

CPE 460 Operating System Design

Lecture 2: Operating Systems Structures

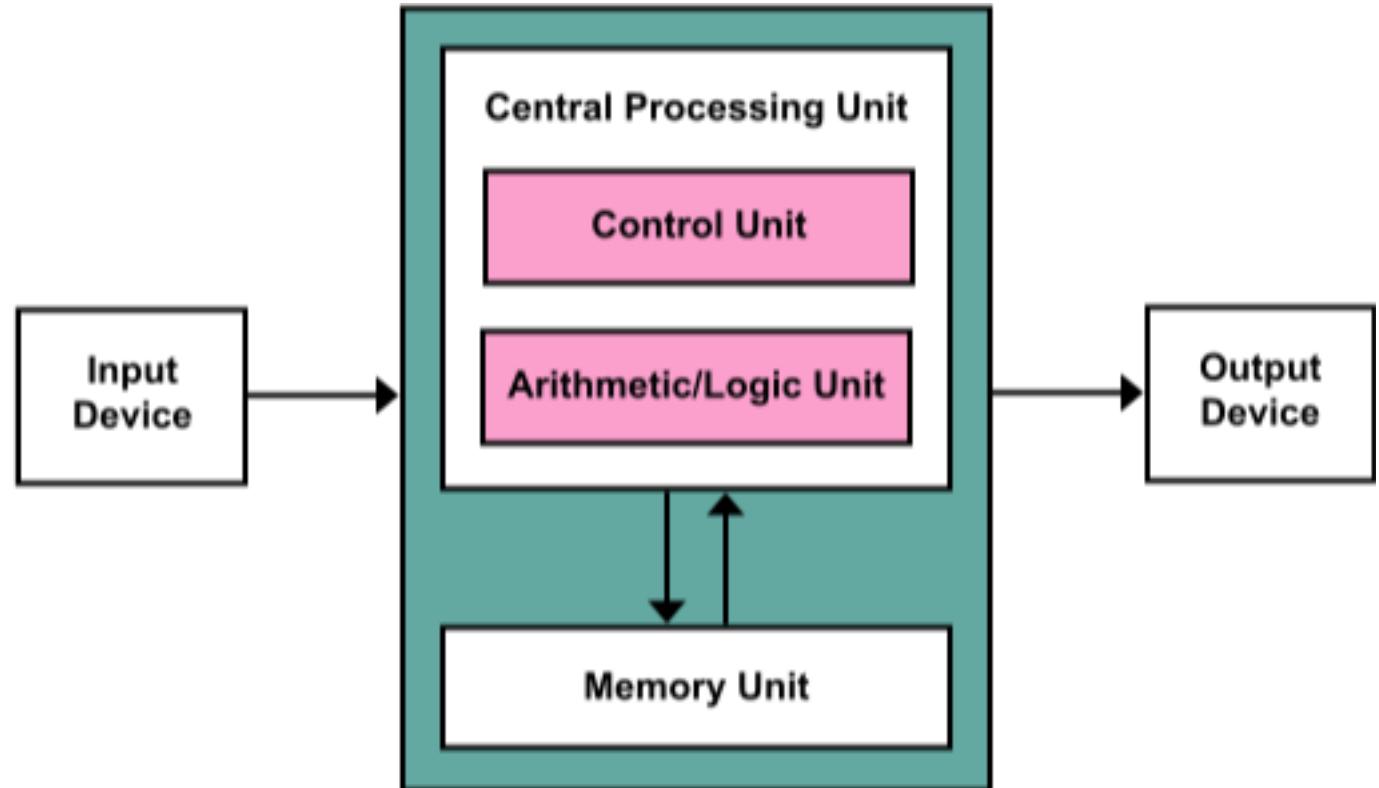
Ahmed Tamrawi

February 13, 2017

1945 Von Neumann Architecture



John von Neumann



Computer System Architecture

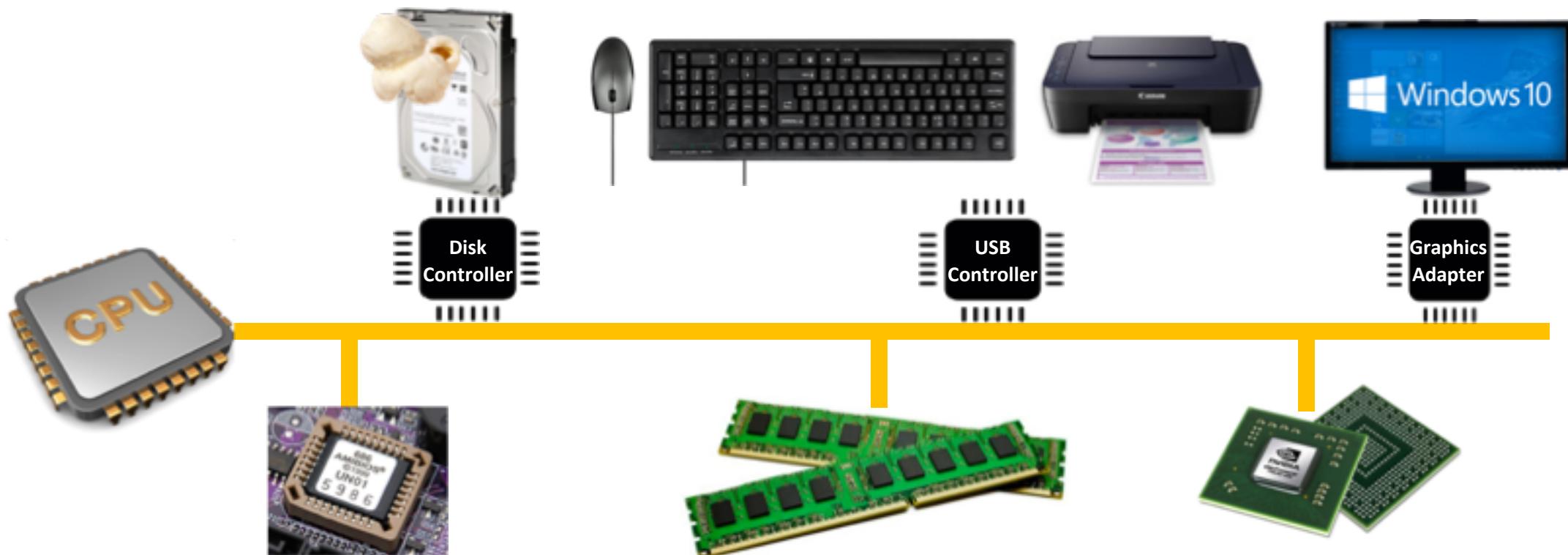
Single Processor System

Clustered System

Multiprocessor System

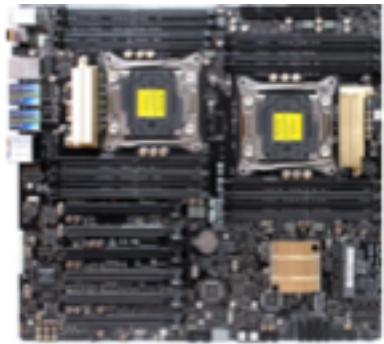
Single Processor System

One main CPU capable of executing a **general-purpose** instruction set



Multiprocessor System

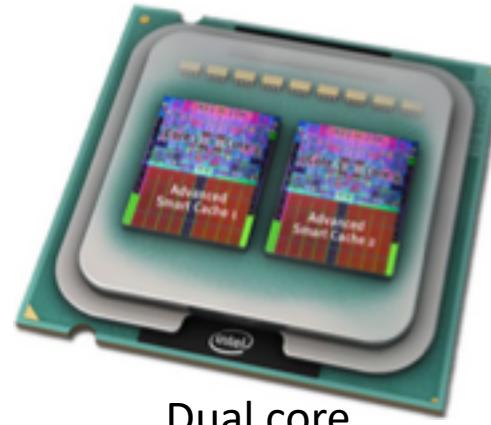
Asymmetric
Multiprocessing



Dual Processor

Two or more processors in close communication, sharing the computer bus and sometimes the clock, memory, and peripheral devices.

Symmetric
Multiprocessing



Dual core

Parallelization

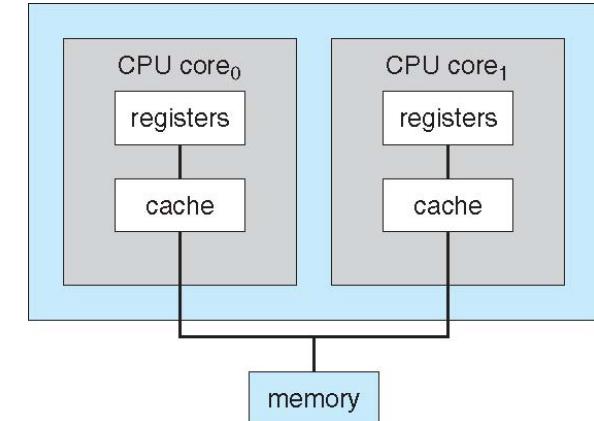
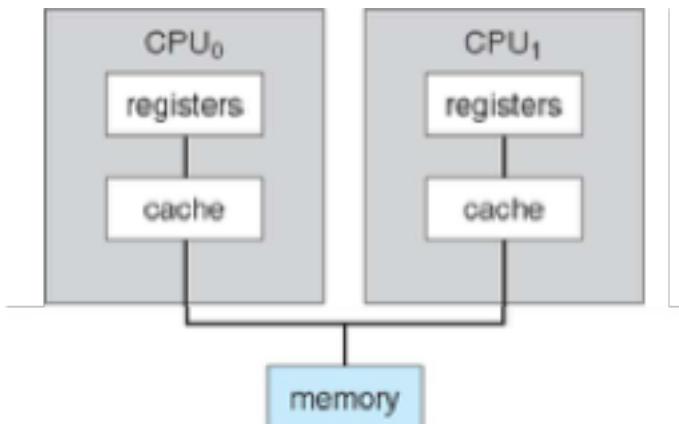
Increased Throughput

Economy of Scale

Increased Reliability

Graceful Degradation

Fault Tolerant



Multiprocessor System

Two or more processors in close communication, sharing the computer bus and sometimes the clock, memory, and peripheral devices.

