

Introdução à WIN32

- Mark E. Russinovich, David A. Solomon, **Microsoft® Windows® Internals, Fourth Edition**, Microsoft Press, 2005 [cap. 2]
- Jeffrey Richter, Christophe Nasarre, *Windows via C/C++*, Fifth Edition, Microsoft Press, 2008 [cap. 1, 2 e 3]
- Microsoft, **Microsoft Developer's Network** (MSDN)

Sumário

- **Arquitectura Windows**
- **Caracteres e *Strings***
- **Objectos do Kernel**
- **Tratamento de erros**



Arquitectura Windows

- Mark E. Russinovich, David A. Solomon, **Microsoft® Windows® Internals, Fourth Edition**, Microsoft Press, 2005 [cap. 2]

Requirements drove the specification of Windows

- **The following requirements drove the specification of Windows NT back in 1989:**
 - Provide a true 32-bit, preemptive, reentrant, virtual memory operating system
 - Run on multiple hardware architectures and platforms
 - Run and scale well on symmetric multiprocessing systems
 - Be a great distributed computing platform, both as a network client and as a server
 - Run most existing 16-bit MS-DOS and Microsoft Windows 3.1 applications
 - Meet government requirements for POSIX 1003.1 compliance
 - Meet government and industry requirements for operating system security
 - Be easily adaptable to the global market by supporting Unicode

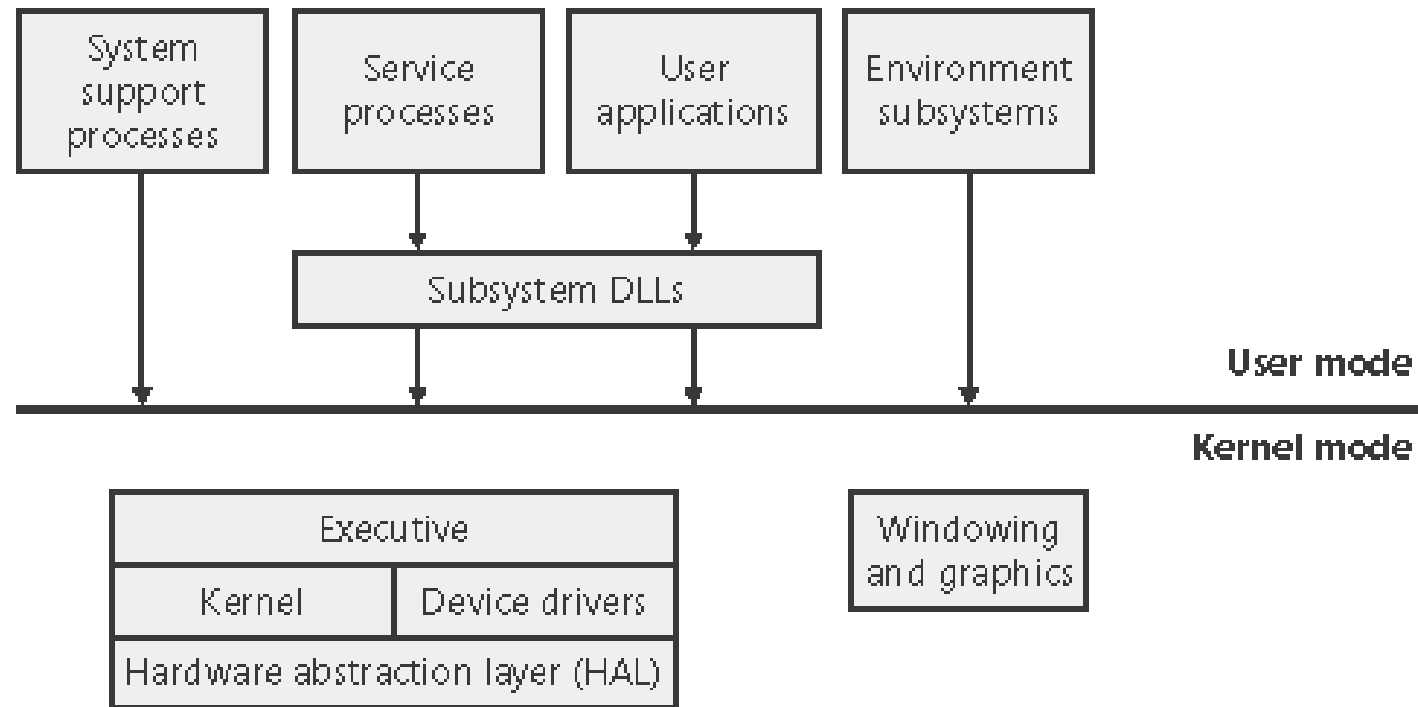


Design goals

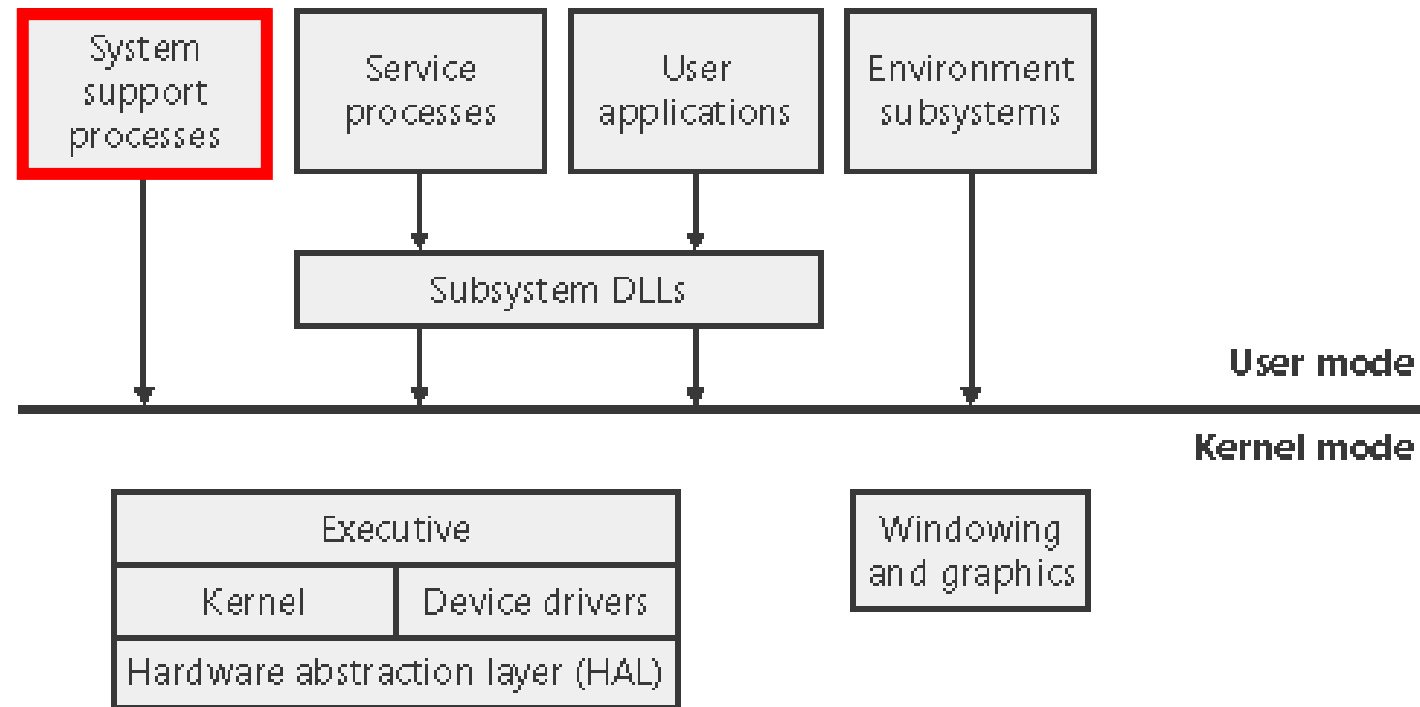
- **Extensibility** - The code must be written to comfortably grow and change as market requirements change
- **Portability** - The system must be able to run on multiple hardware architectures and must be able to move with relative ease to new ones as market demands dictate
- **Reliability and robustness** - The system should protect itself from both internal malfunction and external tampering. Applications should not be able to harm the operating system or other applications
- **Compatibility** - Although Windows NT should extend existing technology, its user interface and APIs should be compatible with older versions of Windows and with MS-DOS. It should also interoperate well with other systems such as UNIX, OS/2, and NetWare
- **Performance** - Within the constraints of the other design goals, the system should be as fast and responsive as possible on each hardware platform



Architecture overview



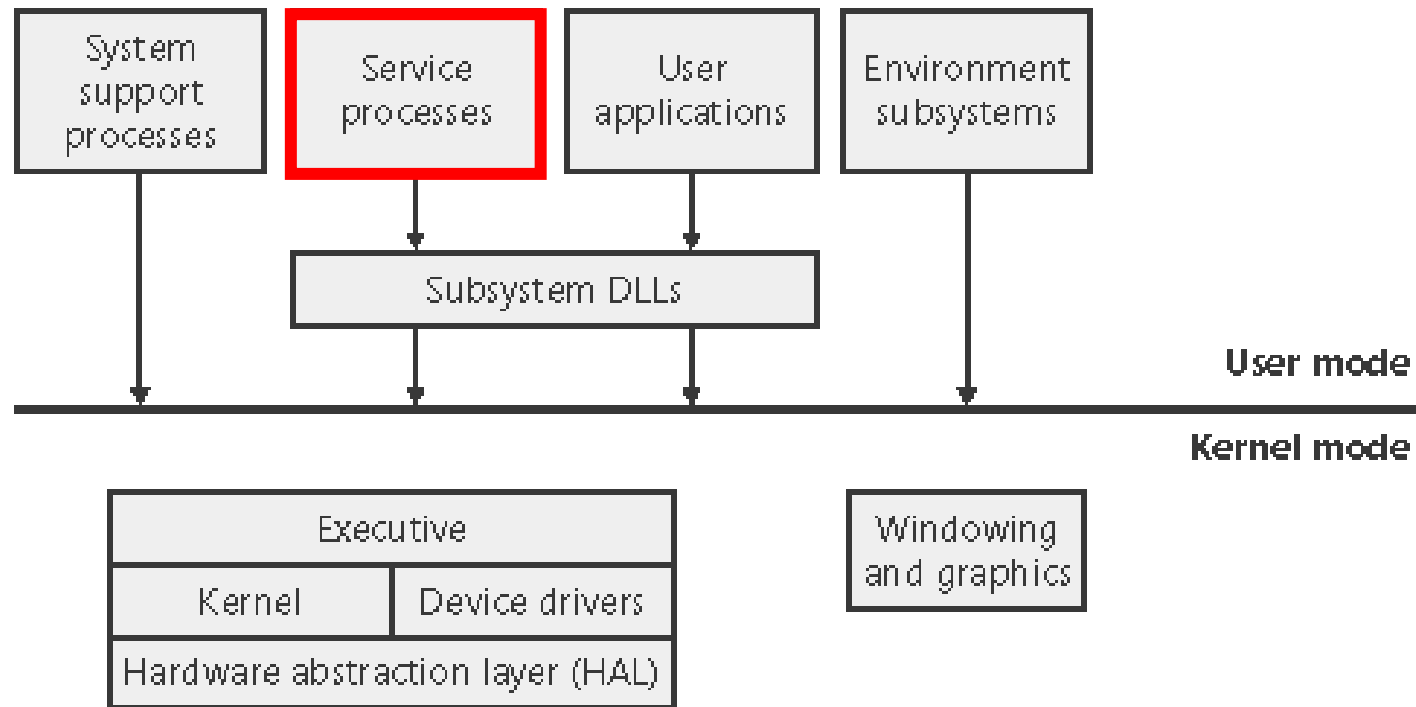
Architecture overview



Such as the logon process and the session manager, that are not Windows services. That is, they are not started by the service control manager



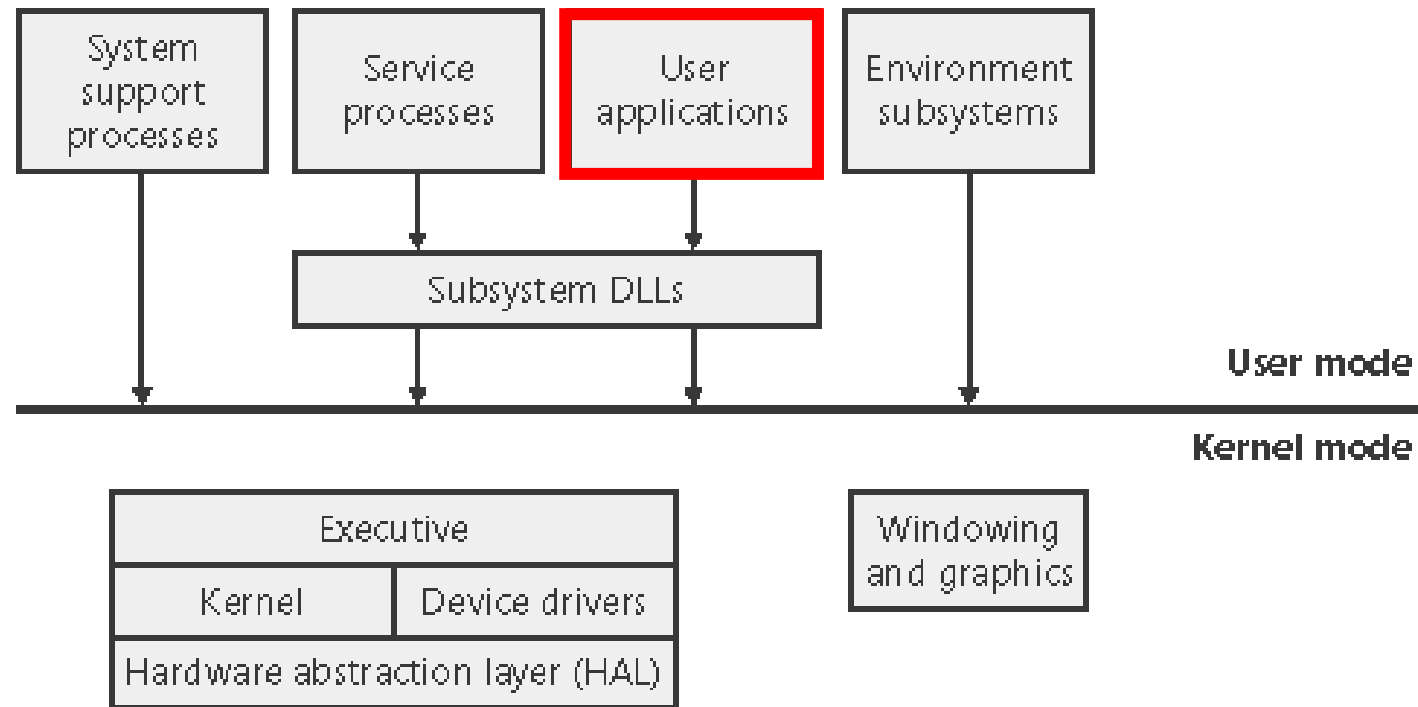
Architecture overview



That host Windows services, such as the Task Scheduler and Spooler services. Services generally have the requirement that they run independently of user logons. Many Windows server applications, such as Microsoft SQL Server and Microsoft Exchange Server, also include components that run as services.



Architecture overview



Which can be one of six types:

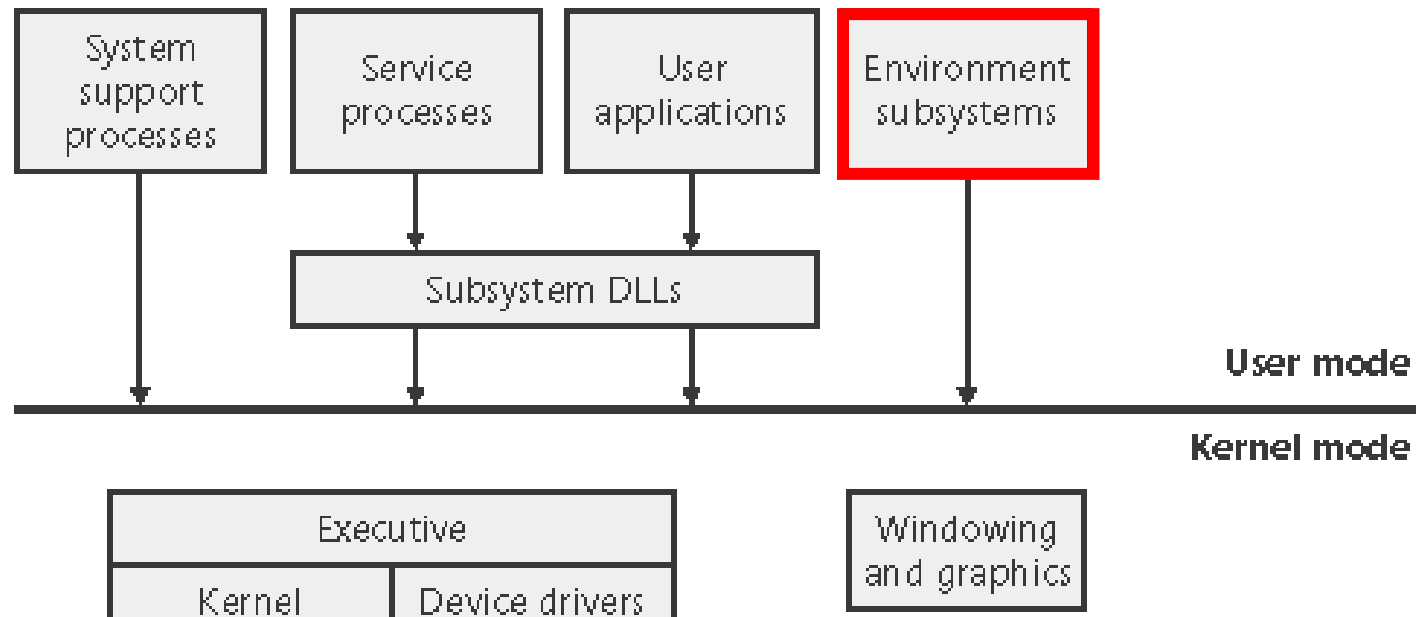
Windows 32-bit, Windows 64-bit,

Windows 3.1 16-bit, MS-DOS 16-bit,

POSIX 32-bit, or OS/2 32-bit.



Architecture overview



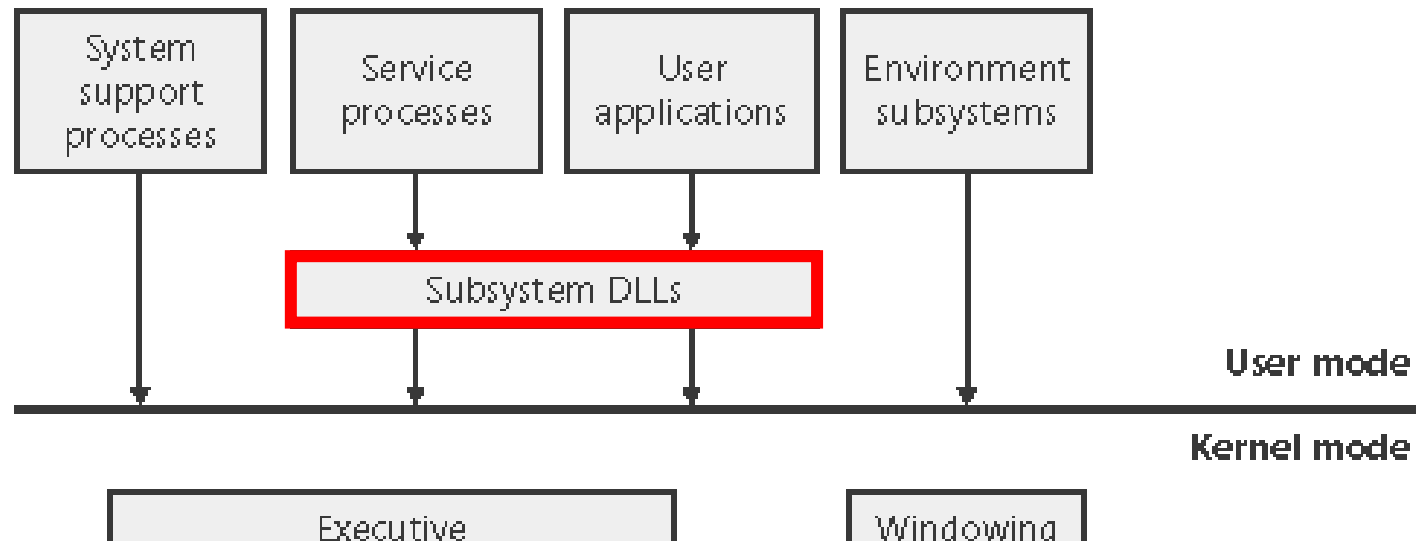
Which implement part of the support for the operating system environment, or personality presented to the user and programmer.

Windows NT originally shipped with three environment subsystems: Windows, POSIX, and OS/2. OS/2 was dropped as of Windows 2000.

As of Windows XP, only the Windows subsystem is shipped in the base product—an enhanced POSIX subsystem is available as part of the free Services for Unix product.



