
7 * it under the terms of the GNU General Public License version 2 as
8 * published by the Free Software Foundation.
9 *
10 * This file is included thrice in entry-common.S
11 */
12 /* 0 */ CALL(sys_restart_syscall)
13 CALL(sys_exit)
14 CALL(sys_fork_wrapper)
15 CALL(sys_read)
16 CALL(sys_write)
17 /* 5 */ CALL(sys_open)
18 CALL(sys_close)
19 CALL(sys_ni_syscall) /* was sys_waitpid */
20 CALL(sys_creat)
21 CALL(sys_link)
22 /* 10 */ CALL(sys_unlink)
23 CALL(sys_execve_wrapper)
24 CALL(sys_chdir)
25 CALL(OBSOLETE(sys_time)) /* used by libc4 */
```

Solaris 10 dtrace 动态跟踪工具

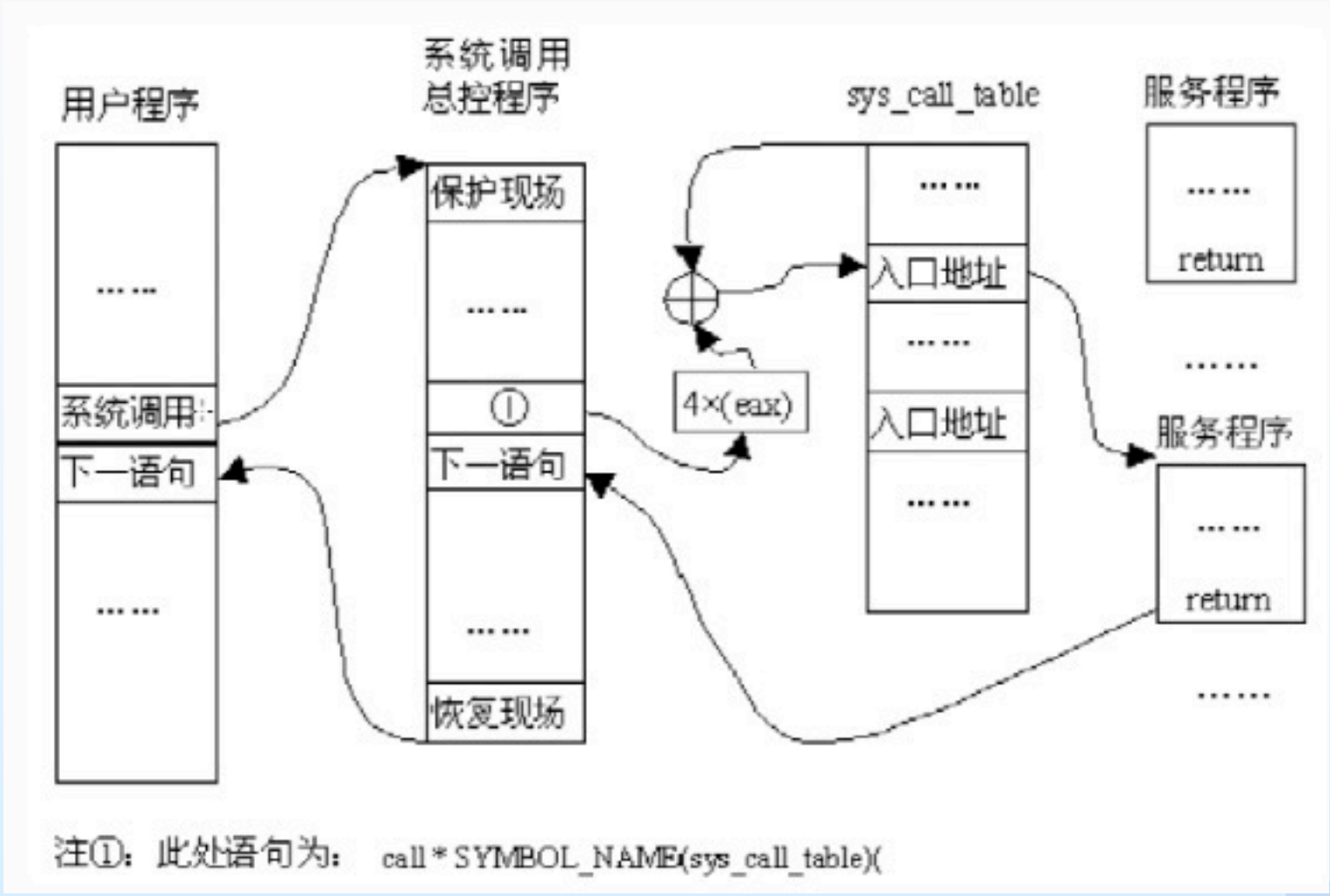
```
# ./all.d 'pgrep xclock' XEventsQueued
dtrace: script './all.d' matched 52377 probes
CPU FUNCTION                                 
  0 -> XEventsQueued                         U
  0   -> _XEventsQueued                      U
  0     -> _X11TransBytesReadable            U
  0     <- _X11TransBytesReadable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

U：用户模式
K：核心模式

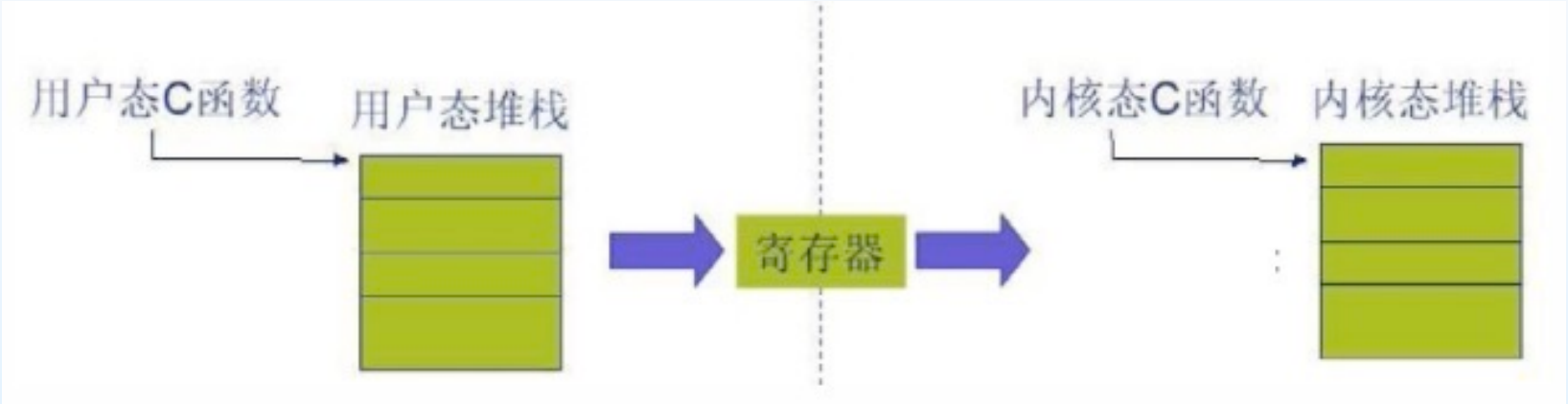
Linux系统调用例子



Linux系统调用执行过程

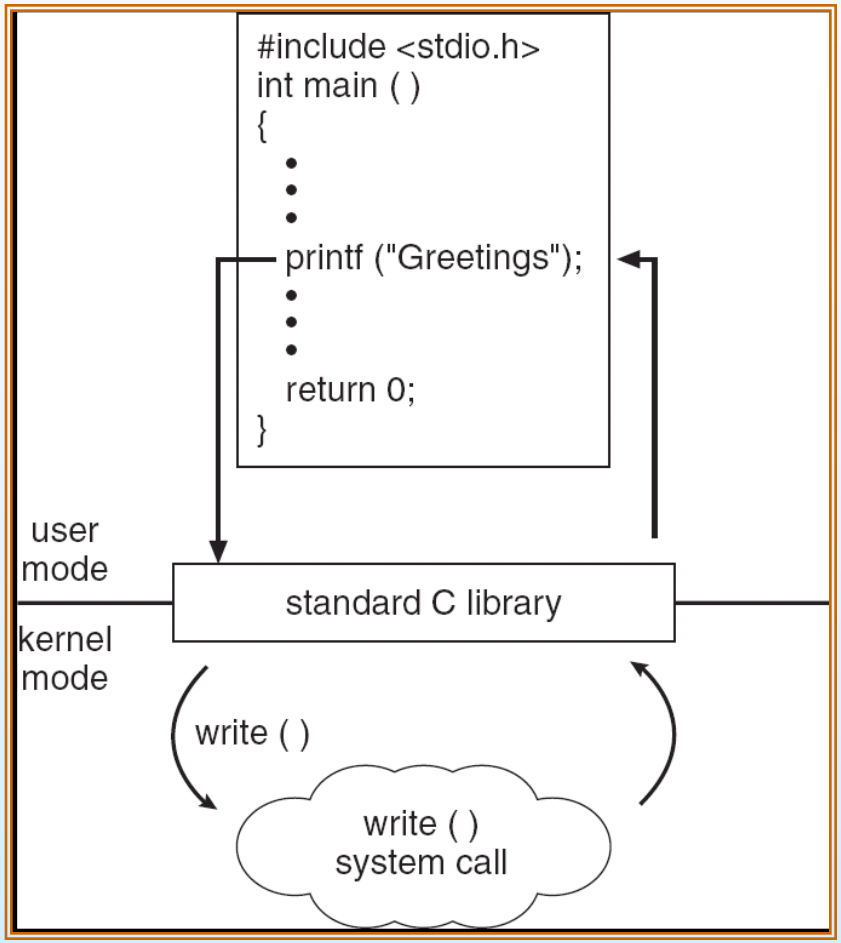


系统调用的参数传递



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() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

思考

系统调用只能在核心态运行

A. 正确
B. 错误

3、操作系统用户界面

- 用户接口-一般有两种形式
  - 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

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

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

7. 优点：简单、健壮、效率高
8. 缺点：使用不方便，界面不美观

Bourne Shell

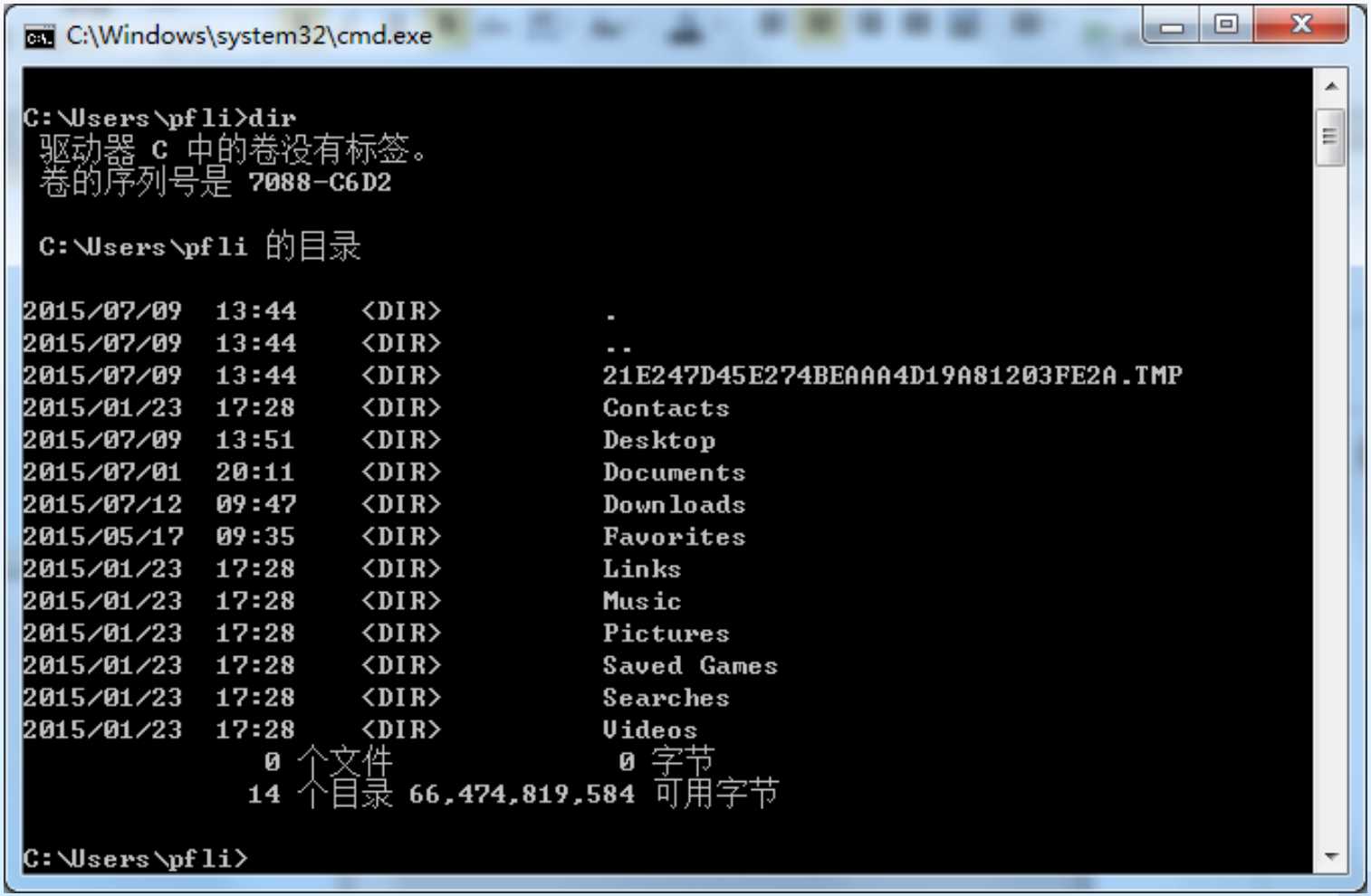
Default

New Info Close Execute Bookmarks

Default Default

```
PBG-Mac-Pro:~ pbg$ w
15:24  up 56 mins, 2 users, load averages: 1.51 1.53 1.65
USER     TTY      FROM              LOGIN@  IDLE WHAT
pbg      console  -                 14:34      50 -
pbg      s000     -                 15:05       - w
PBG-Mac-Pro:~ pbg$ 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:~ pbg$ 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:~ pbg$ pwd
/Users/pbg
PBG-Mac-Pro:~ pbg$ 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:~ pbg$
```

