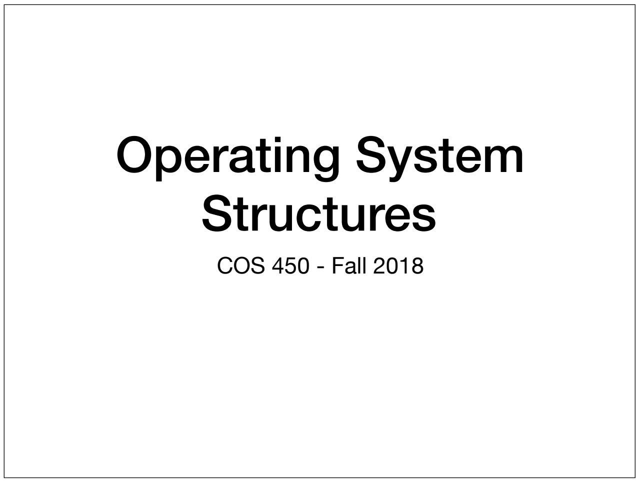
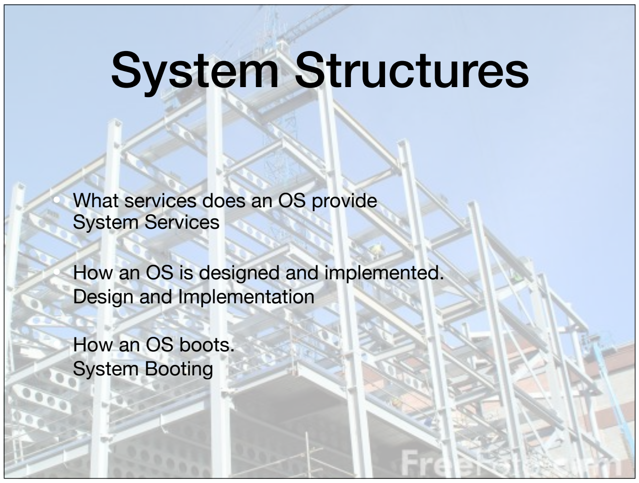


1

Operating System Structures

COS 450 - Fall 2018



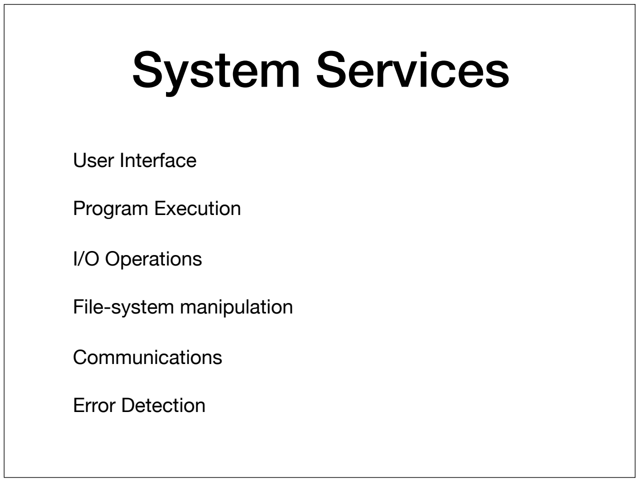
2

System Structures

What services does an OS provide
System Services

How an OS is designed and implemented.
Design and Implementation

How an OS boots.
System Booting



3

System Services

User Interface

Program Execution

I/O Operations

File-system manipulation

Communications

Error Detection

4

System Services

User Interface Services:

- Command line (shell)
- Graphical environment (windows)
- Remote controls

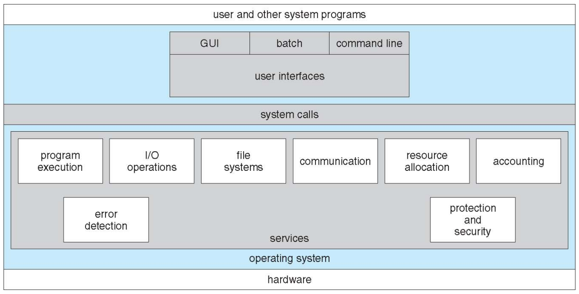
5

System Services

System Calls (API):

- Interface between **user** and **system**
- Mechanism used to access the hardware
- POSIX, Win32, Java API, ...

