

CS307&CS356: Operating Systems

Dept. of Computer Science & Engineering Chentao Wu wuct@cs.sjtu.edu.cn





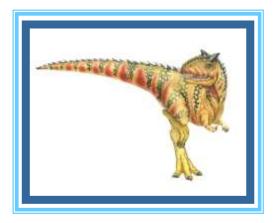


Download lectures

- <u>ftp://public.sjtu.edu.cn</u>
- User: wuct
- Password: wuct123456

• http://www.cs.sjtu.edu.cn/~wuct/os/

Chapter 2: Operating-System Structures



Chapter 2: Operating-System Structures

- Operating System Services
- User and Operating System-Interface
- System Calls
- System Services
- Linkers and Loaders
- Why Applications are Operating System Specific
- Operating-System Design and Implementation
- Operating System Structure
- Building and Booting an Operating System
- Operating System Debugging



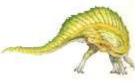


- Identify services provided by an operating system
- Illustrate how system calls are used to provide operating system services
- Compare and contrast monolithic, layered, microkernel, modular, and hybrid strategies for designing operating systems
- Illustrate the process for booting an operating system
- Apply tools for monitoring operating system performance
- Design and implement kernel modules for interacting with a Linux kernel





- Operating systems provide an environment for execution of programs and services to programs and users
- 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), touch-screen, 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



Operating System Services (Cont.)

- One set of operating-system services provides functions that are helpful to the user (Cont.):
 - **File-system manipulation** The file system is of particular interest. Programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.
 - **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 CPU cycles, main memory, file storage, I/O devices.
 - Logging 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

A View of Operating System Services

		user and other	system programs		
		GUI tou	ch screen command	line	
		user i	nterfaces		
		syst	em calls		
program execution	I/O operations	file systems	communication	resource allocation	accounting
	ror ction	se	rvices	a	rection ind curity
		operat	ing system		





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



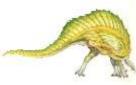


Bourne Shell Command Interpreter

🔴 😑 💿 1. root@r6181-d5-us01:~ (ssh)							
× root@r6181-d5-u • #1 × ssh 💥 #2 × root@r6181-d5-us01 #3							
Last login: Thu Jul 14 08:47:01 on ttys002 iMacPro:~ pbg\$ ssh root@r6181-d5-us01 root@r6181-d5-us01's password:							
Last login: Thu Jul 14 06:01:11 2016 from 172.16.16.162							
[root@r6181-d5-us01 ~]# uptime							
06:57:48 up 16 days, 10:52, 3 users, load average: 129.52, 80.33, 56.55							
[root@r6181-d5-us01 ~]# df -kh Filesystem Size Used Avail Use% Mounted on							
/dev/mapper/vg_ks-lv_root							
50G 19G 28G 41% /							
tmpfs 127G 520K 127G 1% /dev/shm							
/dev/sda1 477M 71M 381M 16% /boot							
/dev/dssd0000 1.0T 480G 545G 47% /dssd_xfs							
tcp://192.168.150.1:3334/orangefs							
12T 5.7T 6.4T 47% /mnt/orangefs							
/dev/gpfs-test 23T 1.1T 22T 5% /mnt/gpfs							
[root@r6181-d5-us01 ~]# [root@r6181-d5-us01 ~]# ps aux sort -nrk 3,3 head -n 5							
root 97653 11.2 6.6 42665344 17520636 ? S <ll 166:23="" bin="" jul13="" lpp="" mmfs="" mmfsd<="" td="" usr=""><td></td></ll>							
root 69849 6.6 0.0 0 0 ? S Jul12 181:54 [vpthread-1-1]							
root 69850 6.4 0.0 0 0 ? S Jul12 177:42 [vpthread-1-2]							
root 3829 3.0 0.0 0 0? S Jun27 730:04 [rp_thread 7:0]							
root 3826 3.0 0.0 0 0 ? S Jun27 728:08 [rp_thread 6:0]							
[root@r6181-d5-us01 ~]# ls -l /usr/lpp/mmfs/bin/mmfsd							
-r-x 1 root root 20667161 Jun 3 2015 /usr/lpp/mmfs/bin/mmfsd							
[root@r6181-d5-us01 ~]#							



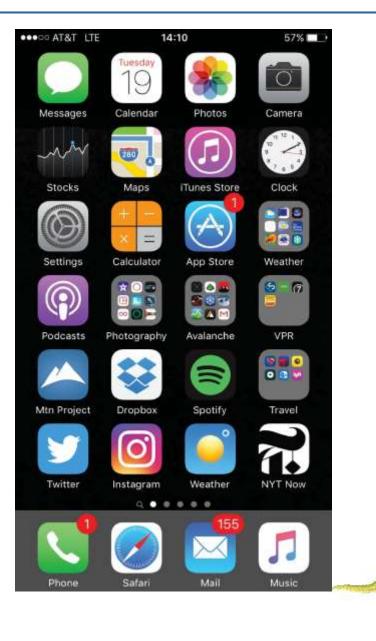
- User-friendly desktop metaphor interface
 - Usually mouse, keyboard, and monitor
 - Icons represent files, programs, actions, etc
 - Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory (known as a folder)
 - Invented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





Touchscreen Interfaces

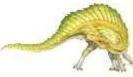
- n Touchscreen devices require new interfaces
 - I Mouse not possible or not desired
 - Actions and selection based on gestures
 - Virtual keyboard for text entry
 - Voice commands





