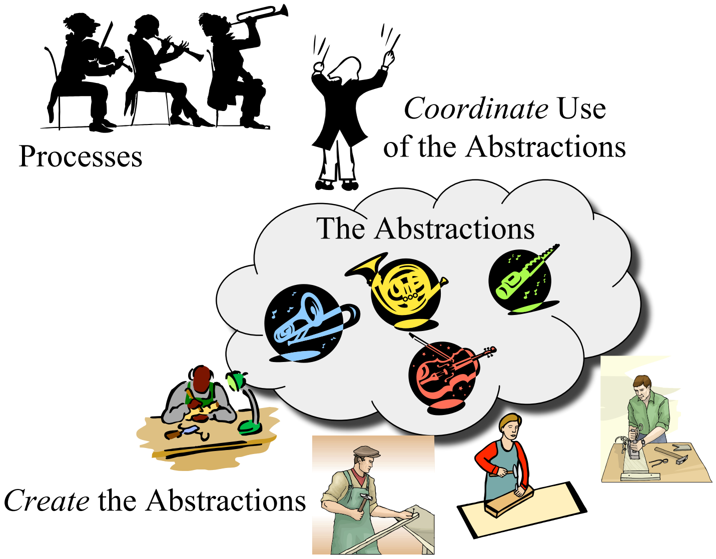
Slide 3-1

Operating System Organization

Purpose of an OS

Slide 3-2

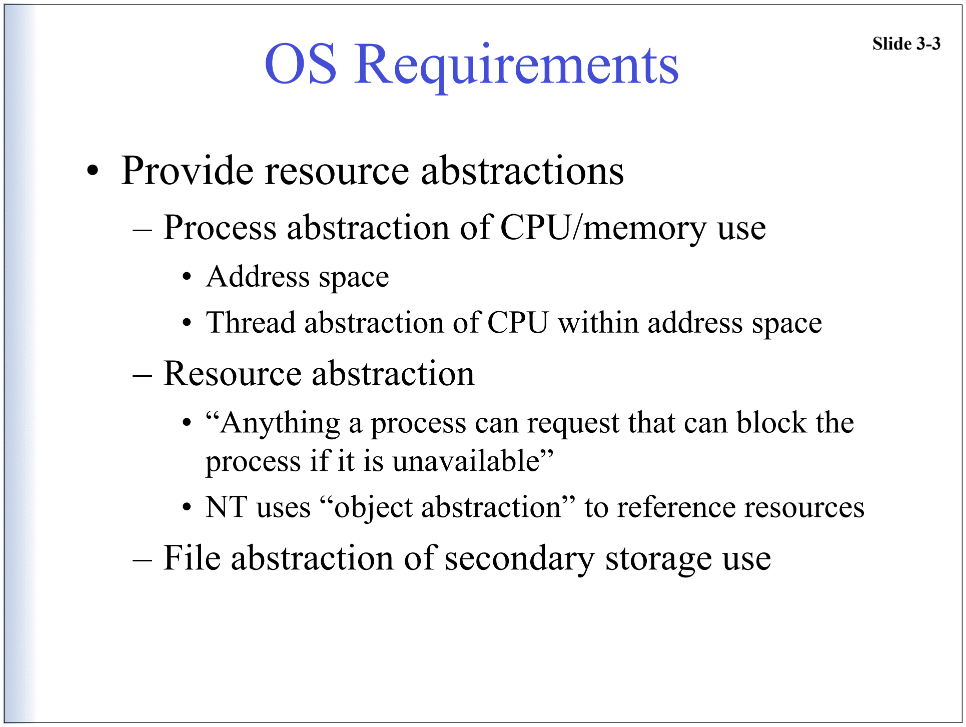


Processes

Coordinate Use of the Abstractions

The Abstractions

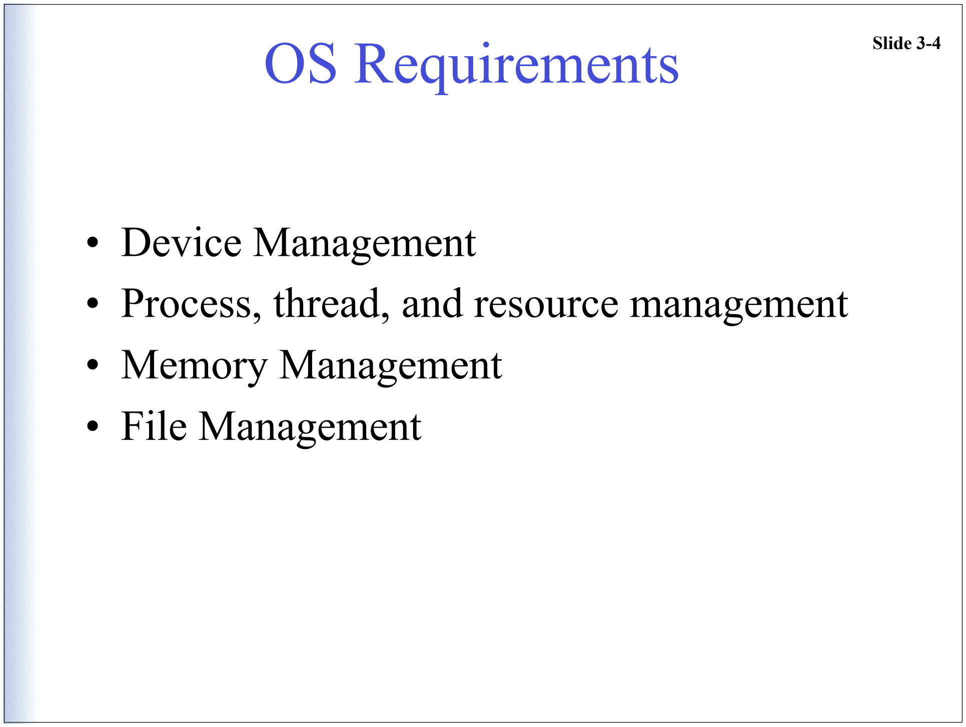
Create the Abstractions



Slide 3-3

OS Requirements

- Provide resource abstractions
  - Process abstraction of CPU/memory use
    - Address space
    - Thread abstraction of CPU within address space
  - Resource abstraction
    - “Anything a process can request that can block the process if it is unavailable”
    - NT uses “object abstraction” to reference resources
  - File abstraction of secondary storage use