6



System Calls

7

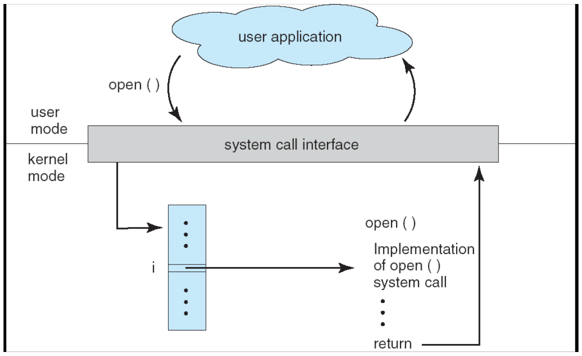
System Call Implementation

Typically a **numbered table**

recall the **Interrupt Vector Table**

Trigger a swap from **user** to **protected** mode.

8



System Call in Action

9

System Call Parameters

Three general methods to pass parameters

in ***registers***.

in ***memory***

in the ***stack***.

10

Pintos System Call

	0xbffffe80	‘A’	…
	0xbffffe7c	2	third argument
	0xbffffe78	256	second argument
	0xbffffe74	42	first argument
Stack Pointer	0xbffffe70	6	system call #
	0xbffffe6c	…	…

do_this(42, 0xbffffe80, 2);

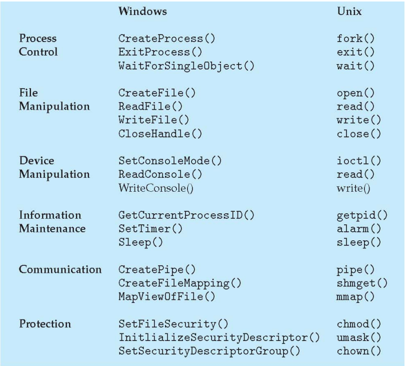
11

System Call Types

Process Control	File Manipulation
Device Manipulation	Information Maintenance
Communications	Protection

The System Call APIs consist of…

12



	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()

System Call Types

Pintos System Calls

```
     SYS_HALT,       /* Halt the operating system. */
       SYS_EXIT,       /* Terminate this process. */
        SYS_EXEC,       /* Start another process. */
 SYS_WAIT,       /* Wait for a child process to die. */
            SYS_CREATE,     /* Create a file. */
            SYS_REMOVE,     /* Delete a file. */
             SYS_OPEN,       /* Open a file. */
        SYS_FILESIZE,   /* Obtain a file's size. */
           SYS_READ,       /* Read from a file. */
            SYS_WRITE,      /* Write to a file. */
     SYS_SEEK,       /* Change position in a file. */
SYS_TELL,       /* Report current position in a file. */
            SYS_CLOSE,      /* Close a file. */
```

13

System Programs

More aptly named **System Utilities**

command interpreter.

file and device manipulation.

status and communication tools.

compiler and debugger?