**Asymmetric
Multiprocessing**

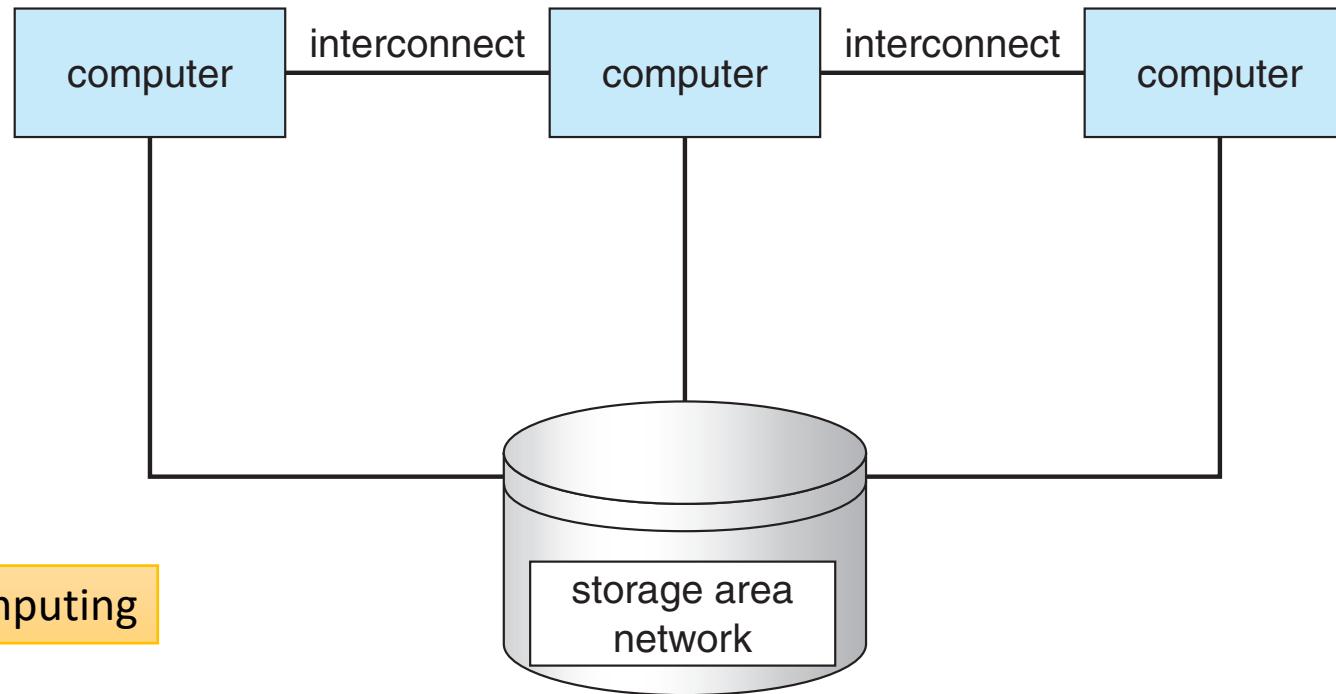
**Symmetric
Multiprocessing**

Clustered System

Multiple systems working together

Asymmetric
Clustering

Symmetric
Clustering



High-Performance Computing

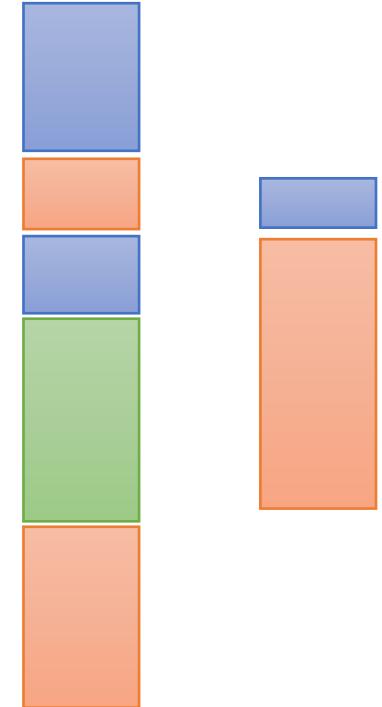
High Availability

Computer System Structure

Multiprogramming (Batch System)

Timesharing (Multitasking)

Multiprogramming (Batch System)

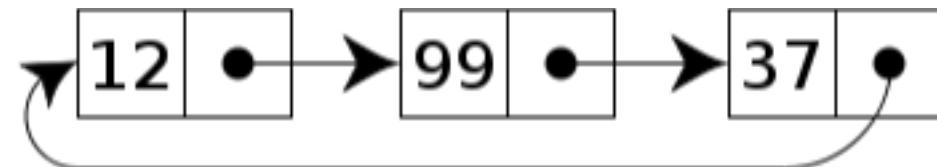
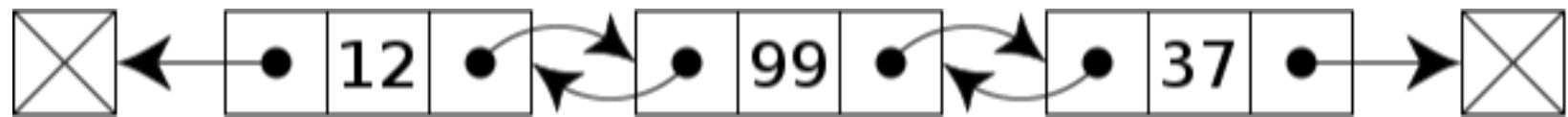
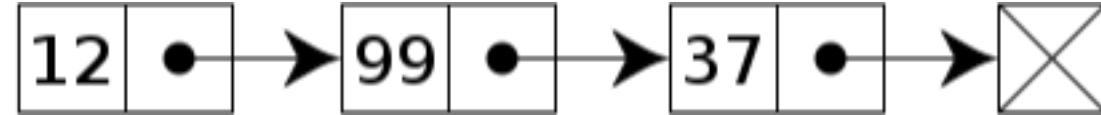


Timesharing (Multitasking)

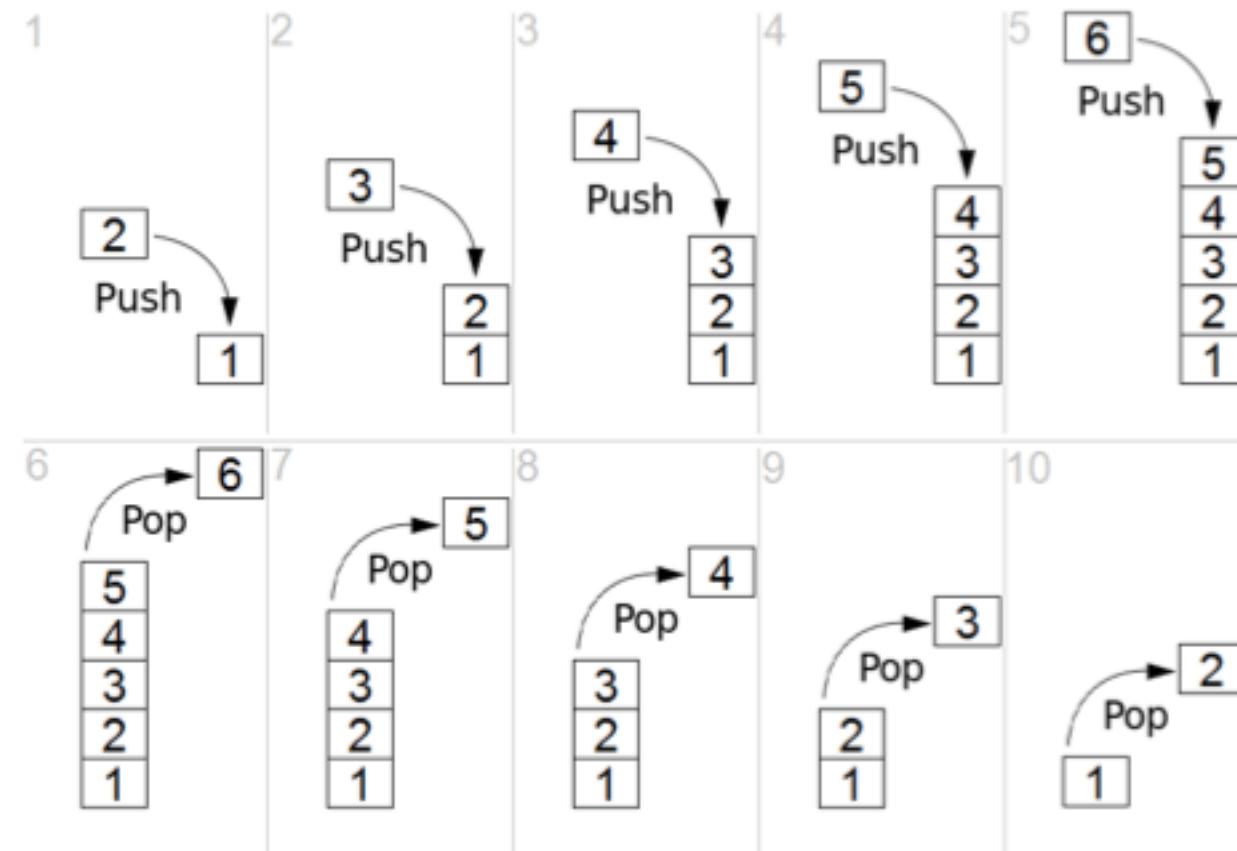


Kernel Data Structures

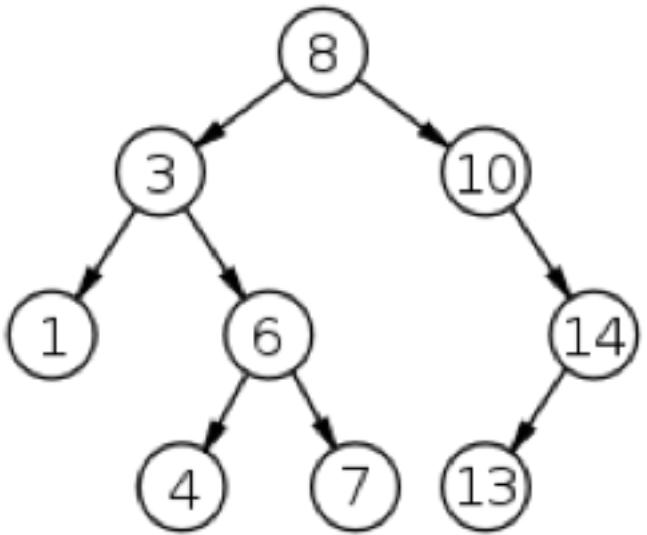
Linked List



Stack

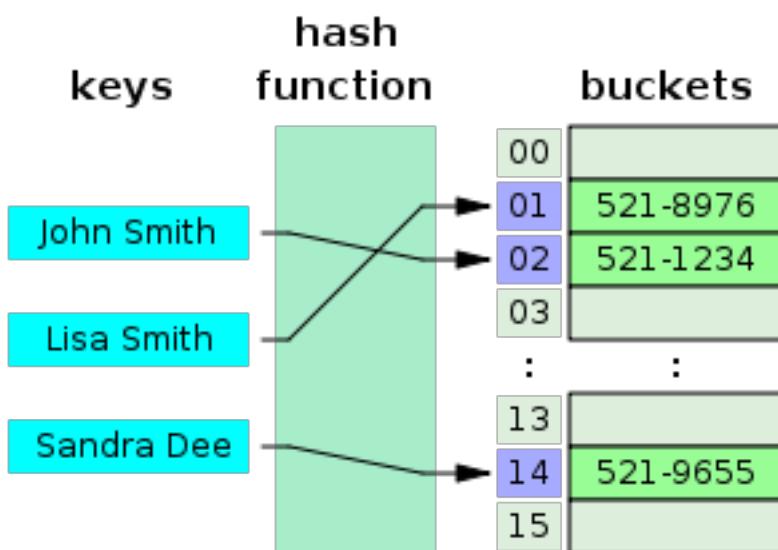


Binary Search Trees



Binary search tree		
Type	tree	
Invented	1960	
Invented by	P.F. Windley, A.D. Booth, A.J.T. Colin, and T.N. Hibbard	
Time complexity in big O notation		
Algorithm	Average	Worst Case
Space	$\Theta(n)$	$O(n)$
Search	$\Theta(\log n)$	$O(n)$
Insert	$\Theta(\log n)$	$O(n)$
Delete	$\Theta(\log n)$	$O(n)$

Hash Table



Hash table		
Type	Unordered associative array	
Invented	1953	
Time complexity in big O notation		
Algorithm	Average	Worst Case
Space	$O(n)$ ^[1]	$O(n)$
Search	$O(1)$	$O(n)$
Insert	$O(1)$	$O(n)$
Delete	$O(1)$	$O(n)$

Bitmap

A string of n binary digits representing the status of n items

10101110001110101



Linux Kernel Repository
<https://github.com/torvalds/linux>

Linux Travolds

```
diff --git a/include/linux/list.h b/include/linux/list.h
--- a/include/linux/list.h
+++ b/include/linux/list.h
@@ -1,10 +1,10 @@
 #ifndef _LINUX_LIST_H
 #define _LINUX_LIST_H
-
+/*
+ * Copyright (C) 2000 Ingo Molnar
+ * Copyright (C) 2000-2001 Linus Torvalds
+ */
+
 #include <linux/types.h>
 #include <linux/compiler.h>
 #include <linux/module.h>
 #include <linux/init.h>
 #include <linux/notifier.h>
```

Doubly Linked List Implementation

<https://github.com/torvalds/linux/blob/master/include/linux/list.h>

```
diff --git a/include/linux/kfifo.h b/include/linux/kfifo.h
--- a/include/linux/kfifo.h
+++ b/include/linux/kfifo.h
@@ -1,10 +1,10 @@
 /*
+ * Copyright (C) 2000 Ingo Molnar
+ * Copyright (C) 2000-2001 Linus Torvalds
+ */
+
 #include <linux/types.h>
 #include <linux/compiler.h>
 #include <linux/module.h>
 #include <linux/init.h>
```

