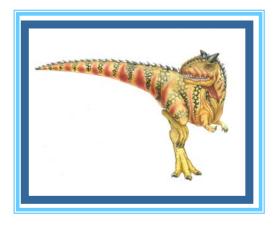
Chapter 2: Operating-System Structures



Operating System Concepts – 8th Edition,

Silberschatz, Galvin and Gagne ©2009

Chapter 2: Operating-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





- 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.



A View of Operating System Services

	user and	d other system pr	ograms			
	GUI	batch	command line			
user interfaces						
system calls						
program I/O file communication resource allocation accounting						
error detection services protection security						
operating system						
hardware						



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.



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
 - lcons 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)



Bourne Shell Command Interpreter

					ł	Tern	ninal				8	
<u>F</u> ile	<u>E</u> dit	View	<u>T</u> erminal	Ta <u>b</u> s	<u>H</u> elp							
fd0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0		-
sd0		0.0	0.2	0.0	0.2	0.0	0.0	0.4	0	0		
d1		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0		- 1
			exten	ded de	vice s	tatist	tics					- 1
devic	e	r/s	w/s	kr/s	kw/s	wait	actv	svc_t	%w	%b		- 1
Fd0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0		- 1
sd0		0.6	0.0	38.4	0.0	0.0	0.0	8.2	0	0		- 1
sd1		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0		- 1
otal root (/va	: 1. @pbg r/tm	1G all -nv64- p/syst	located -vm)-(12 tem-cont	+ 190M /pts)- ents/s	l reser (00:53 cripts	ved = 15-31)# up1	un-200 time	used, 3 17)-(g10	obal)			
root (/va 12:5	: 1. @pbg r/tm 3am @pbg	1G all -nv64- p/sys1 up 9 -nv64-	located -vm)-(12 tem-cont min(s), -vm)-(13	+ 190M /pts)- ents/s 3 us /pts)-	(00:53 cripts ers, (00:53	ved = 15-Ji)# upt load a 15-Ji	1.3G un-200 time averag	used, 1 17)-(g10 1e: 33.2	obal) 29, 6	7.68		
root (/va 12:5 (root (/va	: 1. @pbg r/tm 3am @pbg r/tm	1G all -nv64- p/sys1 up 9 -nv64- p/sys1	located -vm)-(12 tem-cont min(s), -vm)-(13 tem-cont	+ 190M /pts)- ents/s 3 us /pts)- ents/s	l reser (00:53 cripts ers, (00:53 cripts	ved = 15-Ji)# upt load a 15-Ji)# w	1.3G un-200 time averag un-200	used, 3)7)-(g10)e: 33.2)7)-(g10	obal) 29, 6 obal)	7.68	, 36.81	
root (/va 12:5 (root -(/va	: 1. @pbg r/tm 3am @pbg r/tm 7pm	1G all -nv64- p/sys1 up 9 -nv64- p/sys1	located -vm)-(12 tem-cont min(s), -vm)-(13 tem-cont	+ 190M /pts)- ents/s 3 us /pts)- ents/s , 15:2	l reser (00:53 cripts ers, (00:53 cripts	ved = 15-Ju)# upt load a 15-Ju)# w users	1.3G un-200 time averag un-200 , loa	used, 3)7)-(g10)e: 33.7)7)-(g10)d avera	obal) 29, 6 obal)	7.68		
(root -(/va 12:5 (root -(/va 4:0 Jser	: 1. @pbg r/tm 3am @pbg r/tm 7pm	1G all -nv64- p/sys1 up 9 -nv64- p/sys1 up 17	located -vn)-(12 ten-cont min(s), -vn)-(13 ten-cont 7 day(s)	+ 190M /pts)- ents/s 3 us /pts)- ents/s , 15:2 logir	l reser (00:53 cripts ers, (00:53 cripts 4, 3	ved = 15-Ji)# up load a 15-Ji)# w users e J(1.3G un-200 time averag un-200 , loa	used, 3)7)-(g10)e: 33.7)7)-(g10 d avera PCPU v	obal) 29, 6 obal) age: what	7.68 0.09	, 36.81	r/bi
(root (/va 12:5 (root (/va 4:0 Jser root	: 1. @pbg r/tm 3am @pbg r/tm 7pm	1G all -nv64- p/sys1 up 9 -nv64- p/sys1 up 17 tty	located -vn)-(12 ten-cont min(s), -vn)-(13 ten-cont 7 day(s)	+ 190M /pts)- ents/s 3 us /pts)- ents/s , 15:2 logir	(00:53 cripts ers, (00:53 cripts cripts 4, 3 @ idl	ved = 15-Ji)# up load a 15-Ji)# w users e J(1.3G un-200 time averag un-200 , loa CPU	used, 3)7)-(g10)e: 33.7)7)-(g10 d avera PCPU v	obal) 29, 6 obal) age: what	7.68 0.09	, 36.81 , 0.11, 8.66	r/bi
(root -(/va 12:5 (root -(/va 4:0	: 1. @pbg r/tm 3am @pbg r/tm 7pm	1G all -nv64- p/sys1 up 9 -nv64- p/sys1 up 17 tty	located -vm)-(12 tem-cont min(s), -vm)-(13 tem-cont 7 day(s) le	+ 190M /pts)- ents/s 3 us /pts)- ents/s , 15:2 logir	(00:53 cripts cripts cripts (00:53 cripts 4, 3 4, 3 4, 3 0 idl 718day	ved = 15-Ji)# up load a 15-Ji)# w users e J(1.3G un-200 time averag un-200 , loa CPU	used, 3)7)-(g10)e: 33.7)7)-(g10 ud avera PCPU v	obal) 29, 6 obal) age: what	7.68 0.09	, 36.81 , 0.11, 8.66	r/bi