Architecture overview



Under Windows, user applications don't call the native Windows operating system services directly;

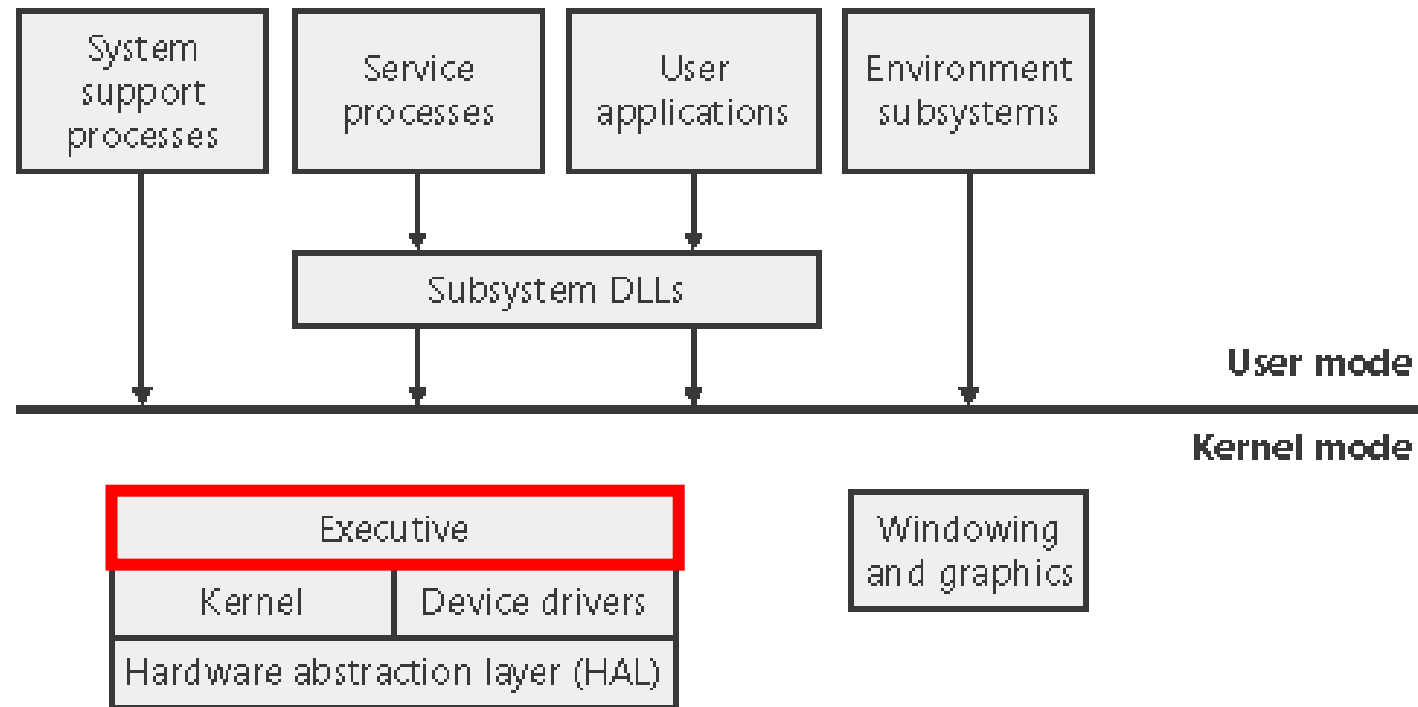
Rather, they go through one or more **subsystem dynamic-link libraries (DLLs)**

The role of the subsystem DLLs is to translate a documented function into the appropriate internal (and generally undocumented) Windows system service calls.

This translation might or might not involve sending a message to the environment subsystem process that is serving the user application.



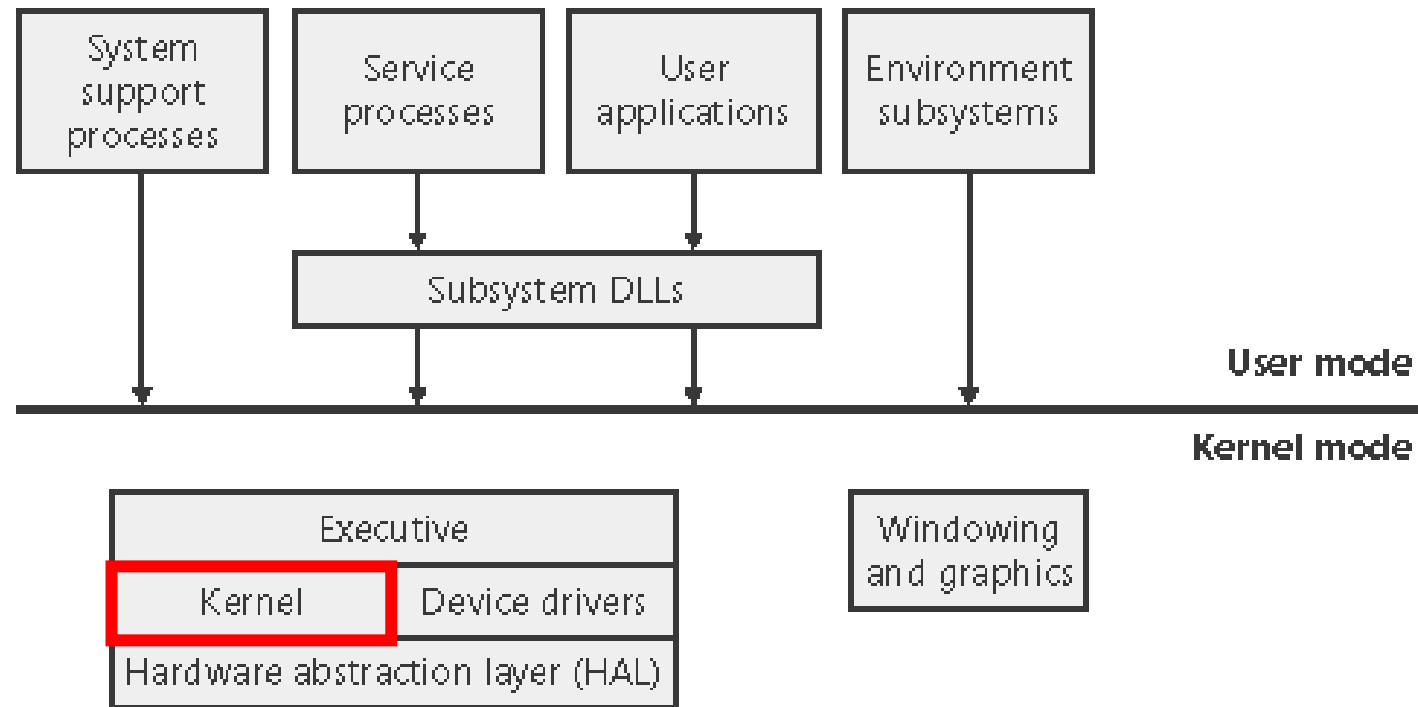
Architecture overview



The Windows executive contains the base operating system services, such as **memory management, process and thread management, security, I/O, networking, and interprocess communication.**



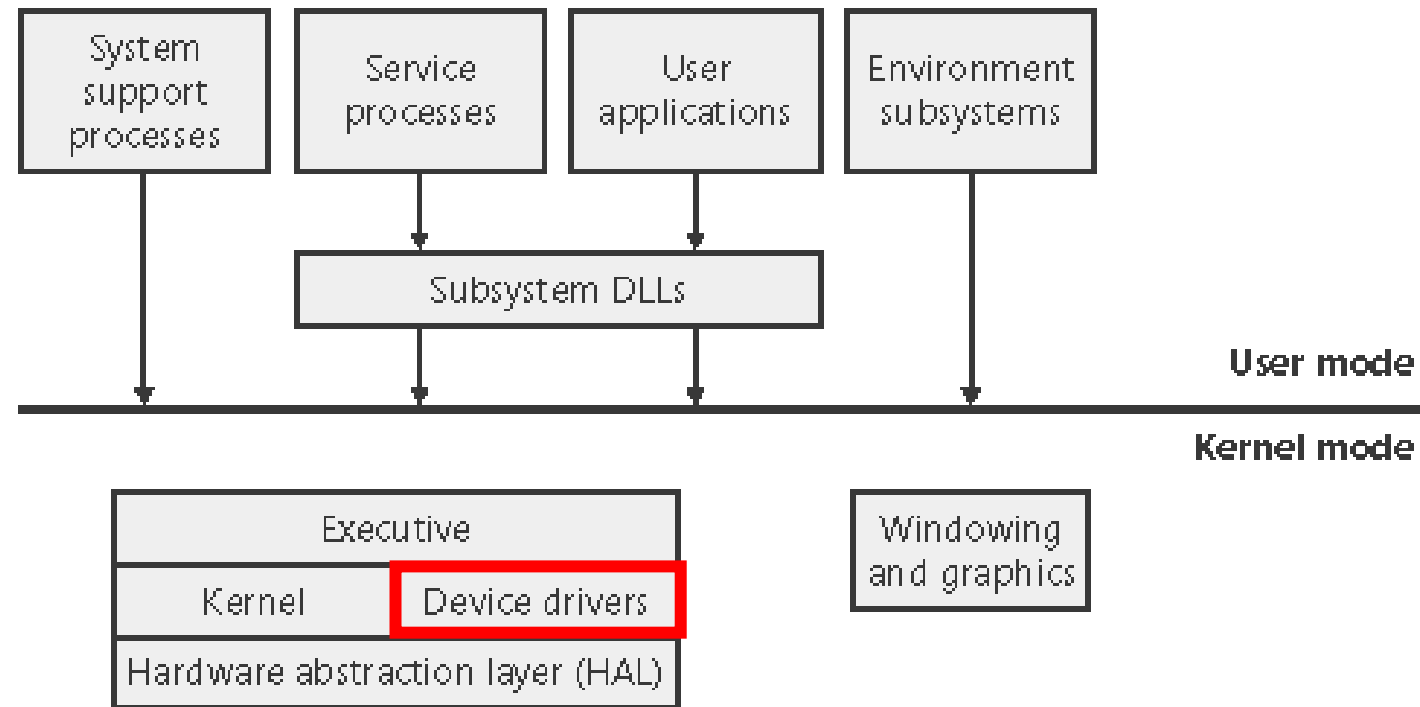
Architecture overview



The Windows kernel consists of low-level operating system functions, such as **thread scheduling, interrupt and exception dispatching**, and **multiprocessor synchronization**. It also provides a set of routines and basic objects that the rest of the executive uses to implement higher-level constructs.



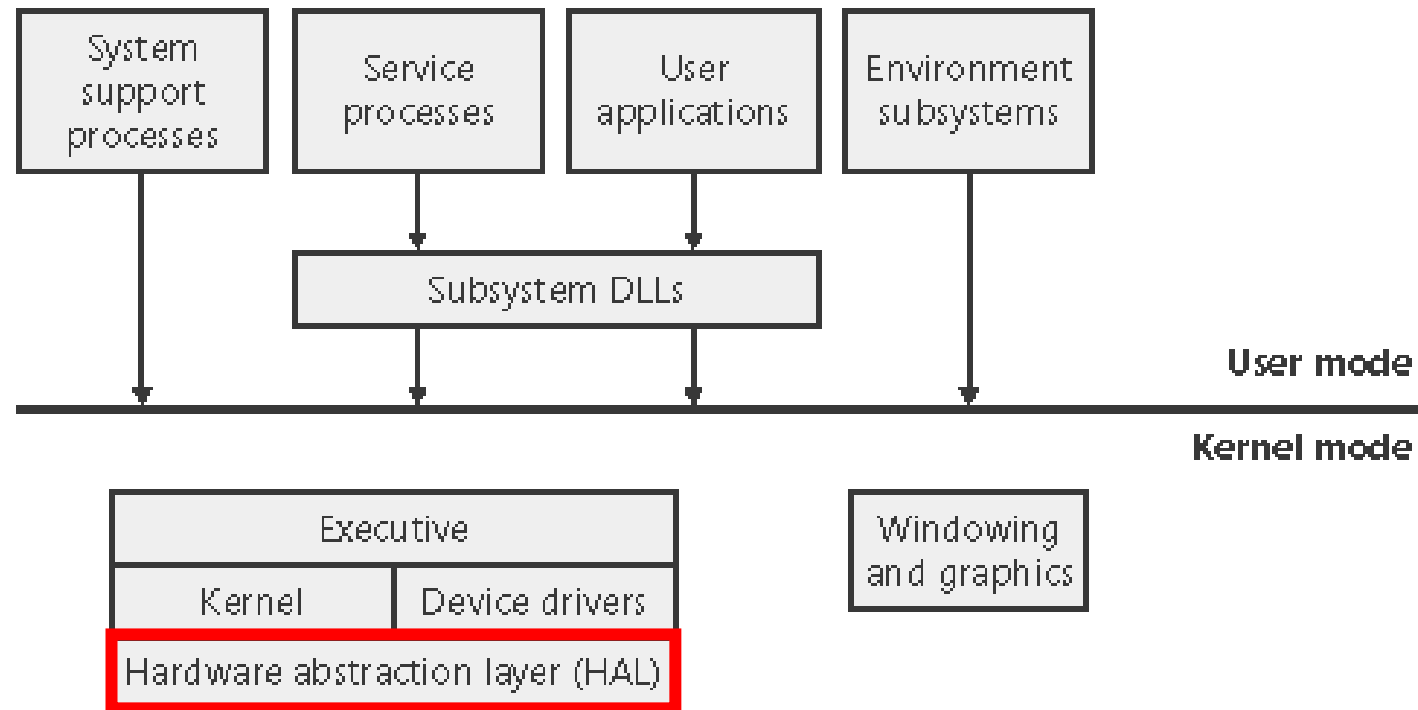
Architecture overview



Device drivers include both hardware device drivers that translate user I/O function calls into specific hardware device I/O requests as well as file system and network drivers.



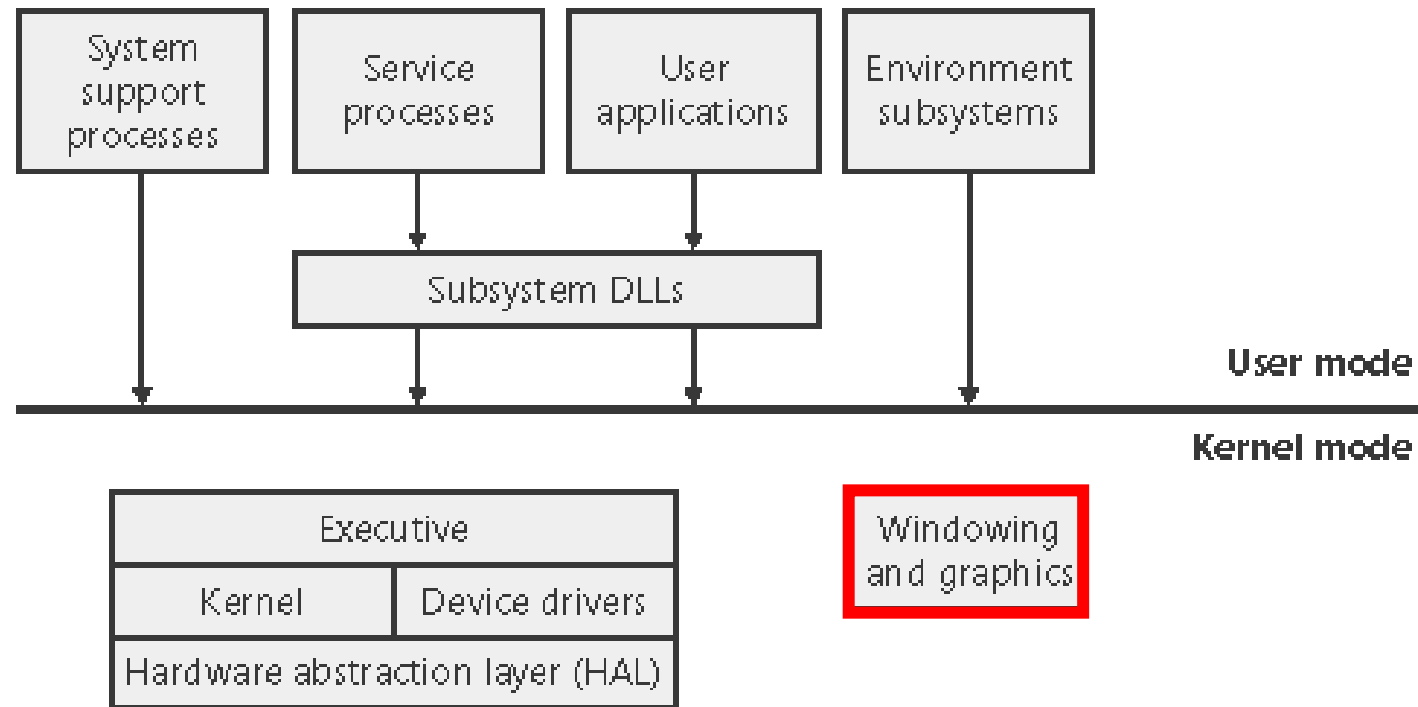
Architecture overview



The hardware abstraction layer (HAL) is a layer of code that isolates the kernel, device drivers, and the rest of the Windows executive from platform-specific hardware differences (such as differences between motherboards).



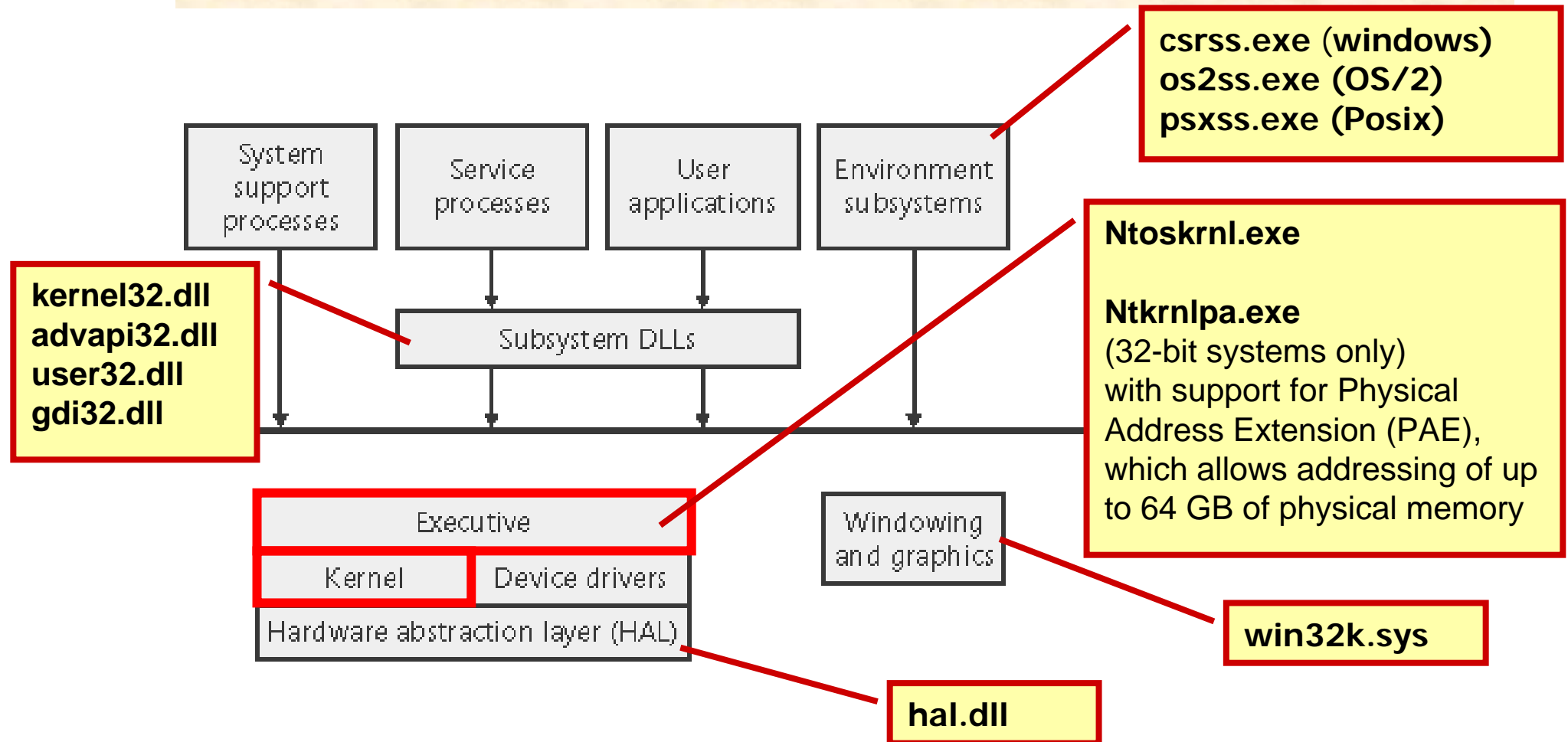
Architecture overview



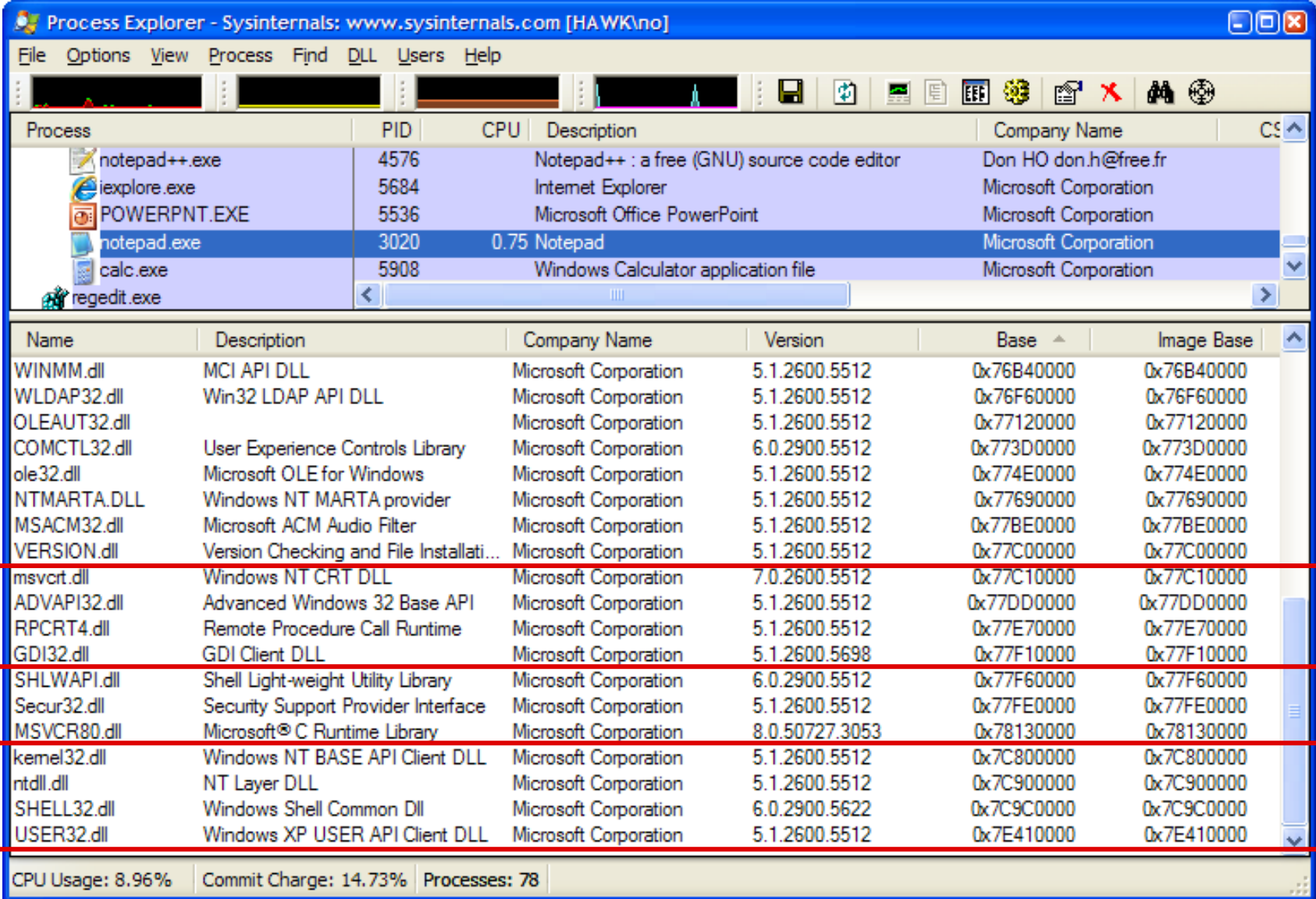
The windowing and graphics system implements the graphical user interface (GUI) functions (better known as the Windows USER and GDI functions), such as dealing with windows, user interface controls, and drawing.