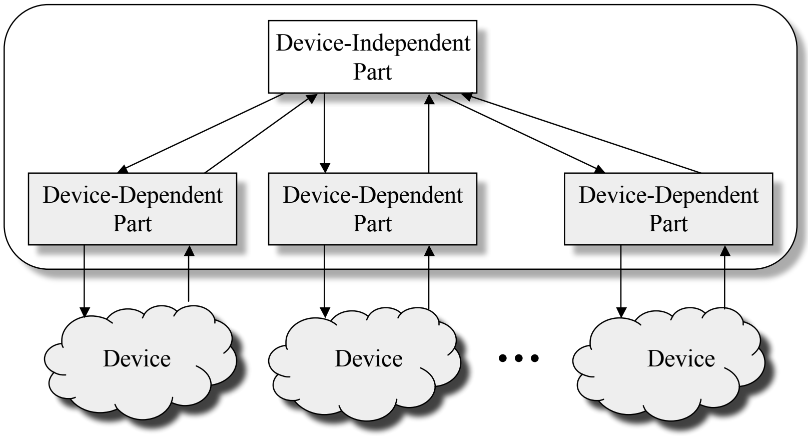
Slide 3-4

OS Requirements

- Device Management
- Process, thread, and resource management
- Memory Management
- File Management

Device Management

Slide 3-5



Virtual Device Drivers

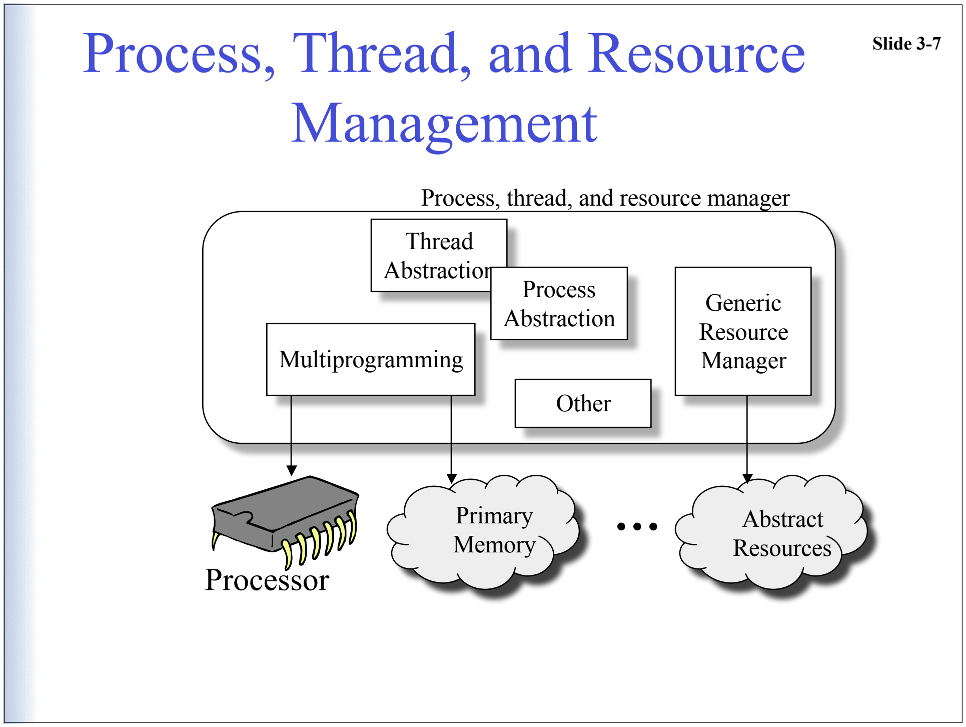
Slide 3-6

used in virtualization environments

emulate a piece of hardware - illusion of accessing real hardware

How does it work?

Attempts by the guest operating system
to access the hardware are routed to the virtual device
driver in the host operating system as function calls