The Mac OS X GUI

🧯 Grab File Edit Ca	pture Window Help		<u> </u>	• ==== 6	🖲 🖪 🖼 Man) 🗳 ⁺= Ω C 🔺 📢 15	:06 EDT Mon 2 Jul 💉 🕅 🔘
000		🗇 fig-dir		0	the second s	El
	1 2	0 0 0 0 0	0 🛒 🥄 📷 🙆 🔍 Selectio		11111107	Breachen Stript
Drop Stack +	🚔 * Faverites * Documents *	Music Movies Pictures Desktop Ap	olications = ZPBG = ZPBGE = iDisk =		(603.010)	
0	🔄 😳 2007-06.s10upgrade 👔	Desktop 🕲 Computer 🜘 fig-dir			10	and the second s
Empty				-		27 14 Mecentoshille
Volumes +	Name fig=1.05a.pd ⁺	▲ Kind ◯ FDF	Date Modified 6/24/07, 1:05PM 1	Size Application 06.1 KB Sk m		TELE POSTERIE
Network	# fig-1.0Ea svg	O SVG		77KB Inkscape		E FI
Freedom Stripe	👻 fig-2.0.png	Fortable Network Crachi		39.2 KB Freview	THE A	Unclused
Untitled	 fig=2.0.svg fig=2.0.s 	O SVG		L4 3 KB Inkscape 80 1 KB Preview	1.4	203/01
🔄 Untitled 2	101 mg ann	ALL PLANE MADE IN	A MARINE ALCON M	COLLAR TRAFER		-
@ ZPBG ±					TIFF	
en ZPBGE ± © iDisk ±					1107	Unitable 2 Udication de definee
IDisk Peter Baer Galvin's iPod						
Peter saler Galvin's Pou	•					0
						LEAST
						SHERE OF ALL AND ALL AND
					Proving Has	191
stat.*					lefe +	
T pbg					Name: fg-2.0a fg-2.0a tiff	Rater Stor Colorine Disc
A Applications					Kind: TIFF Document	White the state free
Documents Games					UTI: sublicitiff URL: dfs://	12
Utilities					CALVENT; PBC@BLACKS_N /ZPEG	9985
🗇 tmp					Size: 385 KB (901.236 bytes) data: 901.236 bytes	1680 ALC MED AC MER Rose
IIII Desktop					Physical: 881 KE (902,144	123
Favorites					zytes) Modified: Today, 2:53PM	
∲ Music ≝ Movies		E le History +	Selection Path	3+2	Attributes: Today, 2:5 3PM Owner: 2bg (101)	29308 303.37 GB - 35 GB free
Pictures	👻 fig 2.0a		👻 fig 2.0a		Croup: admin (\$0)	
📁 Sites			🗊 fig-dir		Permission: -nwx(700) Path: /Volumes/2P3G-1/imp/	, O
J# Public			beok		303k/os8-cir,fig-d r/ fig-2,0a.tiff	Journiar Brop Box
Preferences			imp		Application: Preview Volume: ZPBG	
J L brary			E ZPEG		Capacity: 735.4 CB	ល
projects					Free: 734.3 CB Format: SMB	Nevada b54
🗇 consult					Mount Point: /Volumes/2P8G-1	
	🗏 🗉 🔚 C Sitems	1 of 5 Items selected - 734.3 GB	weilable, 5.1 GB used	-1		
				and the second		So laris 10 allas
		000	ddress Book			
and the second se			Q Apple	0.00		
					A A Q operating system	(C) JEL Windows XP
			Apple Computer In		-	Windows XP Professional Mac
OOO PCa		Directories			Dictionary	
	n	Last Import		TO OT	o-er-at-ing sys-tem	
A00500000	i.	Last mport		no	un U V	G-Force 3.7.1 (Platinum).ding
				the	e software that supports a computer's basic func	lions.
Dec Hex Oct Bin Deg R	ad () RPN Info Tape		main 1-800-MY-APPLE		such as scheduling tasks, executing applications	and
x ² x ³ x ⁿ x ¹ Rn					controlling peripherals.	
			other 800-275-2273			and the second se
$\begin{array}{c c} \hline \hline \sqrt{x} \hline \sqrt{x} \hline \sqrt{x} \hline x^{1/n} \hline \hline 1/x \hline x \\ \hline \end{array}$	6 <u>x~Μ</u> π		home page http://www.appie.com			and the second
7 8 9 + 4	C M- Sin Cos Ta-		work 1 Infinite Loop			
4 5 6 - 0			Cupert no CA 95014	~		
			United States	4		(Second Second
1 2 3 × De			Mote	Ŧ		
0. Exp ÷ =	M re 10° 2° e°	E E	Edit 2 found	in.		Trast





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?

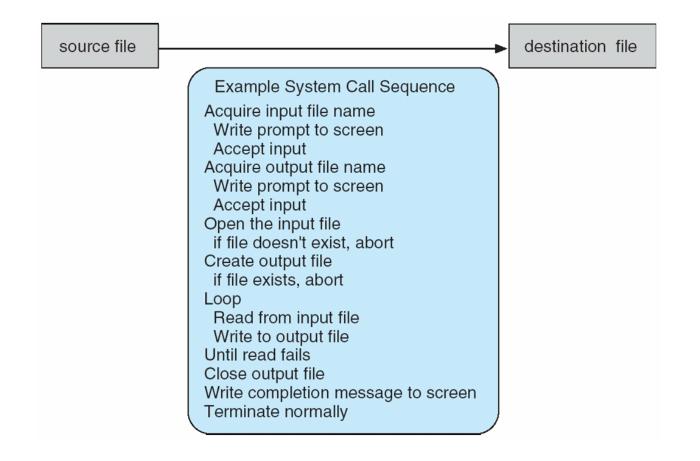
(Note that the system-call names used throughout this text are generic)