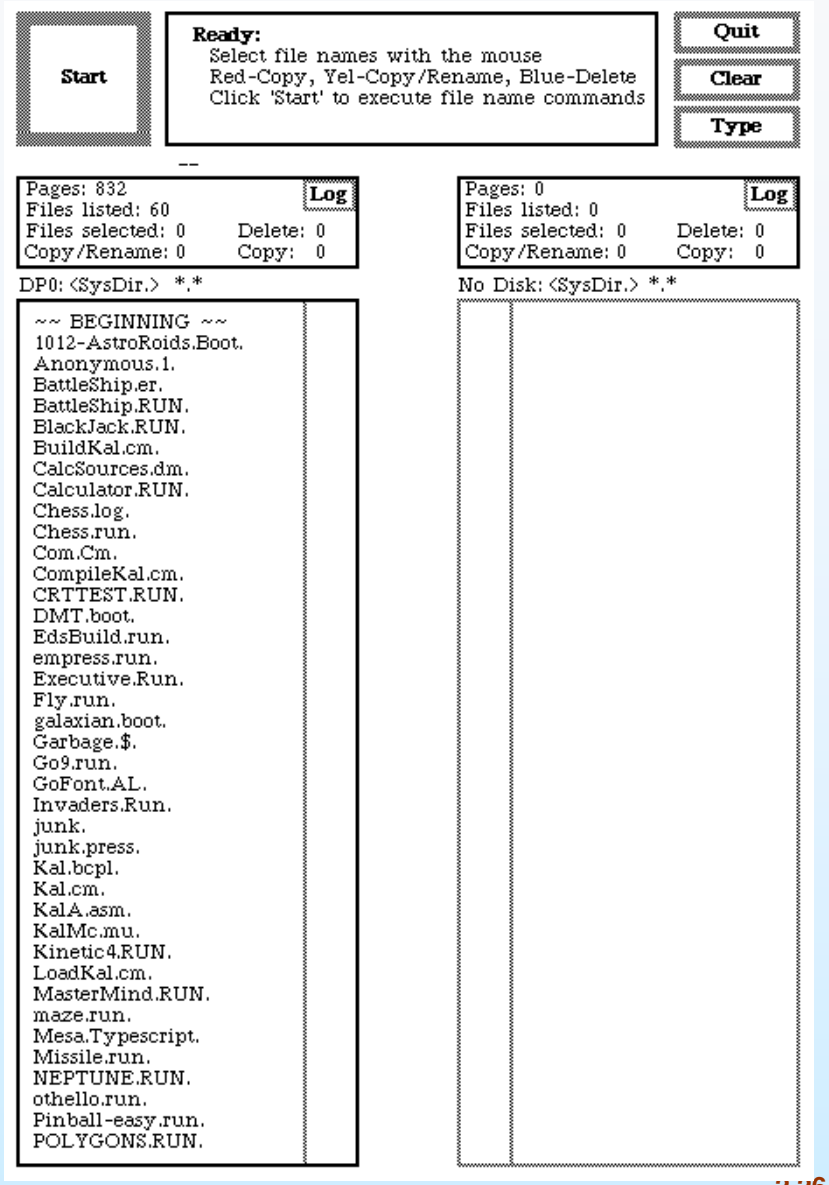
MS-DOS

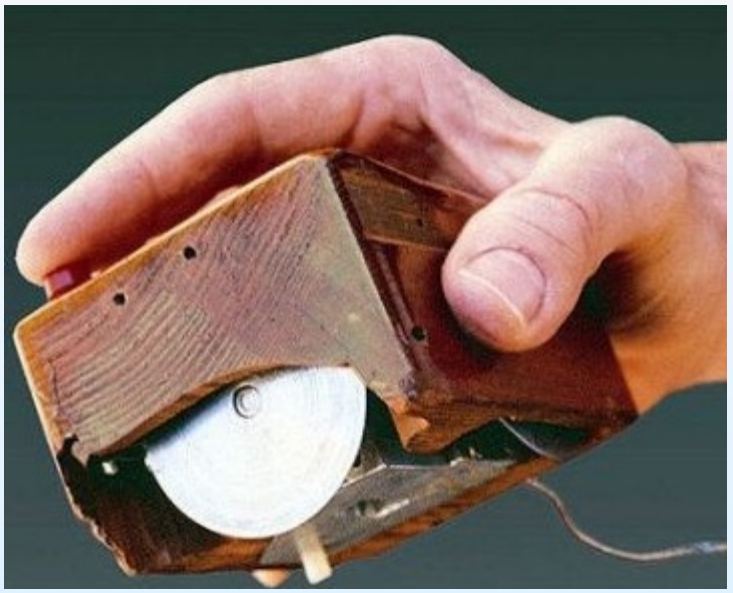

C:\Windows\system32\cmd.exe
C:\Users\pfli>dir
驱动器 C 中的卷没有标签。
卷的序列号是 7088-C6D2
C:\Users\pfli 的目录
2015/07/09 13:44 <DIR> .
2015/07/09 13:44 <DIR> ..
2015/07/09 13:44 <DIR> 21E247D45E274BEAAA4D19A81203FE2A.TMP
2015/01/23 17:28 <DIR> Contacts
2015/07/09 13:51 <DIR> Desktop
2015/07/01 20:11 <DIR> Documents
2015/07/12 09:47 <DIR> Downloads
2015/05/17 09:35 <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
  - ① 通常使用鼠标、键盘和监视器
  - ② 常用元素：图标、窗口、滚动条等
  - ③ **图标**代表文件、程序、系统功能等
  - ④ 不同对象上鼠标按钮导致不同的动作
  - ⑤ 用户命令以**鼠标操作**为主
  - ⑥ GUI首次出现在 Xerox PARC
  - ⑦ 优点：操作方便、界面直观、漂亮
  - ⑧ 缺点：效率差、不够健壮