The Mac OS X GUI

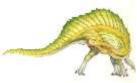
A second second second database database second second second second				+		THE STREET	and the second second			
and the state of t	11 41	In the second		1.1.1	Contra I	I IN AN ADDRESS	A BOARD	241 24		
2 mg 2 byok 2 saTD-dt 2 m text-dt -	1001	13 - are - ov Oc-dr - taxt-di - Chasters - ch2 - 11111 and					a arran a car			
1.010	10111-01111	- Alwan	bacher ward, ore per-		10.010 h	[No.		Sector Sector	1 10	100
7544	2541.13074	a, DEX OTABIN	1.2 Mill Michael, Horonte 1.2 Mill Michael, Horonte		1 A 447 THE	1.04amt		Tatas, 2-09 PM		214.001
anial anial	\$214.5354a	5 Million (1990)	1.0 Mar			Burners 10x1 2010-00-00 at 3.00.17 PM	1.1	Talks 5100 PM		1111 1100
el citie	1014011-00346	at their Delaws	T.C.M. TTV/R.MOTH		李西	+ 3cmim 6hot 2018-03-09 at 3:05.15 PM		Tales, 2412 PM	15.0 km	PHO maps
	101-14, 101-148 (\$1-45, 1014-148)	S 17 Am			and the second se	Scient Star 2019-09-09 at 3:05.15 PM		Talley, 2405 PM	7.0180	PNG stage
400	Total & had be	a films	9, dictonery		1	 Bitem Stat 2018-08-08-212-37/04/96 	4	Long. 2111 PM	7,3368	Phil Kase
the levis	1015-010-00	Linoi-in	(west)			+ Colligation IV General Its Sector		taby, 1081W		1.000
-19.44	41.41.2 50.04	A PROVIDENCE	Contract Contract			 Bussier, Solut 2019; CH+ CH+ CH+ CH 41 10, 30, 72 A Screen 10x4 2019; 2019; 20, 70, 41 10, 24 19; 3 		Talley 10:30 459 Talley, 10:35 444	100 40	1912 (1944) 1912 (1944)
192	41171.037.46	5.01.000	- B derke beint niger tellerari	Station of the		Sciven Shot 3018-09-09 et 10.12.407		Talley, 10-12-4M		PHG mage
9108	311/14,508 248	is billion.	- Diction & Deserv	ALC: NO. OF		+ Chronofiyne Doclamenta		Talay, 2494 AM		Parents.
2 M2 1 M2	\$1517 X 218-00 \$1517 X 518 34	4.3.1996 3.3.1996	- Digner & Denny	Aa		2 Roman 2012 20 TE-128-128 at \$123.00 Ph		Termine, 5.15 FV	187.00	The second
100	22271,005,000	vi E Harr	1	1		= 3cmm Skit 2018-00-08 at 9.24.42 PM		Yearson 0.54764		1910 armint
1042	\$11/1 AD-08 \$11/1 A 328 M	6. 2.7148 3. V. Tana	Permanent er ser na	and the second sec		 Actuary Shot 2016-00-08 at 3 12.4778 at sector performance v53 	4	Technology 3 12 MM		HIG maker
	121/10.008 14	a class a Liter.	 ddukoto transv-meter ti 	Distancey	1	 Integration performance vical Integration (Sept 2018-03-03 art \$2.09.18 PK) 		Terrerials, 2 28-704	110 100	
• · · · · · · · · · · · · · · · · · · ·	MATE AR 16	a L'hor	Antopycki fatromate.	and the second se		= 50mm/66at 2018-09-08 at 508.41 Ph		Perforation, 1 (18,714)	TTRI MAR	Ho mage
	1/11/19, 2010-048	4.1.19mm 4.111mm	The local sector of the lo			in Sceam Bhot 2018-00-06 at 5.08.22 PM		Weinley TOEPH	1110.028	Phil maps
•	MULTINE NO. NO.	with Man	1999.8			- Sceen Shat 2019-09-09 at 5/05/22 PB		TREBUIER, \$105 PM		Phil stage
a - 183	22213.025.000	10 EXten	 Introduction 	The manage	÷	 Buistan (Rol 2018-09-08 at 9.04.33 Pt) 		YORKING \$114 PG	118.308	
	\$1517 p. 628 mm. \$1517 h. 508 Mail	S. LDINE 9.1 Mate	(man)	Station Sphill	2	 Bureari Shot 2018-03-03 at 5.04.27 Pe Screen Shot 2018-09-08 at 5.04.09 Pe 		Yesterlan, 5 D4 PM	111 619	(115) = (a)p. (115), major
100	\$127.8.325-68	to Lithole	and and	Bratine 20019		= Schart Shot 2018-09-08 at 4.88.21 PM		Vesterates, 4.58 FM		PHS mage
-0124	11111.0476	a titler	B DEB JORTON - Disc			# 3umer/3his 2015-09-08 at 8.54 12 Pr		VALUE AND ADD FM		PROT IN MICH.
	\$251° 5, 578 360 \$332° 6, 543 46			_	100000	a: Borrer: Shar 2018-09-08 at 6.53.40 Ph	4	Yespenders, 8.2.2.P.M	111.000	(NG 9988
an	211115, 1121-04				1000 Accession	# Screen Sher 2010-00-06 er 4.44,3776		TETRITIC BALLING		Hid map.
185	111-1.118-00				and the second se	Scelan Shot 2018-09-08 at 4.42.42 PM	4	Terpenies, 8:62 (4)	210.00	Phili praga
	2012 12 835 No. 2012 No. 2012 No. 2012				The Parent of Street Trees	 Screen (But 2019-OF-OF of 4.41.37 PS		Yesherike, 8:31 PM Yoyikishe, 9:33 PM	79.10	Part mage
	10111-018-98				All row	a: Schen Sher 2018-00-00 at 3,47,41 PM		Termines, 147 PM		THE PAGE
199	8,151-5, 5 (5.76.76)				and the second se	a - Besserv Silver 2018-09-00 at 3.40.43 Pb		Venerates 344.04		Phù siaje
	11173.128.94				1000	Bussen Mint 2016 (29 (2) (11 (10))22 A	vid .	Yoleville, 37, 88 198	anna sea	TWO INVESTIGATION
raidy man	111-1 115 10				i interesting inte	12 Scenary Diver 2018-03-08 et 11-45-45.4	ew .	Yerleying, 17:48-454		1990 mage
	0.0710.028.000				1000 000	Screen Shot 2018-09-08 et 17.44 36.4	viet .	Televise, TLASAM		1940 mage
					(C) H	the state of the s		The second second		
	11175-111-10 11175-111-10		CALIFORNIA CONTRACTOR		All Party and		Lovier		100	
		NO. CO. CO. CO.	The submer of				ing / -			ALC: NOT
	ALC: L SUB-ME	at these Areas there have the	the second s		STATUS -	a manager - Brann - Cabin Branners	Longe and Longe on service	-		all states
		A Design And the loss the	the second s		STATUS -	a manager i brann - fran - brann ann - fran - brann ann - fran - brann	Lovier	-		Distantes Distantes
	And a second second	and have a second block have been	100-100 TO 100-100-100			a manager - Brann - Cabin Branners	Longe and Longe on service	-		
		A Design And the loss the	100-100 TO 100-100-100	here and in such		a manager i brann - fran - brann ann - fran - brann ann - fran - brann	Apply Compu	-		
		and have a second block have been	100-100 TO 100-100-100	here and in such		a manager i brann - fran - brann ann - fran - brann ann - fran - brann	Apple Compu	tel Inc.		
		and have a second block have been	100-100 TO 100-100-100	here and in such		a manager i brann - fran - brann ann - fran - brann ann - fran - brann	Apply Compu	tel Inc.		
		and have a second block have been	100-100 TO 100-100-100			a manager i brann - fran - brann ann - fran - brann ann - fran - brann	Apple Compu	tel Inc.		
		and have a second block have been			A The second sec	a manager i brann - fran - brann ann - fran - brann ann - fran - brann	Apple Computer	tel Inc.		
		and have a second block have been		100 mm	a Marine Second	a manager i brann - fran - brann ann - fran - brann ann - fran - brann	Apple Computer	tel Inc.		
		and have a second block have been		1000 000 00000 00000000000000000000000	M Backgourd 1	a manager i brann - fran - brann ann - fran - brann ann - fran - brann	Apple Computer	tel Inc.		
and a second sec		and have a second block have been		Internet in the second	a Marine Second	a manager i brann - fran - brann ann - fran - brann ann - fran - brann	Apple Computer	tel Inc.		