Architecture overview



Subsystem DLLs



Process Explorer - Sysinternals: www.sysinternals.com [HAWK\nu]

File Options View Process Find DLL Users Help

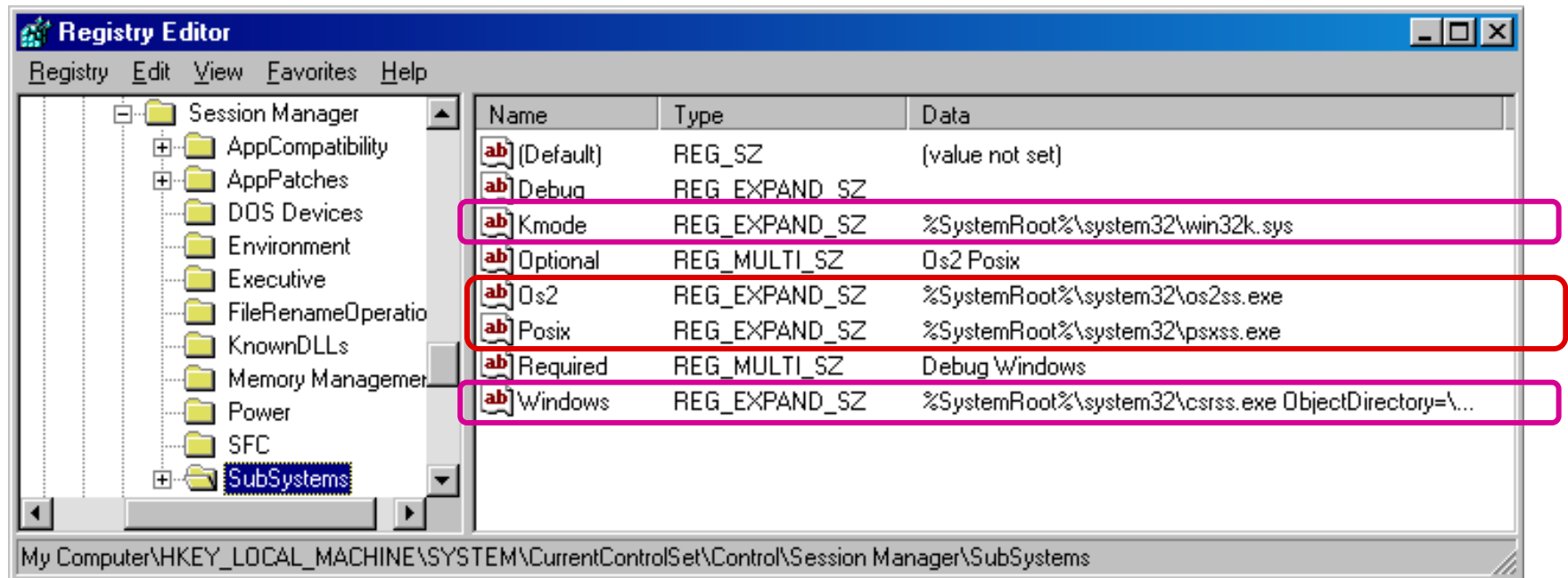
Process	PID	CPU	Description	Company Name	CS
notepad++.exe	4576		Notepad++ : a free (GNU) source code editor	Don HO don.h@free.fr	
ieexplore.exe	5684		Internet Explorer	Microsoft Corporation	
POWERPNT.EXE	5536		Microsoft Office PowerPoint	Microsoft Corporation	
notepad.exe	3020	0.75	Notepad	Microsoft Corporation	
calc.exe	5908		Windows Calculator application file	Microsoft Corporation	
regedit.exe					

Name	Description	Company Name	Version	Base	Image Base
WINMM.dll	MCI API DLL	Microsoft Corporation	5.1.2600.5512	0x76B40000	0x76B40000
WLDAP32.dll	Win32 LDAP API DLL	Microsoft Corporation	5.1.2600.5512	0x76F60000	0x76F60000
OLEAUT32.dll		Microsoft Corporation	5.1.2600.5512	0x77120000	0x77120000
COMCTL32.dll	User Experience Controls Library	Microsoft Corporation	6.0.2900.5512	0x773D0000	0x773D0000
ole32.dll	Microsoft OLE for Windows	Microsoft Corporation	5.1.2600.5512	0x774E0000	0x774E0000
NTMARTA.DLL	Windows NT MARTA provider	Microsoft Corporation	5.1.2600.5512	0x77690000	0x77690000
MSACM32.dll	Microsoft ACM Audio Filter	Microsoft Corporation	5.1.2600.5512	0x77BE0000	0x77BE0000
VERSION.dll	Version Checking and File Installati...	Microsoft Corporation	5.1.2600.5512	0x77C00000	0x77C00000
msvcrt.dll	Windows NT CRT DLL	Microsoft Corporation	7.0.2600.5512	0x77C10000	0x77C10000
ADVAPI32.dll	Advanced Windows 32 Base API	Microsoft Corporation	5.1.2600.5512	0x77DD0000	0x77DD0000
RPCRT4.dll	Remote Procedure Call Runtime	Microsoft Corporation	5.1.2600.5512	0x77E70000	0x77E70000
GDI32.dll	GDI Client DLL	Microsoft Corporation	5.1.2600.5698	0x77F10000	0x77F10000
SHLWAPI.dll	Shell Light-weight Utility Library	Microsoft Corporation	6.0.2900.5512	0x77F60000	0x77F60000
Secur32.dll	Security Support Provider Interface	Microsoft Corporation	5.1.2600.5512	0x77FE0000	0x77FE0000
MSVCR80.dll	Microsoft® C Runtime Library	Microsoft Corporation	8.0.50727.3053	0x78130000	0x78130000
kernel32.dll	Windows NT BASE API Client DLL	Microsoft Corporation	5.1.2600.5512	0x7C800000	0x7C800000
ntdll.dll	NT Layer DLL	Microsoft Corporation	5.1.2600.5512	0x7C900000	0x7C900000
SHELL32.dll	Windows Shell Common Dll	Microsoft Corporation	6.0.2900.5622	0x7C9C0000	0x7C9C0000
USER32.dll	Windows XP USER API Client DLL	Microsoft Corporation	5.1.2600.5512	0x7E410000	0x7E410000

CPU Usage: 8.96% Commit Charge: 14.73% Processes: 78



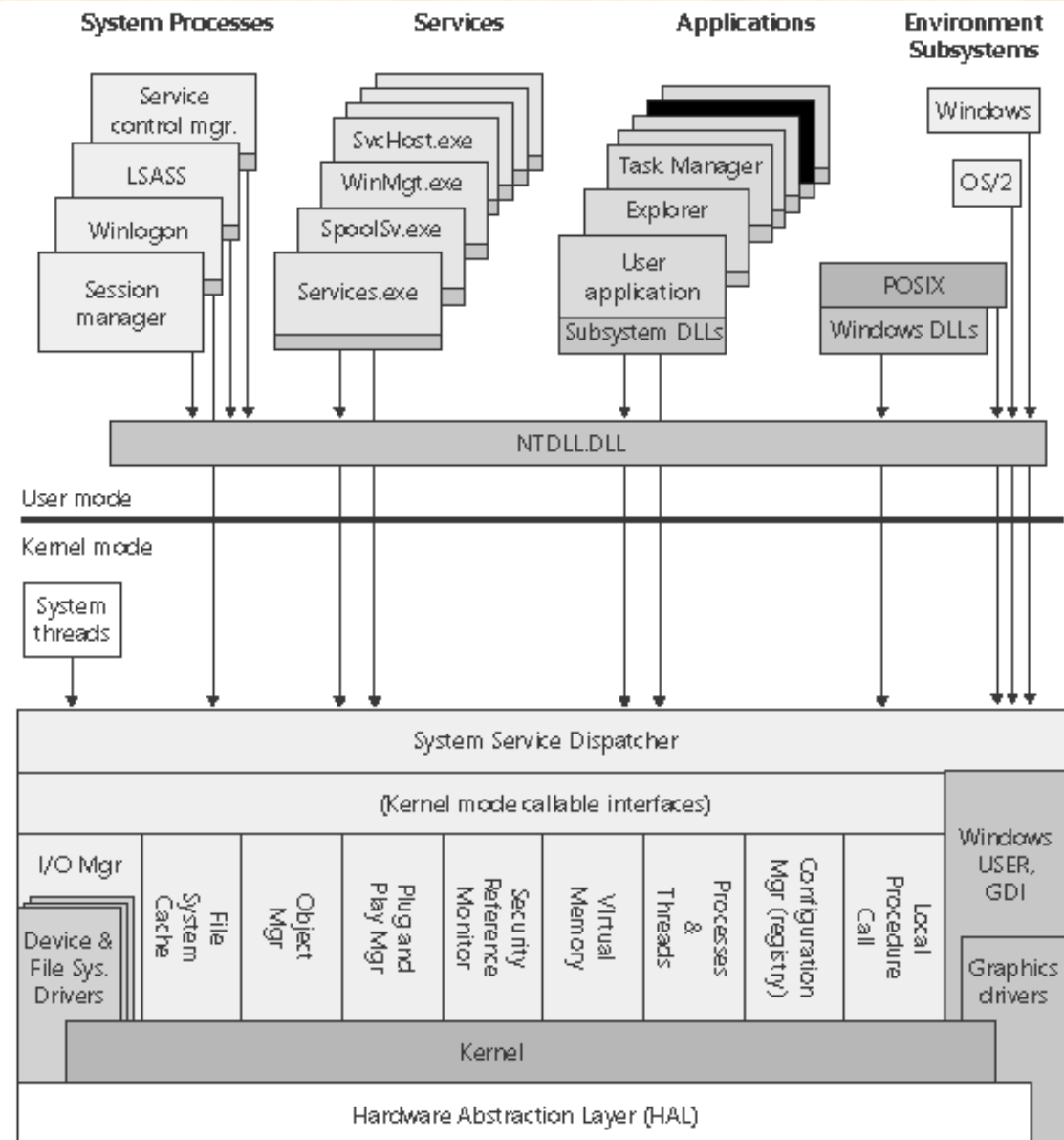
Environment subsystems



A informação de iniciação dos subsistemas é guardada na chave do registry
HKLM\SYSTEM\Current-ControlSet\Control\Session Manager\SubSystems



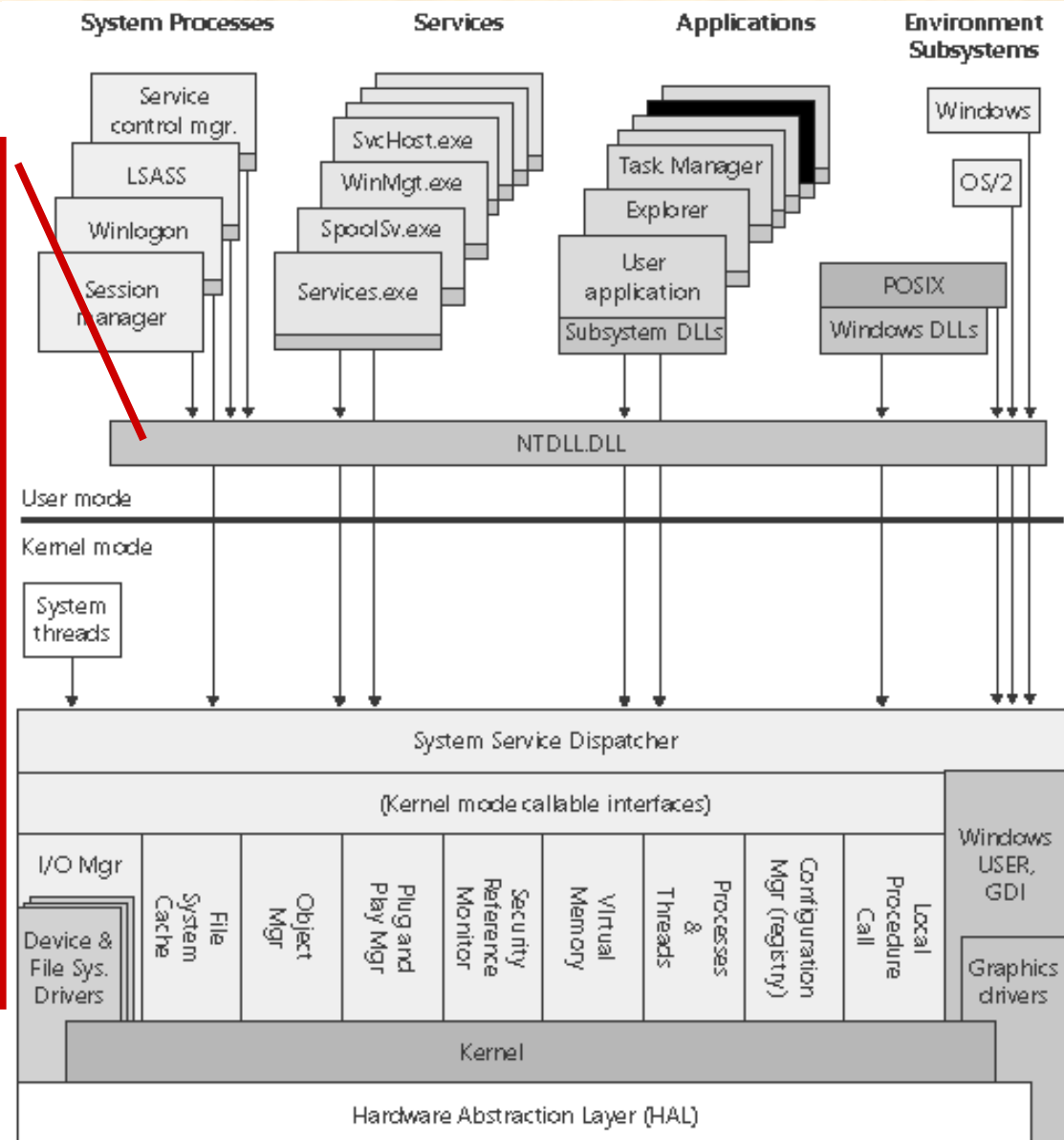
Architecture



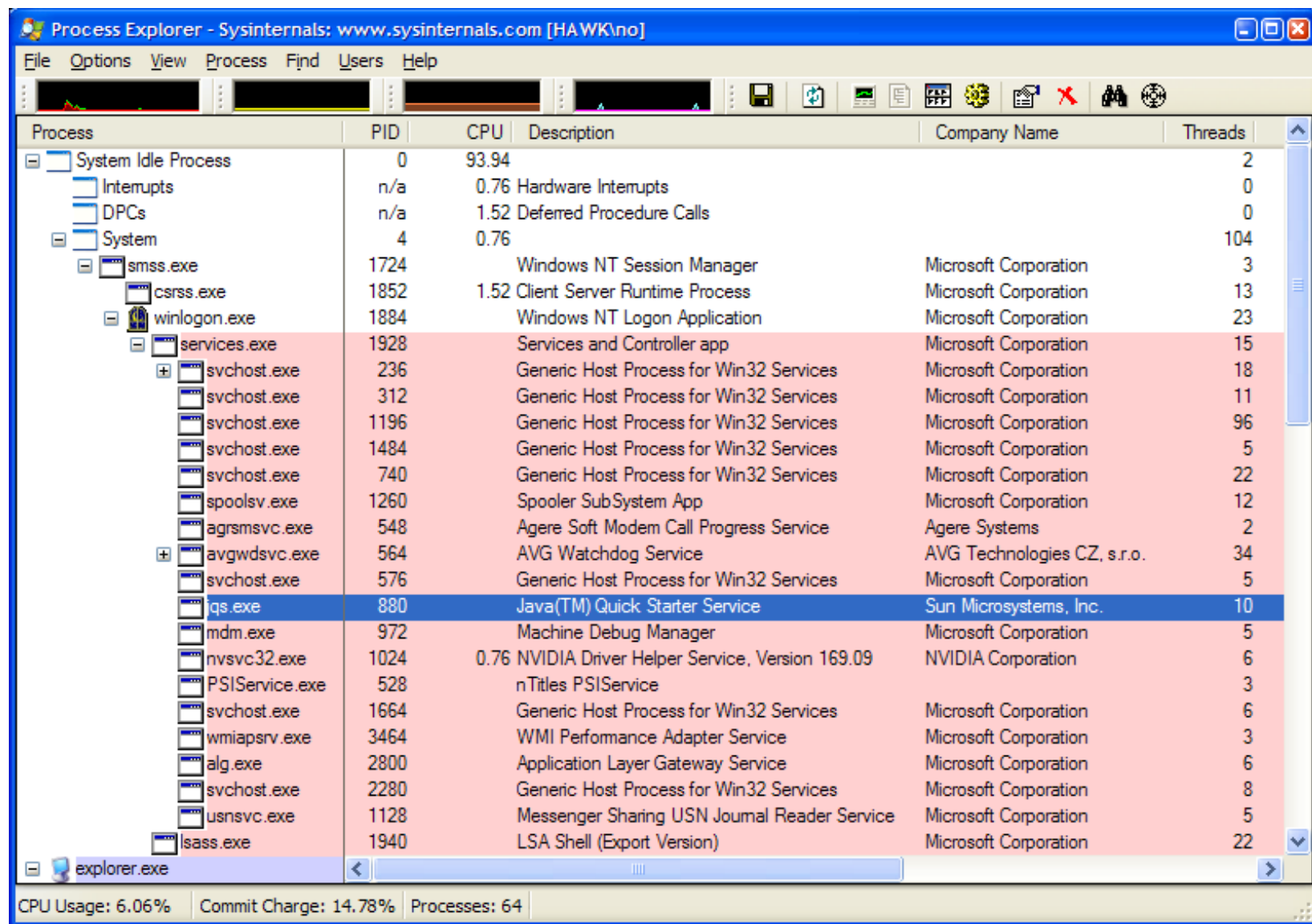
Architecture

Ntdll.dll is a special system support library primarily for the use of subsystem DLLs. It contains two types of functions:

- Group of functions provides the **interface to the Windows executive system services** that can be called from user mode (e.g. NtCreateFile, NtSetEvent)
- **Internal support functions used by subsystems**, subsystem DLLs, and other native images (e.g. image loader, the heap manager, Windows subsystem process communication functions)



Architecture – System Processes



Process Explorer - Sysinternals: www.sysinternals.com [HAWK\no]

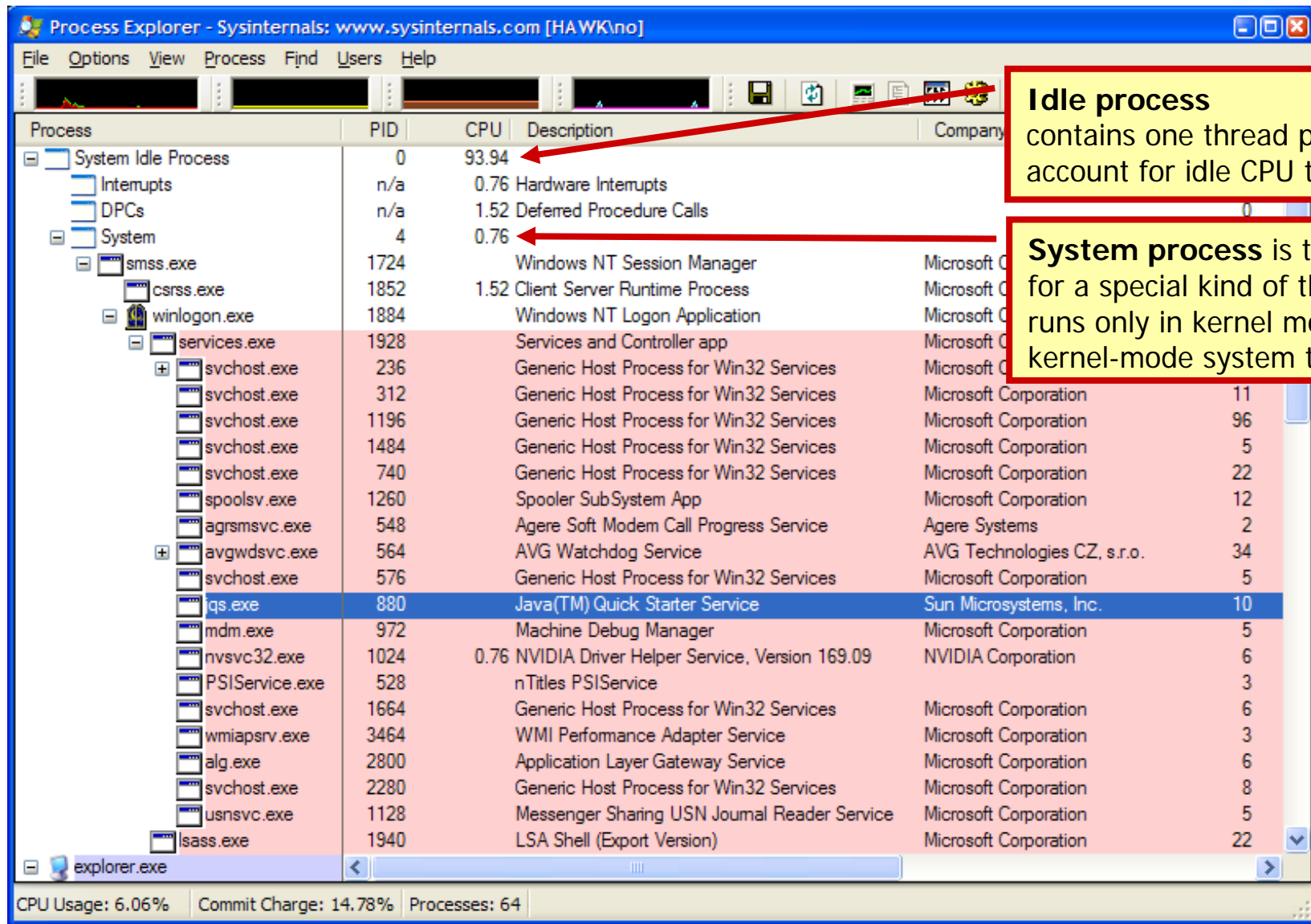
File Options View Process Find Users Help