Process, Thread, and Resource Management

Slide 3-7

Process, thread, and resource manager

Thread Abstraction

Process Abstraction

Generic Resource Manager

Multiprogramming

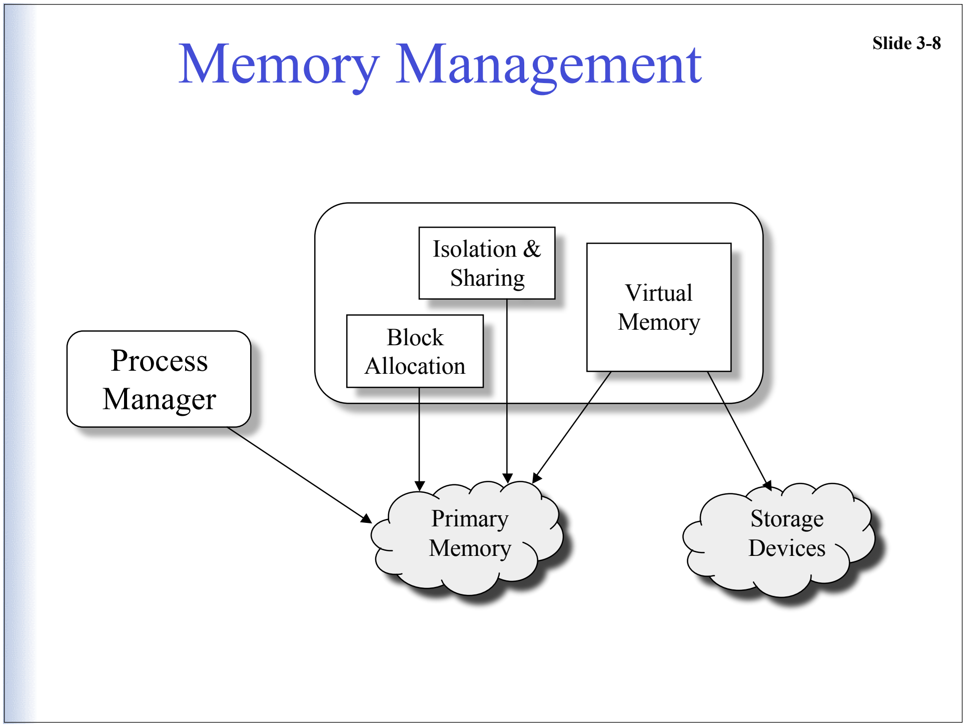
Other

Processor

Primary Memory

• • •

Abstract Resources



Memory Management

Slide 3-8

Isolation & Sharing

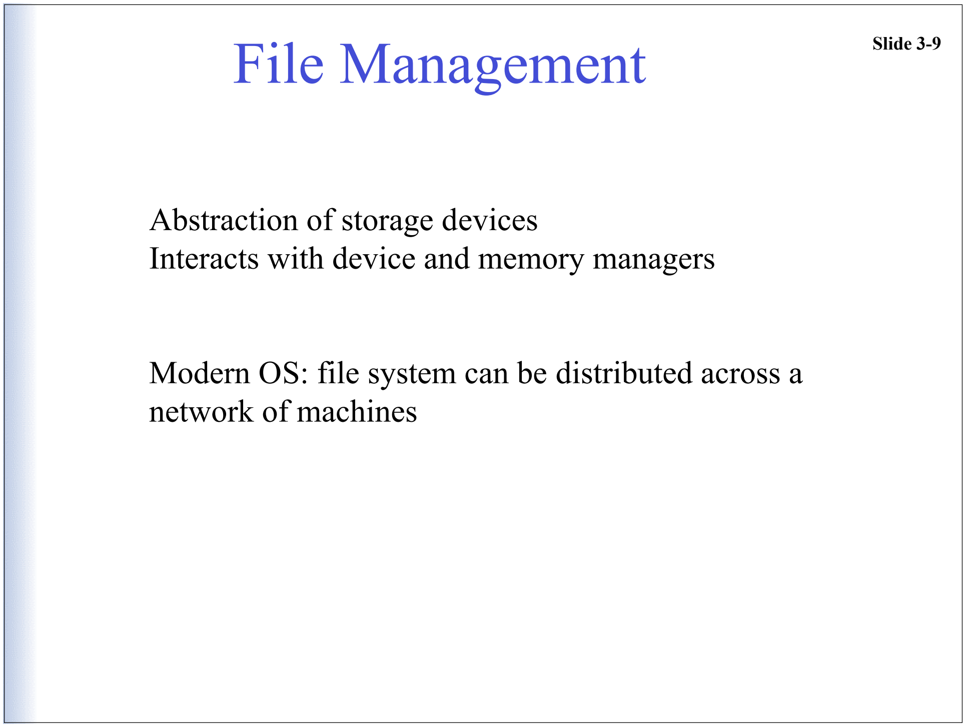
Virtual Memory

Block Allocation

Process Manager

Primary Memory

Storage Devices

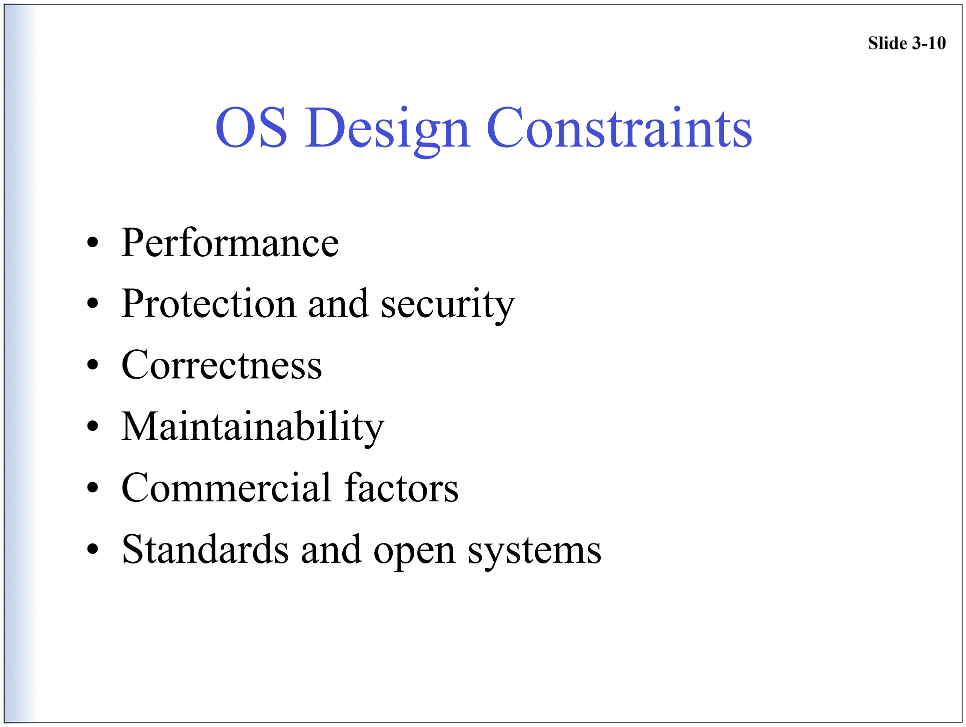


Slide 3-9

File Management

Abstraction of storage devices
Interacts with device and memory managers

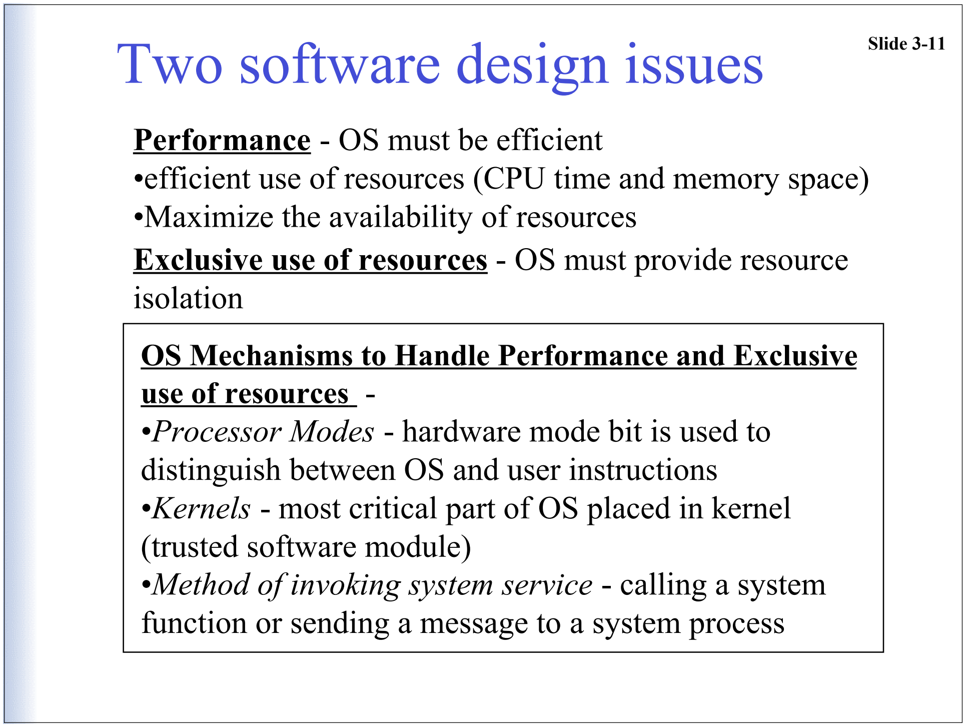
Modern OS: file system can be distributed across a network of machines



Slide 3-10

OS Design Constraints