第一个图形界面——Xerox Alto





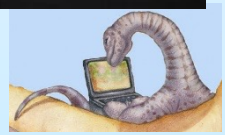
The Mac OS X GUI

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

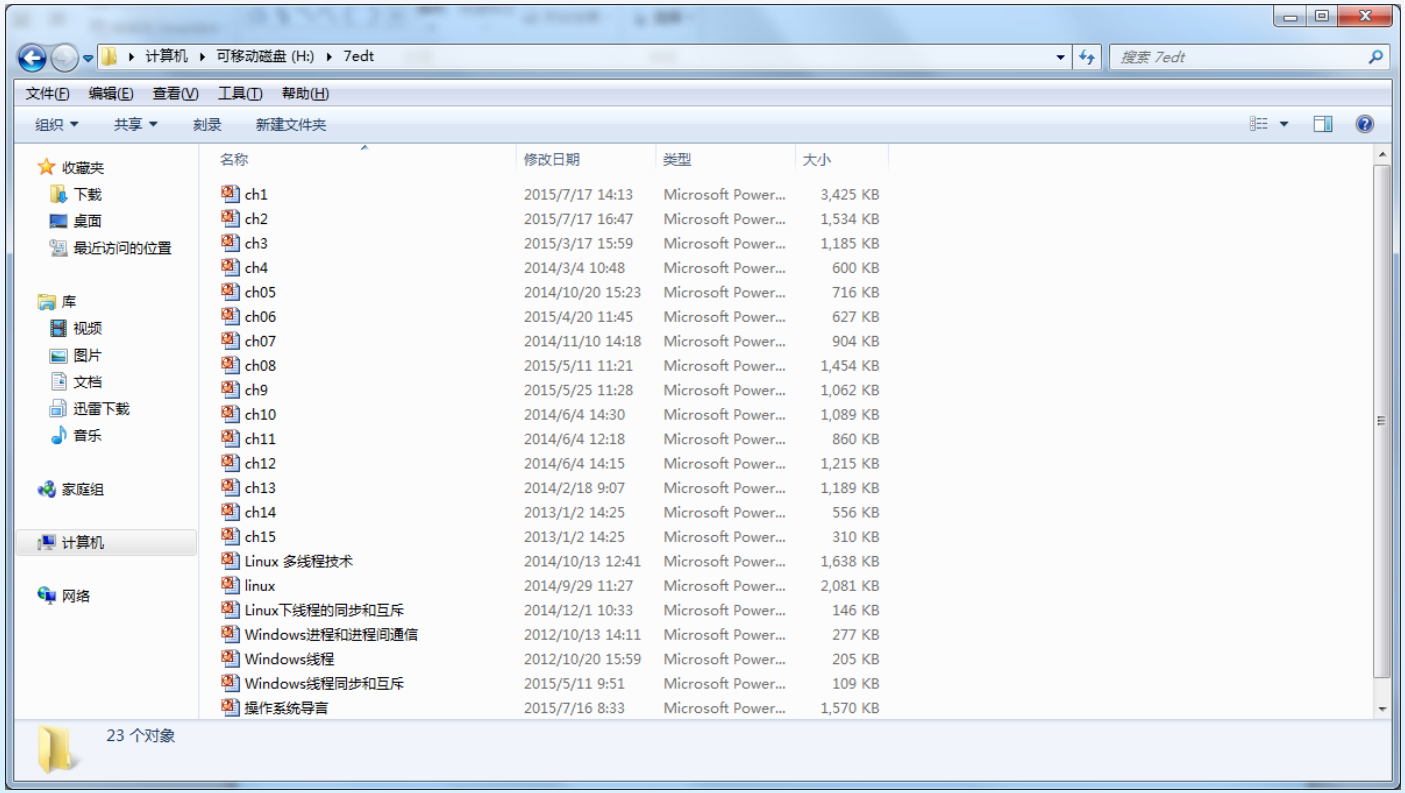
系统程序

- 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

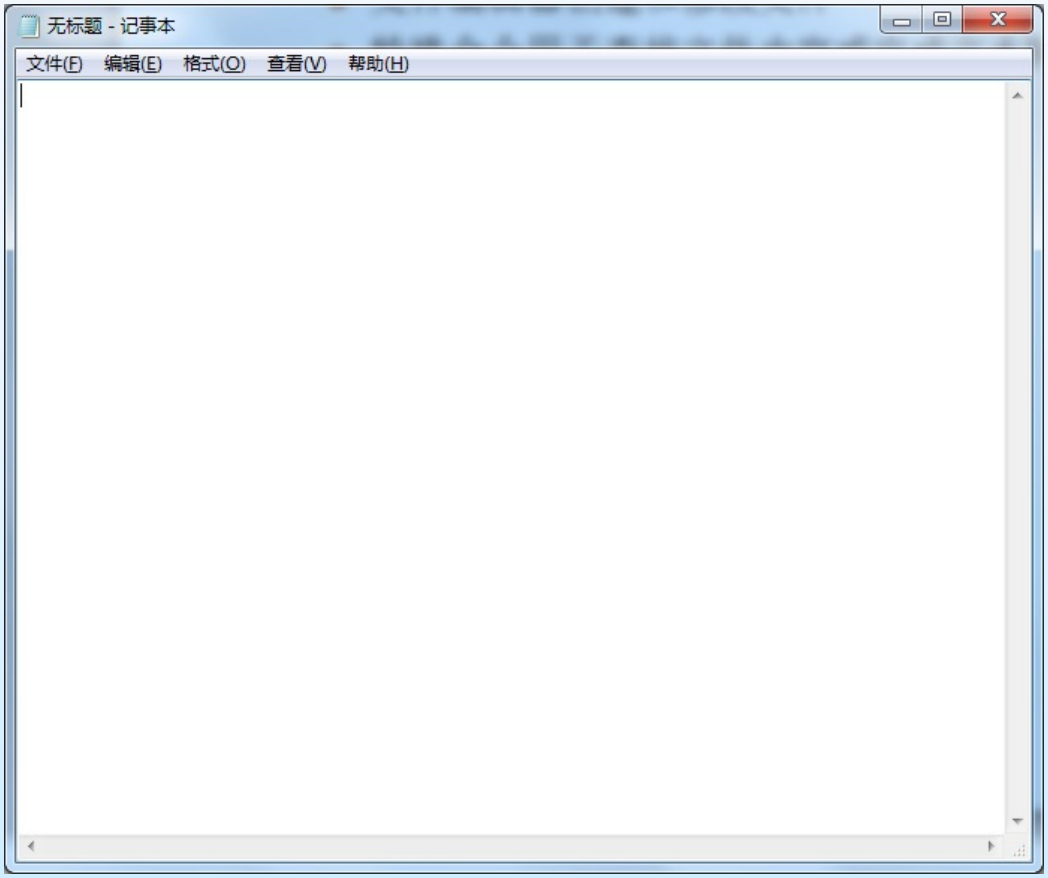


系统程序功能

- 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



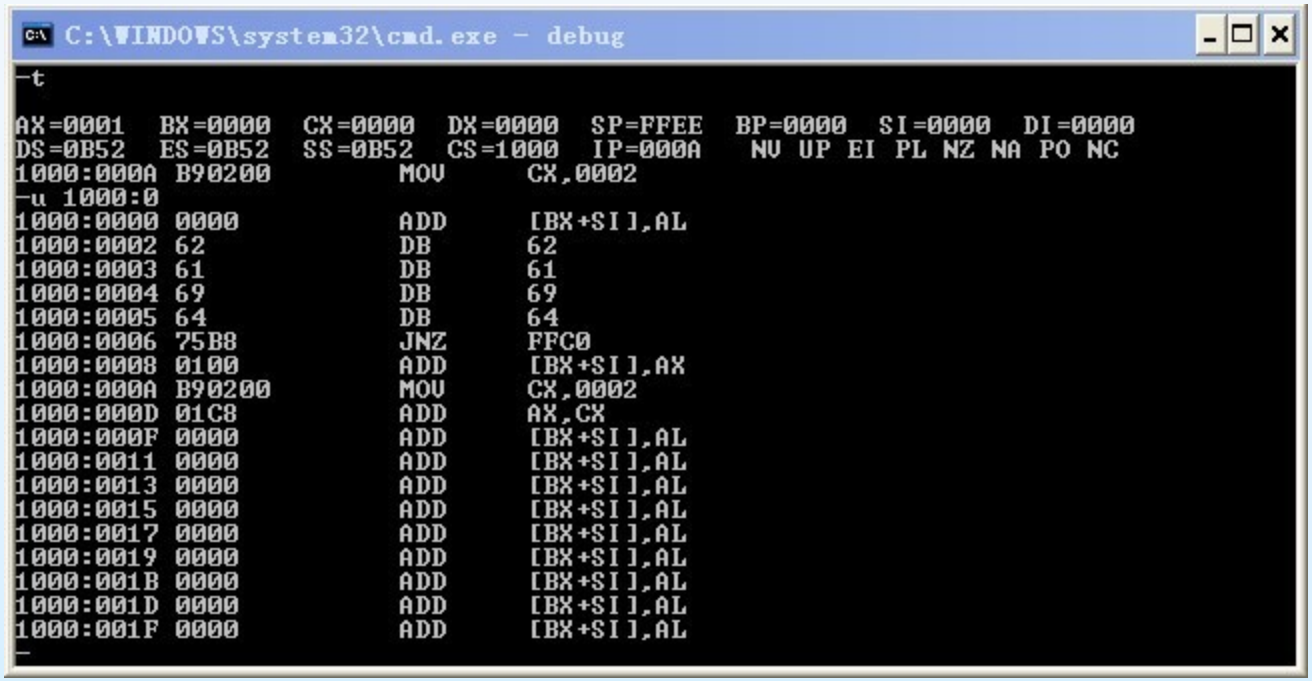
系统程序功能

- 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



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

思考

手机操作系统常用的人机交互界面是GUI。

A. 正确
B. 错误

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）

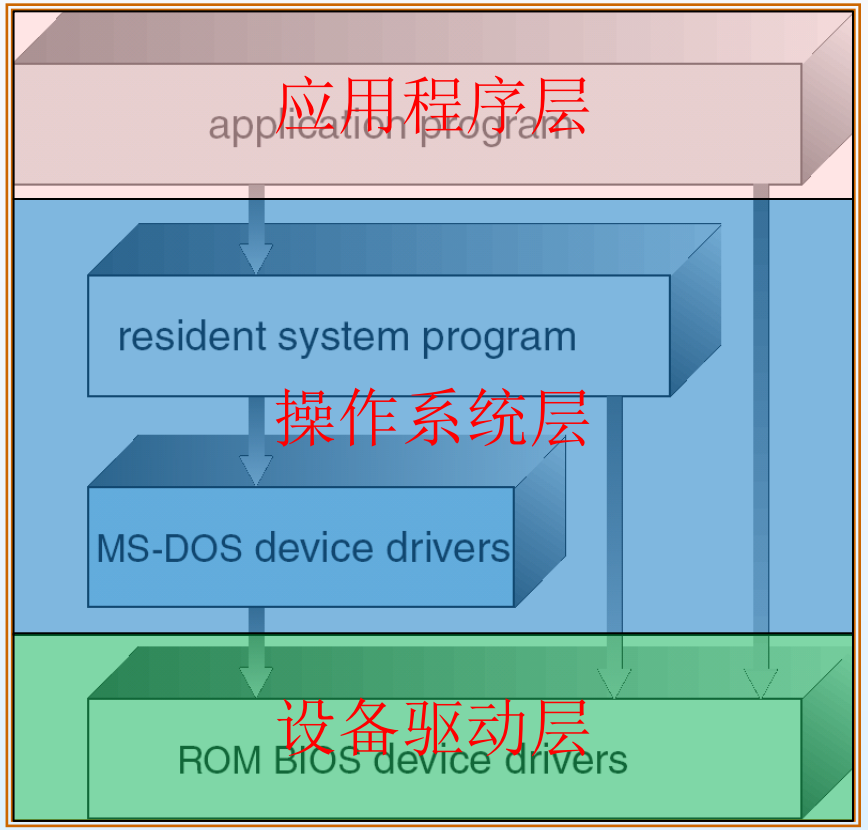
通用操作系统是个庞然大物

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

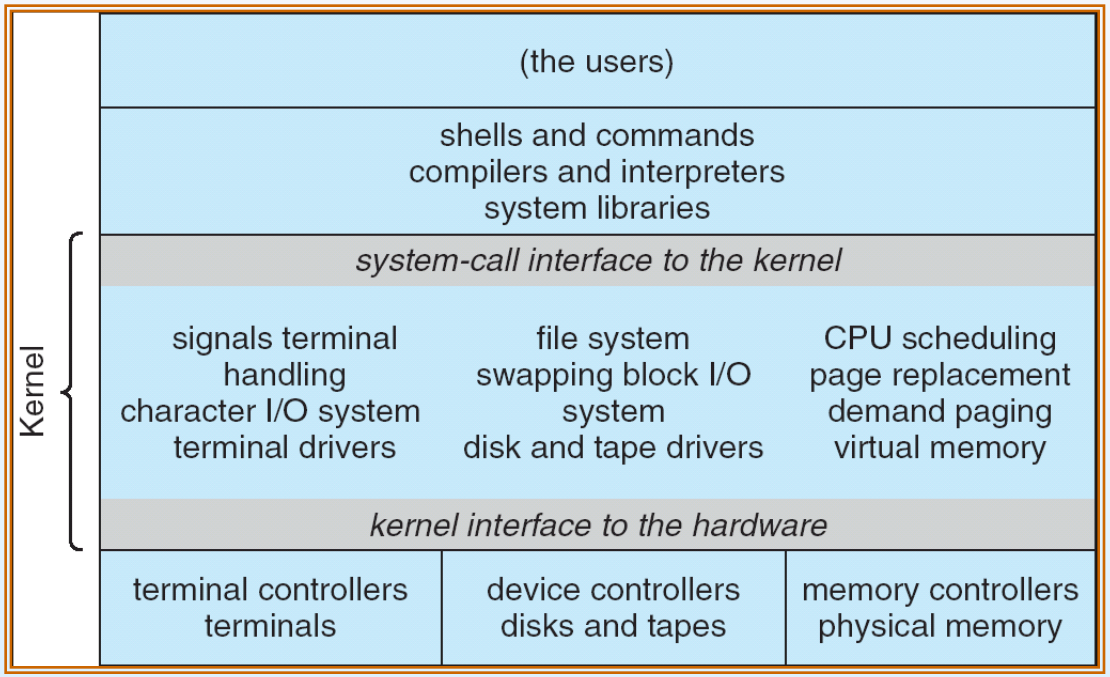
操作系统结构设计

- 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

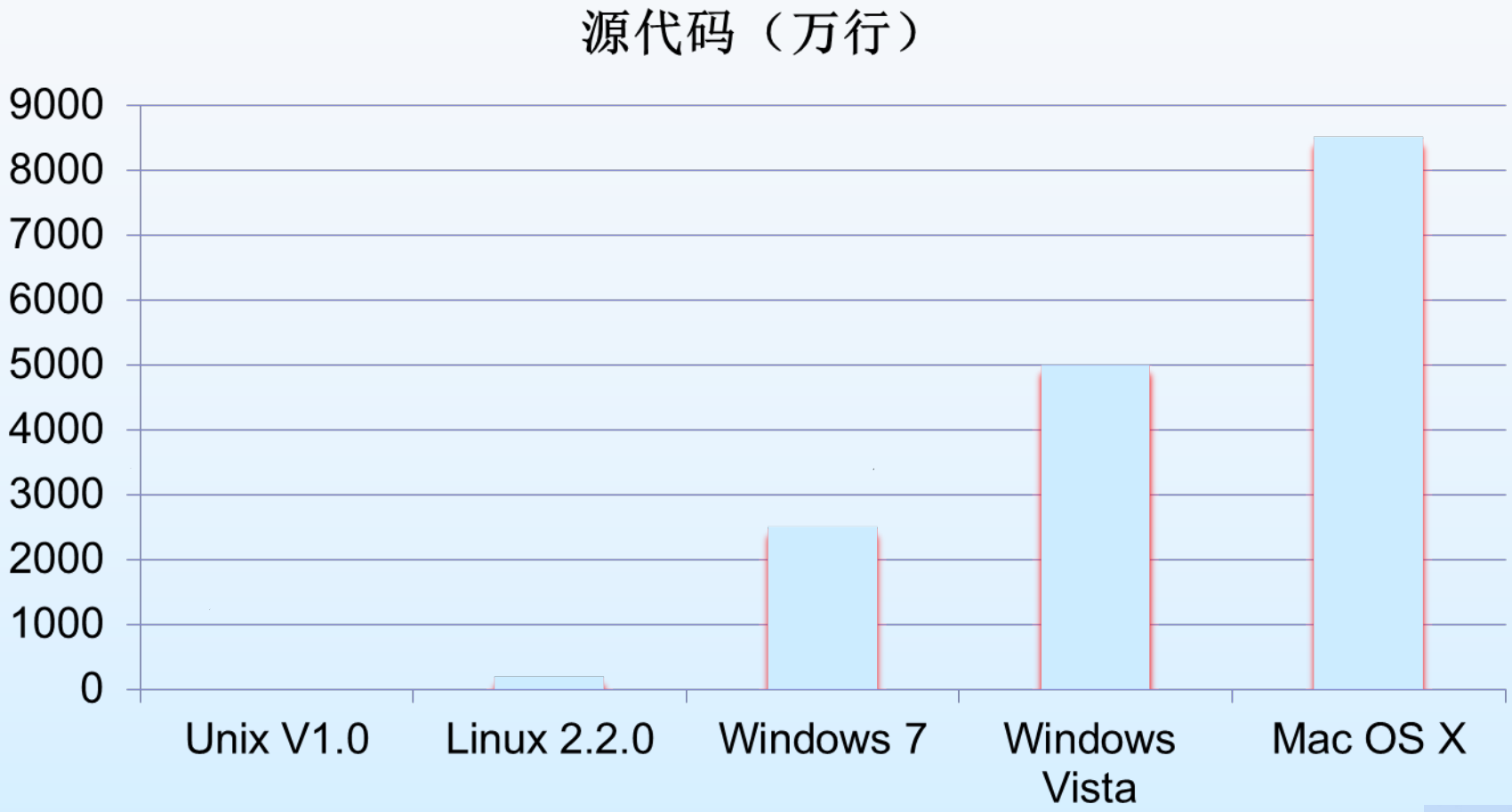


简单结构-早期UNIX

- 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



操作系统规模

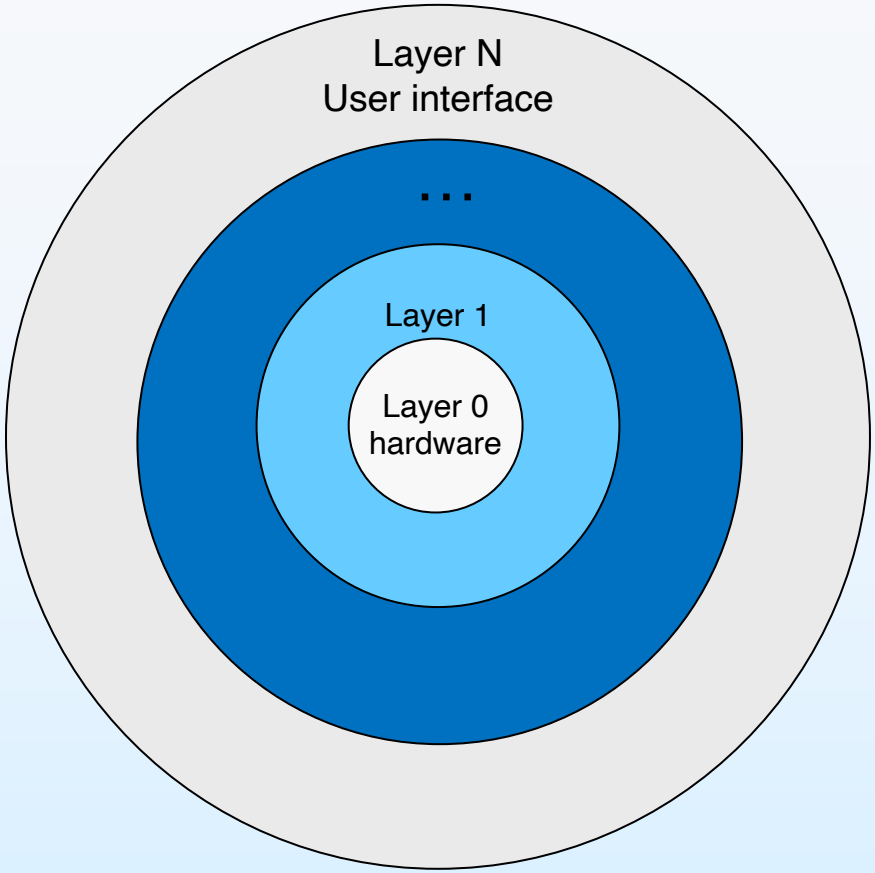

源代码（万行）
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
Unix V1.0
Linux 2.2.0
Windows 7
Windows Vista
Mac OS X