FIFO Queue Implementation

<https://github.com/torvalds/linux/blob/master/include/linux/kfifo.h>

```
diff --git a/include/linux/rbtree.h b/include/linux/rbtree.h
--- a/include/linux/rbtree.h
+++ b/include/linux/rbtree.h
@@ -1,10 +1,10 @@
 /*
+ * Copyright (C) 2000 Ingo Molnar
+ * Copyright (C) 2000-2001 Linus Torvalds
+ */
+
 #include <linux/types.h>
 #include <linux/compiler.h>
 #include <linux/module.h>
 #include <linux/init.h>
```

Red Black Trees Implementation

<https://github.com/torvalds/linux/blob/master/include/linux/rbtree.h>



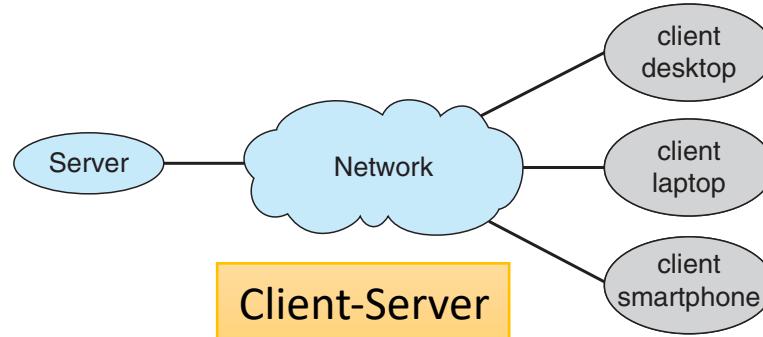
Computing Environments



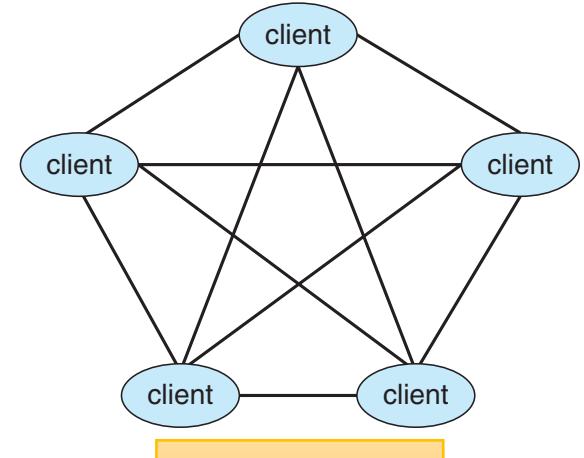
Traditional Computing



Mobile



Client-Server



Peer-to-Peer



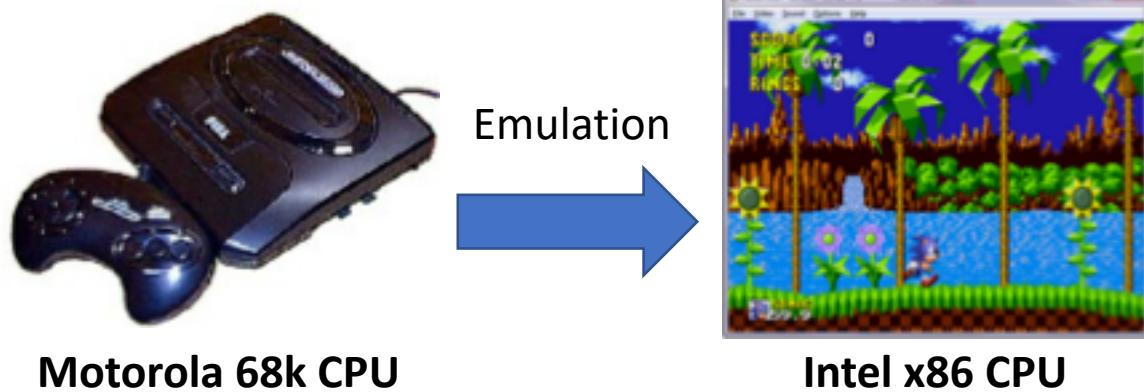
Cloud Computing



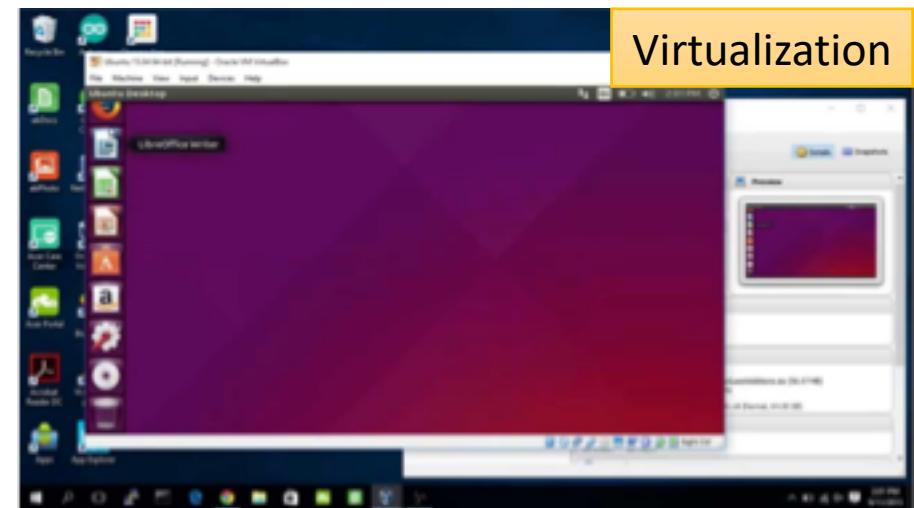
Real-Time Embedded Systems

Emulation & Virtualization

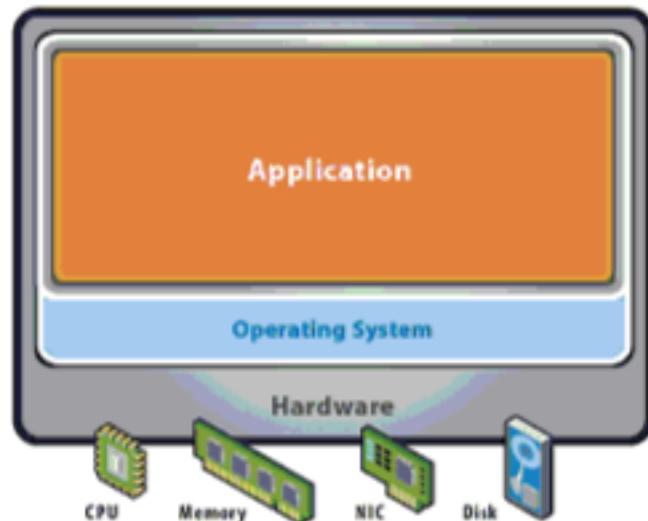
Allows operating systems to run applications within other OSes



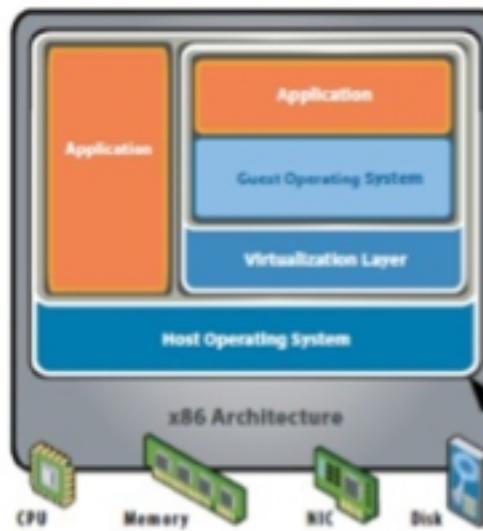
Emulation used when source CPU type different from target type



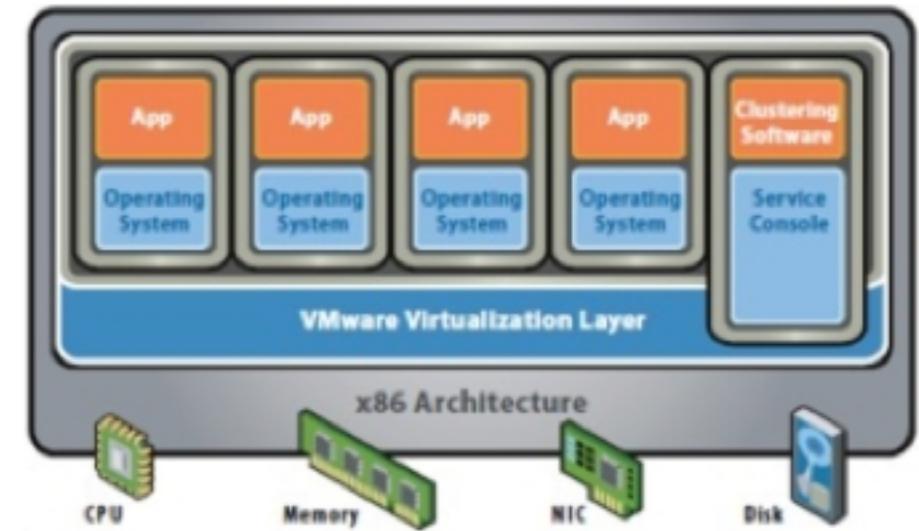
OS natively compiled for CPU, running guest OSes also natively compiled



