

Windows 2000 System Architecture

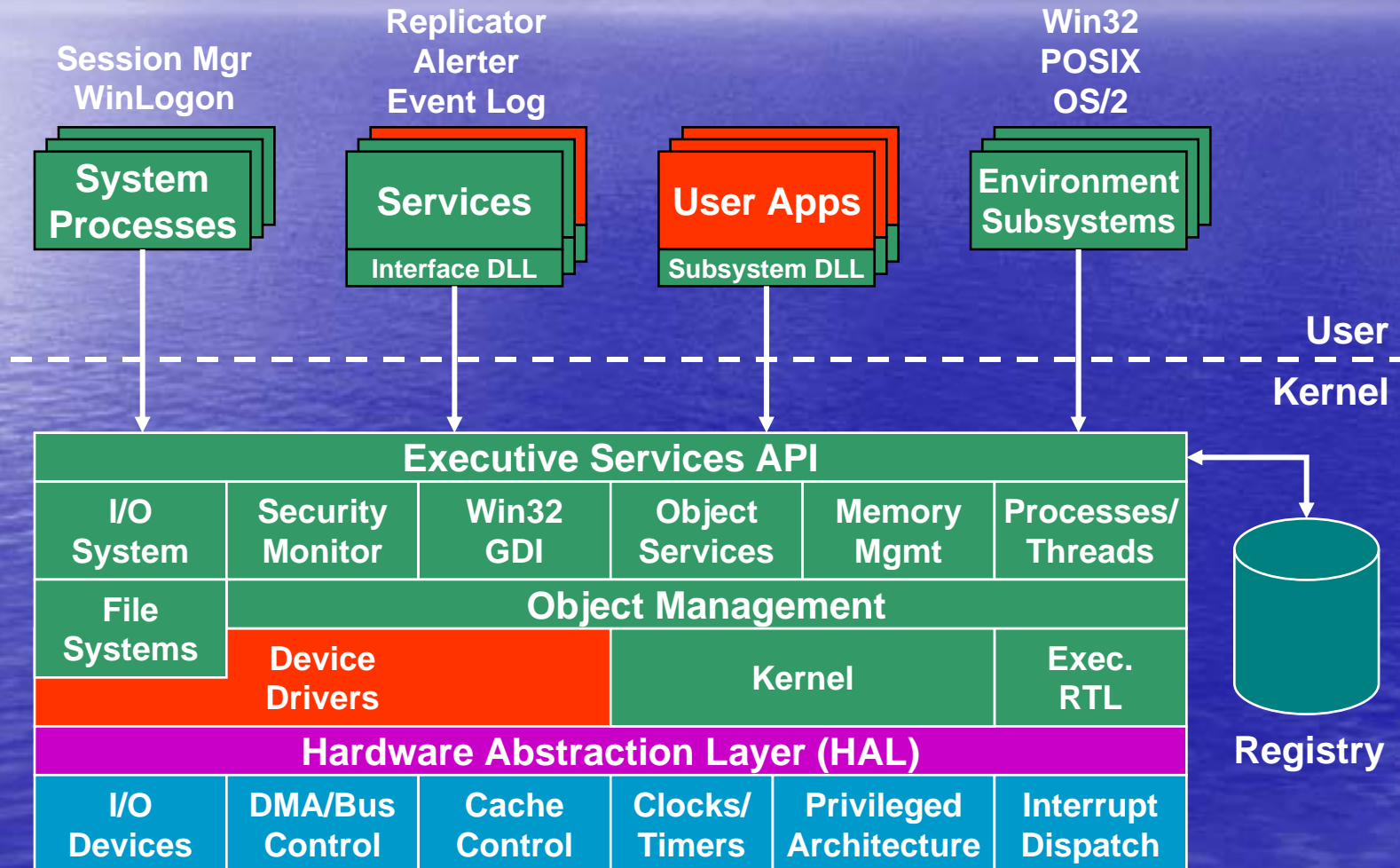
Computing Department,
Lancaster University, UK