14

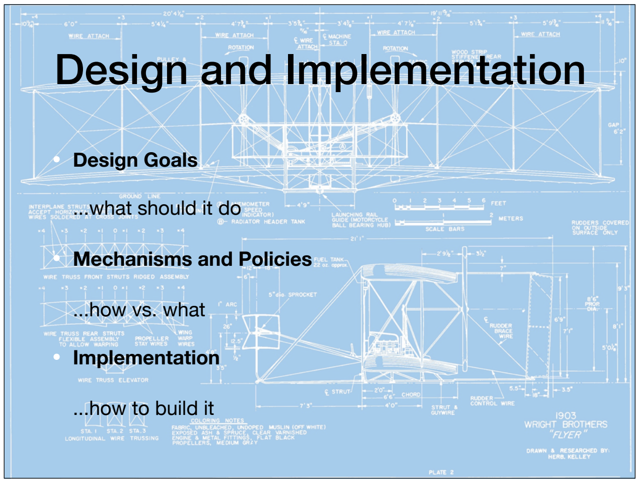
System Structures

System Services

Design and Implementation

System Booting

15

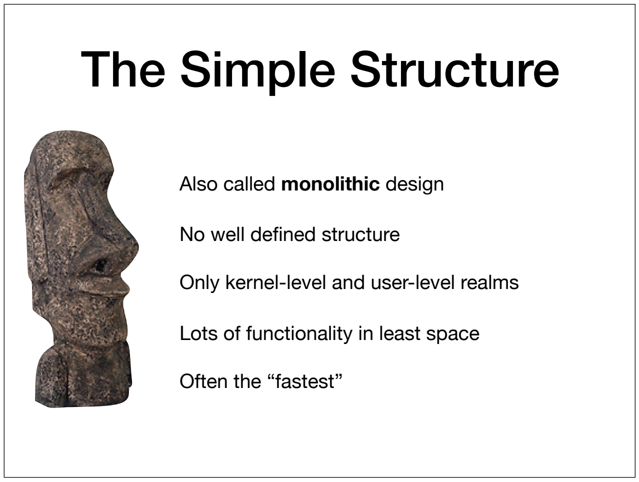


16

Design and Implementation

- **Design Goals**
  - ...what should it do
- **Mechanisms and Policies**
  - ...how vs. what
- **Implementation**
  - ...how to build it

1903
WRIGHT BROTHERS
"FLYER"



17

The Simple Structure

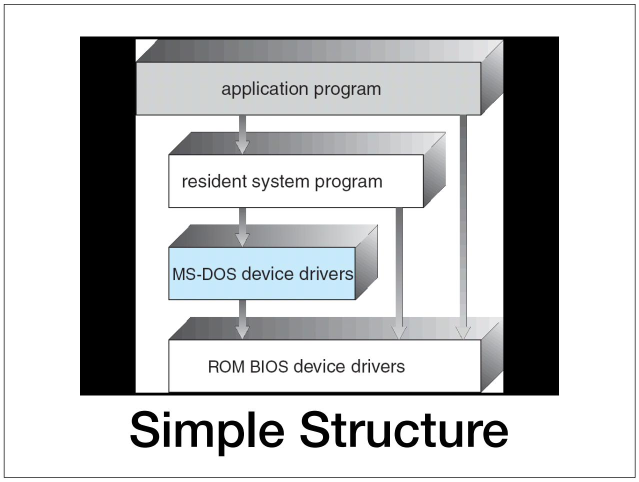
Also called **monolithic** design

No well defined structure

Only kernel-level and user-level realms

Lots of functionality in least space

Often the "fastest"



