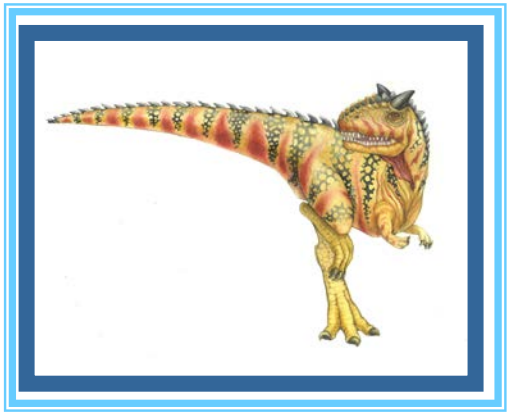
Chapter 2: System Structures



Chapter 2: System Structures

- Operating System Services
- User Operating System Interface
- System Calls
- Types of System Calls
- System Programs
- Operating System Design and Implementation
- Operating System Structure
- Virtual Machines
- Operating System Debugging
- Operating System Generation
- System Boot

Objectives

- To describe the services an operating system provides to users, processes, and other systems
- To discuss the various ways of structuring an operating system
- To explain how operating systems are installed and customized and how they boot

Operating System Services

- One set of operating-system services provides functions that are helpful to the user:
  - User interface - Almost all operating systems have a user interface (UI)
    - Varies between Command-Line (CLI), Graphics User Interface (GUI), Batch
  - Program execution - The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
  - I/O operations - A running program may require I/O, which may involve a file or an I/O device
  - File-system manipulation - The file system is of particular interest. Obviously, programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.

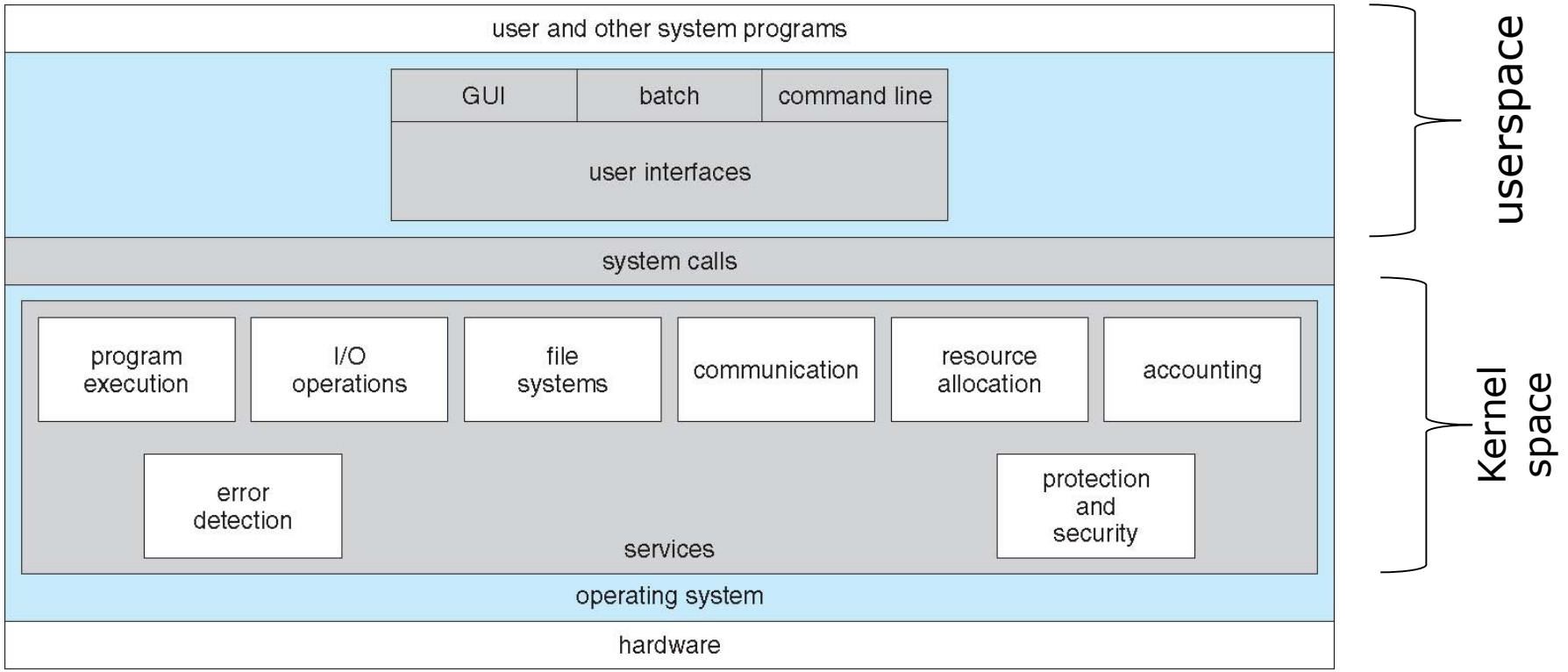
Operating System Services (Cont)

- One set of operating-system services provides functions that are helpful to the user (Cont):
  - Communications – Processes may exchange information, on the same computer or between computers over a network
    - Communications may be via shared memory or through message passing (packets moved by the OS)
  - Error detection – OS needs to be constantly aware of possible errors
    - May occur in the CPU and memory hardware, in I/O devices, in user program
    - For each type of error, OS should take the appropriate action to ensure correct and consistent computing
    - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

Operating System Services (Cont)

- Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing
  - **Resource allocation -** When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
    - Many types of resources - Some (such as CPU cycles, main memory, and file storage) may have special allocation code, others (such as I/O devices) may have general request and release code
  - **Accounting -** To keep track of which users use how much and what kinds of computer resources
  - **Protection and security -** The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other
    - **Protection** involves ensuring that all access to system resources is controlled
    - **Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts
    - If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link.

A View of Operating System Services



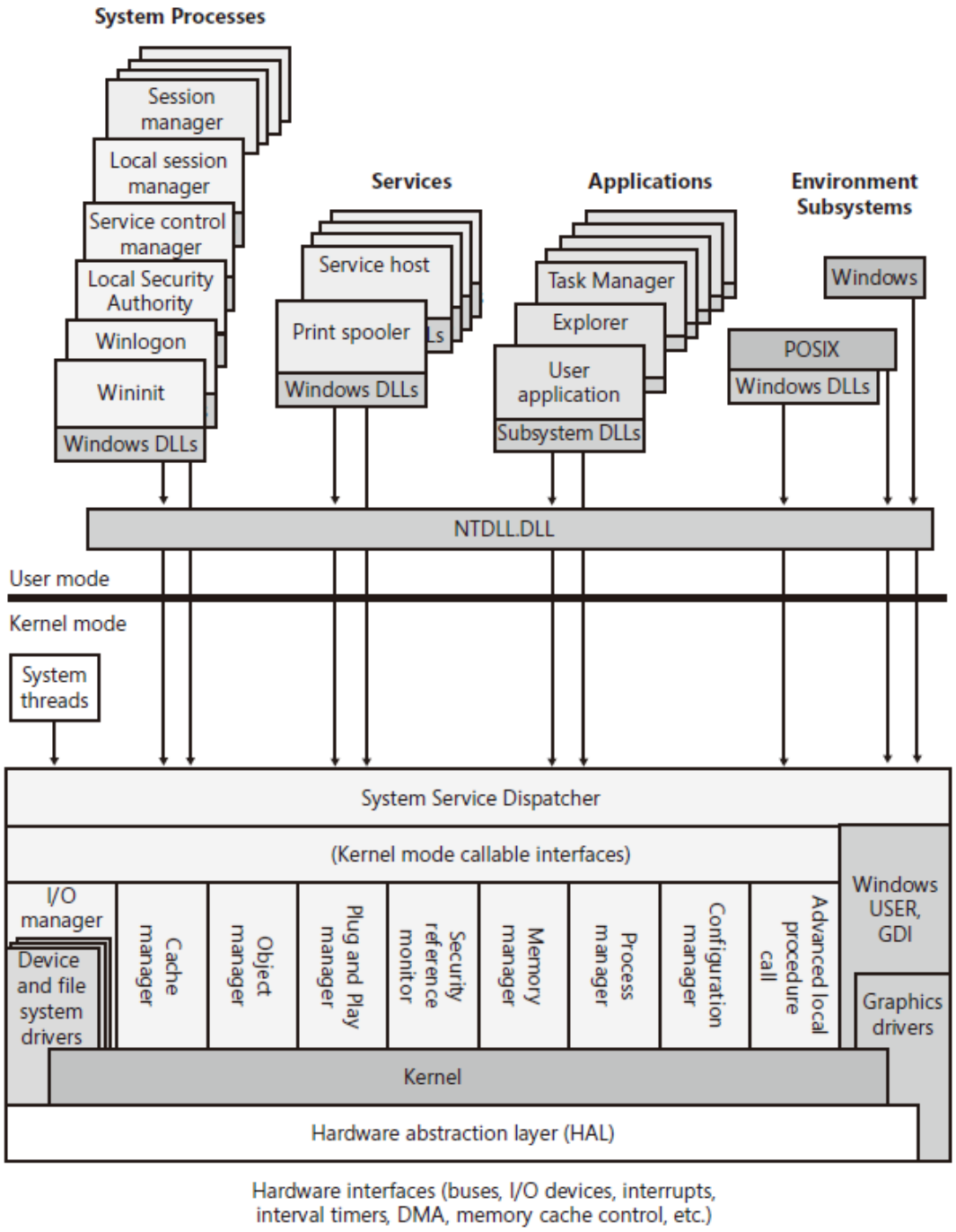
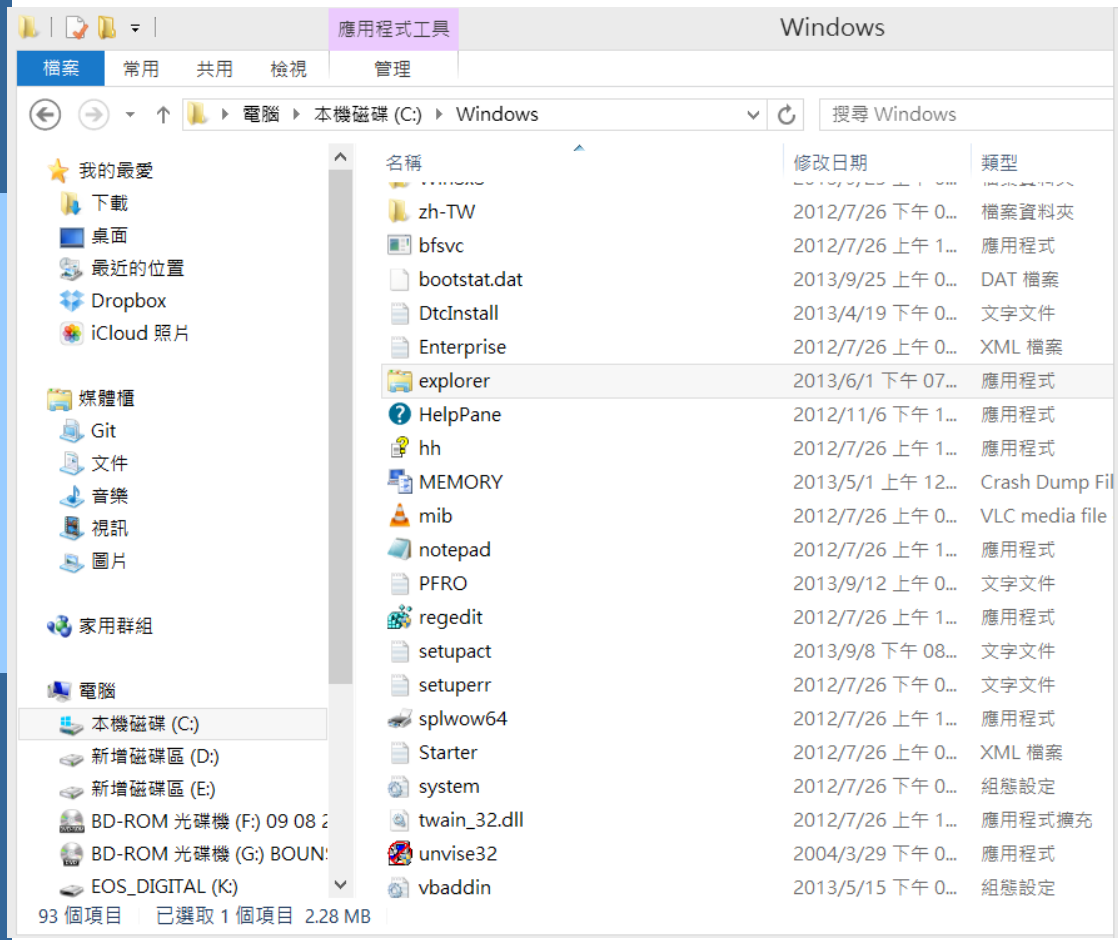


FIGURE 2-3 Windows architecture

System Programs

- System programs provide a convenient environment for program development and execution. They can be divided into:
  - File manipulation
  - Status information
  - File modification
  - Programming language support
  - Program loading and execution
  - Communications
  - Application programs
- Most users' view of the operation system is defined by system programs, not the actual system calls

System Programs



System Programs

linux1.cs.nctu.edu.tw - PuT

```
 3. For rights of other users, please don't
    please use (re)nice/taskset/cpuset to l
cesses,
    and please use ipcrm to clear shared me
= Disk Usage ================================
 Mail:

 Home:

= Process ====================================
  PID TTY          TIME CMD
25288 pts/132  00:00:00 csShell
25366 pts/132  00:00:00 ps
= Information ================================
 Current Time: Wed Sep 25 20:58:55 CST 2013
 Online Users: 29
= CSCC Announce ==============================

                                CS Compute
linux1 [/u/faculty/ysw] -ysw- %
linux1 [/u/faculty/ysw] -ysw- %
linux1 [/u/faculty/ysw] -ysw- % passwd
Changing password for ysw.
(current) UNIX password:
```

linux1.cs.nctu.edu.tw - PuTTY

```
linux1 [/u/faculty/ysw] -ysw- % which passwd
/usr/bin/passwd
linux1 [/u/faculty/ysw] -ysw- % ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 47808 Jan 27  2013 /usr/bin/passwd
linux1 [/u/faculty/ysw] -ysw- %
```

linux1.cs.nctu.edu.tw - PuTTY

```
linux1 [/u/faculty/ysw] -ysw- % ps -ef|grep passwd
root     25425 25288  0 20:59 pts/132  00:00:00 passwd
ysw      25634 25533  0 21:01 pts/138  00:00:00 grep passwd
linux1 [/u/faculty/ysw] -ysw- %
```

System Programs

linux1.cs.nctu.edu.tw - PuTTY

```
top - 21:04:28 up 205 days,  8:27, 30 users,  load average: 0.00, 0.01, 0.05
Tasks: 493 total,   1 running, 489 sleeping,   0 stopped,   3 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni, 99.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   8170640 total,  1937204 used,  6233436 free,   332016 buffers
KiB Swap:  2097144 total,  1022384 used,  1074760 free,   318944 cached

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
25656 ysw       20   0   23200   1980   1208 R   0.7  0.0   0:00.10 top
12603 pyhsu     20   0   53888    920     72 S   0.3  0.0 303:01.43 mosh-server
13104 pyhsu     20   0   52324    728     60 S   0.3  0.0 291:17.01 mosh-server
    1 root      20   0   39144   2964   1228 S   0.0  0.0   0:50.61 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:03.31 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   8:49.93 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    7 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/u:0H
    8 root      rt   0       0      0      0 S   0.0  0.0   0:02.48 migration/0
    9 root      20   0       0      0      0 S   0.0  0.0  73:16.12 rcu_preempt
   10 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
   11 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_sched
   12 root      rt   0       0      0      0 S   0.0  0.0   3:55.84 watchdog/0
   13 root      rt   0       0      0      0 S   0.0  0.0   0:43.63 watchdog/1
   14 root      20   0       0      0      0 S   0.0  0.0   5:12.49 ksoftirqd/1
   15 root      rt   0       0      0      0 S   0.0  0.0   0:01.18 migration/1
   17 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:0H
   18 root      rt   0       0      0      0 S   0.0  0.0   0:42.95 watchdog/2
   19 root      20   0       0      0      0 S   0.0  0.0   3:43.19 ksoftirqd/2
```

System Programs

- Provide a convenient environment for program development and execution
  - Some of them are simply user interfaces to system calls; others are considerably more complex
- File management - Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories
- Status information
  - Some ask the system for info - date, time, amount of available memory, disk space, number of users
  - Others provide detailed performance, logging, and debugging information
  - Typically, these programs format and print the output to the terminal or other output devices
  - Some systems implement a registry - used to store and retrieve configuration information

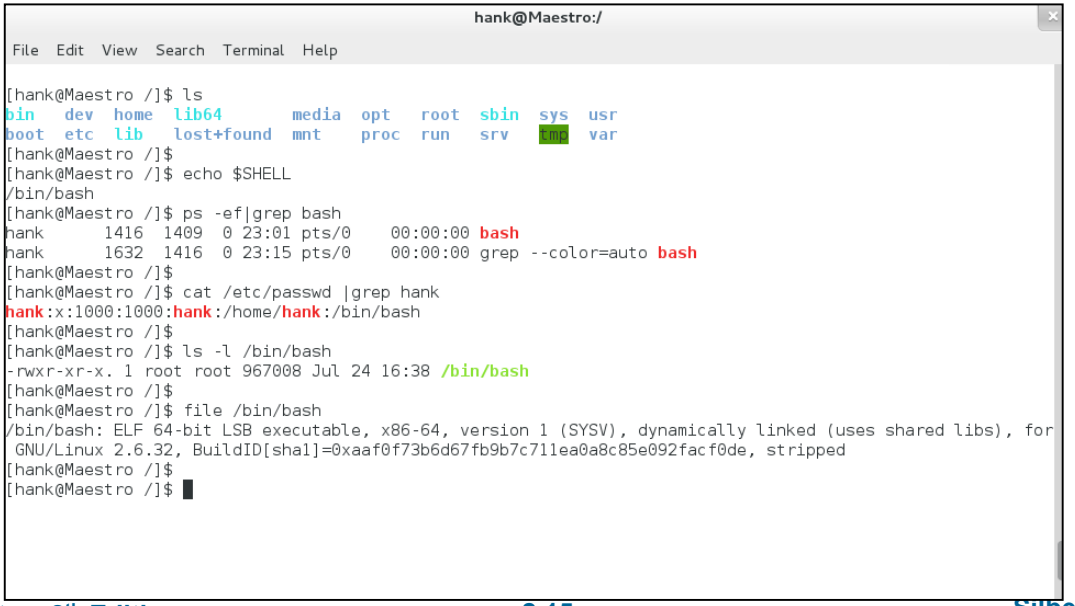
System Programs (cont'd)

- File modification
  - Text editors to create and modify files
  - Special commands to search contents of files or perform transformations of the text
- Programming-language support - Compilers, assemblers, debuggers and interpreters sometimes provided
- Program loading and execution- Absolute loaders, relocatable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level and machine language
- Communications - Provide the mechanism for creating virtual connections among processes, users, and computer systems
  - Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another

User Operating System Interface - CLI

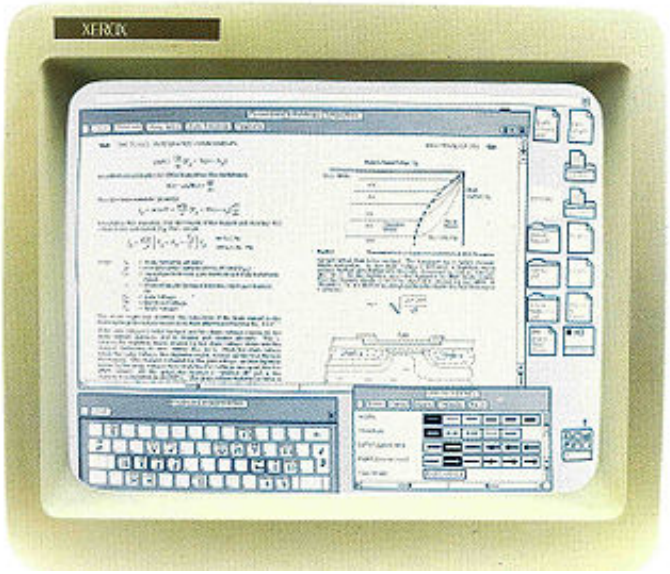
Command Line Interface (CLI) or command interpreter allows direct command entry

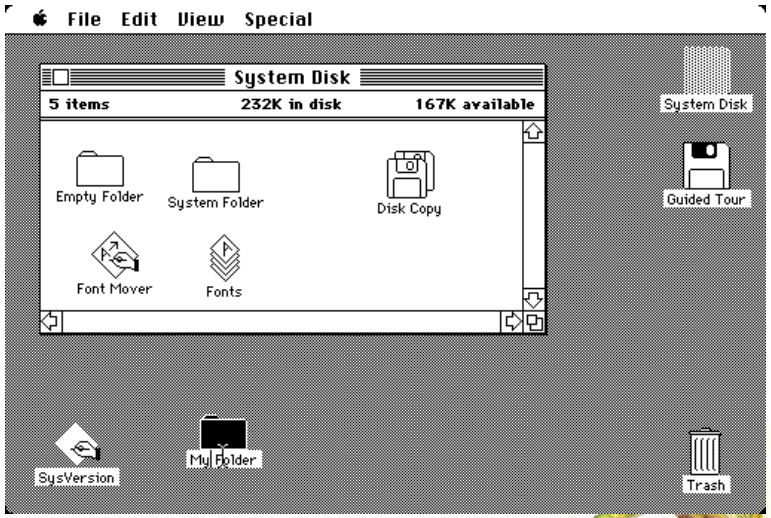
- Sometimes implemented in kernel, sometimes by systems program
- Sometimes multiple flavors implemented – shells
- Primarily fetches a command from user and executes it
  - Sometimes commands built-in, sometimes just names of programs
  - If the latter, adding new features doesn't require shell modification



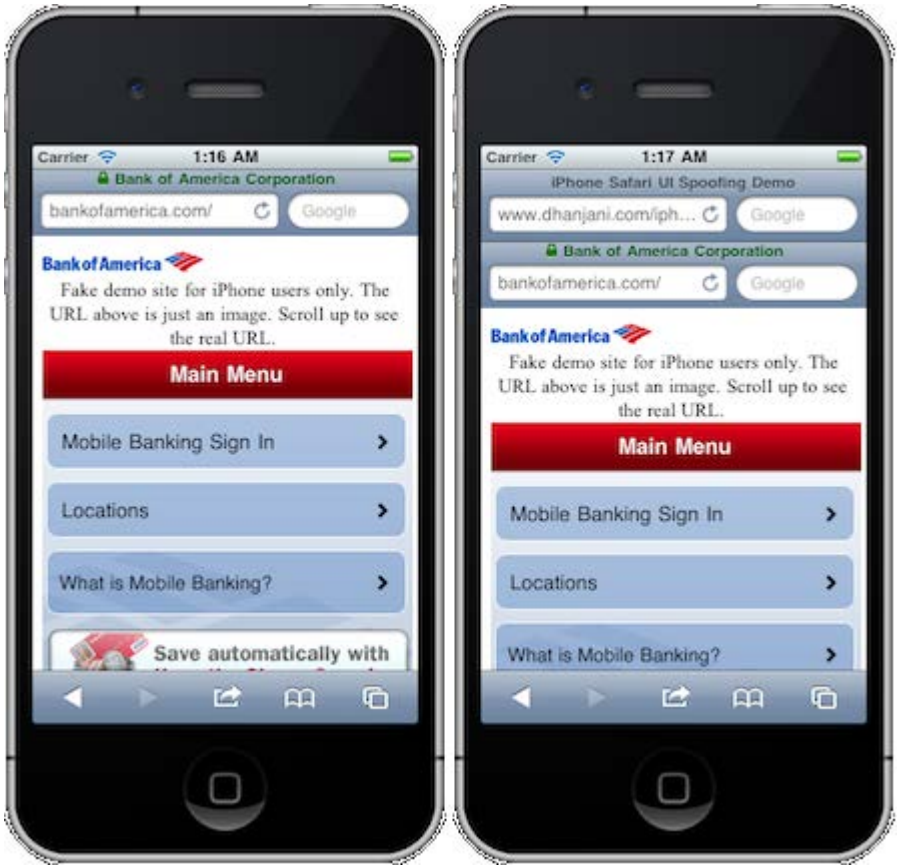
User Operating System Interface - GUI

- User-friendly desktop metaphor interface
  - Usually mouse, keyboard, and monitor
  - Icons represent files, programs, actions, etc
  - Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory (known as a folder)
  - Invented at Xerox PARC
- Many systems now include both CLI and GUI interfaces
  - Microsoft Windows is GUI with CLI "command" shell
  - Apple Mac OS X as "Aqua" GUI interface with UNIX kernel underneath and shells available
  - Solaris is CLI with optional GUI interfaces (Java Desktop, KDE)

