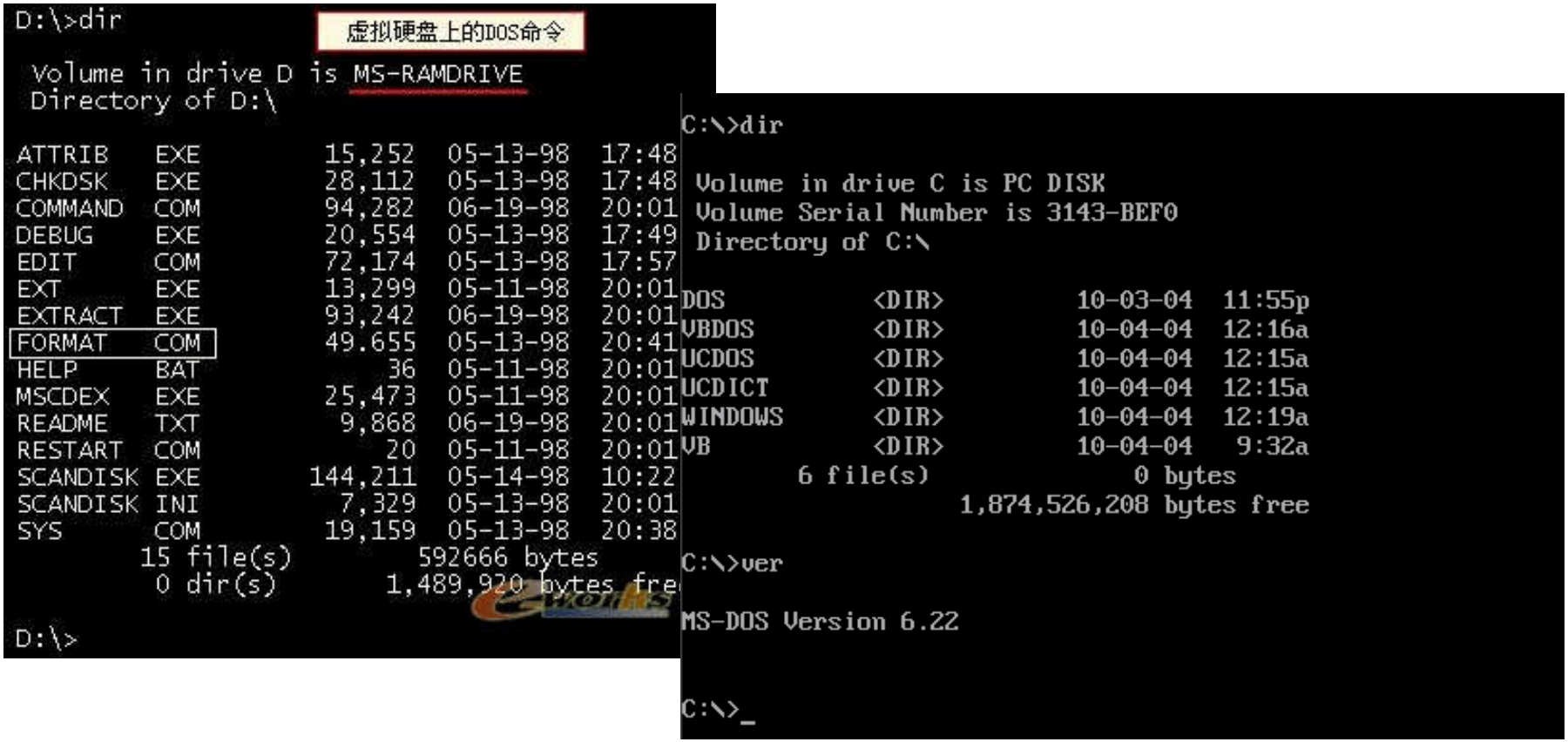
十一、Windows系统

Windows历史：MS-DOS

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 1981.8.12
- Microsoft在Windows之前制造的操作系统
  - MS-DOS是Microsoft Disk Operating System的简称，意即由美国微软公司(Microsoft)提供的磁盘操作系统。在Windows 95以前，DOS是PC兼容电脑的最基本配备，而MS-DOS则是最普遍使用的PC兼容DOS
  - 最基本的MS-DOS系统由一个基于MBR的BOOT引导程序和三个文件模块组成。这三个模块是输入输出模块(IO.SYS)、文件管理模块(MSDOS.SYS)及命令解释模块(COMMAND.COM)。除此之外，微软还在零售的MS-DOS系统包中加入了若干标准的外部程序(即外部命令)，这才与内部命令(即由COMMAND.COM解释执行的命令)一同构建起一个在磁盘操作时代相对完备的人机交互环境

Windows历史：MS-DOS

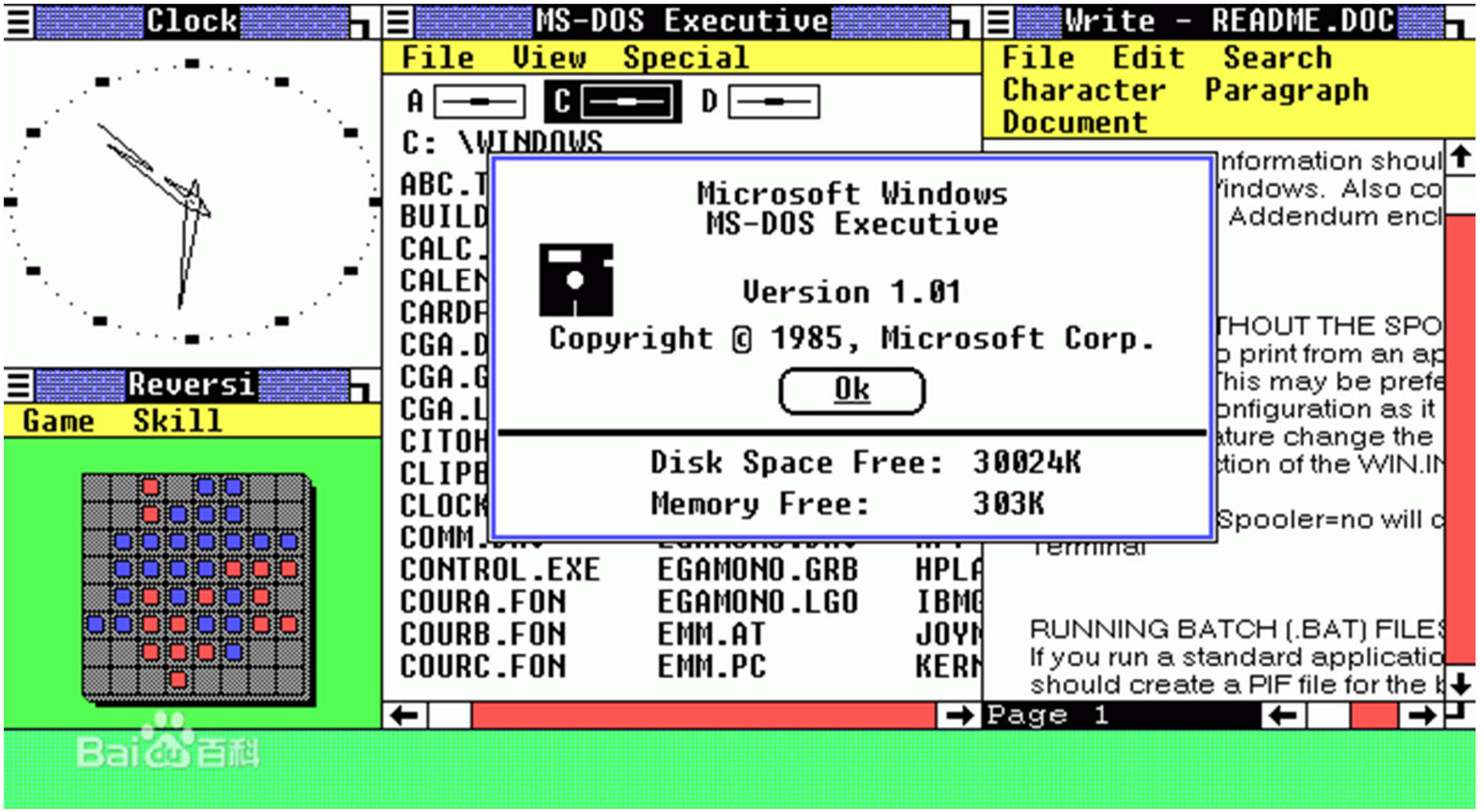


Windows历史：Windows 1.0

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 1985.11.20
- Windows 1.0基于MS-DOS操作系统。Microsoft Windows 1.0是Windows系列的第一个产品
- Windows 1.0中鼠标作用得到特别的重视，用户可以通过点击鼠标完成大部分的操作
- Windows 1.0 自带了一些简单的应用程序，包括日历、记事本、计算器等等
- Windows 1.0 可以显示256种颜色，窗口可以任意缩放，当窗口最小化的时候桌面上会有专门的空间放置这些窗口（其实就是现在的任务栏）
- 在Windows 1.0中已经出现了控制面板（Control Panel），对驱动程序、虚拟内存有了明确的定义，不过功能非常有限

Windows历史：Windows 1.0

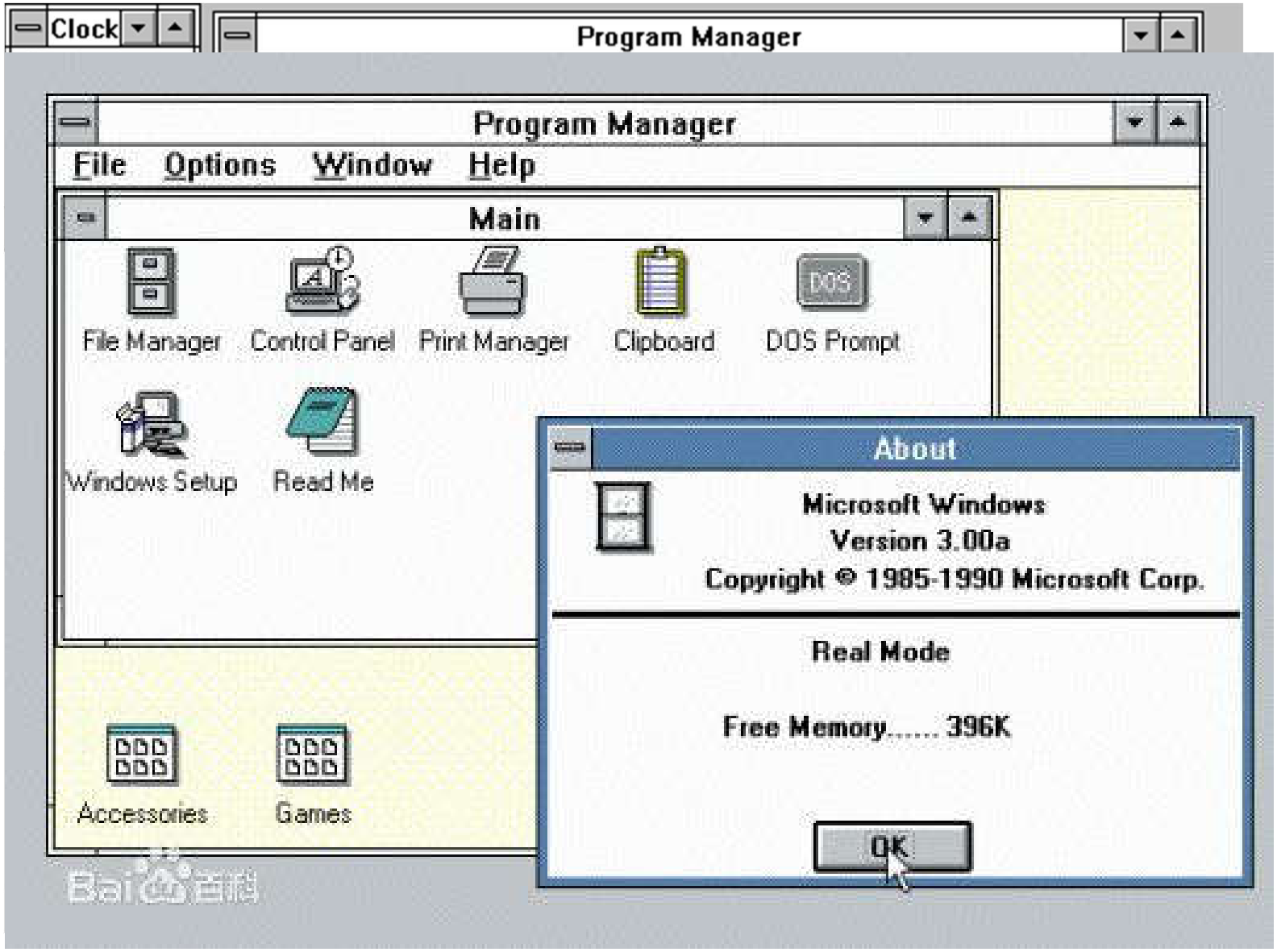


Windows历史：Windows 3

- 1990.5.22
- Windows 3.0具备了模拟32位操作系统的功能，图片显示效果大有长进，对当时最先进的386处理器有良好的支持。这个系统还提供了对虚拟设备驱动（VxDs）的支持，极大改善了系统的可扩展性，计算机用户再不必在购买Windows3.0时煞费苦心地查证自己的硬件是否可以被系统支持了，因为他完全可以另外安装一个驱动程序
- Windows 3.0使用了一组新的图标，这让他的面貌得到很大改观，再也不是一幅灰头土脸的样子了。不过这并不是微软的独创，而是模仿了苹果公司Macintosh的设计。直到今天苹果电脑（OS X）的图标设计仍然是计算机中的上上之品