A A A A A A A A A A A A A A A A A A A		and have a second block have been		4 10 10 10 10 10 10 10 10 10 10 10 10 10	A Ministration of the second s	a manager i brann - fran - brann ann - fran - brann ann - fran - brann	Apple Computer	tel Inc.		
aperating system SourceHo January		and have a second block have been		4 10 10 10 10 10 10 10 10 10 10 10 10 10	ATT A Long the second s	a manager i brann - fran - brann ann - fran - brann ann - fran - brann	Apple Computer	tel Inc.	290	
Transaction for supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports and the supports a proceeding to value and with an end of the supports and the supports are supported to value and with an end of the supports are supports and the supports are supports a		and have a second block have been		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	And Annual State	a manager i brann - fran - brann ann - fran - brann ann - fran - brann	Annue A Darie Annue A Darie Compu- annie France A Mella annie	Get Inc.		2425
and and a second		and have a second block have been		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	A Ministration of the second s	a manager i brann - fran - brann ann - fran - brann ann - fran - brann	Apple Computer	tel Inc.		
Annual Control of Cont		and have a second block have been		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	And Annual State	a manager i brann - fran - brann ann - fran - brann ann - fran - brann	And Demonstrate Address Computer Address Computer and Address C	Set Inc.	290	2425
Annual Control of Cont				1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	And Annual State	a manager i brann - fran - brann ann - fran - brann ann - fran - brann	And Demonstrate Address Computer Address Computer and Address C	tief Inc.	290	2425
Transaction for supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports and the supports a proceeding to value and with an end of the supports and the supports are supported to value and with an end of the supports are supports and the supports are supports a		DSSD HIGH PERFOR		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	And Annual State	a manager i brann - fran - brann ann - fran - brann ann - fran - brann	And Demonstrate Address Computer Address Computer and Address C	Set Inc.	290	2425
Transaction for supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports and the supports a proceeding to value and with an end of the supports and the supports are supported to value and with an end of the supports are supports and the supports are supports a		DSSD HIGH PERFOR		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	And Annual State	a manager i brann - fran - brann ann - fran - brann ann - fran - brann	Lines In Line or one Apple Compu- instantion and Apple and Computed Apple and Com	541 Inc. 	290.	2425
Transaction for supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports and the supports a proceeding to value and with an end of the supports and the supports are supported to value and with an end of the supports are supports and the supports are supports a		DSSD HIGH PERFOR PARALLEL FILE SYST		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	And Annual State	a manager i brann - fran - brann ann - fran - brann ann - fran - brann	And Demonstrate Address Computer Address Computer and Address C	tief Inc.	290.	2425
Transaction for supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports a proceeding to value and with an end of the supports and the supports a proceeding to value and with an end of the supports and the supports are supported to value and with an end of the supports are supports and the supports are supports a		DSSD HIGH PERFOR		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	And Annual State	a manager i brann - fran - brann ann - fran - brann ann - fran - brann	Lines In Line acres Apple Compu- instruction Computer States Computer	541 Inc. 	290. 1	2425
Transcence (Second Second Seco		DSSD HIGH PERFOR PARALLEL FILE SYST CLICK TO EDIT MASTER SUBTITLE S		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	And Annual State	a manager i brann - fran - brann ann - fran - brann ann - fran - brann		660 Inc. 100 100 00 00 100 00 00 100 00 100 1	290	2425 1 9 8
Transcence (Second Second Seco		DSSD HIGH PERFOR PARALLEL FILE SYST CLICK TO EDIT MASTER SUBTITLE S		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	And Annual State	a manager i brann - fran - brann ann - fran - brann ann - fran - brann	Lines Internet and Annual Annua Annual Annual Annu	560 hrs. 	290	2425
Transmission of the second second distance of the second second distance of the second dist		DSSD HIGH PERFOR PARALLEL FILE SYST CLICK TO EDIT MASTER SUBSTITUE S		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	And Annual State	a manager i brann - fran - brann ann - fran - brann ann - fran - brann		660 Inc. 100 100 00 00 100 00 00 100 00 100 1		
Transmission of the second second distance of the second second distance of the second dist		DSSD HIGH PERFOR PARALLEL FILE SYST CLICK TO EDIT MASTER SUBSTITUE S		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	And Annual State	a manager i brann - fran - brann ann - fran - brann ann - fran - brann		660 Inc. 100 100 00 00 100 00 00 100 00 100 1		
Commence of the option of the company of the compa		DSSD HIGH PERFOR PARALLEL FILE SYST CLICK TO EDIT MASTER SUBSTITUE S		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	And Annual State	a manager i brann - fran - brann ann - fran - brann ann - fran - brann		660 Inc. 100 100 00 00 100 00 00 100 00 100 1		2425 1 9 8
Transmission of the second second distance of the second second distance of the second dist		DSSD HIGH PERFOR PARALLEL FILE SYST CLICK TO EDIT MASTER SUBSTITUE S		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	And Annual State	a manager i brann - fran - brann ann - fran - brann ann - fran - brann		See Inc.		
Commence of the option of the company o		DSSD HIGH PERFOR PARALLEL FILE SYST CLICK TO EDIT MASTER SUBSTITUE S		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	And Annual State	a manager i brann - fran - brann ann - fran - brann ann - fran - brann		660 Inc. 100 100 00 00 100 00 00 100 00 100 1		
Transmission of the second second distance of the second second distance of the second dist		DSSD HIGH PERFOR PARALLEL FILE SYST CLICK TO EDIT MASTER SUBSTITUE S			And Annual State	a manager i brann - fran - brann ann - fran - brann ann - fran - brann		See Inc.		