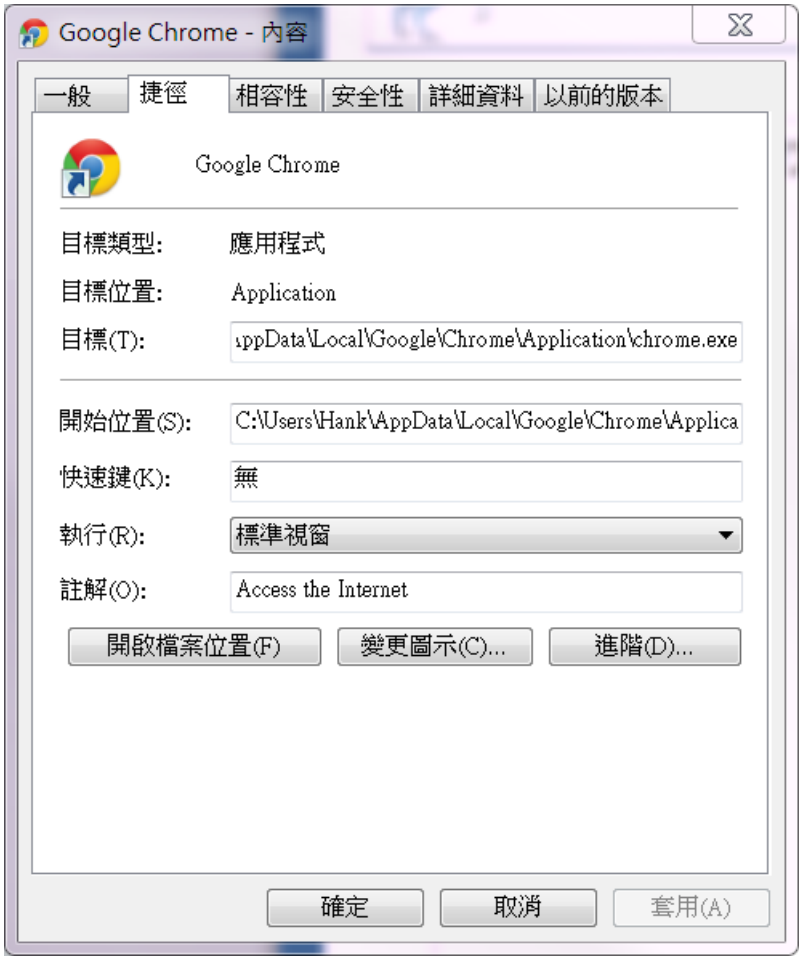


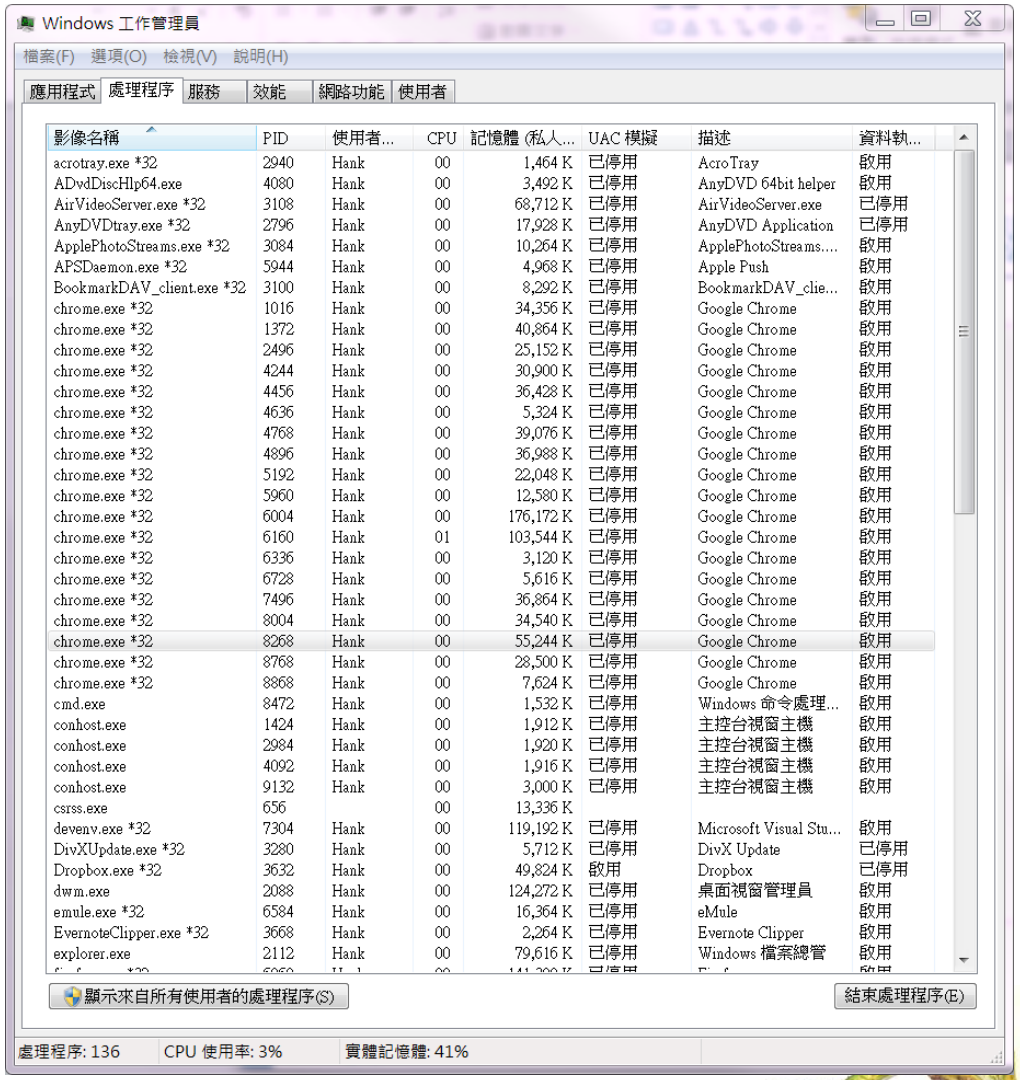


UI Spoofing


Carrier
1:16 AM
Bank of America Corporation
bankofamerica.com/
Google
Bank of America
Fake demo site for iPhone users only. The URL above is just an image. Scroll up to see the real URL.
Main Menu
Mobile Banking Sign In
Locations
What is Mobile Banking?
Save automatically with
Carrier
1:17 AM
iPhone Safari UI Spoofing Demo
www.dhanjani.com/iph...
Google
Bank of America Corporation
bankofamerica.com/
Google
Bank of America
Fake demo site for iPhone users only. The URL above is just an image. Scroll up to see the real URL.
Main Menu
Mobile Banking Sign In
Locations
What is Mobile Banking?

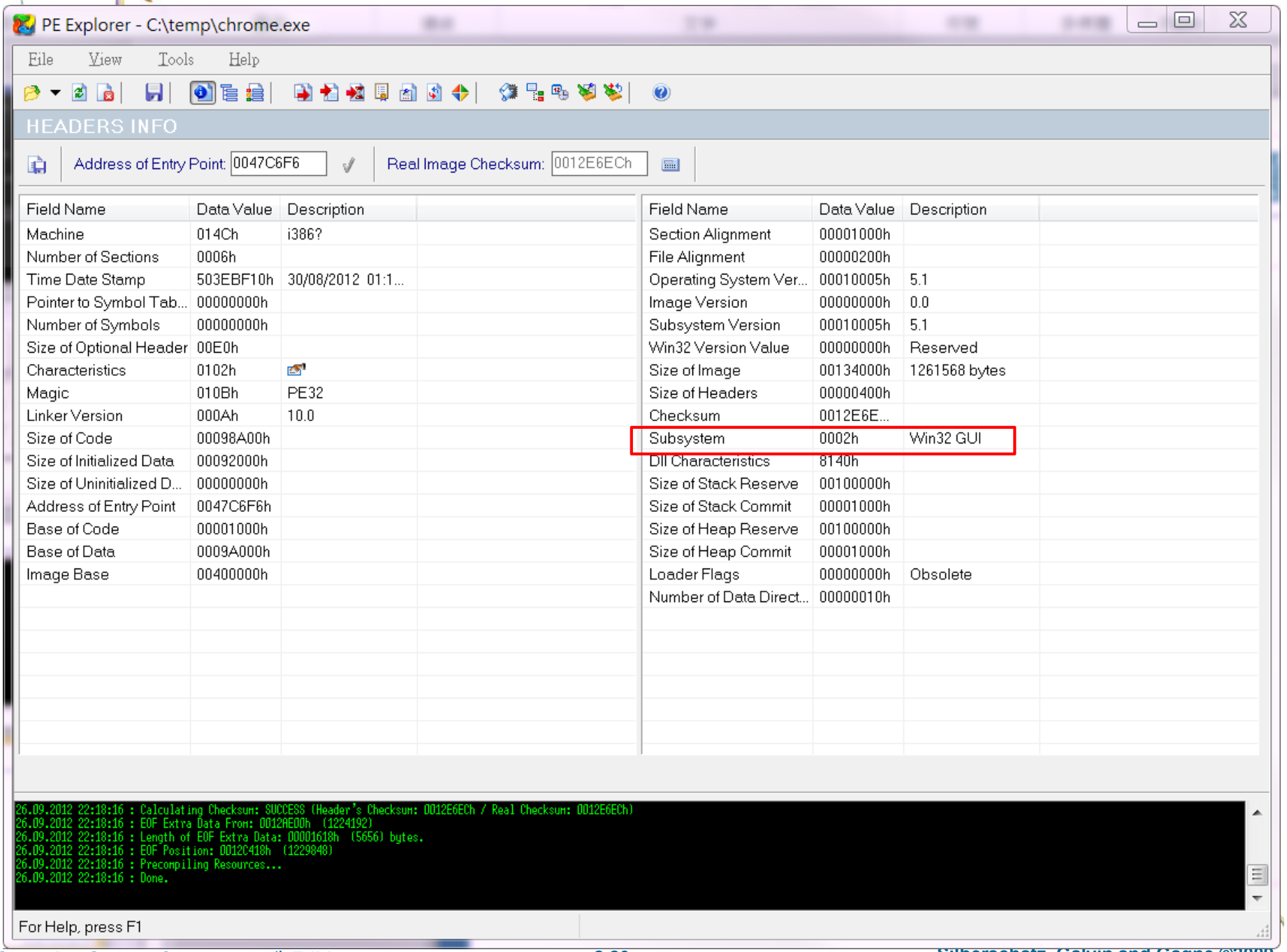
Program Execution

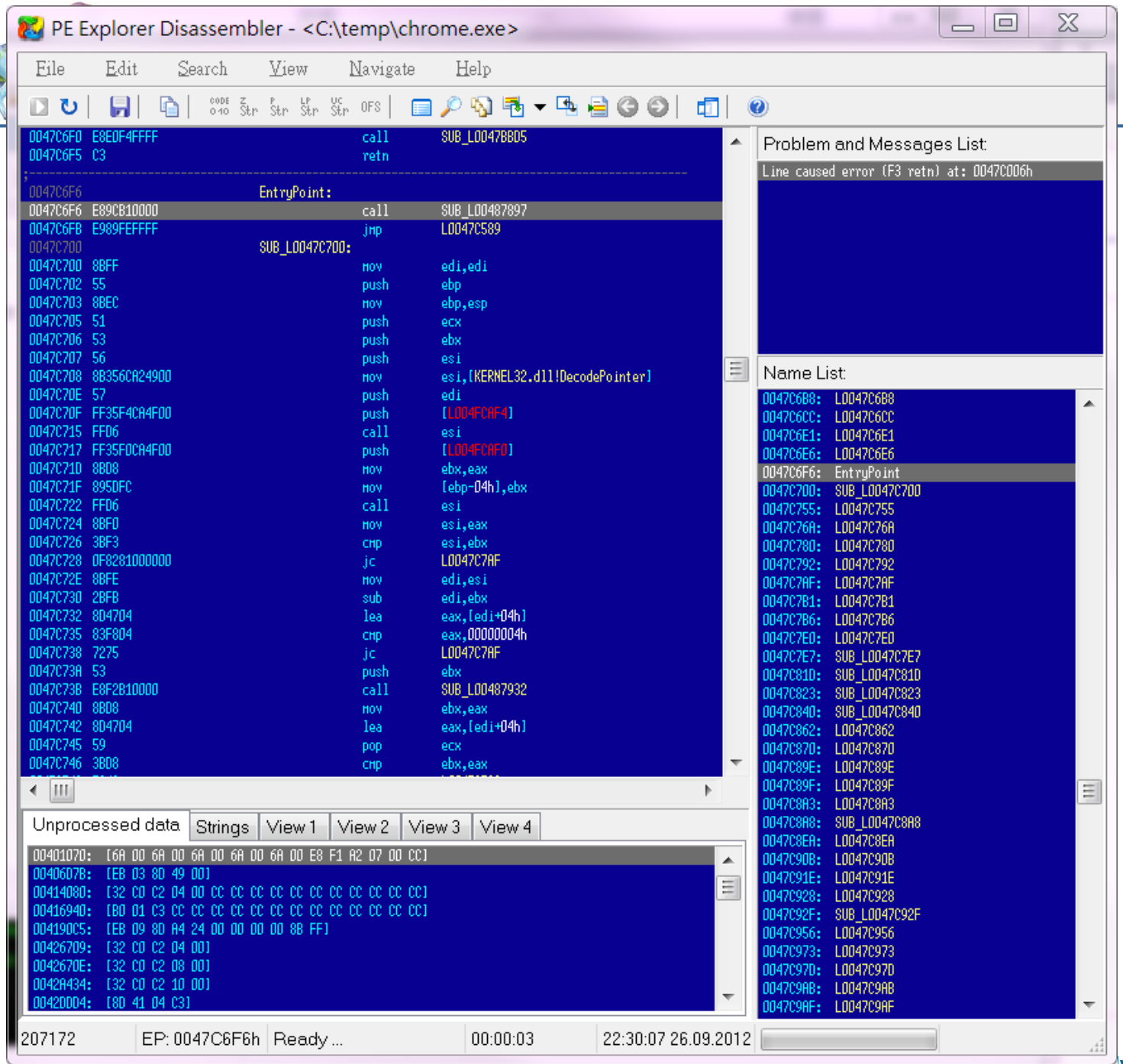


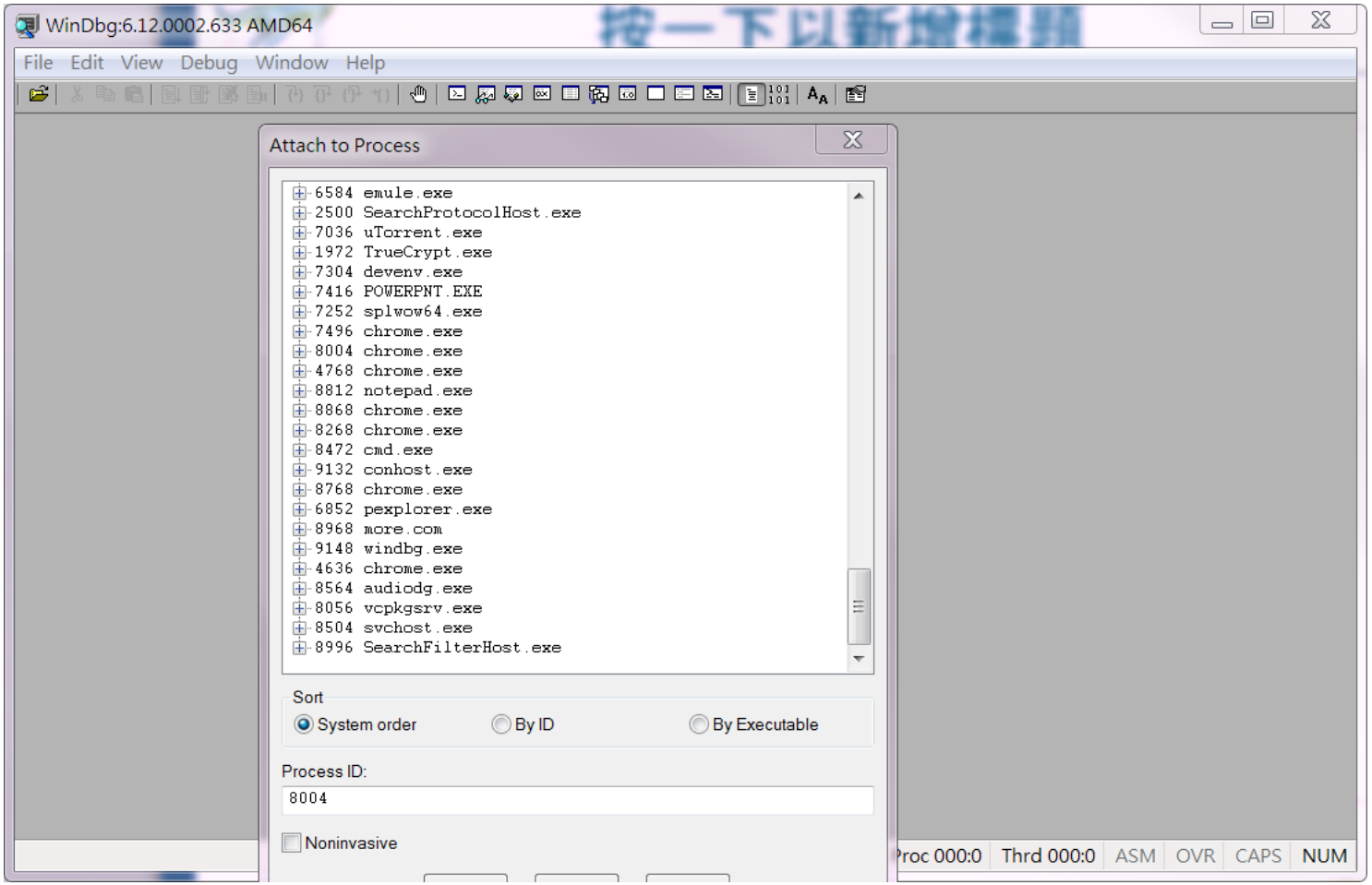


C:\Windows\system32\cmd.exe

```
C:\temp>dumpbin /headers chrome.exe| more
Microsoft (R) COFF/PE Dumper Version 10.00.40219.01
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file chrome.exe

PE signature found

File Type: EXECUTABLE IMAGE

FILE HEADER VALUES
             14C machine (x86)
               6 number of sections
        503EBF10 time date stamp Thu Aug 30 09:17:04 2012
               0 file pointer to symbol table
               0 number of symbols
              E0 size of optional header
             102 characteristics
                   Executable
                   32 bit word machine

OPTIONAL HEADER VALUES
             10B magic # (PE32)
           10.00 linker version
           98A00 size of code
           92000 size of initialized data
               0 size of uninitialized data
           7C6F6 entry point (0047C6F6)
            1000 base of code
           9A000 base of data
          400000 image base (00400000 to 00533FFF)
            1000 section alignment
             200 file alignment
            5.01 operating system version
            0.00 image version
            5.01 subsystem version
               0 Win32 version
          134000 size of image
             400 size of headers
          12E6EC checksum
               2 subsystem (Windows GUI)
            8140 DLL characteristics
                   Dynamic base
                   NX compatible
                   Terminal Server Aware
          100000 size of stack reserve
            1000 size of stack commit
```


PE Explorer - C:\temp\chrome.exe
File View Tools Help
HEADERS INFO
Address of Entry Point: 0047C6F6
Real Image Checksum: 0012E6ECh
Field Name | Data Value | Description
Machine | 014Ch | i386?
Number of Sections | 0006h
Time Date Stamp | 503EBF10h | 30/08/2012 01:1...
Pointer to Symbol Tab... | 00000000h
Number of Symbols | 00000000h
Size of Optional Header | 00E0h
Characteristics | 0102h
Magic | 010Bh | PE32
Linker Version | 000Ah | 10.0
Size of Code | 00098A00h
Size of Initialized Data | 00092000h
Size of Uninitialized D... | 00000000h
Address of Entry Point | 0047C6F6h
Base of Code | 00001000h
Base of Data | 0009A000h
Image Base | 00400000h
Field Name | Data Value | Description
Section Alignment | 00001000h
File Alignment | 00000200h
Operating System Ver... | 00010005h | 5.1
Image Version | 00000000h | 0.0
Subsystem Version | 00010005h | 5.1
Win32 Version Value | 00000000h | Reserved
Size of Image | 00134000h | 1261568 bytes
Size of Headers | 00000400h
Checksum | 0012E6E...
Subsystem | 0002h | Win32 GUI
Dll Characteristics | 8140h
Size of Stack Reserve | 00100000h
Size of Stack Commit | 00001000h
Size of Heap Reserve | 00100000h
Size of Heap Commit | 00001000h
Loader Flags | 00000000h | Obsolete
Number of Data Direct... | 00000010h
26.09.2012 22:18:16 : Calculating Checksum: SUCCESS (Header's Checksum: 0012E6ECh / Real Checksum: 0012E6ECh)
26.09.2012 22:18:16 : EOF Extra Data From: 0012AED0h (1224192)
26.09.2012 22:18:16 : Length of EOF Extra Data: 00001618h (5656) bytes.
26.09.2012 22:18:16 : EOF Position: 0012C418h (1229848)
26.09.2012 22:18:16 : Precompiling Resources...
26.09.2012 22:18:16 : Done.
For Help, press F1


PE Explorer Disassembler - <C:\temp\chrome.exe>
File Edit Search View Navigate Help
0047C6F0 E8EDF4FFFF call SUB_L0047BBD5
0047C6F5 C3 retn
0047C6F6 EntryPoint:
0047C6F6 E89CB10000 call SUB_L00487897
0047C6FB E989FEFFFF jmp L0047C589
0047C700 SUB_L0047C700:
0047C700 8BFF mov edi,edi
0047C702 55 push ebp
0047C703 8BEC mov ebp,esp
0047C705 51 push ecx
0047C706 53 push ebx
0047C707 56 push esi
0047C708 8B356CA24900 mov esi,[KERNEL32.dll!DecodePointer]
0047C70E 57 push edi
0047C70F FF35F4CA4F00 push [L004FCAF4]
0047C715 FFD6 call esi
0047C717 FF35F0CA4F00 push [L004FCAF0]
0047C71D 8BD8 mov ebx,eax
0047C71F 895DFC mov [ebp-04h],ebx
0047C722 FFD6 call esi
0047C724 8BF0 mov esi,eax
0047C726 3BF3 cmp esi,ebx
0047C728 0F8281000000 jc L0047C7AF
0047C72E 8BFE mov edi,esi
0047C730 2BFB sub edi,ebx
0047C732 8D4704 lea eax,[edi+04h]
0047C735 83F804 cmp eax,00000004h
0047C738 7275 jc L0047C7AF
0047C73A 53 push ebx
0047C73B E8F2B10000 call SUB_L00487932
0047C740 8BD8 mov ebx,eax
0047C742 8D4704 lea eax,[edi+04h]
0047C745 59 pop ecx
0047C746 3BD8 cmp ebx,eax
Problem and Messages List:
Line caused error (F3 retn) at: 0047C006h
Name List:
0047C6B8: L0047C6B8
0047C6CC: L0047C6CC
0047C6E1: L0047C6E1
0047C6E6: L0047C6E6
0047C6F6: EntryPoint
0047C700: SUB_L0047C700
0047C755: L0047C755
0047C76A: L0047C76A
0047C780: L0047C780
0047C792: L0047C792
0047C7AF: L0047C7AF
0047C7B1: L0047C7B1
0047C7B6: L0047C7B6
0047C7E0: L0047C7E0
0047C7E7: SUB_L0047C7E7
0047C81D: SUB_L0047C81D
0047C823: SUB_L0047C823
0047C84D: SUB_L0047C84D
0047C862: L0047C862
0047C87D: L0047C87D
0047C89E: L0047C89E
0047C89F: L0047C89F
0047C8A3: L0047C8A3
0047C8A8: SUB_L0047C8A8
0047C8EA: L0047C8EA
0047C90B: L0047C90B
0047C91E: L0047C91E
0047C928: L0047C928
0047C92F: SUB_L0047C92F
0047C956: L0047C956
0047C973: L0047C973
0047C97D: L0047C97D
0047C9AB: L0047C9AB
0047C9AF: L0047C9AF
Unprocessed data Strings View 1 View 2 View 3 View 4
00401070: [6A 00 6A 00 6A 00 6A 00 6A 00 E8 F1 A2 07 00 CC]
00406D7B: [EB 03 8D 49 00]
00414080: [32 C0 C2 04 00 CC CC CC CC CC CC CC CC CC CC CC]
00416940: [B0 01 C3 CC CC CC CC CC CC CC CC CC CC CC CC CC]
004190C5: [EB 09 8D A4 24 00 00 00 00 8B FF]
00426709: [32 C0 C2 04 00]
0042670E: [32 C0 C2 08 00]
00428434: [32 C0 C2 10 00]
0042DDD4: [8D 41 04 C3]
207172 EP: 0047C6F6h Ready ... 00:00:03 22:30:07 26.09.2012


