Unit 2: Windows 2000 Architecture

2.1. Structuring of the Windows 2000 Operating System

Windows 2000 System Architecture and System Mechanisms

Requirements & Design Goals

Architecture Overview

Key System Components

Trap Dispatching

Object Manager

Synchronization

Local Procedure Calls

Requirements and Design Goals

- 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 (Client & 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

Goals (contd.)

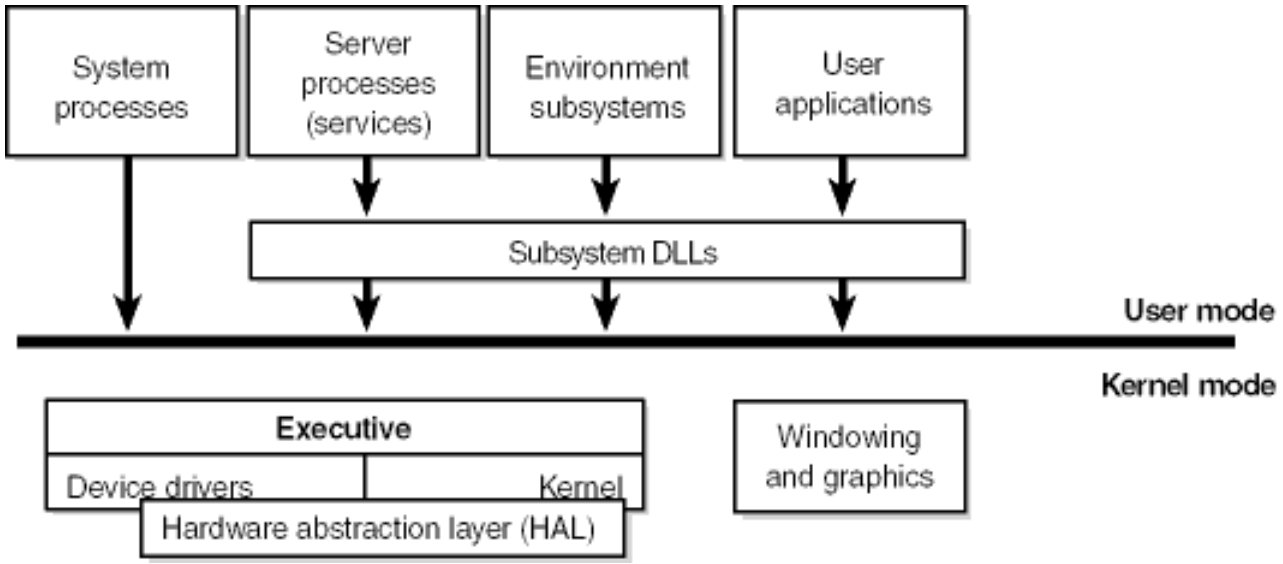
- **Extensibility**
  - Code must be able to grow and change as market requirements change.
- **Portability**
  - The system must be able to run on multiple hardware architectures and must be able to move with relative ease to new ones as market demands dictate.
- **Reliability and Robustness**
  - Protection against internal malfunction and external tampering.
  - Applications should not be able to harm the OS or other running applications.
- **Compatibility**
  - User interface and APIs should be compatible with older versions of Windows as well as older operating systems such as MS-DOS.
  - It should also interoperate well with 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.

Microkernel Operating Systems

- Client/server systems fall within a spectrum
  - some doing very little work in kernel mode and others doing more.
- Carnegie Mellon University Mach operating system
  - contemporary example of the client/server microkernel system,
  - implements minimal kernel that comprises thread scheduling, message passing, virtual memory, and device drivers
  - Everything else, including various APIs, file systems, and networking, runs in user mode.
- Commercial implementations of Mach run file system, networking, and memory management in kernel mode
- The reason: the pure microkernel design is too slow
  - Windows NT 3.51 was comparable to Mach
  - Windows NT 4.0 moved significant part of Win32 subsystem (GDI, Window Manager) into kernel

Windows 2000 Architecture (simplified)

- User mode versus kernel mode
- More crashes due to Win32 execution in kernel mode?
  no ! Important user-space server would even crash microkernel OS

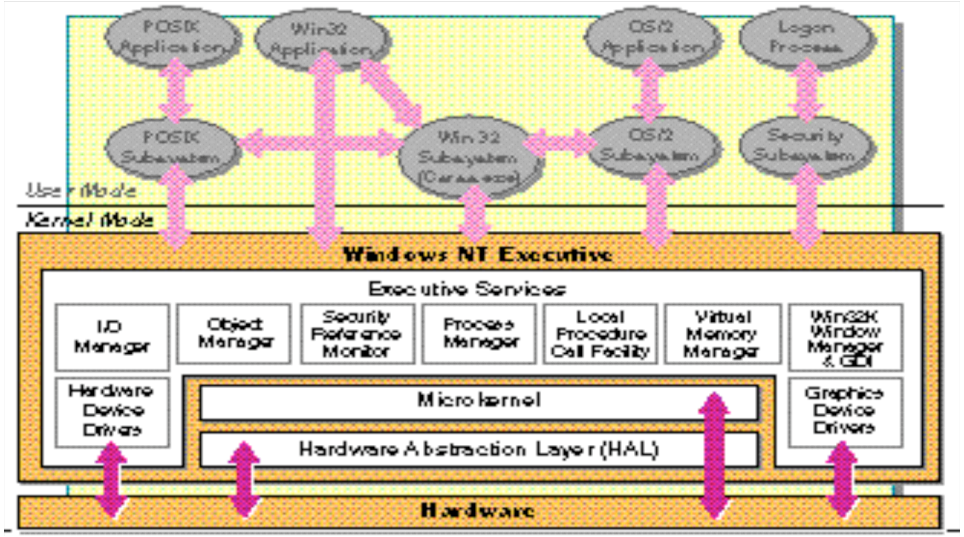


Process Types (user proc.)