Microsoft®
Windows™
Version 3.0
Copyright © Microsoft Corporation 1985-1990. All Rights Reserved.
Baidu百科

Windows历史：Windows 3



Windows历史：Windows 95

- 1995.8.24
- Windows 95是一个混合的16位/32位Windows系统，其内核版本号为NT4.0
- Windows 95是微软之前独立的操作系统MS-DOS和Windows产品的直接后续版本
- 第一次抛弃了对前一代16位的支持，因此它要求英特尔公司的80386处理器或者在保护模式下运行于一个兼容的速度更快的处理器。它以对GUI的重要的改进和底层工作（underlying workings）为特征。同时也是第一个特别捆绑了一个版本的DOS的Windows版本（Microsoft DOS 6.22）

Windows历史：Windows 95

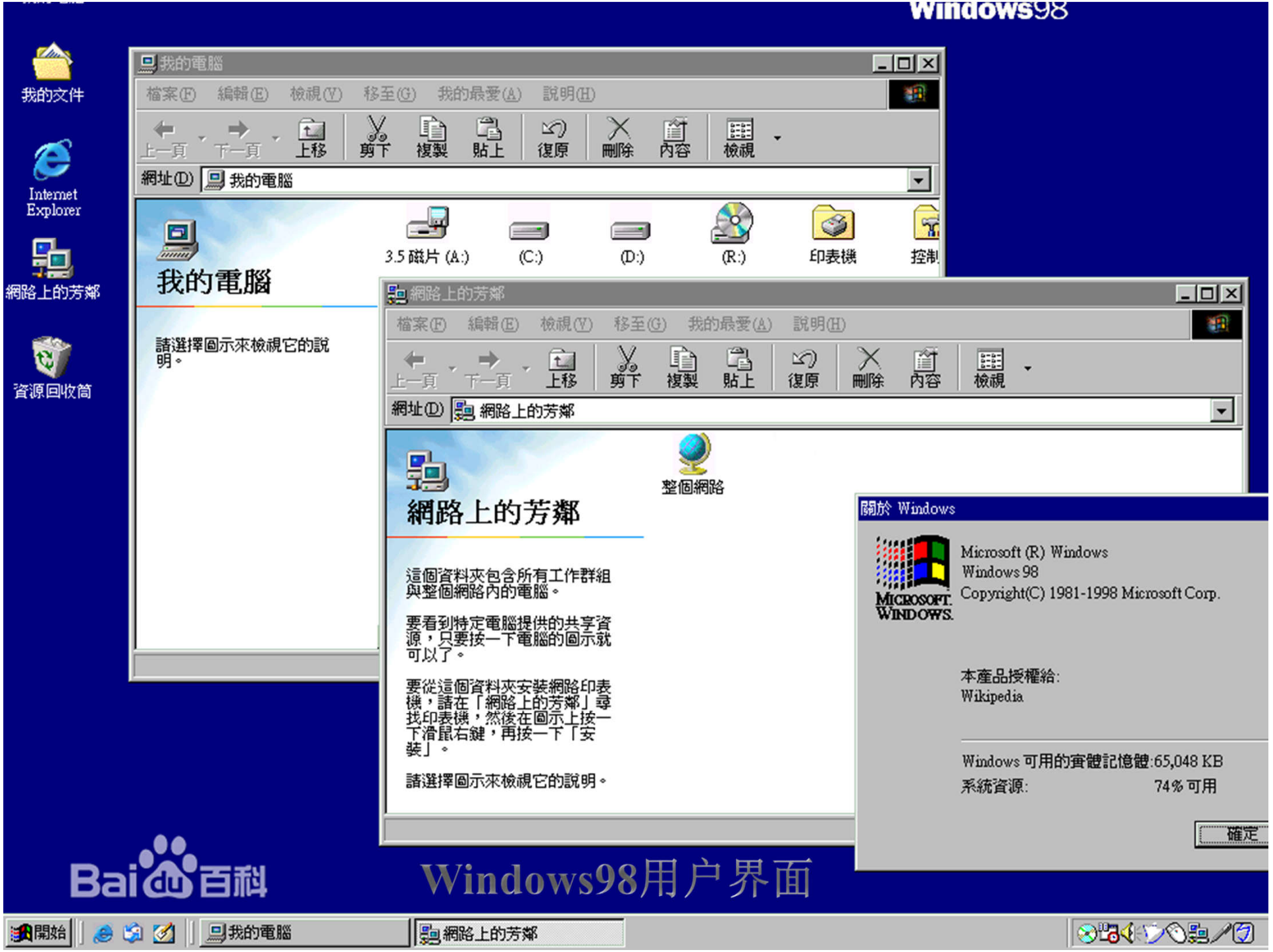
Windows历史：Windows 95

Windows历史：Windows 98

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 1998.6.25
- Windows 98是一个发行于1998年6月25日的混合16位/32位的Windows系统，其内核版本号为4.1，开发代号为Memphis。这个新的系统是基于Windows 95上编写的，它改良了硬件标准的支持，例如MMX和AGP。其它特性包括对FAT32文件系统的支持、多显示器、WebTV的支持和整合到Windows图形用户界面的Internet Explorer，称为活动桌面(Active Desktop)
- Windows 98的最大特点就是把微软的IE浏览器技术整合到了Windows里面，从而更好地满足了用户访问Internet资源的需要

Windows历史：Windows 98


Windows98
我的文件
Internet Explorer
網路上的芳鄰
資源回收筒
我的電腦
檔案(F) 編輯(E) 檢視(V) 移至(G) 我的最愛(A) 說明(H)
上一頁 下一頁 上移 剪下 複製 貼上 復原 刪除 內容 檢視
網址(D) 我的電腦
我的電腦
請選擇圖示來檢視它的說明。
3.5 磁片 (A:)
(C:)
(D:)
(R:)
印表機
網路上的芳鄰
網址(D) 網路上的芳鄰
整個網路
網路上的芳鄰
這個資料夾包含所有工作群組與整個網路內的電腦。
要看到特定電腦提供的共享資源，只要按一下電腦的圖示就可以了。
要從這個資料夾安裝網路印表機，請在「網路上的芳鄰」尋找印表機，然後在圖示上按一下滑鼠右鍵，再按一下「安裝」。
請選擇圖示來檢視它的說明。
關於 Windows
Microsoft (R) Windows
Windows 98
Copyright(C) 1981-1998 Microsoft Corp.
本產品授權給:
Wikipedia
Windows 可用的實體記憶體:65,048 KB
系統資源: 74% 可用
確定
Baidu百科
Windows98用户界面
開始
我的電腦
網路上的芳鄰

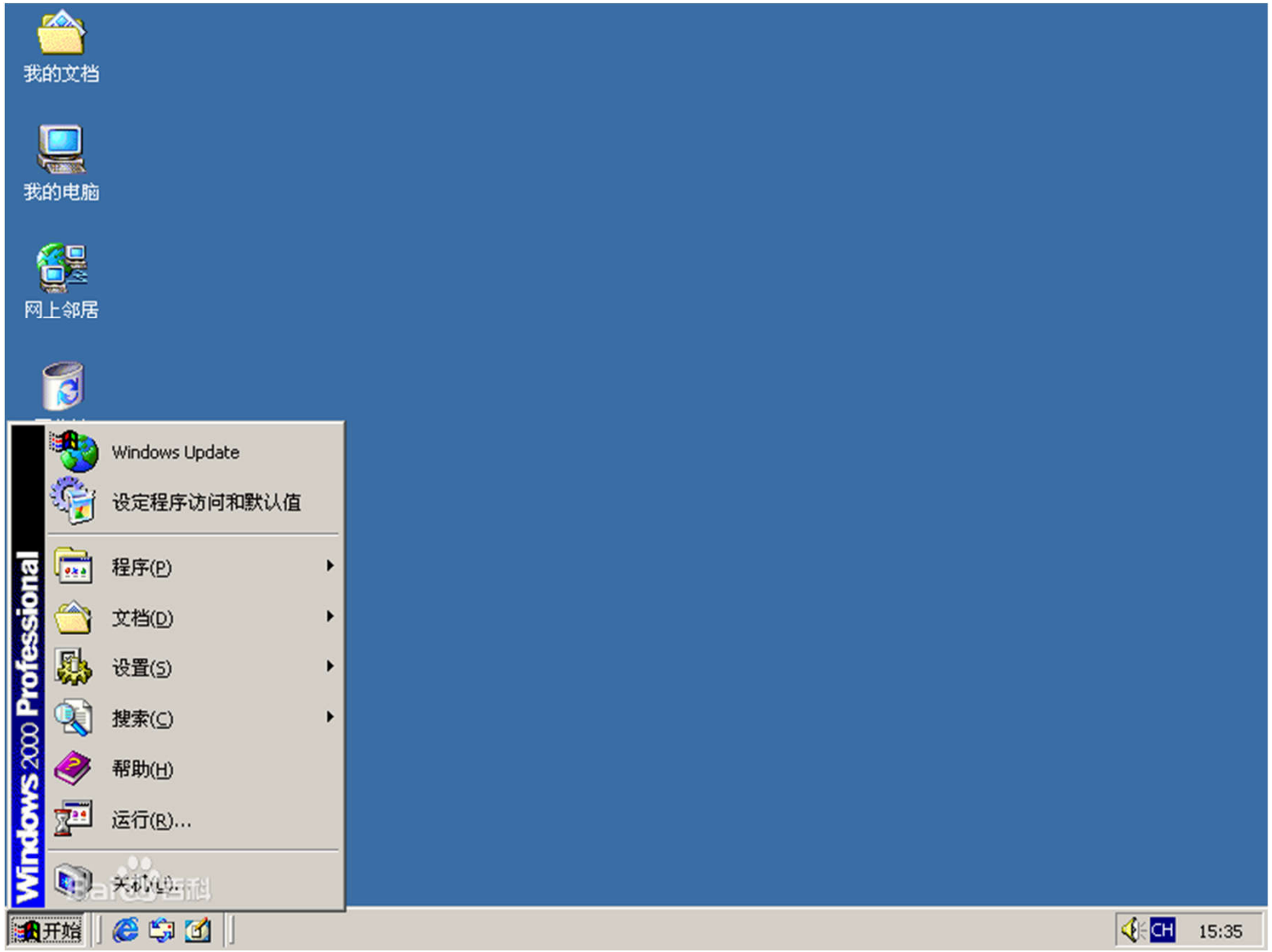
Windows历史：Windows 2000

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 2000.2.17
- Microsoft Windows 2000（中文：微软视窗操作系统2000，简称Win2K），是由微软公司发行的Windows NT系列的32位视窗操作系统。起初被称为Windows NT 5.0。英文版于2000年2月17日在美国旧金山正式发布，中文版于同年3月20日在中国北京正式推出。Windows 2000是一个可中断的、图形化的面向商业环境的操作系统，为单一处理器或对称多处理器的32位Intel x86电脑而设计