18

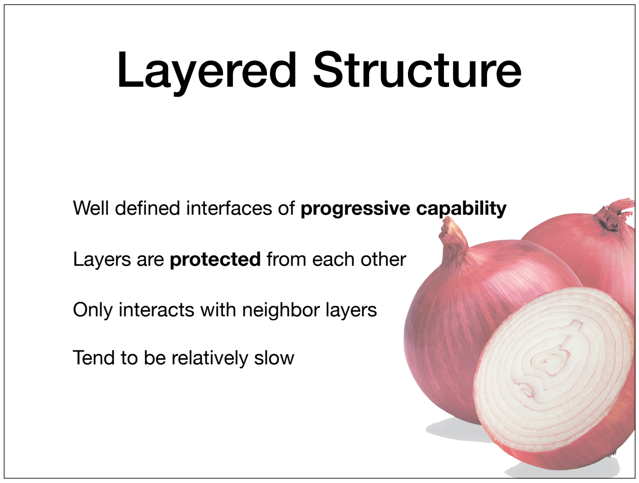
application program

resident system program

MS-DOS device drivers

ROM BIOS device drivers

Simple Structure



Layered Structure

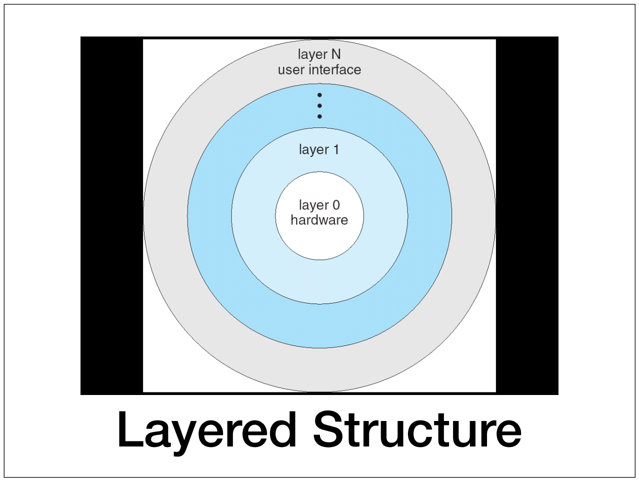
Well defined interfaces of **progressive capability**

Layers are **protected** from each other

Only interacts with neighbor layers

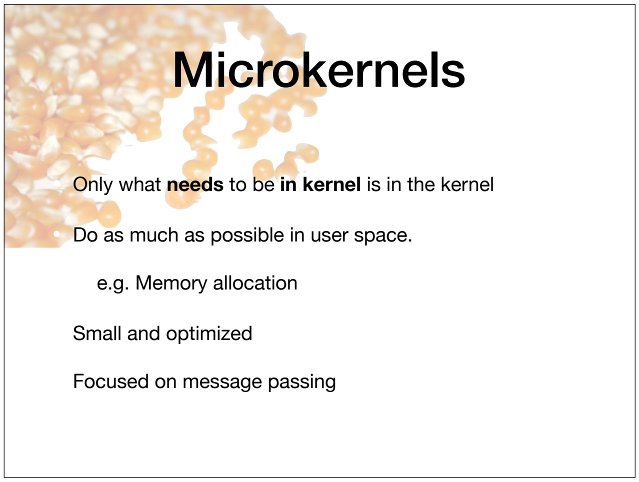
Tend to be relatively slow

19



Layered Structure

20



Microkernels

Only what **needs** to be **in kernel** is in the kernel

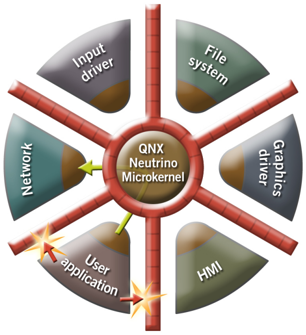
Do as much as possible in user space.

e.g. Memory allocation

Small and optimized

Focused on message passing

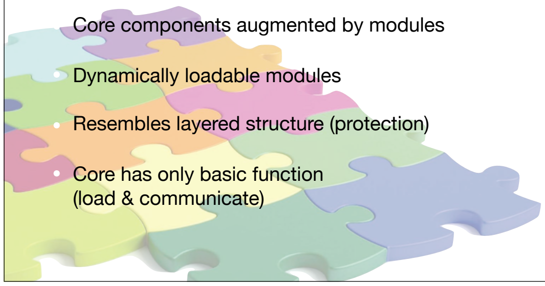
21



Microkernel

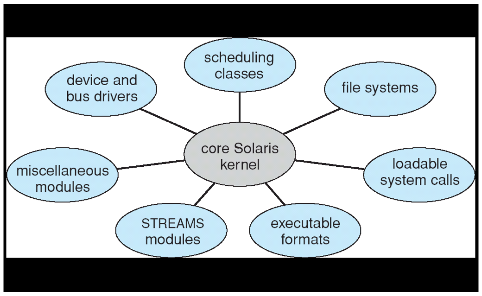
22

Modular Structure



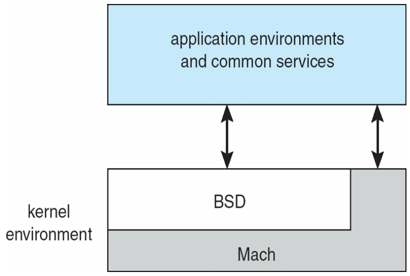
- Object-oriented techniques
- Core components augmented by modules
- Dynamically loadable modules
- Resembles layered structure (protection)
- Core has only basic function (load & communicate)

23



Modular Structure

24



application environments and common services

kernel environment

BSD

Mach

Mac OS X Hybrid

25

Virtual Machines

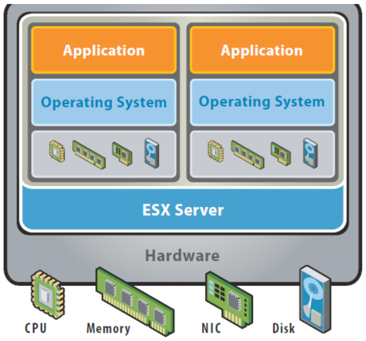
Continuation of Layered Structure

Virtualize the **hardware**

Creates the illusion of hardware that an operating system can run on

Wicked Awesome!