- System support processes:
  - logon process, session manager
  - Not started by the service controller
- Server processes that are Windows 2000 services:
  - Event log, scheduler service
  - Components of add-on apps: SQL server, exchange server
- Environment subsystems (personalities):
  - Win32, POSIX, OS/2 1.2
  - Subsystem DLLs (documented function -> NT service call)
- User applications (5 types):
  - Win32, Windows 3.1, MS-DOS, POSIX, OS/2 1.2

Kernel mode components

- NT *executive*: memory, process, thread mang., security, I/O, IPC
- NT *kernel*: low-level OS func – scheduling, interrupts, exceptions, multiprocessor synch.



Portability

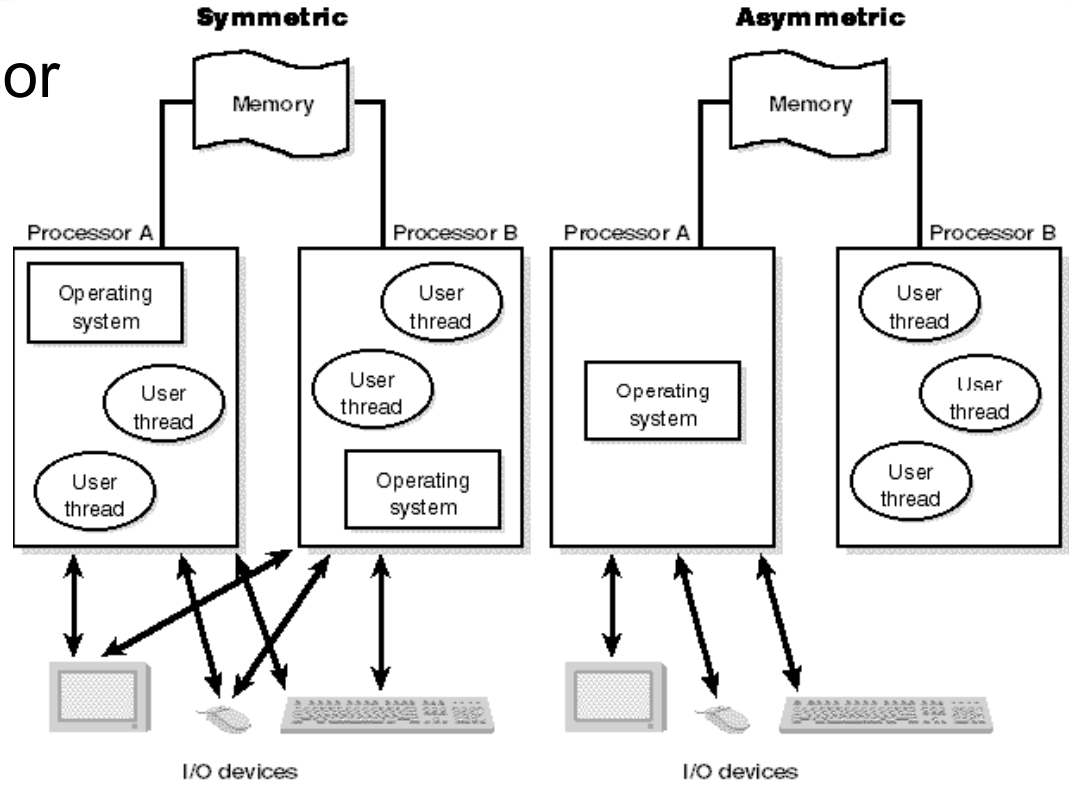
- HAL (Hardware Abstraction Layer):
  - support for x86 (initial), MIPS (initial), Alpha AXP, PowerPC (NT 3.51), Itanium (Windows 2000)
  - Machine-specific functions located in HAL
- Layered design:
  - architecture-specific functions located in kernel

- Windows 2000 is written in C
  - (OS executive, utilities, drivers)
- UI and graphics subsystem
  - written in C++
- HW-specific/performance-sensitive parts
  - written in assembly lang: int trap handler, context switching

Symmetric Multiprocessing (SMP)