Process	PID	CPU	Description	Company Name	Threads
System Idle Process	0	93.94			2
Interrupts	n/a	0.76	Hardware Interrupts		0
DPCs	n/a	1.52	Deferred Procedure Calls		0
System	4	0.76			104
smss.exe	1724		Windows NT Session Manager	Microsoft Corporation	3
csrss.exe	1852	1.52	Client Server Runtime Process	Microsoft Corporation	13
winlogon.exe	1884		Windows NT Logon Application	Microsoft Corporation	23
services.exe	1928		Services and Controller app	Microsoft Corporation	15
svchost.exe	236		Generic Host Process for Win32 Services	Microsoft Corporation	18
svchost.exe	312		Generic Host Process for Win32 Services	Microsoft Corporation	11
svchost.exe	1196		Generic Host Process for Win32 Services	Microsoft Corporation	96
svchost.exe	1484		Generic Host Process for Win32 Services	Microsoft Corporation	5
svchost.exe	740		Generic Host Process for Win32 Services	Microsoft Corporation	22
spoolsv.exe	1260		Spooler SubSystem App	Microsoft Corporation	12
agrsmsvc.exe	548		Agere Soft Modem Call Progress Service	Agere Systems	2
avgwdsvc.exe	564		AVG Watchdog Service	AVG Technologies CZ, s.r.o.	34
svchost.exe	576		Generic Host Process for Win32 Services	Microsoft Corporation	5
qs.exe	880		Java(TM) Quick Starter Service	Sun Microsystems, Inc.	10
mdm.exe	972		Machine Debug Manager	Microsoft Corporation	5
nvsvc32.exe	1024	0.76	NVIDIA Driver Helper Service, Version 169.09	NVIDIA Corporation	6
PSIService.exe	528		nTitles PSIService		3
svchost.exe	1664		Generic Host Process for Win32 Services	Microsoft Corporation	6
wmiapsrv.exe	3464		WMI Performance Adapter Service	Microsoft Corporation	3
alg.exe	2800		Application Layer Gateway Service	Microsoft Corporation	6
svchost.exe	2280		Generic Host Process for Win32 Services	Microsoft Corporation	8
usnsvc.exe	1128		Messenger Sharing USN Journal Reader Service	Microsoft Corporation	5
lsass.exe	1940		LSA Shell (Export Version)	Microsoft Corporation	22
explorer.exe					

CPU Usage: 6.06% Commit Charge: 14.78% Processes: 64



Architecture – System Processes



Process Explorer - Sysinternals: www.sysinternals.com [HAWK\no]

File Options View Process Find Users Help

Process	PID	CPU	Description	Company
System Idle Process	0	93.94		
Interrupts	n/a	0.76	Hardware Interrupts	
DPCs	n/a	1.52	Deferred Procedure Calls	
System	4	0.76		
smss.exe	1724		Windows NT Session Manager	Microsoft Corporation
csrss.exe	1852	1.52	Client Server Runtime Process	Microsoft Corporation
winlogon.exe	1884		Windows NT Logon Application	Microsoft Corporation
services.exe	1928		Services and Controller app	Microsoft Corporation
svchost.exe	236		Generic Host Process for Win32 Services	Microsoft Corporation
svchost.exe	312		Generic Host Process for Win32 Services	Microsoft Corporation
svchost.exe	1196		Generic Host Process for Win32 Services	Microsoft Corporation
svchost.exe	1484		Generic Host Process for Win32 Services	Microsoft Corporation
svchost.exe	740		Generic Host Process for Win32 Services	Microsoft Corporation
spoolsv.exe	1260		Spooler SubSystem App	Microsoft Corporation
agrsmsvc.exe	548		Agere Soft Modem Call Progress Service	Agere Systems
avgwdsvc.exe	564		AVG Watchdog Service	AVG Technologies CZ, s.r.o.
svchost.exe	576		Generic Host Process for Win32 Services	Microsoft Corporation
qs.exe	880		Java(TM) Quick Starter Service	Sun Microsystems, Inc.
mdm.exe	972		Machine Debug Manager	Microsoft Corporation
nvsvc32.exe	1024	0.76	NVIDIA Driver Helper Service, Version 169.09	NVIDIA Corporation
PSIService.exe	528		nTitles PSIService	
svchost.exe	1664		Generic Host Process for Win32 Services	Microsoft Corporation
wmiapsrv.exe	3464		WMI Performance Adapter Service	Microsoft Corporation
alg.exe	2800		Application Layer Gateway Service	Microsoft Corporation
svchost.exe	2280		Generic Host Process for Win32 Services	Microsoft Corporation
usnsvc.exe	1128		Messenger Sharing USN Journal Reader Service	Microsoft Corporation
lsass.exe	1940		LSA Shell (Export Version)	Microsoft Corporation
explorer.exe				

CPU Usage: 6.06% Commit Charge: 14.78% Processes: 64

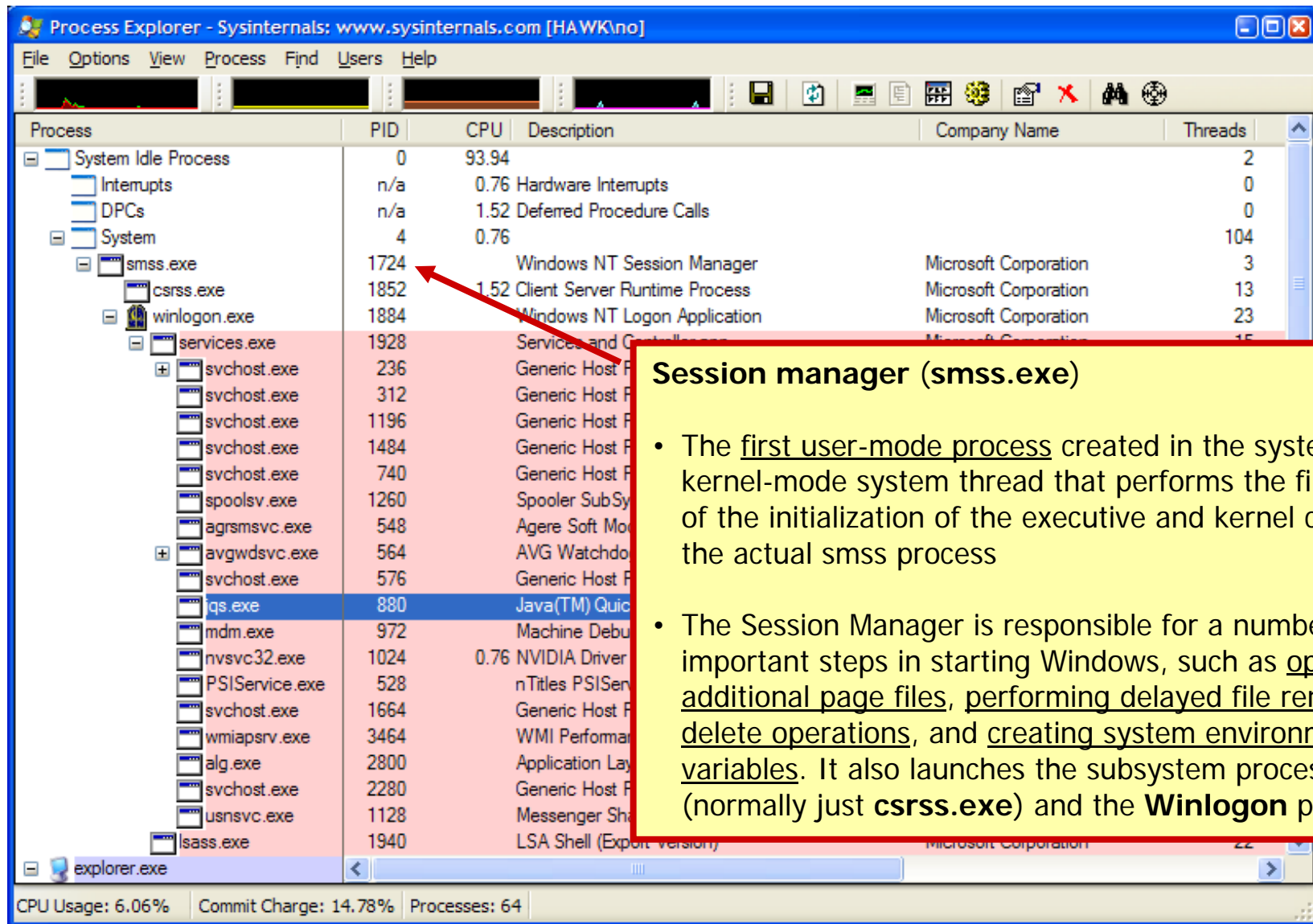
Idle process

contains one thread per CPU to account for idle CPU time

System process is the home for a special kind of thread that runs only in kernel mode - a kernel-mode system thread



Architecture – System Processes



Process Explorer - Sysinternals: www.sysinternals.com [HAWK\no]

Process	PID	CPU	Description	Company Name	Threads
System Idle Process	0	93.94			2
Interrupts	n/a	0.76	Hardware Interrupts		0
DPCs	n/a	1.52	Deferred Procedure Calls		0
System	4	0.76			104
smss.exe	1724		Windows NT Session Manager	Microsoft Corporation	3
csrss.exe	1852	1.52	Client Server Runtime Process	Microsoft Corporation	13
winlogon.exe	1884		Windows NT Logon Application	Microsoft Corporation	23
services.exe	1928		Services and Controller	Microsoft Corporation	15
svchost.exe	236		Generic Host Process		
svchost.exe	312		Generic Host Process		
svchost.exe	1196		Generic Host Process		
svchost.exe	1484		Generic Host Process		
svchost.exe	740		Generic Host Process		
spoolsv.exe	1260		Spooler SubSystem		
agrsmsvc.exe	548		Agere Soft Modem		
avgwdsvc.exe	564		AVG Watchdog Service		
svchost.exe	576		Generic Host Process		
java.exe	880		Java(TM) Quick		
mdm.exe	972		Machine Debug		
nvsvc32.exe	1024	0.76	NVIDIA Driver		
PSIServ.exe	528		nTitles PSIServ		
svchost.exe	1664		Generic Host Process		
wmiapsrv.exe	3464		WMI Performance		
alg.exe	2800		Application Layer		
svchost.exe	2280		Generic Host Process		
usnsvc.exe	1128		Messenger Service		
lsass.exe	1940		LSA Shell (Export version)	Microsoft Corporation	22
explorer.exe					

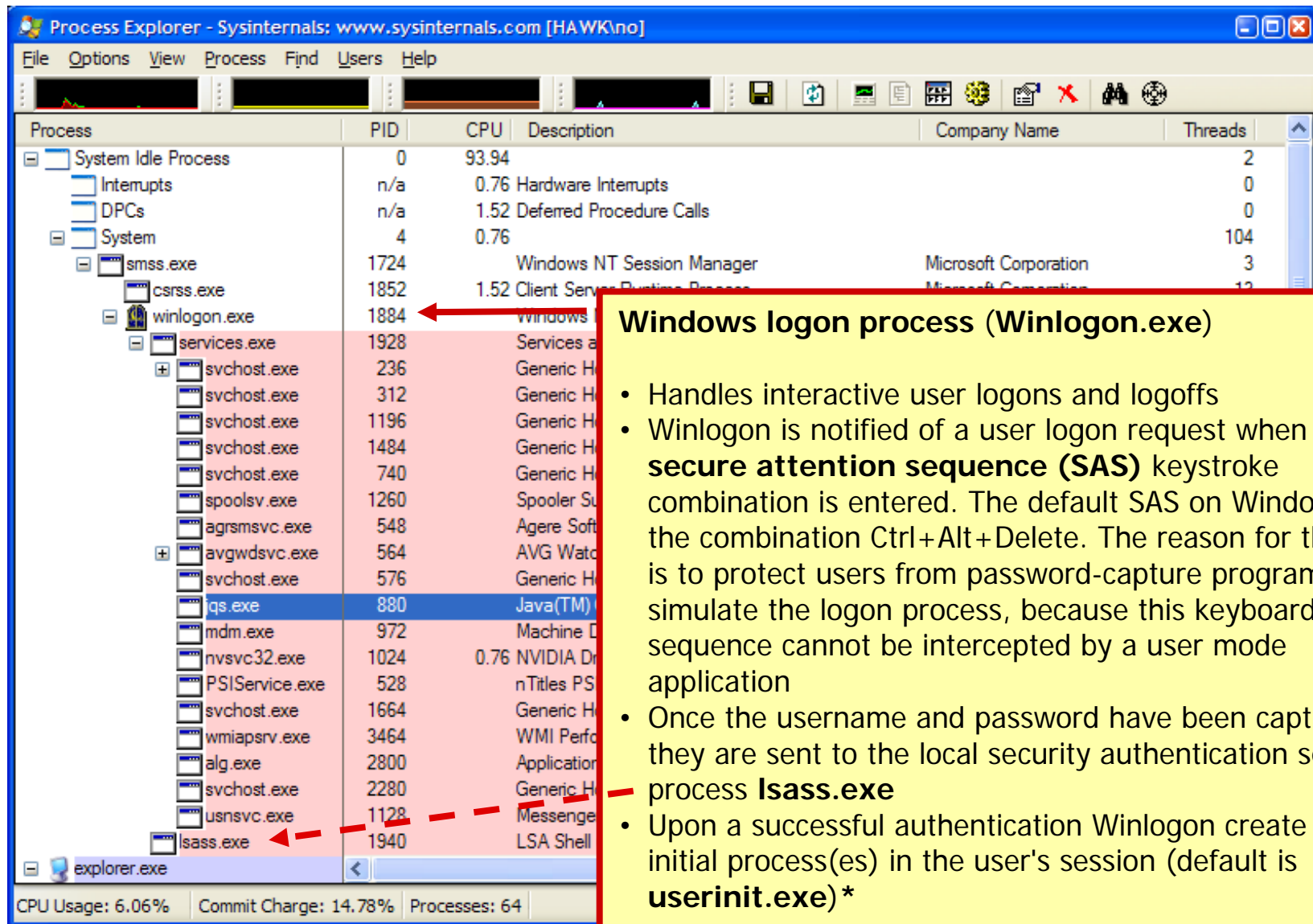
CPU Usage: 6.06% Commit Charge: 14.78% Processes: 64

Session manager (smss.exe)

- The first user-mode process created in the system. The kernel-mode system thread that performs the final phase of the initialization of the executive and kernel creates the actual smss process
- The Session Manager is responsible for a number of important steps in starting Windows, such as opening additional page files, performing delayed file rename and delete operations, and creating system environment variables. It also launches the subsystem processes (normally just **csrss.exe**) and the **Winlogon** process



Architecture – System Processes



Process Explorer - Sysinternals: www.sysinternals.com [HAWK\no]

Process	PID	CPU	Description	Company Name	Threads
System Idle Process	0	93.94			2
Interrupts	n/a	0.76	Hardware Interrupts		0
DPCs	n/a	1.52	Deferred Procedure Calls		0
System	4	0.76			104
smss.exe	1724		Windows NT Session Manager	Microsoft Corporation	3
csrss.exe	1852	1.52	Client Server Runtime Process	Microsoft Corporation	12
winlogon.exe	1884		Windows logon process	Microsoft Corporation	12
services.exe	1928		Services	Microsoft Corporation	12
svchost.exe	236		Generic Host Process for Windows Services	Microsoft Corporation	12
svchost.exe	312		Generic Host Process for Windows Services	Microsoft Corporation	12
svchost.exe	1196		Generic Host Process for Windows Services	Microsoft Corporation	12
svchost.exe	1484		Generic Host Process for Windows Services	Microsoft Corporation	12
svchost.exe	740		Generic Host Process for Windows Services	Microsoft Corporation	12
spoolsv.exe	1260		Spooler Service	Microsoft Corporation	12
agrsmsvc.exe	548		Agere Software	Agere Systems, Inc.	12
avgwdsvc.exe	564		AVG Watchdog Service	AVG Technologies	12
svchost.exe	576		Generic Host Process for Windows Services	Microsoft Corporation	12
java.exe	880		Java(TM) Virtual Machine	Oracle Corporation	12
mdm.exe	972		Machine Domain Monitor	Microsoft Corporation	12
nvsvc32.exe	1024	0.76	NVIDIA Display Driver Service	NVIDIA Corporation	12
PSIService.exe	528		nTitles PSIService	nTitles	12
svchost.exe	1664		Generic Host Process for Windows Services	Microsoft Corporation	12
wmiapsrv.exe	3464		WMI Performance	Microsoft Corporation	12
alg.exe	2800		Application Layer Gateway Service	Microsoft Corporation	12
svchost.exe	2280		Generic Host Process for Windows Services	Microsoft Corporation	12
usnsvc.exe	1128		Messenger Service	Microsoft Corporation	12
lsass.exe	1940		LSA Shell	Microsoft Corporation	12
explorer.exe			Windows Explorer	Microsoft Corporation	12

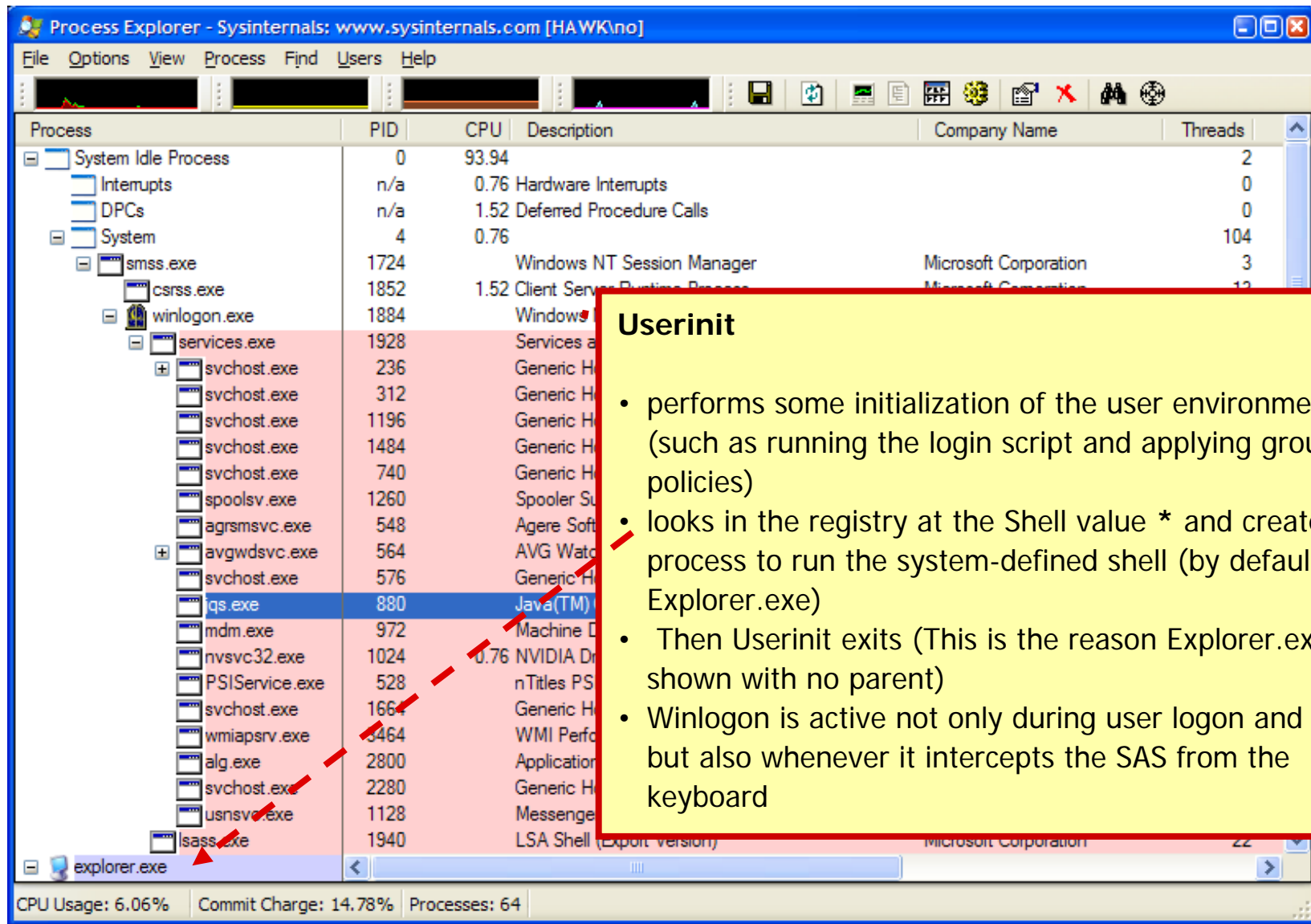
CPU Usage: 6.06% Commit Charge: 14.78% Processes: 64