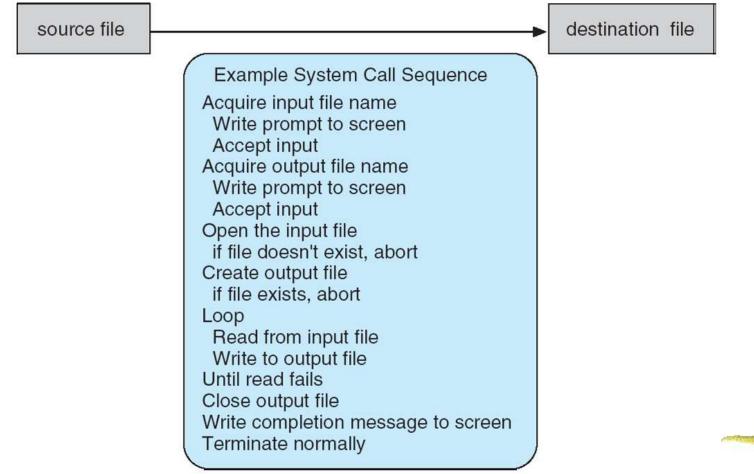
- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level Application Programming Interface (API) rather than direct system call use
- Three most common APIs are Win32 API for Windows, POSIX API for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

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





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





Example of Standard API

EXAMPLE OF STANDARD API

As an example of a standard API, consider the read() function that is available in UNIX and Linux systems. The API for this function is obtained from the man page by invoking the command

man read

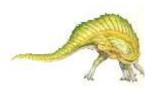
on the command line. A description of this API appears below:

#include	<unistd.h></unistd.h>	
ssize_t	read(int	fd, void *buf, size_t count)
return value	function name	parameters

A program that uses the read() function must include the unistd.h header file, as this file defines the ssize_t and size_t data types (among other things). The parameters passed to read() are as follows:

- int fd—the file descriptor to be read
- void *buf —a buffer into which the data will be read
- size_t count—the maximum number of bytes to be read into the buffer

On a successful read, the number of bytes read is returned. A return value of 0 indicates end of file. If an error occurs, read() returns -1.

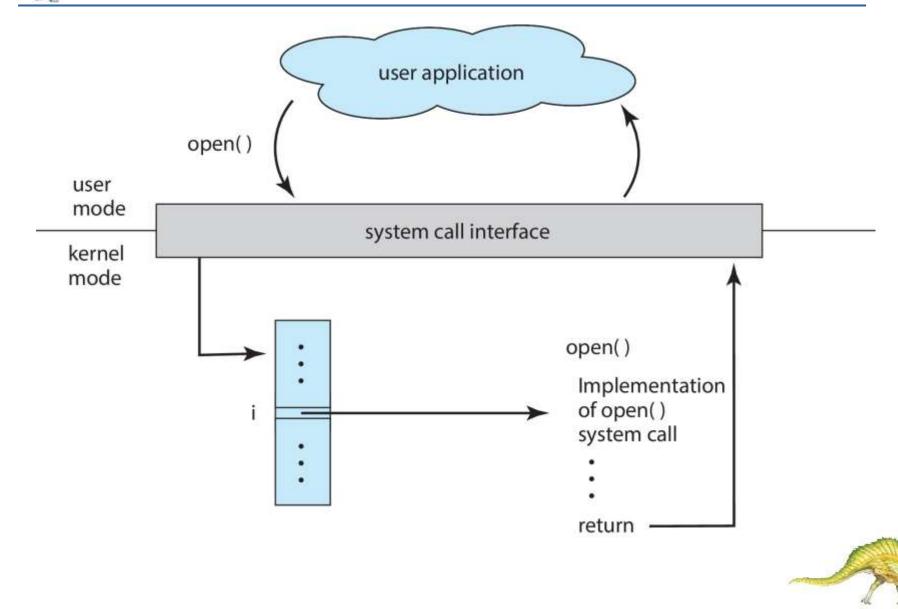




- Typically, a number associated with each system call
 - System-call interface maintains a table indexed according to these numbers
- The system call interface invokes the intended system call in OS kernel and returns status of the system call and any return values
- The caller need know nothing about 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)