- No master processor
- Up to 32 PE

- W2K Pro: 2
- W2K S: 4
- W2K/AS: 8



- Modified HAL for more than 8 processors

HKLM\System\CurrentControlSet\SessionManager\LicensedProcessors

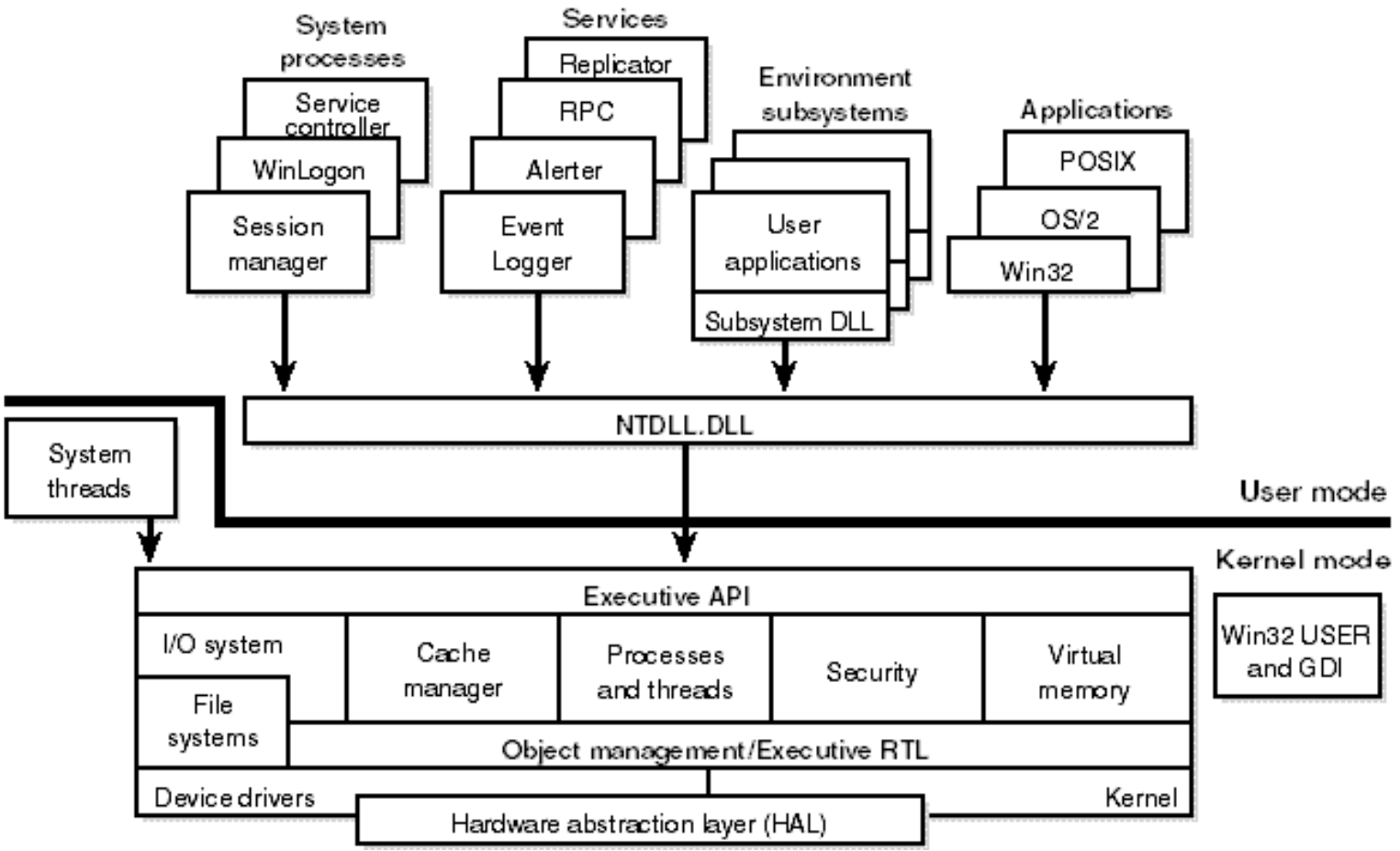
SMP supported by OS

- OS code runs on every processor; preemptable
  - Exception: scheduling & interrupt handling
- Multithreading; potentially simultaneous execution
- Fine-grained synchronization in kernel/device drivers
- Multithreaded server processes
- Flexible object sharing; IPC
  - Shared memory, message passing
- Single version of W2K:
  - SMP requires different HALs and kernels (on CD):
  - NTOSKRNL.EXE – uniprocessor executive/kernel
  - NTKRNLMP.EXE – multiprocessor executive/kernel (same sources)
  - Selection at installation time, file is always installed as \winnt\system32\NTOSKRNL.EXE

Windows 2000 Professional vs. Windows 2000 Server

- Same source; scheduling handled differently
- Services:
  - network management and directory services: Active Directory
  - Disk FT features (striping with parity and mirroring)
  - Services for Macintosh: file and printer sharing, user admin
  - Gateway Service for NetWare
  - TCP/IP: Domain Name System (DNS) and Dynamic Host Configuration Protocol (DHCP)
  - Remote boot server for diskless MS-DOS, Win3.1, Win95 PCs
- Licensing:
  - W2K Pro: 10 netw. Conn; 10 printer/file sharing conn.