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

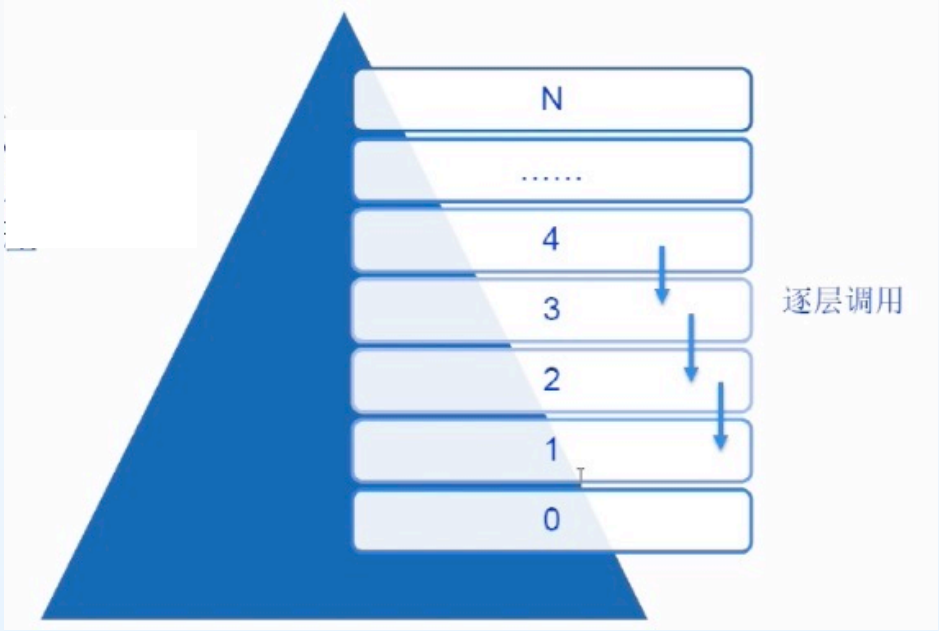
操作系统结构设计

- 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



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



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

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

第二层 储存管理

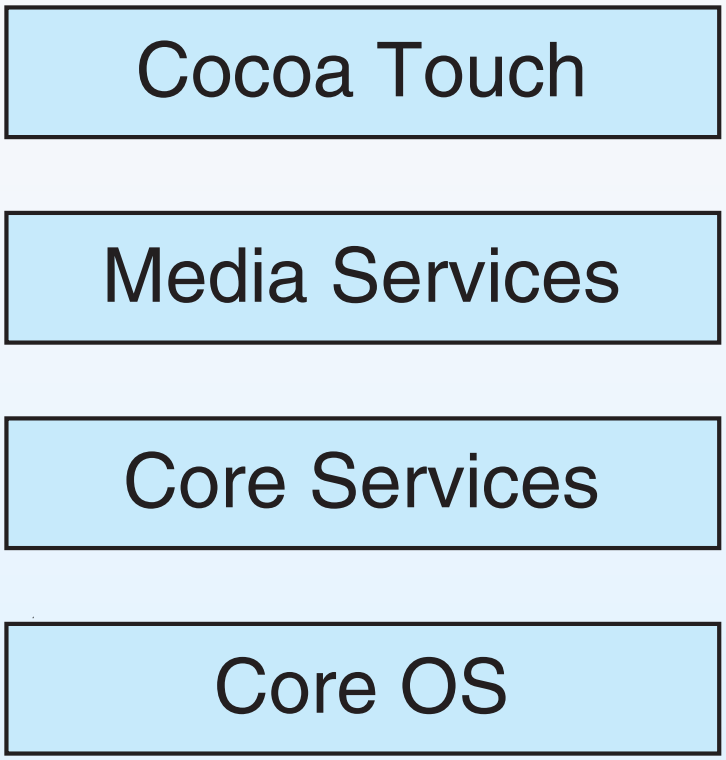
第一层 CPU调度与信号

操作系统

第零层 硬件设施

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

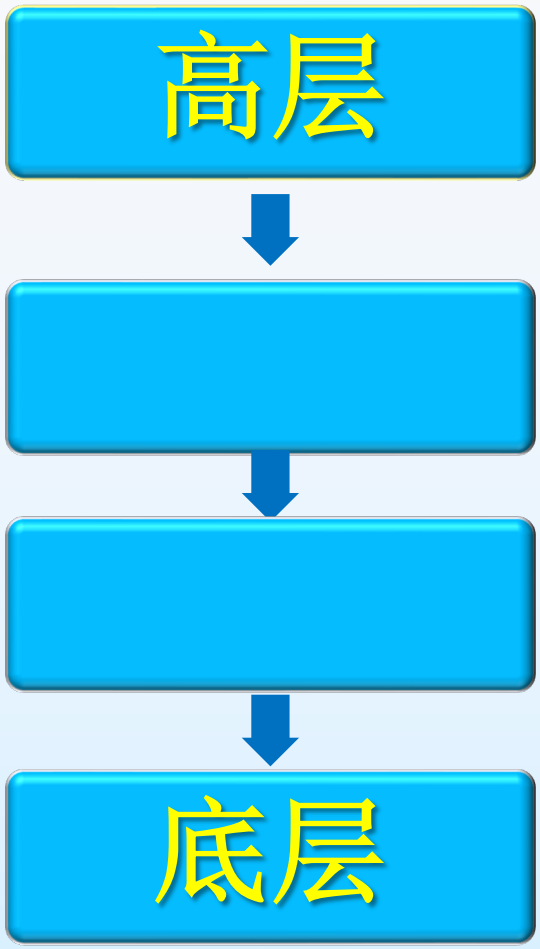


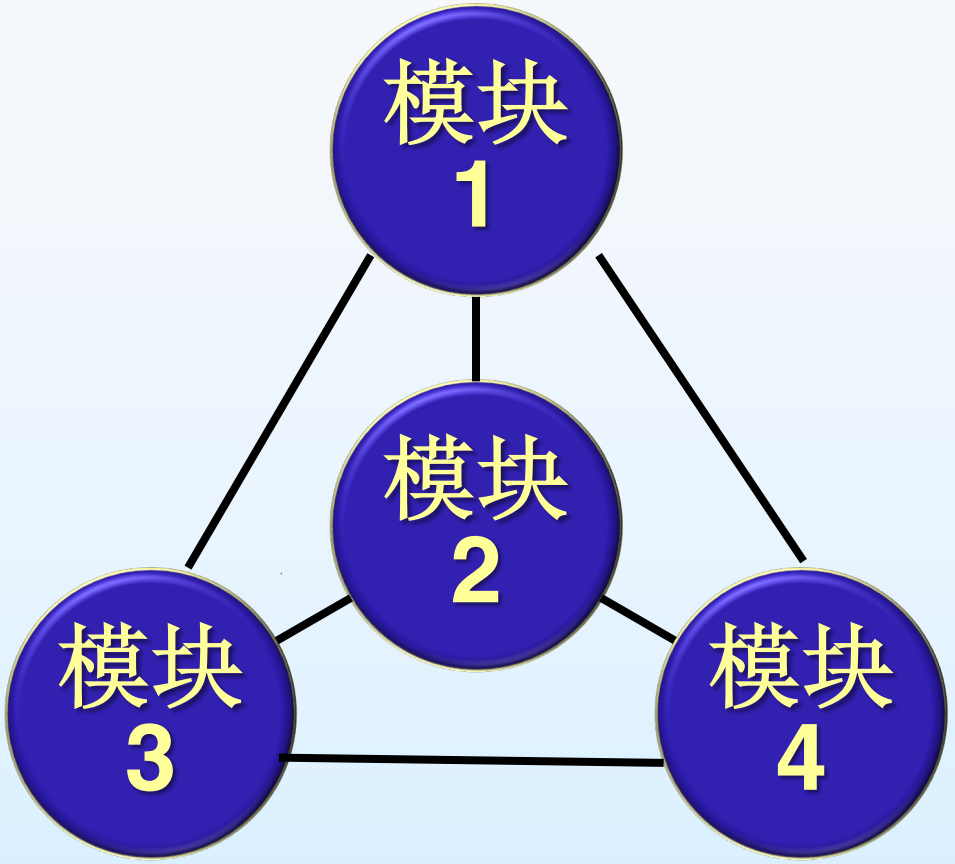
- 在层次结构中，第n层能够调用第（）层提供的服务。
  - A. n-1
  - B. n+1
  - C. 0~n-1
  - D. 以上都不对

模块结构

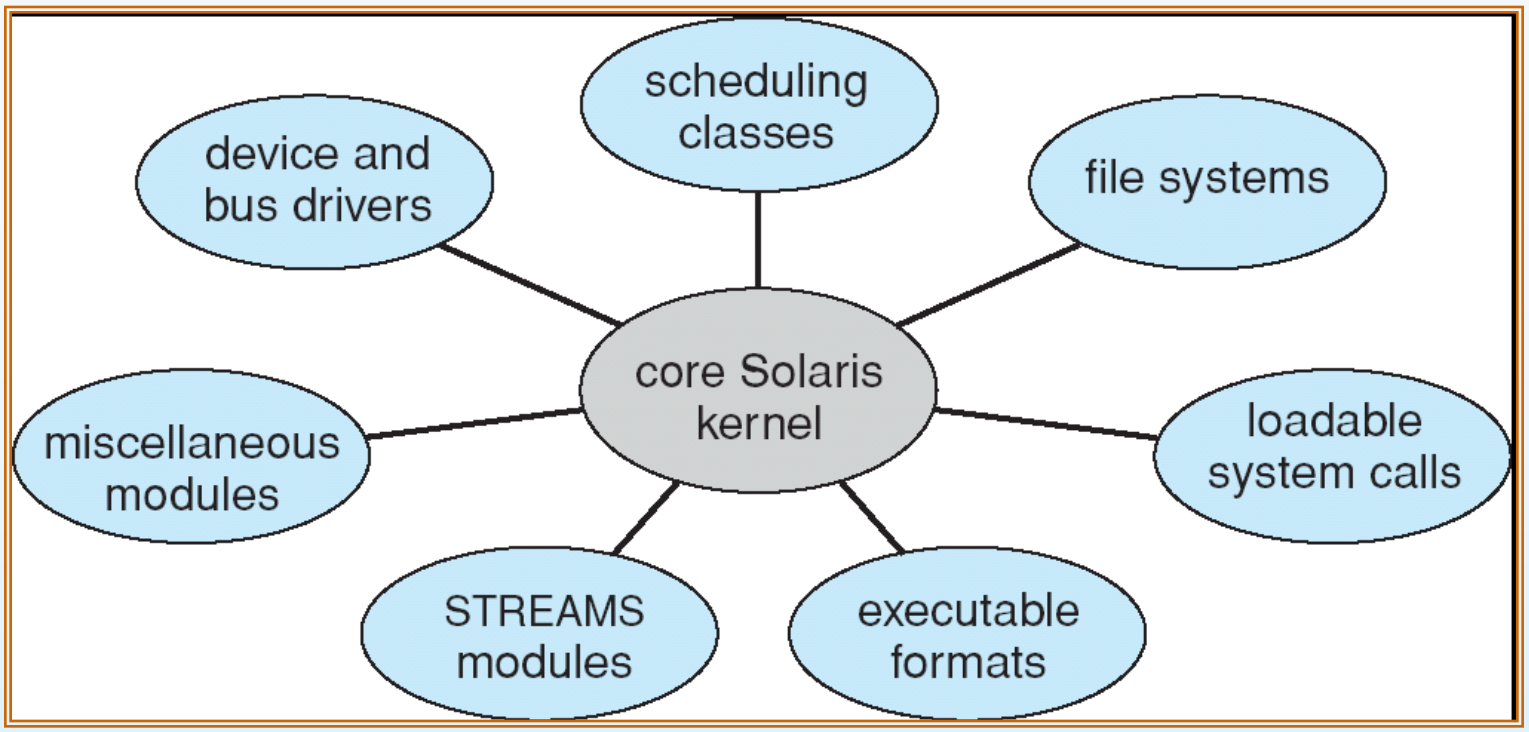
- 大部分现代操作系统采用模块结构
  - 使用面向对象方法
  - 每个核心部件分开为多个模块
  - 每个与其他组件（模块）的会话被称为接口
  - 每个组件（模块）在需要时被加载到内核
- 类似于分层方法，但更灵活