No Virtualization



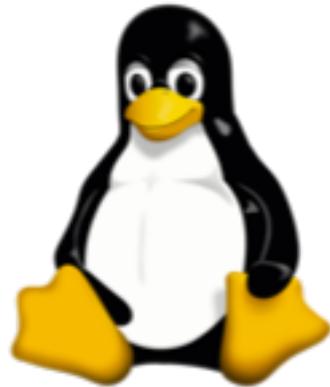
OS-Level Virtualization



Full Virtualization



open source



<https://github.com/torvalds/linux>



<http://www.apple.comopensource/>



FreeBSD®

<https://svnweb.freebsd.org/base/>

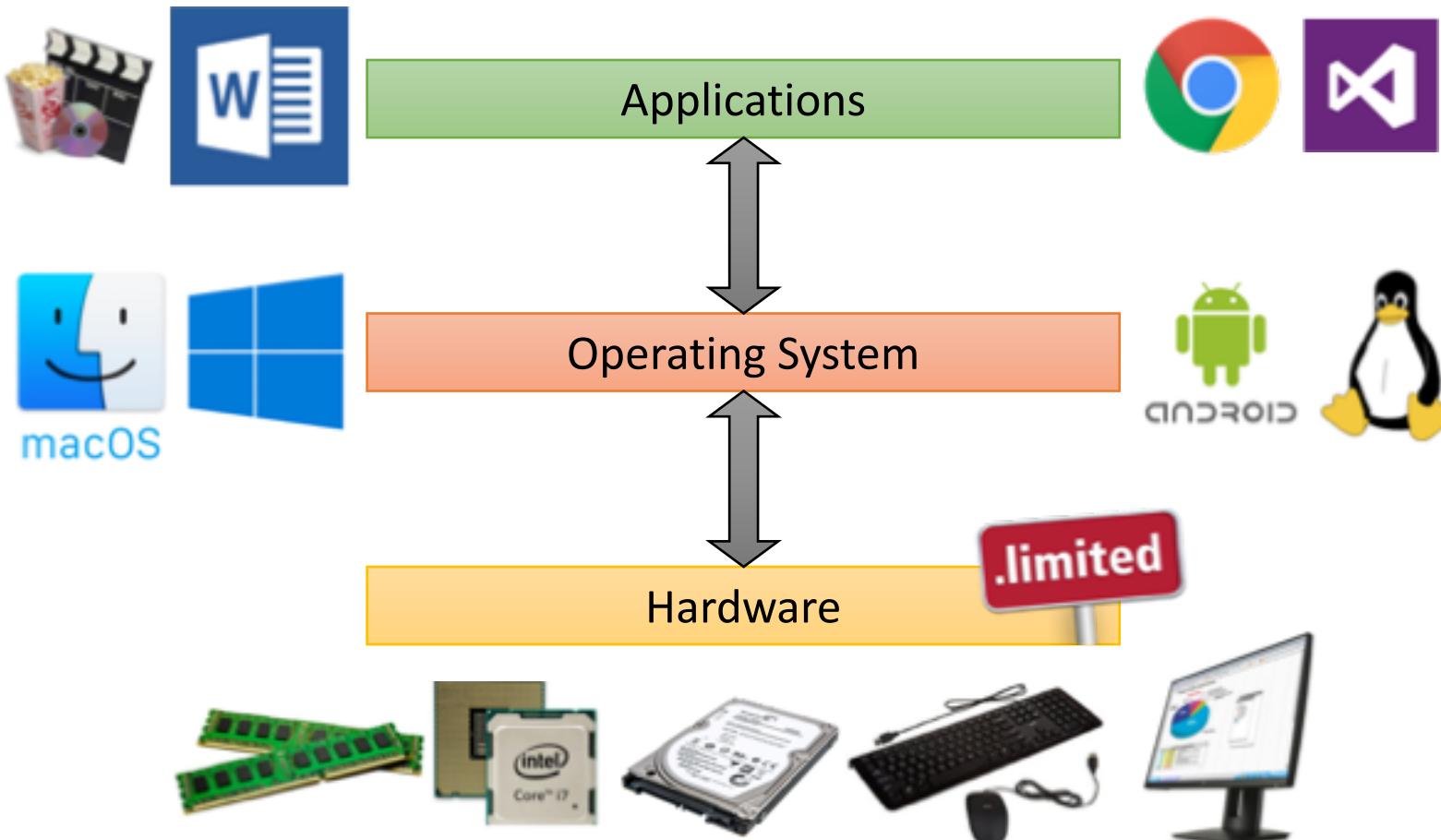


<https://github.com/android>

Realistic View of Operating System

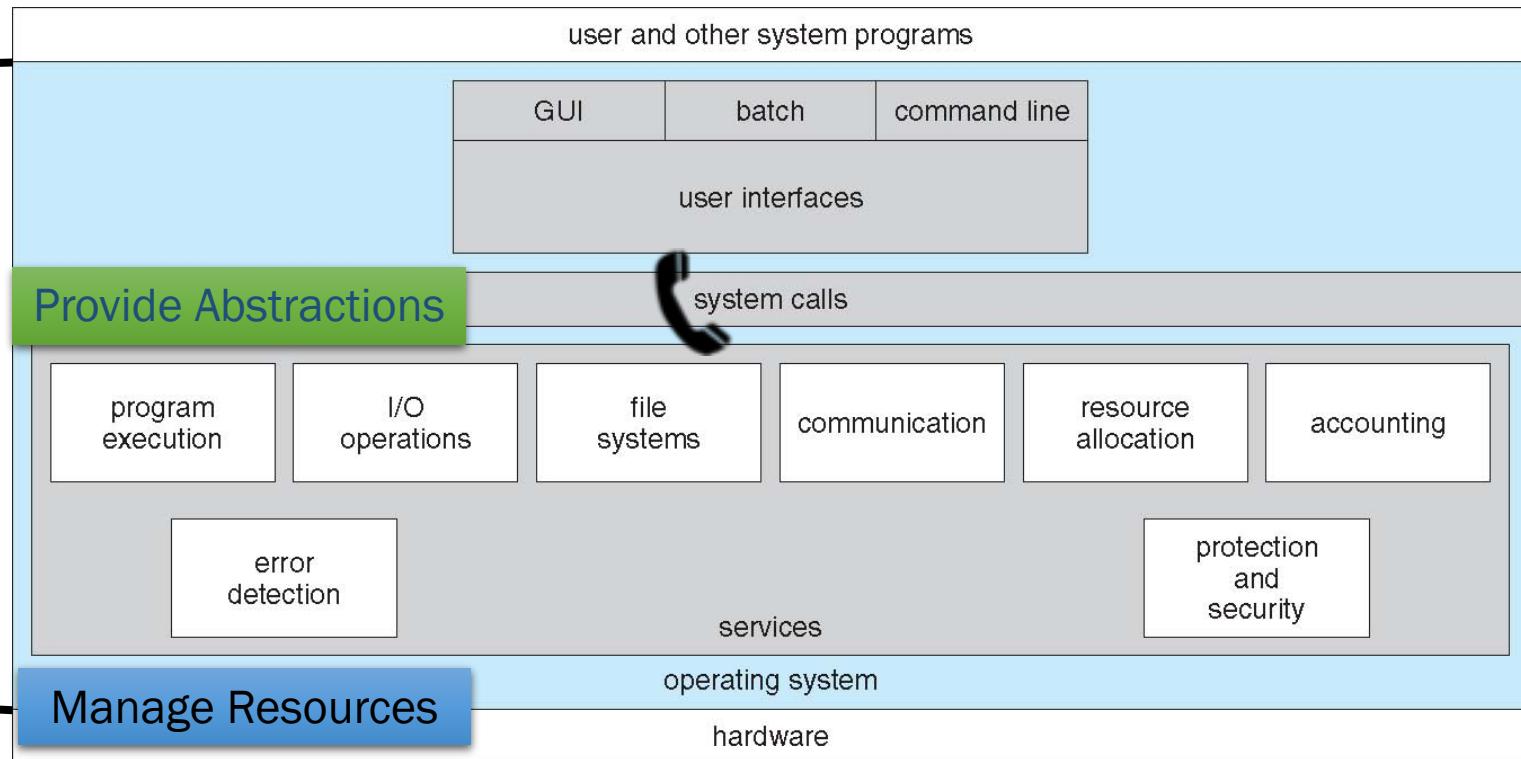
Manage Resources

Provide Abstractions



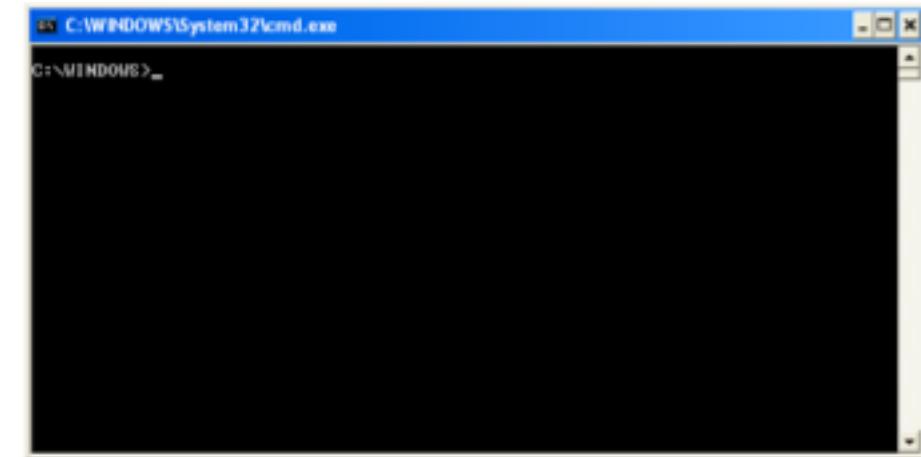
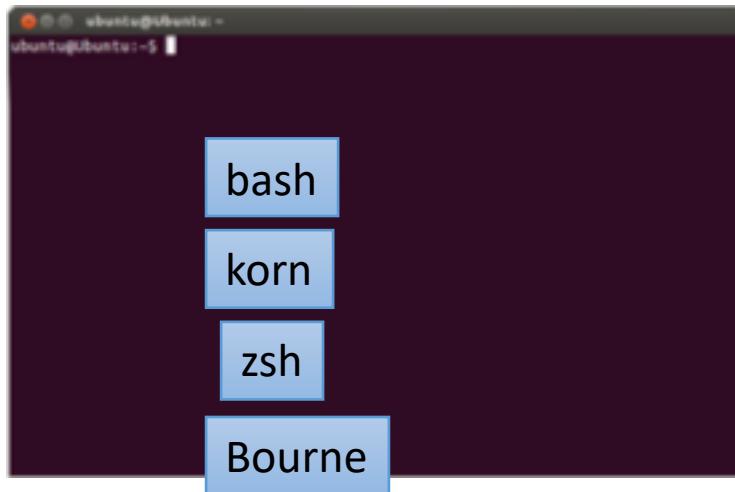
A View of Operating System Services

*Operating systems provide an environment for execution of programs
and services to programs and users*



User Operating System Interface - CLI

CLI or command interpreter allows direct command entry



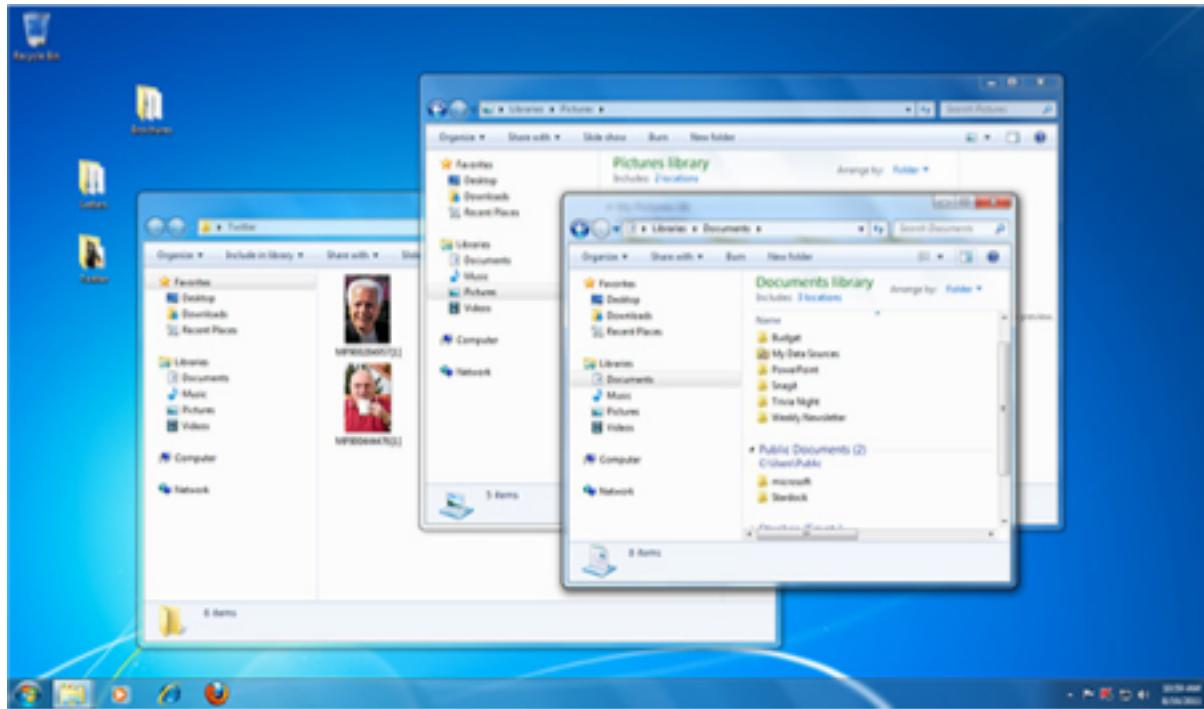
Sometimes implemented in kernel,
sometimes by systems program

Primarily fetches a command from
user and executes it

Sometimes commands built-in,
sometimes just names of programs

User Operating System Interface - GUI

User-friendly desktop metaphor interface



Many systems now include both CLI
and GUI interfaces

User Operating System Interface - Touchscreen Interfaces

Touchscreen devices require new interfaces



An operating system is interrupt driven





System Call



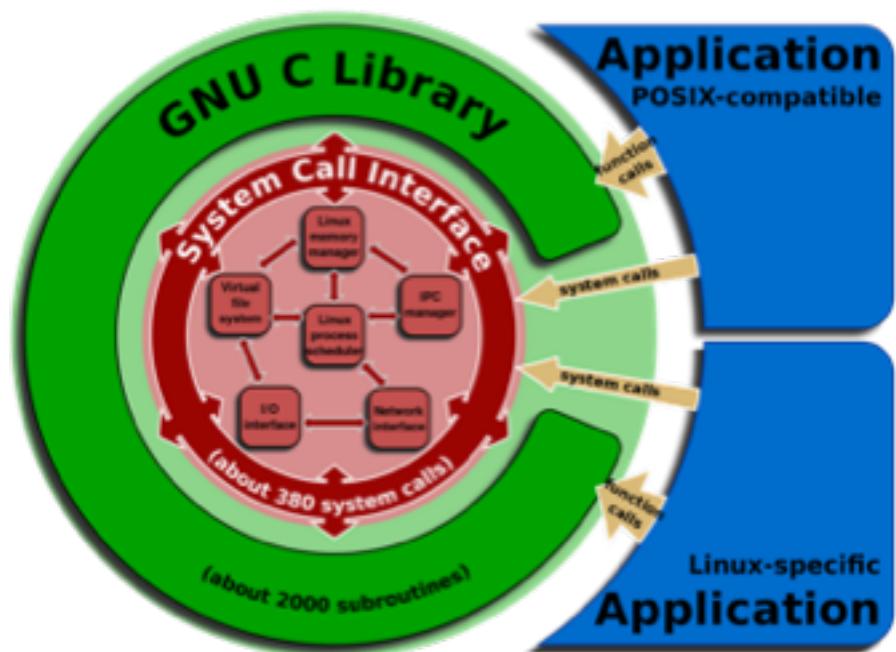


Software Interrupt (Trap)