Overview

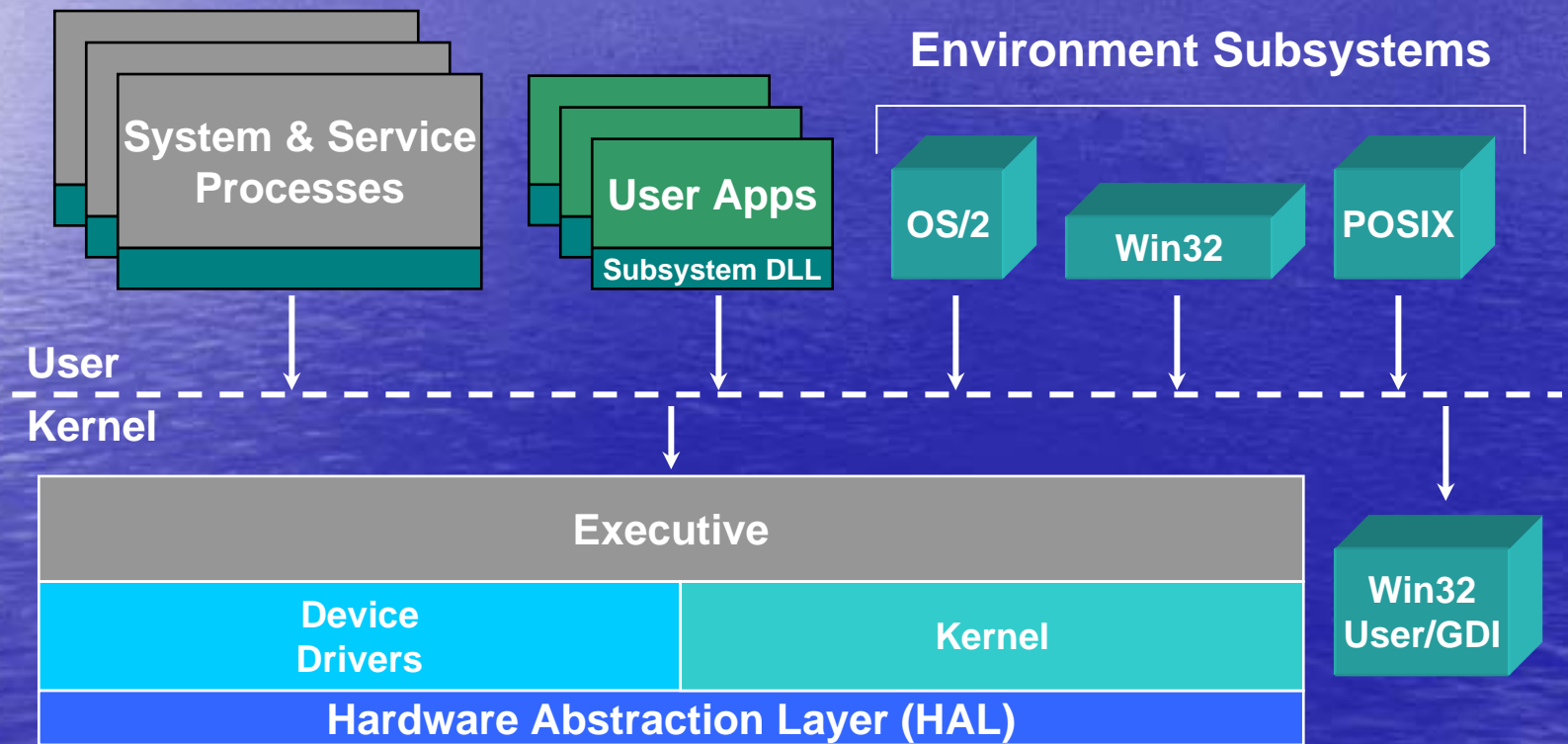
- Goals
 - Explain internal architecture and operation of core Windows 2000 components
 - Discuss the role of subsystems and their operation

System Architecture

Windows 2000 Architecture



Windows 2000 Simplified Architecture



Architecture Components (1)

- Windows 2000 Executive
 - Upper layers of the operating system
 - Provides “generic operating system” functions
 - Creating/deleting processes and threads
 - Memory management
 - I/O
 - Interprocess communication
 - Security
 - Executes in kernel mode
 - API not documented!
 - Accessed indirectly via subsystem APIs