- Performance
- Protection and security
- Correctness
- Maintainability
- Commercial factors
- Standards and open systems



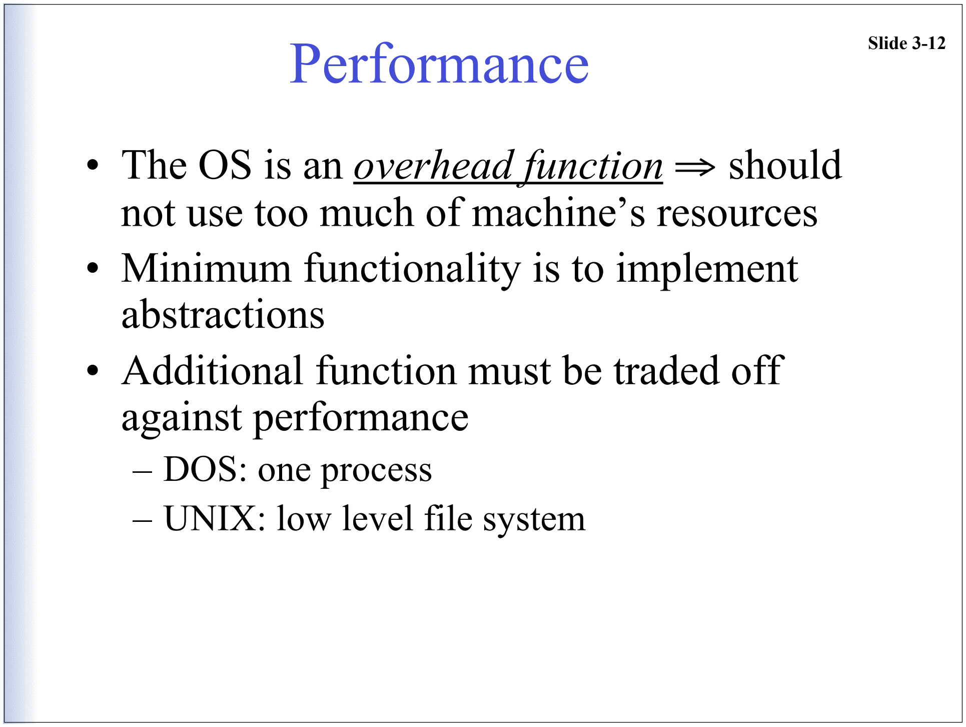
Slide 3-11

Two software design issues

Performance - OS must be efficient
•efficient use of resources (CPU time and memory space)
•Maximize the availability of resources

Exclusive use of resources - OS must provide resource isolation

OS Mechanisms to Handle Performance and Exclusive use of resources -
•*Processor Modes* - hardware mode bit is used to distinguish between OS and user instructions
•*Kernels* - most critical part of OS placed in kernel (trusted software module)
•*Method of invoking system service* - calling a system function or sending a message to a system process