API – System Call – OS Relationship





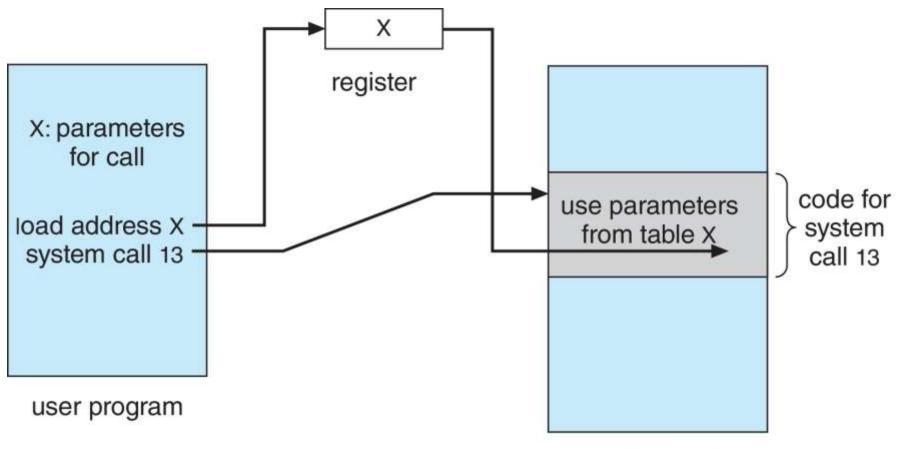
System Call Parameter Passing

- 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



operating system





Types of System Calls

Process control

- create process, terminate process
- end, abort
- Ioad, execute
- get process attributes, set process attributes
- wait for time
- wait event, signal event
- allocate and free memory
- Dump memory if error
- Debugger for determining bugs, single step execution
- Locks for managing access to shared data between processes



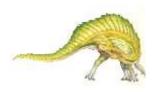
Types of System Calls (cont.)

- File management
 - create file, delete file
 - open, close file
 - read, write, reposition
 - get and set file attributes
 - Device management
 - request device, release device
 - read, write, reposition
 - get device attributes, set device attributes
 - logically attach or detach devices





- Information maintenance
 - get time or date, set time or date
 - get system data, set system data
 - get and set process, file, or device attributes
- Communications
 - create, delete communication connection
 - send, receive messages if message passing model to host name or process name
 - From client to server
 - Shared-memory model create and gain access to memory regions
 - transfer status information
 - attach and detach remote devices





Types of System Calls (Cont.)

Protection

- Control access to resources
- Get and set permissions
- Allow and deny user access

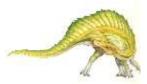




EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

The following illustrates various equivalent system calls for Windows and UNIX operating systems.

	Windows	Unix
Process	CreateProcess()	fork()
control	ExitProcess()	exit()
	WaitForSingleObject()	wait()
File	CreateFile()	open()
management	ReadFile()	read()
	WriteFile()	write()
	CloseHandle()	close()
Device	SetConsoleMode()	ioctl()
management	ReadConsole()	read()
	WriteConsole()	write()
Information	GetCurrentProcessID()	getpid()
maintenance	SetTimer()	alarm()
	Sleep()	<pre>sleep()</pre>
Communications	CreatePipe()	pipe()
	CreateFileMapping()	shm_open()
	MapViewOfFile()	mmap()
Protection	SetFileSecurity()	chmod()
	<pre>InitlializeSecurityDescriptor()</pre>	umask()
	SetSecurityDescriptorGroup()	chown()



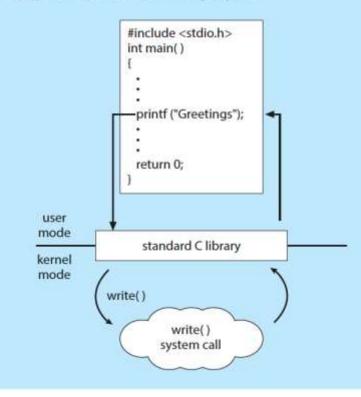


Standard C Library Example

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

THE STANDARD C LIBRARY

The standard C library provides a portion of the system-call interface for many versions of UNIX and Linux. As an example, let's assume a C program invokes the printf() statement. The C library intercepts this call and invokes the necessary system call (or calls) in the operating system—in this instance, the write() system call. The C library takes the value returned by write() and passes it back to the user program:

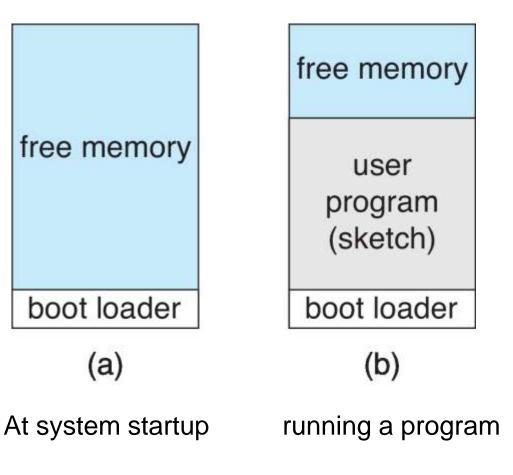


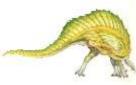




Example: Arduino

- Single-tasking
- No operating system
- Programs (sketch) loaded via USB into flash memory
- Single memory space
- Boot loader loads program
- Program exit -> shell reloaded







Example: FreeBSD

Unix variant	high	
Multitasking	memory	kernel
User login -> invoke user' s choice of shell		free memory
Shell executes fork() system call to create process		process C
 Executes exec() to load program into process 		interpreter
 Shell waits for process to terminate or continues with user commands 		process B
Process exits with:	í	
code = 0 – no error	low memory	process D
 code > 0 – error code 	And any of a straight and former 1	



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





System Services (cont.)