Example of System Calls

System call sequence to copy the contents of one file to another file

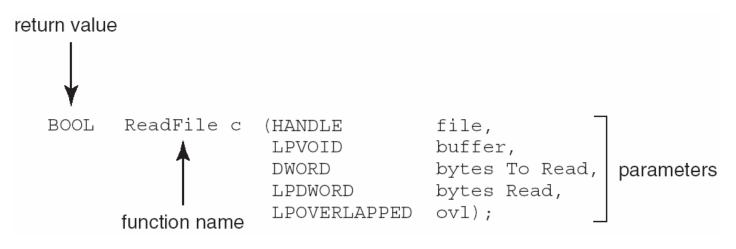






Example of Standard API

- 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



Silberschatz, Galvin and Gagne ©2009

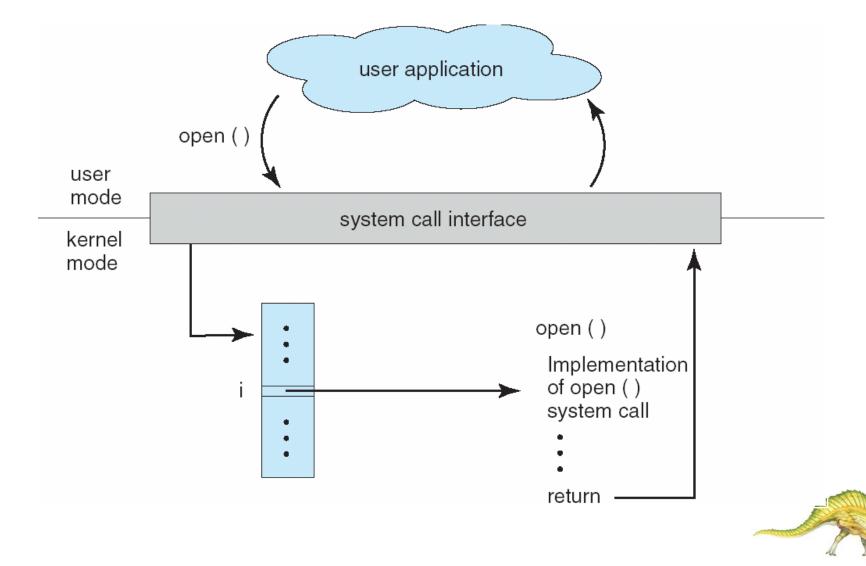


System Call Implementation

- 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)