WinDbg:6.12.0002.633 AMD64
File Edit View Debug Window Help
Attach to Process
6584 emule.exe
2500 SearchProtocolHost.exe
7036 uTorrent.exe
1972 TrueCrypt.exe
7304 devenv.exe
7416 POWERPNT.EXE
7252 splwow64.exe
7496 chrome.exe
8004 chrome.exe
4768 chrome.exe
8812 notepad.exe
8868 chrome.exe
8268 chrome.exe
8472 cmd.exe
9132 conhost.exe
8768 chrome.exe
6852 pexplorer.exe
8968 more.com
9148 windbg.exe
4636 chrome.exe
8564 audiodg.exe
8056 vcpkgsrv.exe
8504 svchost.exe
8996 SearchFilterHost.exe
Sort
System order
By ID
By Executable
Process ID:
8004
Noninvasive
Proc 000:0
Thrd 000:0
ASM
OVR
CAPS
NUM

Pid 4456 - WinDbg:6.12.0002.633 AMD64

File Edit View Debug Window Help

Command

```
0:006> !peb
****************************************************************************
***                                                                      ***
***                                                                      ***
***    Your debugger is not using the correct symbols                    ***
***                                                                      ***
***    In order for this command to work properly, your symbol path      ***
***    must point to .pdb files that have full type information.         ***
***                                                                      ***
***    Certain .pdb files (such as the public OS symbols) do not         ***
***    contain the required information.  Contact the group that         ***
***    provided you with these symbols if you need this command to       ***
***    work.                                                             ***
***                                                                      ***
***    Type referenced: wow64!TEB32                                      ***
***                                                                      ***
****************************************************************************
PEB at 000000007efdf000
    InheritedAddressSpace:    No
    ReadImageFileExecOptions: No
    BeingDebugged:            Yes
    ImageBaseAddress:         00000000013e0000
    Ldr                       00000000770f2640
    Ldr.Initialized:          Yes
    Ldr.InInitializationOrderModuleList: 0000000000364c20 . 00000000003651d0
    Ldr.InLoadOrderModuleList:           0000000000364b10 . 0000000000365570
    Ldr.InMemoryOrderModuleList:         0000000000364b20 . 0000000000365580
            Base TimeStamp                     Module
         13e0000 503ebf10 Aug 30 09:17:04 2012 C:\Users\Hank\AppData\Local\Google\Chrome\Application\chrome.exe
        76fc0000 4ec4aa8e Nov 17 14:32:46 2011 C:\Windows\SYSTEM32\ntdll.dll
        74820000 4e212272 Jul 16 13:32:34 2011 C:\Windows\SYSTEM32\wow64.dll
        747c0000 4e212275 Jul 16 13:32:37 2011 C:\Windows\SYSTEM32\wow64win.dll
        747b0000 4e212273 Jul 16 13:32:35 2011 C:\Windows\SYSTEM32\wow64cpu.dll
```

0:006>

Ln 0, Col 0 | Sys 0:<Local> | Proc 000:1168 | Thrd 006:1c4c | ASM | OVR | CAPS | NUM

Pid 4456 - WinDbg:6.12.0002.633 AMD64

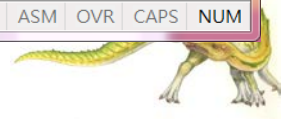
File Edit View Debug Window Help

Command

```
0:006> !dh 0013e0000

File Type: EXECUTABLE IMAGE
FILE HEADER VALUES
     14C machine (i386)
       6 number of sections
503EBF10 time date stamp Thu Aug 30 09:17:04 2012

       0 file pointer to symbol table
       0 number of symbols
      E0 size of optional header
     102 characteristics
            Executable
            32 bit word machine

OPTIONAL HEADER VALUES
     10B magic #
   10.00 linker version
   98A00 size of code
   92000 size of initialized data
       0 size of uninitialized data
   7C6F6 address of entry point
    1000 base of code
         ----- new -----
00000000013e0000 image base
    1000 section alignment
     200 file alignment
       2 subsystem (Windows GUI)
    5.01 operating system version
    0.00 image version
    5.01 subsystem version
  134000 size of image
     400 size of headers
```

0:006>

Ln 0, Col 0 | Sys 0:<Local> | Proc 000:1168 | Thrd 006:1c4c | ASM | OVR | CAPS | NUM



Pid 4456 - WinDbg:6.12.0002.633 AMD64

File Edit View Debug Window Help

Command

```
0:006> u 0013e0000+7C6F6 L30
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\Use
chrome!SetExperimentList2+0x57554:
00000000`0145c6f6 e89cb10000      call    chrome!SetExperimentList2+0x626f5 (00000
00000000`0145c6fb e989feffff      jmp     chrome!SetExperimentList2+0x573e7 (00000
00000000`0145c700 8bff            mov     edi,edi
00000000`0145c702 55              push    rbp
00000000`0145c703 8bec            mov     ebp,esp
00000000`0145c705 51              push    rcx
00000000`0145c706 53              push    rbx
00000000`0145c707 56              push    rsi
00000000`0145c708 8b356ca24701    mov     esi,dword ptr [chrome!SetExperimentList2
00000000`0145c70e 57              push    rdi
00000000`0145c70f ff35f4ca4d01    push    qword ptr [chrome!SetExperimentList2+0xd
00000000`0145c715 ffd6            call    rsi
00000000`0145c717 ff35f0ca4d01    push    qword ptr [chrome!SetExperimentList2+0xd
00000000`0145c71d 8bd8            mov     ebx,eax
00000000`0145c71f 895dfc          mov     dword ptr [rbp-4],ebx
00000000`0145c722 ffd6            call    rsi
00000000`0145c724 8bf0            mov     esi,eax
00000000`0145c726 3bf3            cmp     esi,ebx
00000000`0145c728 0f8281000000    jb      chrome!SetExperimentList2+0x5760d (00000
00000000`0145c72e 8bfe            mov     edi,esi
00000000`0145c730 2bfb            sub     edi,ebx
00000000`0145c732 8d4704          lea     eax,[rdi+4]
00000000`0145c735 83f804          cmp     eax,4
00000000`0145c738 7275            jb      chrome!SetExperimentList2+0x5760d (00000
00000000`0145c73a 53              push    rbx
00000000`0145c73b e8f2b10000      call    chrome!SetExperimentList2+0x62790 (00000
00000000`0145c740 8bd8            mov     ebx,eax
00000000`0145c742 8d4704          lea     eax,[rdi+4]
00000000`0145c745 59              pop     rcx
00000000`0145c746 3bd8            cmp     ebx,eax
0:006>
```

PE Explorer Disassembler - <C:\temp\chrome.exe>

File Edit Search View Navigate Help

```
0047C6F0 E8EDF4FFFF              call    SUB_L0047BBD5
0047C6F5 C3                      retn
;
0047C6F6                 EntryPoint:
0047C6F6 E89CB10000              call    SUB_L00487897
0047C6FB E989FEFFFF              jmp     L0047C589
0047C700                 SUB_L0047C700:
0047C700 8BFF                    mov     edi,edi
0047C702 55                      push    ebp
0047C703 8BEC                    mov     ebp,esp
0047C705 51                      push    ecx
0047C706 53                      push    ebx
0047C707 56                      push    esi
0047C708 8B356CA24900            mov     esi,[KERNEL32.dll!DecodePointer]
0047C70E 57                      push    edi
0047C70F FF35F4CA4F00            push    [L004FCAF4]
0047C715 FFD6                    call    esi
0047C717 FF35F0CA4F00            push    [L004FCAF0]
0047C71D 8BD8                    mov     ebx,eax
0047C71F 895DFC                  mov     [ebp-04h],ebx
0047C722 FFD6                    call    esi
0047C724 8BF0                    mov     esi,eax
0047C726 3BF3                    cmp     esi,ebx
0047C728 0F8281000000            jc      L0047C7AF
0047C72E 8BFE                    mov     edi,esi
0047C730 2BFB                    sub     edi,ebx
0047C732 8D4704                  lea     eax,[edi+04h]
0047C735 83F804                  cmp     eax,00000004h
0047C738 7275                    jc      L0047C7AF
0047C73A 53                      push    ebx
0047C73B E8F2B10000              call    SUB_L00487932
0047C740 8BD8                    mov     ebx,eax
0047C742 8D4704                  lea     eax,[edi+04h]
0047C745 59                      pop     ecx
0047C746 3BD8                    cmp     ebx,eax
```

Unprocessed data | Strings | View 1 | View 2 | View 3 | View 4

```
00401070: [6A 00 6A 00 6A 00 6A 00 6A 00 E8 F1 A2 07 00 CC]
00406D7B: [EB 03 8D 49 00]
0040D08D: [32 C0 C2 04 00 CC CC CC CC CC CC CC CC CC CC CC]
00416940: [B0 01 C3 CC CC CC CC CC CC CC CC CC CC CC CC CC]
004190C5: [EB 09 8D A4 24 00 00 00 00 8B FF]
00426709: [32 C0 C2 04 00]
0042670E: [32 C0 C2 08 00]
0042A434: [32 C0 C2 10 00]
```

VMMap - Sysinternals: www.sysinternals.com

File Edit View Options Help

Process: chrome.exe
PID: 4636

Committed: 169,924 K

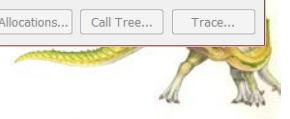
Private Bytes: 61,924 K

Working: 81,360 K

Type	Size	Committed	Private	Total WS	Privat...	Sharea...	Shared WS	Locked WS	Blocks	Largest
Total	222,356 K	169,924 K	61,924 K	81,360 K	47,952 K	33,408 K	33,400 K		562	
Image	88,320 K	86,292 K	2,896 K	19,852 K	824 K	19,028 K	19,020 K		316	37,628 K
Mapped File	3,804 K	3,804 K		396 K		396 K	396 K		7	2,876 K
Shareable	23,996 K	20,800 K		13,980 K		13,980 K	13,980 K		36	2,048 K
Heap	5,888 K	1,804 K	1,804 K	1,796 K	1,796 K				20	1,024 K
Managed Heap										
Stack	7,680 K	344 K	344 K	152 K	152 K				36	1,024 K
Private Data	92,060 K	56,272 K	56,272 K	44,576 K	44,572 K	4 K	4 K		147	16,384 K
Page Table	608 K	608 K	608 K	608 K	608 K					
Free	1,875,340 K									356,160 K

Address	Type	Size	Details	Protection	Committed	Private	Total WS	Privat
011B0000	Heap (Private Data)	256 K	Heap ID: 5 [LOW FRAGMENTATION]	Read/Write	256 K	256 K	252 K	
011F0000	Shareable	992 K		Read	992 K		272 K	
012F0000	Shareable	4 K		Read	4 K		4 K	
01300000	Heap (Private Data)	64 K	Heap ID: 7 [COMPATABILITY]	Read/Write	16 K	16 K	16 K	
01310000	Shareable	280 K		Read	280 K		188 K	
01360000	Thread Stack	256 K	64-bit thread stack	Read/Write/Guard	28 K	28 K	8 K	
013A0000	Private Data	4 K		Read/Write	4 K	4 K	4 K	
013B0000	Private Data	4 K		Read/Write	4 K	4 K	4 K	
013C0000	Private Data	4 K		Read/Write	4 K	4 K	4 K	
013D0000	Private Data	4 K		Read/Write	4 K	4 K	4 K	
013E0000	Image	1,232 K	C:\Users\Hank\AppData\Local\Google\Chrome\Application\chrome.exe	Execute/Read	1,232 K	68 K	404 K	
013E0000	Image	4 K	Header	Read	4 K		4 K	
013E1000	Image	612 K	.text	Execute/Read	612 K		296 K	
0147A000	Image	364 K	.rdata	Read	364 K	4 K	64 K	
014D5000	Image	12 K	.data	Read/Write	12 K	12 K	12 K	
014D8000	Image	4 K	.data	Copy on write	4 K	4 K		
014D9000	Image	16 K	.data	Read/Write	16 K	16 K	16 K	
014DD000	Image	4 K	.tls	Copy on write	4 K	4 K	4 K	
014DE000	Image	172 K	.rsrc	Read	172 K		8 K	
01509000	Image	44 K	.reloc	Read	44 K			
01520000	Private Data	4,096 K		Read/Write	40 K	40 K	40 K	
01920000	Heap (Private Data)	1,024 K	Heap ID: 6 [LOW FRAGMENTATION]	Read/Write	228 K	228 K	228 K	
01A20000	Private Data	4 K		Read/Write	4 K	4 K	4 K	
01A30000	Private Data	4 K		Read/Write	4 K	4 K	4 K	
01A40000	Private Data	4 K		Read/Write	4 K	4 K	4 K	
01A50000	Private Data	4 K		Read/Write	4 K	4 K	4 K	
01A60000	Thread Stack	256 K	64-bit thread stack	Read/Write/Guard	28 K	28 K	8 K	
01AA0000	Private Data	4 K		Read/Write	4 K	4 K	4 K	
01AB0000								

Timeline... Heap Allocations... Call Tree... Trace...

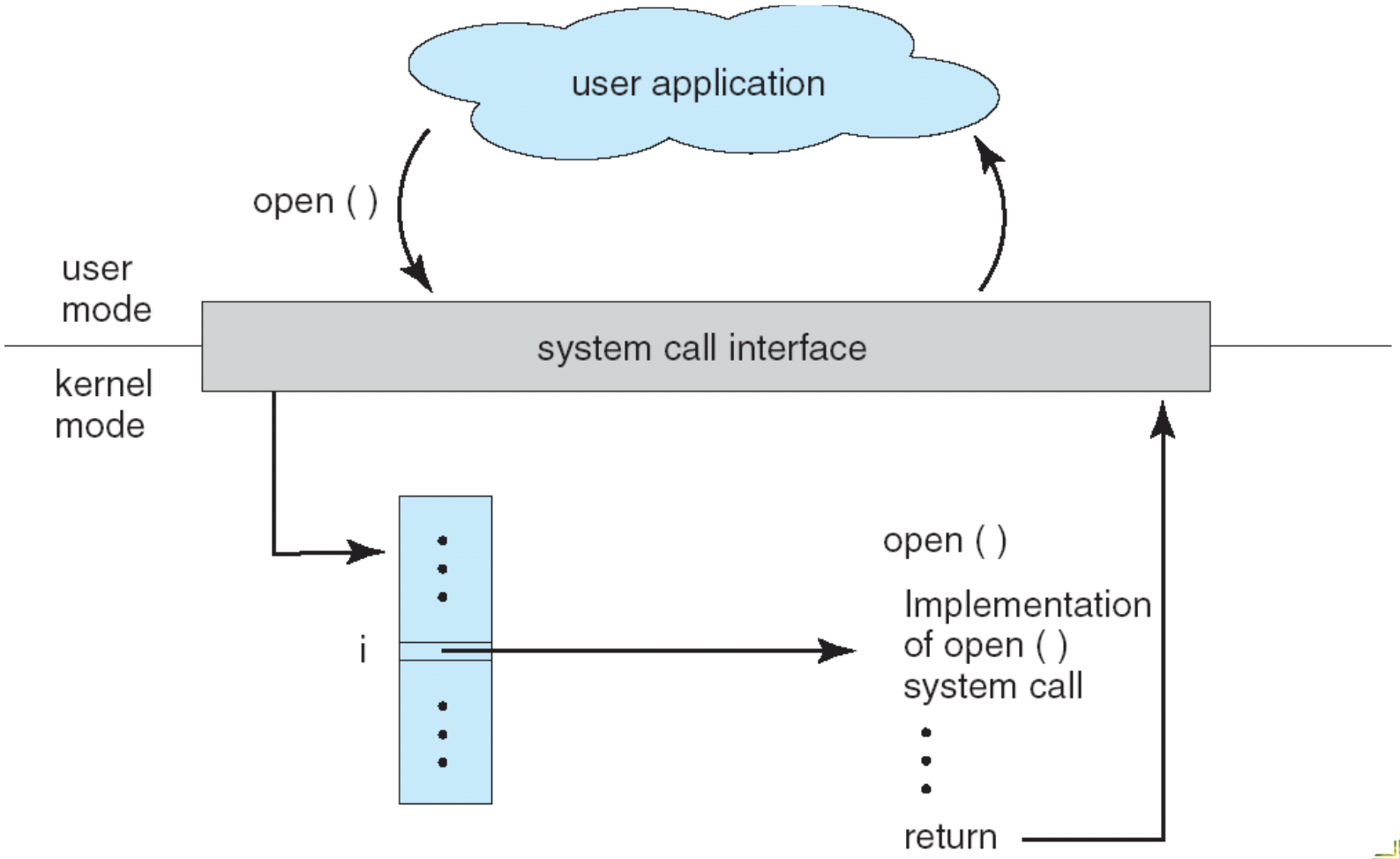


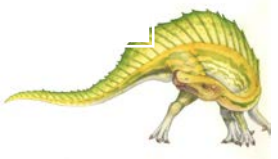
System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level Application Program Interface (API) rather than direct system call use
- Three most common APIs are Win32 API for Windows, POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)
- Why use APIs rather than system calls?

(Note that the system-call names used throughout this text are generic)

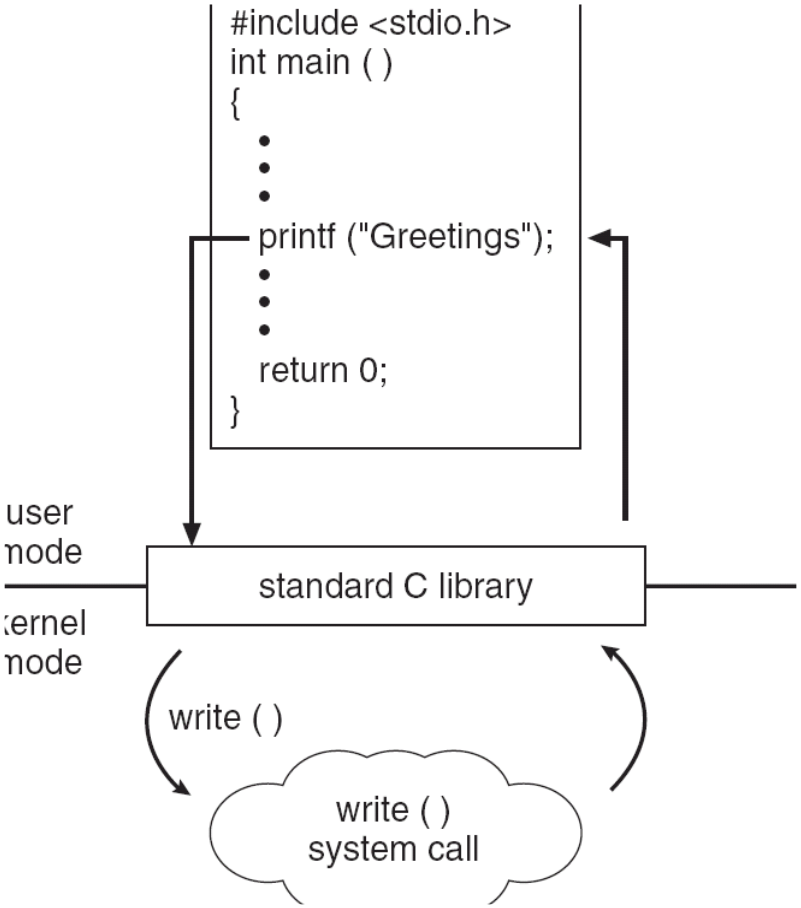
API – System Call – OS Relationship





Standard C Library Example

- C program invoking printf() library call, which calls write() system call



System Calls

- This is what happened in user mode when calling *fopen* on Win7 x86_64

test_open (Debugging) - Microsoft Visual Studio

File Edit View Project Build Debug Team Data Tools Test Analyze Window Help

Debug | x64 | SCANNER_READ_BUFFER_SIZI

Process: [8368] test_open.exe | Thread: [3028] Main Thread | Stack Frame: test_open.exe!wmain(int argc, wchar_t * *

Modules

Name	Path	Optimi...	User C...	Symbol Stat...	Symbol File	Or...	Version	Timestamp	Address	Process
msvcr100d.dll	C:\Windows\System32\...	N/A	N/A	Symbols loa...	C:\Symbols\msvcr1...	5	10.00.4...	2011/6/11 上...	000000000F7D0000-000000000F9A0000	[8368] test_open.exe: Native
kernel32.dll	C:\Windows\System32\k...	N/A	N/A	Symbols loa...	C:\Symbols\kernel3...	3	6.1.760...	2011/7/16 下...	0000000076D70000-0000000076E8F000	[8368] test_open.exe: Native
ntdll.dll	C:\Windows\System32\n...	N/A	N/A	Symbols loa...	C:\Symbols\ntdll.pd...	2	6.1.760...	2011/11/17 ...	00000000770F0000-0000000077299000	[8368] test_open.exe: Native
test_open.exe	D:\temp\test_open\x64\D...	N/A	N/A	Symbols loa...	D:\temp\test_open\...	1		2012/10/1 下...	000000013FD10000-000000013FD1E000	[8368] test_open.exe: Native
KernelBase.dll	C:\Windows\System32\K...	N/A	N/A	Symbols loa...	C:\Symbols\kernelb...	4	6.1.760...	2011/7/16 下...	000007FEFD870000-000007FEFD8DC000	[8368] test_open.exe: Native

Memory 1 | Modules

stdio.h | Disassembly | test_open.cpp

Address: wmain(int, wchar_t * *)

Viewing Options

```
    10:
    11:     fp = fopen("c:\\temp\\a.txt", "wt");
000000013FD11031 48 8D 15 58 57 00 00 lea         rdx,[__xi_z+130h (13FD16790h)]
000000013FD11038 48 8D 0D 59 57 00 00 lea         rcx,[__xi_z+138h (13FD16798h)]
000000013FD1103F FF 15 0B A5 00 00    call        qword ptr [__imp_fopen (13FD1B550h)]
000000013FD11045 48 89 44 24 20       mov         qword ptr [fp],rax
    12:
    13:     fclose(fp);
000000013FD1104A 48 8B 4C 24 20       mov         rcx,qword ptr [fp]
000000013FD1104F FF 15 03 A5 00 00    call        qword ptr [__imp_fclose (13FD1B558h)]
    14:
    15:     return 0;
000000013FD11055 33 C0                xor         eax,eax
```

Find Symbol Results

Call Stack

Name	Lan
ntdll.dll!ZwCreateFile()	
KernelBase.dll!CreateFileW() + 0x2b6 bytes	
kernel32.dll!CreateFileA() + 0xb6 bytes	
msvcr100d.dll!_sopen_helper() + 0x996 bytes	
msvcr100d.dll!_sopen_helper() + 0x257 bytes	
msvcr100d.dll!_sopen_s() + 0x42 bytes	
msvcr100d.dll!_openfile() + 0x956 bytes	
msvcr100d.dll!_fsopen() + 0x279 bytes	
msvcr100d.dll!fopen() + 0x23 bytes	
test_open.exe!wmain(int argc, wchar_t * * argv) Line 11 + 0x14 bytes	C++

Autos | Locals | Watch 1 | Find Symbol Results

Call Stack | Breakpoints | Command Window | Immediate Window | Output

Ready

System Calls

Call Stack

	Name
→	ntdll.dll!ZwCreateFile()
	KernelBase.dll!CreateFileW() + 0x2b6 bytes
	kernel32.dll!CreateFileA() + 0xb6 bytes
	msvcr100d.dll!_sopen_helper() + 0x996 bytes
	msvcr100d.dll!_sopen_helper() + 0x257 bytes
	msvcr100d.dll!_sopen_s() + 0x42 bytes
	msvcr100d.dll!_openfile() + 0x956 bytes
	msvcr100d.dll!_fsopen() + 0x279 bytes
	msvcr100d.dll!fopen() + 0x23 bytes
↓	test_open.exe!wmain(int argc, wchar_t * * argv) Line 11 + 0x14 bytes

Call Stack | Breakpoints | Command Window | Immediate

System Calls

stdio.h | Disassembly | test_open.cpp

Address: wmain(int, wchar_t * *)

Viewing Options

```
0000000077141850 4C 8B D1            mov         r10,rcx
0000000077141853 B8 51 00 00 00      mov         eax,51h
0000000077141858 0F 05               syscall
000000007714185A C3                  ret
000000007714185B 0F 1F 44 00 00      nop         dword ptr [rax+rax]
ZwCreateFile:
0000000077141860 4C 8B D1            mov         r10,rcx
0000000077141863 B8 52 00 00 00      mov         eax,52h
0000000077141868 0F 05               syscall
000000007714186A C3                  ret
000000007714186B 0F 1F 44 00 00      nop         dword ptr [rax+rax]
ZwQueryEvent:
0000000077141870 4C 8B D1            mov         r10,rcx
```

Call Stack

Name
ntdll.dll!ZwCreateFile()
KernelBase.dll!CreateFileW() + 0x2b6 bytes
kernel32.dll!CreateFileA() + 0xb6 bytes
msvcr100d.dll!_sopen_helper() + 0x996 bytes
msvcr100d.dll!_sopen_helper() + 0x257 bytes
msvcr100d.dll!_sopen_s() + 0x42 bytes
msvcr100d.dll!_openfile() + 0x956 bytes
msvcr100d.dll!_fsopen() + 0x279 bytes
msvcr100d.dll!fopen() + 0x23 bytes
test_open.exe!wmain(int argc, wchar_t * * argv) Line 11 + 0x14 bytes

Call Stack | Breakpoints | Command Window | Immediate Window | Output

Ready

Can’t step into this instruction in Visual Studio

System Calls

SYSCALL—Fast System Call

Opcode	Instruction	Op/ En	64-Bit Mode	Compat/ Leg Mode	Description
0F 05	SYSCALL	NP	Valid	Invalid	Fast call to privilege level 0 system procedures.