Programming interface to the services provided by the OS

Typically written in a high-level language (C or C++)

Accessible via a high-level **Application Programming Interface (API)** rather than direct system call use





Create, Delete Communication Connection
Message Passing Model Host/Process Name
Shared-Memory Model
Transfer Status Information
Attach/Detach Remote Devices



Create/Terminate/Load/Execute Process
Get/Set Process Attributes
Wait for Time/Event
wait event, signal event
Allocate/Free/Dump Memory
Locks for Process Synchronization



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



Get/Set Time or Date
Get/Set System Data

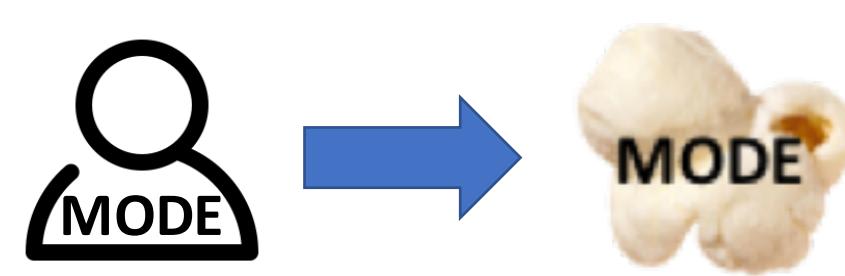
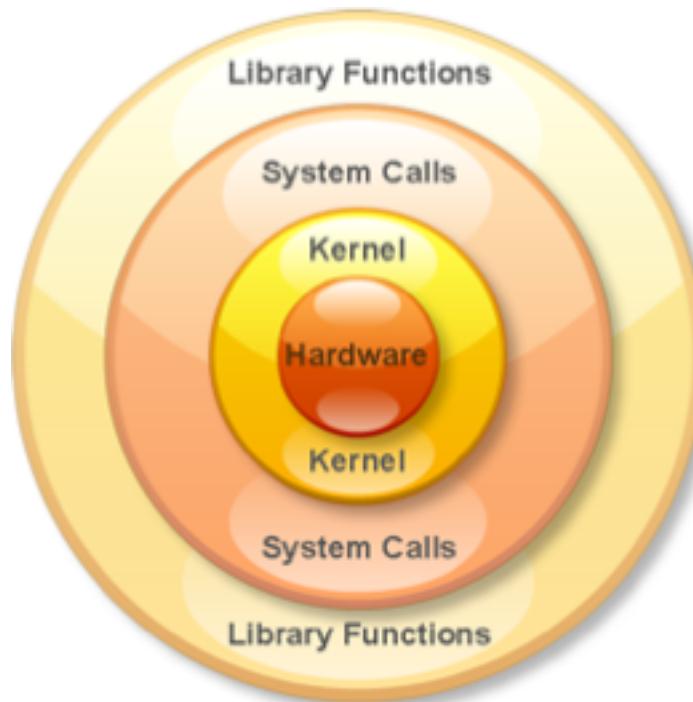


Request/Release/Read/Write Device
Get/Set Device Attributes
Logically Attach/Detach devices



Create/Delete/Open/Close/Read/Write File
Get/Set File Attributes

User processes **cannot** perform *privileged operations* themselves





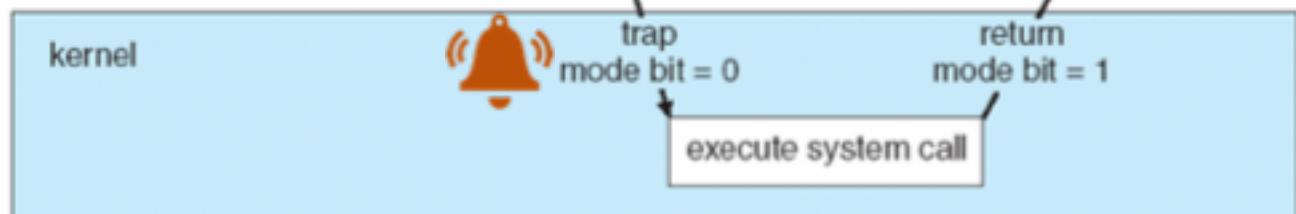
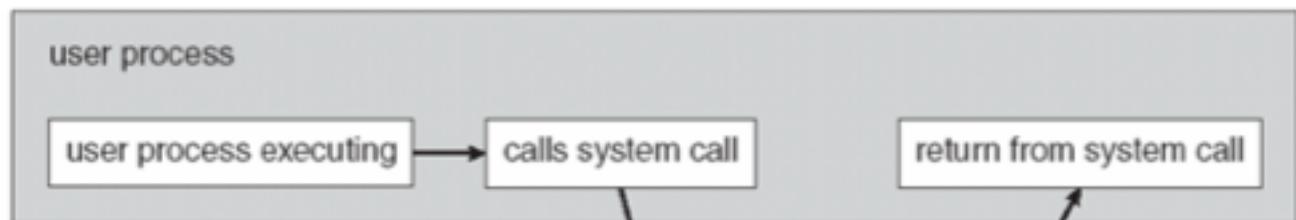
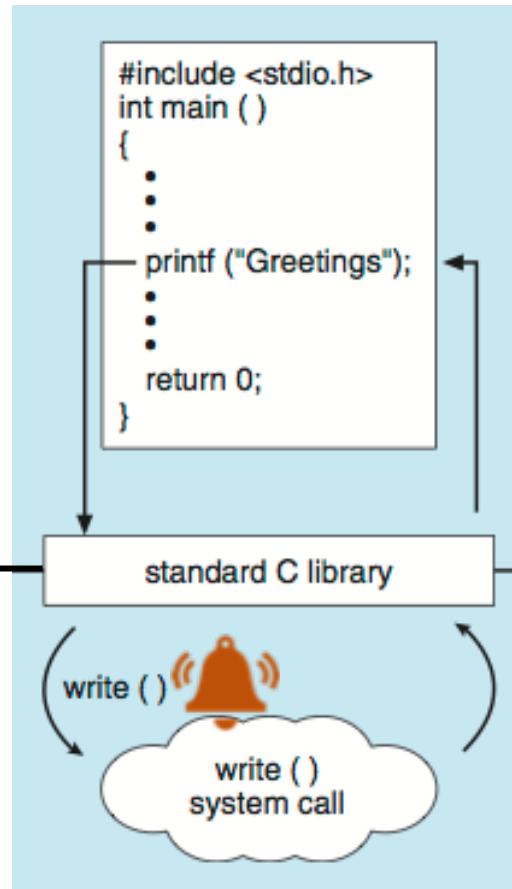
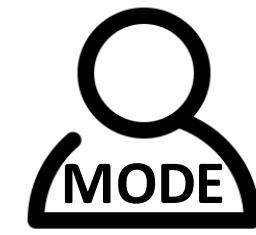
A screenshot of a terminal window titled "ls --color=tty (zsh)". The window shows the following command sequence:

```
trarium ➔ Test ls
TestFile.txt
trarium ➔ Test cp TestFile.txt TestFile-Copy.txt
trarium ➔ Test ls
TestFile-Copy.txt  TestFile.txt
trarium ➔ Test
```

```
strace cp TestFile.txt TestFile-Copy.txt
```

1. Open the input file (TestFile.txt)
2. If (TestFile.txt) does not exist, abort
3. Create the output file (TestFile-Copy.txt)
4. If (TestFile-Copy.txt) exists, abort
5. Loop Until No bytes available in TestFile.txt
 1. Read byte from (TestFile.txt)
 2. Write byte to (TestFile-Copy.txt)
6. Close (TestFile.txt)
7. Close (TestFile-Copy.txt)
8. Terminate normally.

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



How the kernel know which system call to execute?

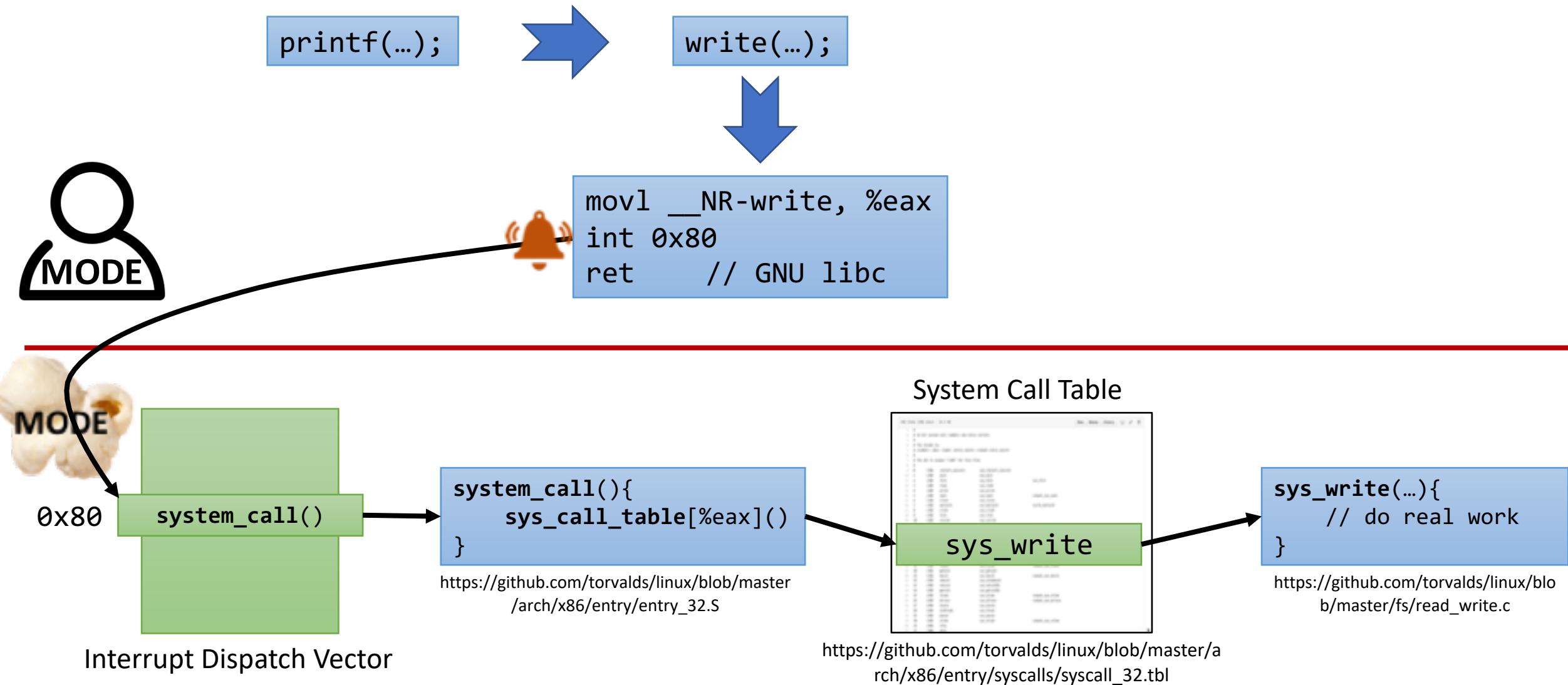
System call interface maintains a **System Call Table (Vector)** with a number corresponding to each system call

The system call interface invokes the intended system call in OS kernel and returns status of the system call and any return values

<https://github.com/torvalds/linux/blob/master/include/linux/syscalls.h>

https://github.com/torvalds/linux/blob/master/arch/x86/entry/syscalls/syscall_32.tbl

How the kernel know which system call to execute?