Windows历史：Windows 2000

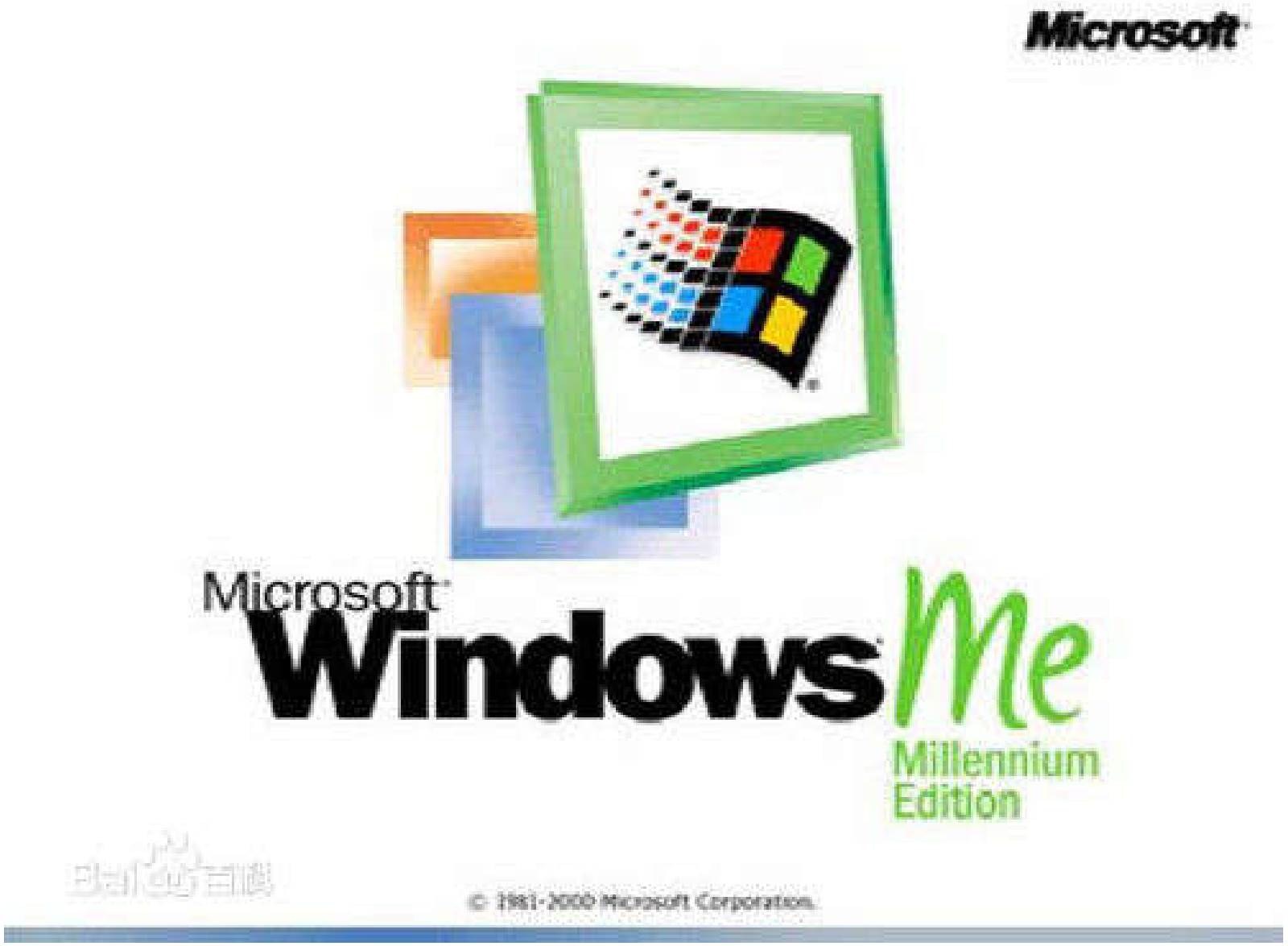
Windows历史：Windows 2000



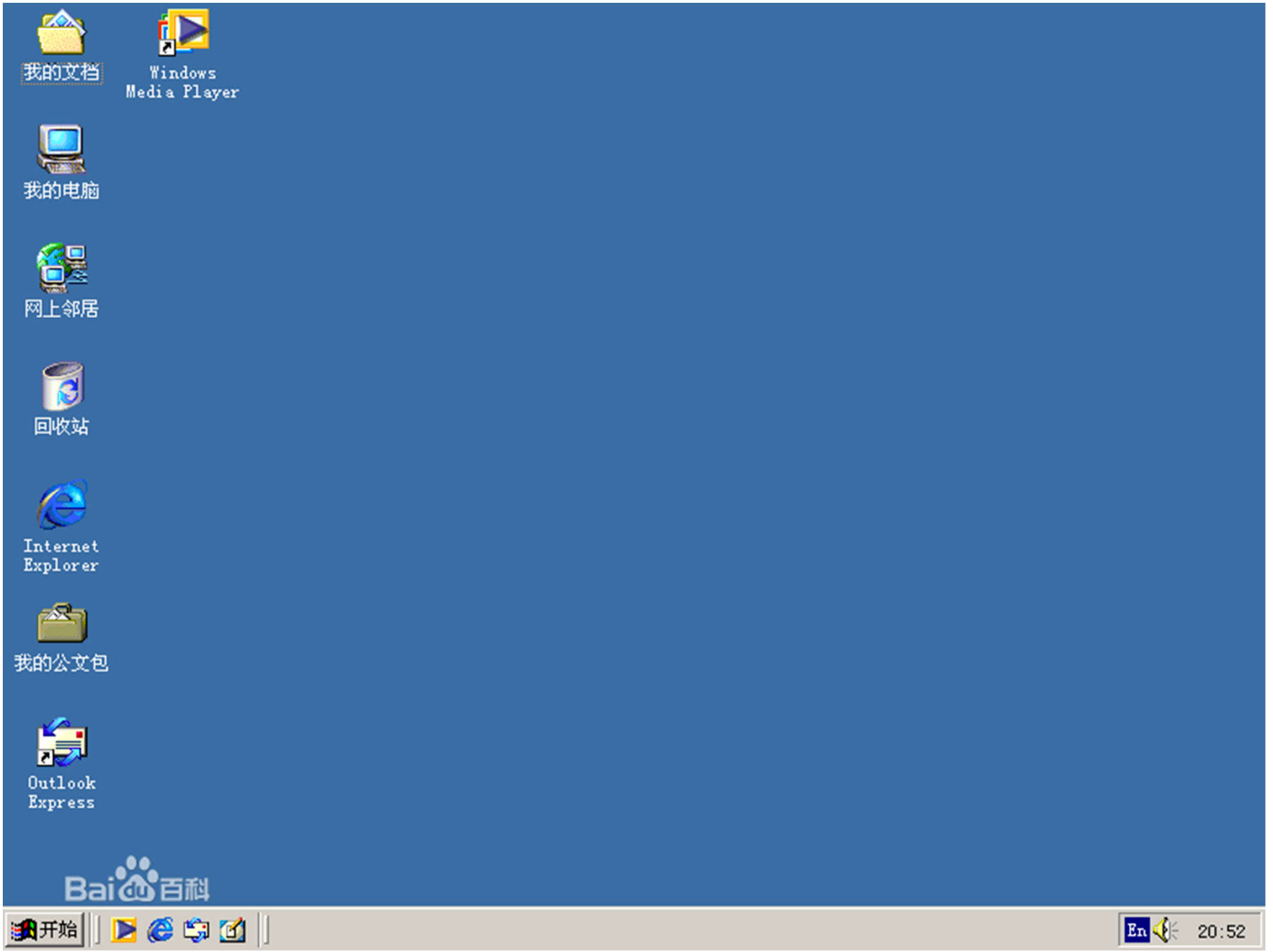
Windows历史：Windows ME

- 2000.9.14
- Windows Me（Windows Millennium Edition）是16位/32位混合的Windows操作系统，由微软公司在2000年9月14日发行。Windows Me同时也是最后一款基于MS-DOS的Windows 9X内核系列的Windows操作系统，正式版本号是：4.9.3000
- Windows Me是在Windows 95和Windows 98的基础上开发的，并具有与Windows 2000相同的界面，而系统内核无大的改进

Windows历史：Windows ME



Windows历史：Windows ME



Windows历史：Windows XP

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 2001.8.25
- Windows XP是美国微软公司发布的一款Windows操作系统，内核版本号为NT 5.1。开发代号为WindowsWhistler
- 微软最初发行了两个版本，家庭版（Home）和专业版（Professional）。家庭版的消费对象是家庭用户，专业版则在家庭版的基础上添加了新的为面向商业的设计的网络认证、双处理器等特性。且家庭版只支持1个处理器，专业版则支持2个。字母“XP”表示英文单词的“体验”（experience）
- Windows XP是基于Windows 2000代码的产品，同时拥有一个新的用户图形界面（叫做月神Luna），它包括了一些细微的修改，其中一些看起来是从Linux的桌面环境（desktop environment）诸如KDE中获得的灵感。带有用户图形的登陆界面就是一个例子

Windows历史：Windows XP

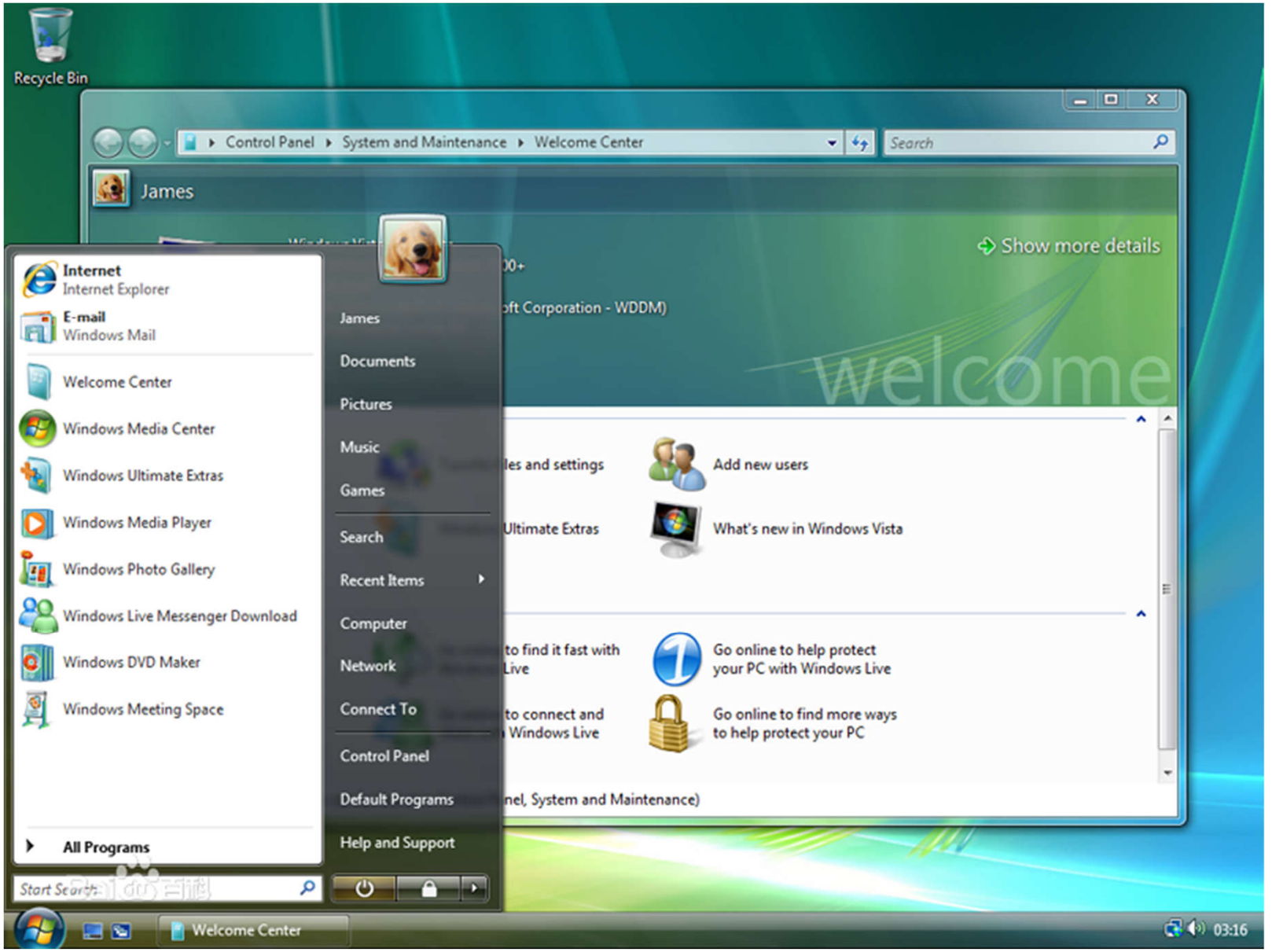
Windows历史：Windows XP

Windows历史： Windows Vista

- 2007.1.30
- Windows Vista较上一个版本Windows XP增加了上百种新功能，其中包括被称为“Aero”的全新图形用户界面、加强后的搜索功能（Windows Indexing Service）、新的媒体创作工具（例如Windows DVD Maker）以及重新设计的网络、音频、输出（打印）和显示子系统。Vista也使用点对点技术（Peer-to-peer）提升了计算机系统在家庭网络中的显示通信能力，将让在不同计算机或装置之间分享文件与多媒体内容变得更简单。针对开发者的方面，Windows Vista使用.NET Framework 3.0版本，比起传统的Windows API更能让开发者能简单写出高品质的程序。微软也在Vista的安全性方面进行改良，Vista较Windows XP增加了用户管理机制（UAC）以及内置的恶意软件查杀工具（Windows Defender）等