Instruction Operand Encoding

Op/En	Operand 1	Operand 2	Operand 3	Operand 4
NP	NA	NA	NA	NA

Description

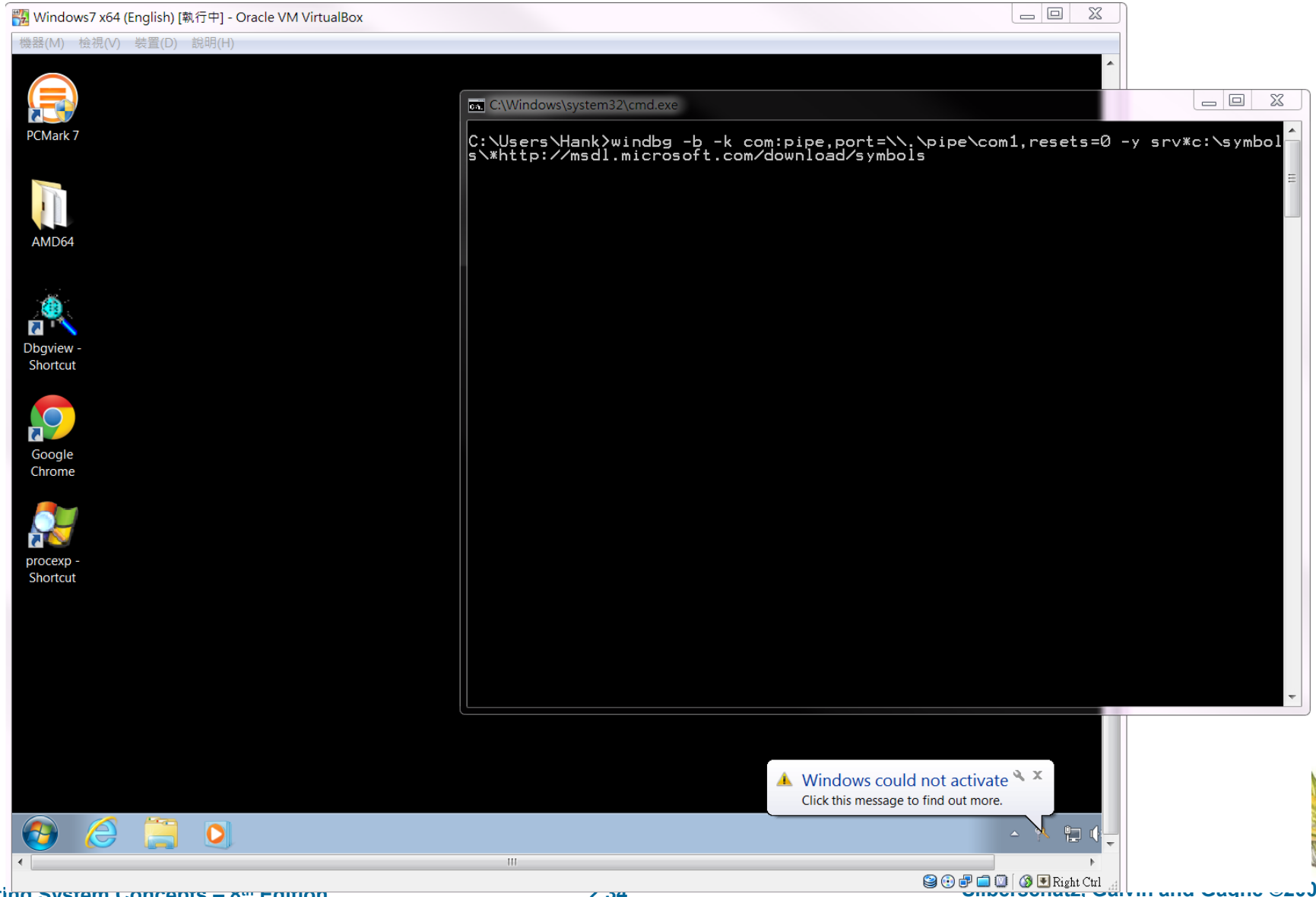
SYSCALL invokes an OS system-call handler at privilege level 0. It does so by loading RIP from the IA32_LSTAR MSR (after saving the address of the instruction following SYSCALL into RCX). (The WRMSR instruction ensures that the IA32_LSTAR MSR always contain a canonical address.)

SYSCALL also saves RFLAGS into R11 and then masks RFLAGS using the IA32_FMASK MSR (MSR address C0000084H); specifically, the processor clears in RFLAGS every bit corresponding to a bit that is set in the IA32_FMASK MSR.

SYSCALL loads the CS and SS selectors with values derived from bits 47:32 of the IA32_STAR MSR. However, the CS and SS descriptor caches are **not** loaded from the descriptors (in GDT or LDT) referenced by those selectors. Instead, the descriptor caches are loaded with fixed values. See the Operation section for details. It is the responsibility of OS software to ensure that the descriptors (in GDT or LDT) referenced by those selector values correspond to the fixed values loaded into the descriptor caches; the SYSCALL instruction does not ensure this correspondence.

The SYSCALL instruction does not save the stack pointer (RSP). If the OS system-call handler will change the stack pointer, it is the responsibility of software to save the previous value of the stack pointer. This might be done prior to executing SYSCALL, with software restoring the stack pointer with the instruction following SYSCALL (which will be executed after SYSRET). Alternatively, the OS system-call handler may save the stack pointer and restore it before executing SYSRET.

System Calls



System Calls

Kernel 'com:pipe,port=\\.\pipe\com1,resets=0' - WinDbg:6.12.0002.633 AMD64

File Edit View Debug Window Help

Command - Kernel 'com:pipe,port=\\.\pipe\com1,resets=0' - WinDbg:6.12.0002.633 AMD64

```
Microsoft (R) Windows Debugger Version 6.12.0002.633 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.

Opened \\.\pipe\com1
Waiting to reconnect...
Connected to Windows 7 7600 x64 target at (Mon Oct  1 21:28:47.199 2012 (UTC + 8:00)), ptr64 TRUE
Kernel Debugger connection established.  (Initial Breakpoint requested)
Symbol search path is: srv*c:\symbols\*http://msdl.microsoft.com/download/symbols;SRV*C:\Symbols*http://msdl.microsoft.com/download/symbols
Executable search path is:
Windows 7 Kernel Version 7600 MP (1 procs) Free x64
Product: WinNt, suite: TerminalServer SingleUserTS
Built by: 7600.16917.amd64fre.win7_gdr.111118-2330
Machine Name:
Kernel base = 0xfffff800`0281d000 PsLoadedModuleList = 0xfffff800`02a59e70
Debug session time: Mon Oct  1 21:27:15.990 2012 (UTC + 8:00)
System Uptime: 0 days 0:16:54.271
Break instruction exception - code 80000003 (first chance)
*******************************************************************************
*                                                                             *
*   You are seeing this message because you pressed either                    *
*       CTRL+C (if you run kd.exe) or,                                        *
*       CTRL+BREAK (if you run WinDBG),                                       *
*   on your debugger machine's keyboard.                                      *
*                                                                             *
*                   THIS IS NOT A BUG OR A SYSTEM CRASH                       *
*                                                                             *
* If you did not intend to break into the debugger, press the "g" key, then   *
* press the "Enter" key now.  This message might immediately reappear.  If it *
* does, press "g" and "Enter" again.                                          *
*                                                                             *
*******************************************************************************
nt!DbgBreakPointWithStatus:
fffff800`028855a0 cc              int     3

kd>
```

Ln 0, Col 0 | Sys 0:KdSrv:S | Proc 000:0 | Thrd 000:0 | ASM | OVR | CAPS | NUM

System Calls

Kernel 'com:pipe,port=\\.\pipe\com1,resets=0' - WinDbg:6.12.0002.633 AMD64

File Edit View Debug Window Help

Command - Kernel 'com:pipe,port=\\.\pipe\com1,resets=0' - WinDbg:6.12.0002.633 AMD64

```
kd> rdmsr c0000082
msr[c0000082] = fffff800`0288c500
kd> u fffff800`0288c500 L35
nt!KiSystemCall64:
fffff800`0288c500 0f01f8          swapgs
fffff800`0288c503 65488942510000000 mov   qword ptr gs:[10h],rsp
fffff800`0288c50c 65488b2425a8010000 mov   rsp,qword ptr gs:[1A8h]
fffff800`0288c515 6a2b            push    2Bh
fffff800`0288c517 65ff342510000000 push    qword ptr gs:[10h]
fffff800`0288c51f 4153            push    r11
fffff800`0288c521 6a33            push    33h
fffff800`0288c523 51              push    rcx
fffff800`0288c524 498bca          mov     rcx,r10
fffff800`0288c527 4883ec08        sub     rsp,8
fffff800`0288c52b 55              push    rbp
fffff800`0288c52c 4881ec58010000  sub     rsp,158h
fffff800`0288c533 488dac2480000000 lea     rbp,[rsp+80h]
fffff800`0288c53b 48899dc0000000  mov     qword ptr [rbp+0C0h],rbx
fffff800`0288c542 4889bdc8000000  mov     qword ptr [rbp+0C8h],rdi
fffff800`0288c549 4889b5d0000000  mov     qword ptr [rbp+0D0h],rsi
fffff800`0288c550 c645ab02        mov     byte ptr [rbp-55h],2
fffff800`0288c554 65488b1c2588010000 mov   rbx,qword ptr gs:[188h]
fffff800`0288c55d 0f0d8bd8010000  prefetchw [rbx+1D8h]
fffff800`0288c564 0fae5dac        stmxcsr dword ptr [rbp-54h]
fffff800`0288c568 650fae142580010000 ldmxcsr dword ptr gs:[180h]
fffff800`0288c571 807b0300        cmp     byte ptr [rbx+3],0
fffff800`0288c575 66c785800000000000 mov   word ptr [rbp+80h],0
fffff800`0288c57e 0f848c000000    je      nt!KiSystemCall64+0x110 (fffff800`0288c610)
fffff800`0288c584 488945b0        mov     qword ptr [rbp-50h],rax
fffff800`0288c588 48894db8        mov     qword ptr [rbp-48h],rcx
fffff800`0288c58c 488955c0        mov     qword ptr [rbp-40h],rdx
```

MSR[IA32_LSTAR]

System Calls

Kernel 'com:pipe,port=\\.\pipe\com1,resets=0' - WinDbg:6.12.0002.633 AMD64

File Edit View Debug Window Help

Command - Kernel 'com:pipe,port=\\.\pipe\com1,resets=0' - WinDbg:6.12.0002.633 AMD64

```
kd> u nt!NtCreateFile L30
nt!NtCreateFile:
fffff800`02b97e70 4c8bdc          mov     r11,rsp
fffff800`02b97e73 4881ec88000000  sub     rsp,88h
fffff800`02b97e7a 33c0            xor     eax,eax
fffff800`02b97e7c 498943f0        mov     qword ptr [r11-10h],rax
fffff800`02b97e80 c744247020000000 mov     dword ptr [rsp+70h],20h
fffff800`02b97e88 89442468        mov     dword ptr [rsp+68h],eax
fffff800`02b97e8c 498943d8        mov     qword ptr [r11-28h],rax
fffff800`02b97e90 89442458        mov     dword ptr [rsp+58h],eax
fffff800`02b97e94 8b8424e0000000  mov     eax,dword ptr [rsp+0E0h]
fffff800`02b97e9b 89442450        mov     dword ptr [rsp+50h],eax
fffff800`02b97e9f 488b8424d8000000 mov     rax,qword ptr [rsp+0D8h]
fffff800`02b97ea7 498943c0        mov     qword ptr [r11-40h],rax
fffff800`02b97eab 8b8424d0000000  mov     eax,dword ptr [rsp+0D0h]
fffff800`02b97eb2 89442440        mov     dword ptr [rsp+40h],eax
fffff800`02b97eb6 8b8424c8000000  mov     eax,dword ptr [rsp+0C8h]
fffff800`02b97ebd 89442438        mov     dword ptr [rsp+38h],eax
fffff800`02b97ec1 8b8424c0000000  mov     eax,dword ptr [rsp+0C0h]
fffff800`02b97ec8 89442430        mov     dword ptr [rsp+30h],eax
fffff800`02b97ecc 8b8424b8000000  mov     eax,dword ptr [rsp+0B8h]
fffff800`02b97ed3 89442428        mov     dword ptr [rsp+28h],eax
fffff800`02b97ed7 488b8424b0000000 mov     rax,qword ptr [rsp+0B0h]
fffff800`02b97edf 49894398        mov     qword ptr [r11-68h],rax
fffff800`02b97ee3 e8c85fffff      call    nt!IopCreateFile (fffff800`02b8deb0)
fffff800`02b97ee8 4881c488000000  add     rsp,88h
fffff800`02b97eef c3              ret
fffff800`02b97ef0 90              nop
fffff800`02b97ef1 90              nop
fffff800`02b97ef2 90              nop
fffff800`02b97ef3 90              nop
fffff800`02b97ef4 90              nop
```

System Calls

User Mode

```
test_open.exe!wmain(int argc, wchar_t * * argv)  Line 11 + 0x14
bytes
msvcr100d.dll!fopen()  + 0x23 bytes
msvcr100d.dll!_fsopen()  + 0x279 bytes
msvcr100d.dll!_openfile()  + 0x956 bytes
msvcr100d.dll!_sopen_s()  + 0x42 bytes
msvcr100d.dll!_sopen_helper()  + 0x257 bytes
msvcr100d.dll!_sopen_helper()  + 0x996 bytes
kernel32.dll!CreateFileA()  + 0xb6 bytes
KernelBase.dll!CreateFileW()  + 0x2b6 bytes
ntdll.dll!ZwCreateFile()  + 0xa bytes
syscall
```

Kernel Mode

```
nt!KiSystemCall64
…..
nt!NtCreateFile
```

System Calls (Linux x86_64)

b.c

```c
#include <stdio.h>

void main()
{

        FILE *fp;

        fp = fopen("/tmp/a.txt", "wt");

        getchar();

        fclose(fp);

}
```

```
[hank@Maestro t]$ gcc -g b.c
[hank@Maestro t]$ ls -al a.out
-rwxrwxr-x. 1 hank hank 9109 Sep 30 23:39 a.out
[hank@Maestro t]$
[hank@Maestro t]$
```

System Calls

```
File  Edit  View  Search  Terminal  Help
[hank@Maestro t]$ gdb ./a.out
GNU gdb (GDB) Fedora (7.4.50.20120120-50.fc17)
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/hank/t/a.out...done.
(gdb) list
1       #include <stdio.h>
2
3       void main()
4       {
5
6               FILE *fp;
7
8               fp = fopen("/tmp/a.txt", "wt");
9
10              getchar();
(gdb) break 8
Breakpoint 1 at 0x400584: file ./b.c, line 8.
(gdb) r
Starting program: /home/hank/t/a.out

Breakpoint 1, main () at ./b.c:8
8               fp = fopen("/tmp/a.txt", "wt");
(gdb)
```

```
Breakpoint 1, main () at ./b.c:8
8               fp = fopen("/tmp/a.txt", "wt");
(gdb) break __libc_open
Breakpoint 2 at 0x35a1817770: __libc_open. (2 locations)
(gdb) c
Continuing.

Breakpoint 2, open64 () at ../sysdeps/unix/syscall-template.S:82
82      T_PSEUDO (SYSCALL_SYMBOL, SYSCALL_NAME, SYSCALL_NARGS)
(gdb) where
#0  open64 () at ../sysdeps/unix/syscall-template.S:82
#1  0x00000035a1c77359 in _IO_file_open (is32not64=<optimized out>, read_write=4, prot=438, posix_mod
e=<optimized out>,
    filename=<optimized out>, fp=0x601010) at fileops.c:240
#2  _IO_new_file_fopen (fp=fp@entry=0x601010, filename=filename@entry=0x400663 "/tmp/a.txt", mode=<op
timized out>,
    mode@entry=0x400660 "wt", is32not64=is32not64@entry=1) at fileops.c:345
#3  0x00000035a1c6bb56 in __fopen_internal (filename=0x400663 "/tmp/a.txt", mode=0x400660 "wt", is32=
1) at ../libio/iofopen.c:93
#4  0x0000000000400593 in main () at ./b.c:8
(gdb)
```

```
[hank@Maestro t]$ ps -ef|grep a.out
hank     16984  1555  0 21:55 pts/0    00:00:00 gdb ./a.out
hank     16986 16984  0 21:55 pts/0    00:00:00 /home/hank/t/a.out
hank     17006  1715  0 22:00 pts/1    00:00:00 grep --color=auto a.out
[hank@Maestro t]$
[hank@Maestro t]$ cat /proc/16986/maps
00400000-00401000 r-xp 00000000 fd:02 1831565                            /home/hank/t/a.out
00600000-00601000 rw-p 00000000 fd:02 1831565                            /home/hank/t/a.out
00601000-00622000 rw-p 00000000 00:00 0                                  [heap]
35a1800000-35a1820000 r-xp 00000000 fd:01 175898                         /usr/lib64/ld-2.15.so
35a1a1f000-35a1a20000 r--p 0001f000 fd:01 175898                         /usr/lib64/ld-2.15.so
35a1a20000-35a1a21000 rw-p 00020000 fd:01 175898                         /usr/lib64/ld-2.15.so
35a1a21000-35a1a22000 rw-p 00000000 00:00 0
35a1c00000-35a1dac000 r-xp 00000000 fd:01 175914                         /usr/lib64/libc-2.15.so
35a1dac000-35a1fac000 ---p 001ac000 fd:01 175914                         /usr/lib64/libc-2.15.so
35a1fac000-35a1fb0000 r--p 001ac000 fd:01 175914                         /usr/lib64/libc-2.15.so
35a1fb0000-35a1fb2000 rw-p 001b0000 fd:01 175914                         /usr/lib64/libc-2.15.so
35a1fb2000-35a1fb7000 rw-p 00000000 00:00 0
7ffff7fe1000-7ffff7fe4000 rw-p 00000000 00:00 0
7ffff7ffd000-7ffff7ffe000 rw-p 00000000 00:00 0
7ffff7ffe000-7ffff7fff000 r-xp 00000000 00:00 0                          [vdso]
7ffffffde000-7ffffffff000 rw-p 00000000 00:00 0                          [stack]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
[hank@Maestro t]$
```