How Parameters are Passed?

Registers

More parameters than registers

```
printf("Hello!");
```



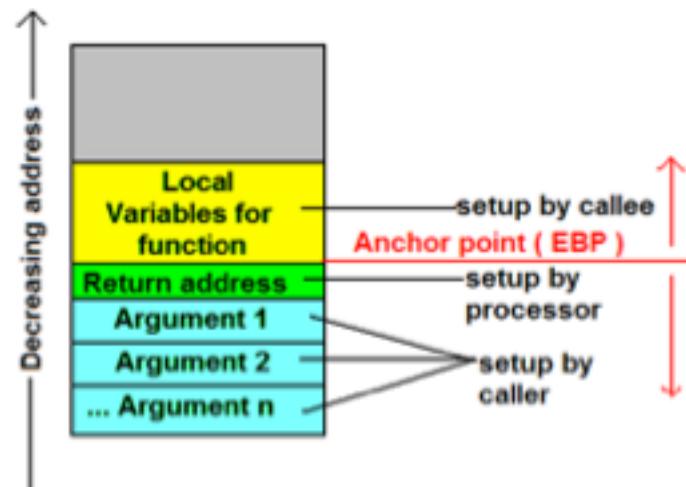
```
write(screen, "Hello!", ...);
```



```
movl "Hello", %edx  
movl __NR-write, %eax  
int 0x80  
ret // GNU libc
```

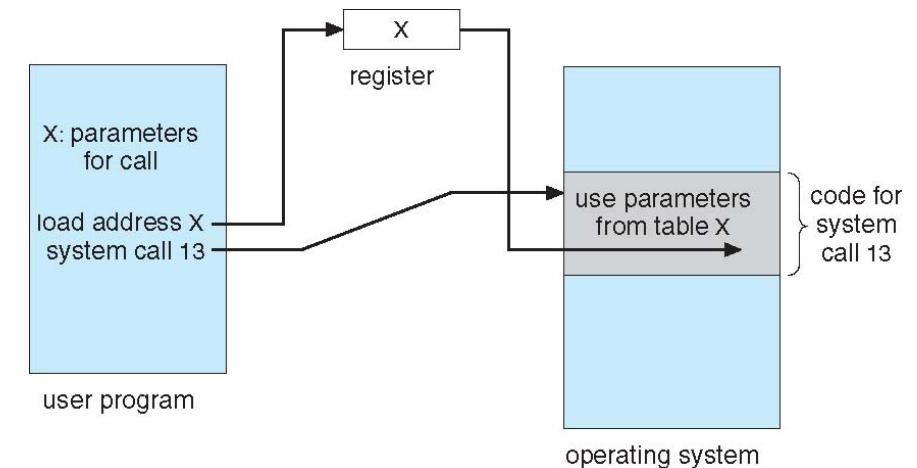
User-Mode Stack

Do not limit the number or length of parameters being passed



Block/Table in Memory

Do not limit the number or length of parameters being passed





Open System Call

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)

OPEN(2) Linux Programmer's Manual **OPEN(2)**

NAME `open`
`open, openat, creat - open and possibly create a file`

SYNOPSIS `#include <sys/types.h>`
`#include <sys/stat.h>`
`#include <fcntl.h>`

```
int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);

int creat(const char *pathname, mode_t mode);

int openat(int dirfd, const char *pathname, int flags);
int openat(int dirfd, const char *pathname, int flags, mode_t mode);
```

Feature Test Macro Requirements for glibc (see `feature_test_macros(7)`):

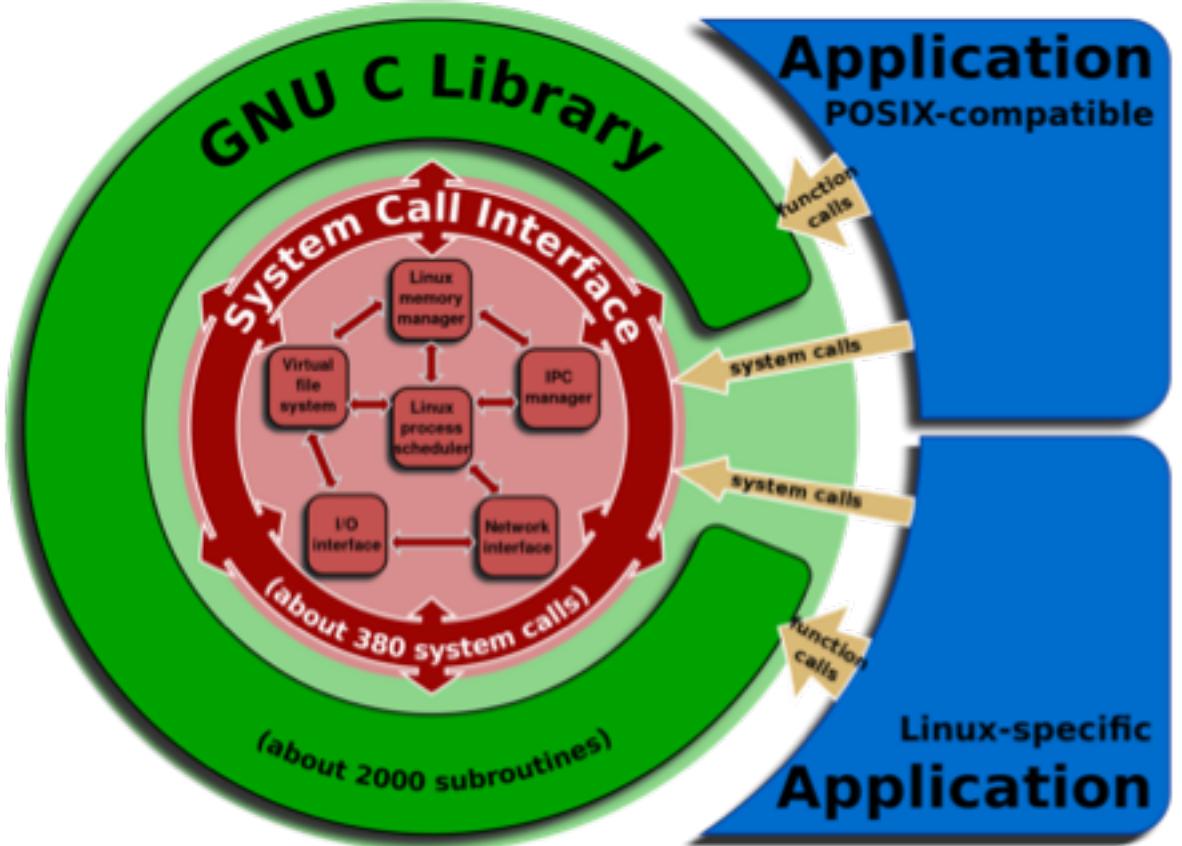
```
openat():
  Since glibc 2.10:
    _POSIX_C_SOURCE >= 200809L
  Before glibc 2.10:
    _ATFILE_SOURCE
```

DESCRIPTION `#include <sys/types.h>`
Given a `pathname` for a file, `open()` returns a file descriptor, a small, nonnegative integer for use in subsequent system calls (`read(2)`, `write(2)`, `lseek(2)`, `fctl(2)`, etc.). The file descriptor returned by a successful call will be the lowest-numbered file descriptor not currently open for the process.

By default, the new file descriptor is set to remain open across an `execve(2)` (i.e., the `FD_CLOEXEC` file descriptor flag described in `fcntl(2)` is initially disabled); the `O_CLOEXEC` flag, described below, can be used to change this default. The file offset is set to the beginning of the file (see `lseek(2)`).

A call to `open()` creates a new `open file description`, an entry in the system-wide table of open files. The open file description records the file offset and the file status flags (see below). A file descriptor is a reference to an open file description; this reference

<http://man7.org/linux/man-pages/man2/open.2.html>



```
#include <unistd.h>
#include <fcntl.h>

int main() {
    int filedesc = open("testfile.txt", O_WRONLY | O_APPEND);
    if(filedesc < 0){
        return 1;
    }
    return 0;
}
```

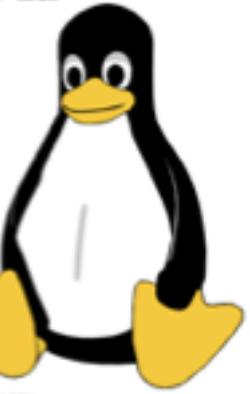
With GNU C Library (Glibc)

```
#include <unistd.h>
#include <fcntl.h>

#define __NR_open 5
_syscall3(long, open, const char *, filename, int, flags, int, mode)

int main() {
    int filedesc = open("testfile.txt", O_WRONLY | O_APPEND);
    if(filedesc < 0){
        return 1;
    }
    return 0;
}
```

Without GNU C Library (Glibc)



About 300 System Calls

<https://github.com/torvalds/linux/blob/master/include/linux/syscalls.h>



POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X)

Windows X86-64 System Call Table (XP/2003/Vista/2008/7/2012/8/10)



Author: Matous 'jiles' Jurek (jiles@cs.tuuk.hlu)
From: Ypsilon

See also Windows XMM System Call Table: <http://j0fka.weebly.com/>

Special thanks to Michael

Layout by Microsoft Word

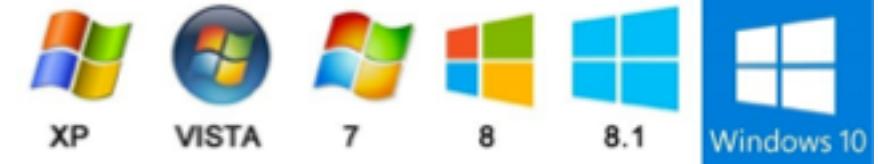
Enter the symbol to highlight (see b):