Key System Components



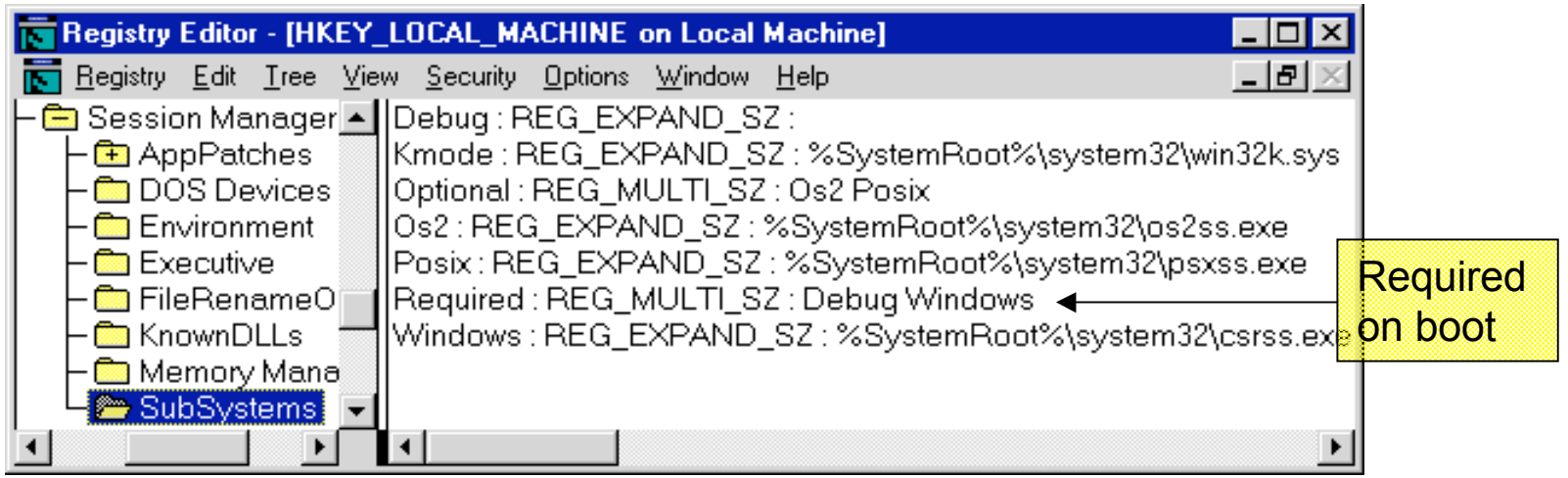
Key Windows 2000 System Files

SERVICES.EXE	Service controller process
WINLOGON.EXE	Logon process
SMSS.EXE	Session manager process
PSXSS.EXE	POSIX subsystem process
OS2SS.EXE	OS/2 subsystem process
CSRSS.EXE*	Win32 subsystem process
NTDLL.DLL	Internal support functions and system service dispatch stubs to executive functions
KERNEL32.DLL, USER32.DLL, GDI32.DLL.	Win32 subsystem DLLs
PSXDLL.DLL	POSIX subsystem DLL
NTOSKRNL.EXE**	Executive and kernel
HAL.DLL	Hardware abstraction layer
WIN32K.SYS	Win32 USER and GDI kernel-mode components

Subsystems

- POSIX (1003.1), OS/2 (Intel only), Win32 (required)
- Executable (.exe) is linked to exactly one subsystem
  - Win32 app cannot use POSIX fork (but: tlist –t)
  - Subsystems can be loaded on demand

(HKLM\System\CurrentControlSet\Control\Session Manager\Subsystems)



Full POSIX subsystem: Interix (from MS); GNU: www.cygnus.com

App calls Subsystem

- Function is entirely implemented in user mode
  - No message sent to environment subsystem process
  - No Win NT executive system service called
  - Examples: *PtInRect(), IsRectEmpty()*
- Function requires one/more calls to NT executive
  - Examples: Win32 *ReadFile() / WriteFile()* implemented using I/O system services *NtReadFile() / NtWriteFile()*
- Function requires some work in environment subsystem process (maintain state of client app)
  - Client/server request (message) to env. Subsystem (LPC facility)
  - Subsystem DLL waits for reply before returning to caller
- Combinations of 2/3: *CreateProcess() / CreateThread()*

Win32 Subsystem

- Environment subsystem process (CSRSS.EXE):
  - Console (text) windows
  - Creating and deleting processes and threads
  - Portions of the support for 16-bit virtual DOS machine (VDM)
  - Other func: *GetTempFile, DefineDosDevice, ExitWindowsEx*
- kernel-mode device driver (WIN32K.SYS):
  - Window manager: manages screen output;
  - input from keyboard, mouse, and other devices
  - user messages to applications.
  - Graphical Device Interface (GDI)

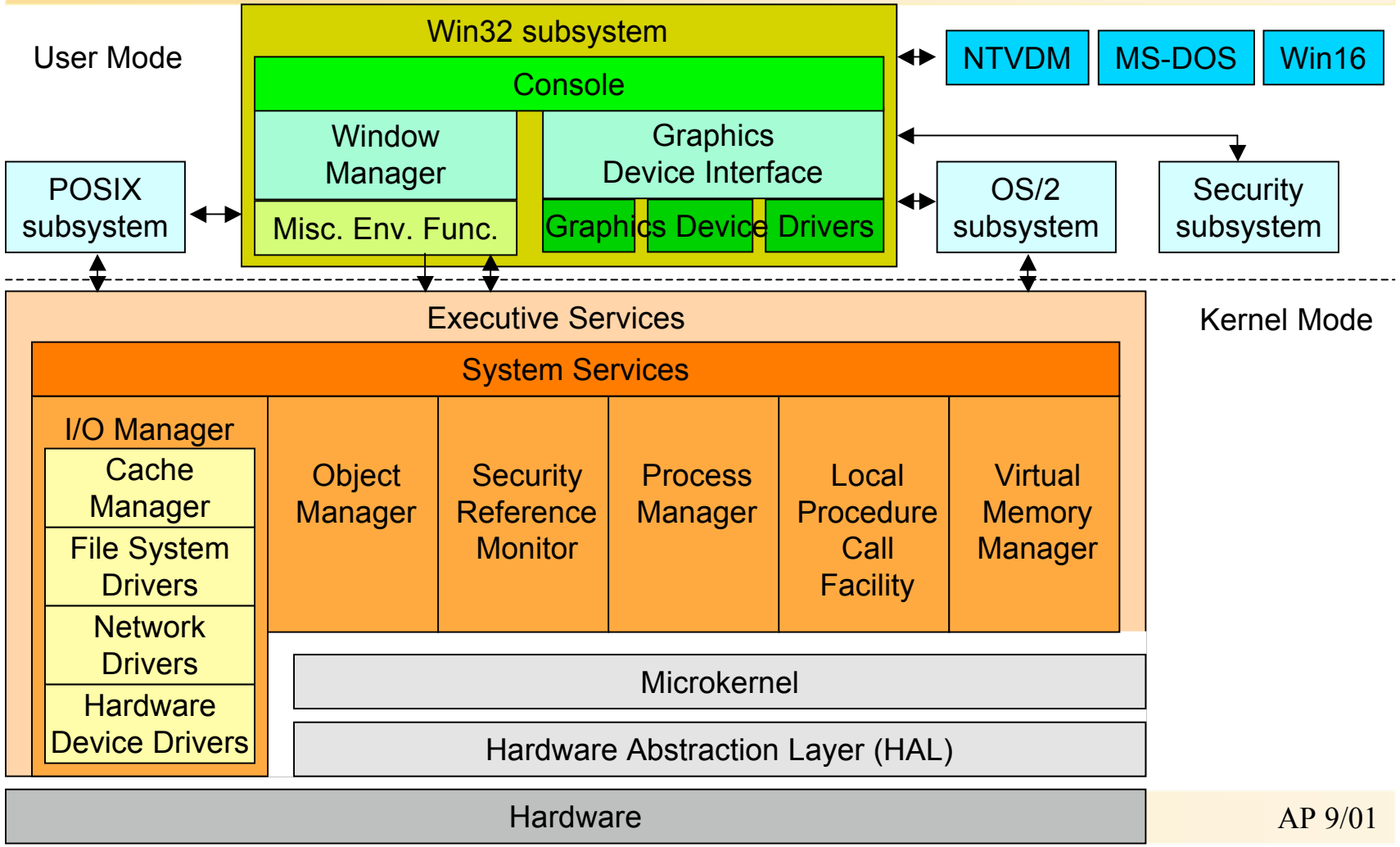
Win32 Subsystem (contd.)