System Calls

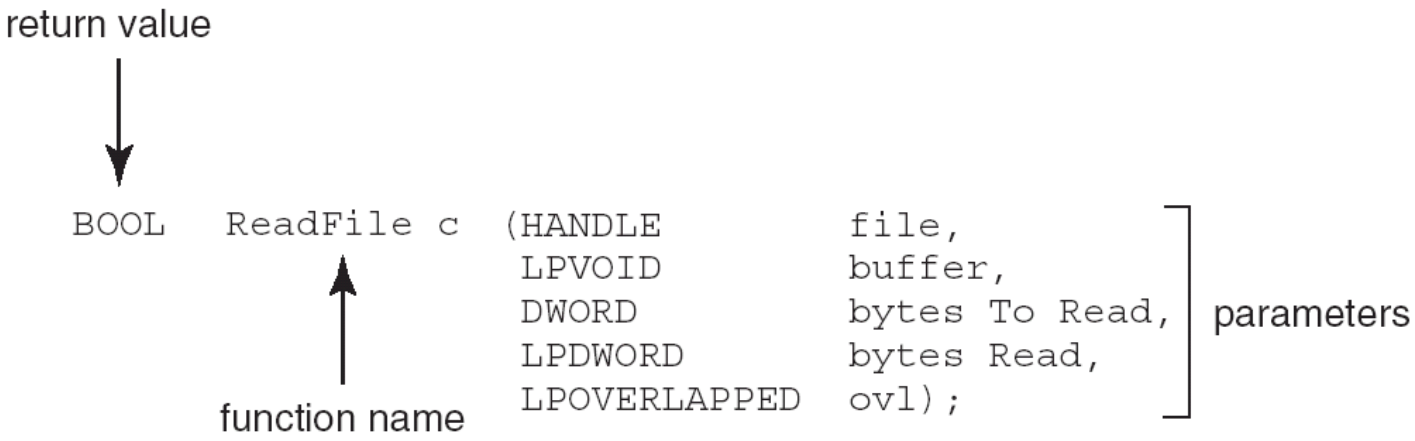
hank@Maestro:~/t

File Edit View Search Terminal Help

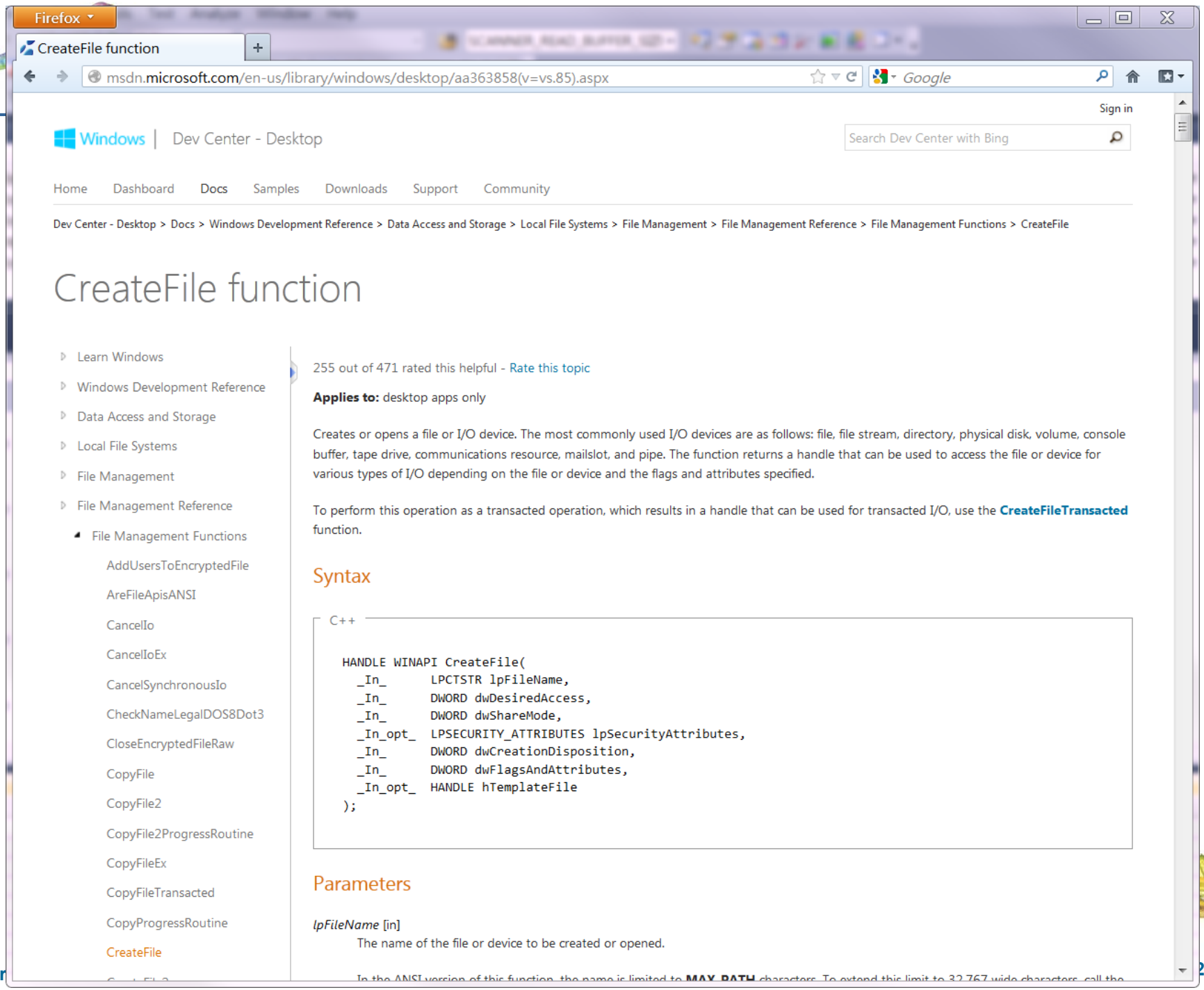
```
timized out>,
    mode@entry=0x400660 "wt", is32not64=is32not64@entry=1) at fileops.c:345
#3  0x00000035a1c6bb56 in __fopen_internal (filename=0x400663 "/tmp/a.txt", mode=0x400660 "wt", is32=
1) at ../libio/iofopen.c:93
#4  0x0000000000400593 in main () at ./b.c:8
(gdb) disassemble __libc_open
Dump of assembler code for function open64:
=> 0x00000035a1ce46e0 <+0>:     cmpl   $0x0,0x2d1acd(%rip)        # 0x35a1fb61b4 <__libc_multiple_thr
eads>
   0x00000035a1ce46e7 <+7>:     jne    0x35a1ce46f9 <open64+25>
   0x00000035a1ce46e9 <+0>:     mov    $0x2,%eax
   0x00000035a1ce46ee <+5>:     syscall
   0x00000035a1ce46f0 <+7>:     cmp    $0xfffffffffffff001,%rax
   0x00000035a1ce46f6 <+13>:    jae    0x35a1ce4729 <open64+73>
   0x00000035a1ce46f8 <+15>:    retq
   0x00000035a1ce46f9 <+25>:    sub    $0x8,%rsp
   0x00000035a1ce46fd <+29>:    callq  0x35a1cff4a0 <__libc_enable_asynccancel>
   0x00000035a1ce4702 <+34>:    mov    %rax,(%rsp)
   0x00000035a1ce4706 <+38>:    mov    $0x2,%eax
   0x00000035a1ce470b <+43>:    syscall
   0x00000035a1ce470d <+45>:    mov    (%rsp),%rdi
   0x00000035a1ce4711 <+49>:    mov    %rax,%rdx
   0x00000035a1ce4714 <+52>:    callq  0x35a1cff500 <__libc_disable_asynccancel>
   0x00000035a1ce4719 <+57>:    mov    %rdx,%rax
   0x00000035a1ce471c <+60>:    add    $0x8,%rsp
   0x00000035a1ce4720 <+64>:    cmp    $0xfffffffffffff001,%rax
   0x00000035a1ce4726 <+70>:    jae    0x35a1ce4729 <open64+73>
   0x00000035a1ce4728 <+72>:    retq
   0x00000035a1ce4729 <+73>:    mov    0x2cb700(%rip),%rcx        # 0x35a1fafe30
   0x00000035a1ce4730 <+80>:    xor    %edx,%edx
   0x00000035a1ce4732 <+82>:    sub    %rax,%rdx
   0x00000035a1ce4735 <+85>:    mov    %edx,%fs:(%rcx)
   0x00000035a1ce4738 <+88>:    or     $0xffffffffffffffff,%rax
   0x00000035a1ce473c <+92>:    jmp    0x35a1ce4728 <open64+72>
End of assembler dump.
(gdb)
```

Example of Standard API

- Consider the ReadFile() function in the
- Win32 API—a function for reading from a file



- A description of the parameters passed to ReadFile()
  - HANDLE file—the file to be read
  - LPVOID buffer—a buffer where the data will be read into and written from
  - DWORD bytesToRead—the number of bytes to be read into the buffer
  - LPDWORD bytesRead—the number of bytes read during the last read
  - LPOVERLAPPED ovl—indicates if overlapped I/O is being used



Firefox

CreateFile function

msdn.microsoft.com/en-us/library/windows/desktop/aa363858(v=vs.85).aspx

Google

Sign in

Windows | Dev Center - Desktop

Search Dev Center with Bing

Home Dashboard Docs Samples Downloads Support Community

Dev Center - Desktop > Docs > Windows Development Reference > Data Access and Storage > Local File Systems > File Management > File Management Reference > File Management Functions > CreateFile

CreateFile function

- Learn Windows
- Windows Development Reference
- Data Access and Storage
- Local File Systems
- File Management
- File Management Reference
  - File Management Functions
    - AddUsersToEncryptedFile
    - AreFileApisANSI
    - CancelIo
    - CancelIoEx
    - CancelSynchronousIo
    - CheckNameLegalDOS8Dot3
    - CloseEncryptedFileRaw
    - CopyFile
    - CopyFile2
    - CopyFile2ProgressRoutine
    - CopyFileEx
    - CopyFileTransacted
    - CopyProgressRoutine
    - CreateFile

255 out of 471 rated this helpful - Rate this topic

Applies to: desktop apps only

Creates or opens a file or I/O device. The most commonly used I/O devices are as follows: file, file stream, directory, physical disk, volume, console buffer, tape drive, communications resource, mailslot, and pipe. The function returns a handle that can be used to access the file or device for various types of I/O depending on the file or device and the flags and attributes specified.

To perform this operation as a transacted operation, which results in a handle that can be used for transacted I/O, use the **CreateFileTransacted** function.

Syntax

C++

```
HANDLE WINAPI CreateFile(
  _In_      LPCTSTR lpFileName,
  _In_      DWORD dwDesiredAccess,
  _In_      DWORD dwShareMode,
  _In_opt_  LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  _In_      DWORD dwCreationDisposition,
  _In_      DWORD dwFlagsAndAttributes,
  _In_opt_  HANDLE hTemplateFile
);
```

Parameters

lpFileName [in]

The name of the file or device to be created or opened.

In the ANSI version of this function, the name is limited to **MAX_PATH** characters. To extend this limit to 32,767 wide characters, call the

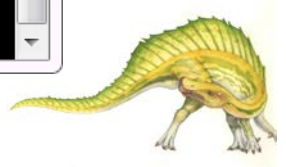
System Call Implementation

- Typically, a number associated with each system call
  - System-call interface maintains a table indexed according to these numbers
- The system call interface invokes intended system call in OS kernel and returns status of the system call and any return values
- The caller need know nothing about how the system call is implemented
  - Just needs to obey API and understand what OS will do as a result call
  - Most details of OS interface hidden from programmer by API
    - Managed by run-time support library (set of functions built into libraries included with compiler)

(/usr/include/asm/unistd_64.h)

linux1:/usr/include/asm

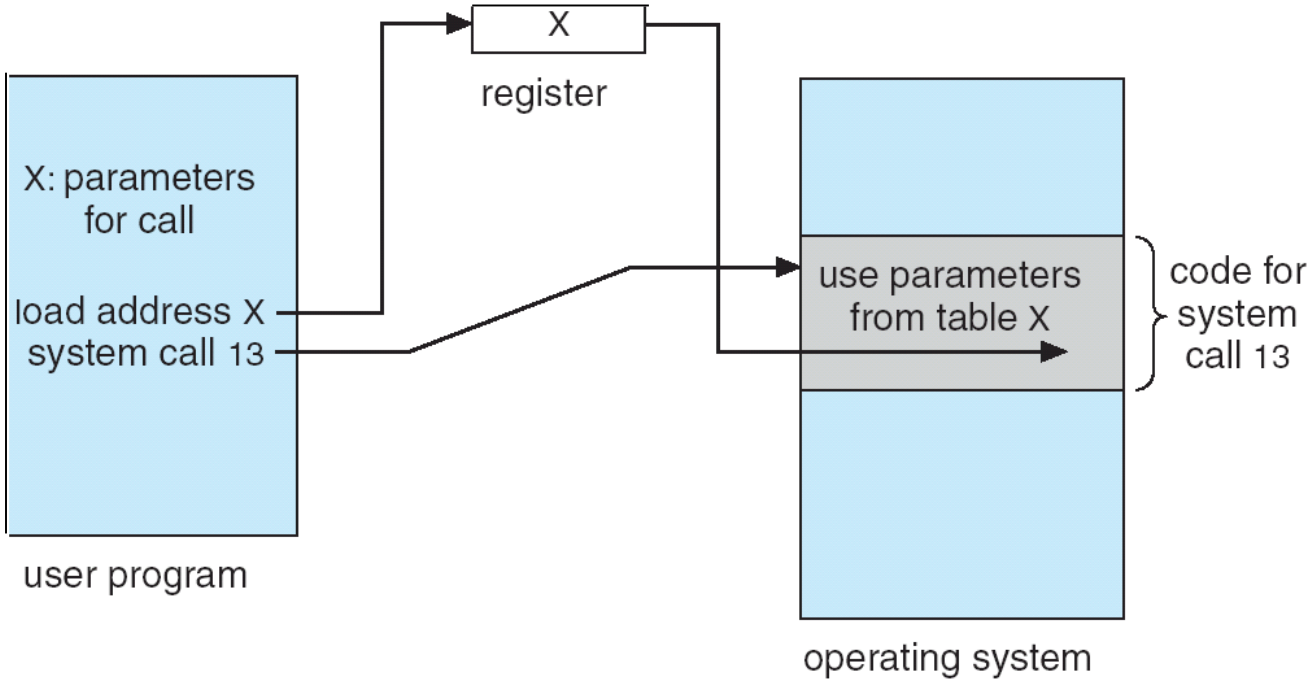
```
#ifndef _ASM_X86_UNISTD_64_H
#define _ASM_X86_UNISTD_64_H

#ifndef __SYSCALL
#define __SYSCALL(a, b)
#endif

/*
 * This file contains the system call numbers.
 *
 * Note: holes are not allowed.
 */

/* at least 8 syscall per cacheline */
#define __NR_read                               0
__SYSCALL(__NR_read, sys_read)
#define __NR_write                              1
__SYSCALL(__NR_write, sys_write)
#define __NR_open                               2
__SYSCALL(__NR_open, sys_open)
#define __NR_close                              3
__SYSCALL(__NR_close, sys_close)
#define __NR_stat                               4
--More--(2%)
```



System Call Parameter Passing

- Often, more information is required than simply identity of desired system call
  - Exact type and amount of information vary according to OS and call
- Three general methods used to pass parameters to the OS
  - Simplest: pass the parameters in *registers*
    - In some cases, may be more parameters than registers
  - Parameters stored in a *block,* or table, in memory, and address of block passed as a parameter in a register
    - This approach taken by Linux and Solaris
  - Parameters placed, or *pushed,* onto the *stack* by the program and *popped* off the stack by the operating system
  - Block and stack methods do not limit the number or length of parameters being passed

Parameter Passing via Table



Parameter Passing

http://www.x86-64.org/documentation/abi-0.99.pdf

A.2.1 Calling Conventions

The Linux AMD64 kernel uses internally the same calling conventions as user-level applications (see section 3.2.3 for details). User-level applications that like to call system calls should use the functions from the C library. The interface between the C library and the Linux kernel is the same as for the user-level applications with the following differences:

1. User-level applications use as integer registers for passing the sequence `%rdi`, `%rsi`, `%rdx`, `%rcx`, `%r8` and `%r9`. The kernel interface uses `%rdi`, `%rsi`, `%rdx`, `%r10`, `%r8` and `%r9`.

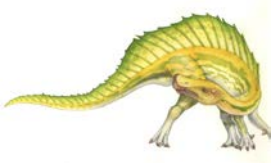
2. A system-call is done via the `syscall` instruction. The kernel destroys registers `%rcx` and `%r11`.

3. The number of the syscall has to be passed in register `%rax`.

4. System-calls are limited to six arguments, no argument is passed directly on the stack.

5. Returning from the `syscall`, register `%rax` contains the result of the system-call. A value in the range between -4095 and -1 indicates an error, it is `-errno`.

6. Only values of class INTEGER or class MEMORY are passed to the kernel.



lxr.linux.no/linux+v3.6/fs/open.c

```
940         return do_file_open_root(dentry, mnt, filename, &op, lookup);
941 }
942 EXPORT_SYMBOL(file_open_root);
943
944 long do_sys_open(int dfd, const char __user *filename, int flags, umode_t mode)
945 {
946         struct open_flags op;
947         int lookup = build_open_flags(flags, mode, &op);
948         char *tmp = getname(filename);
949         int fd = PTR_ERR(tmp);
950
951         if (!IS_ERR(tmp)) {
952                 fd = get_unused_fd_flags(flags);
953                 if (fd >= 0) {
954                         struct file *f = do_filp_open(dfd, tmp, &op, lookup);
955                         if (IS_ERR(f)) {
956                                 put_unused_fd(fd);
957                                 fd = PTR_ERR(f);
958                         } else {
959                                 fsnotify_open(f);
960                                 fd_install(fd, f);
961                         }
962                 }
963                 putname(tmp);
964         }
965         return fd;
966 }
967
968 SYSCALL_DEFINE3(open, const char __user *, filename, int, flags, umode_t, mode)
969 {
970         long ret;
971
972         if (force_o_largefile())
973                 flags |= O_LARGEFILE;
974
975         ret = do_sys_open(AT_FDCWD, filename, flags, mode);
976         /* avoid REGPARM breakage on x86: */
977         asmlinkage_protect(3, ret, filename, flags, mode);
978         return ret;
979 }
980
```

Types of System Calls

- Process control
- File management
- Device management
- Information maintenance
- Communications
- Protection

Examples of Windows and Unix System Calls

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

Operating System Design and Implementation

- Design and Implementation of OS not “solvable”, but some approaches have proven successful
- Internal structure of different Operating Systems can vary widely
- Start by defining goals and specifications
- Affected by choice of hardware, type of system
- *User* goals and *System* goals
  - User goals – operating system should be convenient to use, easy to learn, reliable, safe, and fast
  - System goals – operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient

Operating System Design and Implementation (Cont)

- Important principle to separate

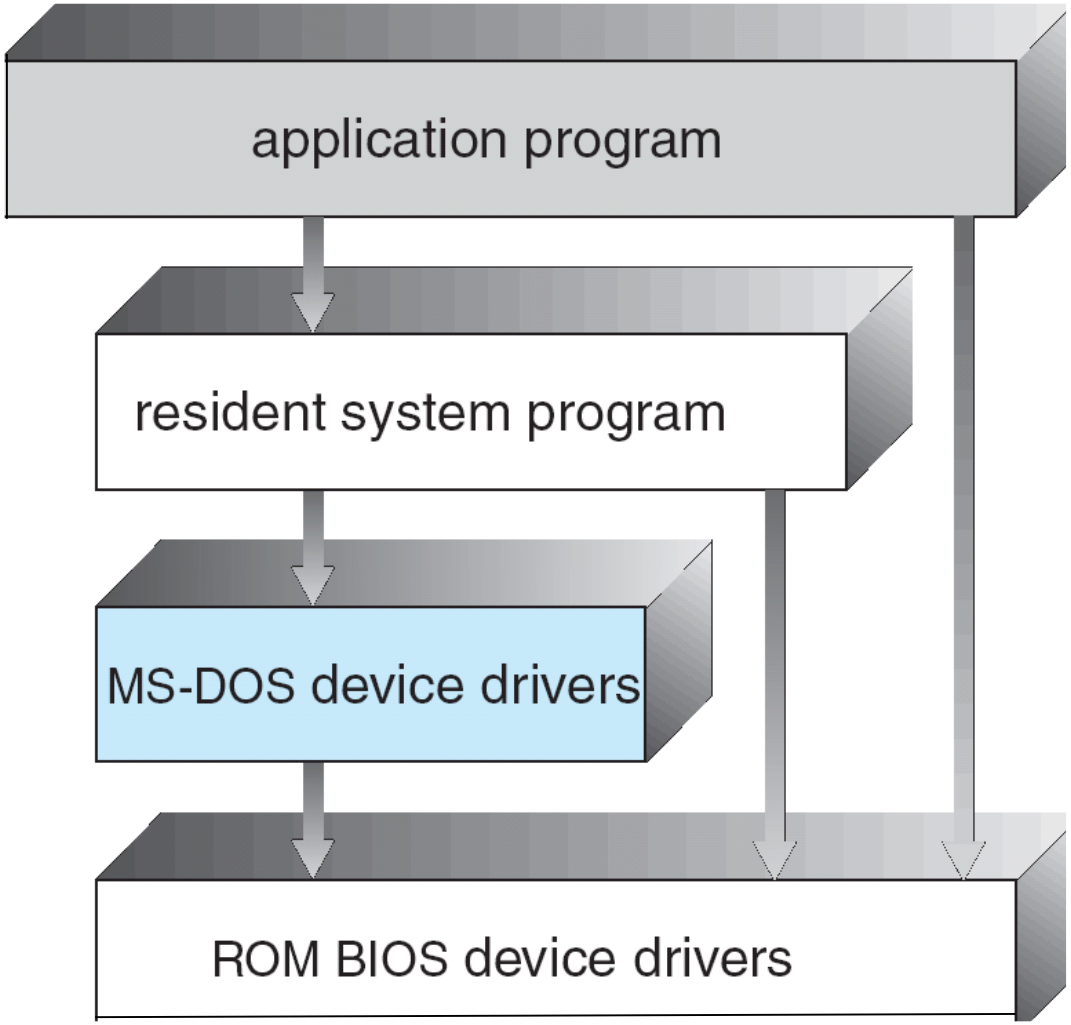
  Policy: What will be done?
  Mechanism: How to do it?

- Mechanisms determine how to do something, policies decide what will be done
  - The separation of policy from mechanism is a very important principle, it allows maximum flexibility if policy decisions are to be changed later

Simple Structure

- MS-DOS – written to provide the most functionality in the least space
  - Not divided into modules
  - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated

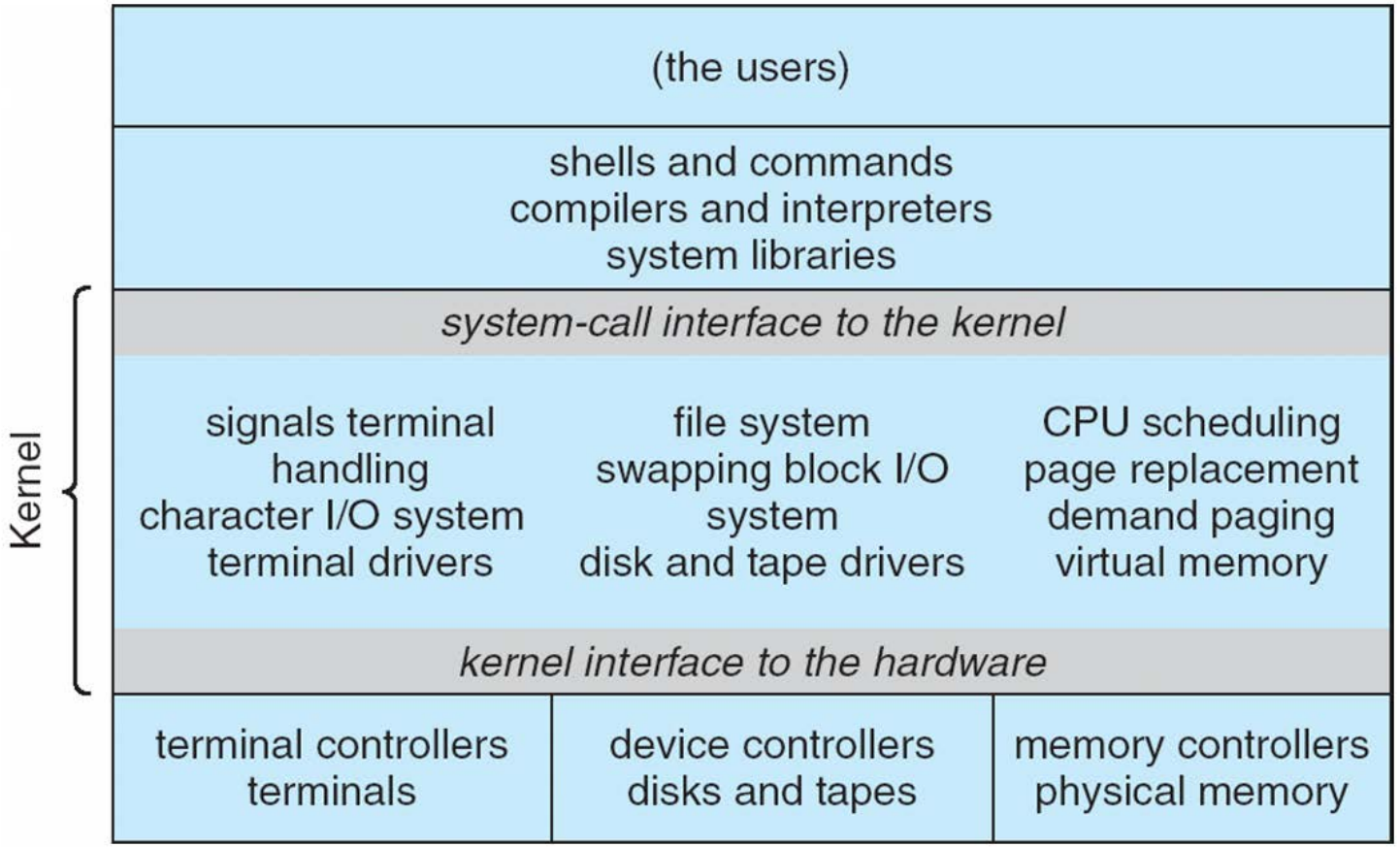
MS-DOS Layer Structure



Layered Approach

- The operating system is divided into a number of layers (levels), **each built on top of lower layers**. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers