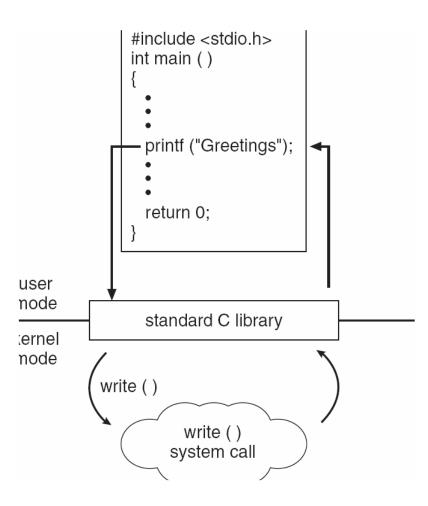
Operating System Concepts – 8th Edition

Silberschatz, Galvin and Gagne ©2009



Standard C Library Example

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





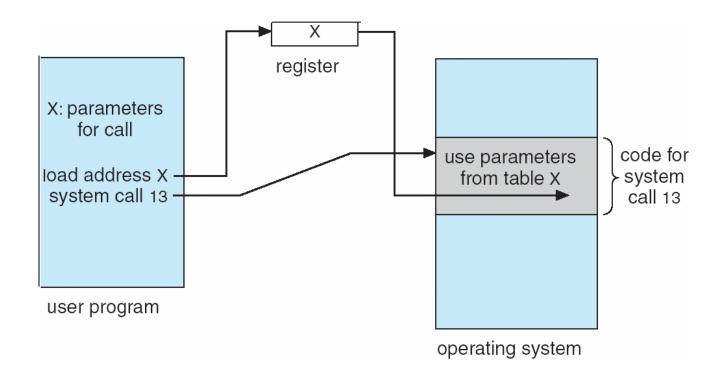
System Call Parameter Passing

- 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







Types of System Calls

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



Examples of Windows and Unix System Calls

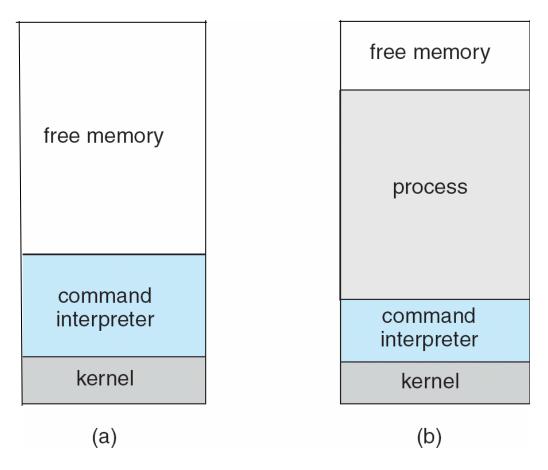
	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information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



Silberschatz, Galvin and Gagne ©2009



MS-DOS execution



(a) At system startup (b) running a program



Operating System Concepts – 8th Edition

Silberschatz, Galvin and Gagne ©2009



process D
free memory
process C
interpreter
process B
kernel





- System programs provide a convenient environment for program development and execution. The 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

- 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



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





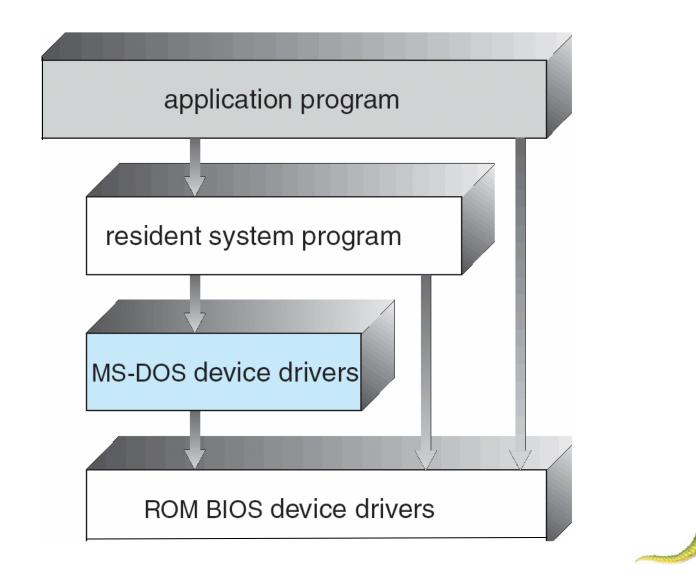
Simple Structure

- 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



Operating System Concepts – 8th Edition

Silberschatz, Galvin and Gagne ©2009



- 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

		(the users)					
	shells and commands compilers and interpreters system libraries						
ſ	syster	m-call interface to the ke	ernel				
Kernel	signals terminal handling character I/O system terminal drivers	file system swapping block I/O system disk and tape drivers	CPU scheduling page replacement demand paging virtual memory				
l	kernel interface to the hardware						
	terminal controllers terminals	device controllers disks and tapes	memory controllers physical memory				