- Subsystem DLLs (such as USER32.DLL, ADVAPI32.DLL, GDI32.DLL, and KERNEL32.DLL)
  - Translate Win32 API functions into calls to NTOSKRNL.EXE and WIN32K.SYS.
- Graphics device drivers
  - graphics display drivers, printer drivers, video miniport drivers

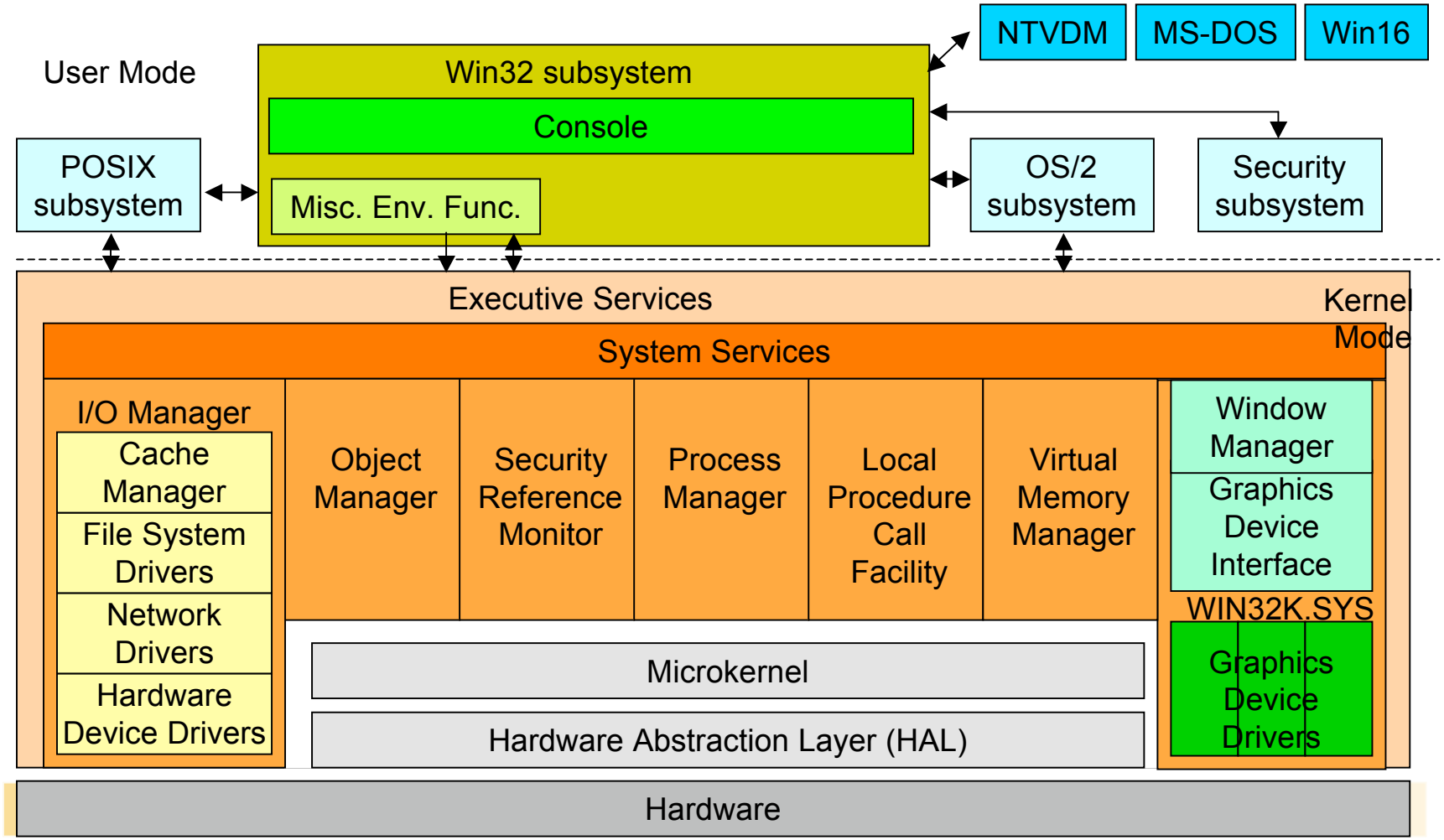
Prior to Windows NT 4.0, the window manager and graphics services were part of the user-mode Win32 subsystem process.