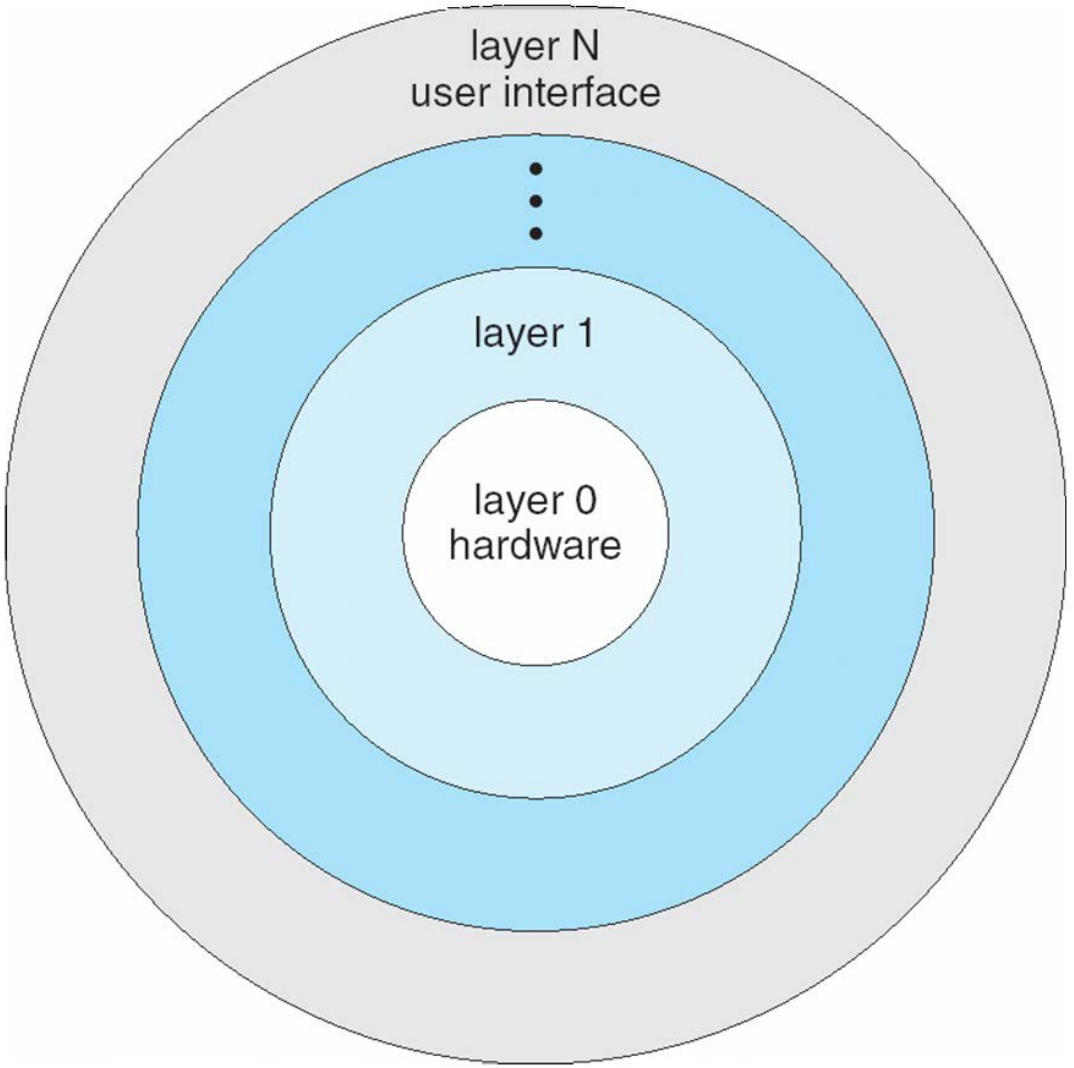
Traditional UNIX System Structure



UNIX

- UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring. The UNIX OS consists of two separable parts
  - Systems programs
  - The kernel
    - Consists of everything below the system-call interface and above the physical hardware
    - Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level

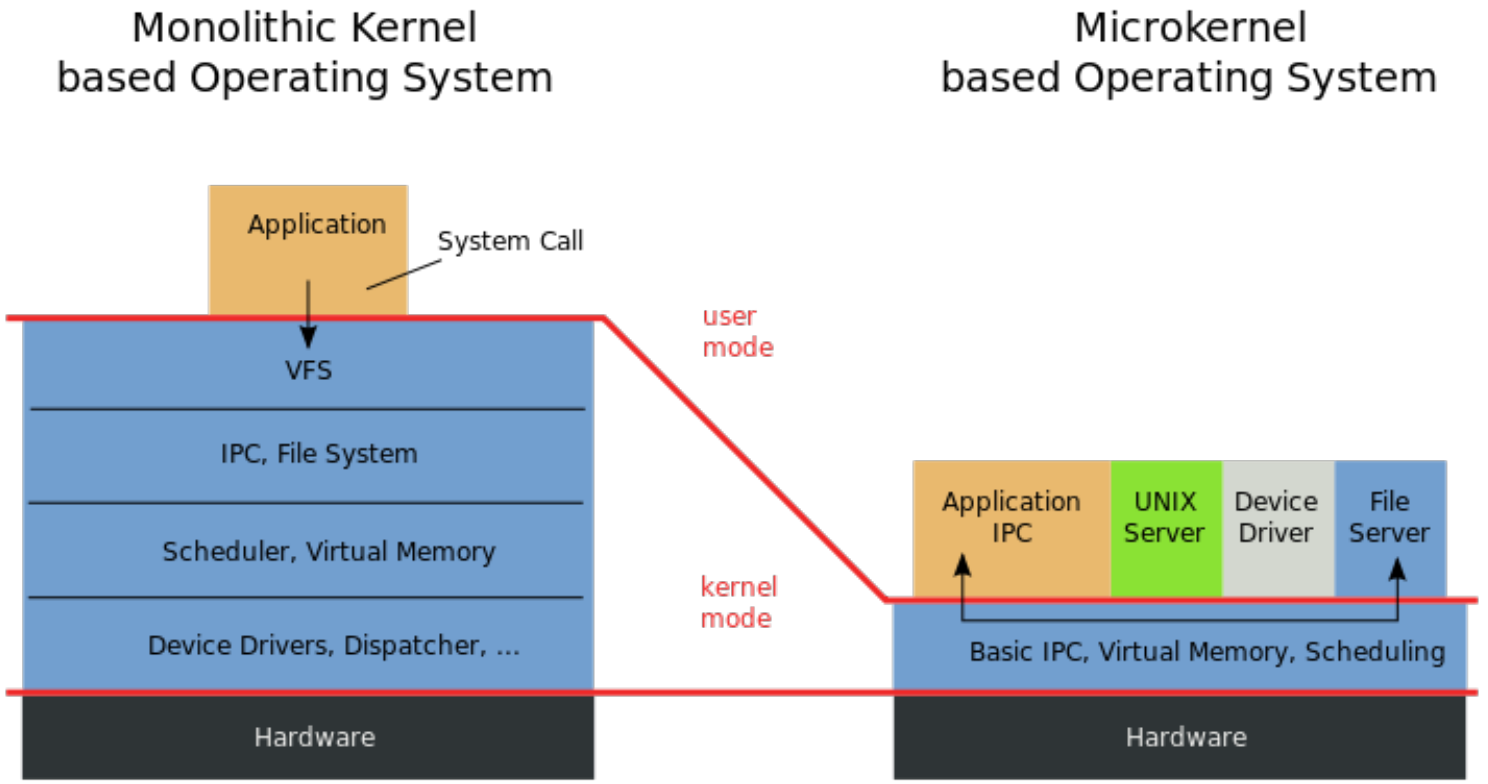
Layered Operating System



Microkernel System Structure

- Moves as much from the kernel into "*user*" space
- Communication takes place between user modules using message passing
- Benefits:
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure
- Detriments:
  - Performance overhead of user space to kernel space communication

Microkernel System Structure



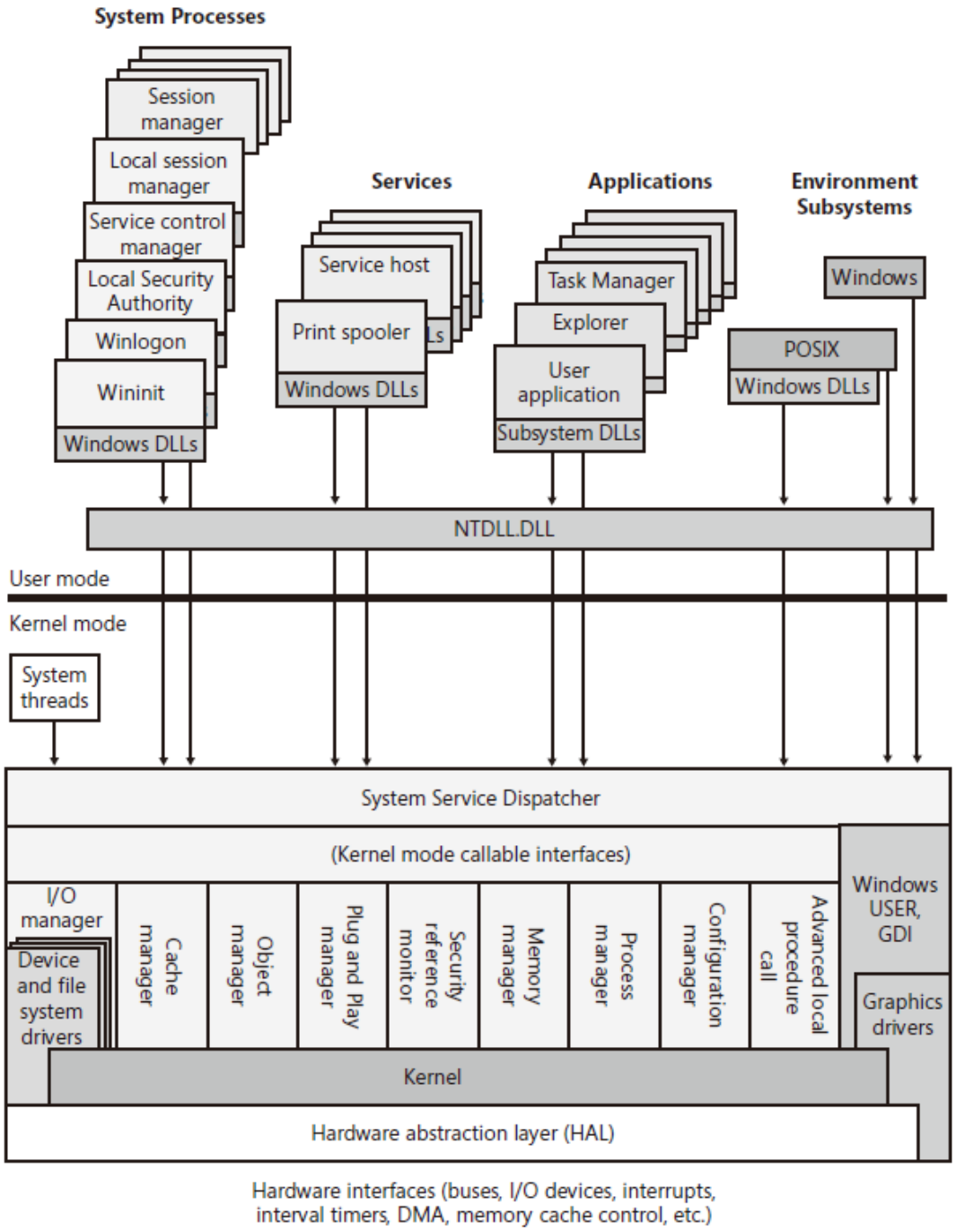
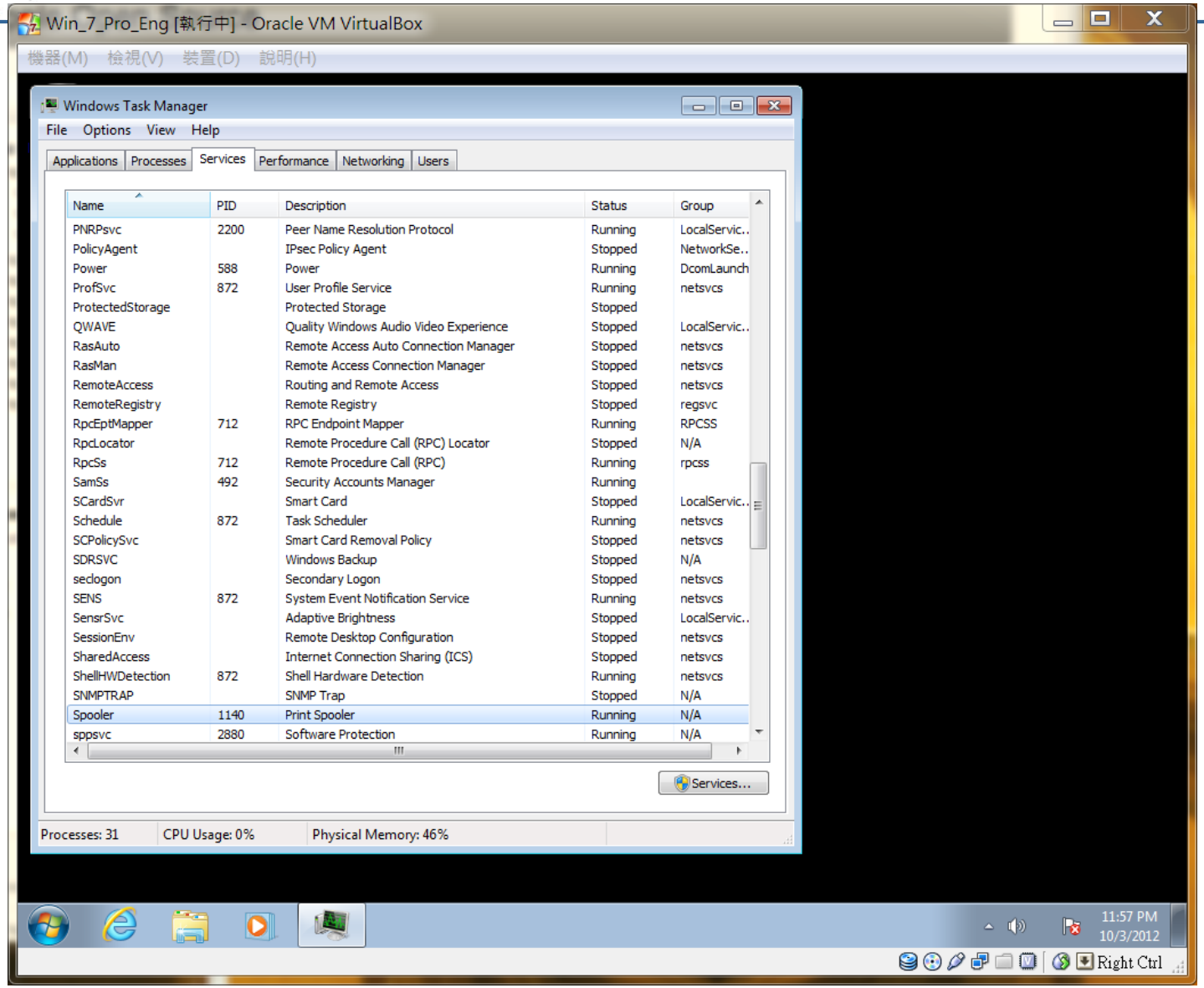
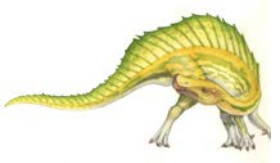


FIGURE 2-3 Windows architecture

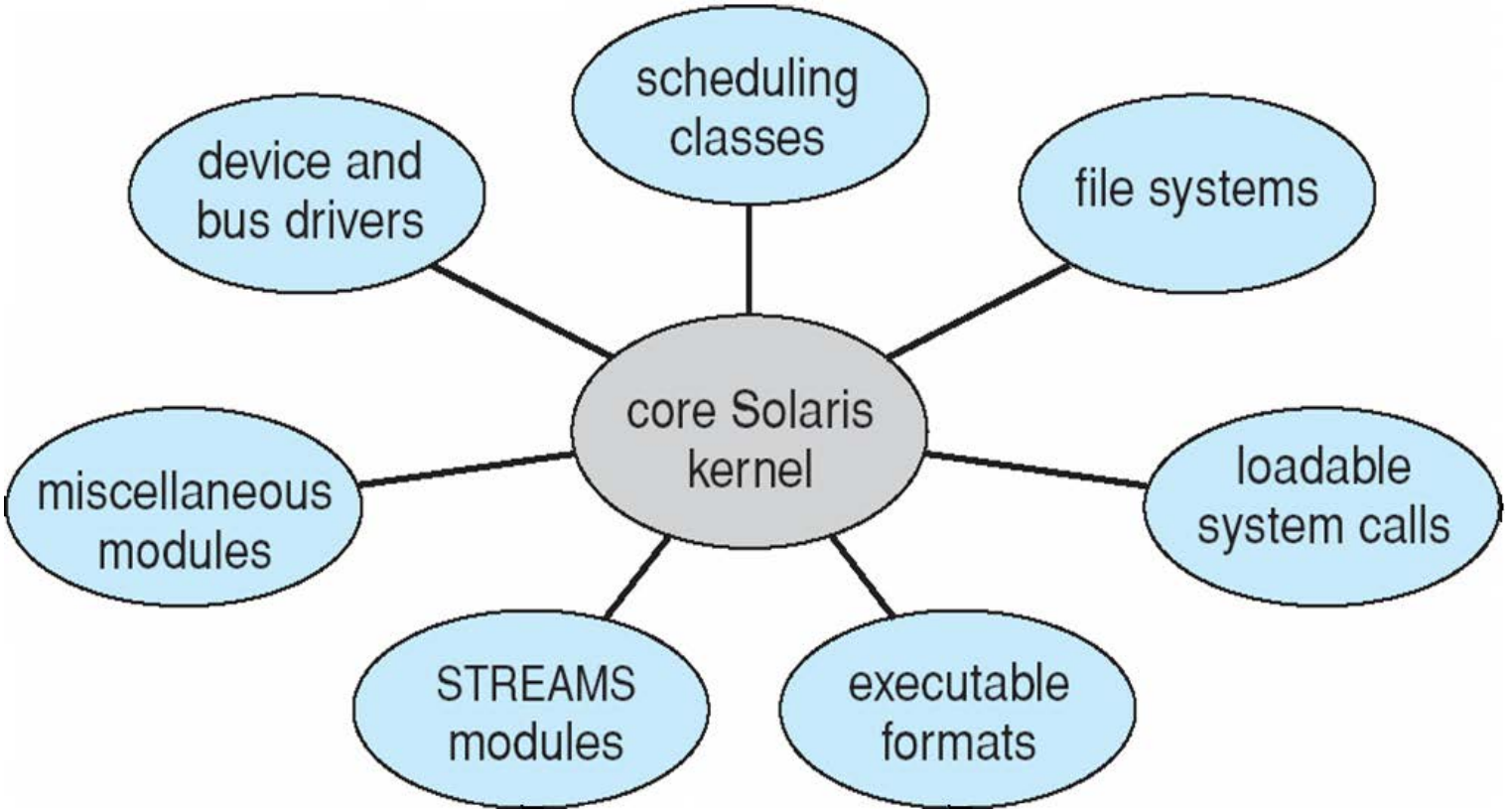

Win_7_Pro_Eng [執行中] - Oracle VM VirtualBox
機器(M) 檢視(V) 裝置(D) 說明(H)
Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users
Name PID Description Status Group
PNRPsvc 2200 Peer Name Resolution Protocol Running LocalServic..
PolicyAgent IPsec Policy Agent Stopped NetworkSe..
Power 588 Power Running DcomLaunch
ProfSvc 872 User Profile Service Running netsvcs
ProtectedStorage Protected Storage Stopped
QWAVE Quality Windows Audio Video Experience Stopped LocalServic..
RasAuto Remote Access Auto Connection Manager Stopped netsvcs
RasMan Remote Access Connection Manager Stopped netsvcs
RemoteAccess Routing and Remote Access Stopped netsvcs
RemoteRegistry Remote Registry Stopped regsvc
RpcEptMapper 712 RPC Endpoint Mapper Running RPCSS
RpcLocator Remote Procedure Call (RPC) Locator Stopped N/A
RpcSs 712 Remote Procedure Call (RPC) Running rpcss
SamSs 492 Security Accounts Manager Running
SCardSvr Smart Card Stopped LocalServic..
Schedule 872 Task Scheduler Running netsvcs
SCPolicySvc Smart Card Removal Policy Stopped netsvcs
SDRSVC Windows Backup Stopped N/A
seclogon Secondary Logon Stopped netsvcs
SENS 872 System Event Notification Service Running netsvcs
SensrSvc Adaptive Brightness Stopped LocalServic..
SessionEnv Remote Desktop Configuration Stopped netsvcs
SharedAccess Internet Connection Sharing (ICS) Stopped netsvcs
ShellHWDetection 872 Shell Hardware Detection Running netsvcs
SNMPTRAP SNMP Trap Stopped N/A
Spooler 1140 Print Spooler Running N/A
sppsvc 2880 Software Protection Running N/A
Services...
Processes: 31 CPU Usage: 0% Physical Memory: 46%
11:57 PM
10/3/2012
Right Ctrl

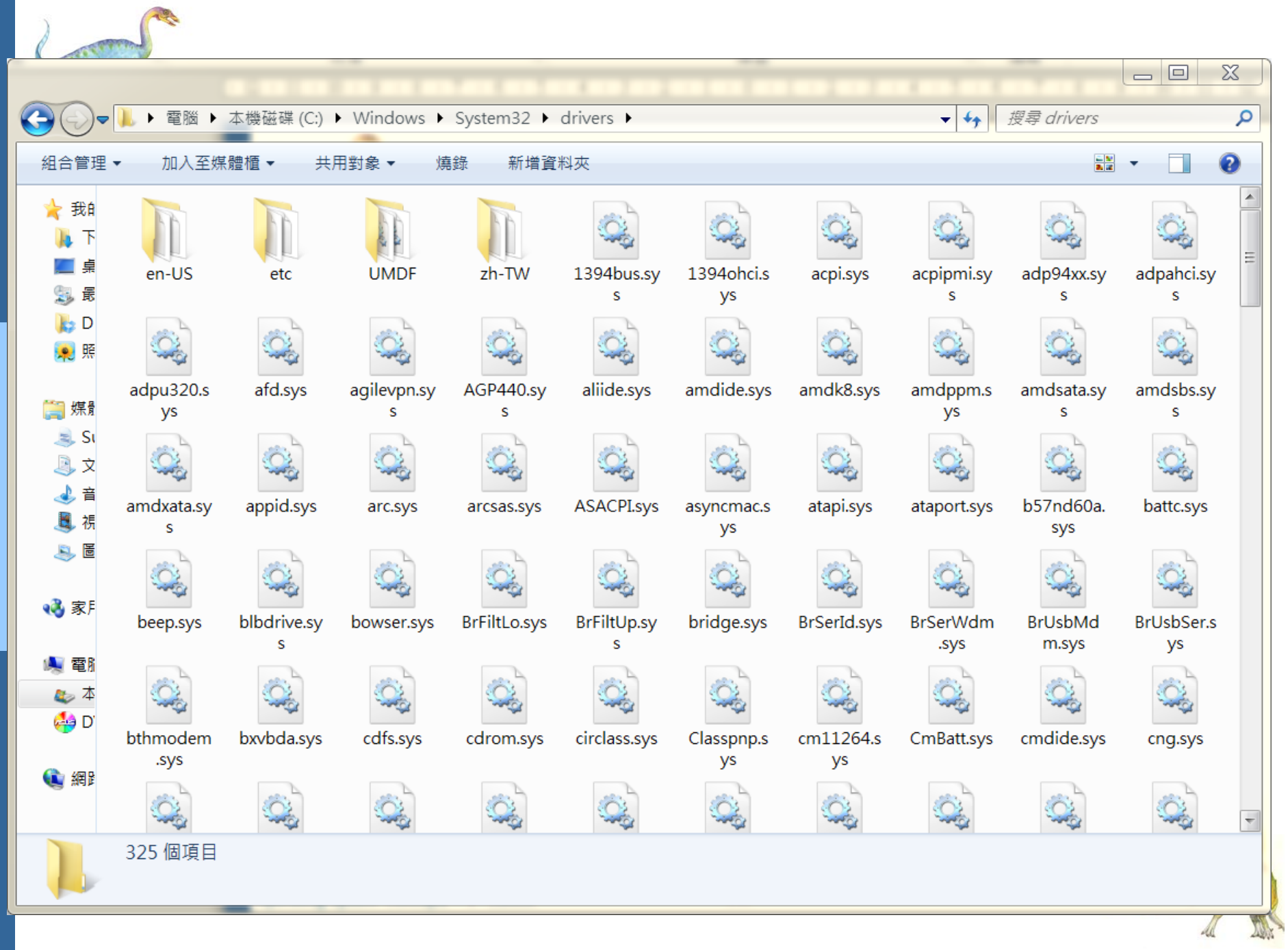


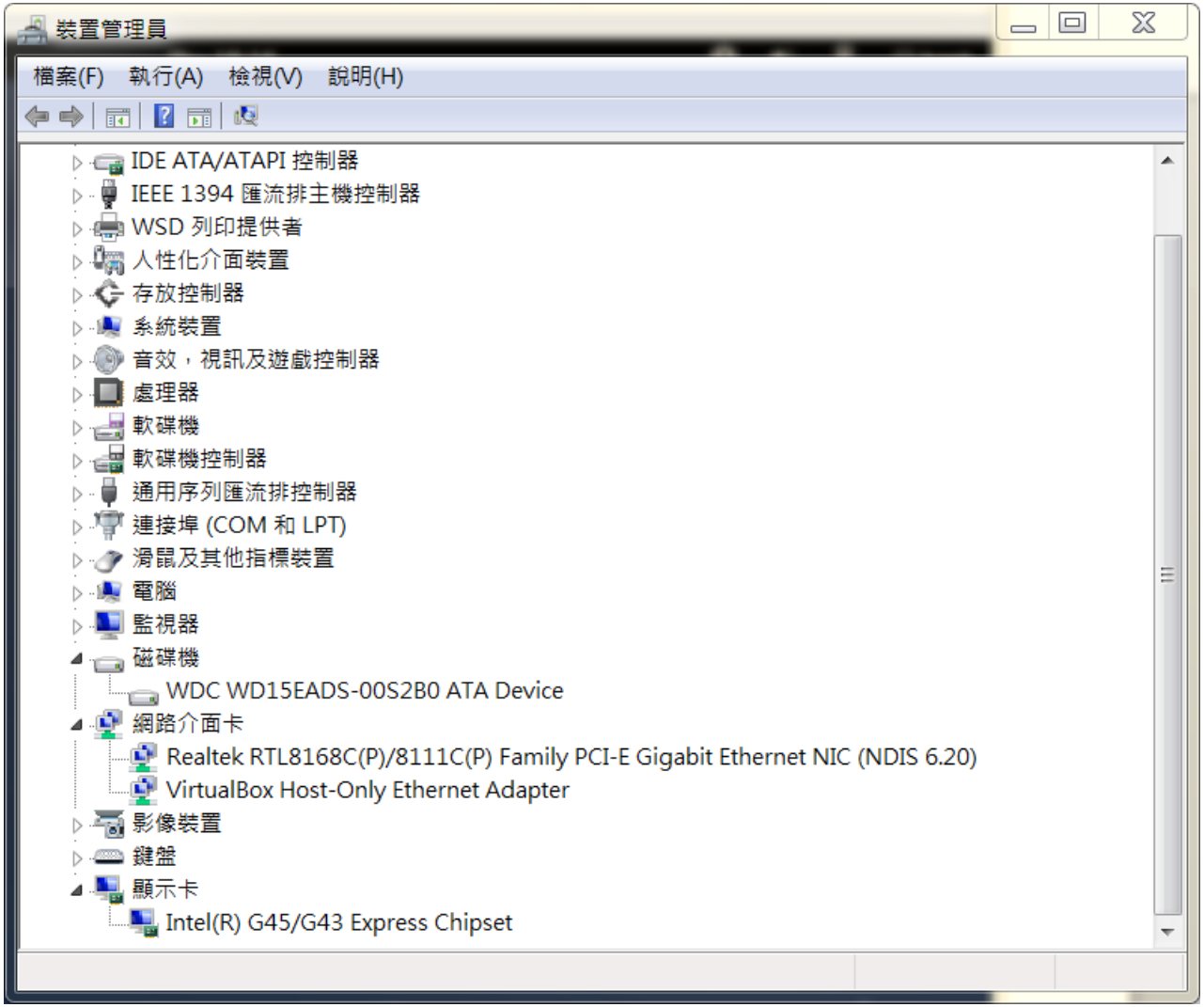
Kernel Modules

- Most modern operating systems implement kernel modules
  - Uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel
- Overall, similar to layers but with more flexible