- 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


高层
底层


模块
1
模块
2
模块
3
模块
4

Solaris 模块

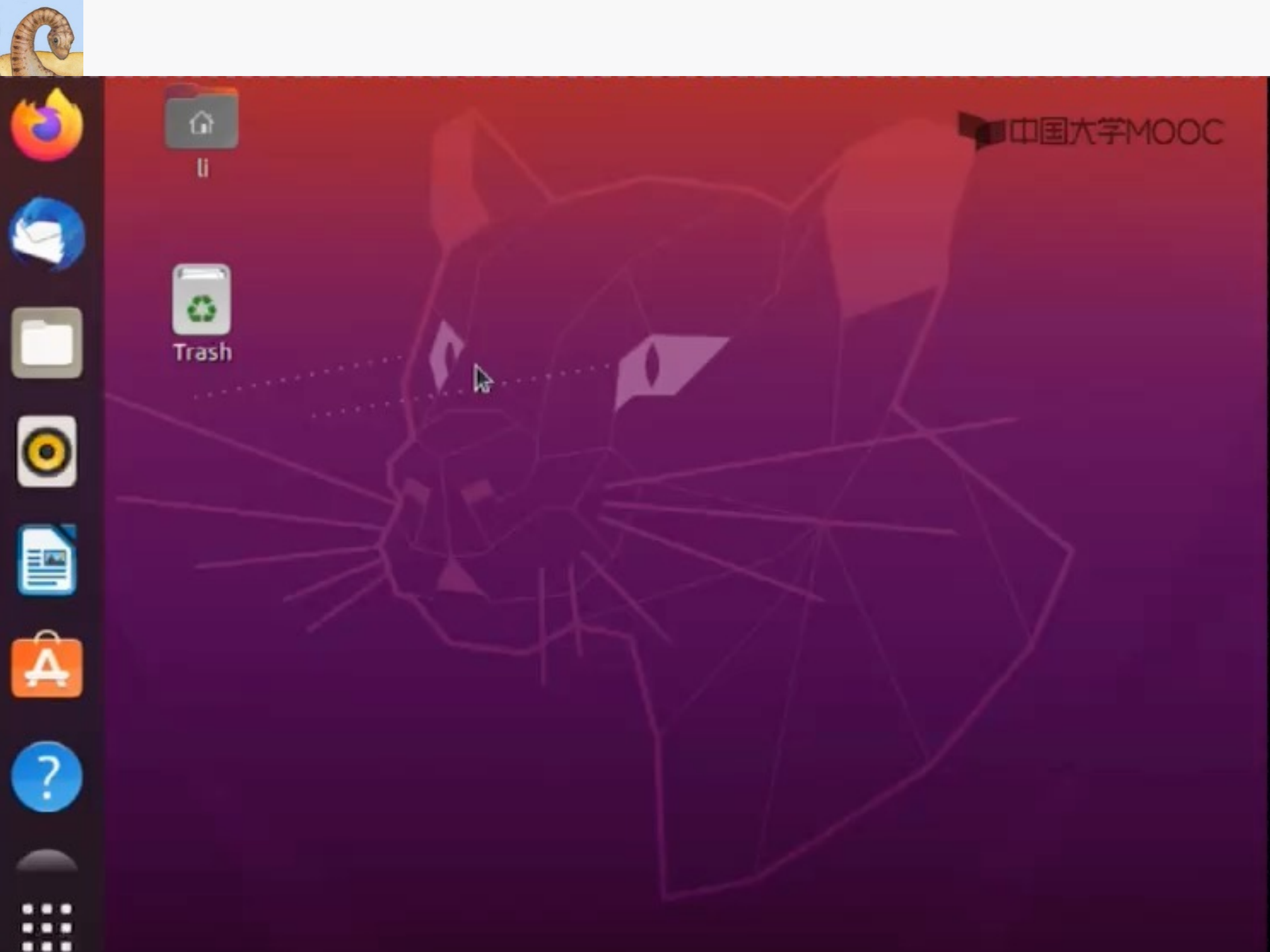

scheduling classes
device and bus drivers
file systems
core Solaris kernel
miscellaneous modules
loadable system calls
STREAMS modules
executable formats

例子：Linux模块

- 模块代码

```
#define MODULE
#include <linux/module.h>
int init_module(void){
        printk("<1>Hello world!\n");
        return 0;
}
void cleanup_module(void){
        printk("<1>Goodbye!\n");
}
MODULE_LICENSE("GPL");
```

14

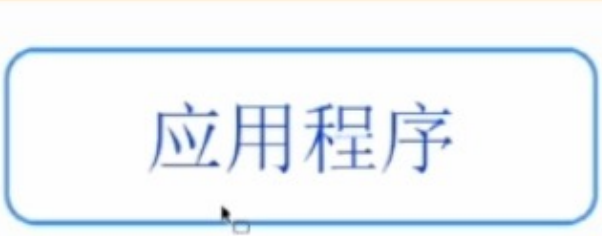

li
Trash
中国大学MOOC

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

中国大学MOOC

微内核

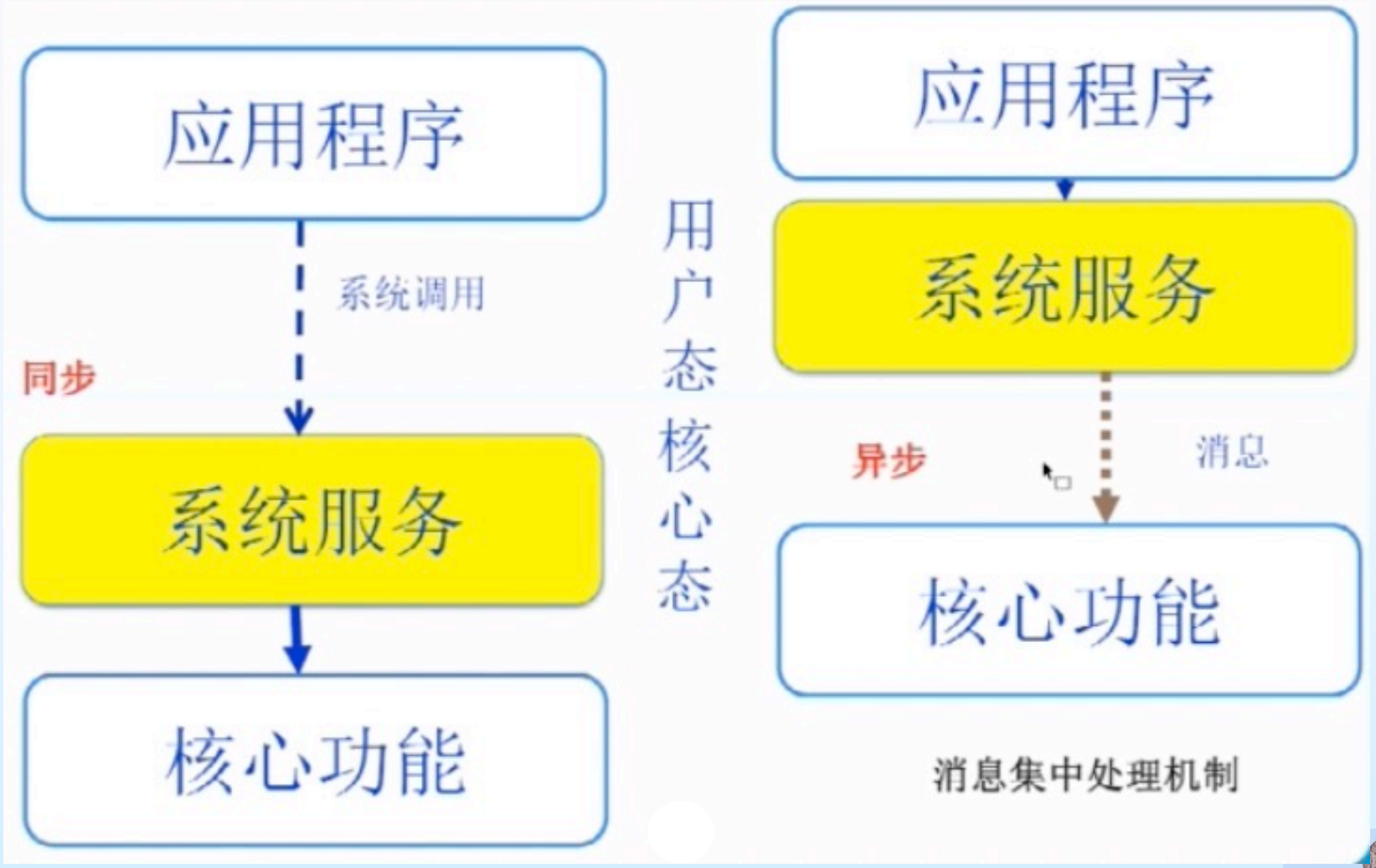


用户态

核心态

18

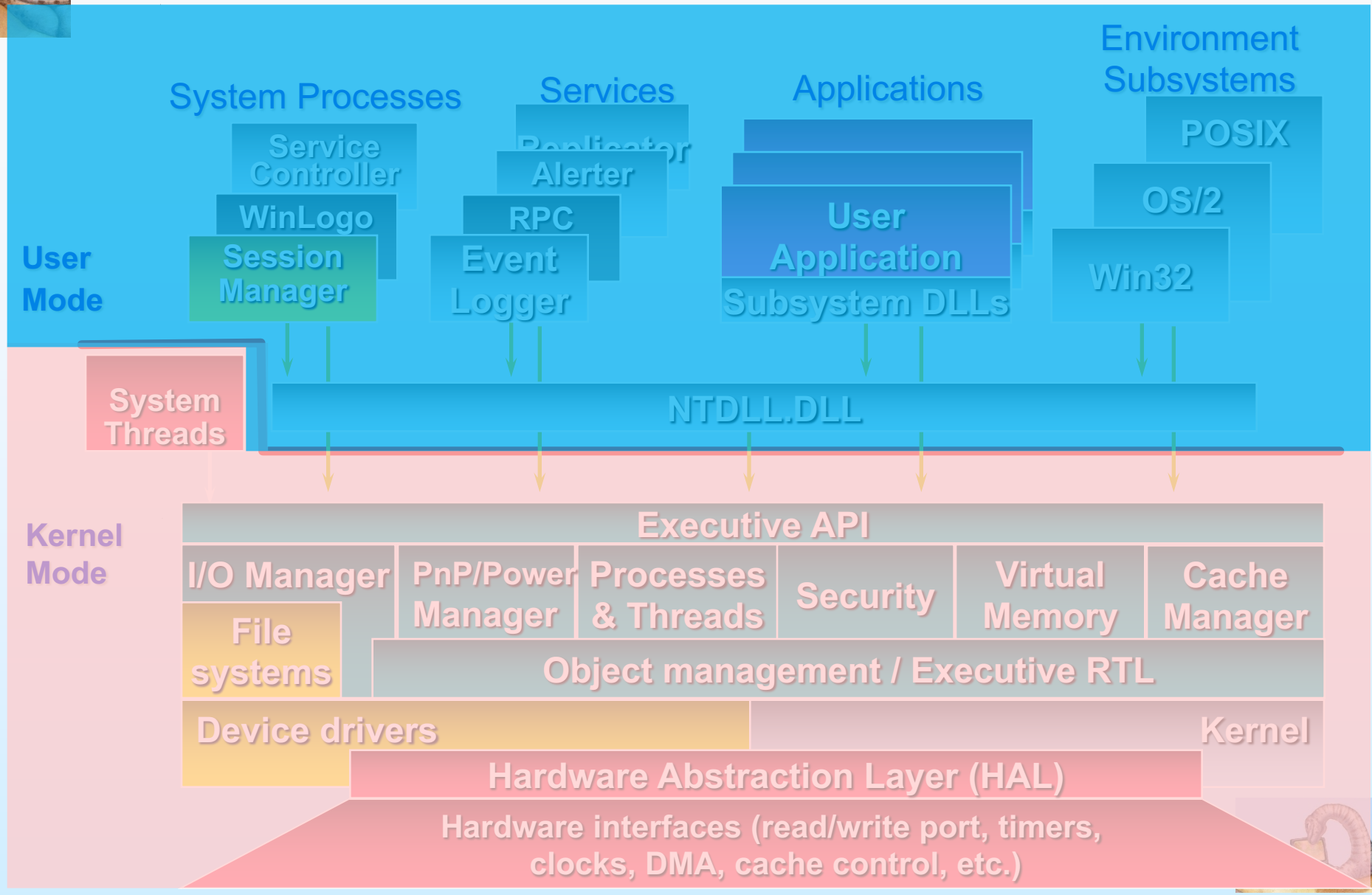
微内核



微内核系统

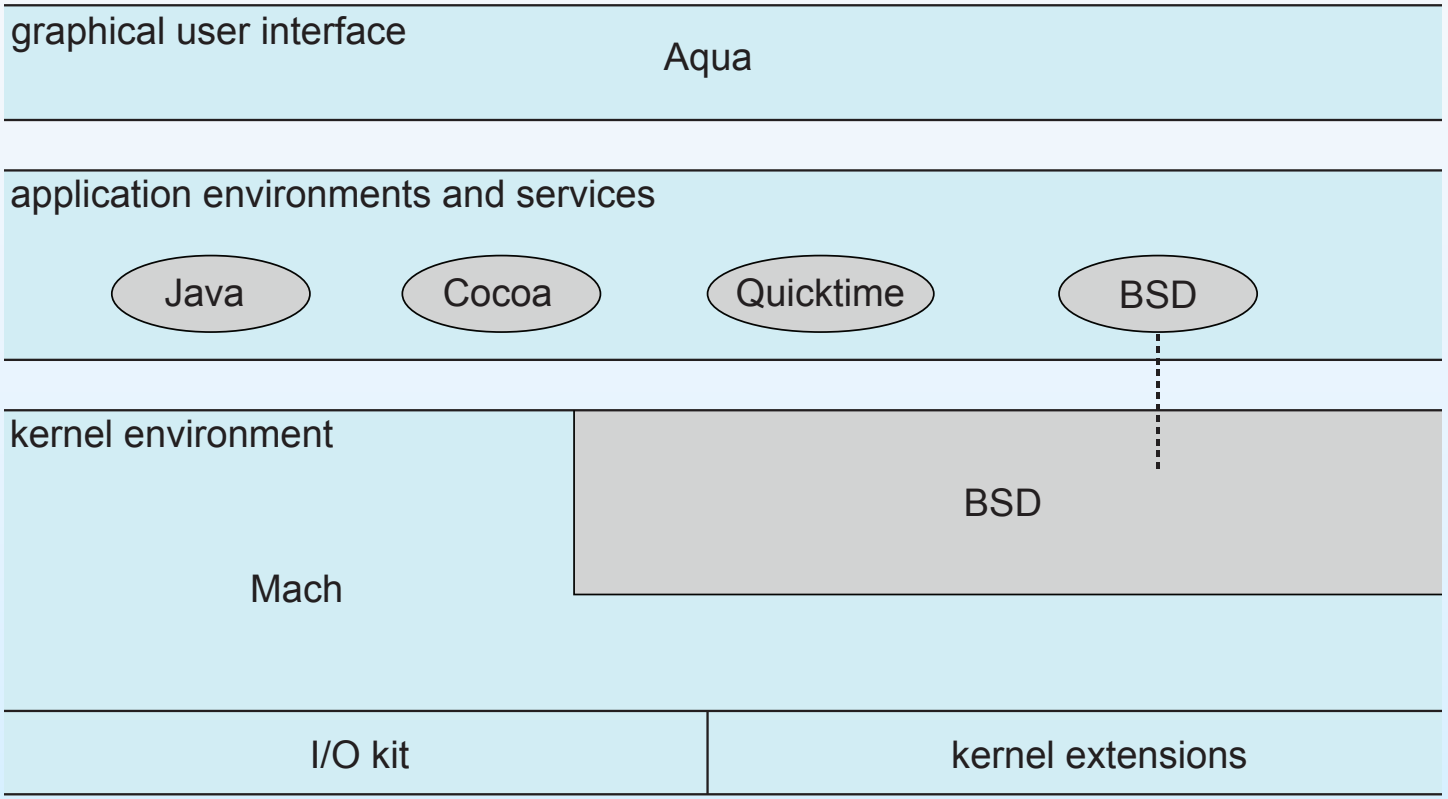
- 第一个微内核系统：CMU的Mach
  - http://www.cs.cmu.edu/afs/cs/project/mach/public/www/mach.html
- Tru64 Unix使用Mach内核
- QNX-基于微内核的实时操作系统
- Windows NT，2000，2003等