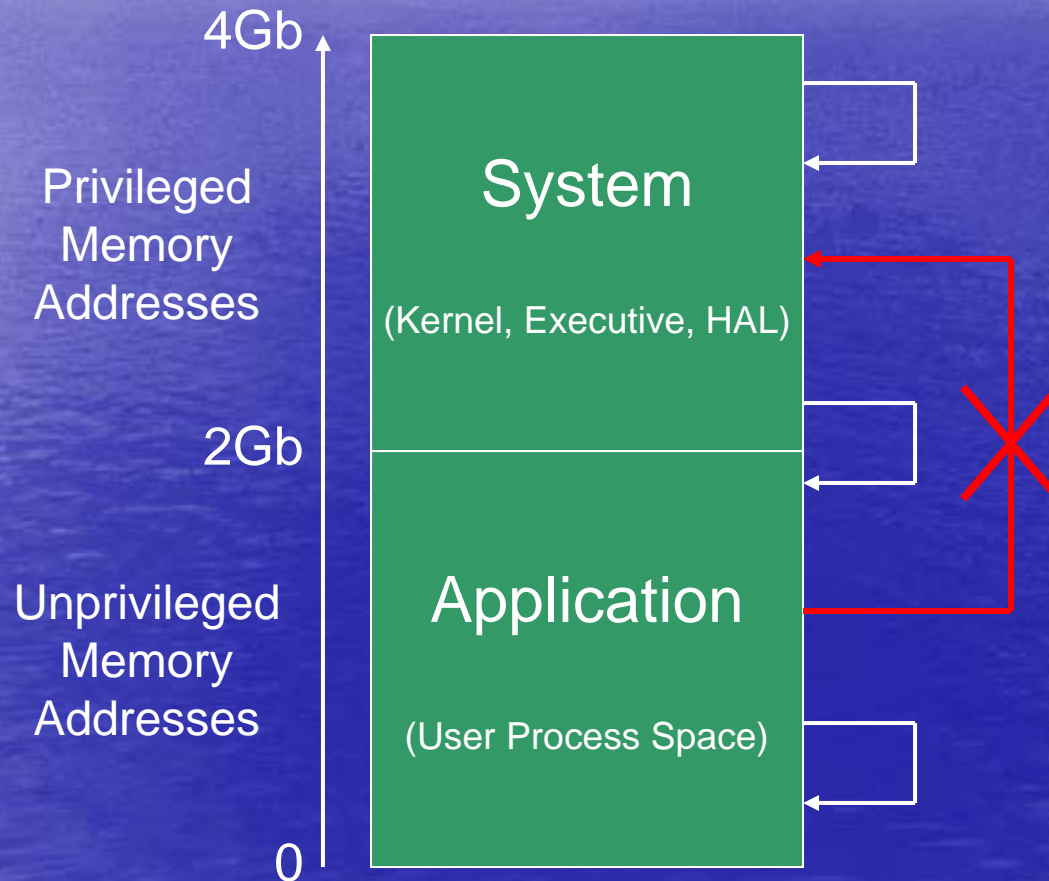
Architecture Components (2)

- Windows 2000 Kernel
 - Low level operating system functions
 - Thread scheduling
 - Interrupt and exception dispatching
 - Multiprocessor synchronisation
- HAL (Hardware Abstraction Layer)
 - Isolates Kernel and Executive from platform specific details
 - E.g. differences between motherboards
 - Presents uniform model of I/O hardware interface to drivers

Architecture Components (3)

- Device Drivers
 - Loadable Kernel modules that interface between I/O manager and relevant hardware
 - File system drivers, network protocol drivers, hardware device drivers
 - Drivers call HAL functions to interface with hardware
 - View loaded drivers within Computer Management
- Win32 User/GDI (Graphics Device Interface)
 - Implements graphical user interface (GUI)
 - Window manipulation, user interface controls
 - Drawing