Is Windows NT Less Stable with Win32 USER and GDI in Kernel Mode?

Windows NT 3.51 Architecture



Windows NT 4.0 Architecture



What remains in Win32 Subsystem?

- Drawing and updating for console or text windows
  - console applications have no notion of repainting a window.
- Process and thread creation and termination
- Network drive letter mapping
- Creation of temporary files

- Win32 applications cause only few context switches to the Win32 subsystem process

POSIX Subsystem

- Windows 2000 implements POSIX.1
  - ISO/IEC 9945-1:1990 or IEEE POSIX standard 1003.1-1990
  - POSIX.1 compliance as specified in Federal Information Processing Standard (FIPS) 151-2 (NIST)
  - POSIX Conformance Document in \HELP in Platform SDK
- support for impl. of POSIX.1 subsystem was mandatory for NT
  - fork service in NT executive
  - hard file links in NTFS
- limited set of services
  - such as process control, IPC, simple character cell I/O
  - POSIX subsystem alone is not a complete programming environment
- POSIX executable cannot
  - create a thread or a window
  - use remote procedure calls (RPCs) or sockets

Microsoft supplies a full POSIX subsystem for Windows 2000 under the Interix product name

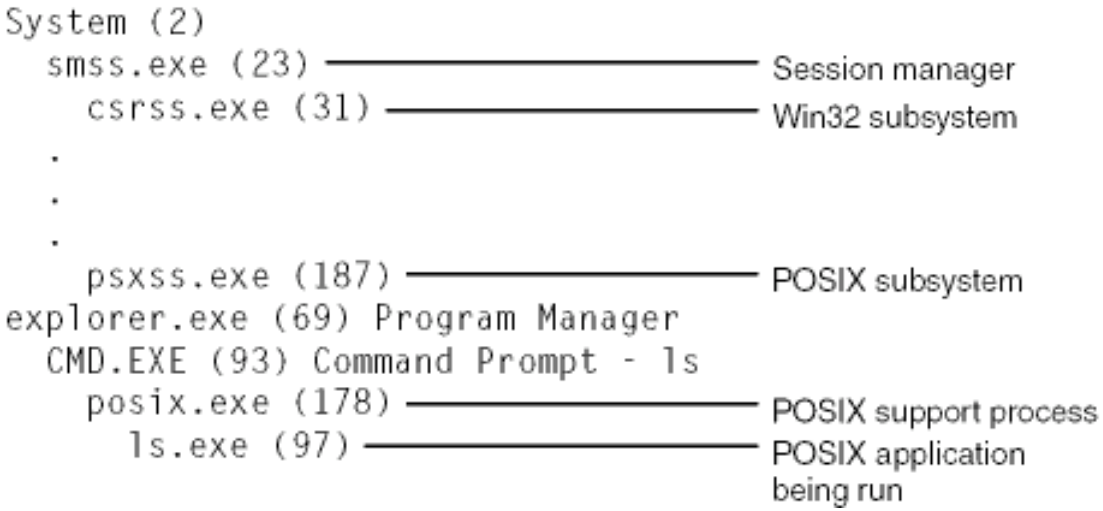
Porting UNIX Apps to NT

- UNIX-to-Win32 porting libraries
  - DataFocus (http://www.datafocus.com/)
  - ConsenSys (http://www.consensys.com/)
  - Cygnus - CygWin GNU tools (http://www.cygnus.com/)
- POSIX subsystem with complete UNIX system service and utilities environment
  - Interix from Microsoft (used to be SoftWay (http://www.opennt.com/)
  - Bought by Microsoft as of Sept. 99
- NT Resource Kit includes optional set of POSIX utilities
- POSIX executables are linked against PSXDLL.DLL
  - Header files in platform SDK

Watching the POSIX Subsystem Start

POSIX subsystem starts on demand:

1. Type *tlist /t*, check that POSIX subsyst. is not running
2. Run *\ntreskit\POSIX\LS.EXE*
3. Run *tlist /t* again, *PSXSS.EXE* is child of *SMSS.EXE*



OS/2 environment subsystem

Limited in usefulness:

- only OS/2 1.2 16-bit char-based or video I/O (VIO) apps
- only on *x*86 systems.