Slide 3-12

Performance

- The OS is an *overhead function* ⇒ should not use too much of machine’s resources
- Minimum functionality is to implement abstractions
- Additional function must be traded off against performance
  - DOS: one process
  - UNIX: low level file system

Exclusive Access to a Resource

Slide 3-13

- Exclusive control of a resource - must be guaranteed
- OS provides mechanism to isolate processes
- OS must also provide ability for processes to share

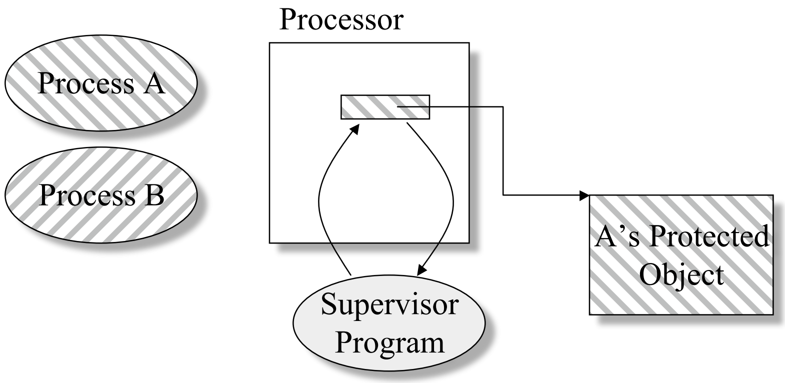
Security Policy - the machine specific strategy for managing access to resources

Trusted software - carefully constructed and part of OS (us)
Kernel

Untrusted software - temporary and unknown (them)
Apps, system software, and OS extensions

Exclusive Access to a Resource

Slide 3-14



Protection & Security

Slide 3-15

- Multiprogramming ⇒ resource sharing
- Therefore, need software-controlled resource isolation
- *Security policy*: Sharing strategy chosen by computer's owner
- *Protection mechanism*: Tool to implement a family of security policies

Processor Modes

Slide 3-16

- Mode bit: *Supervisor* or *User* mode
- Supervisor mode
  - Can execute all machine instructions
  - Can reference all memory locations
- User mode
  - Can only execute a subset of instructions
  - Can only reference a subset of memory locations

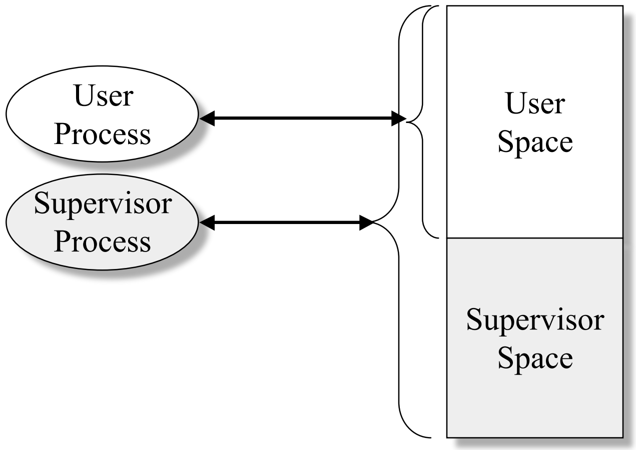
Kernels

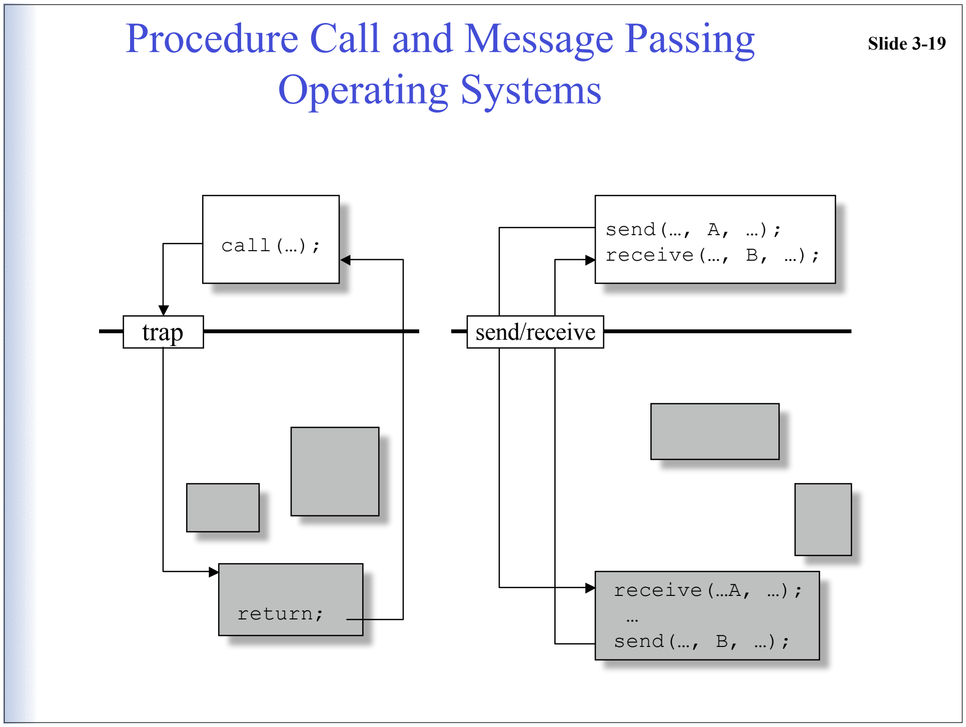
Slide 3-17

- The part of the OS critical to correct operation (trusted software)
- Executes in supervisor mode
- The `trap` instruction is used to switch from user to supervisor mode, entering the OS