Memory Layout in Windows 2000

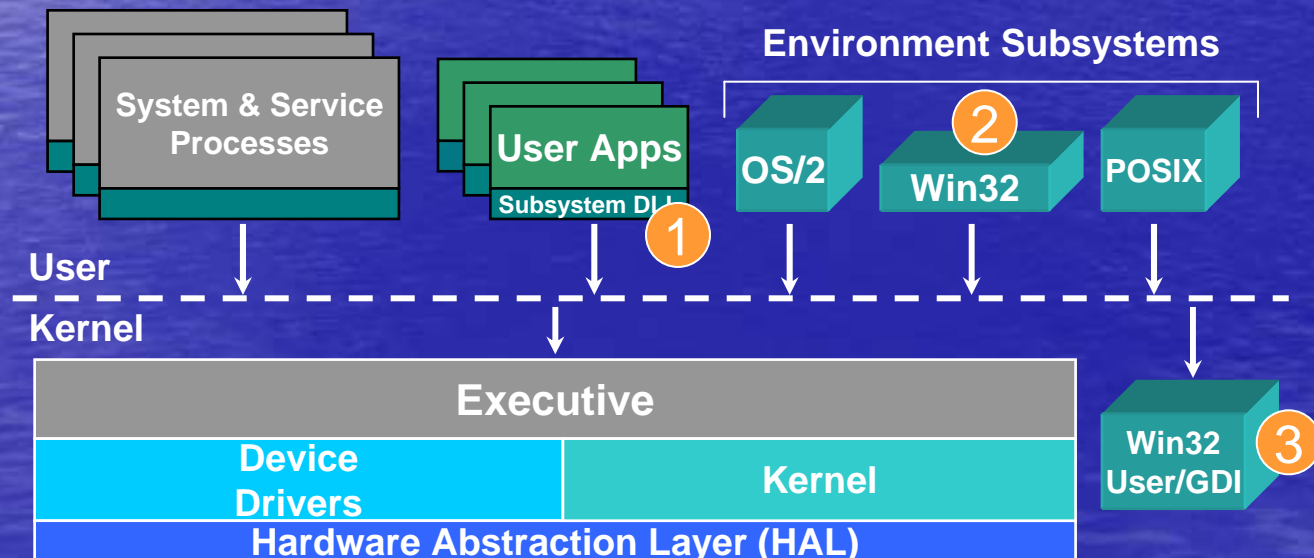


Environment Subsystems

- Three environment subsystems provided with Windows 2000:
 - OS/2
 - POSIX (POSIX 1003.1)
 - Win32
- Provide exposed, documented interface between application and 2000 native API
 - Each subsystem defines a different set of APIs
- Executables are bound to one subsystem only!
- Function calls can't be mixed between subsystems
- User applications don't call Windows 2000 system services directly – they go via subsystem DLLs

Environment Subsystem Components

- 1 API DLLs
 - For Win32: Kernel32.DLL, Gdi32.DLL, User32.DLL
- 2 Subsystem process
 - For Win32: CSRSS.EXE
- 3 Kernel-Mode GDI Code (Win32 only)



Role of Subsystem Components

- API DLLs
 - Export APIs defined by the subsystem
 - Implement them by calling “native” services, or by asking the subsystem process to do the work
- Subsystem process
 - Maintains global state of subsystem
- Win32K.SYS
 - Implements Win32 User/GDI functions
 - Also used by POSIX & OS/2 subsystems to access the display

Win32 Subsystem

- Implemented in CSRSS.EXE process (Client/Server Run-time SubSystem)
 - Supports console (text) windows
 - Creating and deleting Win32 processes/threads
- Kernel mode driver WIN32K.SYS contains
 - Window Manager
 - Keyboard/mouse input, screen output, window displays
 - Graphics Device Interface (GDI)
- Subsystem DLLs map Win32 API functions onto kernel-mode system service calls
 - USER32.DLL, KERNEL32.GDI -> NTOSKRNL

POSIX Subsystem

- **P**ortable **O**perating **S**ystem **I**nterface based on **U**nix
 - Encourages compatibility – aids application porting
 - Included to meet US Government requirements
 - Windows 2000 implements POSIX 1003.1
 - Provides limited set of services
 - Can't create threads, windows or use sockets!
 - Executables linked against POSIX subsystem library (Psxdll.dll)
 - Commercial Unix-to-Win32 library better approach for porting UNIX applications