Windows logon process (Winlogon.exe)

- Handles interactive user logons and logoffs
- Winlogon is notified of a user logon request when the **secure attention sequence (SAS)** keystroke combination is entered. The default SAS on Windows is the combination Ctrl+Alt+Delete. The reason for the SAS is to protect users from password-capture programs that simulate the logon process, because this keyboard sequence cannot be intercepted by a user mode application
- Once the username and password have been captured, they are sent to the local security authentication server process **lsass.exe**
- Upon a successful authentication Winlogon create the initial process(es) in the user's session (default is **userinit.exe**)*



Architecture – System Processes



Process Explorer - Sysinternals: www.sysinternals.com [HAWK\no]

Process	PID	CPU	Description	Company Name	Threads
System Idle Process	0	93.94			2
Interrupts	n/a	0.76	Hardware Interrupts		0
DPCs	n/a	1.52	Deferred Procedure Calls		0
System	4	0.76			104
smss.exe	1724		Windows NT Session Manager	Microsoft Corporation	3
csrss.exe	1852	1.52	Client Server Runtime Process	Microsoft Corporation	12
winlogon.exe	1884		Windows		
services.exe	1928		Services		
svchost.exe	236		Generic H		
svchost.exe	312		Generic H		
svchost.exe	1196		Generic H		
svchost.exe	1484		Generic H		
svchost.exe	740		Generic H		
spoolsv.exe	1260		Spooler S		
agrsmsvc.exe	548		Agere Soft		
avgwdsvc.exe	564		AVG Wato		
svchost.exe	576		Generic H		
qs.exe	880		Java(TM)		
mdm.exe	972		Machine D		
nvsvc32.exe	1024	0.76	NVIDIA Dr		
PSIService.exe	528		nTitles PS		
svchost.exe	1664		Generic H		
wmiapsrv.exe	3464		WMI Perfo		
alg.exe	2800		Application		
svchost.exe	2280		Generic H		
usnsvc.exe	1128		Messenge		
lsass.exe	1940		LSA Shell (Export version)	Microsoft Corporation	22
explorer.exe					

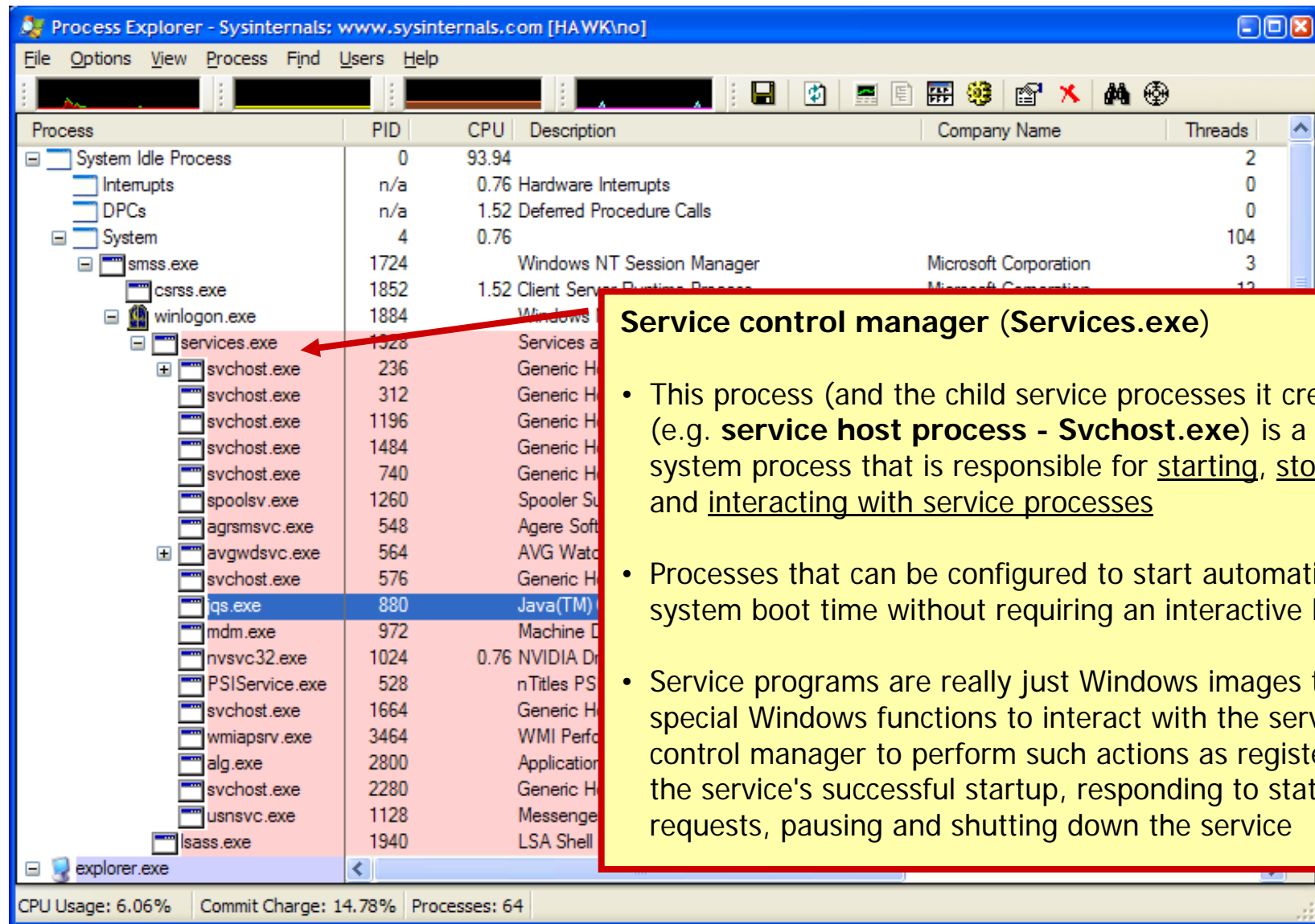
CPU Usage: 6.06% Commit Charge: 14.78% Processes: 64

Userinit

- performs some initialization of the user environment (such as running the login script and applying group policies)
- looks in the registry at the Shell value * and creates a process to run the system-defined shell (by default, Explorer.exe)
- Then Userinit exits (This is the reason Explorer.exe is shown with no parent)
- Winlogon is active not only during user logon and logoff but also whenever it intercepts the SAS from the keyboard



Architecture – System Processes



Process Explorer - Sysinternals: www.sysinternals.com [HAWK\no]

File Options View Process Find Users Help

Process	PID	CPU	Description	Company Name	Threads
System Idle Process	0	93.94			2
Interrupts	n/a	0.76	Hardware Interrupts		0
DPCs	n/a	1.52	Deferred Procedure Calls		0
System	4	0.76			104
smss.exe	1724		Windows NT Session Manager	Microsoft Corporation	3
csrss.exe	1852	1.52	Client Server Runtime Process	Microsoft Corporation	12
winlogon.exe	1884		Windows Logon Process	Microsoft Corporation	12
services.exe	1928		Services Control Manager	Microsoft Corporation	12
svchost.exe	236		Generic Host Process for Windows Services	Microsoft Corporation	12
svchost.exe	312		Generic Host Process for Windows Services	Microsoft Corporation	12
svchost.exe	1196		Generic Host Process for Windows Services	Microsoft Corporation	12
svchost.exe	1484		Generic Host Process for Windows Services	Microsoft Corporation	12
svchost.exe	740		Generic Host Process for Windows Services	Microsoft Corporation	12
spoolsv.exe	1260		Spooler Subsystem	Microsoft Corporation	12
agrsmsvc.exe	548		Agere Software	Agere Systems, Inc.	12
avgwdsvc.exe	564		AVG Watchdog Service	AVG Technologies	12
svchost.exe	576		Generic Host Process for Windows Services	Microsoft Corporation	12
java.exe	880		Java(TM) Virtual Machine	Oracle Corporation	12
mdm.exe	972		Machine Domain Manager	Microsoft Corporation	12
nvsvc32.exe	1024	0.76	NVIDIA Display Driver Service	NVIDIA Corporation	12
PSIService.exe	528		nTitles PSIService	nTitles	12
svchost.exe	1664		Generic Host Process for Windows Services	Microsoft Corporation	12
wmiapsrv.exe	3464		WMI Performance	Microsoft Corporation	12
alg.exe	2800		Application Layer Gateway	Microsoft Corporation	12
svchost.exe	2280		Generic Host Process for Windows Services	Microsoft Corporation	12
usnsvc.exe	1128		Messenger Service	Microsoft Corporation	12
lsass.exe	1940		LSA Shell	Microsoft Corporation	12

explorer.exe

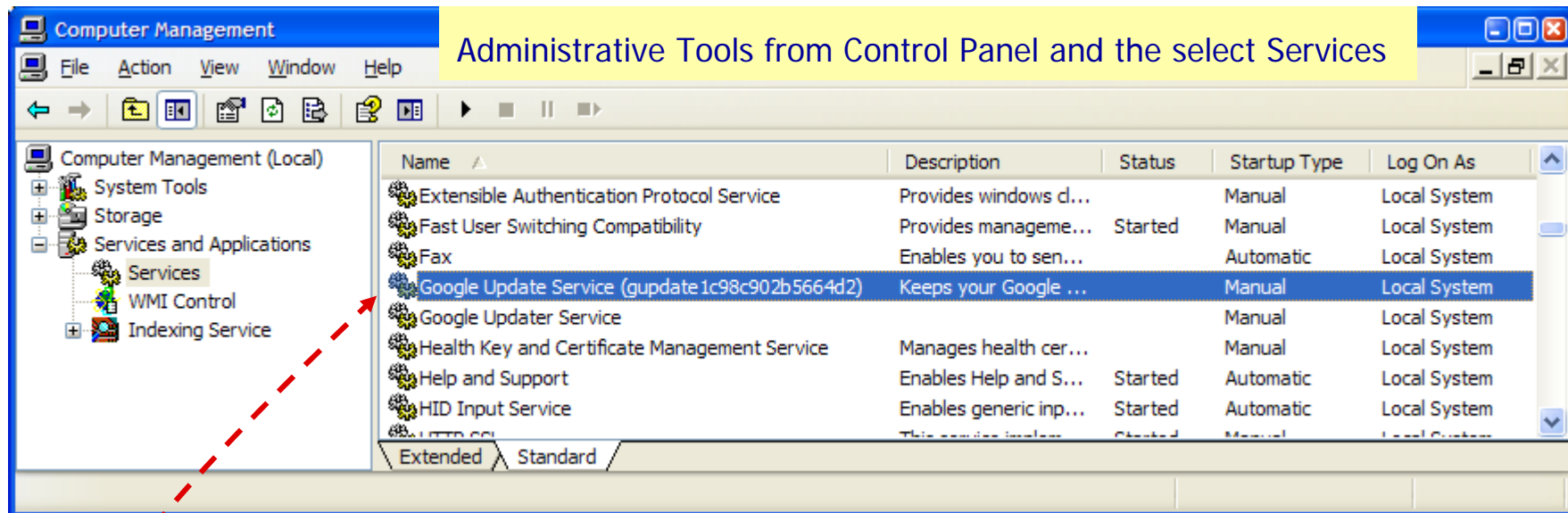
CPU Usage: 6.06% Commit Charge: 14.78% Processes: 64

Service control manager (Services.exe)

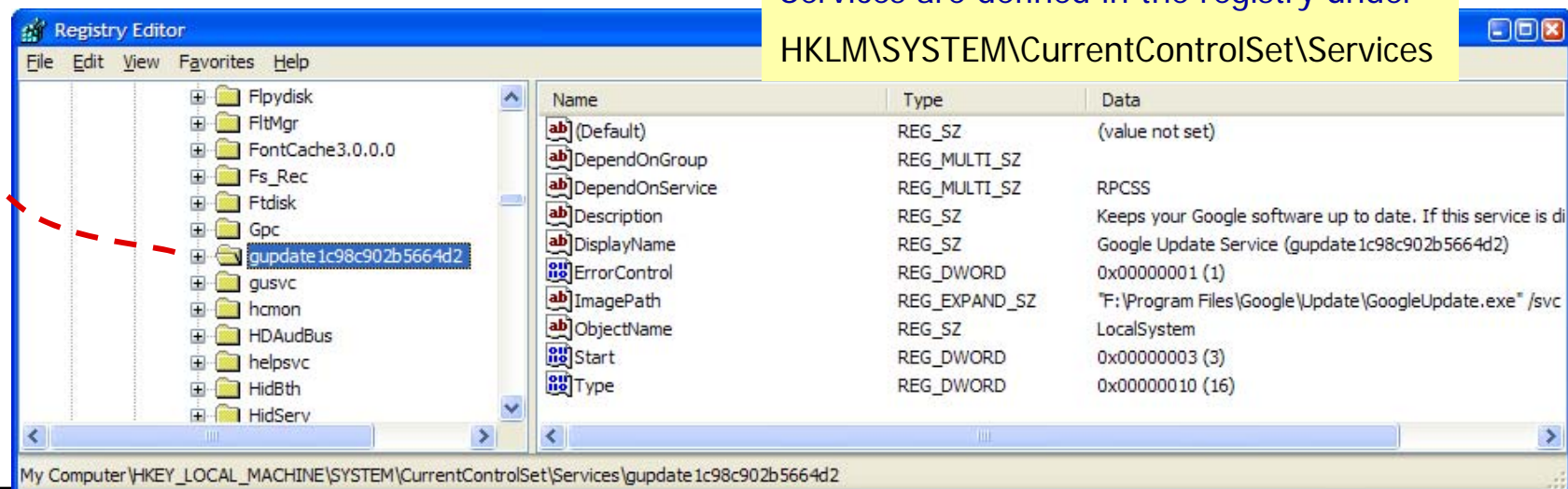
- This process (and the child service processes it creates (e.g. **service host process - Svchost.exe**) is a special system process that is responsible for starting, stopping, and interacting with service processes
- Processes that can be configured to start automatically at system boot time without requiring an interactive logon
- Service programs are really just Windows images that call special Windows functions to interact with the service control manager to perform such actions as registering the service's successful startup, responding to status requests, pausing and shutting down the service



Services – listing installed services



Services are defined in the registry under
HKLM\SYSTEM\CurrentControlSet\Services



Services – listing installed services

The screenshot shows the Windows 'Computer Management' console. In the left-hand tree, 'Services' is selected under 'Services and Applications'. The main pane displays a list of installed services. The 'Google Update Service (gupdate1c98c902b5664d2)' is highlighted. A context menu is open, and the 'Properties' option is selected, opening the 'Google Update Service (gupdate1c98c902b5664d2) Properties' dialog box. The 'General' tab is active, showing the service name, display name, description, path to executable, startup type (Manual), and service status (Stopped). Buttons for Start, Stop, Pause, and Resume are visible. The Start parameters field is empty.

Name	Description	Status	Startup Type	Log On As
Extensible Authentication Protocol Service	Provides windows d...		Manual	Local System
Fast User Switching Compatibility	Provides managem...	Started	Manual	Local System
Fax	Enables you to sen...		Automatic	Local System
Google Update Service (gupdate1c98c902b5664d2)	Keeps your Google ...		Manual	Local System
Google Updater Service				
Health Key and Certificate Management S				
Help and Support				
HID Input Service				

Detailed properties about a service right clicked on the service and select properties

Service name: gupdate1c98c902b5664d2

Display name: Google Update Service (gupdate1c98c902b5664d2)

Description: Keeps your Google software up to date. If this service is disabled or stopped, your Google software

Path to executable: "F:\Program Files\Google\Update\GoogleUpdate.exe" /svc

Startup type: Manual

Service status: Stopped

Start parameters:



Services – Process Explorer

Process Explorer - Sysinternals: www.sysinternals.com [HAWK\no]

File Options View Process Find Users Help

Process	PID	CPU	CSwitch Delta	Description
winlogon.exe	1884			Windows NT
services.exe	1928			55 Services and C
svchost.exe	236			1 Generic Host P
svchost.exe	312			1 Generic Host P
svchost.exe	1196			3 Generic Host P
svchost.exe	1484			Generic Host P
svchost.exe	740			Generic Host P
spoolsv.exe	1260			Spooler SubSys
agrsmvc.exe	548			Agere Soft Mod
avgwdsvc.exe	564			77 AVG Watchdog
svchost.exe	576			Generic Host P
iqs.exe	880			5 Java(TM) Quick
mdm.exe	972			16 Machine Debug
nvsvc32.exe				

CPU Usage: 4.55% Commit Charge: 14.79% Processes: 70

svchost.exe: 1196 Properties

Threads TCP/IP Security Environment Strings
Image Performance Performance Graph Services

Services registered in this process:

Service	Display Name
AudioSrv	Windows Audio
BITS	Background Intelligent Transfer Service
Browser	Computer Browser
CryptSvc	Cryptographic Services
Dhcp	DHCP Client
dmserver	Logical Disk Manager
ERSvc	Error Reporting Service
EventSystem	COM+ Event System
FastUserSwitchingCompatibility	Fast User Switching Compatibility
helpsvc	Help and Support
HidServ	HID Input Service
LanmanServer	Server

Manages audio devices for Windows-based programs. If this service is stopped, audio devices and effects will not function properly. If this service is disabled, any services that explicitly depend on it will fail to start.

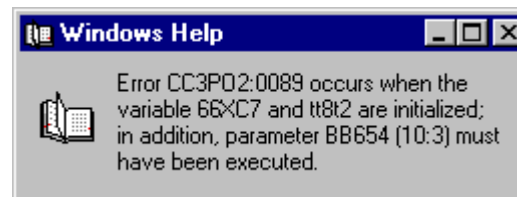
Permissions Stop Pause Resume

OK Cancel

Double-click on a service-hosting process



WIN 32

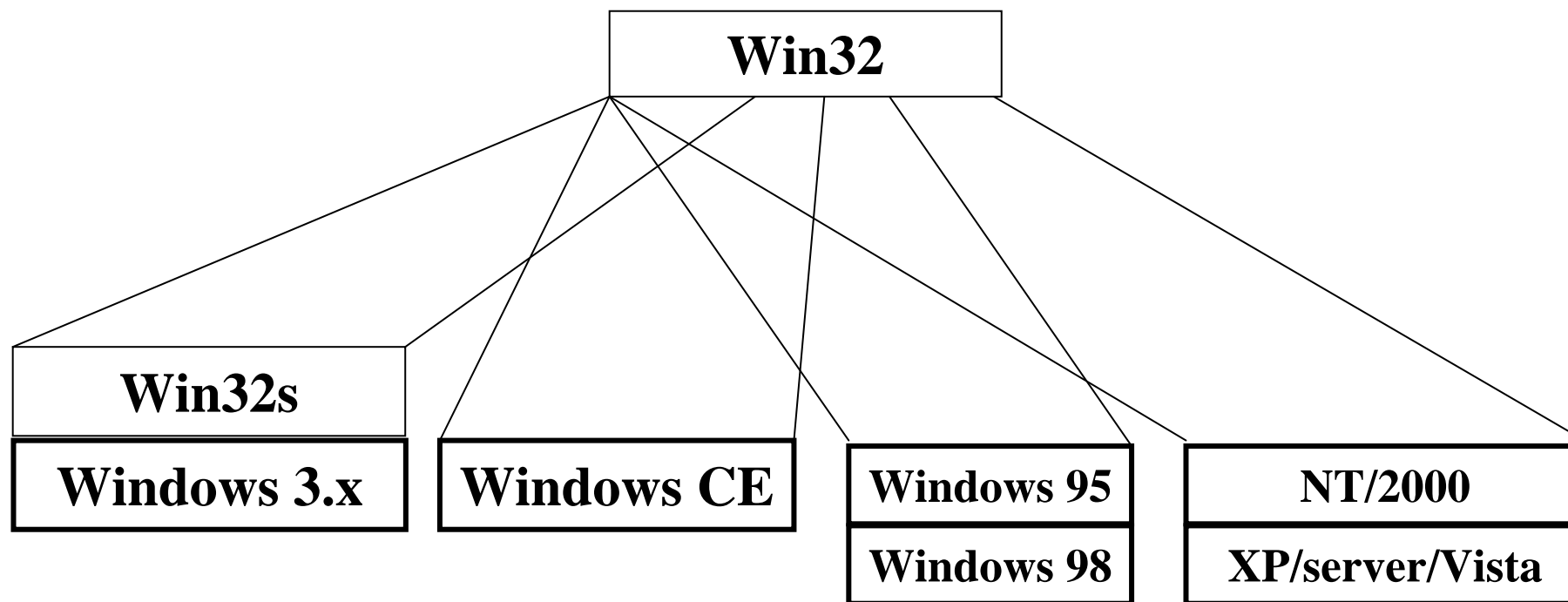


Versões do Windows

- **Versões “mais ligeiras” do Windows (“*desktop use*”)**
 - Windows 1.0-1985, 2.0-1987 , 3.0-1990, 3.1-1992
 - Windows for *Workgroups* 3.11, 1994
 - Windows 95, Windows 98, Windows ME in 2000
- **Versões do Windows baseadas no NT (*New Technology*)**
 - **Windows NT 3.1**, *Workstation, Advanced Server*, 1993,
 - **Windows NT 3.5**, *Workstation, Server*, 1994
 - **Windows NT 3.51** , *Workstation, Server*, 1995
 - **Windows NT 4.0** *Workstation, Server, Server Enterprise*, 1996
 - **Windows 2000** *Professional, Server, Advanced Server e Data Center Server editions*, 2000
 - **Windows XP** *Home, Professional, Media Center e 64-bits editions*, 2001
 - **Windows Server 2003 (.net)**, *Standard, Enterprise, Datacenter, Web e Small Business editions*, 2003
 - **Windows Vista**, *Starter, Home Basic, Home Premium, Professional, Small Business, Enterprise, Ultimate (x64 editions will be available too)*, ~2006
 - Windows Longhorn Server (codename), ~2007
- **Windows CE 1.0-1996, 2.0-1997, 3.0-2000, 4.0-2002, 5.0-2004, 6.0-2006**



Implementações da Win32



Win32s - é uma extensão à API de 16 bits do Windows 3.x para correr aplicações de 32 bits.