Solaris Kernel Modular Approach




電腦 ▸ 本機磁碟 (C:) ▸ Windows ▸ System32 ▸ drivers ▸
搜尋 drivers
組合管理
加入至媒體櫃
共用對象
燒錄
新增資料夾
en-US
etc
UMDF
zh-TW
1394bus.sys
1394ohci.sys
acpi.sys
acpipmi.sys
adp94xx.sys
adpahci.sys
adpu320.sys
afd.sys
agilevpn.sys
AGP440.sys
aliide.sys
amdide.sys
amdk8.sys
amdppm.sys
amdsata.sys
amdsbs.sys
amdxata.sys
appid.sys
arc.sys
arcsas.sys
ASACPI.sys
asyncmac.sys
atapi.sys
ataport.sys
b57nd60a.sys
battc.sys
beep.sys
blbdrive.sys
bowser.sys
BrFiltLo.sys
BrFiltUp.sys
bridge.sys
BrSerId.sys
BrSerWdm.sys
BrUsbMdm.sys
BrUsbSer.sys
bthmodem.sys
bxvbda.sys
cdfs.sys
cdrom.sys
circlass.sys
Classpnp.sys
cm11264.sys
CmBatt.sys
cmdide.sys
cng.sys
325 個項目


裝置管理員
檔案(F) 執行(A) 檢視(V) 說明(H)
IDE ATA/ATAPI 控制器
IEEE 1394 匯流排主機控制器
WSD 列印提供者
人性化介面裝置
存放控制器
系統裝置
音效，視訊及遊戲控制器
處理器
軟碟機
軟碟機控制器
通用序列匯流排控制器
連接埠 (COM 和 LPT)
滑鼠及其他指標裝置
電腦
監視器
磁碟機
WDC WD15EADS-00S2B0 ATA Device
網路介面卡
Realtek RTL8168C(P)/8111C(P) Family PCI-E Gigabit Ethernet NIC (NDIS 6.20)
VirtualBox Host-Only Ethernet Adapter
影像裝置
鍵盤
顯示卡
Intel(R) G45/G43 Express Chipset

Kernel Modules

Question:

Are kernel modules isolated from each other?

(just like user processes are isolated from each other)

Kernel Modules

Question:

Is the kernel isolated from the kernel modules?

(just like user processes are isolated from kernel)

Kernel Modules

Question:

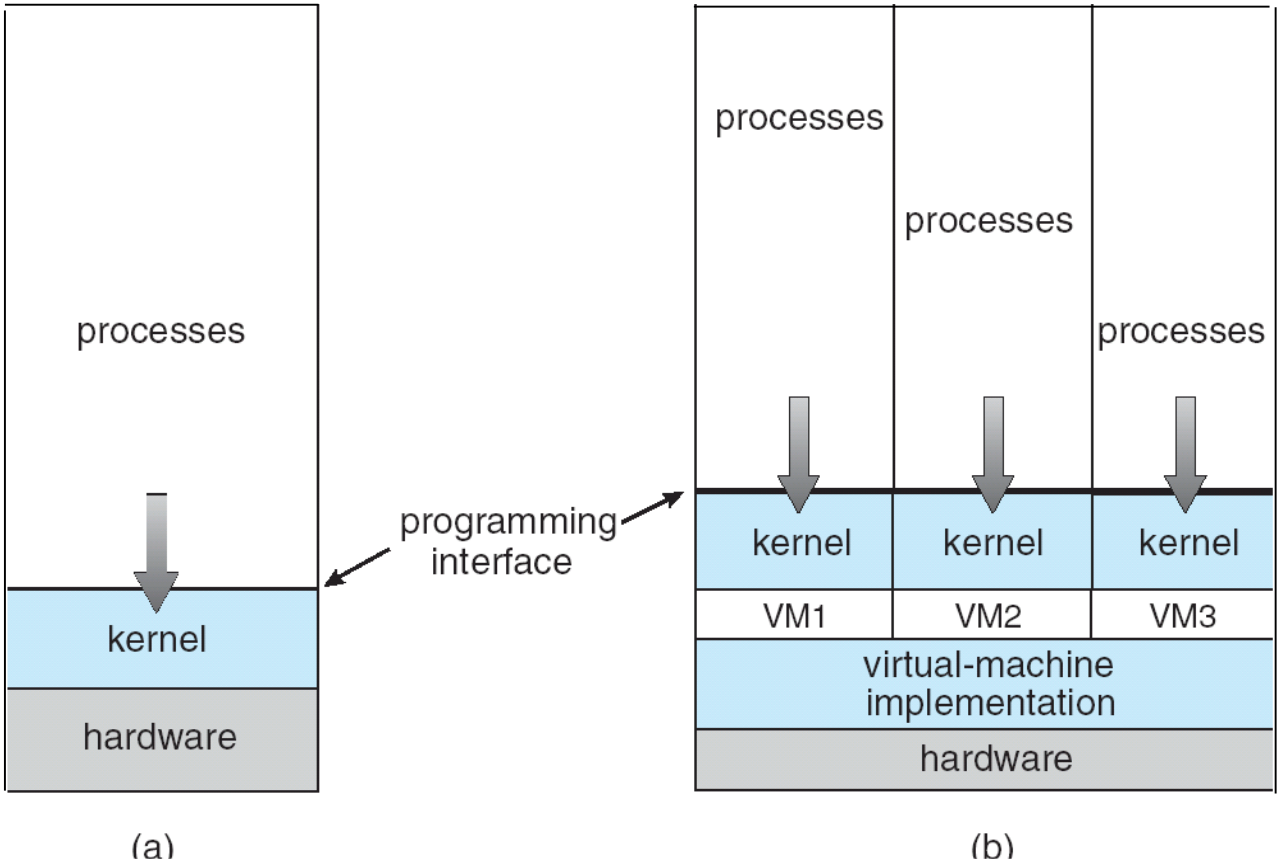
Virtual Machines

- A virtual machine takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware
- A virtual machine provides an interface *identical* to the underlying bare hardware
- The operating system host creates the illusion that a process has its own processor and (virtual memory)
- Each guest provided with a (virtual) copy of underlying computer

Virtual Machines History and Benefits

- First appeared commercially in IBM mainframes in 1972
- Fundamentally, multiple execution environments (different operating systems) can share the same hardware
- Protect from each other
- Some sharing of file can be permitted, controlled
- Commutate with each other, other physical systems via networking
- Useful for development, testing
- Consolidation of many low-resource use systems onto fewer busier systems
- “Open Virtual Machine Format”, standard format of virtual machines, allows a VM to run within many different virtual machine (host) platforms

Virtual Machines (Cont)

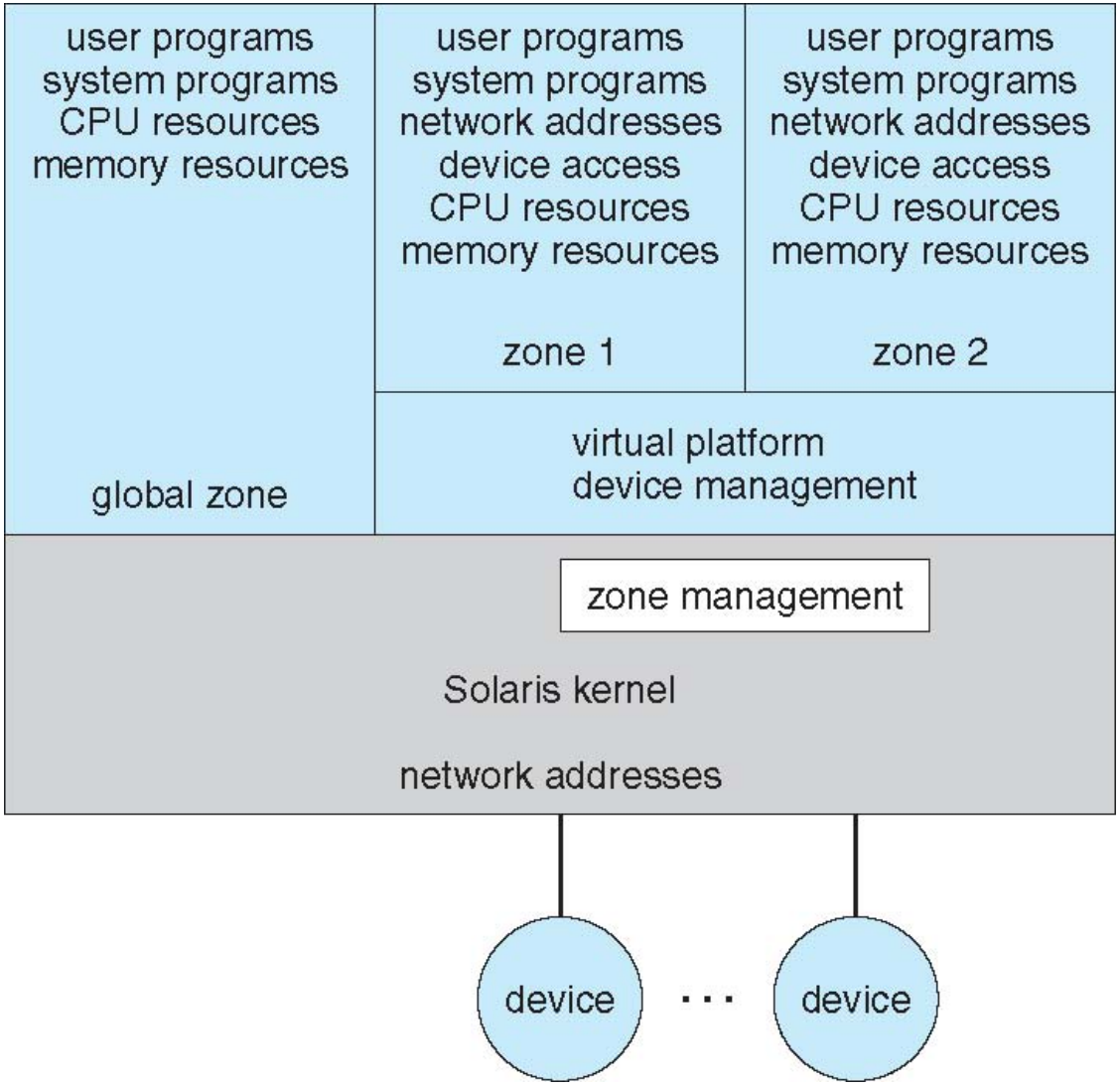


(a) Nonvirtual machine (b) virtual machine

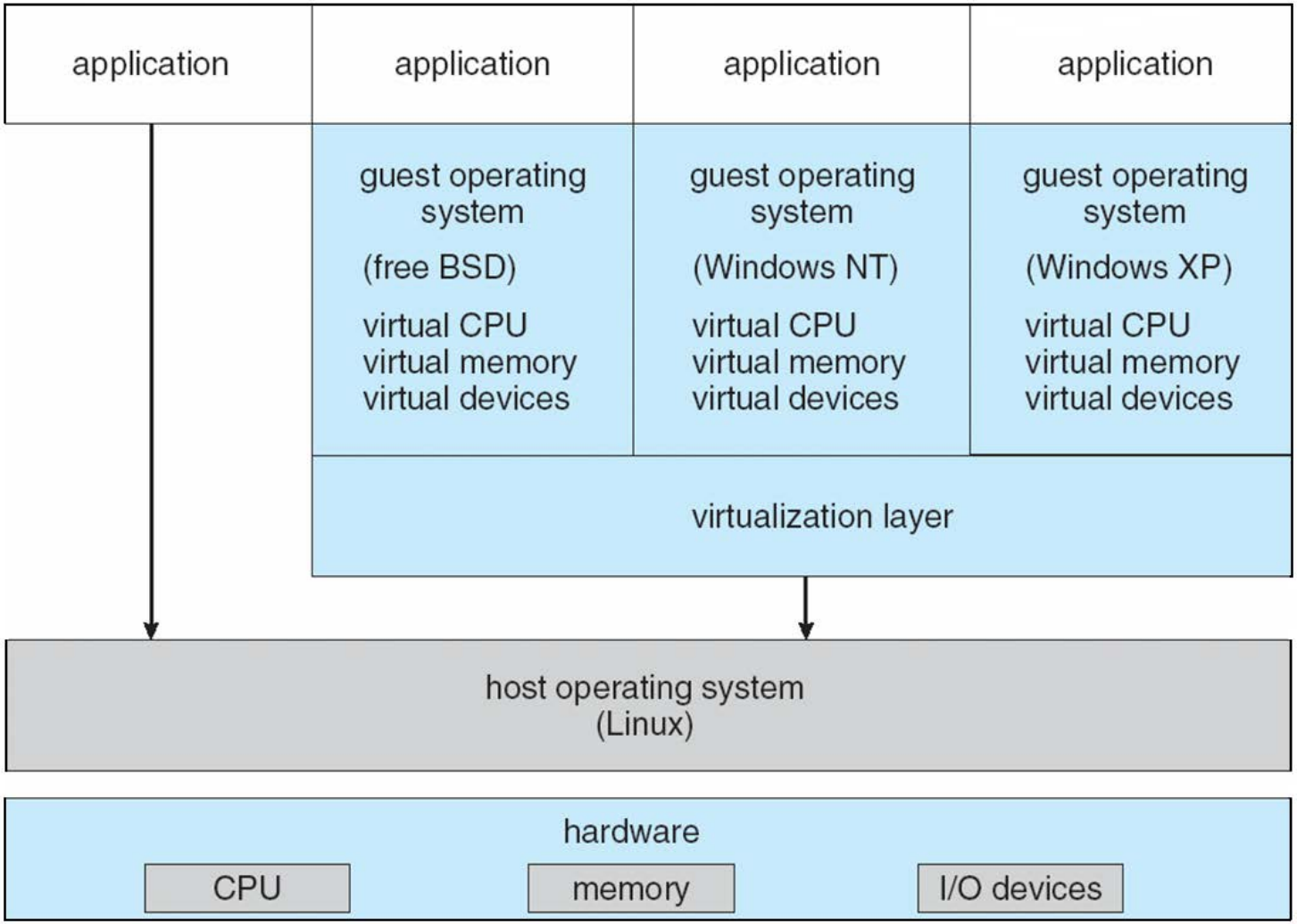
Para-virtualization

- Presents guest with system similar but not identical to hardware
- Guest must be modified to run on paravirtualized hardwareF
- Guest can be an OS, or in the case of Solaris 10 applications running in containers

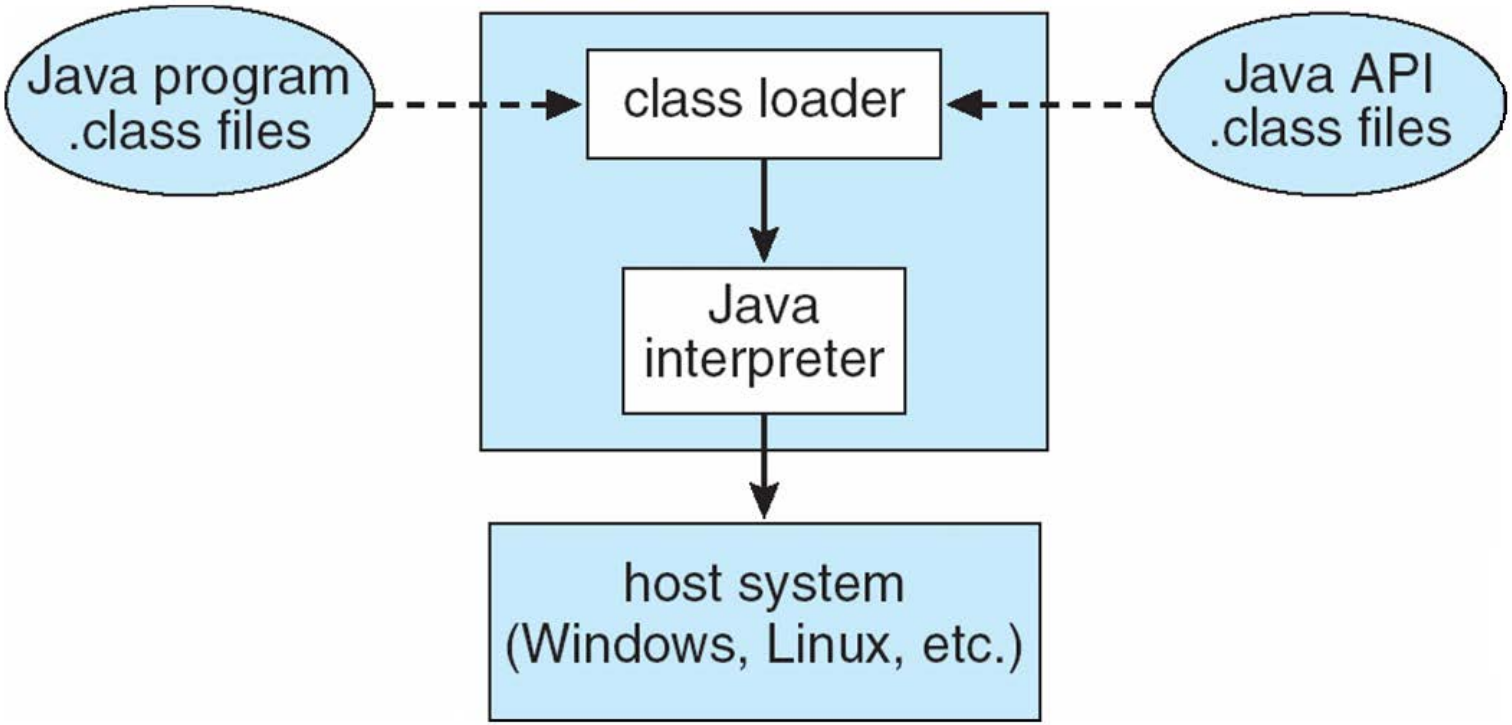
Solaris 10 with Two Containers



VMware Architecture



The Java Virtual Machine



Emulation vs. Virtualization

bochs.sourceforge.net

THE CROSS PLATFORM IA-32 EMULATOR

Current Release: *Bochs 2.6*

bochs
think inside the bochs.

- Home Page
- SF Project Page

Search

Welcome to the Bochs IA-32 Emulator Project

Bochs is a highly portable open source IA-32 (x86) PC emulator written in C++, that runs on most popular platforms. It

developer.android.com/tools/devices/emulator.html

Training API Guides Reference Tools

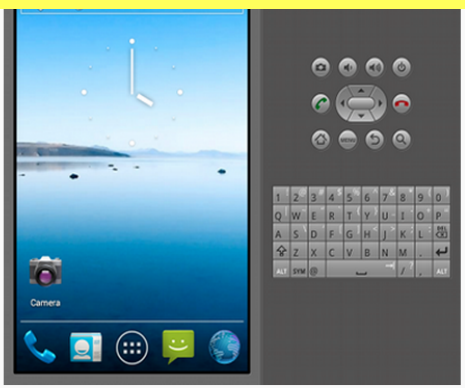
Developer Tools
Download
Workflow
Setting Up Virtual Devices
Setting Up Projects
Building and Running
Testing
Debugging
Publishing
Tools Help
Revisions
Extras
Samples

Using the Android Emulator

The Android SDK includes a virtual mobile device emulator that runs on your computer. The emulator lets you prototype, develop and test Android applications without using a physical device.

The Android emulator mimics all of the hardware and software features of a

ls. It
press" using
t also
with any



To let you model and test your application more easily, the emulator utilizes

What's the difference between these "emulators" and (Vmware,Virtualbox,..) ?

QEMU
open source processor emulator

Search

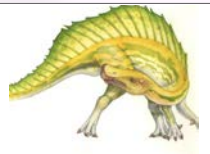
About
Home
Get
Download
Contribute
Start Here

Main Page

QEMU is a generic and open source machine emulator

When used as a machine emulator, QEMU can run OSes and programs made for one machine (e.g. an ARM board) on a different machine (e.g. your own PC). By using dynamic translation, it achieves very good performance.