Windows历史： Windows Vista

Windows历史： Windows Vista



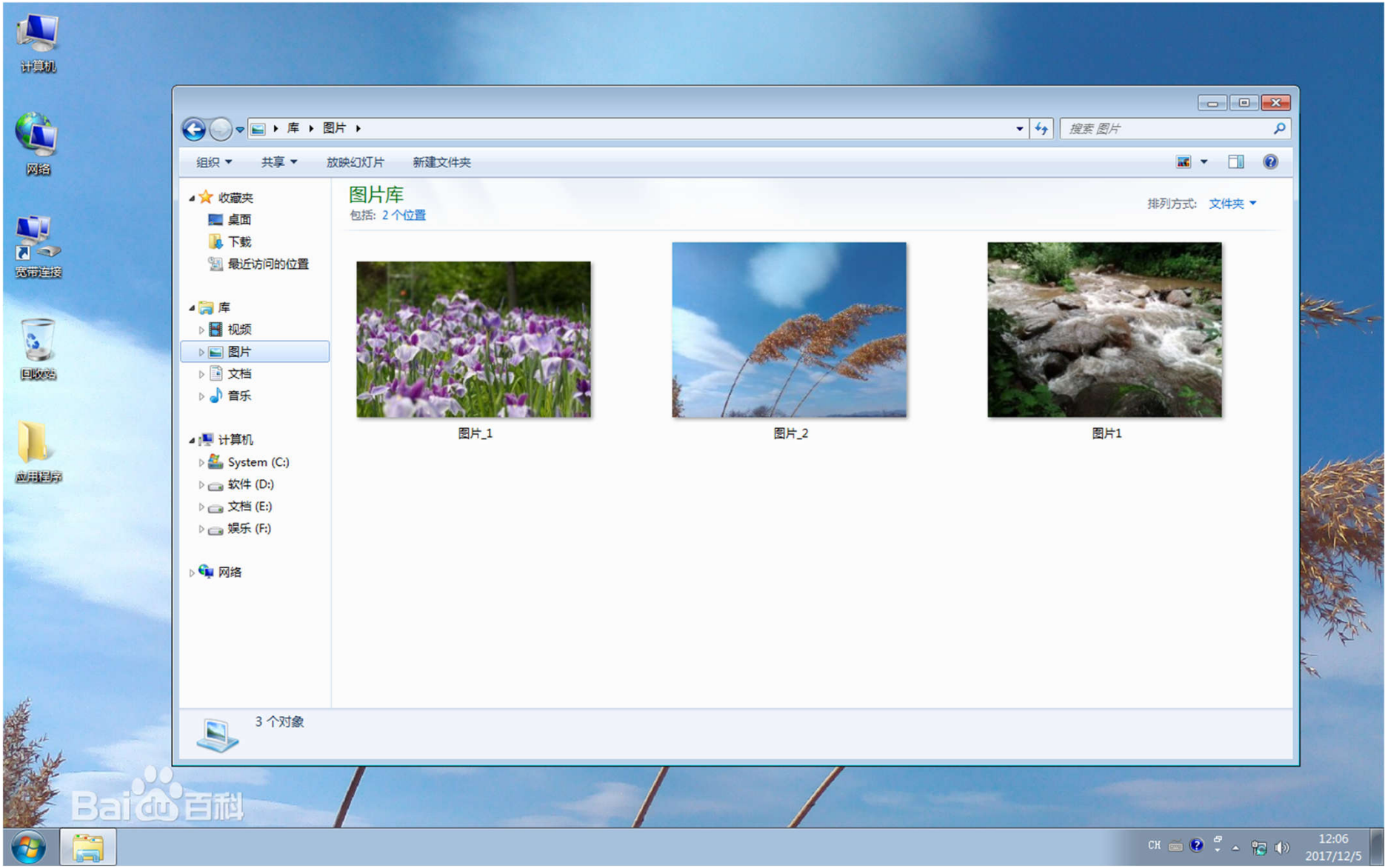
Windows历史： Windows 7

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 2009.10.22
- Windows 7是微软于2009年发布的，开始支持触控技术的Windows桌面操作系统，其内核版本号为NT6.1。在Windows 7中，集成了DirectX 11和Internet Explorer 8。DirectX 11作为3D图形接口，不仅支持未来的DX11硬件，还向下兼容当前的DirectX 10和10.1硬件。DirectX 11增加了新的计算shader技术，可以允许GPU从事更多的通用计算工作，而不仅仅是3D运算，这可以鼓励开发人员更好地将GPU作为并行处理器使用。Windows 7还具有超级任务栏，提升了界面的美观性和多任务切换的使用体验。通过开机时间的缩短，硬盘传输速度的提高等一系列性能改进，Windows 7的系统要求并不低于Windows Vista，不过当时的硬件已经很强大了。到2012年9月，Windows 7的占有率已经超越Windows XP，成为世界上占有率最高的操作系统

Windows历史： Windows 7

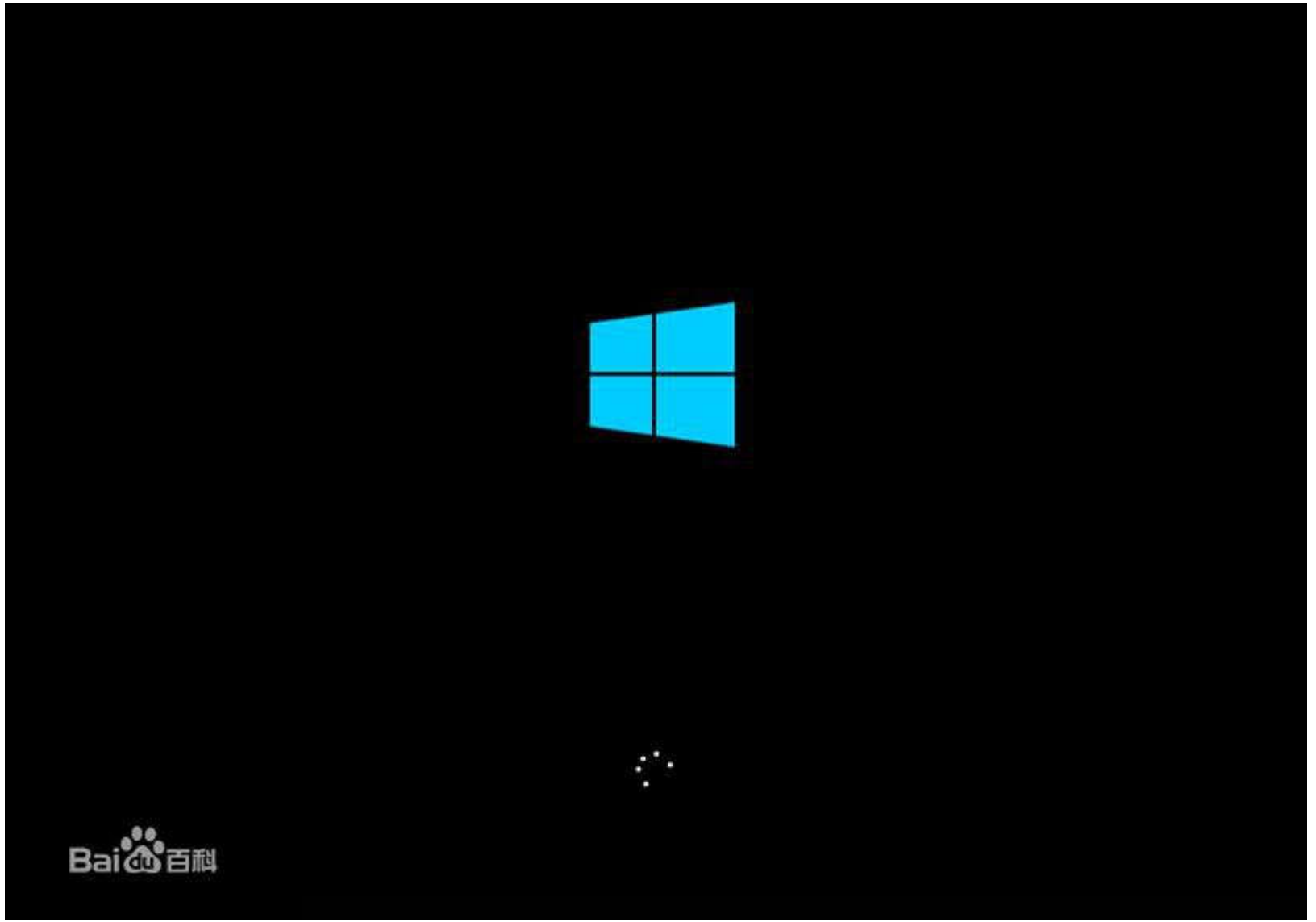
Windows历史： Windows 7



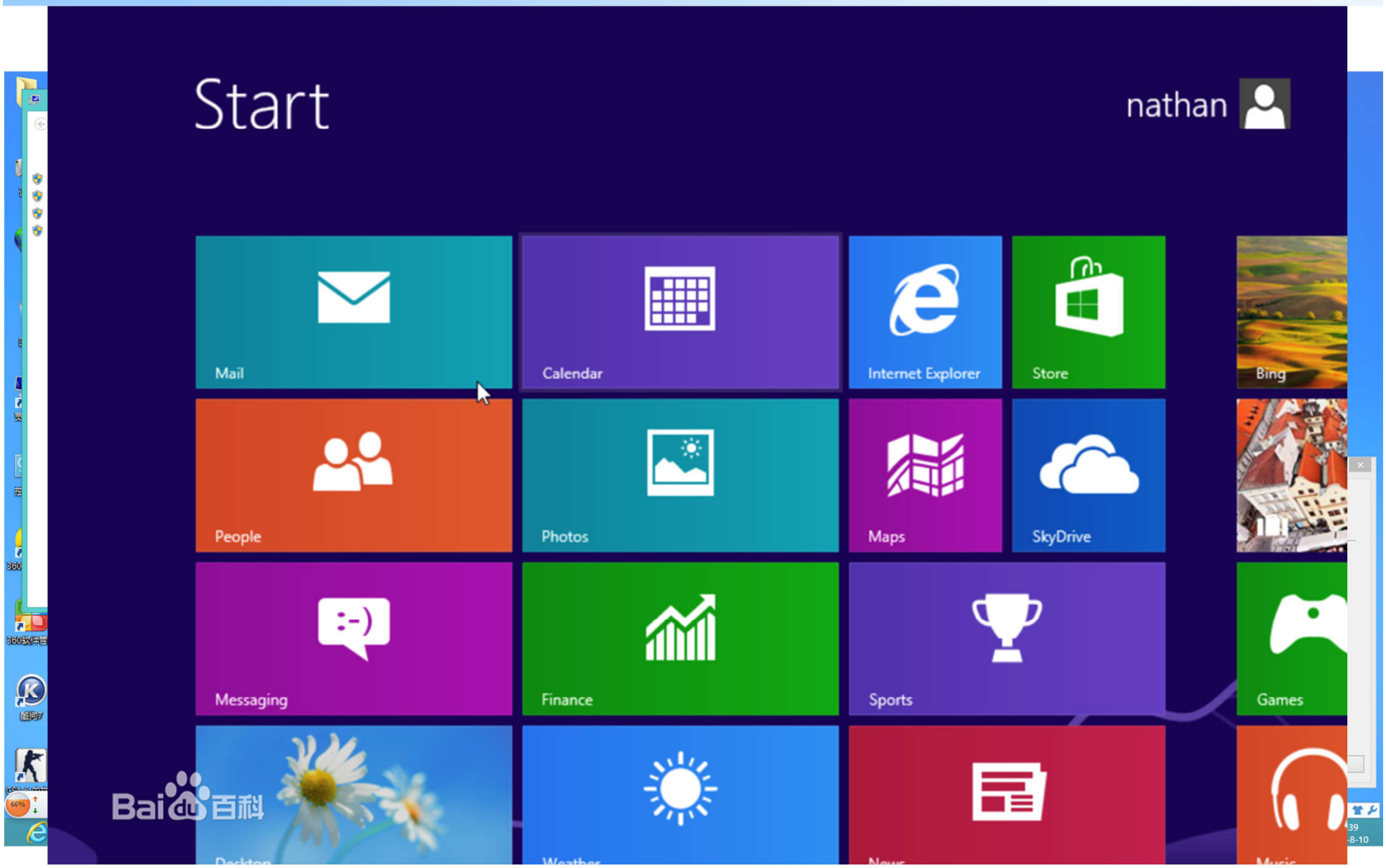
Windows历史： Windows 8

- 2012.10.26
- Windows8是由微软公司开发的，是第一款带有Metro界面的桌面操作系统，内核版本号为NT6.2。该系统旨在让人们的日常的平板电脑操作更加简单和快捷，为人们提供高效易行的工作环境Windows 8支持来自Intel、AMD和ARM的芯片架构。Windows Phone 8采用和Windows 8相同的NT内核。2011年9月14日，Windows 8开发者预览版发布，宣布兼容移动终端，微软将苹果的IOS、谷歌的Android视为Windows 8在移动领域的主要竞争对手。2012年8月2日，微软宣布Windows 8开发完成，正式发布RTM版本；10月25号正式推出Windows 8，微软自称触摸革命将开始

Windows历史： Windows 8



Windows历史： Windows 8



Windows历史： Windows 10

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 2015.7.29
- 2015年1月21日，微软在华盛顿发布新一代Windows系统，并表示向运行Windows7、Windows 8.1以及Windows Phone 8.1的所有设备提供，用户可以在Windows 10发布后的第一年享受免费升级服务。2月13日，微软正式开启Windows 10手机预览版更新推送计划。3月18日，微软中国官网正式推出了Windows 10中文介绍页面。4月22日，微软推出了Windows Hello和微软Passport用户认证系统，微软又公布了名为“Device Guard”（设备卫士）的安全功能。4月29日，微软宣布Windows 10将采用同一个应用商店，即可展示给Windows 10覆盖的所有设备用，同时支持Android和iOS程序。7月29日，微软发布Windows 10正式版