Windows CE - implementa parte da Win32 API mas com muitas limitações visto que deverá correr em máquinas sem disco e com pouca memória.

Windows 95 - é uma implementação da Win32 API mas com limitações visto que deverá correr em máquinas com 386 e 4Mb.

Windows NT/2000/XP/Vista - implementam a totalidade da API Win32.



Caracteres e *Strings*

Representação de texto

Código ASCII

- **American Standard Code for Information Interchange (ANSI X3.4-1986)**
- **Código de representação de caracteres a 8 bits apenas 256 símbolos**
- **Então como representar todos os caracteres e símbolos de todo o mundo incluindo as escritas modernas mas também antigas, por exemplo o chinês, japonês, árabe, hebreu, etc**
 - Solução 1: Representar alguns caracteres a 16 bits (DBCS – *Double Byte Character Strings*)
 - Solução 2: Codificação UNICODE



Código ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Source: www.asciitable.com



Código ASCII estendido

128	Ç	144	É	160	á	176	░	193	⌞	209	⌚	225	Β	241	±
129	ü	145	æ	161	í	177	▒	194	⌟	210	⌛	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	⌠	211	⌜	227	π	243	≤
131	â	147	ô	163	ú	179		196	⌡	212	⌝	228	Σ	244	∫
132	ä	148	ö	164	ñ	180	⌢	197	⌣	213	⌞	229	σ	245	∫
133	à	149	ò	165	Ñ	181	⌣	198	⌤	214	⌟	230	μ	246	÷
134	â	150	û	166	²	182	⌤	199	⌥	215	⌠	231	τ	247	≈
135	ç	151	ù	167	°	183	⌥	200	⌦	216	⌡	232	Φ	248	°
136	ê	152	—	168	¿	184	⌦	201	⌧	217	⌢	233	Θ	249	·
137	ë	153	Ö	169	—	185	⌧	202	⌨	218	⌣	234	Ω	250	·
138	è	154	Ü	170	¬	186	⌨	203	〈	219	■	235	δ	251	√
139	ï	156	£	171	½	187	〈	204	〉	220	■	236	∞	252	—
140	î	157	¥	172	¼	188	〉	205	=	221	■	237	φ	253	²
141	ï	158	—	173	¡	189	〉	206	⌫	222	■	238	ε	254	■
142	Ä	159	ƒ	174	«	190	⌫	207	⌬	223	■	239	∩	255	
143	Å	192	ℒ	175	»	191	⌫	208	⌭	224	α	240	≡		

Source: www.asciitable.com



Unicode

- Proposto inicialmente pela Apple e Xerox em 1988
- Em 1991 foi criado um consórcio para desenvolver e promover o Unicode. No consórcio estavam empresas como Apple, Compaq, Hewlett-Packard, IBM, Microsoft, Oracle, Silicon Graphics, Sybase, Unisys e Xerox (lista completa em <http://www.unicode.org>)
- Unicode define 1.114.112 *code points* de 0hex até 10FFFFhex
- É habitual indicar um Unicode *code point* escrevendo U+<número hexadecimal>
- capacidade para representar os caracteres das várias línguas e outros símbolos, como por exemplo símbolos matemáticos e técnicos

Plane	Range	Description	Abbreviation
0	0000–FFFF	Basic Multilingual Plane	BMP
1	10000–1FFFF	Supplementary Multilingual Plane	SMP
2	20000–2FFFF	Supplementary Ideographic Plane	SIP
3 to 13	30000–DFFFF	currently unassigned	
14	E0000–EFFFF	Supplementary Special-purpose Plane	SSP
15	F0000–FFFFF	Supplementary Private Use Area-A	
16	100000–10FFFF	Supplementary Private Use Area-B	



Unicode

- **Estão definidos standards para a representação dos Unicode (Unicode Transformation Format - UTF)**
 - UTF-8
 - Codificação a 1 byte, a 2 bytes, a 3 bytes ou a 4 bytes
 - UTF-16
 - codificação a 2 bytes
 - 16 bits não é suficiente para todos os caracteres Unicode nessas situações o UTF-16 usa *surrogates* (uma forma de utilizar 4 bytes na codificação).
 - O Windows utiliza este tipo de codificação
 - UTF-32
 - Codifica todos os caracteres a 4 bytes



Unicode

Código a 16 bits

Partes mostradas:

Basic Latin

Latin Supplement

Latin Extended-A

Greek

Arabic

Musical Symbols

Arrows

Geometric Shapes

Source:

<http://www.unicode.org/Public/5.0.0/charts/CodeCharts.pdf>

Basic Latin

	000	001	002	003	004	005	006	007
0	NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	~ 0060	p 0070
1	SOH 0011	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
2	STX 0012	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
3	ETX 0013	DC3 0013	# 0023	3 0033	C 0043	S 0053	c 0063	s 0073
4	EOT 0014	DC4 0014	\$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
5	ENG 0015	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
6	ACK 0016	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
7	BEL 0017	ETB 0017	' 0027	7 0037	G 0047	W 0057	g 0067	w 0077
8	BS 0018	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
9	HT 0019	EM 0019) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
A	LF 001A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
B	VT 001B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
C	FF 001C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	 007C
D	CR 001D	GS 001D	- 002D	= 003D	M 004D] 005D	m 006D	} 007D
E	SO 001E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
F	SI 001F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

Latin Supplement

	008	009	00A	00B	00C	00D	00E	00F
0	XXX 0000	DCS 0000	NB SP 0000	° 0000	À 00C0	Đ 00D0	à 00E0	đ 00F0
1	XXX 0001	PU1 0001	¡ 00A1	± 00B1	Á 00C1	Ñ 00D1	á 00E1	ñ 00F1
2	BPH 0002	PU2 0002	¢ 00A2	² 00B2	Â 00C2	Ò 00D2	â 00E2	ò 00F2
3	NBH 0003	STS 0003	£ 00A3	³ 00B3	Ã 00C3	Ó 00D3	ã 00E3	ó 00F3
4	XXX 0004	CCH 0004	¤ 00A4	´ 00B4	Ä 00C4	Ô 00D4	ä 00E4	ô 00F4
5	NEL 0005	MW 0005	¥ 00A5	µ 00B5	Å 00C5	Õ 00D5	å 00E5	õ 00F5
6	SSA 0006	SPA 0006	¦ 00A6	¶ 00B6	Æ 00C6	Ö 00D6	æ 00E6	ö 00F6
7	ESA 0007	EPA 0007	§ 00A7	· 00B7	Ç 00C7	× 00D7	ç 00E7	÷ 00F7
8	HTS 0008	SOS 0008	¨ 00A8	¸ 00B8	È 00C8	Ø 00D8	è 00E8	ø 00F8
9	HTJ 0009	XXX 0009	© 00A9	¹ 00B9	É 00C9	Ù 00D9	é 00E9	ù 00F9
A	VTS 000A	SCI 000A	ª 00AA	º 00BA	Ê 00CA	Ú 00DA	ê 00EA	ú 00FA
B	PLD 000B	CSI 000B	« 00AB	» 00BB	Ë 00CB	Û 00DB	ë 00EB	û 00FB
C	PLU 000C	ST 000C	¼ 00AC	¼ 00BC	Ì 00CC	Ü 00DC	ì 00EC	ü 00FC
D	RI 000D	OSC 000D	½ 00AD	½ 00BD	Í 00CD	Ý 00DD	í 00ED	ý 00FD
E	SS2 000E	PM 000E	¾ 00AE	¾ 00BE	Î 00CE	Þ 00DE	î 00EE	þ 00FE
F	SS3 000F	APC 000F	¿ 00AF	¿ 00BF	Ï 00CF	ß 00DF	ï 00EF	ÿ 00FF



Unicode

	010	011	012	013	014	015	016	017
0	Ā	Ð	Ġ	İ	ı	Ō	Š	Ů
1	ā	ð	ġ	ı	Ĺ	ō	š	ů
2	Ǻ	Ē	Ģ	Ĳ	ı	Œ	Ŧ	Ū
3	ǻ	ē	ĝ	ij	Ŋ	œ	ţ	ų
4	Ą	Ĕ	Ĥ	Ĵ	ń	Ŕ	Ť	Ŵ
5	ą	ĕ	ĥ	ĵ	ņ	ŕ	ť	ŵ
6	Ć	Ė	Ħ	Ķ	ņ	Ŗ	Ŧ	Ŷ
7	ć	ė	ħ	ķ	Ņ	ŗ	ţ	ŷ
8	Ĉ	Ė	Ĭ	κ	ñ	Ř	Ů	Ÿ
9	ĉ	ę	ĩ	Ĺ	ñ	ř	ů	ž
A	Ċ	Ė	Ĭ	Í	Ŋ	Ŝ	Ū	ž
B	ċ	ę	ĩ	Ĺ	ŋ	ś	ū	Ž
C	Č	Ġ	Ĭ	ı	Ō	Ŝ	Ů	ž
D	č	ġ	ĩ	Ĺ	ō	ŝ	ů	Ž
E	Ď	Ģ	ı	Ŗ	Ō	Ş	Ů	ž
F	ď	ĝ	ı	Ĺ	ō	ş	ů	ƒ

Latin Extended-A

Greek

	037	038	039	03A	03B	03C	03D	03E	03F
0			ı	Π	ı	π	θ	η	κ
1			Α	Ρ	α	ρ	θ	η	κ
2			Β		β	ς	Υ	ϖ	ϗ
3			Γ	Σ	γ	σ	Υ	ϖ	ϗ
4			Δ	Τ	δ	τ	Υ	ϖ	ϗ
5			Ε	Υ	ε	υ	φ	ϗ	€
6			Α	Ζ	Φ	ζ	φ	ϗ	€
7			Η	Χ	η	χ	ζ	€	
8			Ε	Θ	Ψ	θ	ψ	Q	€
9			Η	Ι	Ω	ι	ω	Q	€
A			Ι	Κ	İ	κ	ı	Σ	Χ
B			Λ	Υ	λ	υ	ς	τ	
C			Ο	Μ	α	μ	ο	Φ	Θ
D			Ν	Ε	ν	ύ	φ	σ	
E			Υ	Ξ	ή	ξ	ώ	ζ	τ
F			Ω	Ο	ı	ο		ζ	τ

Arabic

	060	061	062	063	064	065	066	067
0				ذ	-	ؤ	٠	١
1			ء	ر	ف	٠	١	أ
2			آ	ز	ق	٠	٢	أ
3			أ	س	ك	٠	٣	إ
4			ؤ	ش	ل	٠	٤	٠
5			إ	ص	م	٠	٥	أ
6			ئ	ض	ن		٦	و
7			ا	ط	ه		٧	ؤ
8			ب	ظ	و		٨	ئ
9			ة	ع	ى		٩	ث
A			ت	غ	ي		٪	ث
B			؛	ث	٠		ر	پ
C			ج	٠			٠	ت
D			ح	٠			*	ت
E			خ	٠			ر	پ
F			؟	د	٠		و	ث



Unicode

Musical Symbols

	1D10	1D11	1D12	1D13	1D14	1D15	1D16	1D17
0								
1								
2								
3								
4								
5								
6								
7								
8								
9								
A								
B								
C								
D								
E								
F								

Arrows

	219	21A	21B	21C	21D	21E	21F
0							
1							
2							
3							
4							
5							
6							
7							
8							
9							
A							
B							
C							
D							
E							
F							

Geometric Shapes

	25A	25B	25C	25D	25E	25F
0						
1						
2						
3						
4						
5						
6						
7						
8						
9						
A						
B						
C						
D						
E						
F						



ASCII/Unicode na Win32

- **O sistema operativo Windows 95 e suas evoluções, Windows 98 e ME, não suportam Unicode nativo**
 - Se um aplicação utilizar código Unicode quando faz uma chamada de sistema as strings são em primeiro lugar convertidas para ASCII, é feita a chamada de sistemas e as strings de resposta são convertidas novamente para Unicode
- **O Windows NT e suas evoluções, 2000, XP, 2003 (.NET), Vista, suportam nativamente o Unicode continuando, no entanto, a suportar o ASCII**
 - Se um aplicação utilizar código ASCII quando faz uma chamada de sistema as strings são em primeiro lugar convertidas para Unicode, é feita a chamada de sistemas e as strings de resposta são convertidas novamente para ASCII
- **O Windows CE só suporta Unicode**
- **A concepção de código na disciplina de S.O. assume o compromisso de fazer código que possa ser compilado nas várias famílias do sistema operativo Windows**



Caracteres a 8 e 16 bits no ANSI C

- **Declaração de caracteres a 8 bits:**

```
char c      = 's';  
char *str   = "Ola";  
char a[]    = "ola";
```

- **Declaração de caracteres a 16 bits (wide char):**

```
wchar_t c      = 's';           // wchar_t definido em wchar.h  
wchar_t *str    = L"Ola";  
wchar_t a[]     = L"Ola";
```

No Windows um `wchar_t` representa um caracter Unicode de 16 bits no formato UTF-16

- **A biblioteca de Run-Time C contém versões de funções para caracteres a 8 e 16 bits, exemplos:**

```
8bits  <-> 16bits  
strlen <-> wcslen  
strcpy <-> wcscpy  
strcmp <-> wcscmp  
printf <-> wprintf  
scanf  <-> wscanf
```

Vai-se utilizar compilação
condicional para se obter as
versões 8 e 16 bits



ASCII/Unicode - RT/C

■ Utilização do tipo genérico de caracter TCHAR

- Definido no ficheiro de definições tchar.h (não compatível com ANSI C)
- As definições do ficheiro tchar.h são expandias em função da definição do símbolo _UNICODE
- A definição do símbolo _UNICODE deve aparecer antes dos includes ou através das opções do compilador

```
#define _UNICODE // para c run-time header files
```

■ Símbolo _UNICODE definido

- Sim: uso das versões com suporte de caracteres wide (16 bits)
- Não: uso das versões com suporte de caracteres single (8 bits)

■ Macros de utilização condicional

- TCHAR para caracter (expande para wchar_t ou char)
- _TEXT() ou _T para caracteres ou strings (expande para L() ou não)

■ Exemplos:

- `TCHAR c0 = _T('s'), c1 = _TEXT('s');`
- `TCHAR *str0 = _T("Ola");`
- `TCHAR *str1 = _TEXT("Ola");`



ASCII/Unicode - RT/C

■ Macros para as funções da biblioteca ANSI que expandem para as versões de 8 ou 16 bits exemplos:

- `8bits` <-> `16bits` <-> `tchar.h` (condicional)
- `strlen` <-> `wcslen` <-> `_tcslen`
- `printf` <-> `wprintf` <-> `_tprintf`
- `scanf` <-> `wscanf` <-> `_stscanf`

■ Função Main genérica

- `_tmain` expande para
 - `main` (8 bits) ou
 - `wmain` (16bits))

```
#ifdef _UNICODE
#define _tcslen wcslen
#else
#define _tcslen strlen
#endif
```

Nota: a função `wsprintf` não suporta a formatação de *floats*



ASCII/Unicode - Exemplo

```
// ListArgs.cpp : Print the arguments on the console.

#define _UNICODE    // for c run-time header files

#include <tchar.h>    // for _tprintf definition _TEXT
#include <stdio.h>    // for printf and wprintf

int _tmain(int argc, TCHAR* argv[])
{
    _tprintf( _TEXT("Listagem dos argumentos:\n") );
    for(int i=0; i<argc; ++i) {
        _tprintf( _TEXT("Argumento[%d] = %s\n"), i, argv[i]);
    }
    return 0;
}
```



ASCII/Unicode na Win32

- A Microsoft definiu os seus próprios tipos de caracteres, utilizados na Win32 API, para separar dos tipos da linguagem C
- Estas definições encontram-se em WinNT.h

```
typedef char    CHAR;    // An 8-bit character
typedef wchar_t WCHAR;   // A 16-bit character
```

```
// Pointer to 8-bit character(s)
```

```
typedef        CHAR *PCHAR;
typedef        CHAR *PSTR;
typedef CONST CHAR *PCSTR;
```

```
// Pointer to 16-bit character(s)
```

```
typedef        WCHAR *PWCHAR;
typedef        WCHAR *PWSTR;
typedef CONST WCHAR *PCWSTR;
```



ASCII/Unicode na Win32

- A escrita de código genérica é possível através de tipos que expandem para as versões ASCII ou UNICODE em função da definição do símbolo de compilação UNICODE
- Essas definições encontram-se em WinNT.h

```
#ifdef UNICODE
typedef WCHAR      PTCH;
typedef WCHAR      *PTSTR;
typedef CONST WCHAR *PCTSTR;
#define __TEXT(quote) L##quote
#else
typedef CHAR       PTCH;
typedef CHAR       *PTSTR;
typedef CONST CHAR *PCTSTR;
#define __TEXT(quote) quote
#endif
```

```
#define TEXT(quote) __TEXT(quote)
```



ASCII/Unicode na Win32

- A API possui funções que têm como argumento strings. Geralmente, são fornecidas duas versões da mesma função

```
HWND WINAPI CreateWindowExW(  
    DWORD dwExStyle,  
    PCWSTR pClassName,    // A Unicode string  
    PCWSTR pWindowName,   // A Unicode string  
    ...);
```

```
HWND WINAPI CreateWindowExA(  
    DWORD dwExStyle,  
    PCSTR pClassName,     // An ANSI string  
    PCSTR pWindowName,    // An ANSI string  
    ...);
```

- No entanto podemos utilizar uma macro genérica que expande para uma das funções anteriores em função da definição do símbolo UNICODE

```
#ifndef UNICODE  
#define CreateWindowEx CreateWindowExW  
#else  
#define CreateWindowEx CreateWindowExA  
#endif
```



Funções de *string* seguras RT/C

- Muitas das funções de manipulação de strings da biblioteca de run-time C não possuem um argumento para o programador indicar a dimensão máxima do buffer
- Assim as funções não consegue determinar se está a corromper a memória (aceder fora da memória associada ao buffer - *overflow buffer*), e.g `strcat`, `strcpy`, ...
- Este comportamento pode ser explorado por código malicioso (e.g. virus)
- A Microsoft disponibiliza um novo conjunto de funções que substituem as funções originais consideradas inseguras e definidas em `StrSafe.h`
- As novas funções possuem o sufixo `_s`, basicamente possuem um novo argumento indicando a dimensão máxima do buffer e procedem à validação de todos os seus argumentos (apontadores diferente de nulos, ausência de *overflow buffer*)

```
PTSTR _tcscpy (PTSTR strDestination, PCTSTR strSource);  
errno_t _tcscpy_s(PTSTR strDestination,  
                  size_t numberOfCharacters, PCTSTR strSource);
```



Funções de *string* seguras RT/C

- Estão definidas outro conjunto de funções na biblioteca de RT/C que oferecem um melhor controlo nas operações sobre as strings
- Ao contrário das funções anteriores estas deixam no buffer a informação que for possível colocar (*truncation*)
- Existem funções que no seu nome têm:

- Cch (*count of characters*) em que as dimensões são indicadas em caracteres e deve-se usar a macro `_countof()` para obter esse valor

```
HRESULT StringCchCopy(PTSTR pszDest,  
                      size_t cchDest, PCTSTR pszSrc);  
HRESULT StringCchCopyEx(PTSTR pszDest, size_t cchDest,  
                       PCTSTR pszSrc, PTSTR *ppszDestEnd,  
                       size_t *pcchRemaining, DWORD dwFlags);
```

- Cb (*Count of bytes*) em que as dimensões são indicadas em bytes devendo-se utilizar o operador `sizeof`

```
HRESULT StringCbCopy(LPTSTR pszDest,  
                    size_t cbDest, LPCTSTR pszSrc );
```



ASCII e UNICODE (resumo)

- Pense nas strings como arrays de caracteres em vez de arrays de char ou bytes
- Utilize tipos genéricos para representar caracteres e strings, como por exemplo, TCHAR e PTSTR
- Utilize tipos de dados explícitos para representar bytes (BYTE), apontadores para byte(PBYTE) e buffers de dados(BYTE buf[10], PBYTE *pbuf)
- Utilize a macro TEXT na definição de literais de caracteres ou strings:
 - TCHAR c = TEXT('b');
 - TCHAR str[] = TEXT("ola")
- Substitua os tipos comprometidos com um determinado tipo de codificação de caracteres por um tipo genérico (PSTR por PTSTR)
- Ter em consideração os problemas de aritmética com strings:
 - O nº de caracteres de uma string: $\text{sizeof}(szBuffer) / \text{sizeof}(TCHAR)$
 - Reservar memória para uma string: $\text{malloc}(nCharacters * \text{sizeof}(TCHAR))$