Supervisor and User Memory

Slide 3-18





Procedure Call and Message Passing Operating Systems

Slide 3-19

```
call(…);
```

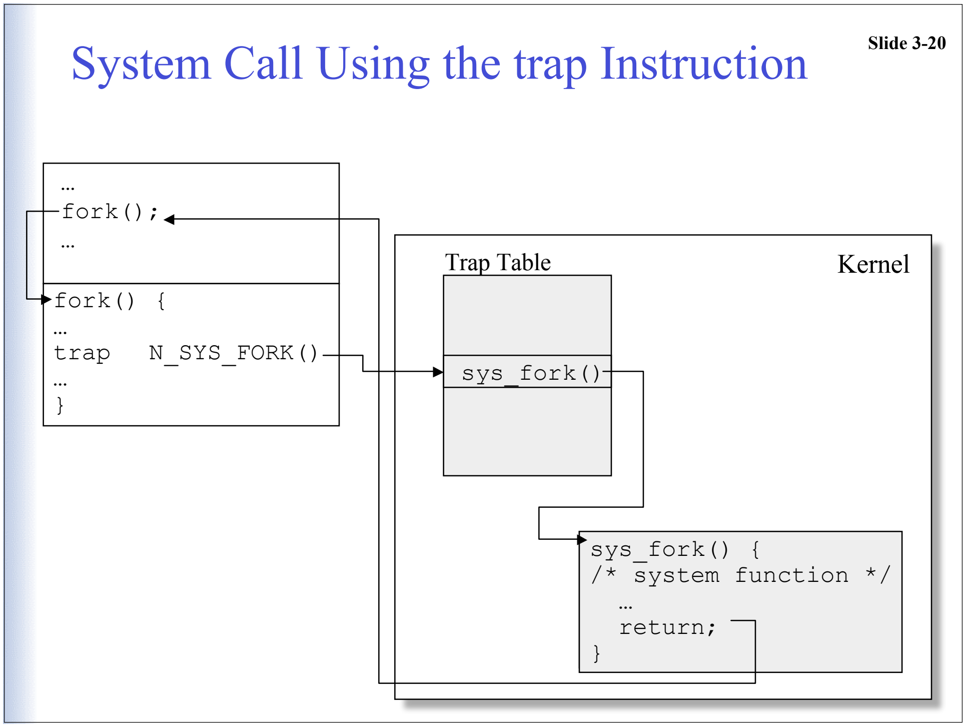
trap

```
return;
```

```
send(…, A, …);
receive(…, B, …);
```

send/receive

```
receive(…A, …);
 …
send(…, B, …);
```



System Call Using the trap Instruction

Slide 3-20

```
…
fork();
…
```

```
fork() {
…
trap   N_SYS_FORK()
…
}
```

Trap Table

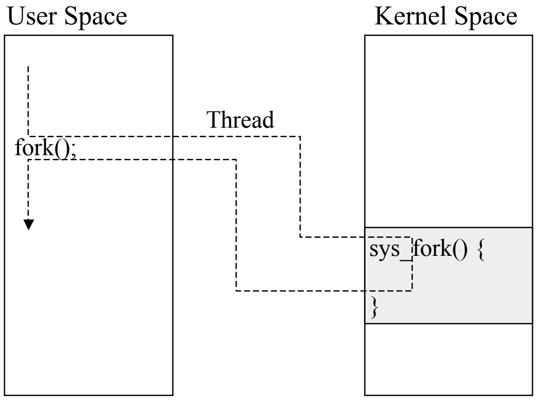
Kernel

```
sys_fork()
```

```
sys_fork() {
/* system function */
  …
  return;
}
```

A Thread Performing a System Call

Slide 3-21



User Space

Kernel Space

Thread

fork();