Windows历史： Windows 10

Windows历史： Windows 10

哈爾濱工業大學 HARBIN INSTITUTE OF TECHNOLOGY

Windows 内核基础

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

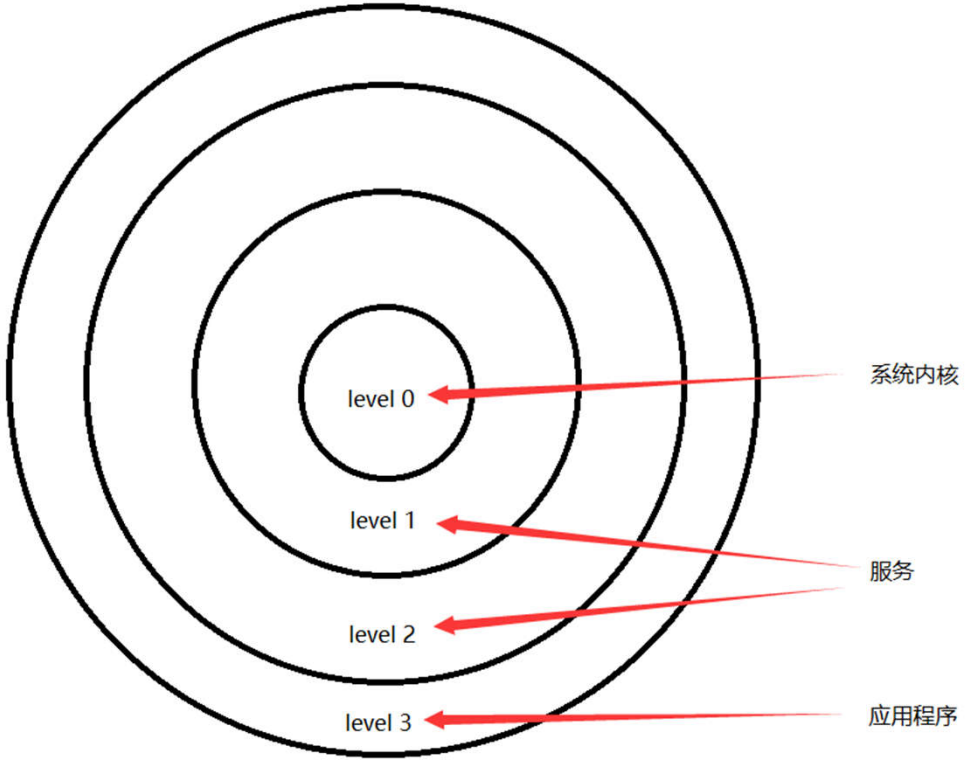
- 现代操作系统一般分为应用层和内核层两部分
- 应用层进程通过系统调用进入内核，由系统底层完成相应的功能
- 一是指系统内核本身，二是指第三方软件以内核模块方式加载的驱动文件

权限级别

- 系统内核层：又叫零环（Ring0，简称“R0”），是CPU的4个运行级别中的一个
- CPU设计者将CPU的运行级别从内向外分为4个，依次为R0、R1、R2、R3，运行权限从R0到R3依次降低
- R0拥有最高执行权限，R3拥有最低执行权限

权限级别

- CPU设计制造商在设计之初是让R0运行内核，让R1、R2运行设备驱动，让R3运行应用程序



权限级别

- 访问者和受访者
- 访问者是动态的，其主动去访问各种资源，其特权是动态变化的
- 受访者是静态的，他就是被访问的资源，其特权应该是保持不变的

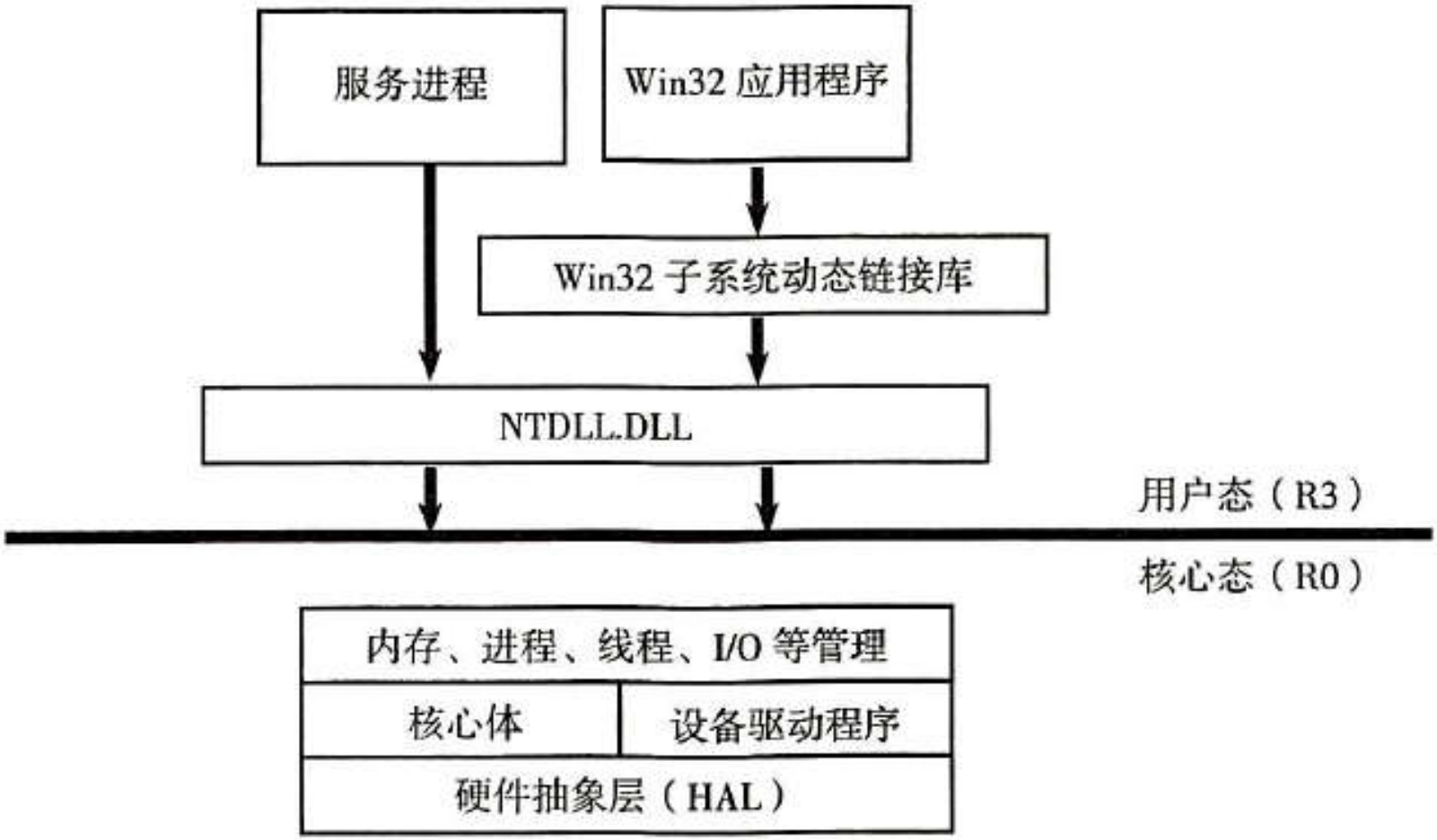
权限级别

- 建立特权机制是为了通过特权来检查合法性，即主要发生在访问者去访问受访者的时候，检查内容就是访问者的特权级和受访者的特权级是否匹配

- 操作系统中，特权级按照权力从大到小依次分为0、1、2、3级别

- 操作系统位于0级特权，可以直接控制硬件，掌控各种核心数据；系统程序分别位于1级特权或2级特权，主要是一些虚拟机、驱动程序等系统服务；而一般的应用程序运行在3级特权

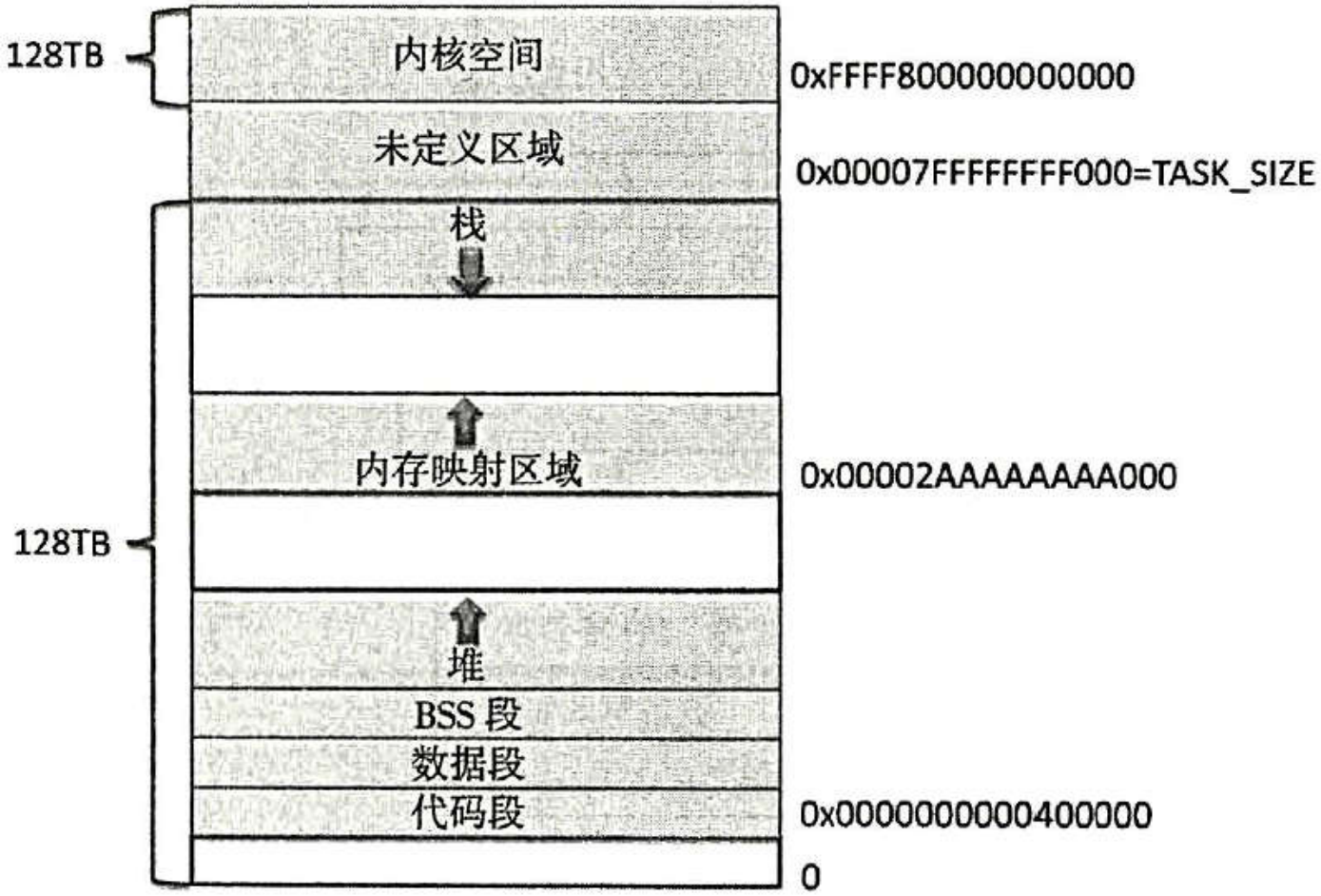
权限级别


服务进程
Win32 应用程序
Win32 子系统动态链接库
NTDLL.DLL
用户态（R3）
核心态（R0）
内存、进程、线程、I/O 等管理
核心体
设备驱动程序
硬件抽象层（HAL）

内存空间布局


0xffffffff
2GB 内核空间
0x7fffffff
64KB 非法区域
0x7fff0000
进程空间
0x0000ffff
64KB NULL 空间
0x00000000

内存空间布局


128TB
内核空间
0xFFFF800000000000
未定义区域
0x00007FFFFFFFF000=TASK_SIZE
栈
内存映射区域
0x00002AAAAAAAA000
128TB
堆
BSS 段
数据段
代码段
0x0000000000400000
0

Windows 与内核启动过程

- 1. 启动自检阶段
  - 在打开电源时，计算机开始自检过程，从BIOS中载入必要的指令，然后进行一系列的自检操作，进行硬件的初始化检查（包括内存、硬盘、键盘等），同时在屏幕上显示信息
- 2. 初始化启动阶段
  - 自检完成后，根据CMOS的设置，BIOS加载启动盘，将主引导记录（MBR）中的引导代码载入内存。接着，启动过程由MBR来执行。启动代码搜索MBR中的分区表，找出活动分区，将第1个扇区中的引导代码载入内存。引导代码检测当前使用的文件系统，查找ntldr文件，找到之后将启动它。BIOS将控制权转交给ntldr，由ntldr完成操作系统的启动工作

Windows 与内核启动过程

- 3. Boot加载阶段
  - 先从启动分区加载ntldr，然后对ntldr进行如下设置
    - 设置内存模式
    - 启动一个简单的文件系统，以定位boot.ini、ntoskrnl、Hal等启动文件
    - 读取boot.ini文件
- 4. 检测和配置硬件阶段
  - 在这个阶段会检查和配置一些硬件设备，例如系统固件、总线和适配器、显示适配器、键盘、通信端口、磁盘、软盘、输入设备（例如鼠标）、并口、ISA总线上运行的设备等

Windows 与内核启动过程

- 5. 内核加载阶段
  - ntldr 将首先加载Windows内核Ntoskrnl.exe和硬件抽象层（HAL）。HAL会对硬件底层的特性进行隔离，为操作系统提供统一的调用接口
  - ntldr从注册表的HKEY_LOCAL_MACHINE\System\CurrentControlSet键下读取这台机器安装的驱动程序，然后依次加载驱动程序。初始化底层设备驱动，在注册表的HKEY_LACAL_MACHINE\System\CurrentControlSet\Services 键下查找“Start”键的值为0和1的设备驱动

Windows 与内核启动过程

- 5. 内核加载阶段
  - “Start”键的值可以为0、1、2、3、4，数值越小，启动越早
  - SERVICE_BOOT_START（0）表示内核刚刚初始化，此时加载的都是与系统核心有关的重要驱动程序，例如磁盘驱动
  - SERVICE SYSTEM_START（1）稍晚一；SERVICE_AUTO_START（2）是从登录界面出现的时候开始，如果登录速度较快，很可能驱动还没有加载就已经登录了
  - SERVICE_DEMAND_START（3）表示在需要的时候手动加载
  - SERVICE_DISABLED（4）表示禁止加载