Windows 2000 微内核结构



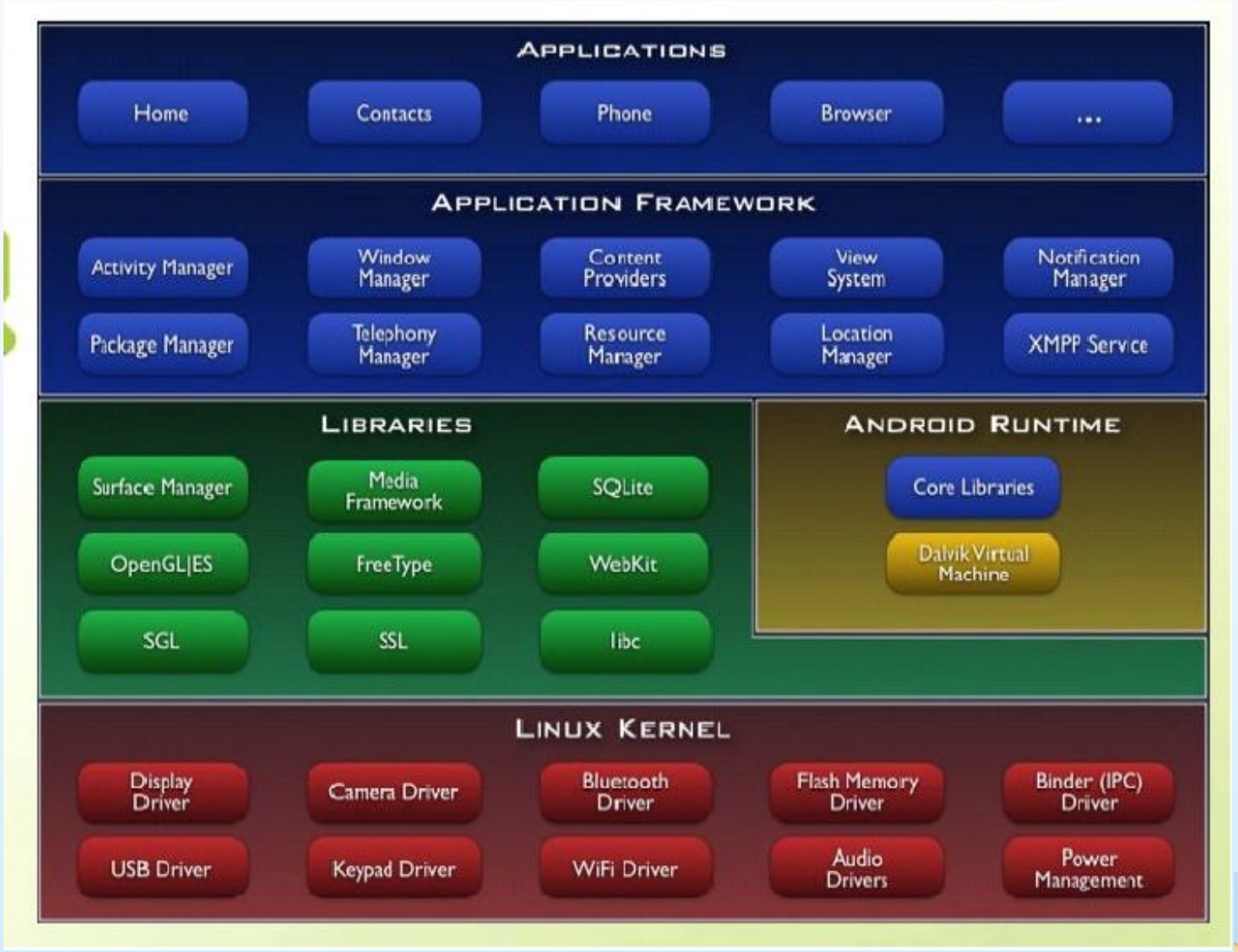
混合结构--Mac OS X

- 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



以下操作系统中，采用微内核结构的是（）。

A.Unix **B.**Mac OS X **C.**Windows 2000 **D.**Andriod

6、虚拟机（VM）

- 什么是虚拟机？
- 三种类型的虚拟机
  - JAVA虚拟机
  - 工作站虚拟机
  - 服务器虚拟机

虚拟机 Virtual Machines

- 通过软件模拟的具有完整硬件系统功能的、运行在一个完全隔离环境中的完整计算机系统
- 物理计算机的资源被共享，以创建虚拟机
- 虚拟机概念提供对系统资源的完全保护，因为每个虚拟机同其他虚拟机隔离
- 虚拟机是研发操作系统的完美载体
- 由于需要对物理机器进行精确复制，虚拟机实现困难

虚拟机实现

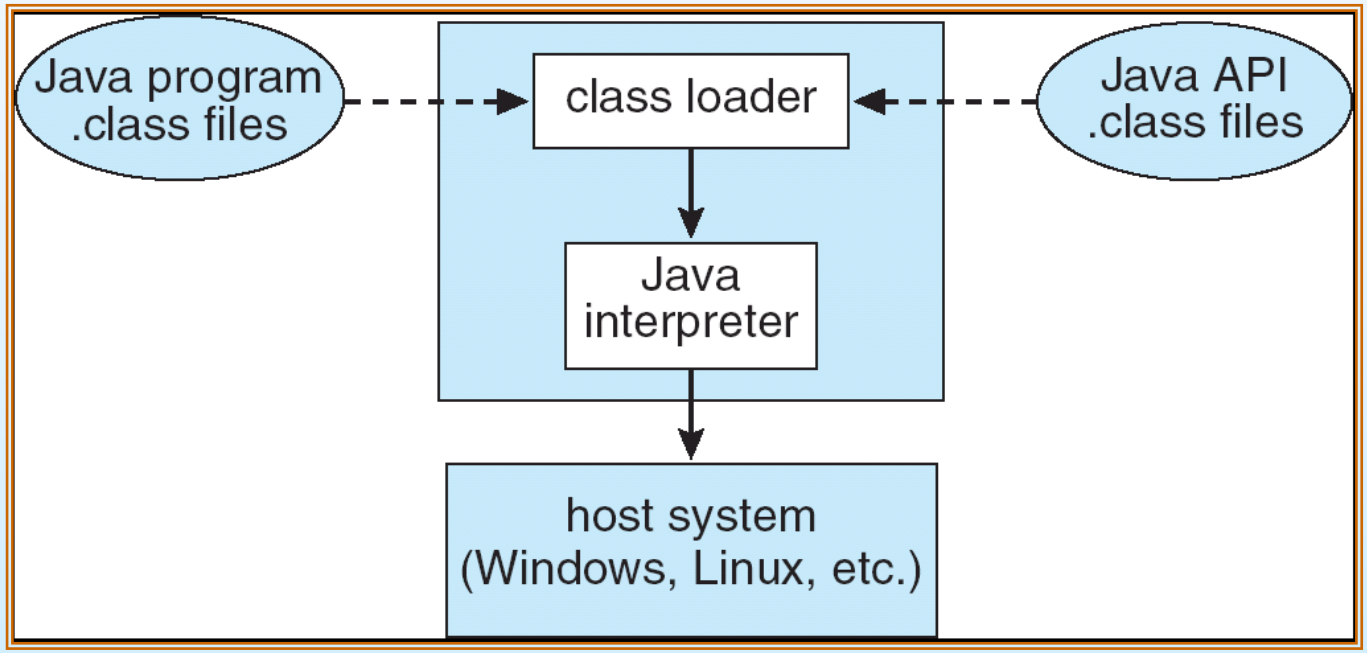
- 虚拟机实现的三种途径
  - 高级语言虚拟机
  - 工作站虚拟机
  - 服务器虚拟机

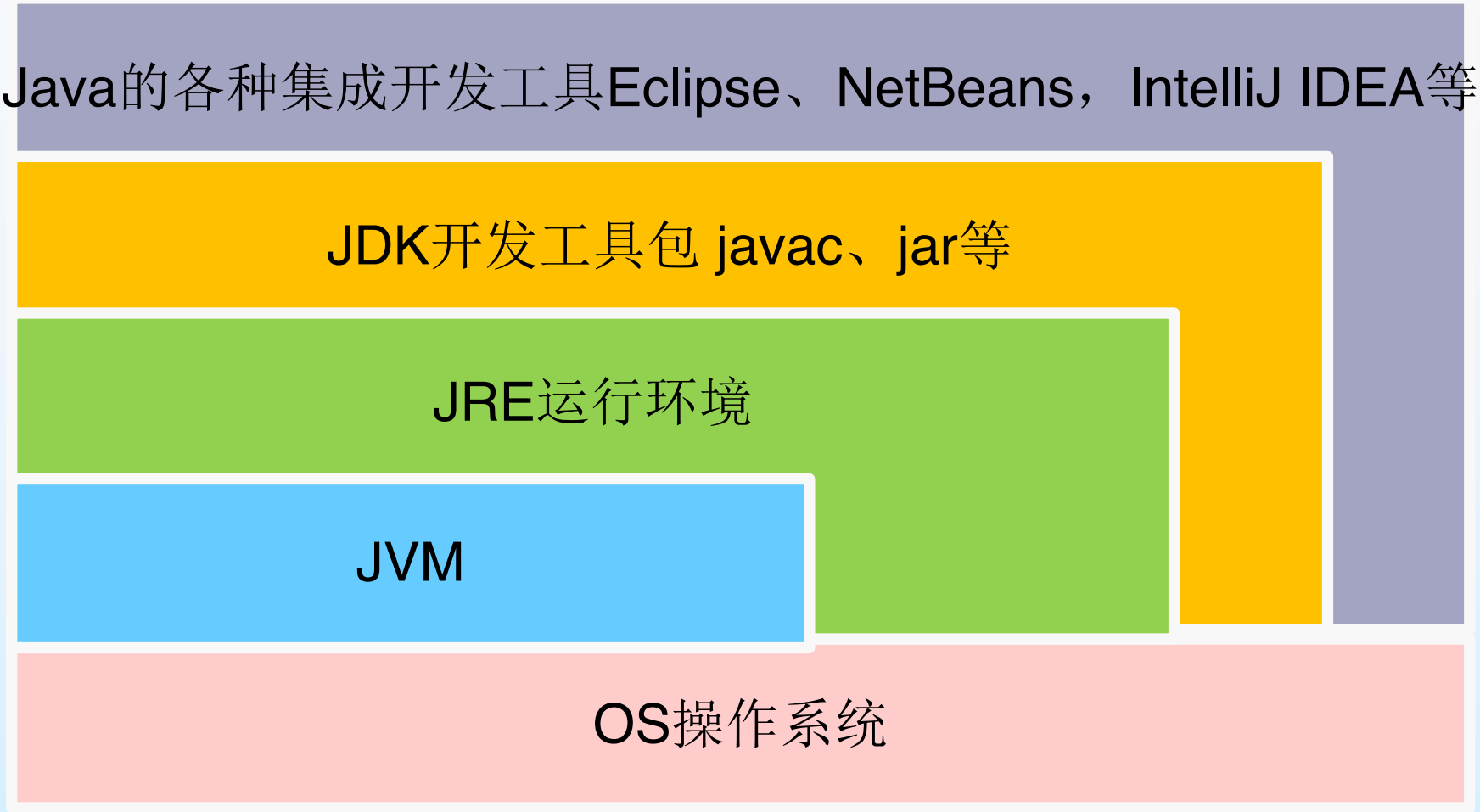
虚拟机实现

- 高级语言虚拟机主要功能是提供一个代码运行的容器，并模拟代码的执行，使得代码能够跨平台运行。
- 工作站虚拟机建立在操作系统之上，是操作系统中的操作系统，也称为Guest OS（客户操作系统），目的是多个操作系统可以同时在一个计算机上使用。
- 服务器虚拟机目的是把一个物理计算机虚拟化为多个虚拟机，使得多用户、多操作系统在一个物理计算机上并存。

Java 虚拟机-程序的跨平台

- Java VM、Oracle Hotspot VM、IBM V9 VM、Zing VM、CLDC-HI和应用在手持设备上的Dalvik VM
- JVM是Java语言的解释器，是可运行Java代码的假想计算机。
- Java虚拟机有自己硬体架构，如处理器、堆栈、寄存器等
- 具有相应的指令系统
- JVM屏蔽了与具体操作系统平台相关的信息，使得Java程序只需生成在Java虚拟机上运行的目标代码就可以在多种平台上不加修改地运行




Java的各种集成开发工具Eclipse、NetBeans，IntelliJ IDEA等
JDK开发工具包 javac、jar等
JRE运行环境
JVM
OS操作系统