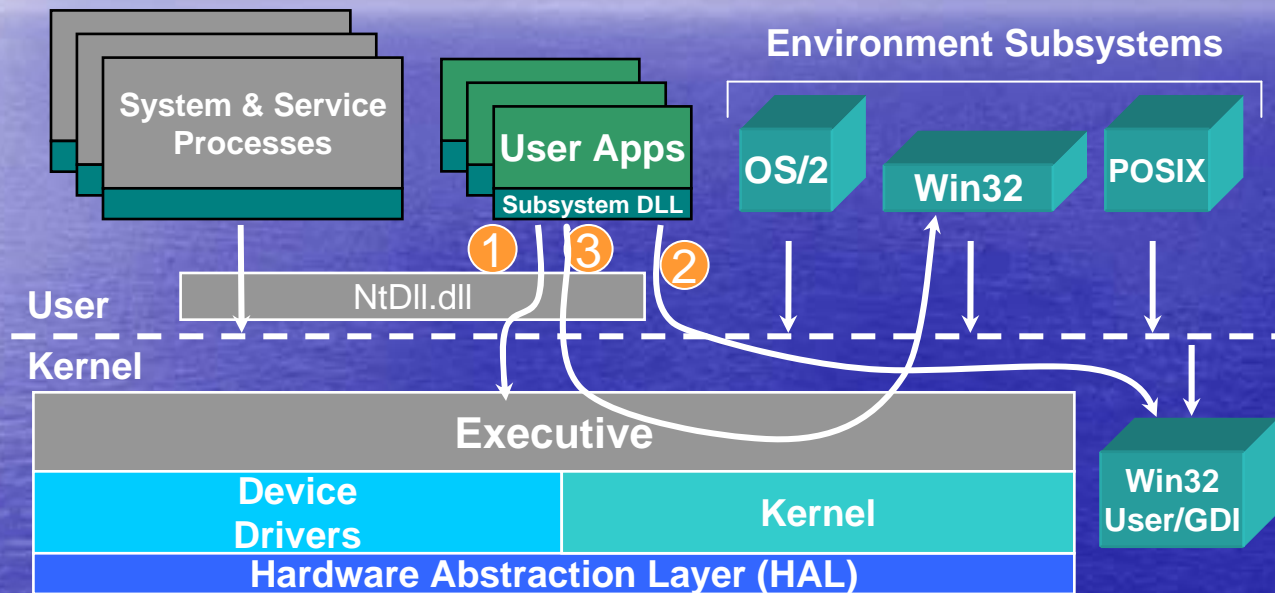
OS/2 Subsystem

- Limited in usefulness
- Supports OS/2 1.2 16-bit character & video I/O (VIO) applications
- OS/2 applications attempting to access hardware directly aren't supported
- Both OS/2 and POSIX subsystems start automatically the first time an associated application is started

Environment Subsystems

- Provide exposed, documented interface between applications and native APIs
- Each subsystem defines a set of APIs
 - Implemented by invoking native APIs
 - Subsystem “wraps up” native APIs
 - Example: Win32 Kernel32.dll invokes kernel/base services in Ntdll.dll
- When application calls subsystem DLL function, one of the following occurs:
 - Function implemented completely in user mode inside subsystem DLL (no Win2k system services called)
 - Function calls Windows 2000 Executive
 - Function requires work to be done in subsystem process (sends messages to subsystem)