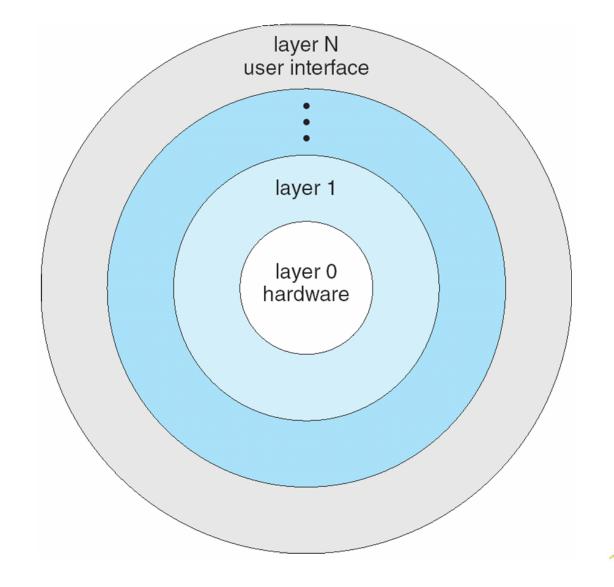




- UNIX limited by hardware functionality, the original UNIX operating system had limited structuring. The UNIX OS consists of two separable parts
 - Systems programs
 - The kernel
 - Consists of everything below the system-call interface and above the physical hardware
 - Provides the file system, CPU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