Windows 与内核启动过程

- 6. Windows 的会话管理启动
  - 驱动程序加载完成，内核会启动会话管理器。这是一个名为smss.exe的程序，是Windows系统中第1个创建的用户模式进程，其作用如下
    - 创建系统环境变量
    - 加载win32k.sys，它是Windows子系统的内核模式部分
    - 启动csrss.exe，它是Windows子系统的用户模式部分
    - 启动winlogon.exe
    - 创建虚拟内存页面文件
    - 执行上次系统重启前未完成的重命名工作（PendingFileRename）

Windows 与内核启动过程

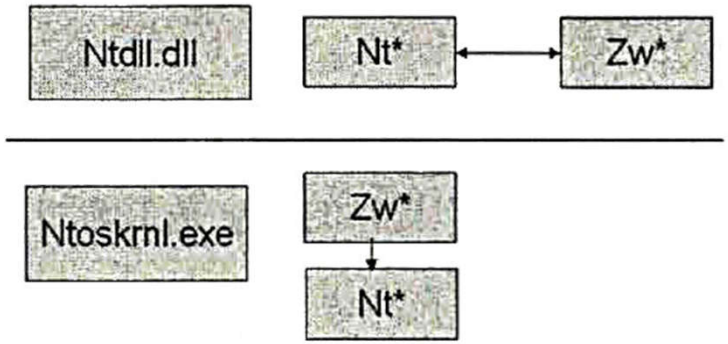
- 7. 登录阶段
  - Windows子系统启动的winlogon.exe系统服务提供对Windows用户的登录和注销的支持，可以完成如下工作
    - 启动服务子系统（services.exe），也称服务控制管理器（SCM）
    - 启动本地安全授权（LSA）过程（lsass.exe）
    - 显示登录界面
  - 登录组件将用户的账号和密码安全地传送给LSA进行认证处理。如果用户提供的信息是正确的，能够通过认证，就允许用户对系统进行访问

Windows 与内核启动过程

- 8. Windows 7和WindowsXP启动过程的区别
  - BIOS通过自检后，将MBR载入内存并执行，引导代码找到启动管理器Bootmgr
  - Bootmgr寻找活动分区boot文件夹中的启动配置数据BCD文件，读取并组成相应语言的启动菜单，然后在屏幕上显示多操作系统选择画面
  - 选择Windows7系统后，Bootmgr 就会读取系统文件windowslsystem32\winload.exe，并将控制权交给winload.exe
  - Winload.exe加载Windows7的内核、硬件、服务等，然后加载桌面等信息，从而启动整个Windows7系统

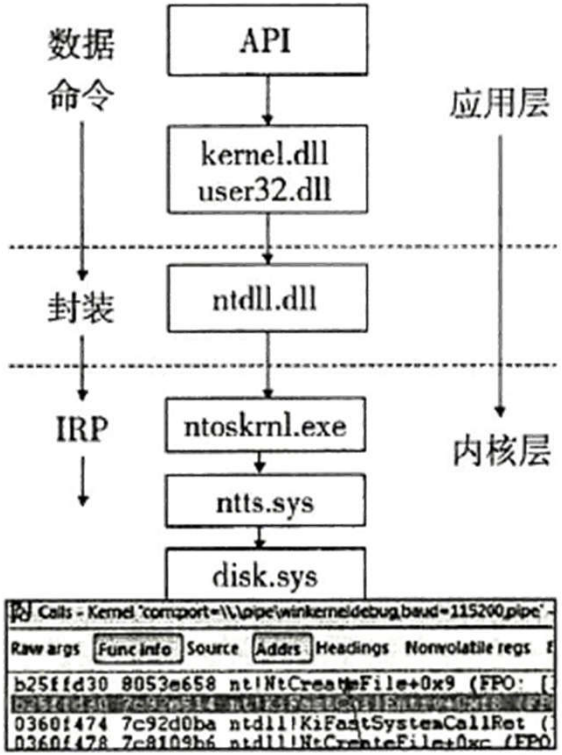
Windows R3与R0通信

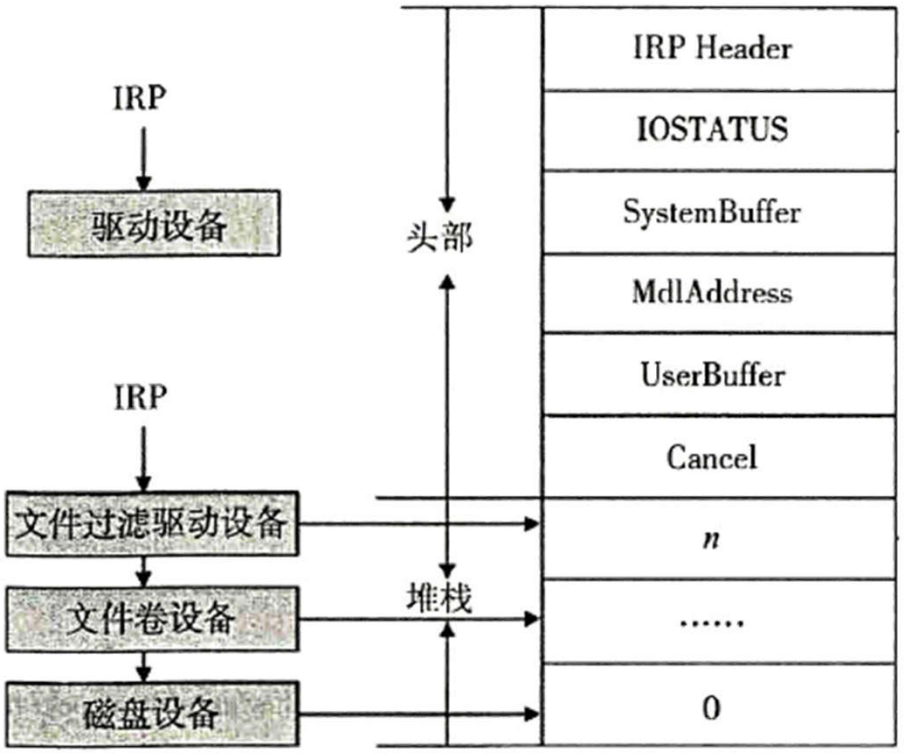
- 当应用程序调用一个有关I/O的API（例如WriteFile）时，实际上这个API被封装在应用层的某个DLL库（例如kernel32.dll和user32.dll）文件中。而DLL动态库中的函数的更底层的函数包含在ntdll.dll文件中，也就是会调用在ntdll.dll中的Native API函数。ntdll.dll中的Natvie API函数是成对出现的，分别以“Nt”和“Zw”开头（例如ZwCreateFile、NtCreateFile）。
- 当kernel32.dll中的API通过ntdll.dll执行时，会完成参数的检查工作，再调用一个中断（int2Eh 或者SysEnter指令），从R3层进入R0。在内核ntoskrnl.exe中有一个SSDT，里面存放了与ntdll.dll中对应的SSDT系统服务处理函数，即内核态的Nt*系列函数，它们与ntdl.dll中的函数一一对应



Windows R3与R0通信

- 在这个过程中，应用层的命令和数据会被系统的I/O管理器封装在一个叫作IRP的结构中。之后，IRP会将R3发下来的数据和命令逐层发送给下层的驱动创建的设备对象进行处理，完成对应的功能





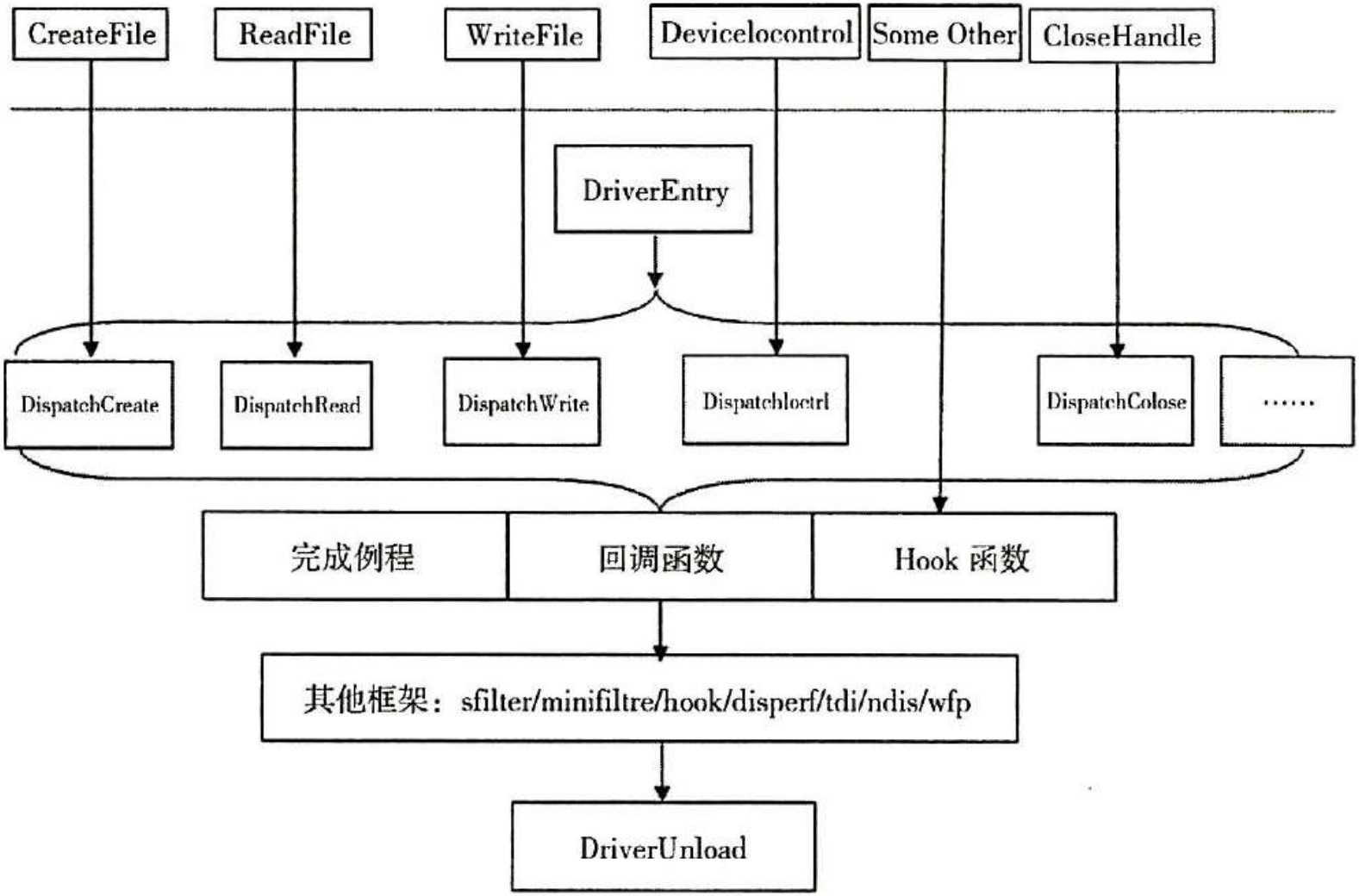
Windows R3与R0通信

```
kd> kb
ChildEBP RetAddr  Args to Child
f8341a4c 804ef129 822f5f18 81fb04f8 81fb04f8 ntmodeldrv!DispatchCreate+0x19
f8341a5c 80579696 822f5f00 81e7f334 f8341c04 nt!IopfCallDriver+0x31
f8341b3c 805b5d74 822f5f18 00000000 81e7f290 nt!IopParseDevice+0xa12
f8341bc4 805b211d 00000000 f8341c04 00000040 nt!ObpLookupObjectName+0x56a
f8341c18 8056c2a3 00000000 00000000 090cb801 nt!ObOpenObjectByName+0xeb
f8341c94 8056cc1a 0012e124 c0100080 0012e0c4 nt!IopCreateFile+0x407
f8341cf0 8056f32c 0012e124 c0100080 0012e0c4 nt!IoCreateFile+0x8e
f8341d30 8053e648 0012e124 c0100080 0012e0c4 nt!NtCreateFile+0x30
f8341d30 7c92e4f4 0012e124 c0100080 0012e0c4 nt!KiFastCallEntry+0xf8
0012e080 7c92d09c 7c8109a6 0012e124 c0100080 ntdll!KiFastSystemCallRet
0012e084 7c8109a6 0012e124 c0100080 0012e0c4 ntdll!ZwCreateFile+0xc
0012e11c 7c801a53 00000000 c0000000 00000000 kernel32!CreateFileW+0x35f
0012e140 0042de78 00483f84 c0000000 00000000 kernel32!CreateFileA+0x30
0012fe8c 0042e2f1 00330032 00330030 7ffdf000 main!TestDriver+0x48 [d:\malloc
0012ff6c 0042eba7 00000001 003d31c8 003d3210 main!main+0x61 [d:\mallocfree_n
0012ffb8 0042ea7f 0012fff0 7c817067 00330032 main!__tmainCRTStartup+0x117 [f
0012ffc0 7c817067 00330032 00330030 7ffdf000 main!mainCRTStartup+0xf [f:\dd\
0012fff0 00000000 0042bbef 00000000 78746341 kernel32!BaseProcessStart+0x23
```