Highlight
Show all Hide all

System Call Number	Windows XP (Labeled)	Windows Server 2003 (Labeled)	Windows Vista (Labeled)	Windows Server 2008 (Labeled)	Windows 7 (Labeled)	Windows Server 2012 (Labeled)	Windows 8 (Labeled)	Windows 10 (Labeled)
0x00000000								
0x00000001								
0x00000002								
0x00000003								
0x00000004								
0x00000005								
0x00000006								
0x00000007								
0x00000008								
0x00000009								
0x0000000A								
0x0000000B								
0x0000000C								
0x0000000D								
0x0000000E								
0x0000000F								
0x00000010								
0x00000011								
0x00000012								
0x00000013								
0x00000014								
0x00000015								
0x00000016								
0x00000017								
0x00000018								
0x00000019								
0x0000001A								
0x0000001B								
0x0000001C								
0x0000001D								
0x0000001E								
0x0000001F								
0x00000020								
0x00000021								
0x00000022								
0x00000023								
0x00000024								
0x00000025								
0x00000026								
0x00000027								
0x00000028								
0x00000029								
0x0000002A								
0x0000002B								
0x0000002C								
0x0000002D								
0x0000002E								
0x0000002F								
0x00000030								
0x00000031								
0x00000032								
0x00000033								
0x00000034								
0x00000035								
0x00000036								
0x00000037								
0x00000038								
0x00000039								
0x0000003A								
0x0000003B								
0x0000003C								
0x0000003D								
0x0000003E								
0x0000003F								
0x00000040								
0x00000041								
0x00000042								
0x00000043								
0x00000044								
0x00000045								
0x00000046								
0x00000047								
0x00000048								
0x00000049								
0x0000004A								
0x0000004B								
0x0000004C								
0x0000004D								
0x0000004E								
0x0000004F								
0x00000050								
0x00000051								
0x00000052								
0x00000053								
0x00000054								
0x00000055								
0x00000056								
0x00000057								
0x00000058								
0x00000059								
0x0000005A								
0x0000005B								
0x0000005C								
0x0000005D								
0x0000005E								
0x0000005F								
0x00000060								
0x00000061								
0x00000062								
0x00000063								
0x00000064								
0x00000065								
0x00000066								
0x00000067								
0x00000068								
0x00000069								
0x0000006A								
0x0000006B								
0x0000006C								
0x0000006D								
0x0000006E								
0x0000006F								
0x00000070								
0x00000071								
0x00000072								
0x00000073								
0x00000074								
0x00000075								
0x00000076								
0x00000077								
0x00000078								
0x00000079								
0x0000007A								
0x0000007B								
0x0000007C								
0x0000007D								
0x0000007E								
0x0000007F								
0x00000080								
0x00000081								
0x00000082								
0x00000083								
0x00000084								
0x00000085								
0x00000086								
0x00000087								
0x00000088								
0x00000089								
0x0000008A								
0x0000008B								
0x0000008C								
0x0000008D								
0x0000008E								
0x0000008F								
0x00000090								
0x00000091								
0x00000092								
0x00000093								
0x00000094								
0x00000095								
0x00000096								
0x00000097								
0x00000098								
0x00000099								
0x0000009A								
0x0000009B								
0x0000009C								
0x0000009D								
0x0000009E								
0x0000009F								
0x000000A0								
0x000000A1								
0x000000A2								
0x000000A3								
0x000000A4								
0x000000A5								
0x000000A6								
0x000000A7								
0x000000A8								
0x000000A9								
0x000000AA								
0x000000AB								
0x000000AC								
0x000000AD								
0x000000AE								
0x000000AF								
0x000000B0								
0x000000B1								
0x000000B2								
0x000000B3								
0x000000B4								
0x000000B5								
0x000000B6								
0x000000B7								
0x000000B8								
0x000000B9								
0x000000BA								
0x000000BB								
0x000000BC								
0x000000BD								
0x000000BE								
0x000000BF								
0x000000C0								
0x000000C1								
0x000000C2								
0x000000C3								
0x000000C4								
0x000000C5								
0x000000C6								
0x000000C7								
0x000000C8								
0x000000C9								
0x000000CA								
0x000000CB								
0x000000CC								
0x000000CD								
0x000000CE								
0x000000CF								
0x000000D0								
0x000000D1								
0x000000D2								
0x000000D3								
0x000000D4								
0x000000D5								
0x000000D6								
0x000000D7								
0x000000D8								
0x000000D9								
0x000000DA								
0x000000DB								
0x000000DC								
0x000000DD								
0x000000DE								
0x000000DF								
0x000000E0								
0x000000E1								
0x000000E2								
0x000000E3								
0x000000E4								
0x000000E5								
0x000000E6								
0x000000E7								
0x000000E8								
0x000000E9								
0x000000EA					</td			

About 400 System Calls

http://j00ru.vexillium.org/ntapi_64/



Win32 API for Windows

BONUS

Add a System Call to Linux that prints:
“I am awesome!”

References:

- https://www.youtube.com/watch?v=5rr_VoQC0gE
- <http://franksthinktank.com/howto/addsyscall/>
- <https://tssurya.wordpress.com/2014/08/19/adding-a-hello-world-system-call-to-linux-kernel-3-16-0/>



Create, Delete Communication Connection
Message Passing Model Host/Process Name
Shared-Memory Model
Transfer Status Information
Attach/Detach Remote Devices



Create/Terminate/Load/Execute Process
Get/Set Process Attributes
Wait for Time/Event
wait event, signal event
Allocate/Free/Dump Memory
Locks for Process Synchronization



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



Create/Delete/Open/Close/Read/Write File
Get/Set File Attributes



Get/Set Time or Date
Get/Set System Data



Request/Release/Read/Write Device
Get/Set Device Attributes
Logically Attach/Detach devices

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() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

System Programs (Utilities)

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 Manipulation

Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories

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

Background Services (Daemons)

*Launch at boot time
Some for system startup, then terminate
Some from system boot to shutdown
Disk checking, process scheduling, error logging
Run in user context not kernel context*

Status Information

System Info, Hardware Status, Registry

Programming Language Support

Compilers, assemblers, debuggers and interpreters sometimes provided

Application Programs

*Run by users
Not typically considered part of OS
Launched by command line, mouse click, finger poke*

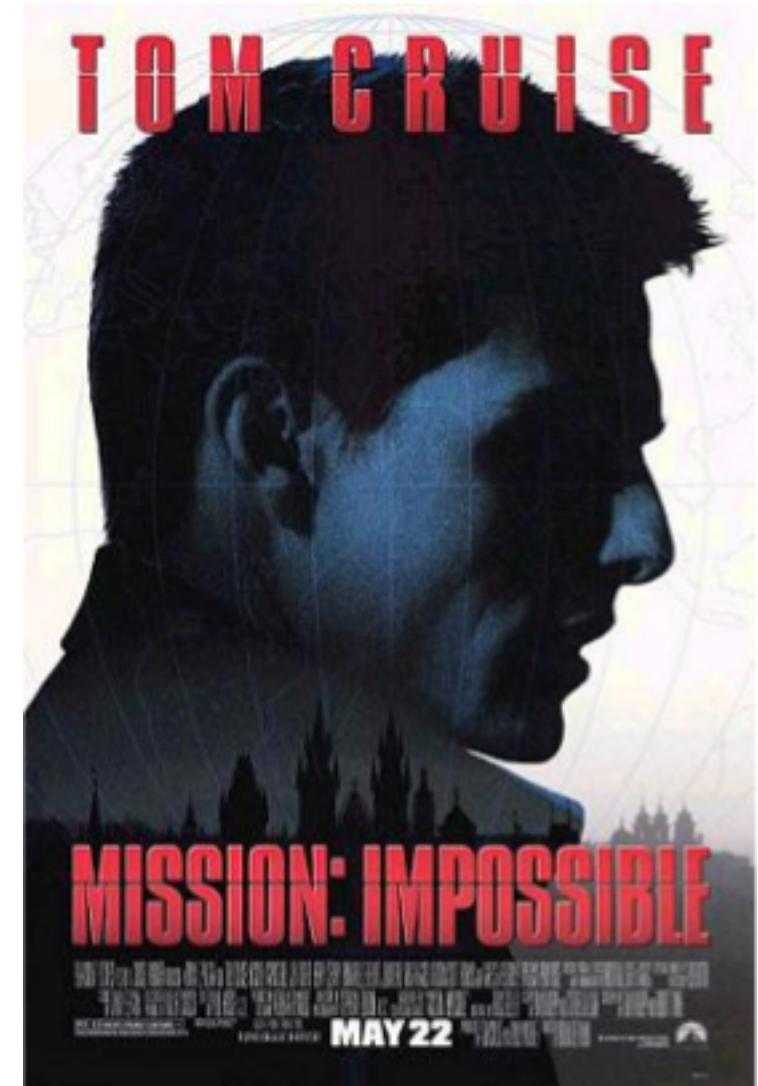
Operating System Design and Implementation

There is no perfect OS, but some have proven to be successful

Define the User/System Goals

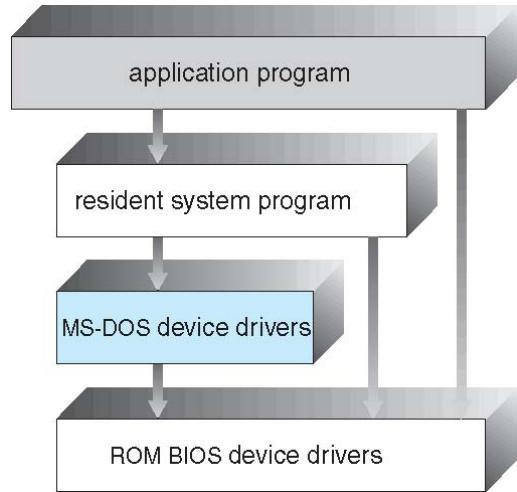
Define the Policies (**What** will be done?) and Mechanisms (**How** to do it?)

Internal structure of different Operating Systems can vary widely; Affected by choice of hardware, type of system

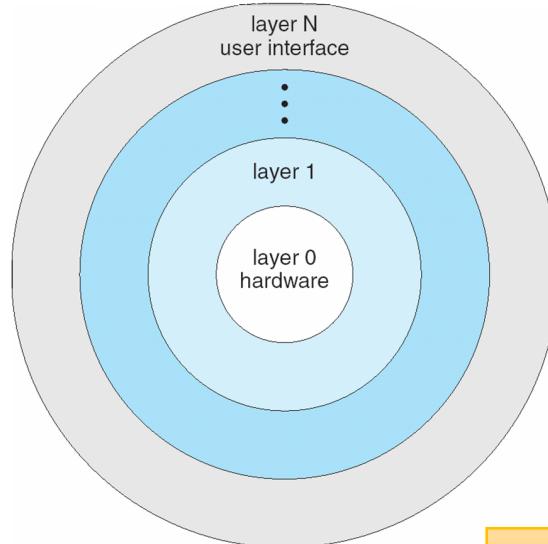


Operating System Structure

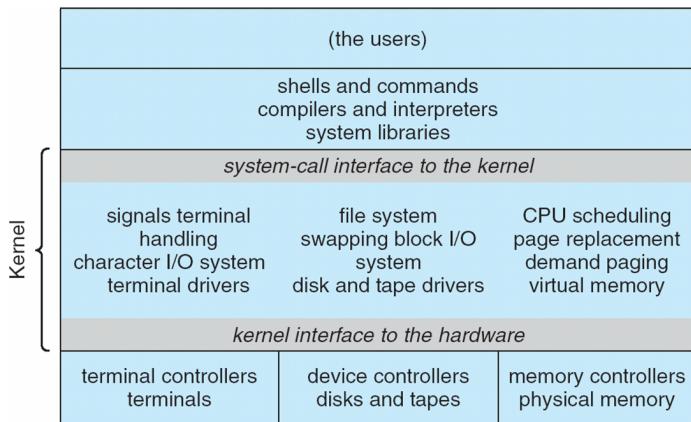
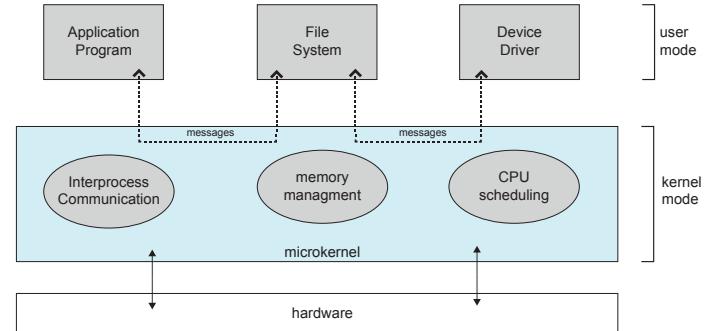
Simple Structure