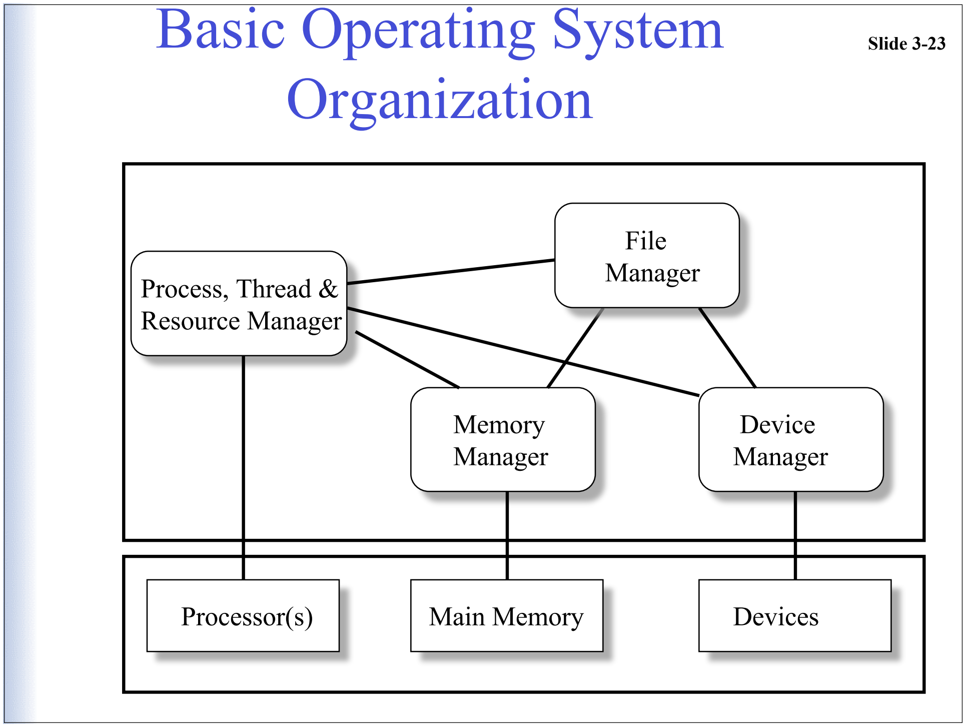
sys_fork() {

}

Correctness & Maintainability

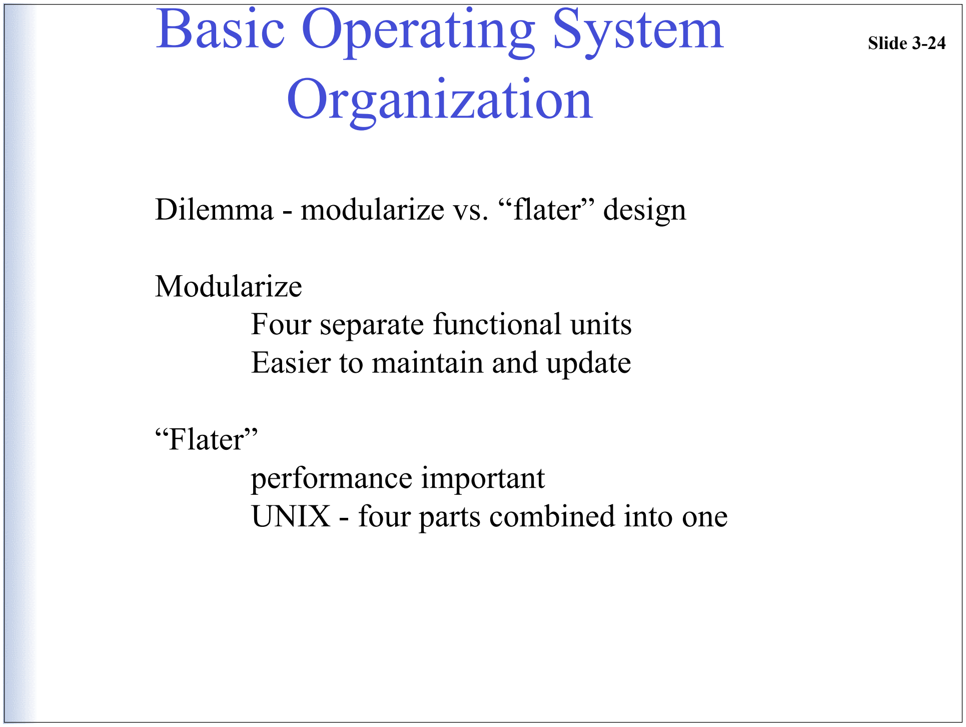
Slide 3-22

- Security depends on correct operation of software ⇒ *trusted* vs *untrusted* software
- Maintainability relates to ability of software to be changed
- If either is sufficiently important, can limit the function of the OS
  - Guiding a manned spaceship
  - Managing a nuclear reactor



Basic Operating System Organization

Slide 3-23



Basic Operating System Organization

Slide 3-24

Dilemma - modularize vs. “flater” design

Modularize

Four separate functional units

Easier to maintain and update

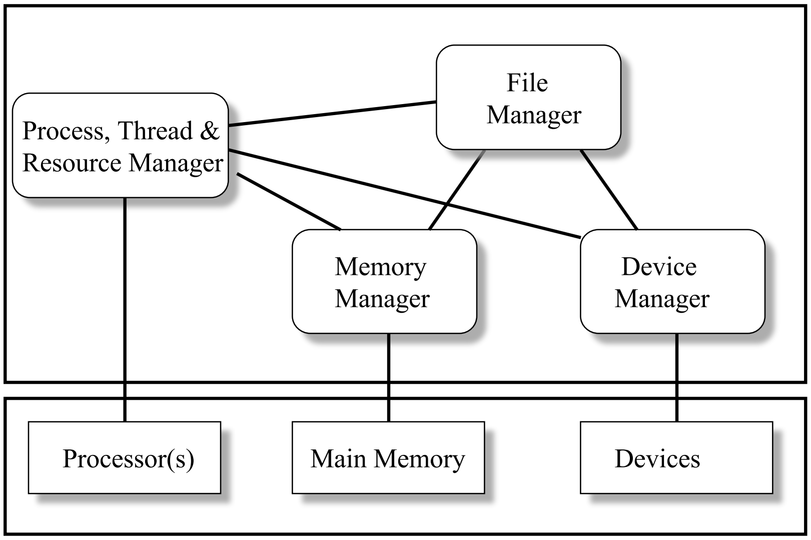
“Flater”

performance important

UNIX - four parts combined into one

Basic Operating System Organization

Slide 3-25



MicroKernel

Slide 3-26

MicroKernel - only essential “trusted” code

- thread scheduling
- hardware device management
- fundamental protection mechanisms
- other basic functions

remainder of the 4 - into user code

Must use system call to microkernel

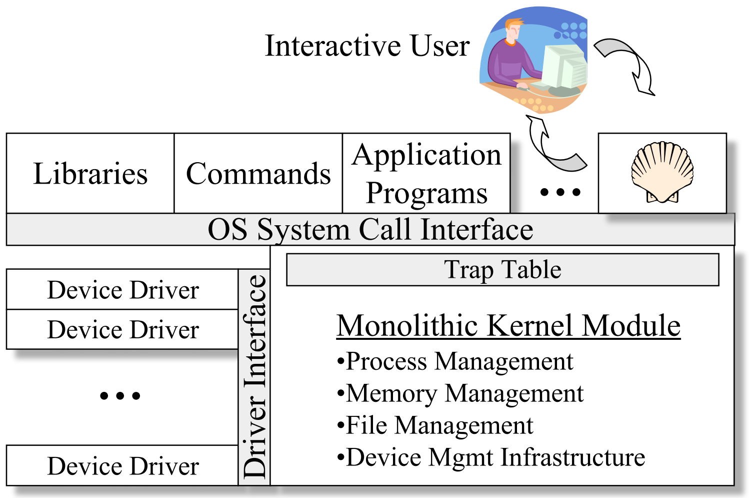
Slide 3-27

Modern OS Kernels

Unix - first to support multiprogramming and networking
Windows version - more widely used