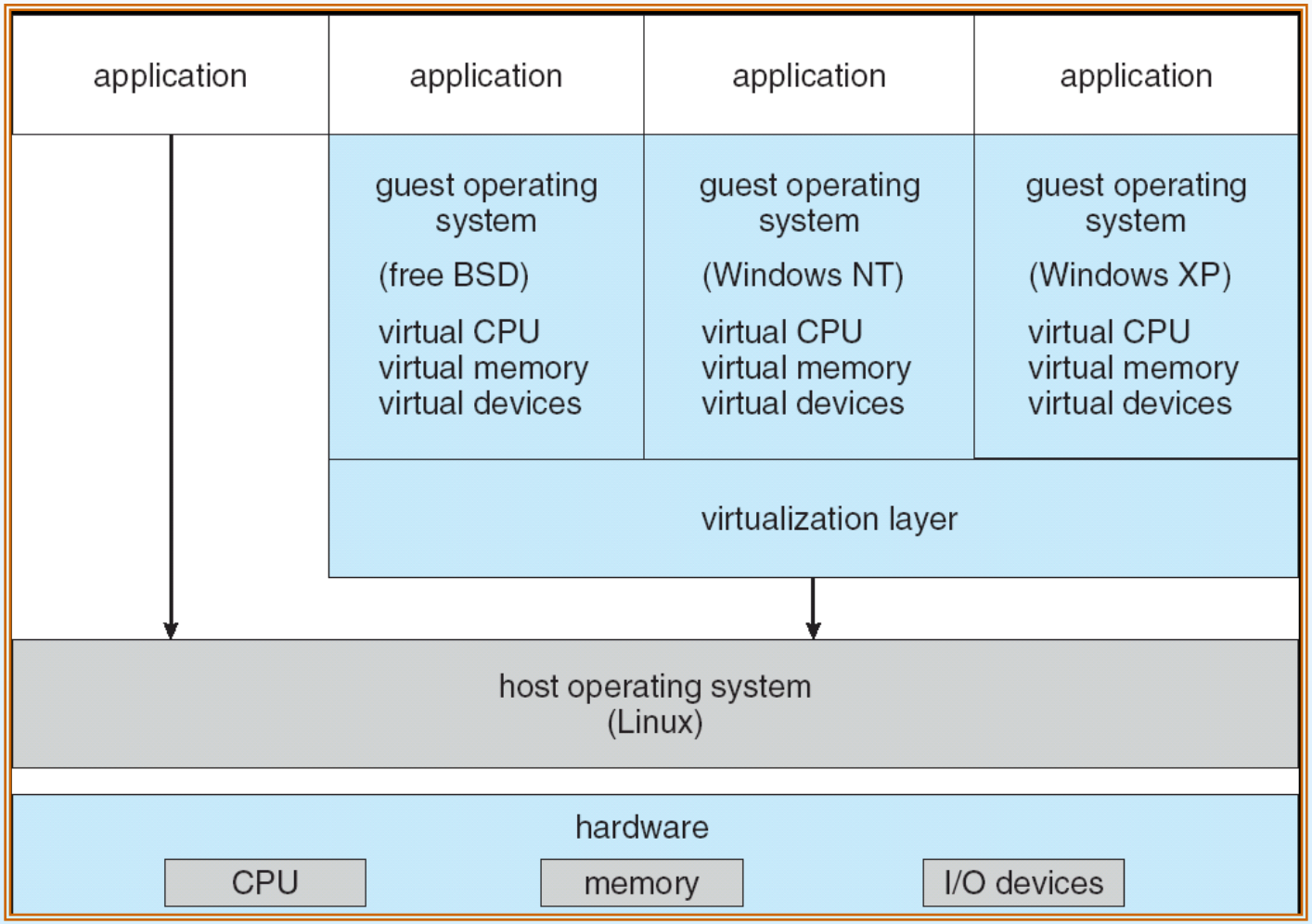
工作站虚拟机

- 2个概念
  - 安装在硬件上的操作系统称为
    - 宿主操作系统（Host OS）
  - 安装在操作系统上的操作系统称为
    - 客户操作系统（Guest OS）

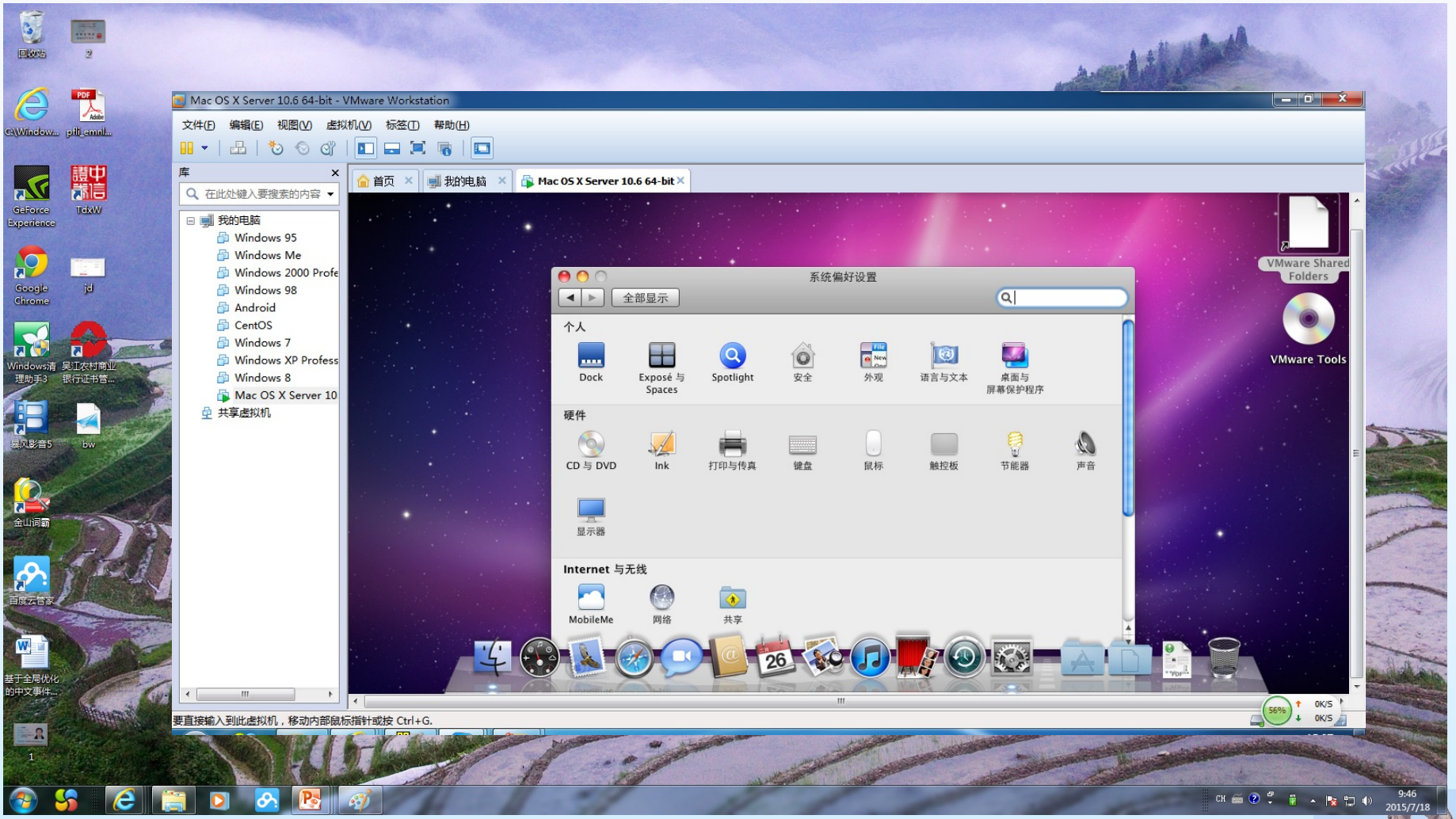
工作站虚拟机

- 工作站虚拟机作为一个软件安装在宿主操作系统中，在工作站虚拟机中就可以安装客户操作系统
- 优点：可以同时在一个计算机上使用多个操作系统，包括一个宿主操作系统和若干个客户操作系统
- 常用的工作站虚拟机软件包括：
  - VMWare Station、Virtual Box、Virtual PC和Parallels Desktop等。

VMware Station

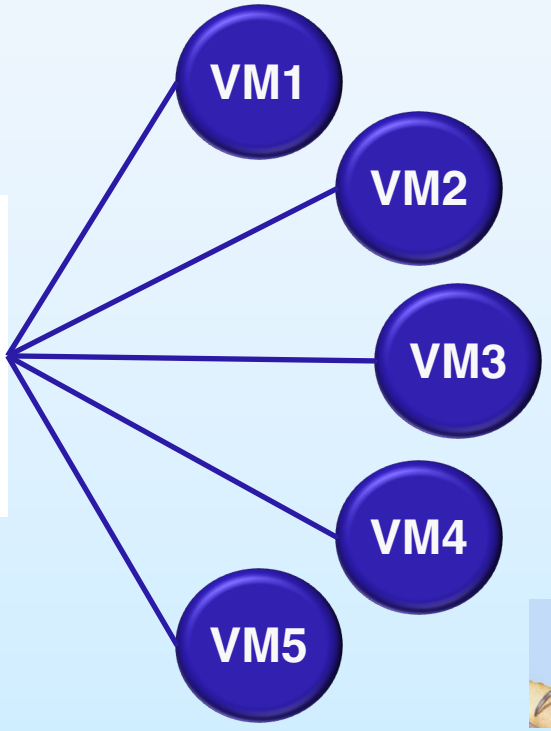


使用例子


Mac OS X Server 10.6 64-bit - VMware Workstation
文件(F) 编辑(E) 视图(V) 虚拟机(V) 标签(T) 帮助(H)
库
在此处键入要搜索的内容
我的电脑
Windows 95
Windows Me
Windows 2000 Profe
Windows 98
Android
CentOS
Windows 7
Windows XP Profess
Windows 8
Mac OS X Server 10
共享虚拟机
首页
我的电脑
Mac OS X Server 10.6 64-bit
系统偏好设置
全部显示
个人
Dock
Exposé 与 Spaces
Spotlight
安全
外观
语言与文本
桌面与屏幕保护程序
硬件
CD 与 DVD
Ink
打印与传真
键盘
鼠标
触控板
节能器
声音
显示器
Internet 与无线
MobileMe
网络
共享
VMware Shared Folders
VMware Tools
要直接输入到此虚拟机，移动内部鼠标指针或按 Ctrl+G.
56%
0K/S
0K/S
9:46
2015/7/18

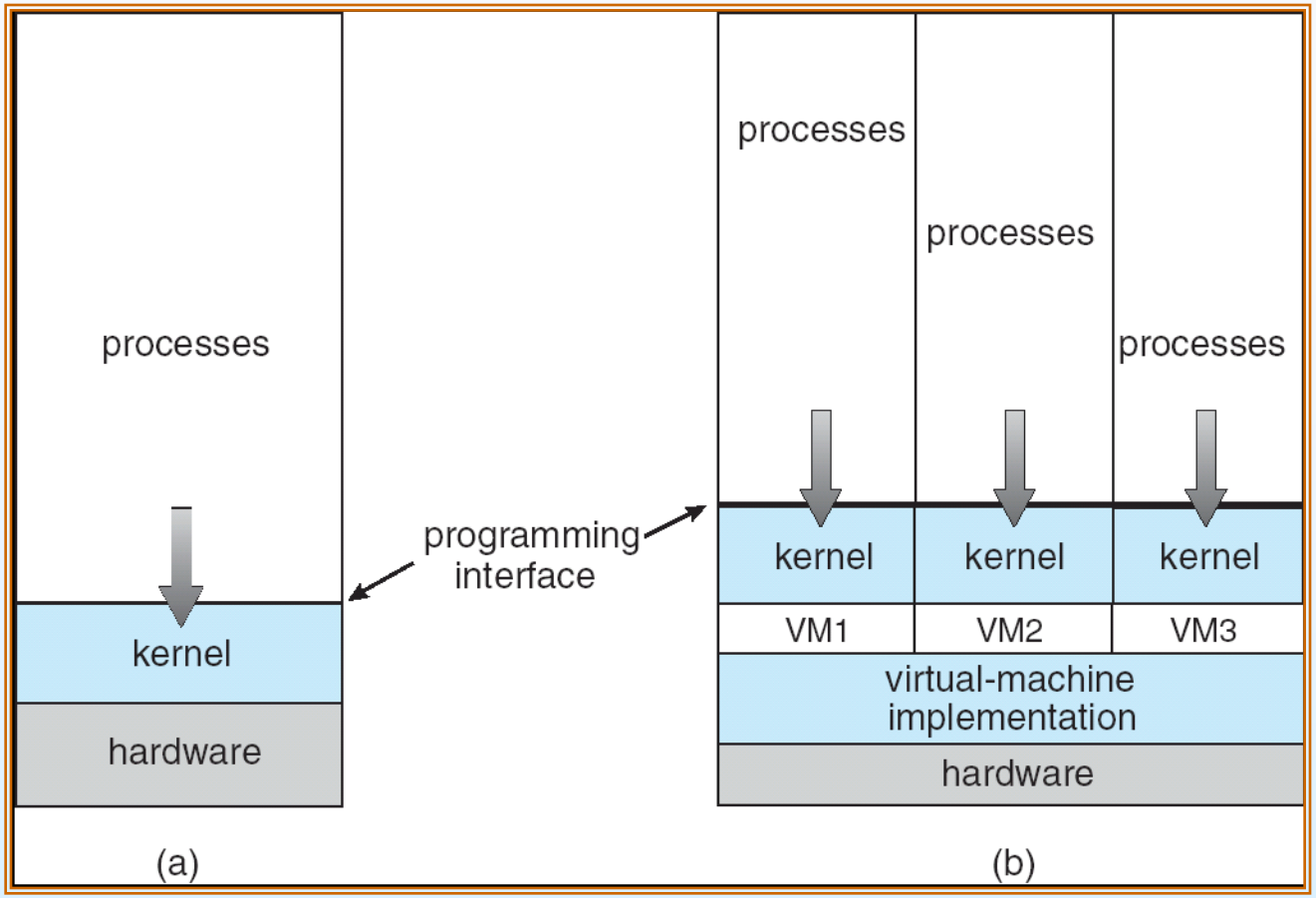
服务器虚拟机

- 将服务器物理资源抽象成逻辑资源，让一台服务器变成几台甚至上百台相互隔离的服务器虚拟机。
- 常用模式：
  - 一虚多：将一台服务器虚拟成多台服务器虚拟机
  - 多虚一：将多个独立的物理服务器虚拟为一个服务器虚拟机
- 优点
  - 安全性好
  - 资源共享
  - 可扩展性好
  - 便于隔离
  - 性价比高