- 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 Services (Cont.)

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



System Services (Cont.)

Background Services

- Launch at boot time
 - Some for system startup, then terminate
 - Some from system boot to shutdown
- Provide facilities like disk checking, process scheduling, error logging, printing
- Run in user context not kernel context
- Known as services, subsystems, daemons

Application programs

- Don't pertain to system
- Run by users
- Not typically considered part of OS
- Launched by command line, mouse click, finger poke

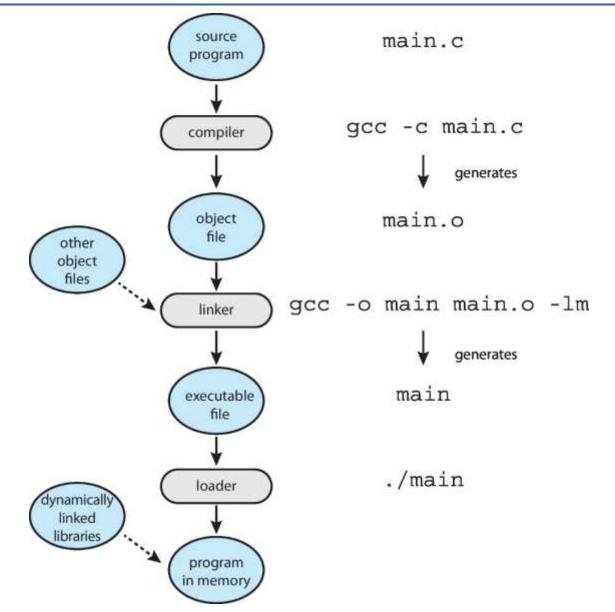




Linkers and Loaders

- Source code compiled into object files designed to be loaded into any physical memory location relocatable object file
- Linker combines these into single binary executable file
 - Also brings in libraries
- Program resides on secondary storage as binary executable
- Must be brought into memory by loader to be executed
 - Relocation assigns final addresses to program parts and adjusts code and data in program to match those addresses
- Modern general purpose systems don't link libraries into executables
 - Rather, dynamically linked libraries (in Windows, DLLs) are loaded as needed, shared by all that use the same version of that same library (loaded once)
- Object, executable files have standard formats, so operating system knows how to load and start them

The Role of the Linker and Loader





Why Applications are Operating System Specific

- Apps compiled on one system usually not executable on other operating systems
- Each operating system provides its own unique system calls
 - Own file formats, etc
- Apps can be multi-operating system
 - Written in interpreted language like Python, Ruby, and interpreter available on multiple operating systems
 - App written in language that includes a VM containing the running app (like Java)
 - Use standard language (like C), compile separately on each operating system to run on each
- Application Binary Interface (ABI) is architecture equivalent of API, defines how different components of binary code can interface for a given operating system on a given architecture, CPU, etc.



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 the design 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, errorfree, 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 (example – timer)
- Specifying and designing an OS is highly creative task of software engineering





- Much variation
 - Early OSes in assembly language
 - Then system programming languages like Algol, PL/1
 - Now C, C++
- Actually usually a mix of languages
 - Lowest levels in assembly
 - Main body in C
 - Systems programs in C, C++, scripting languages like PERL, Python, shell scripts
- More high-level language easier to port to other hardware
 - But slower
- Emulation can allow an OS to run on non-native hardware





Operating System Structure

- General-purpose OS is very large program
- Various ways to structure ones
 - Simple structure MS-DOS
 - More complex -- UNIX
 - Layered an abstraction
 - Microkernel -Mach





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





kernel

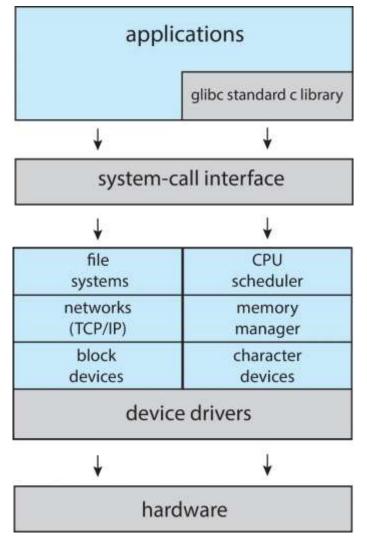
Traditional UNIX System Structure

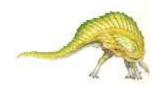
Beyond simple but not fully layered

(the users)						
	shells and commands compilers and interpreters system libraries system-call interface to the 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			
	kernel interface to the hardware					
	terminal controllers terminals	device controllers disks and tapes	memory controllers physical memory			