Layered Operating System



Silberschatz, Galvin and Gagne ©2009



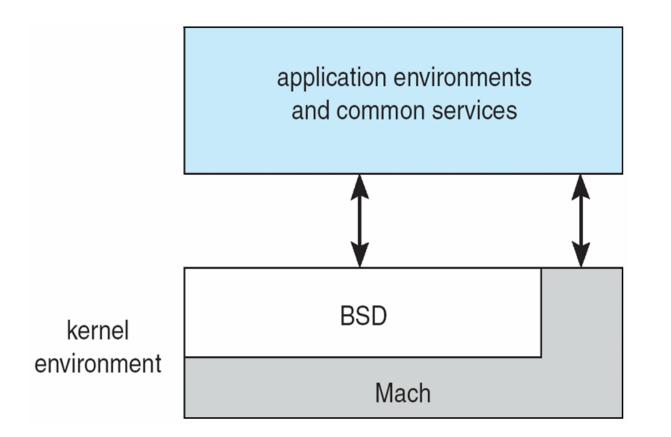
Microkernel System Structure

- 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





Mac OS X Structure





Operating System Concepts – 8th Edition

Silberschatz, Galvin and Gagne ©2009



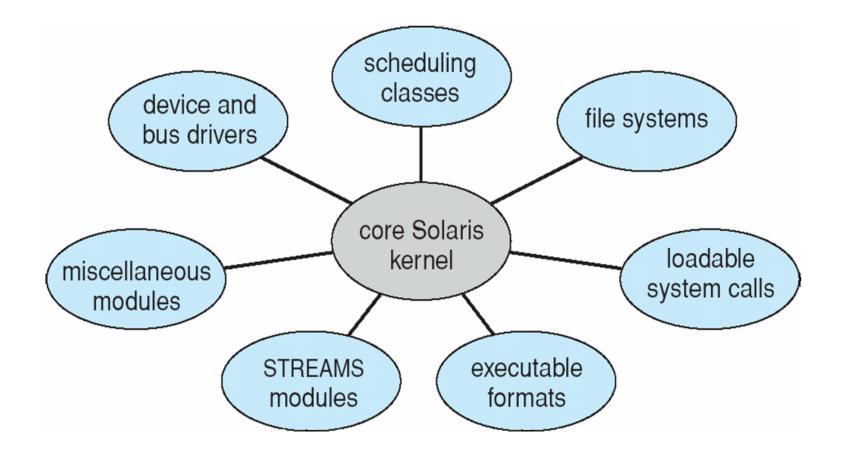
Modules

- 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 Modular Approach





Operating System Concepts – 8th Edition



- 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



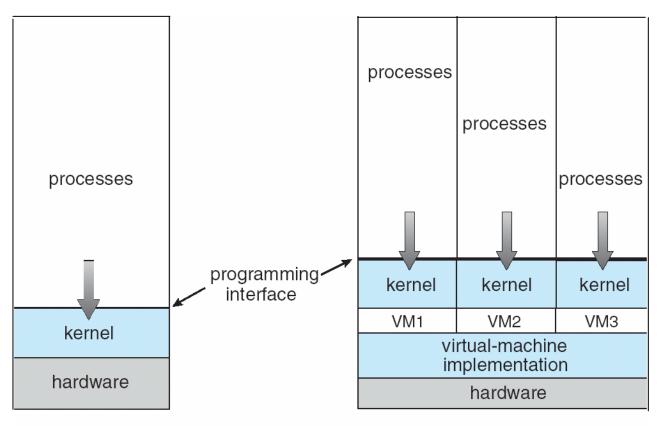
Virtual Machines History and Benefits

- 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)

(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

user programs system programs CPU resources memory resources	user programs system programs network addresses device access	user programs system programs network addresses device access			
	CPU resources memory resources zone 1	CPU resources memory resources zone 2			
global zone	virtual platform device management				
zone management					
Solaris kernel					
network addresses					
device device					

Operating System Concepts – 8th Edition



VMware Architecture

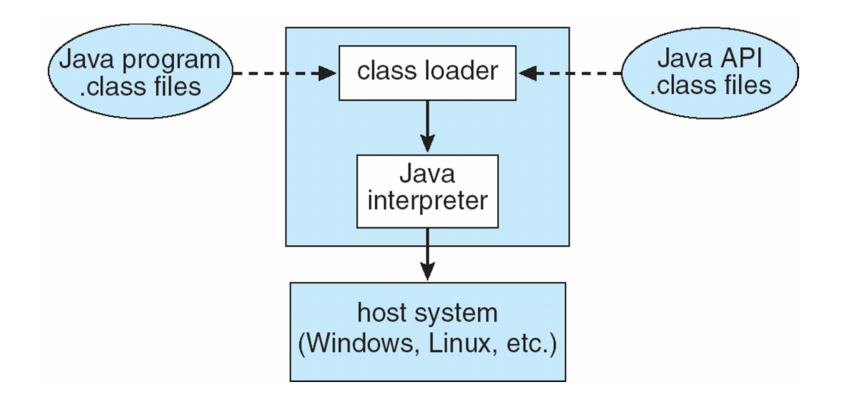
applic	cation	application	application	application		
		guest operating system (free BSD) virtual CPU virtual memory virtual devices	guest operating system (Windows NT) virtual CPU virtual memory virtual devices	guest operating system (Windows XP) virtual CPU virtual memory virtual devices		