Windows R3与R0通信

- 内核主要由各种驱动（在磁盘上是sys文件）组成，Windows系统自带的（例如ntfs.sys、tcpip.sys、win32k.sys或由第三方软件厂商提供的
- 驱动加载之后，会生成对应的设备对象，并可以选择向R3提供一个可供访问和打开的符号链接。常见的盘符C、D等其实都是文件系统驱动创建的设备对象的符号链接
- 应用层程序可以根据内核驱动的符号链接名调用CreateFile()函数打开。在获得一个句柄（Handle）之后，程序就可以调用应用层函数与内核驱动进行通信了，例如ReadFile () 、WriteFile ()及DeviceloControl ()等
- 内核驱动一旦执行了DriverEntry0入口函数，就可以接收R3层的通信请求了。在内核驱动中专门有一组分发派遣函数用来分别响应应用层的调用请求

Windows R3与R0通信



内核函数

- Windows内核部分会调用一些内核层的函数。这些函数都以固定的前缀开始，分别属于内核中不同的管理模块。通过这些前缀，根据函数名就可以大致知道这个函数所属的层次和模块了

内核函数

- Ex：管理层。“Ex”是“Executive”的开头两个字母
- Ke：核心层。“Ke”是“Kernel”的开头两个字母
- HAL：硬件抽象层。“HAL”是“Hardware Abstraction Layer”的缩写
- Ob：对象管理。“Ob”是“Object”的开头两个字母
- MM：内存管理。“MM”是“Memory Manager”的缩写
- Ps：进程（线程）管理。“Ps”表示“Process”
- Se：安全管理。“Se”是“Security”的开头两个字母
- Io： I/O管理
- Fs：文件系统。“Fs”是“File System”的缩写
- Cc：文件缓存管理。“Cc”表示“Cache”
- Cm：系统配置管理。“Cm”是“Configuration Manager”的缩写
- Pp：即插即用管理。“Pp”表示“PnP”
- Rtl：运行时程序库。“Rtl”是“Runtime Library”的缩写
- Zw/Nt：对应于SSDT中的服务函数，例如与文件或者注册表相关的操作函数
- Flt:Minifilter文件过滤驱动中调用的函数
- Ndis:Ndis 网络框架中调用的函数

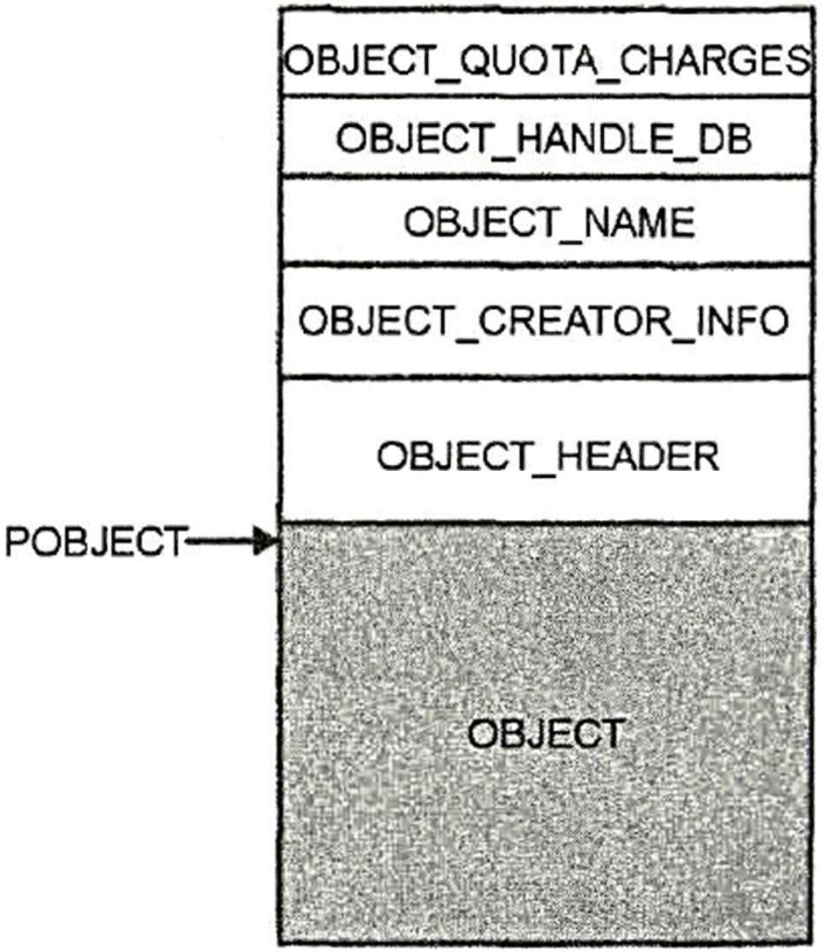
内核对象

- 在Windows内核中有一种很重要的数据结构管理机制，那就是内核对象
- 应用层的进程、线程、文件、驱动模块、事件、信号量等对象或者打开的句柄在内核中都有与之对应的内核对象
- Dispatcher对象：
- I/O对象：DEVICE_OBJECT、DRIVER_OBJECT、FILE_OBJECT等
- 其他对象：进程对象（EPROCESS）与线程对象（ETHREAD ）

内核对象

- 一个Windows内核对象可以分为对象头和对象体两部分。在对象头中至少有1个OBJECT_HEADER和对象额外信息。对象体紧接着对象头中的OBJECT_HEADER
- 一个对象指针总是指向对象体而不是对象头。如果要访问对象头，需要将对象体指针减去一个特定的偏移值，以获取OBJECT_HEADER 结构，通过OBJECT_HEADER结构定位从而访问其他对象结构辅助信息
- 对象体内部一般会有1个type和1个size成员，用来表示对象的类型和大小

内核对象


OBJECT_QUOTA_CHARGES
OBJECT_HANDLE_DB
OBJECT_NAME
OBJECT_CREATOR_INFO
OBJECT_HEADER
POBJECT
OBJECT

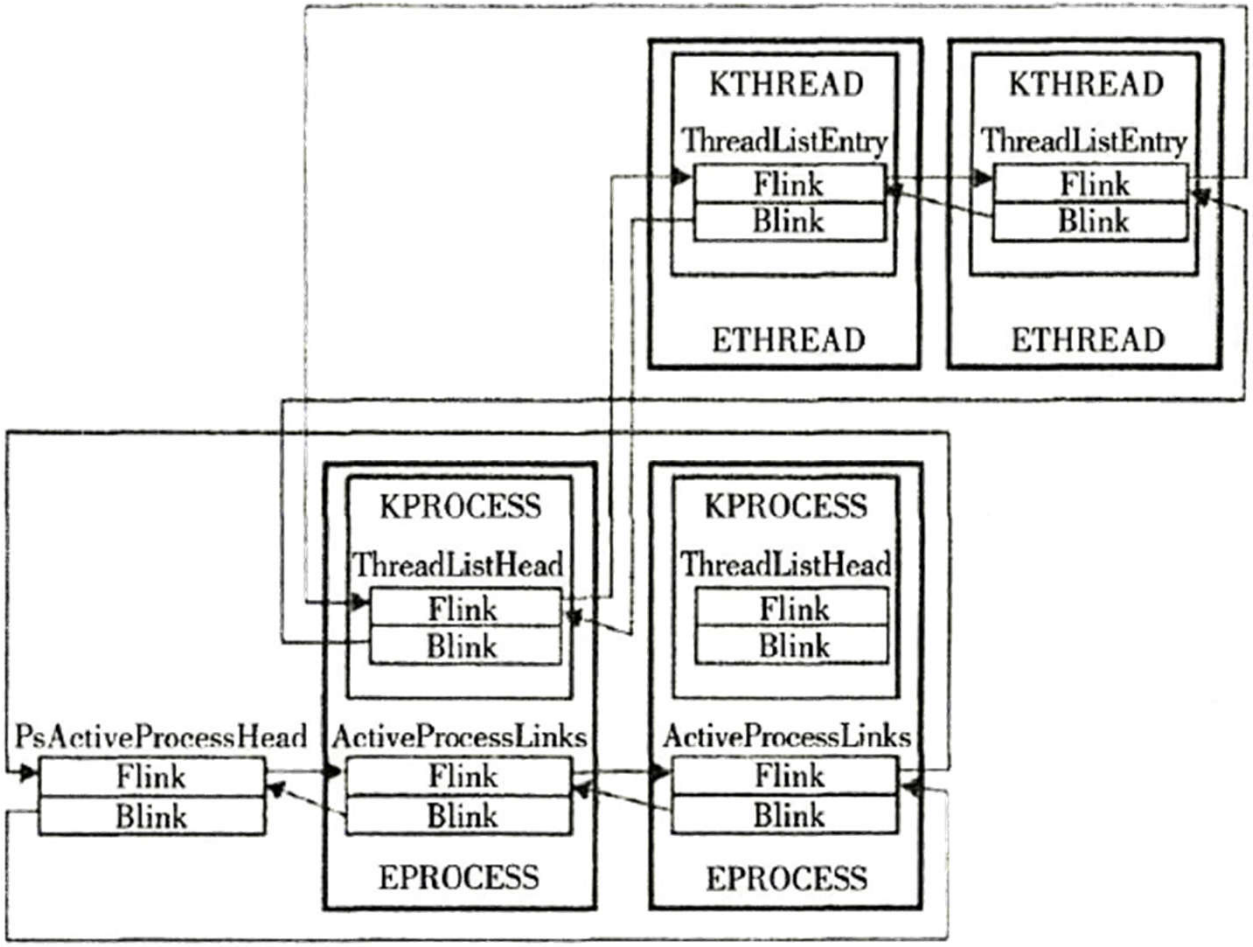
内核对象

- EPROCESS用于在内核中管理进程的各种信息，每个进程都对应于一个EPROCESS结构，用于记录进程执行期间的各种数据
- 它是一个不透明的结构（Opaque Structure），具体成员并未导出，并随着操作系统版本的变化而变化
- 所有进程的EPROCESS内核结构都被放入一个双向链表，R3在枚举系统进程的时候，通过遍历这个链表获得了进程的列表。因此，有的Rootkit 会试图将自己进程的EPROCESS结构从这个链表中摘掉，从而达到隐藏自己的目的

内核对象

- ETHREAD结构是线程的内核管理对象。每个线程都有一个对应的ETHREAD结构
- ETHREAD结构也是一个不透明的结构，具体成员并未导出，而且会随着操作系统版本的变化而变化。在ETHREAD结构中，第1个成员就是线程对象KTHREAD成员，所有的ETHREAD结构也被放在一个双向链表里进行管理

内核对象

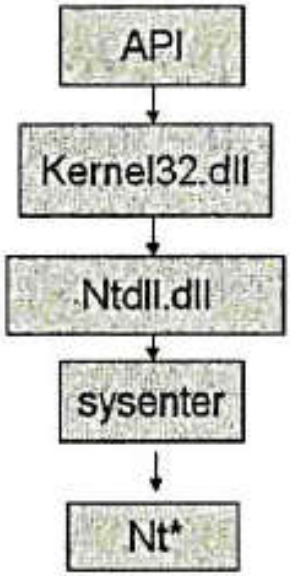

KTHREAD
ThreadListEntry
Flink
Blink
ETHREAD
KTHREAD
ThreadListEntry
Flink
Blink
ETHREAD
KPROCESS
ThreadListHead
Flink
Blink
KPROCESS
ThreadListHead
Flink
Blink
PsActiveProcessHead
Flink
Blink
ActiveProcessLinks
Flink
Blink
EPROCESS
ActiveProcessLinks
Flink
Blink
EPROCESS

内核对象

- SSDT：“SSDT”的全称是“System Services Descriptor Table”（系统服务描述符表），在内核中的实际名称是“KeServiceDescriptorTable”
- SSDT用于处理应用层通过kernel32.d11下发的各个API操作请求。ntdll.dll中的API是一个简单的包装函数，当kernel32.dll中的API通过ntdll.dll时，会先完成对参数的检查，再调用一个中断（int 2Eh或者SysEnter指令），从而实现从R3层进入R0层，并将要调用的服务号（也就是SSDT数组中的索引号index值）存放到寄存器EAX中，最后根据存放在EAX中的索引值在SSDT数组中调用指定的服务（Nt*系列函数）

内核对象

- SSDT



索引1	索引2	索引3	索引4	索引5	……	索引N
服务1 (NtXxx)	服务2	服务3	服务4	服务5	……	服务N

内核对象

- TEB：应用层中的结构
- TEB（Thread environment block，线程环境块）结构中包含了系统频繁使用的一些与线程相关的数据。进程中的每个线程（系统线程除外）都有一个自己的TEB。一个进程的所有TEB都存放在从0x7FFDE000开始的线性内存中，每4KB为一个完整的TEB

内核对象

- TEB

```
lkd> !teb
TEB at 7ffdd000
    ExceptionList:          00949794
    StackBase:              00950000
    StackLimit:             00943000
    SubSystemTib:           00000000
    FiberData:              00001e00
    ArbitraryUserPointer:   00000000
    Self:                   7ffdd000
    EnvironmentPointer:     00000000
    ClientId:               00000774 . 00000200
    RpcHandle:              00000000
    Tls Storage:            00000000
    PEB Address:            7ffdf000
    LastErrorValue:         0
    LastStatusValue:        c0000139
    Count Owned Locks:      0
    HardErrorMode:          0
```