Monolithic plus modular design

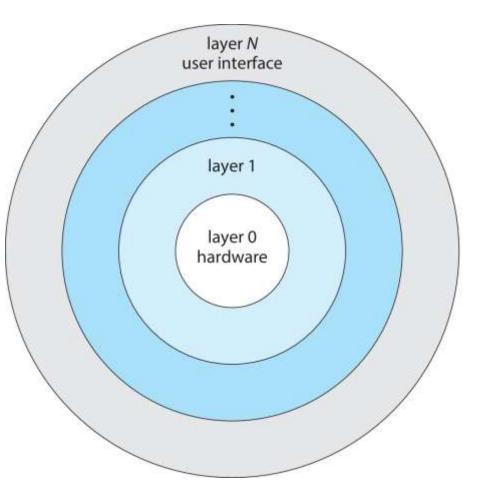






Layered Approach

- 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 lowerlevel layers



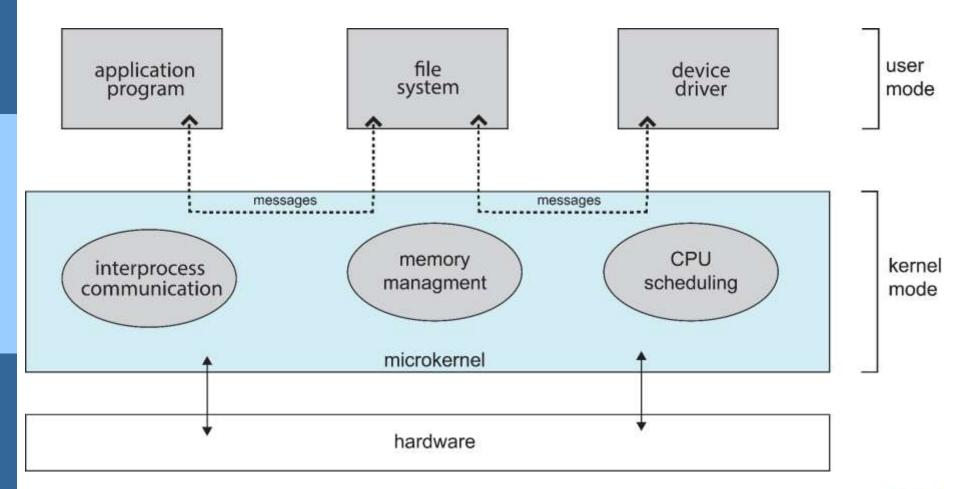


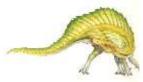


- Moves as much from the kernel into user space
- Mach example of microkernel
 - Mac OS X kernel (Darwin) partly based on Mach
- Communication takes place between user modules using message passing
- Benefits:
 - Easier to extend a microkernel
 - Easier to port the operating system to new architectures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











- Many modern operating systems implement loadable kernel modules (LKMs)
 - 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
 - Linux, Solaris, etc



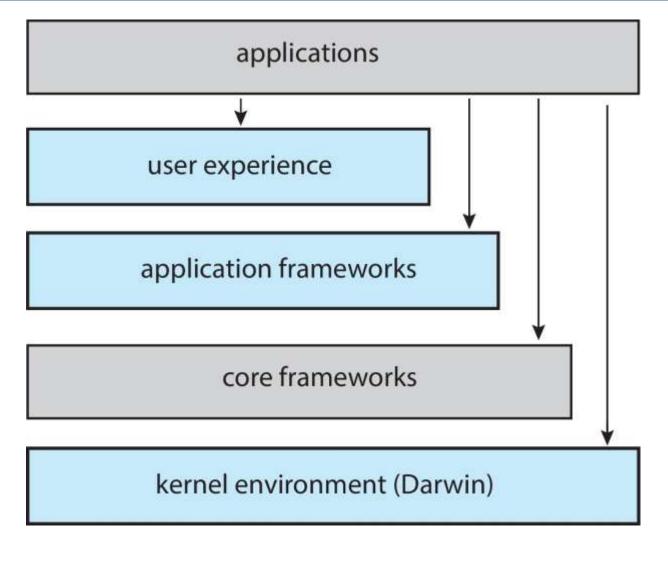


- Most modern operating systems are actually not one pure model
 - Hybrid combines multiple approaches to address performance, security, usability needs
 - Linux and Solaris kernels in kernel address space, so monolithic, plus modular for dynamic loading of functionality
 - Windows mostly monolithic, plus microkernel for different subsystem *personalities*
- Apple Mac OS X hybrid, layered, Aqua UI plus Cocoa programming environment
 - Below is kernel consisting of Mach microkernel and BSD Unix parts, plus I/O kit and dynamically loadable modules (called kernel extensions)



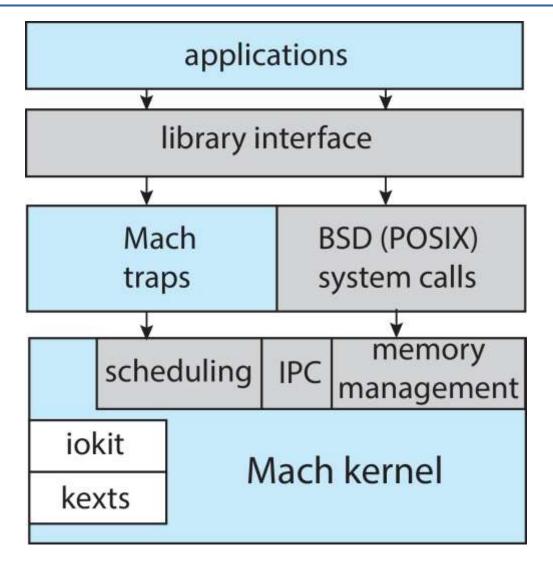


macOS and iOS Structure





Darwin









- Apple mobile OS for *iPhone*, *iPad*
 - Structured on Mac OS X, added functionality
 - Does not run OS X applications natively
 - Also runs on different CPU architecture (ARM vs. Intel)
 - Cocoa Touch Objective-C API for developing apps
 - Media services layer for graphics, audio, video
 - Core services provides cloud computing, databases
 - Core operating system, based on Mac OS X kernel

Cocoa Touch		
Media Services		
Core Services		
Core OS		





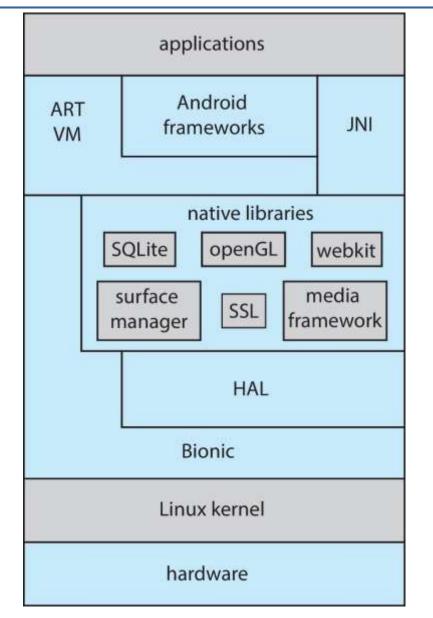
Android

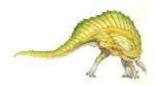
- Developed by Open Handset Alliance (mostly Google)
 - Open Source
- Similar stack to IOS
- Based on Linux kernel but modified
 - Provides process, memory, device-driver management
 - Adds power management
- Runtime environment includes core set of libraries and Dalvik virtual machine
 - Apps developed in Java plus Android API
 - Java class files compiled to Java bytecode then translated to executable than runs in Dalvik VM
- Libraries include frameworks for web browser (webkit), database (SQLite), multimedia, smaller libc