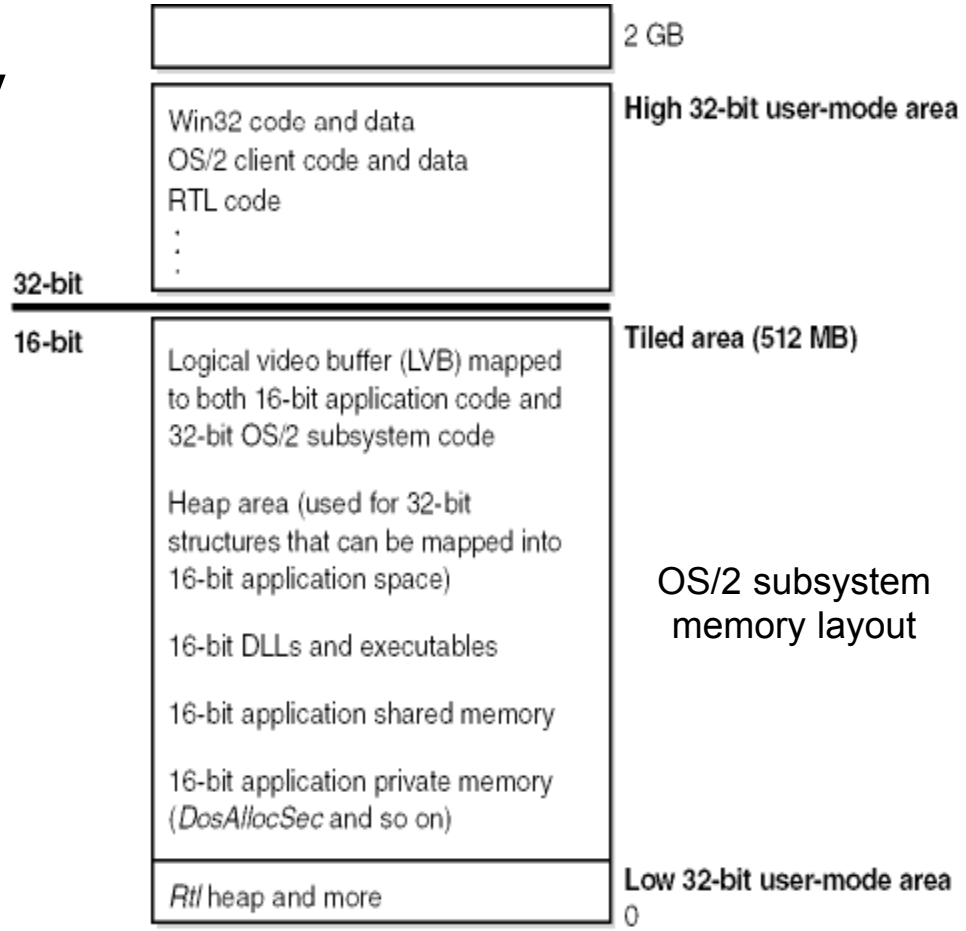
Add-on OS/2 1.2 PresentationManager

- can't run OS/2 2.*x* (or later) applications.

- 64 Priority levels are mapped on NT 0..15 (no RT)
- OS/2 subsystem starts automatically

OS/2 Memory

- Up to 512 MB memory for OS/2 app.
  - Virtual addr space is reserved up front
  - Commit/decommit on request of 16 bit OS/2 apps



OS/2 subsystem memory layout

NTDLL.DLL

Support library for use of subsystem DLLs:

- System service dispatch stubs to NT executive system services
  - NtCreateFile, NtSetEvent
  - More than 200
  - Most of them are accessible through Win32

  Stubs call service-dispatcher/kernel-mode service in NTOSKRNL.EXE
- Support functions used by subsystems
  - Image loader (Ldr...)
  - Heap manager
  - Win32 subsyst. Comm. func. (Csr...)
  - Runtime library func. (Rtl...)
  - User-mode asynch. procedure call (APC) dispatcher, exception disp.

Executive

Upper layer of NTOSKRNL.EXE (kernel: lower layer)

Contains:

- Exported func., callable through NTDLL.DLL, Win32...
- Exported func., not currently available though subsyst
  - LPCs, query functions: NtQueryInformationxxx
  - Specialized functions: NtCreatePagingFile
- Doc. functions callable from kernel mode, NT DDK
- Internal support routines

Executive components

- Process and thread manager
- Virtual memory manager
- Security reference monitor: protection/auditing
- I/O system: device independent I/O
- Cache manager: uses mem.manag. - mapped files
- Object manager: processes, threads, synch. objects
- LPC facility: flexible, optimized version of DCE RPC
- Run-time library: math, string, data types
- Support routines: syst. Mem. Alloc., paged/nonpaged

Kernel

Most fundamental operations in NT

- Thread scheduling and dispatching
- Trap handling and exception dispatching
- Interrupt handling and dispatching
- Multiprocessor synchronization
- Base kernel objects for executive

Never paged out of memory

Never preempted

Small, compact, portable, efficient: C, assembly lang.

  - no probes for parameter accessibility
  - Some functions documented in DDK (Ke...)

Kernel objects

- Little overhead, small, efficient
- Control objects:
  - Kernel process object
  - Asynchronous procedure call object
  - Deferred procedure call object
  - Interrupt object
- Dispatcher objects
  - Synchronization objects
  - Kernel thread, mutex (mutant), kernel event pair, semaphor, timer, waitable timer,
- Kernel supports set of interfaces that are portable and semantically identical accross architectures

Small amount of x86-specific interfaces to support old MS-DOS programs:

GDT/LDT are x86 HW specific

Hardware Abstraction Layer

Loadable kernel module (HAL.DLL)

- Low-level interface to NT hardware platform
- Hides I/O interface, interrupt controllers, MP comm.
  - Architecture-specific, machine-dependent details
- Device driver call HAL routines for platform-dep. Info
- Only one HAL.DLL is installed
  - Many HAL*.DLL on distribution media
  - VMS may choose HAL at boot time

Device Drivers

- Loadable kernel modules
- Don‘t manipulate hardware, but call parts of HAL
  - Written in C/C++ typically
  - Source code portable accross CPU architectures !!