26



Application

Application

Operating System

Operating System

ESX Server

Hardware

CPU

Memory

NIC

Disk

Virtual Machine

27

28

Simulation

Emulate a different architecture

Must emulate all hardware

Simulate guest instructions

Slow.

29

Implementation

Must provide a *complete* machine.

Only one ***kernel*** mode.
virtual user and *virtual kernel* modes

AMD provides hardware support for this.

VMWare, QEMU, bochs, VirtualBox, ...

30

System Structures

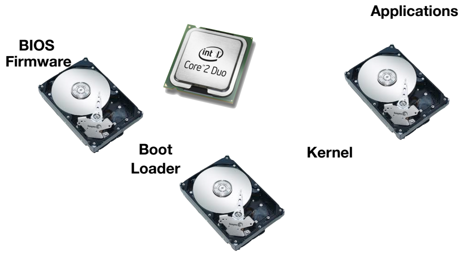
System Services

Design and Implementation

System Booting

31

System Boot



Applications

BIOS Firmware

Boot Loader

Kernel

32

System Structures

System Services

Design and Implementation

System Booting

33

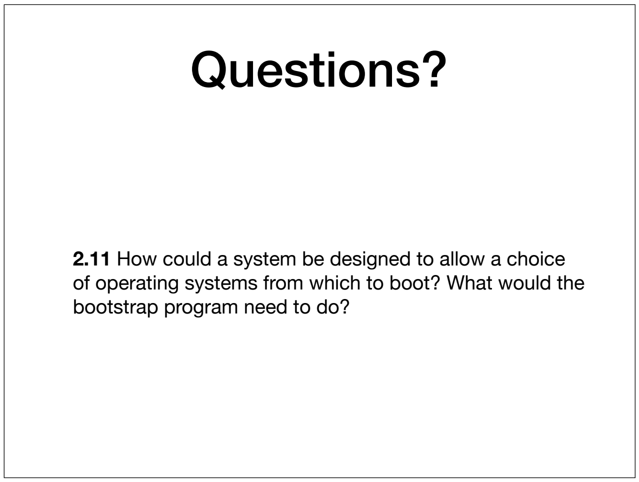
Summary

Services Provided

- Through System Call Interface

Designs and Implementation

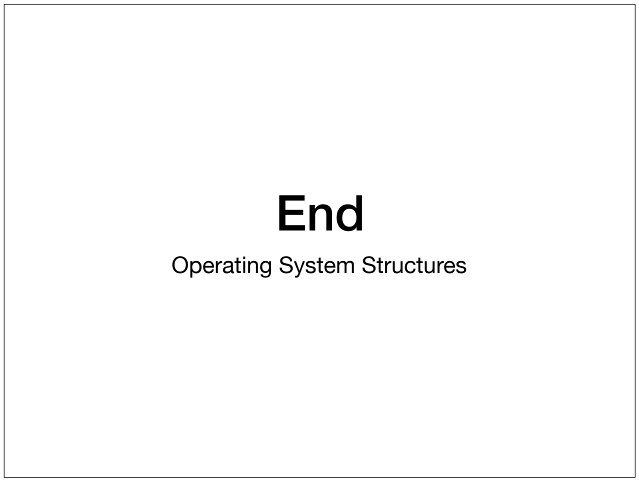
- Simple, Layered, Microkernel, Modular
- Virtual Machines, Simulation, Para-virtualization.



34

Questions?

2.11 How could a system be designed to allow a choice of operating systems from which to boot? What would the bootstrap program need to do?



35

End

Operating System Structures