\downarrow \downarrow						
host operating system (Linux)						
hardware						

hardware
CPU memory I/O devices

Operating System Concepts – 8th Edition



The Java Virtual Machine







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





# ./all.d `pgrep xclock` XEventsQueued dtrace: script './all.d' matched 52377 CPU FUNCTION	probes
0 -> XEventsQueued	U
0 -> XEventsQueued	U
0	U
0 <- X11TransBytesReadable	U
0 -> _X11TransSocketBytesReadable	e U
0 <- X11TransSocketBytesreadable	e U
0 -> ioctl	U
0 -> ioctl	K
0 -> getf	K
0 -> set_active_fd	K
0 <- set_active_fd	K
0 <- getf	K
0	K
0 <- get_udatamodel	K
0 -> releasef	K
0	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< td=""><td>U</td></xeventsqueued<>	U
0 <- XEventsQueued	U



Operating System Concepts – 8th Edition



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

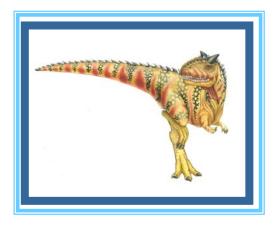




- Operating system must be made available to hardware so hardware can start it
 - Small piece of code bootstrap loader, locates the kernel, loads it into memory, and starts it
 - Sometimes two-step process where **boot block** at fixed location loads bootstrap loader
 - When power initialized on system,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



End of Chapter 2



Operating System Concepts – 8th Edition,