When used as a virtualizer, QEMU achieves near native performances by executing the guest code directly on the host CPU. QEMU supports virtualization when executing under the Xen hypervisor or using the KVM



Emulation vs. Virtualization

```
add esp, 4
xor eax, eax
lgdt [gdtr]
jmp 0x8:complete_flush
```

emulator

```
while(1) {
I=instr_fetch();

switch(opcode(I)) {
case MOV:
…
case ADD:
…
}
}
```

Hardware

```
add esp, 4
xor eax, eax
lgdt [gdtr]
jmp 0x8:complete_flush
```

virtualizer

```
lgdt( gdtr)
change_code_seg(0x8, complete_flush)
```

Hardware

Operating-System Debugging

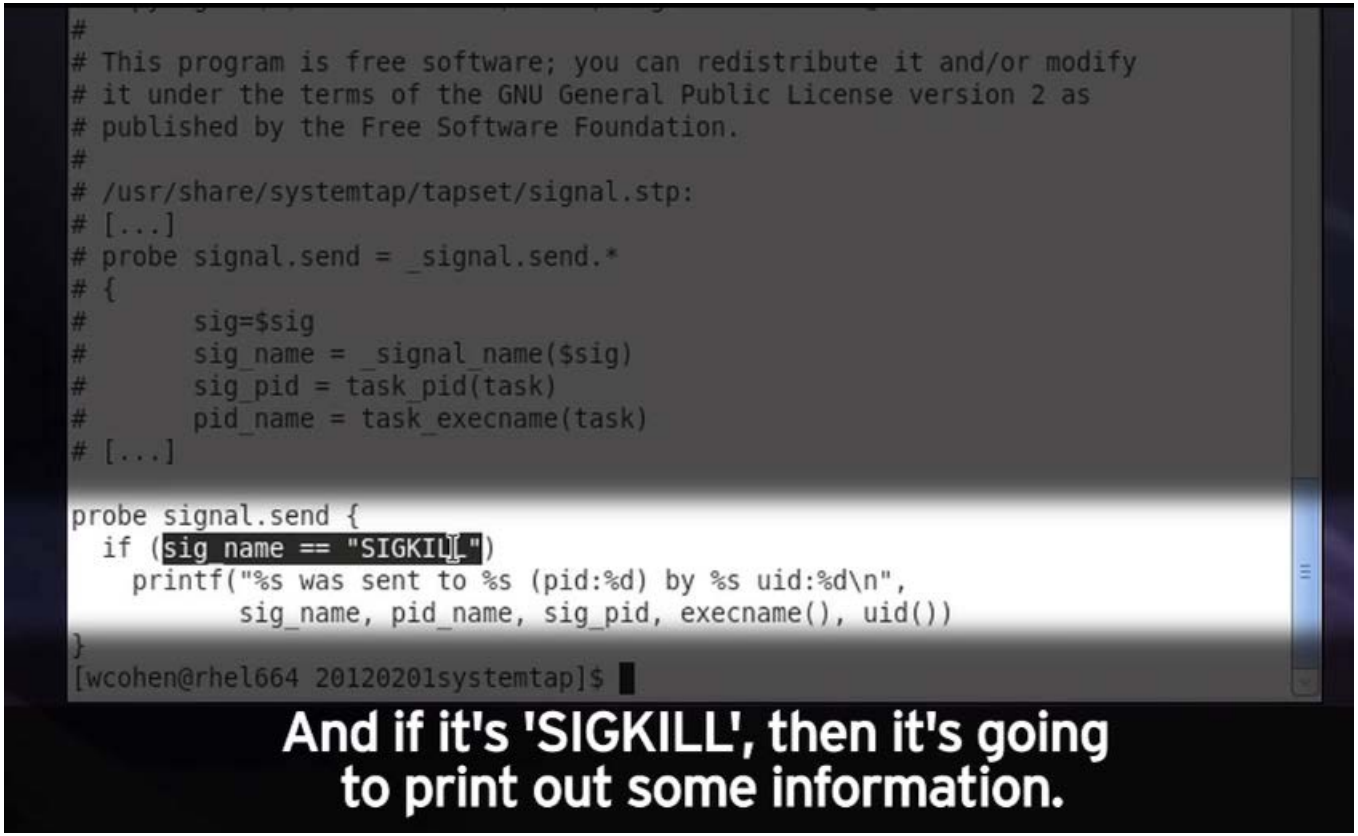
- Debugging is finding and fixing errors, or bugs
- OSes generate log files containing error information
- Failure of an application can generate core dump file capturing memory of the process
- Operating system failure can generate crash dump file containing kernel memory
- Beyond crashes, performance tuning can optimize system performance
- Kernighan's Law: "Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."
- DTrace tool in Solaris, FreeBSD, Mac OS X allows live instrumentation on production systems
  - Probes fire when code is executed, capturing state data and sending it to consumers of those probes

Solaris 10 dtrace Following System Call

```
# ./all.d 'pgrep xclock' XEventsQueued
dtrace: script './all.d' matched 52377 probes
CPU FUNCTION
  0 -> XEventsQueued                       U
  0   -> _XEventsQueued                    U
  0     -> _X11TransBytesReadable          U
  0     <- _X11TransBytesReadable          U
  0     -> _X11TransSocketBytesReadable    U
  0     <- _X11TransSocketBytesreadable    U
  0     -> ioctl                           U
  0       -> ioctl                         K
  0         -> getf                        K
  0           -> set_active_fd             K
  0           <- set_active_fd             K
  0         <- getf                        K
  0         -> get_udatamodel              K
  0         <- get_udatamodel              K
...
  0         -> releasef                    K
  0           -> clear_active_fd           K
  0           <- clear_active_fd           K
  0           -> cv_broadcast              K
  0           <- cv_broadcast              K
  0         <- releasef                    K
  0       <- ioctl                         K
  0     <- ioctl                           U
  0   <- _XEventsQueued                    U
  0 <- XEventsQueued                       U
```

Linux SystemTap

- The “Dtrace” for Linux
- http://sourceware.org/systemtap/
- http://www.youtube.com/watch?v=WSylxDC6obQ



Linux SystemTap

```
#
# /usr/share/systemtap/tapset/signal.stp:
# [...]
# probe signal.send = _signal.send.*
# {
#       sig=$sig
#       sig_name = _signal_name($sig)
#       sig_pid = task_pid(task)
#       pid_name = task_execname(task)
# [...]

probe signal.send {
  if (sig_name == "SIGKILL")
    printf("%s was sent to %s (pid:%d) by %s uid:%d\n",
           sig_name, pid_name, sig_pid, execname(), uid())
}
[wcohen@rhel664 20120201systemtap]$ stap /usr/share/doc/systemtap-1.6/examples/process/sigkill.stp
SIGKILL was sent to firefox (pid:3093) by killall uid:500
```

Now the next question is:
what is causing the killall?

Operating System Generation

- Operating systems are designed to run on any of a class of machines; the system must be configured for each specific computer site

- SYSGEN program obtains information concerning the specific configuration of the hardware system
- *Booting* – starting a computer by loading the kernel
- *Bootstrap program* – code stored in ROM that is able to locate the kernel, load it into memory, and start its execution

System Boot

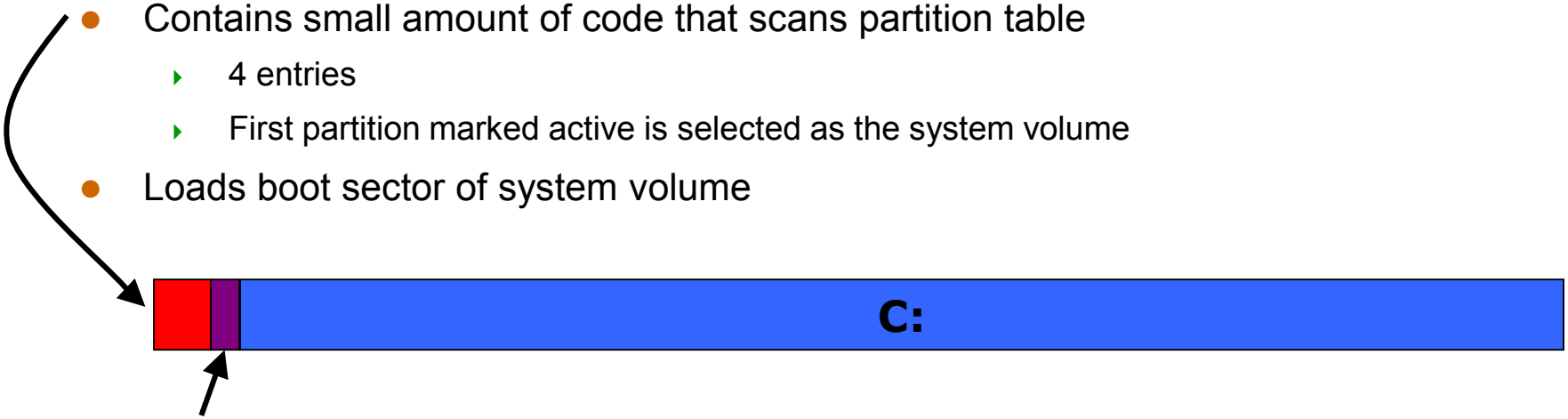
- Operating system must be made available to hardware so hardware can start it
  - Small piece of code – **bootstrap loader**, locates the kernel, loads it into memory, and starts it
  - Sometimes two-step process where **boot block** at fixed location loads bootstrap loader
  - When power initialized on system, execution starts at a fixed memory location
    - Firmware used to hold initial boot code

x86 and x64 Windows Boot Process

- Boot begins during installation when Setup writes various things to disk
- System volume:
  - Master Boot Record (MBR)
  - Boot sector
  - NTLDR – NT Boot Loader
  - NTDETECT.COM
  - BOOT.INI
  - SCSI driver – Ntbootdd.sys (not present on all systems)
- Boot volume:
  - System files – %SystemRoot%: Ntoskrnl.exe, Hal.dll, etc.

The Boot Process

1. BIOS
   - Reads MBR from boot device
2. MBR
   - Contains small amount of code that scans partition table
     - 4 entries
     - First partition marked active is selected as the system volume
   - Loads boot sector of system volume



3. Boot sector (NT-specific code)
   - Reads root directory of volume and loads NTLDR

x86 and x64 Boot Process

4. NTLDR
   - Moves system from 16-bit to 32-bit mode and enables paging
   - Reads and uses Ntbootdd.sys to perform disk I/O if the boot volume is on a SCSI disk different than the system volume
     - This is a copy of the SCSI miniport driver used when the OS is booted
   - Reads Boot.ini
     - Boot.ini selections point to boot drive
     - Specifies OS boot selections and optional switches (most for debugging/troubleshooting) that passed to kernel during boot
   - If more than one selection, NTLDR displays boot menu (with timeout)
   - If you select a 64-bit installation, NTLDR moves the CPU into 64-bit mode

Boot Process

4. NTLDR (continued)
   - Once boot selection made, user can type F8 to get to special boot menu
     - Last Known Good, Safe modes, hardware profile, Debugging mode
   - NTLDR loads and executes Ntdetect.com to perform BIOS hardware detection (x86 and x64 only)
     - Later saved into HKLM\Hardware\Description
   - NTLDR loads:
     - Ntoskrnl.exe, Hal.dll, and Bootvid.dll (and Kdcom.dll for XP and later)
     - The registry SYSTEM hive (\Windows\System32\Config\System)
       - Later this becomes HKLM\System
     - Based on the SYSTEM hive, the boot drivers are loaded
       - Boot driver: critical to boot process (e.g. boot file system driver)
     - Transfers control to main entry point of Ntoskrnl.exe

Boot Process

D:\temp\ntoskrnl.asm - Sublime Text 2 (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

ntoskrnl.asm

```
531731    00000001402B46E5: CC                int         3
531732    00000001402B46E6: 66 66 0F 1F 84 00  nop         word ptr [rax+rax]
531733                      00 00 00 00
531734    KiSystemStartup:
531735    00000001402B46F0: 48 83 EC 38        sub         rsp,38h
531736    00000001402B46F4: 4C 89 7C 24 30     mov         qword ptr [rsp+30h],r15
531737    00000001402B46F9: 4C 8B FC           mov         r15,rsp
531738    00000001402B46FC: 48 89 0D 45 9A FF  mov         qword ptr [KeLoaderBlock],rcx
531739                      FF
531740    00000001402B4703: 48 8B 51 48        mov         rdx,qword ptr [rcx+48h]
531741    00000001402B4707: 48 8D 05 72 D7 F3  lea         rax,[1401F1E80h]
531742                      FF
531743    00000001402B470E: 48 85 D2           test        rdx,rdx
531744    00000001402B4711: 48 0F 44 D0        cmove       rdx,rax
531745    00000001402B4715: 48 89 51 48        mov         qword ptr [rcx+48h],rdx
531746    00000001402B4719: 4C 8B D2           mov         r10,rdx
531747    00000001402B471C: 48 81 EA 80 01 00  sub         rdx,180h
531748                      00
531749    00000001402B4723: 48 89 52 18        mov         qword ptr [rdx+18h],rdx
531750    00000001402B4727: 4C 89 52 20        mov         qword ptr [rdx+20h],r10
531751    00000001402B472B: 41 0F 20 C0        mov         r8,cr0
531752    00000001402B472F: 4C 89 82 C0 01 00  mov         qword ptr [rdx+1C0h],r8
531753                      00
```

.* Aa “” KiSystemStartup Find Find Prev Find All

1 of 2 matches Spaces: 2 Plain Text

The Boot Process (cont)

5. Ntoskrnl.exe (splash screen appears)
   - Initializes kernel subsystems in two phases:
     - First phase is object definition (process, thread, driver, etc)
     - Second builds on the base that the objects provide
     - This is done in the context of a kernel-mode system thread that becomes the idle thread
   - I/O Manager starts boot-start drivers and then loads and starts system-start drivers

Driver Load Order

- Every driver has a key in HKLM\System\CurrentControlSet\Services
  - Type: 1 for driver, 2 for file system driver, others are Win32 services
  - Start: 0 = boot, 1 = system, 2 = auto, 3 = manual, 4 = disabled
- Some drivers need fine-grained control over load order to satisfy dependencies with other drivers
  - A driver's optional Group value controls load order within a start phase (boot, system, auto)
    - HKLM\System\CurrentControlSet\Control\ServiceGroupOrder
  - A driver's optional Tag value control's startup within its group
  - Note: Plug-and-play (discussed in the I/O section) controls load order of PnP drivers
- Special case: the file system driver for the boot volume is always loaded and started, regardless of what its start type is
- Lab: run LoadOrd from Sysinternals to see driver ordering

Boot Process

5. Ntoskrnl.exe (continued)
   - Creates the Session Manager process (\Windows\System32\Smss.exe), the first user-mode process
6. Smss.exe:
   - Runs programs specified in BootExecute e.g. autochk, the native API version of chkdsk
   - Processes “Delayed move/rename” commands
     - Used to replace in-use system files by hotfixes, service packs, etc.
   - Initializes the paging files and rest of Registry (hives or files)
   - Loads and initializes kernel-mode part of Win32 subsystem (Win32k.sys)
   - Starts Csrss.exe (user-mode part of Win32 subsystem)
   - Starts Winlogon.exe

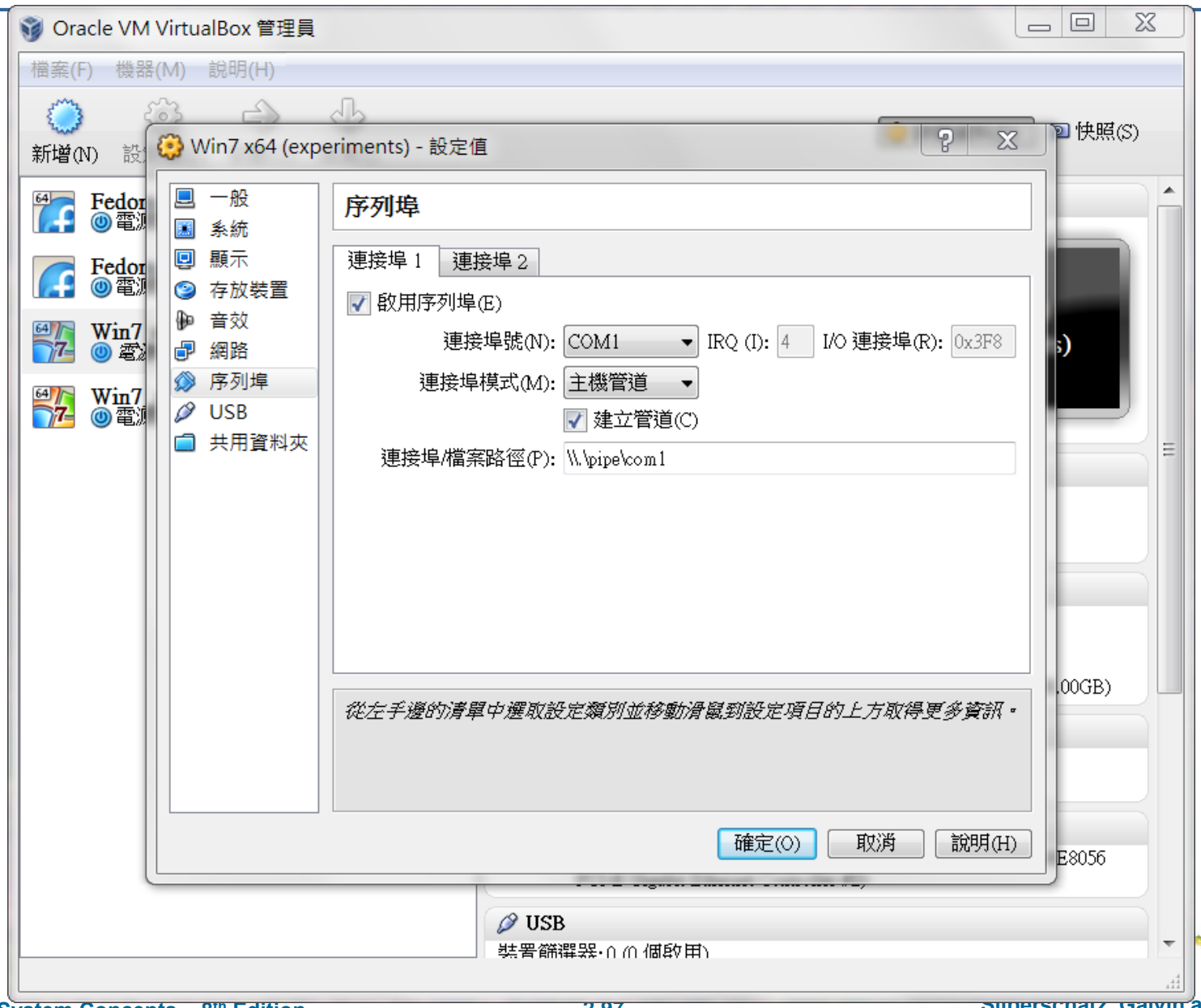
Boot Process

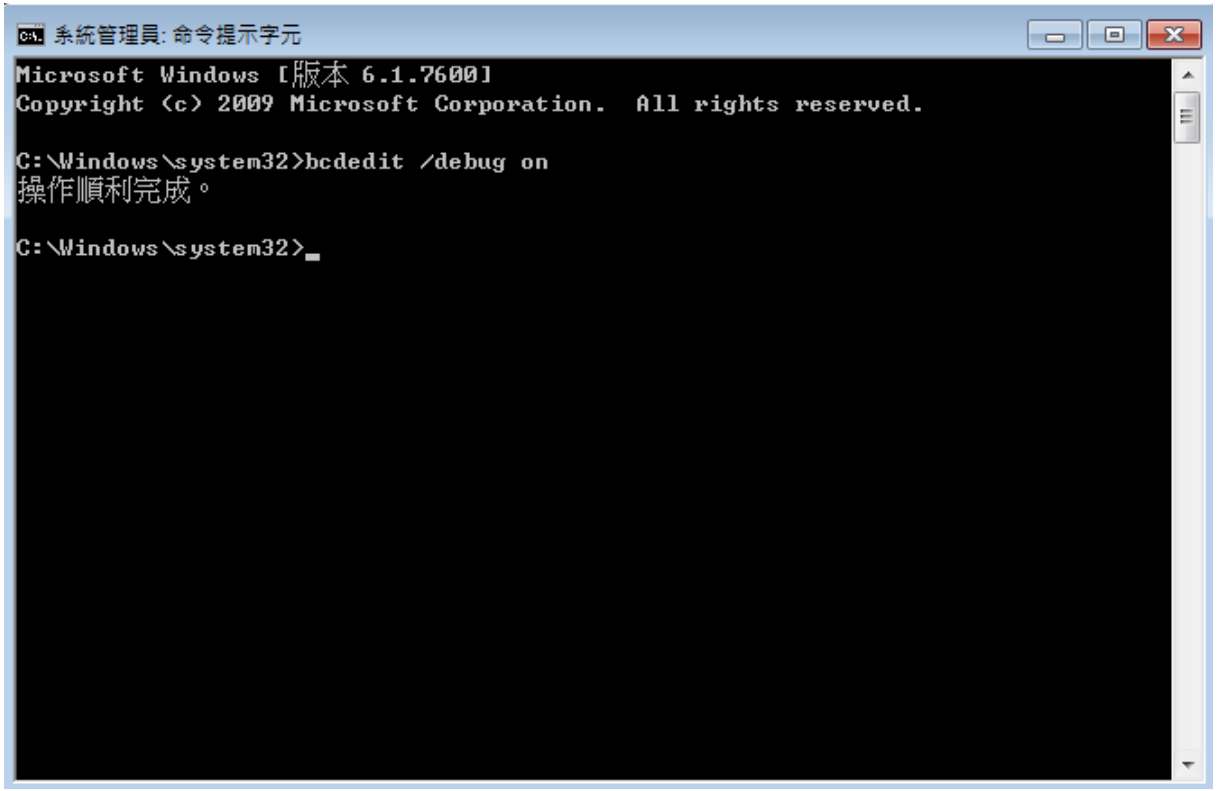
7. Winlogon.exe:
   - Starts Lsass.exe (Local Security Authority)
   - Loads GINA DLL (Graphical Identification and Authentication)
     - Default is Msgina.dll
     - Displays logon dialog
   - Starts Services.exe (the service controller)
8. Services.exe starts Win32 services marked as “automatic” start
   - Also includes any drivers marked Automatic start
   - Service startup continues asynchronous to logons

- End of normal boot process

Tools and Resources

- Debugging Tools for Windows
  - http://www.microsoft.com/whdc/devtools/debugging/default.mspx
- Sysinternals
  - http://technet.microsoft.com/zh-tw/sysinternals
- Virtualbox
  - http://www.virtualbox.org/


Oracle VM VirtualBox 管理員
檔案(F)
機器(M)
說明(H)
新增(N)
快照(S)
Win7 x64 (experiments) - 設定值
一般
系統
顯示
存放裝置
音效
網路
序列埠
USB
共用資料夾
序列埠
連接埠 1
連接埠 2
啟用序列埠(E)
連接埠號(N):
COM1
IRQ (I):
4
I/O 連接埠(R):
0x3F8
連接埠模式(M):
主機管道
建立管道(C)
連接埠/檔案路徑(P):
\\.\pipe\com1
從左手邊的清單中選取設定類別並移動滑鼠到設定項目的上方取得更多資訊。
確定(O)
取消
說明(H)
USB




```
windbg –b –k com:pipe,port=\\.\pipe\com1,resets=0 –y
srv*c:\symbols\*http://msdl.microsoft.com/download/symbols
```

Win7 x64 (experiments) [執行中] - Orac

機器(M) 裝置(D) 說明(H)

資源回收筒

DAEMON Tools Lite

Mozilla Firefox

Kernel 'com:pipe,port=\\.\pipe\com1,resets=0' - WinDbg:6.12.0002.633 AMD64

File Edit View Debug Window Help

Command - Kernel 'com:pipe,port=\\.\pipe\com1,resets=0' - WinDbg:6.12.0002.633 AMD64

```
Microsoft (R) Windows Debugger Version 6.12.0002.633 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.

Opened \\.\pipe\com1
Waiting to reconnect...
Connected to Windows 7 7600 x64 target at (Sun Feb 27 09:33:45.493 2011 (UTC + 8:00)), ptr64 TRUE
Kernel Debugger connection established.  (Initial Breakpoint requested)
Symbol search path is: srv*c:\symbols\*http://msdl.microsoft.com/download/symbols
Executable search path is:
Windows 7 Kernel Version 7600 MP (1 procs) Free x64
Product: WinNt, suite: TerminalServer SingleUserTS
Built by: 7600.16617.amd64fre.win7_gdr.100618-1621
Machine Name:
Kernel base = 0xfffff800`03a4f000 PsLoadedModuleList = 0xfffff800`03c8ce50
Debug session time: Sun Feb 27 09:33:43.105 2011 (UTC + 8:00)
System Uptime: 0 days 0:08:09.843
Break instruction exception - code 80000003 (first chance)
*******************************************************************************
*                                                                             *
*   You are seeing this message because you pressed either                    *
*       CTRL+C (if you run kd.exe) or,                                        *
*       CTRL+BREAK (if you run WinDBG),                                       *
*   on your debugger machine's keyboard.                                      *
*                                                                             *
*                   THIS IS NOT A BUG OR A SYSTEM CRASH                       *
*                                                                             *
* If you did not intend to break into the debugger, press the "g" key, then   *
* press the "Enter" key now.  This message might immediately reappear.  If it *
* does, press "g" and "Enter" again.                                          *
*                                                                             *
*******************************************************************************
nt!RtlpBreakWithStatusInstruction:
fffff800`03ab77a0 cc              int     3
```

kd>

Ln 0, Col 0 | Sys 0:KdSrv:S | Proc 000:0 | Thrd 000:0 | ASM | OVR | CAPS | NUM

2011/2/27

Right Ctrl

End of Chapter 2