Android Architecture





Building and Booting an Operating System

- Operating systems generally designed to run on a class of systems with variety of perpherals
- Commonly, operating system already installed on purchased computer
 - But can build and install some other operating systems
 - If generating an operating system from scratch
 - Write the operating system source code
 - Configure the operating system for the system on which it will run
 - Compile the operating system
 - Install the operating system
 - Boot the computer and its new operating system





Building and Booting Linux

- Download Linux source code (<u>http://www.kernel.org</u>)
- Configure kernel via "make menuconfig"
- Compile the kernel using "make"
 - Produces vmlinuz, the kernel image
 - Compile kernel modules via "make modules"
 - Install kernel modules into vmlinuz via "make modules_install"
 - Install new kernel on the system via "make install"





- When power initialized on system, execution starts at a fixed memory location
- Operating system must be made available to hardware so hardware can start it
 - Small piece of code bootstrap loader, BIOS, stored in ROM or EEPROM locates the kernel, loads it into memory, and starts it
 - Sometimes two-step process where boot block at fixed location loaded by ROM code, which loads bootstrap loader from disk
 - Modern systems replace BIOS with Unified Extensible Firmware Interface (UEFI)
- Common bootstrap loader, GRUB, allows selection of kernel from multiple disks, versions, kernel options
- Kernel loads and system is then running
- Boot loaders frequently allow various boot states, such as single user mode



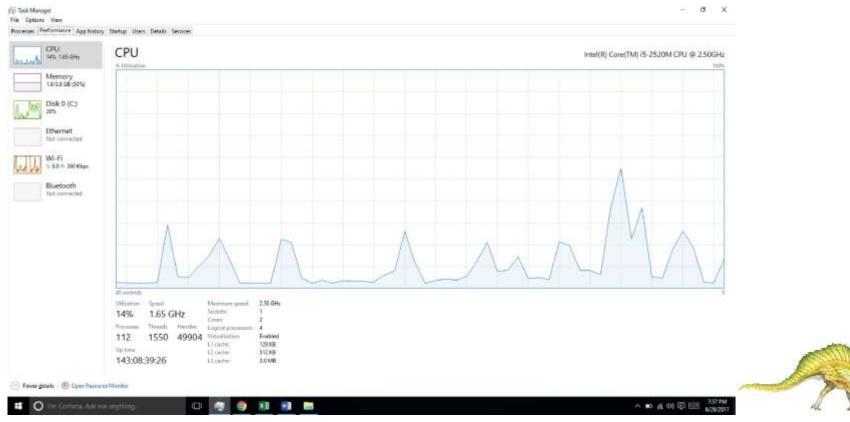
Operating-System Debugging

- Debugging is finding and fixing errors, or bugs
- Also performance tuning
- OS 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
 - Sometimes using *trace listings* of activities, recorded for analysis
 - Profiling is periodic sampling of instruction pointer to look for statistical trends

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."



- Improve performance by removing bottlenecks
- OS must provide means of computing and displaying measures of system behavior
- For example, "top" program or Windows Task Manager





Tracing

- Collects data for a specific event, such as steps involved in a system call invocation
- Tools include
 - strace trace system calls invoked by a process
 - gdb source-level debugger
 - perf collection of Linux performance tools
 - tcpdump collects network packets





BCC

- Debugging interactions between user-level and kernel code nearly impossible without toolset that understands both and an instrument their actions
- BCC (BPF Compiler Collection) is a rich toolkit providing tracing features for Linux
 - See also the original DTrace
- For example, disksnoop.py traces disk I/O activity

TIME(s)	Т	BYTES	LAT(ms)
1946.29186700	R	8	0.27
1946.33965000	R	8	0.26
1948.34585000	W	8192	0.96
1950.43251000	R	4096	0.56
1951.74121000	R	4096	0.35

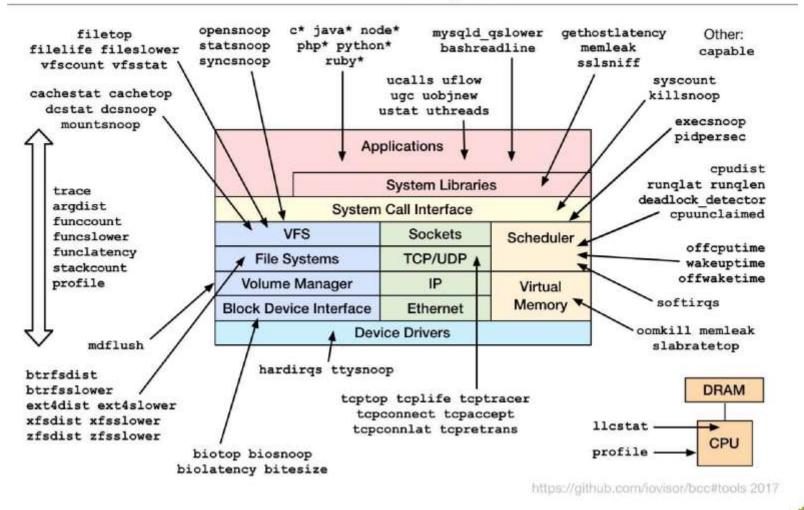
Many other tools (next slide)





Linux bcc/BPF Tracing Tools

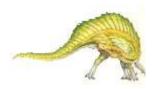
Linux bcc/BPF Tracing Tools





Exercises at the end of Chapter 2 (OS book)

• 2.2, 2.5, 2.7



End of Chapter 2