Types:

- Hardware device drivers: implement device/network I/O
- File system drivers: file I/O <-> device I/O
- Filter drivers: disk mirroring, encryption
- Network redirectors and servers: send/receive remote I/O requests

List Drivers

- Control Panel -> Devices: installed drivers
- DRIVERS.EXE / pstat: loaded drivers

```
D:\home> drivers
  ModuleName    Code    Data     Bss   Paged    Init          LinkDate
-------------------------------------------------------------------------
ntoskrnl.exe  270272   40064       0  434816   82880  Sun May 11 05:10:39 1997
     hal.dll   20384    2720       0    9344   11936  Mon Mar 10 21:39:20 1997
   atapi.sys   22368    1088       0       0     768  Sat Apr 04 00:06:15 1998
SCSIPORT.SYS    9792      32       0   15840    2208  Sat Apr 04 00:05:43 1998
  CPQSPM.sys    4896      64       0       0     544  Thu Feb 05 14:39:28 1998
    Disk.sys    3328       0       0    7072    1600  Fri Apr 25 03:27:46 1997
  CLASS2.SYS    7040       0       0    1632    1152  Fri Apr 25 03:23:43 1997
 ScsiPwr.sys    8576    1248       0       0       0  Mon Sep 09 11:39:25 1996
    Ntfs.sys   68160    5408       0  269632    8704  Fri Apr 18 03:02:31 1997
  Floppy.SYS    1088     672       0    7968    6112  Wed Jul 17 05:31:09 1996
   Cdrom.SYS   12608      32       0    3072    3104  Wed Jul 17 05:31:29 1996
  Fs_Rec.SYS      64       0       0    2912    1152  Mon Mar 10 21:51:19 1997
    Null.SYS       0       0       0     288     416  Wed Jul 17 05:31:21 1996
  KSecDD.SYS    1280     224       0    3456    1024  Thu Jul 18 01:34:19 1996
    Beep.SYS    1184       0       0       0     704  Wed Apr 23 20:19:43 1997
```

Startup: Session Manager (SMSS)

First user-mode process (kernel calls ExInitializeSystem)

1. Creates LPC port (\SmApiPort); waits for load-subsystem/create session client requests; 2 threads
2. Creates system environment variables
3. Defines symbolic links for MS-DOS device names (COM1, LPT1)
4. Creates addtional paging files
5. Opens known DLLs (efficiency; re-use of pages)
6. Loads kernel-mode part of Win32 subsystem (WIN32K.SYS)
7. Starts subsystem processes (POSIX, OS/2 start on demand)
8. Starts logon process (WINLOGON)
9. Creates LPC ports for debug event messages (DbgSsApiPort, DbgUiApiPort) and threads to listen on these ports

waits for CSRSS & WINLOGON; **crashes NT on termination**

WINLOGON

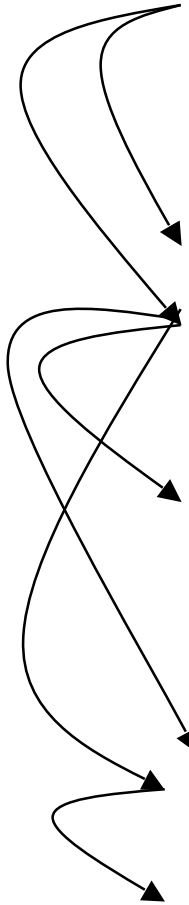
- Secure attention sequence (SAS) keystroke
  - Protection from password-capture programs that simulate logon
  - Default sequence: CTRL-ALT-DEL
- User/pass sent to local security authentication server
- USERINIT.EXE is created -> starts shell and exits
  - Default: explorer.exe
- Identification/authentication in replaceable DLL
  - GINA (Graph. Id. And Auth.) – default: MSGINA:DLL
  - WINLOGON can load add. Network provider DLLs (secondary Auth.)
- WINLOGON remains active
  - NT Security dialog box on SAS keystroke
  - Demo: how do I change my password?

Local Security Authentication Server (LSASS)

- Receives requests from WINLOGON
- Calls authentication package (DLL)
- Performs verification (SAM – part of registry)

  (Security Accounts Manager)
- LSASS generates access token object
  - Contains user‘s security profile
- WINLOGON creates initial shell using this token
  - Child processes inherit access token (default)

System Process Tree



smss.exe	Session Manager The first “created” process ;Takes parameters from \Registry\Machine\System\CurrentControlSet\Control\ Session Manager Launches required subsystems (csrss) and then winlogon
csrss.exe	Win32 subsystem
winlogon.exe	Logon process: Launches services.exe, lsass.exe, and nddeagnt.exe; presents first login prompt; presents “enter username and password” dialog When someone logs in, launches userinit.exe
services.exe	Service Controller; also, home for many Windows NT-supplied services Starts processes for services not part of services.exe (driven by \Registry\Machine\System\ CurrentControlSet\Services)
lsass.exe	Local Security Authentication Server
userinit.exe	Started after logon; starts desktop (Explorer.Exe) and exits (hence does not show up in tlist output; Explorer appears to be an orphan)
explorer.exe	and its children are the creators of all interactive apps

Service Controller (SERVICES)

- Service process or device driver ?
- User mode service processes:
  - Like UNIX „deamon processes“ / VMS „detached processes“
  - Automatic start at system startup
  - Manual start: Win32 StartService, ControlPanel->Services
- W2K components as services:
  - Spooler, event log, RPC support,
  - networking components
- Services have 3 names:
  - Process name, registry name, display name

```
WINLOGON.EXE (34)
  SERVICES.EXE (40)
    SPOOLSS.EXE (70)
    daccess.exe (82)
    CPQALERT.EXE (85)
    dssvc.exe (97)
    mgasc.exe (103)
      mgactrl.exe (107)
    RPCSS.EXE (109)
      AGENTSVR.EXE (313)
    TCPSVCS.EXE (131)
```