Subsystem Function Paths



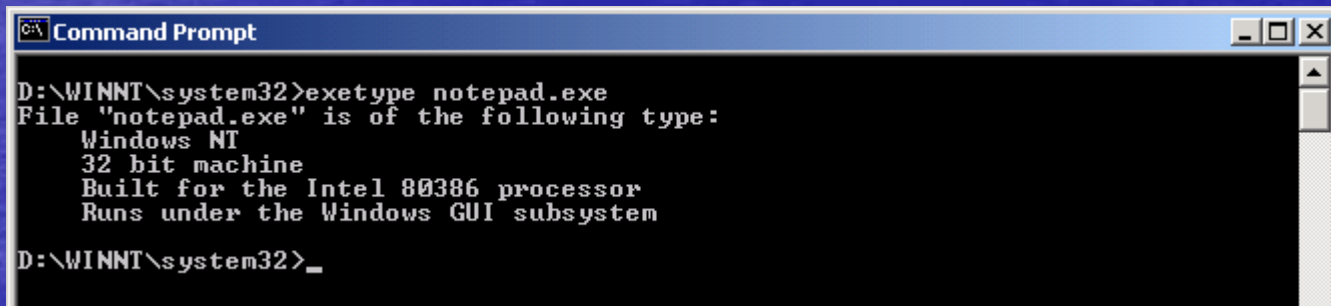
- ① Most Win32 Kernel APIs
- ② Most Win32 User/GDI APIs
- ③ A Few Win32 APIs

Image Headers

- Subsystem for each .exe specified in image header

IMAGE_SUBSYSTEM_UNKNOWN	0	Unknown Subsystem
IMAGE_SUBSYSTEM_NATIVE	1	Image doesn't require subsystem
IMAGE_SUBSYSTEM_WINDOWS_GUI	2	Win32 subsystem (graphical app)
IMAGE_SUBSYSTEM_WINDOWS_CUI	3	Win32 subsystem (character app)
IMAGE_SUBSYSTEM_OS2_CUI	5	OS/2 subsystem
IMAGE_SUBSYSTEM_POSIX_CUI	7	POSIX subsystem

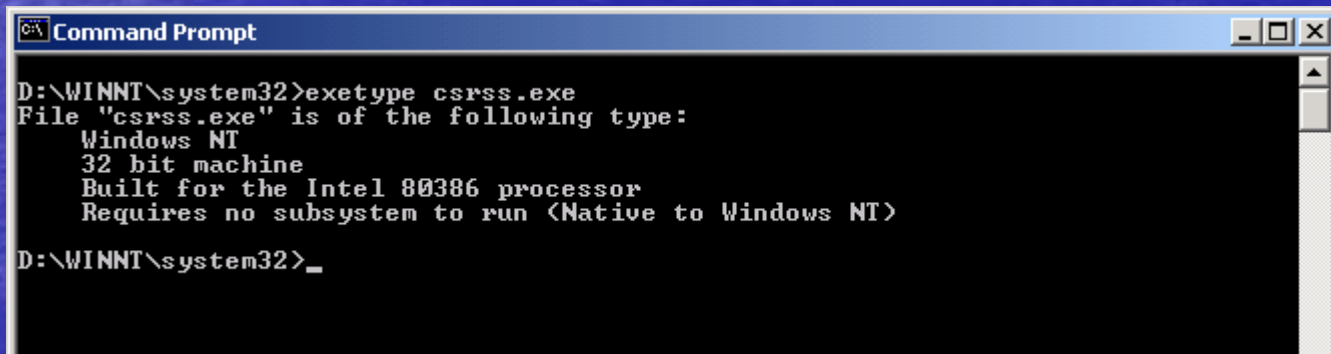
Use `exetype` (from Resource Kit) to view header:



```
Command Prompt
D:\WINNT\system32>exetype notepad.exe
File "notepad.exe" is of the following type:
  Windows NT
  32 bit machine
  Built for the Intel 80386 processor
  Runs under the Windows GUI subsystem
D:\WINNT\system32>_
```

Native Images

- .EXE not linked against any subsystem
 - Interfaces to Executive directly via NTDLL.DLL
- Two examples
 - Smss.exe (Session Manager)
 - Csrss.exe (Win32 subsystem)



```
Command Prompt
D:\WINNT\system32>exetype csrss.exe
File "csrss.exe" is of the following type:
  Windows NT
  32 bit machine
  Built for the Intel 80386 processor
  Requires no subsystem to run (Native to Windows NT)
D:\WINNT\system32>_
```

Experimenting with POSIX



- Locate POSIX applications on Windows 2000 Resource Kit
 - Open command prompt
 - Look at subsystem type for one of the images (e.g. "exetype ls.exe")
 - Type "ls ls.exe" and compare with "ls LS.EXE"
 - Why does one work?
 - Check that POSIX subsystem process is now running (psxss.exe)