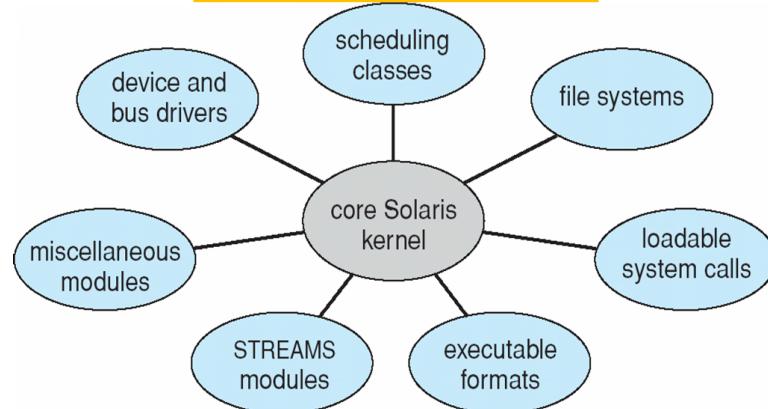
Layered Structure



Microkernel Structure



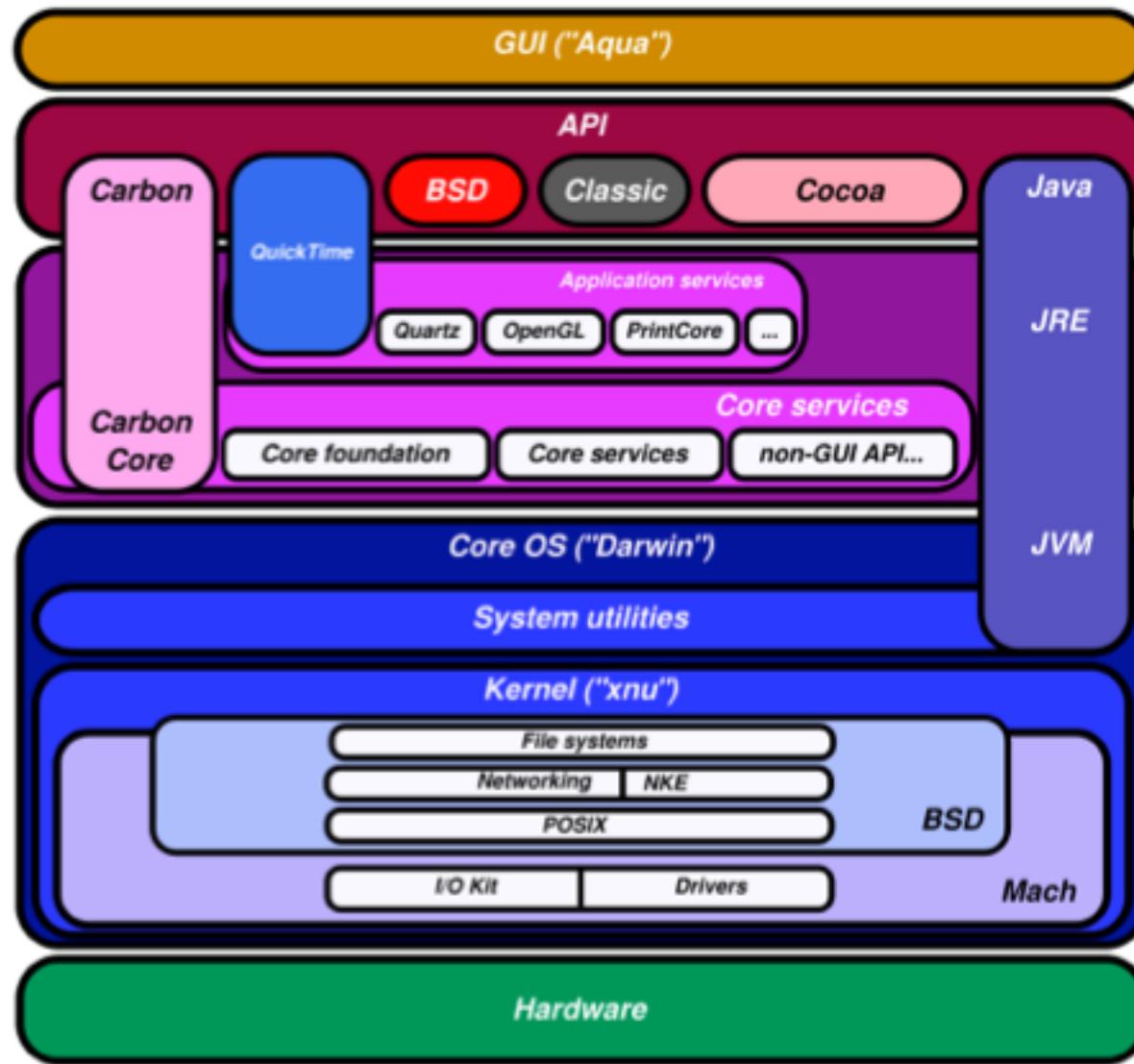
Modular Structure



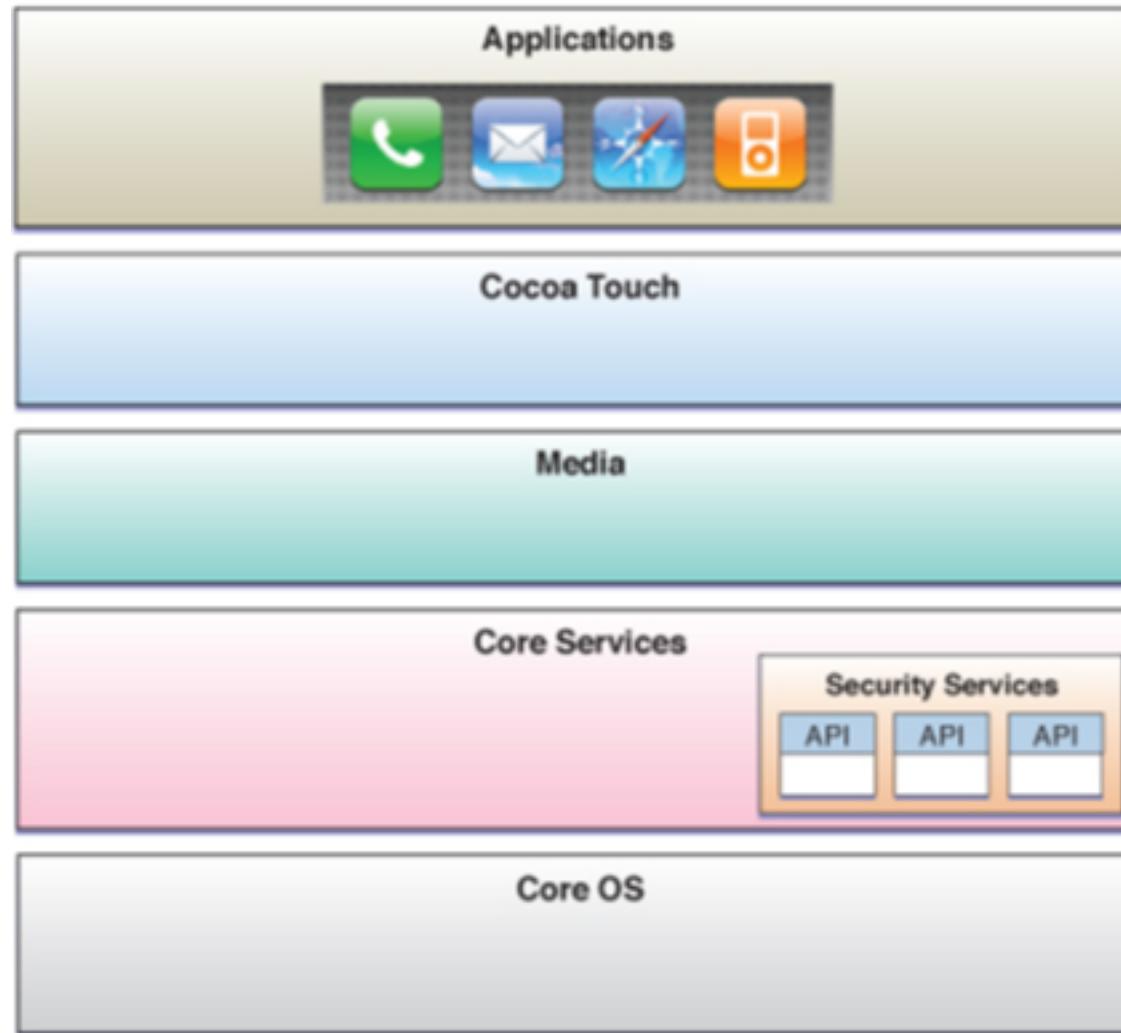
Hybrid Structure



Mac OS Structure



iOS Structure



Android Structure



Operating System Debugging

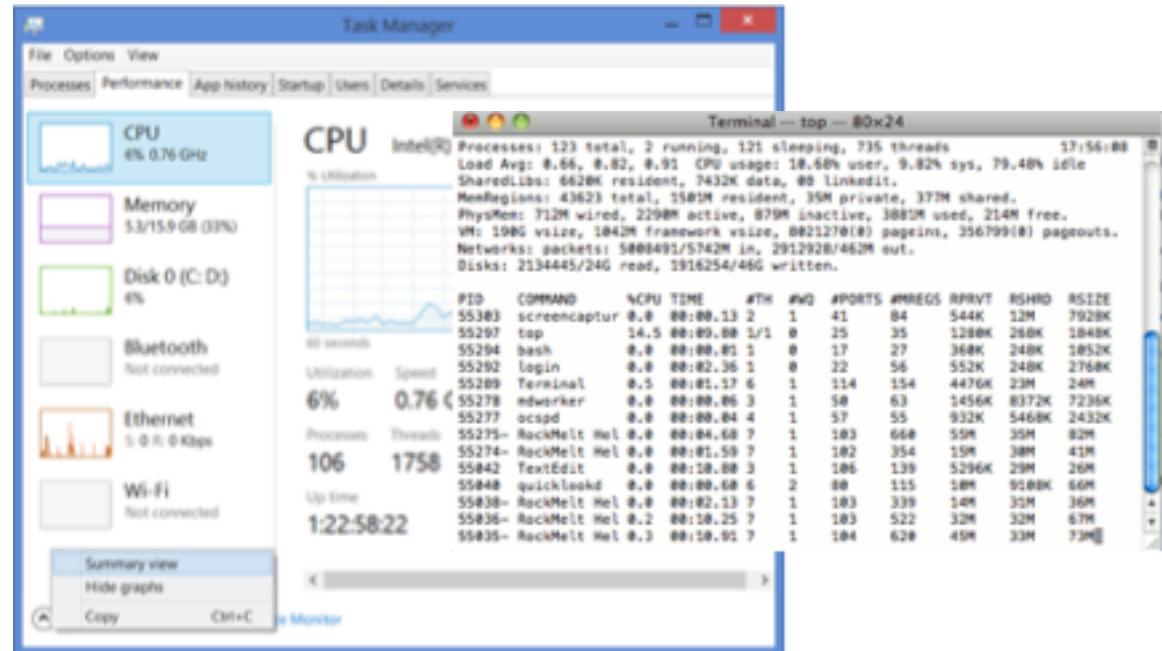
Kernighan's Law: "Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."

OS generates *log* files containing error information

Application failure can generate **core dump** file capturing memory of the process

OS failure can generate **crash dump** file containing kernel memory

OS must provide means of computing and displaying measures of system behavior



Operating System Generation

Operating systems are designed to run on any of a class of machines; the system must be configured by obtaining information concerning the specific configuration of the hardware system

Branch: master + linux / arch /		
		Create new file Upload files Find file History
	torvalds mm: optimize PageWaiters bit use for unlock_page() ↗	Latest commit b6fe138 5 days ago
...		
	alpha clocksource: Use a plain u64 instead of cycle_t	7 days ago
	arc Merge tag 'arc-4.10-rc1-part2' of git://git.kernel.org/pub/scm/linux/...	9 days ago
	arm Merge branch 'timers-urgent-for-linus' of git://git.kernel.org/pub/sc...	7 days ago
	arm64: don't pull uaccess.h into ".S"	6 days ago
	avr32 clocksource: Use a plain u64 instead of cycle_t	7 days ago
	blackfin Merge branch 'timers-urgent-for-linus' of git://git.kernel.org/pub/sc...	7 days ago
	c6x clocksource: Use a plain u64 instead of cycle_t	7 days ago
	cris Replace <asm/uaccess.h> with <linux/uaccess.h> globally	8 days ago
	frv Replace <asm/uaccess.h> with <linux/uaccess.h> globally	8 days ago
	h8300 Replace <asm/uaccess.h> with <linux/uaccess.h> globally	8 days ago
	hexagon clocksource: Use a plain u64 instead of cycle_t	7 days ago
	ia64 clocksource: Use a plain u64 instead of cycle_t	7 days ago
	m32r Replace <asm/uaccess.h> with <linux/uaccess.h> globally	8 days ago
	m68k clocksource: Use a plain u64 instead of cycle_t	7 days ago
	metag Merge branch 'smo-urgent-for-linus' of git://git.kernel.org/pub/scm/...	7 days ago
	microblaze clocksource: Use a plain u64 instead of cycle_t	7 days ago
	mips Merge branch 'timers-urgent-for-linus' of git://git.kernel.org/pub/sc...	7 days ago
	mn10300 clocksource: Use a plain u64 instead of cycle_t	7 days ago
	nios2 clocksource: Use a plain u64 instead of cycle_t	7 days ago
	openrisc clocksource: Use a plain u64 instead of cycle_t	7 days ago
	perfc clocksource: Use a plain u64 instead of cycle_t	7 days ago
	powerpc powerpc: Fix build warning on 32-bit PPC	7 days ago

<https://github.com/torvalds/linux/tree/master/arch>



つづく