Aplicações Win32



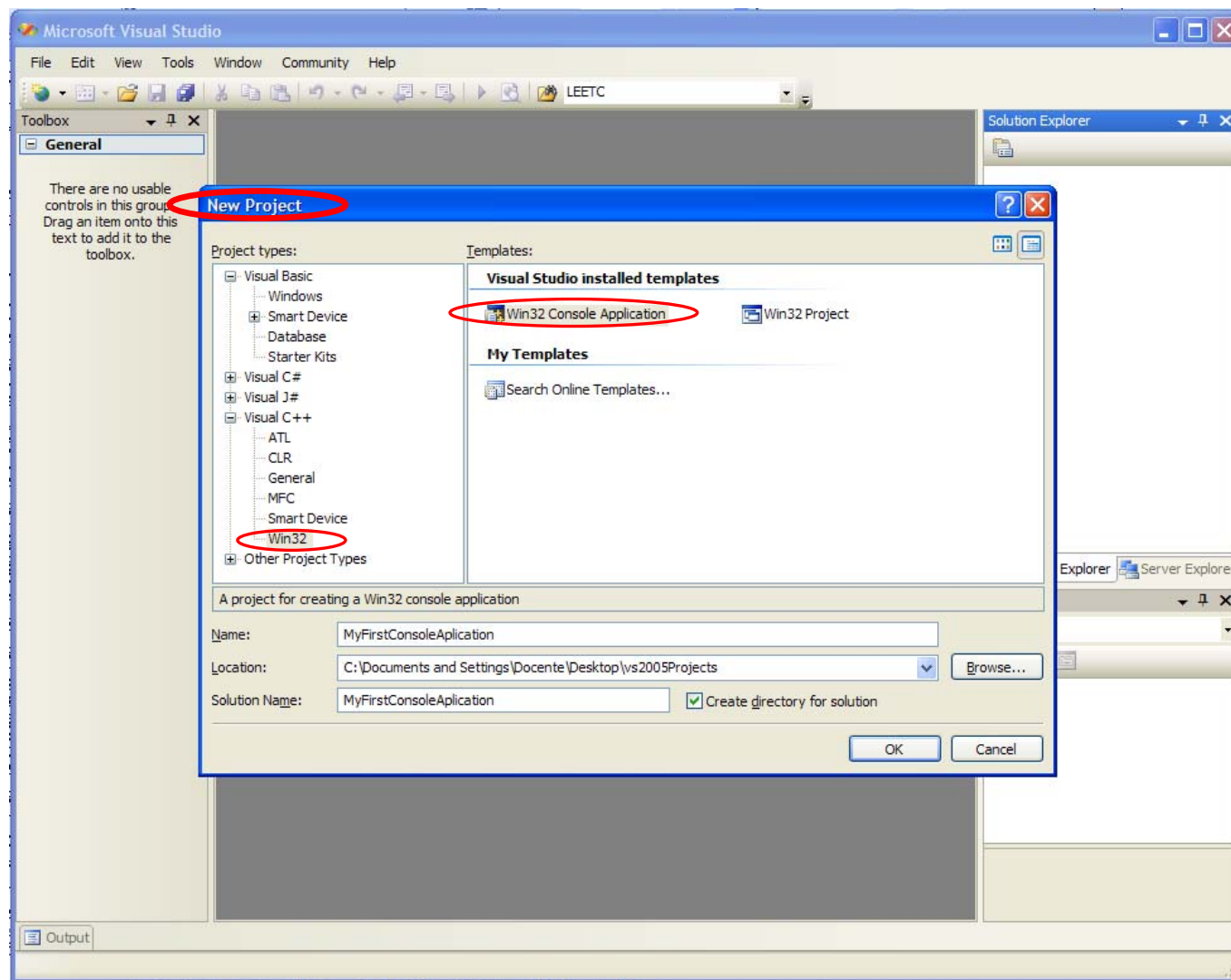
Windows.h

O ficheiro <windows.h> faz o *import* dos seguintes ficheiros (de *include*):

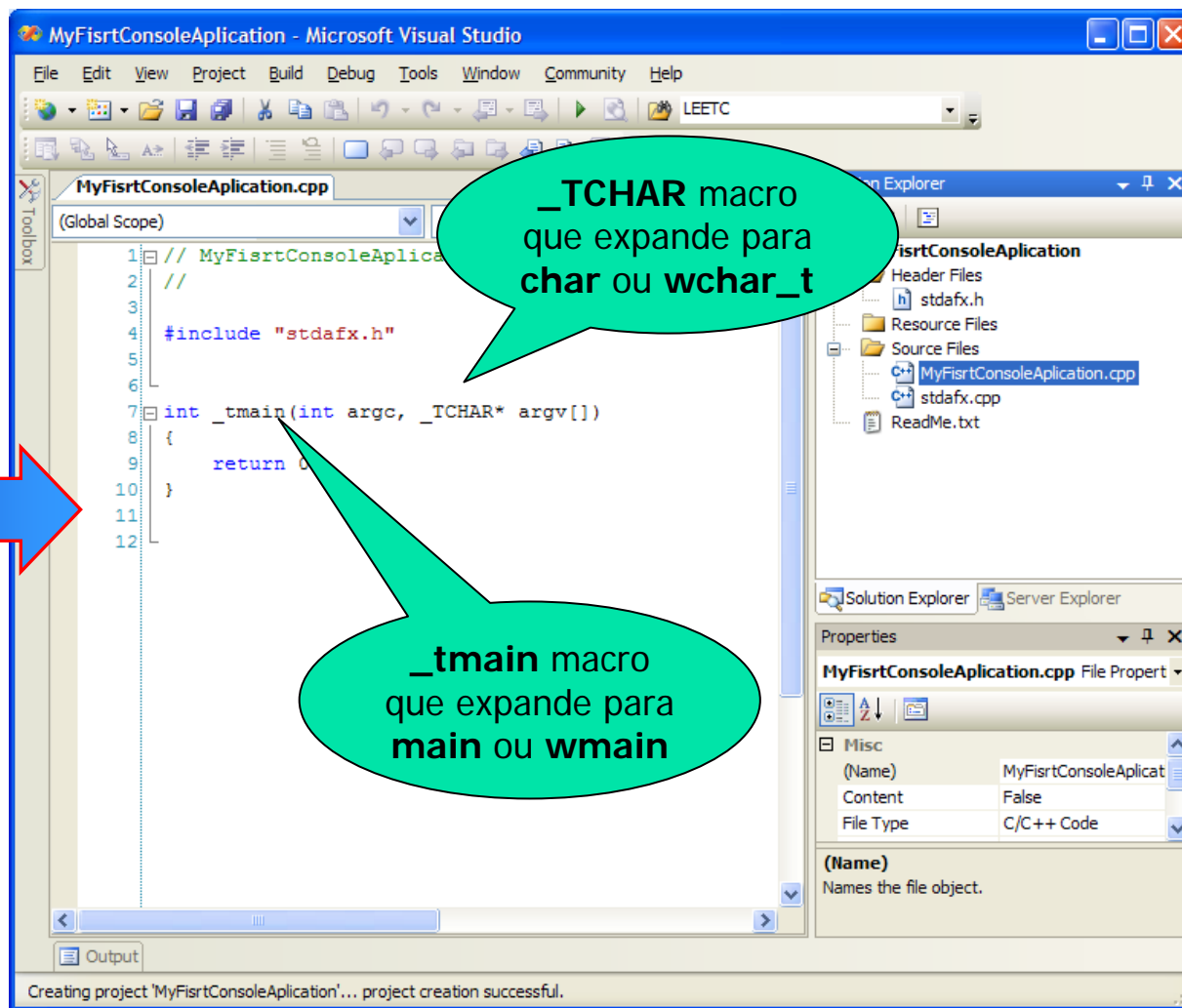
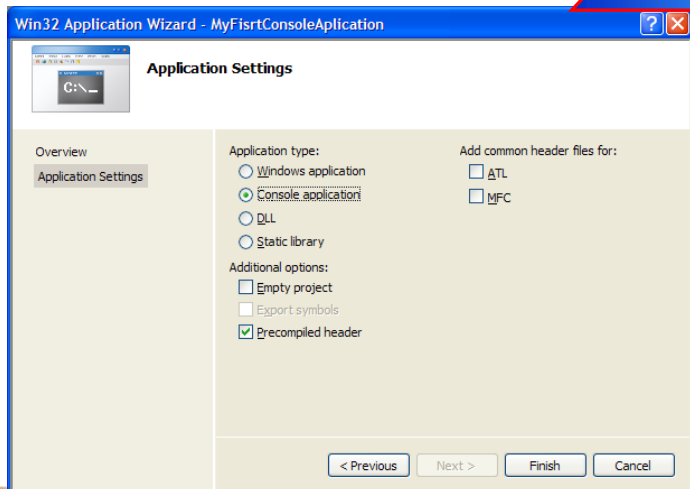
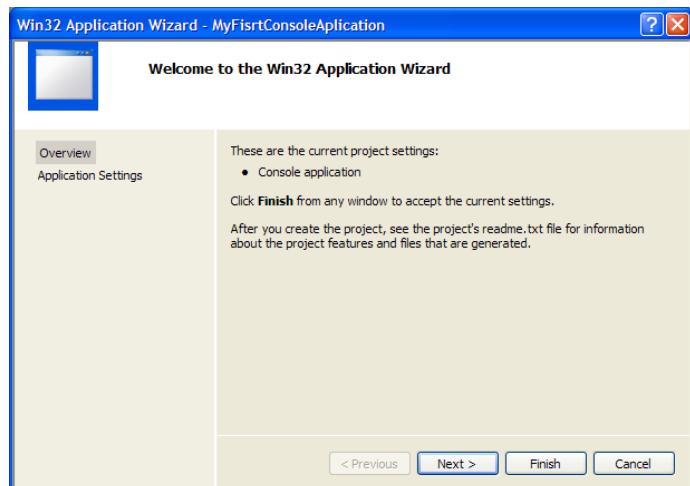
- **WinDEF.h** – Definição de tipos básicos
- **WinNT.h** – Definição de tipos para suporte UNICODE
- **WinBASE.h** – Interface das funções do *Kernel*
- **WinUSER.h** – Funções da Interface com utilizador (*user interface*)
- **WinGDI.h** – Funções da interface gráfica (*Graphics Device Interface*)



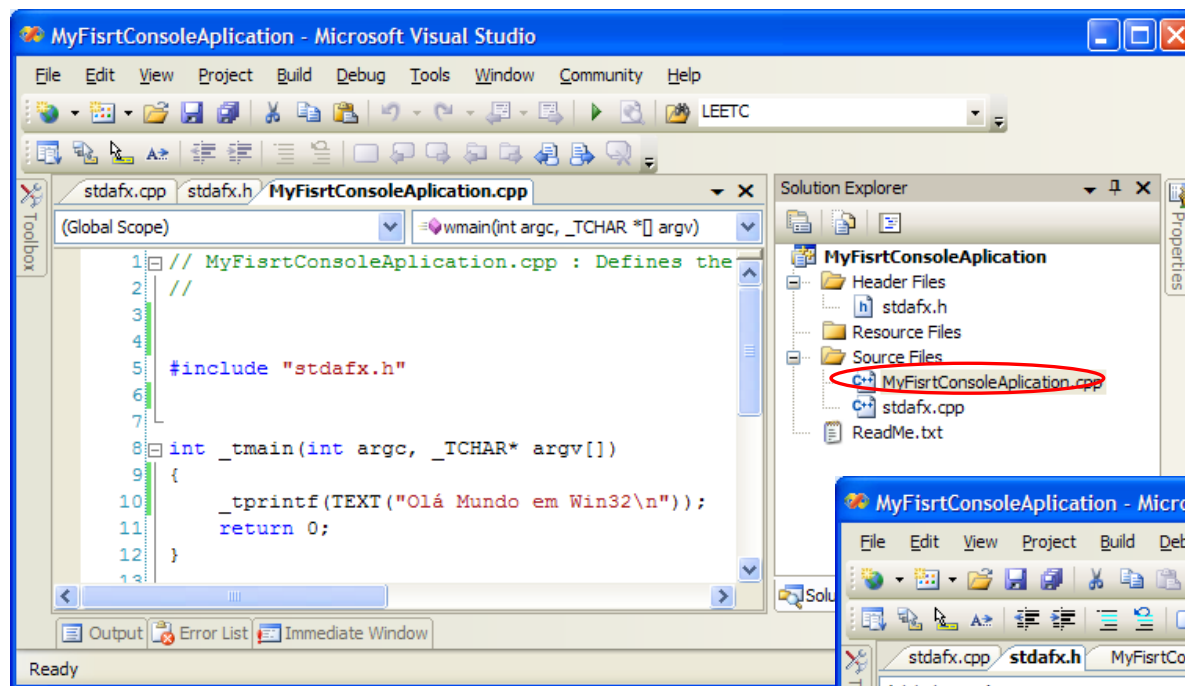
Criação de um projecto no *Visual Studio 2005* para uma aplicação em *modo de consola*



Criação de um projecto no *Visual Studio 2005* para uma aplicação em *modo de consola*



Olá Mundo



MyFisrtConsoleApplication - Microsoft Visual Studio

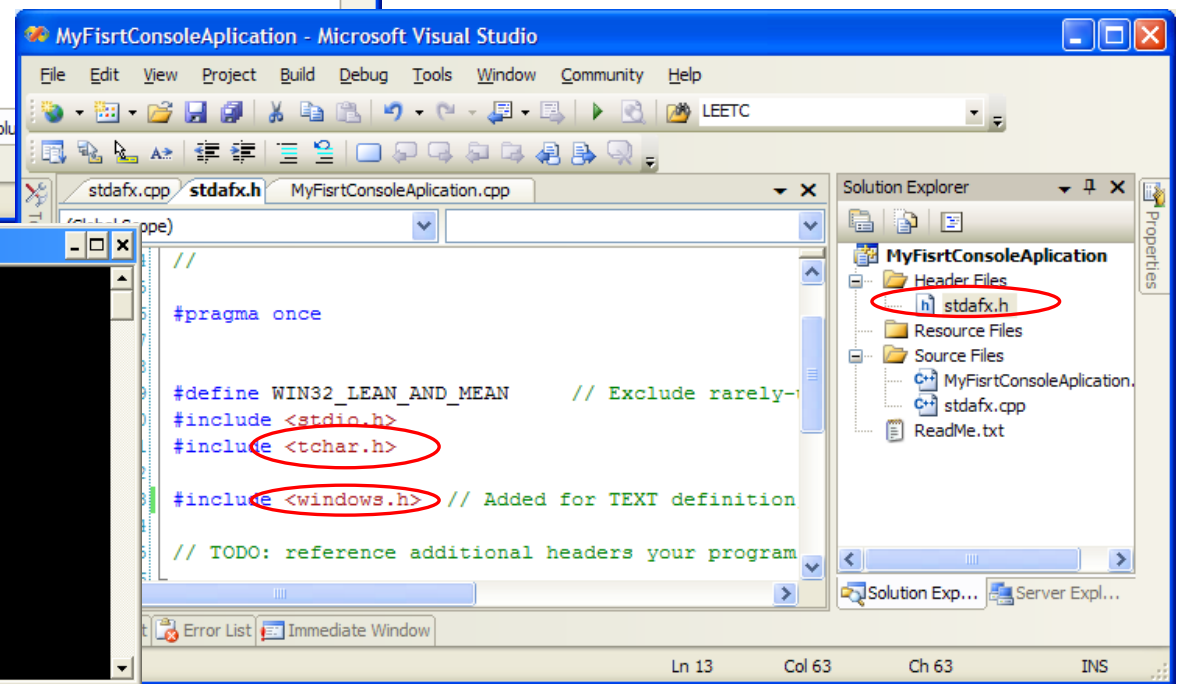
```
1 // MyFisrtConsoleApplication.cpp : Defines the
2 //
3 //
4
5 #include "stdafx.h"
6
7
8 int _tmain(int argc, _TCHAR* argv[])
9 {
10     _tprintf(TEXT("Olá Mundo em Win32\n"));
11     return 0;
12 }
```

Solution Explorer

- MyFisrtConsoleApplication
 - Header Files
 - stdafx.h
 - Resource Files
 - Source Files
 - MyFisrtConsoleApplication.cpp
 - stdafx.cpp
 - ReadMe.txt

C:\WINDOWS\system32\cmd.exe

```
010 Mundo em Win32
Press any key to continue . . .
```



MyFisrtConsoleApplication - Microsoft Visual Studio

```
//
#pragma once

#define WIN32_LEAN_AND_MEAN // Exclude rarely-
#include <stdio.h>
#include <tchar.h>
#include <windows.h> // Added for TEXT definition
// TODO: reference additional headers your program
```

Solution Explorer

- MyFisrtConsoleApplication
 - Header Files
 - stdafx.h
 - Resource Files
 - Source Files
 - MyFisrtConsoleApplication.cpp
 - stdafx.cpp
 - ReadMe.txt



Localização – “*Locale*”

Escrita na consola adequada ao Local



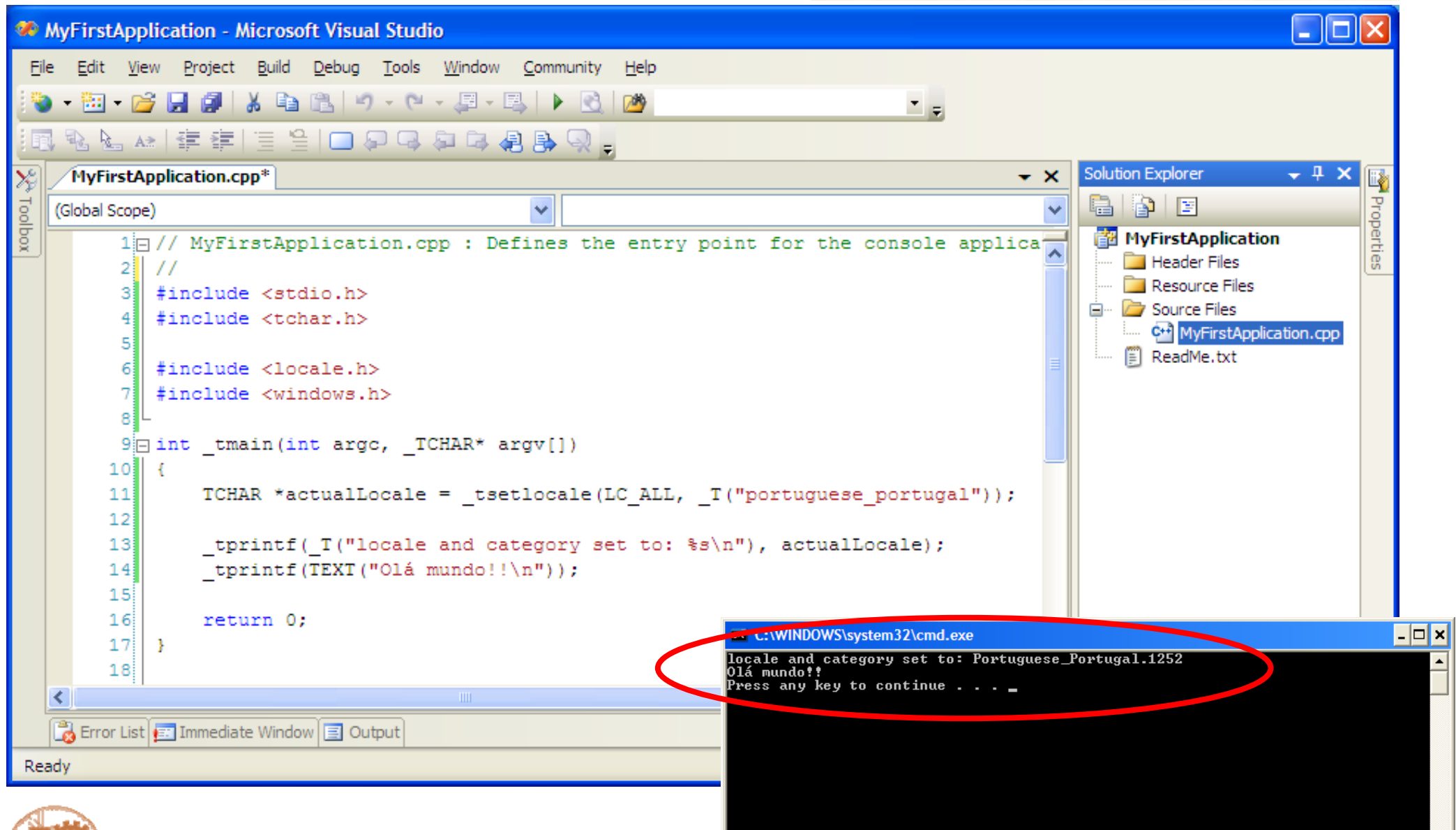
Localização

- **"Locale"** – Refere a localização (*País/Região e Língua*) para o qual podemos configurar certos aspectos do nosso programa.
- **Categorias dependentes da localização:**
 - Formatação de datas;
 - Formatação de valores monetários;
 - Impressão de caracteres próprios da língua.
 - ...
- **Alteração do "locale"**
 - `_tsetlocale(int category, TCHAR *)`
 - macro que expande para `setlocale(...)[8bits]` ou `_wsetlocale(...)[16bits]`
- **Categorias** (*ver MSDN para detalhes*):
 - LC_ALL – (Afeta todas as categorias)
 - LC_COLLATE
 - LC_CTYPE
 - LC_MONETARY
 - LC_NUMERIC
 - LC_TIME
- **Local:** `"portuguese_portugal"`

```
TCHAR *actualLocale = _tsetlocale(LC_ALL, _T("portuguese_portugal"));
```



Exemplo



The image shows a screenshot of the Microsoft Visual Studio IDE. The main window displays the source code for `MyFirstApplication.cpp`. The code is as follows:

```
// MyFirstApplication.cpp : Defines the entry point for the console application.
//
#include <stdio.h>
#include <tchar.h>
#include <locale.h>
#include <windows.h>

int _tmain(int argc, _TCHAR* argv[])
{
    TCHAR *actualLocale = _tsetlocale(LC_ALL, _T("portuguese_portugal"));
    _tprintf(_T("locale and category set to: %s\n"), actualLocale);
    _tprintf(TEXT("Olá mundo!!\n"));
    return 0;
}
```

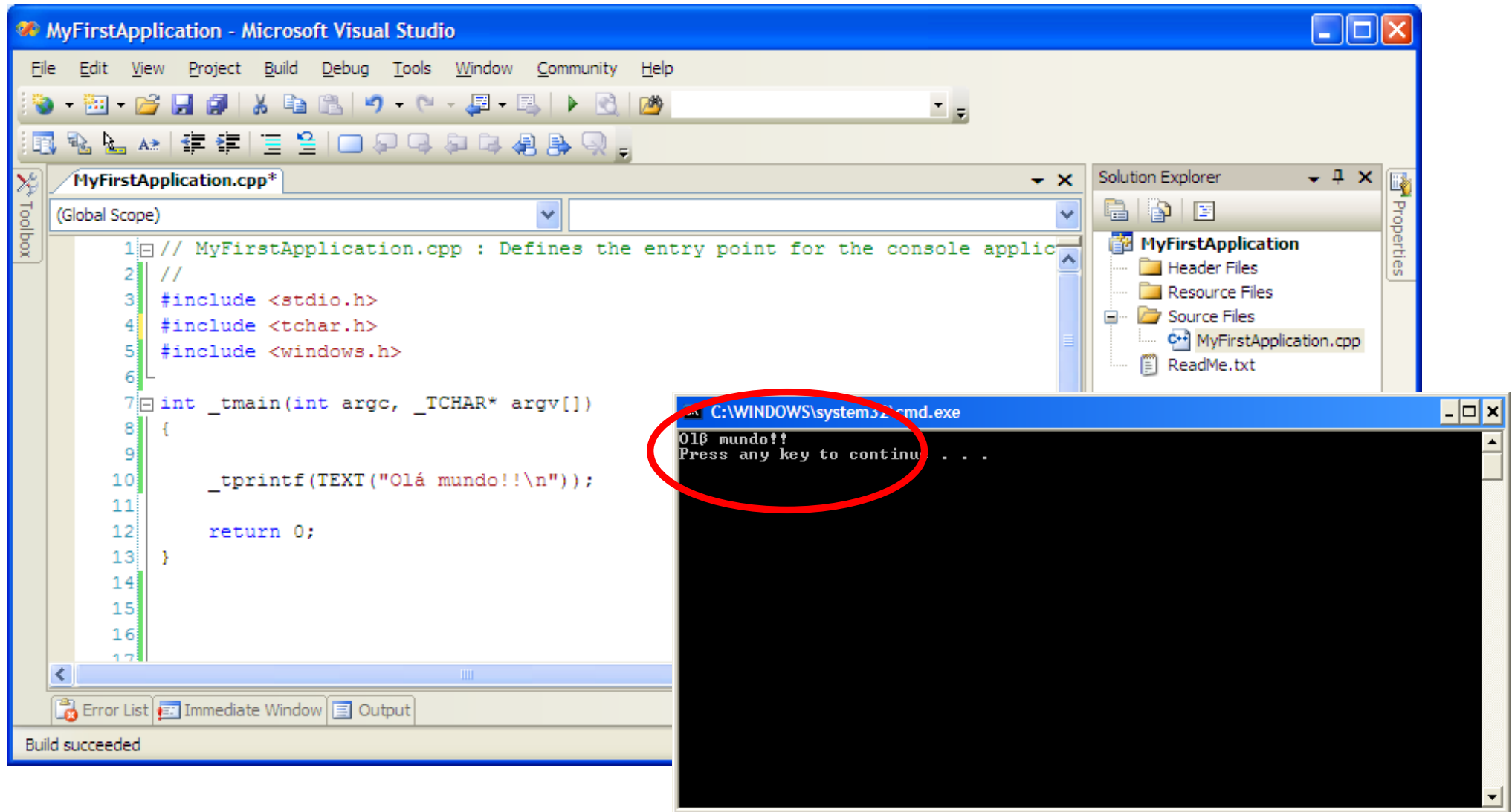
The Solution Explorer on the right shows the project structure for `MyFirstApplication`, including `Header Files`, `Resource Files`, `Source Files`, and the `MyFirstApplication.cpp` file. Below the code editor, a command prompt window is open, showing the output of the program:

```
C:\WINDOWS\system32\cmd.exe
locale and category set to: Portuguese_Portugal.1252
Olá mundo!!
Press any key to continue . . .
```

The command prompt window is circled in red. The status bar at the bottom of the IDE shows "Ready".