内核对象

- PEB：PEB（Process Environment Block，进程环境块）存在于用户地址空间中，记录了进程的相关信息。每个进程都有自己的PEB信息

```
typedef struct _PEB {
  BYTE                          Reserved1[2];
  BYTE                          BeingDebugged;
  BYTE                          Reserved2[1];
  PVOID                         Reserved3[2];
  PPEB_LDR_DATA                 Ldr;
  PRTL_USER_PROCESS_PARAMETERS  ProcessParameters;
  BYTE                          Reserved4[104];
  PVOID                         Reserved5[52];
  PPS_POST_PROCESS_INIT_ROUTINE PostProcessInitRoutine;
  BYTE                          Reserved6[128];
  PVOID                         Reserved7[1];
  ULONG                         SessionId;
} PEB, *PPEB;
```

内核对象

- PEB

```
lkd> dt _PEB 0x7ffdf000
nt!_PEB
   +0x000 InheritedAddressSpace    : 0 ''
   +0x001 ReadImageFileExecOptions : 0 ''
   +0x002 BeingDebugged            : 0 ''
   +0x003 SpareBool                : 0 ''
   +0x004 Mutant                   : 0xffffffff
   +0x008 ImageBaseAddress         : 0x01000000
   +0x00c Ldr                      : 0x00191e90 _PEB_LDR_DATA
   +0x010 ProcessParameters        : 0x00020000 _RTL_USER_PROCESS_PARAMETERS
   +0x014 SubSystemData            : (null)
   +0x018 ProcessHeap              : 0x00090000
   +0x01c FastPebLock              : 0x7c99d600 _RTL_CRITICAL_SECTION
   +0x020 FastPebLockRoutine       : 0x7c921000
   +0x024 FastPebUnlockRoutine     : 0x7c9210e0
   +0x028 EnvironmentUpdateCount   : 1
   +0x02c KernelCallbackTable      : 0x77d12970
   //更多代码略
```

Windows程序

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- Windows编程有两种方法：
  - windows c方式(SDK)，SDK编程就是直接调用windows的API进行编程
  - c++方式：即对SDK函数进行包装，如VC的MFC，BCB的OWL等。MFC把这些API封闭起来，共有一百多个类组成
- Win api就是应用程序和windows之间通讯的一个编程界面
- Windows提供了上千个API函数，以方便程序员来编写应用程序

Windows程序

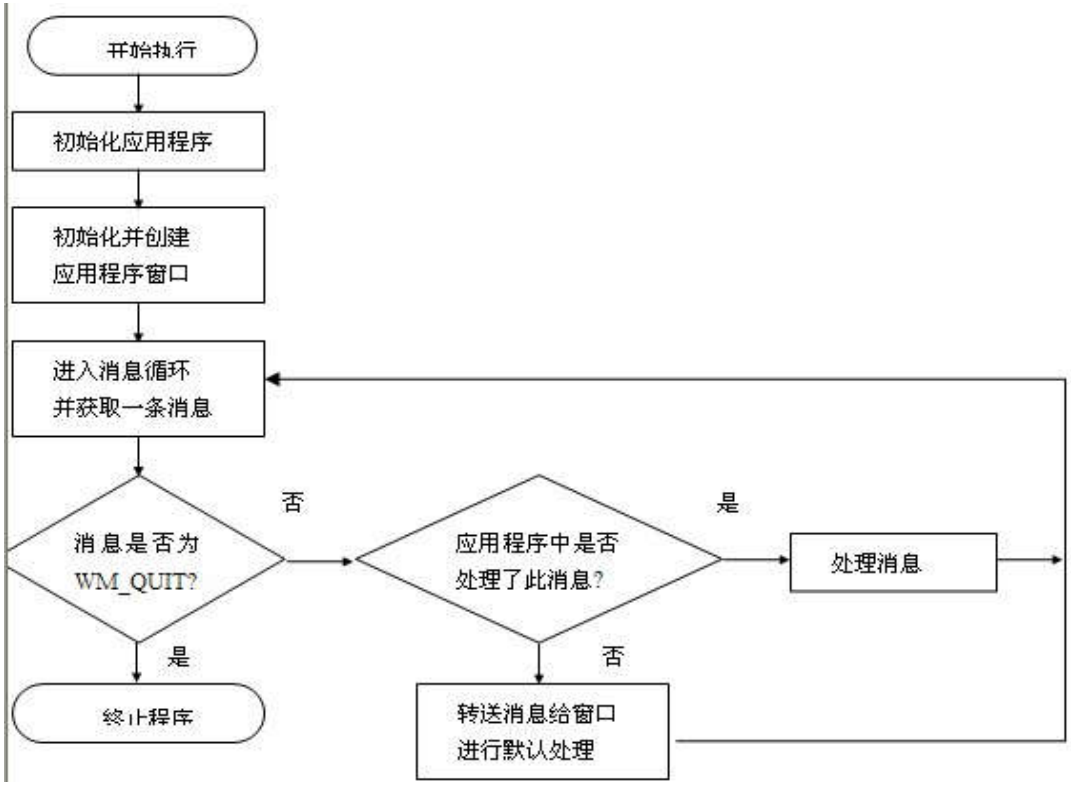
哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- WinSDK程序设计就是API方式的windows程序设计。SDK，全称Software Developers Kit,意思是软件开发工具箱

- MFC，全称Microsoft Foundation Classes，微软把WinAPI进行封装的类库。它是一个类的集合，通过覆盖WinAPI，为编程提供了一个面向对象的界面。它使windows程序员能够利用C++面象对象的特性进行编程，类似BCB的OWL，Delphi的VCL组件。它把那些进行SDK编程时最繁琐的部分提供给程序员，使之专注于功能的实现

Windows程序

- 基本上是这样运行的，程序从WinMain()开始，然后进入一个message loop，程序在这里等待发给它的所有消息然后一一处理，直到接收到WM_QUIT的消息的时候，message loop终止，程序结束。所以整个主程序运行的过程就是等待消息，接收消息，然后处理消息的过程



Windows程序

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 利用Windows API函数编写Windows应用程序必须首先了解以下内容：
  - 窗口
  - 事件驱动
  - 句柄
  - 消息

窗口

WinMain函数 功能：
- 注册窗口类，建立窗口及执行必要的初始化
- 进入消息循环，根据接受的消息调用相应的处理过程
- 当消息循环检索到WM_QUIT时终止程序运行

三个基本的组成部分：函数说明、初始化和消息循环

WinMain函数说明

WinMain函数的说明如下：

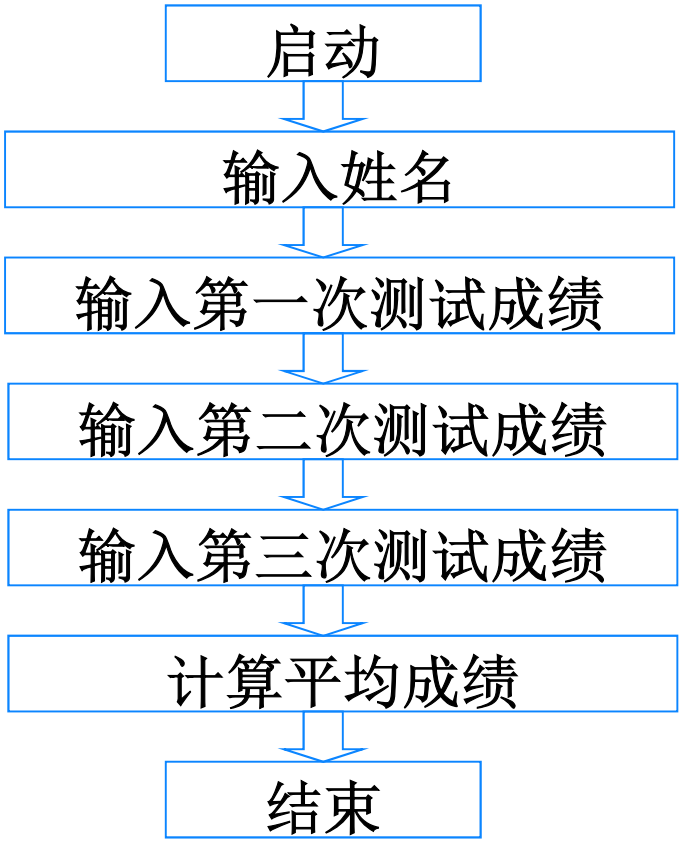
```
int WINAPI WinMain
( HINSTANCE hThisInst,          //应用程序当前实例句柄
  HINSTANCe hPrevInst,          //应用程序其他实例句柄
  LPSTR lpszCmdLine,            //指向程序命令行参数的指针
  Int nCmdShow                  //应用程序开始执行时窗口显示方式的整数值标识
)
```

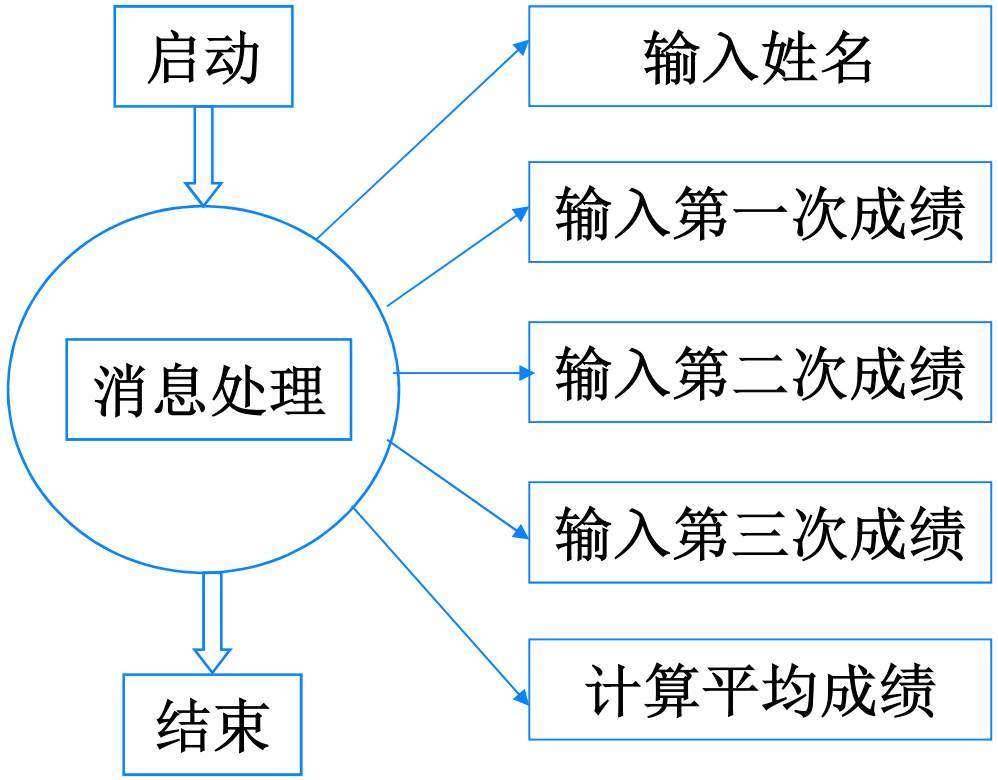
注意！Win是多任务管理的，同一应用程序的多个窗口可能会同时存,Win系统对每个窗口的执行称为一个实例,并用一个实例句柄来唯一标识

Dos的过程驱动与Windows的事件驱动

- DOS程序主要使用顺序的，过程驱动的程序设计方法。顺序的，过程驱动的程序有一个明显的开始，明显的过程及一个明显的结束，因此程序能直接控制程序事件或过程的顺序
- 而Windows的驱动方式是事件驱动，就是不由事件的顺序来控制，而是由事件的发生来控制，所有的事件是无序的，作为一个windows程序员，对正在开发的应用程序要发出或要接收的消息进行排序和管理


1920
HIT


启动
输入姓名
输入第一次测试成绩
输入第二次测试成绩
输入第三次测试成绩
计算平均成绩
结束


启动
消息处理
结束
输入姓名
输入第一次成绩
输入第二次成绩
输入第三次成绩
计算平均成绩

句柄

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 句柄(handle)：在标准C库中句柄用来对文件输入输出。在Windows环境中，句柄是用来标识项目的，这些项目包括：
  - *.模块(module)
    *.任务(task)
    *.实例(instance)
    *.文件(file)
    *.内存块(block of memory)
    *.菜单(menu)
    *.控制(control)
    *.字体(font)
    *.资源(resource),包括图标(icon)，光标(cursor)，字符串(string) *.GDI对象(GDI object),包括位图(bitmap),画刷(brush),元文件(metafile),调色板(palette),画笔(pen),区域(region),以及设备描述表(device context)

句柄

- 在Win32里，句柄是指向一个无值型对象（void *）的指针，是一个4字节长的数据
- 句柄并不是一个真正意义上的指针。从结构上看，句柄的确是一个指针，尽管它没有指向用于存储某个对象的内存位置，而实际上句柄指向的是一个包含了对该对象进行的引用的位置
- 通常一个句柄就可以传递我们所要做的事情。句柄是系统用来标识不同对象类型的工具，如窗口、菜单等，这些东西在系统中被视为不同类型的对象，用不同的句柄将他们区分开来。

句柄

句柄是一个4字节长的数值，用于标识应用程序中不同的对象和同类对象中不同的实例

窗口
按钮
图标
滚动条
输出设备
控制
文件

常用句柄类型及其说明

HWND	窗口句柄	HDC	设备环境句柄
HBITMAP	位图句柄	HCURSOR	光标句柄
HICON	图标句柄	HFONT	字体句柄
HMENU	菜单句柄	HPEN	画笔句柄
HFILE	文件句柄	HBRUSH	画刷句柄
HINSTANCE	当前实例句柄		