虚拟机

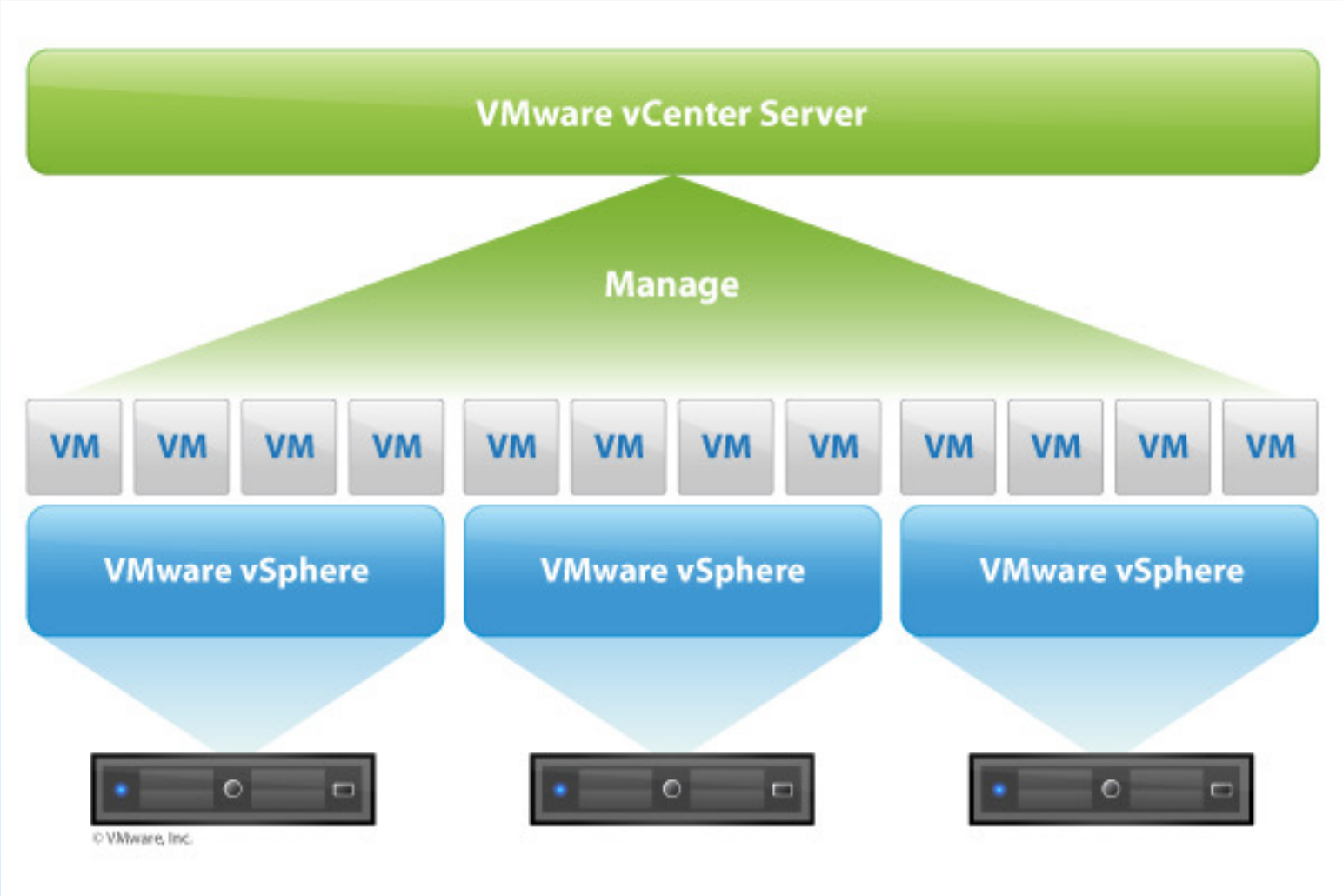
传统的服务器



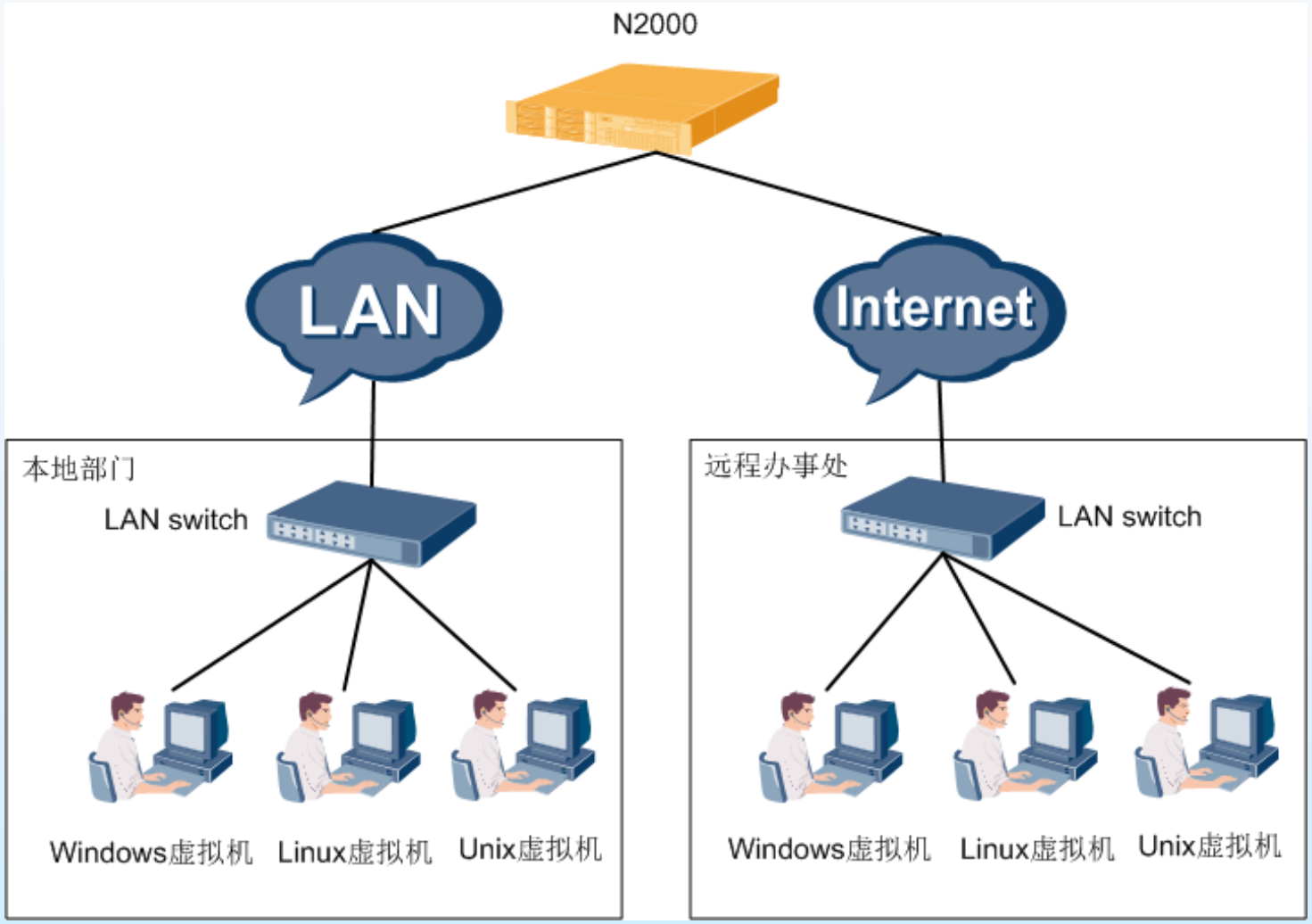
服务器的虚拟机形式

在硬件上安装虚拟化的软件

Vmware vSphere



使用例子


N2000
LAN
Internet
本地部门
LAN switch
Windows虚拟机
Linux虚拟机
Unix虚拟机
远程办事处
LAN switch
Windows虚拟机
Linux虚拟机
Unix虚拟机

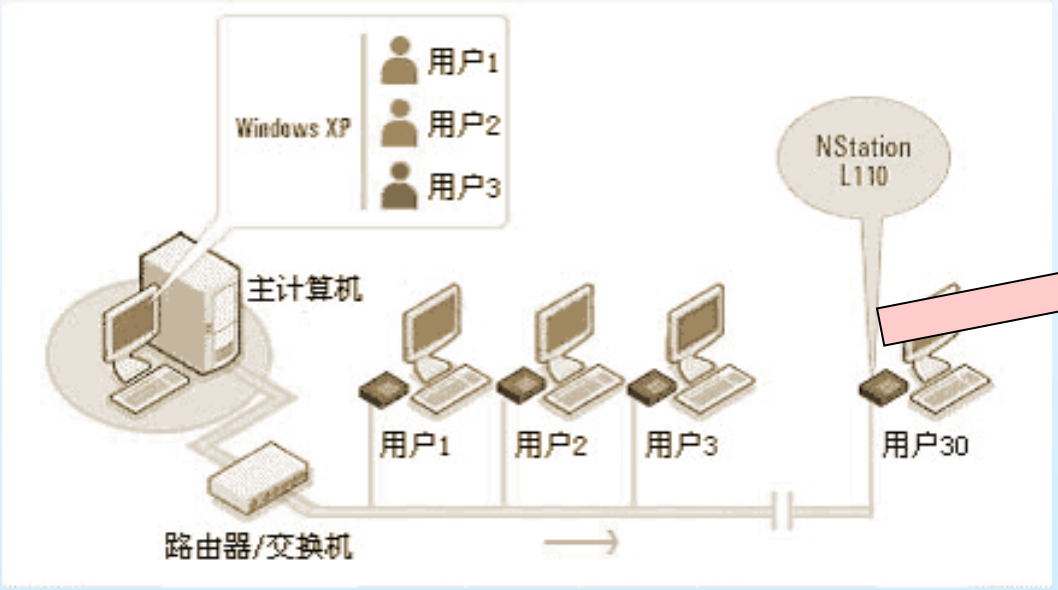
使用例子

主机系统: 172.30.5.232

虚拟机	策略	配置的 vCPU	建议的 vCPU	建议的 CPU 需求 (%)	配置的内存	推荐的内存
360Fix_5.18	Default Policy	1 个 vCPU	1 个 vCPU	3.8%	0.5 GB	224 MB
AD-WS08R2-172.30.5.15	Default Policy	1 个 vCPU	1 个 vCPU	2%	2 GB	384 MB
gcserver-172.30.5.3	Default Policy	2 个 vCPU	1 个 vCPU	2.3%	2 GB	256 MB
KMS-WinSer-172.30.5.22	Default Policy	1 个 vCPU	1 个 vCPU	7.7%	2 GB	320 MB
ln-ser-172.30.5.25	Default Policy	2 个 vCPU	1 个 vCPU	21%	2 GB	576 MB
UI VM	Default Policy	2 个 vCPU	1 个 vCPU	7.4%	4 GB	2,496 MB
View-Composer-172.30.5.52	Default Policy	2 个 vCPU	1 个 vCPU	1.4%	2 GB	224 MB
VM-XP-01	Default Policy	1 个 vCPU	1 个 vCPU	1.2%	0.5 GB	160 MB
XP-172.30.5.26	Default Policy	1 个 vCPU	1 个 vCPU	6.1%	1 GB	288 MB
xzfwzxsp-172.18.133.16	Default Policy	2 个 vCPU	1 个 vCPU	5.2%	2 GB	256 MB
zs.gc.gov.cn-172.30.5.31	Default Policy	1 个 vCPU	1 个 vCPU	0.73%	1 GB	160 MB

用户如何使用虚拟机

- 两种模式
  - 软件模式：在计算机中安装一个远程登录的软件，如Windows的远程桌面
  - 硬件模式：一种轻量级的连接模式，用户不需要PC，只要有一个手机大小的终端设备。



- JVM是一种
  - A. 服务器虚拟机
  - B. 工作站虚拟机
  - C. 高级语言虚拟机
  - D. 以上都不是

- 安装在硬件上的虚拟机软件是
  - A. 服务器虚拟机
  - B. 工作站虚拟机
  - C. 高级语言虚拟机
  - D. 以上都不是

5、操作系统设计和实现

操作系统设计目标

- OS的设计和实现没有完整的解决方案
- 不同类型操作系统的内部结构不同
- 从定义系统的**目标**和**规格**开始
- 系统设计受到硬件选择和系统类型的影响
- 用户目标和系统目标
  - 用户目标 – 系统方便和容易使用、易学、可靠、安全和快速
  - 开发目标 – 系统容易设计、实现和维护，也应该灵活、可靠、高效且没有错误

策略和机制分离

- 区分的重要原则

  策略(Policy): 做什么?
  机制(Mechanism): 怎么做?

- 机制决定如何做, 策略决定做什么
  - 策略与机制的区分对于**灵活性**来说很重要
  - 策略可能会随时间或位置而有所改变

实现

- 开发语言
  - 早期：汇编语言
  - 目前：高级语言（ C, C++）
- 实际：混合多种语言
  - 底层用汇编语言
  - 主体用C语言
  - 系统程序用C, C++, PERL, Python, shell scripts等
- 考量
  - 汇编语言：运行高效，但编程耗时，不易移植
  - 高级语言：运行效率差，但编程高效，易移植