Exemplo sem Localização



Tratamento de erros

Tratamento de erros

- As chamadas de sistema na situação de erro devolvem um valor que permite à aplicação verificar o sucesso ou insucesso do serviço;
- Na situação de erro o sistema mantém o código do último erro que ocorreu (no contexto de cada tarefa);
- Este código descreve de uma forma detalhada a situação que o provocou;
- As aplicações podem obter este código através da função `GetLastError()`
- Através da função `FormatMessage()` é possível converter o código do erro numa informação textual (dependente da língua do sistema operativo)
- De forma a facilitar a apresentação dos erros foram definidas as seguintes funções de apoio aos exemplos: `ReportErrorSystem()` e `FatalErrorSystem()` definidas nos ficheiros `SesError.h` e `SesError.cpp`



Funções para apresentação de erros

Estas funções têm uma utilização idêntica à função `printf()` da linguagem C, mas apresentam o seu output numa `MessageBox`.

Ambas mostram o último erro de sistema, mas a função `FatalErrorSystem` também termina o próprio processo.

```
void ReportErrorSystem( const char *fmtStr, ... );  
void FatalErrorSystem( const char *fmtStr, ... );
```

Existem definidas mais duas funções semelhantes às anteriores sem apresentarem o código de erro nem a informação textual do erro:

```
void ReportErrorUser( const char *fmtStr, ... );  
void FatalErrorUser( const char *fmtStr, ... );
```



Um exemplo utilização *FatalErrorSystem*

```
int _tmain(int argc, _TCHAR* argv[])
{
    HANDLE hFile = CreateFile(TEXT("DummyFile"), 0, 0, NULL, OPEN_EXISTING, 0, NULL);

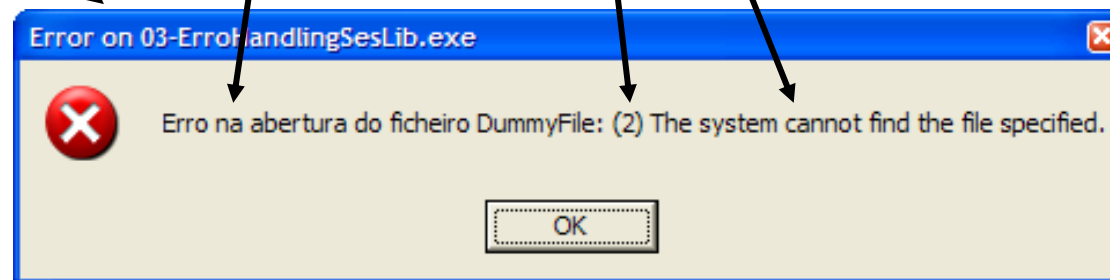
    if (hFile == INVALID_HANDLE_VALUE)
        FatalErrorSystem(TEXT("Erro na abertura do ficheiro %s"), TEXT("DummyFile"));

    return 0;
}
```

No caso de ocorrer um erro
este vai ser apresentado na
janela seguinte

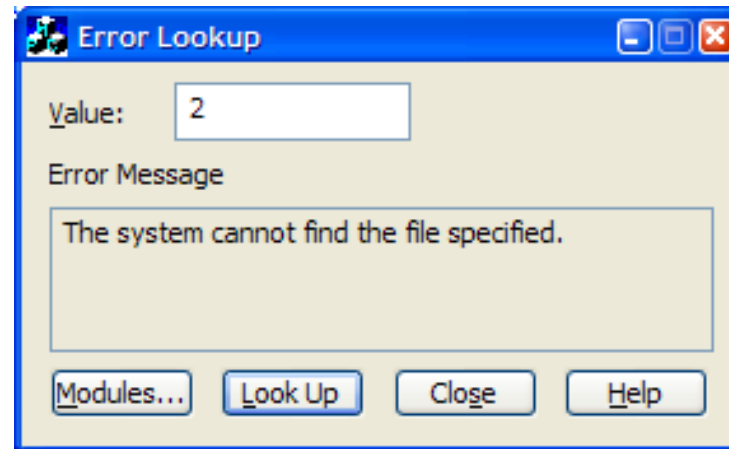
Código do erro (GetLastError)

Descrição textual (FormatMessage)



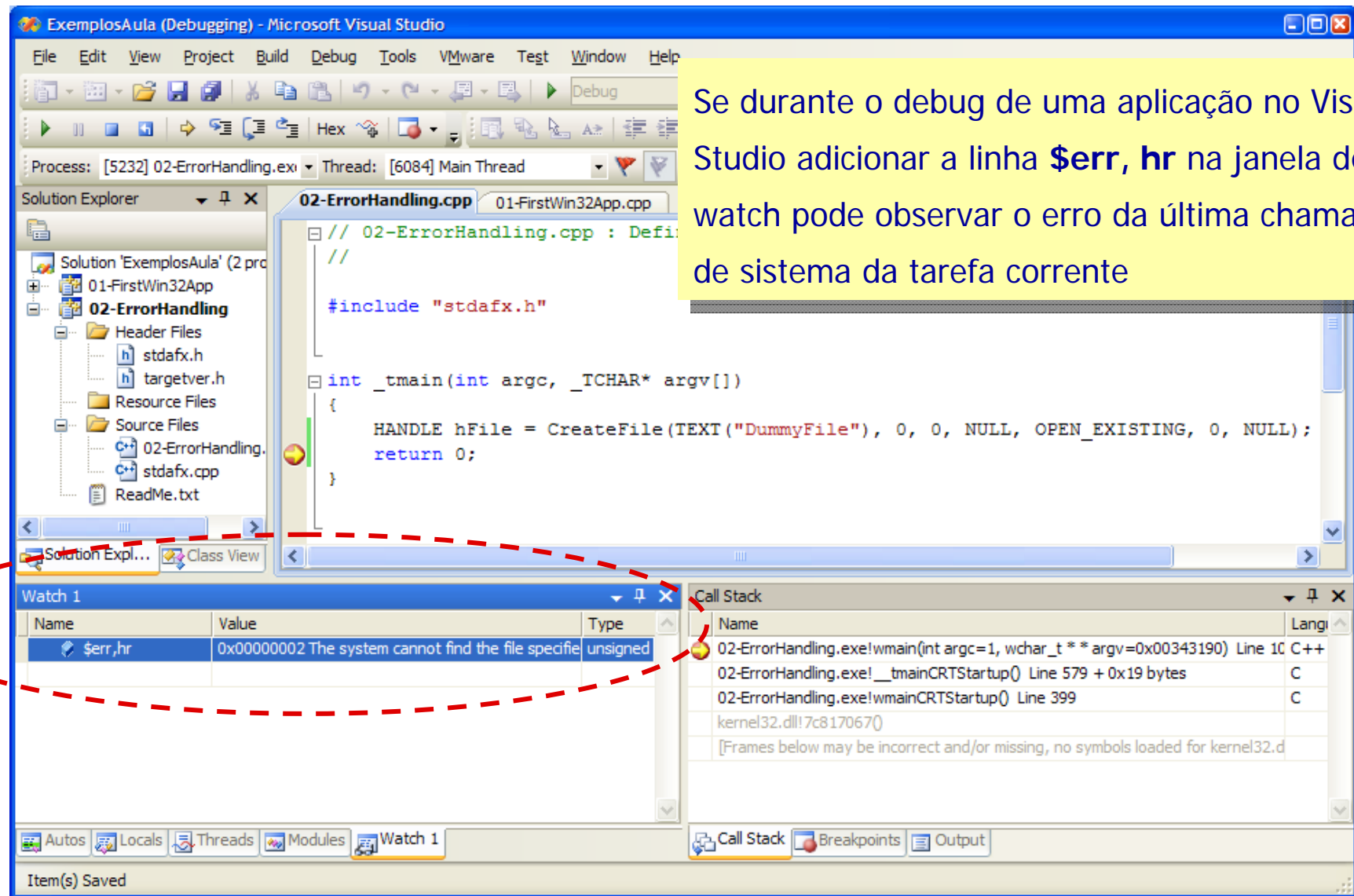
Conversão do código de erro

O Visual Studio possui o utilitário Error Lookup (menu Tools -> Error Lookup) que permite determinar a mensagem de erro associada a um código de erro



Conversão do código de erro

Se durante o debug de uma aplicação no Visual Studio adicionar a linha **\$err, hr** na janela de watch pode observar o erro da última chamada de sistema da tarefa corrente



The screenshot displays the Visual Studio IDE with the following components:

- Process:** [5232] 02-ErrorHandling.exe, Thread: [6084] Main Thread
- Solution Explorer:** Shows the project structure for 'ExemplosAula' (2 projects), including '01-FirstWin32App' and '02-ErrorHandling'. The '02-ErrorHandling' project contains 'Header Files' (stdafx.h, targetver.h), 'Resource Files', 'Source Files' (02-ErrorHandling.cpp, stdafx.cpp), and 'ReadMe.txt'.
- Code Editor:** Displays the source code for '02-ErrorHandling.cpp'. The code includes 'stdafx.h' and defines a function 'int _tmain(int argc, _TCHAR* argv[])'. Inside the function, 'HANDLE hFile = CreateFile(TEXT("DummyFile"), 0, 0, NULL, OPEN_EXISTING, 0, NULL);' is called, followed by 'return 0;'. A red arrow points to the 'CreateFile' call.
- Watch Window:** Shows a single variable '\$err,hr' with a value of '0x00000002' and a description 'The system cannot find the file specified'. The type is 'unsigned'.
- Call Stack:** Shows the sequence of function calls leading to the error. The top frame is '02-ErrorHandling.exe!wmain(int argc=1, wchar_t ** argv=0x00343190) Line 10 C++'. Below it are '02-ErrorHandling.exe!__mainCRTStartup() Line 579 + 0x19 bytes C' and '02-ErrorHandling.exe!wmainCRTStartup() Line 399 C'. The bottom frame is 'kernel32.dll!7c817067()'. A note at the bottom states: '[Frames below may be incorrect and/or missing, no symbols loaded for kernel32.d]'. The language is set to 'C++'.



Objectos do Kernel

Objectos do *Kernel*

(*kernel objects*)

Exemplos de objectos
Kernel

File, Mailslot, pipe,...
Mutex, Semaphore, Event, ...
Process, Thread

Características dos objectos do *kernel* :

- ⊗ são manipulados pelas funções da WIN32 API e controlados pelo SO
- ⊗ são referenciados por um **HANDLE**
- ⊗ têm **um descritor de segurança** associado, ...
- ⊗ têm um contador de referências

O *Kernel* proporciona : partilha, protecção, acesso/nome,...



Uso de Objectos do *Kernel*

Criar um objecto *Kernel*

```
Exemplo:HANDLE CreateMutex(..., MutexName);
```

Partilhar um objecto já existente

```
Exemplo:HANDLE OpenMutex(access_permissions,...,MutexName);
```

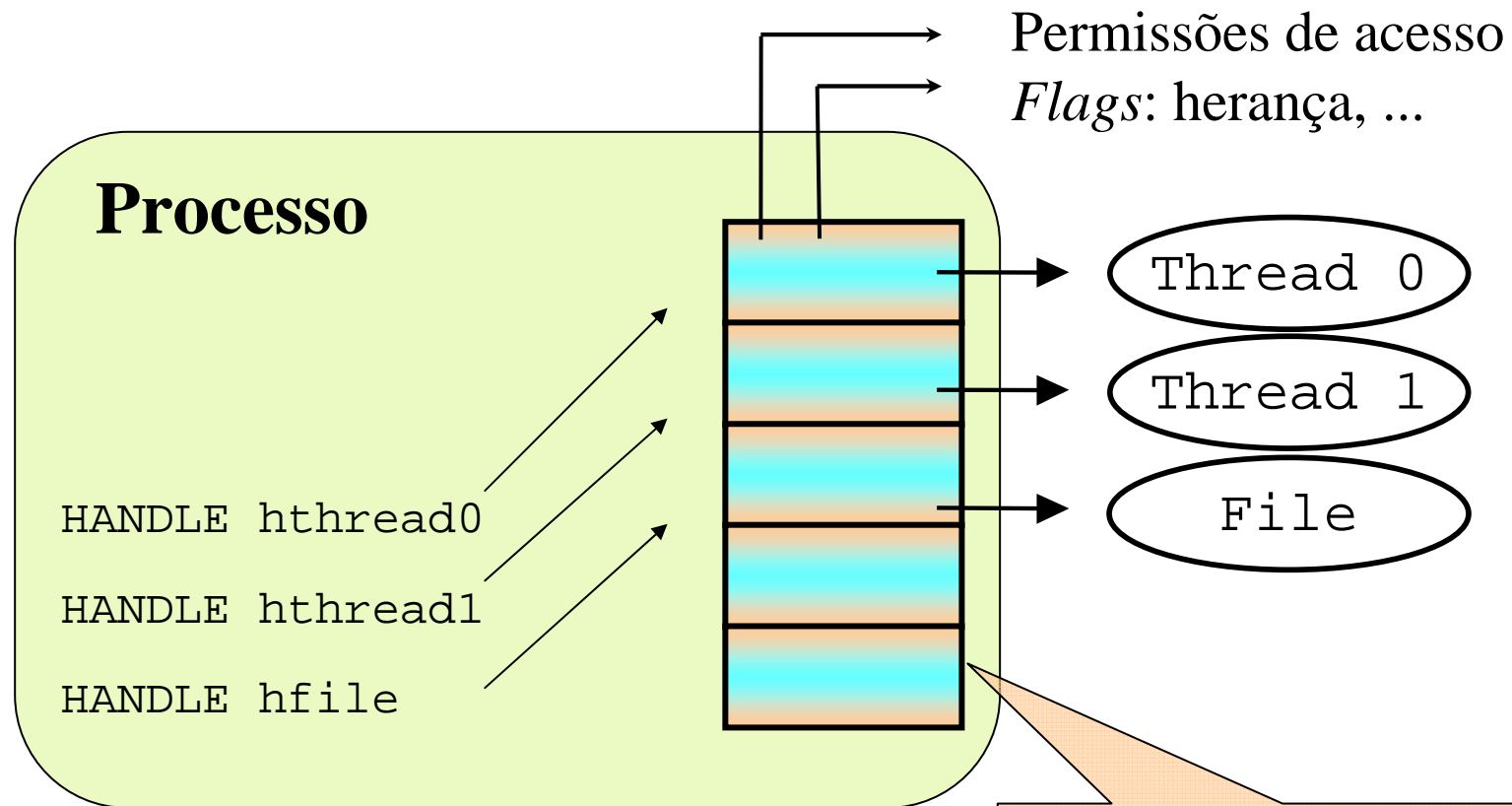
“Eliminar” uma referência (*handle*) para o objecto

```
BOOL CloseHandle(HANDLE hObject);
```

Um objecto só é eliminado do sistema quando o seu número de referências for zero.



Objectos do *kernel* num processo

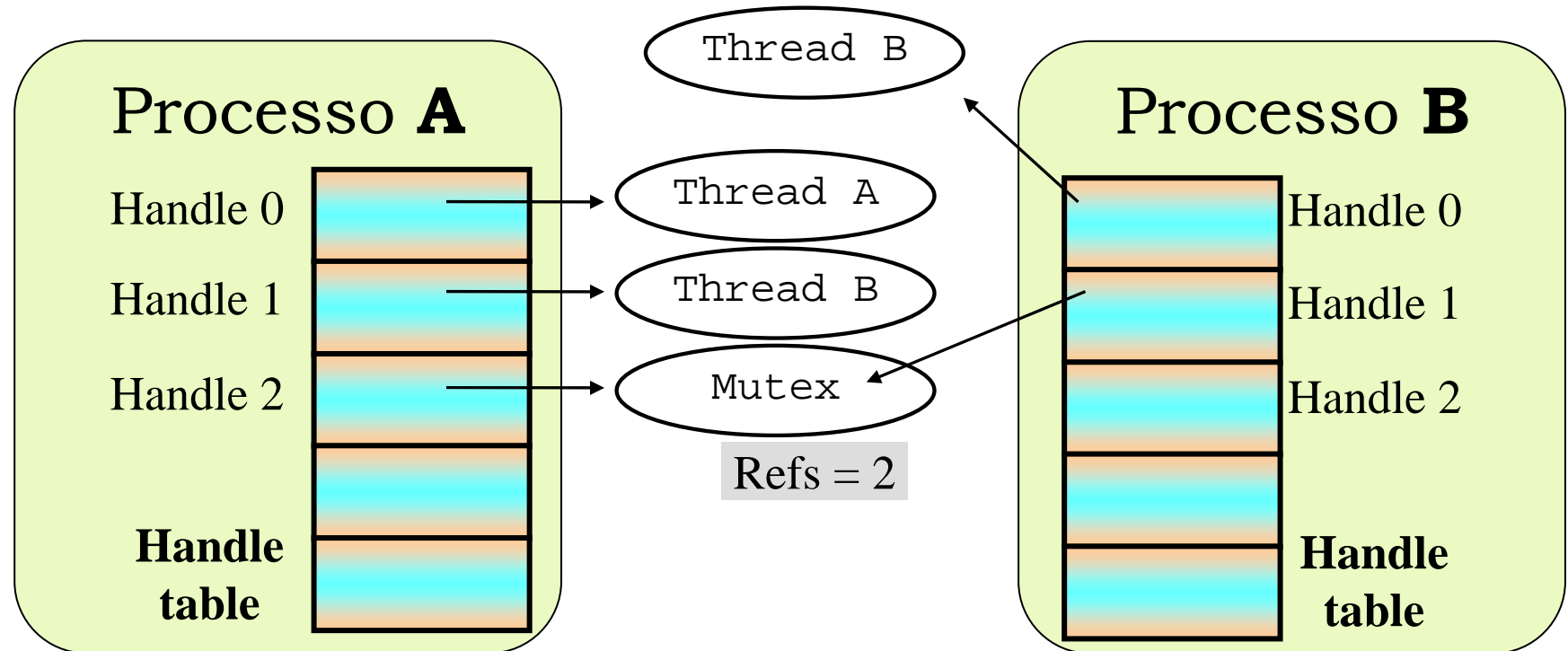


Os HANDLEs são simples referências para entradas da tabela de *handles* do processo

Tabela de *handles* de objectos do *kernel*



Partilha de objectos



Partilha por:

acesso ao objecto usando o nome

passagem do *handle* ao outro processo:

por herança

por evocação da função `DuplicateHandle`



Win32 *Data Types*

Alguns exemplos

BOOL	Boolean variable (should be TRUE or FALSE).
CALLBACK	Calling convention for callback functions.
WINAPI	Calling convention for the Win32 API.
WNDPROC	Pointer to an application-defined window procedure.
DLGPROC	Pointer to an application-defined dialog box callback procedure
DWORD	32-bit unsigned integer.
HANDLE	Handle to an object.
HMENU	Handle to a menu.
LPARAM, WPARAM	Message parameters.
WCHAR	16-bit Unicode character.
WORD	16-bit unsigned integer.
LPCSTR	Ptr to a constant null-terminated string of 8-bit Windows (ANSI) chars.
LPCWSTR	Ptr to a constant null-terminated string of 16-bit Unicode characters.
LPVOID	Pointer to any type.
LPDWORD	Pointer to a DWORD.
LRESULT	Signed result of message processing.
PCWSTR	Ptr to a constant null-terminated string of 16-bit Unicode characters.
PHANDLE	Pointer to a HANDLE.