Slide 3-28

The UNIX Architecture



Interactive User

Libraries

Commands

Application Programs

• • •

OS System Call Interface

Device Driver

Device Driver

• • •

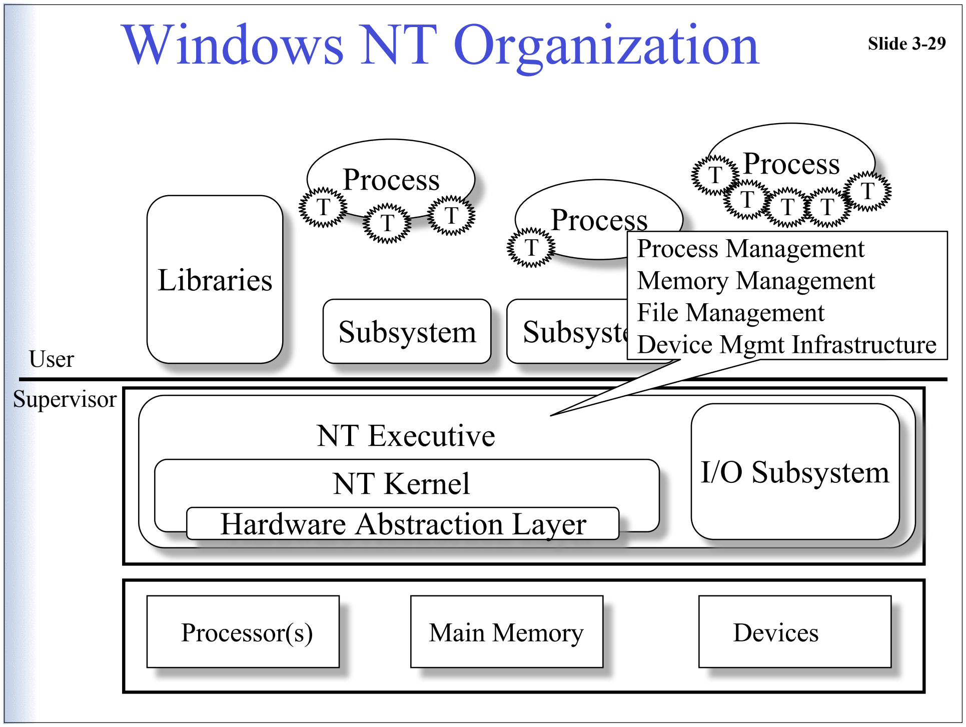
Device Driver

Driver Interface

Trap Table

Monolithic Kernel Module

- Process Management
- Memory Management
- File Management
- Device Mgmt Infrastructure



Windows NT Organization

Slide 3-29

Process

T T T

Process

T

Process

T T T T T

Libraries

Subsystem

Subsyste

Process Management
Memory Management
File Management
Device Mgmt Infrastructure

User

Supervisor

NT Executive

NT Kernel

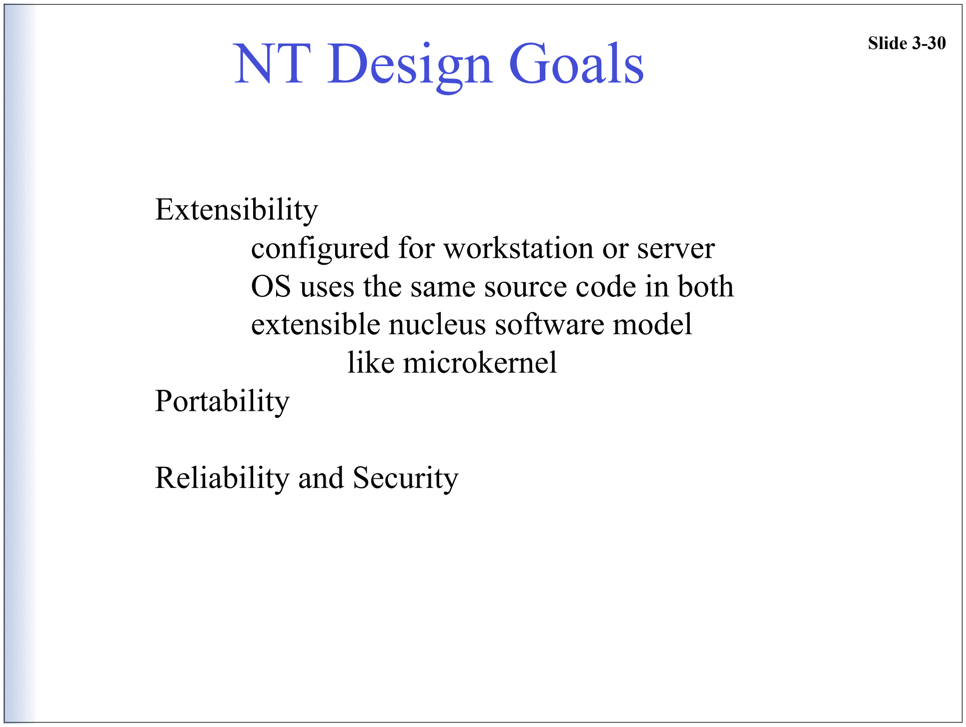
Hardware Abstraction Layer

I/O Subsystem

Processor(s)

Main Memory

Devices



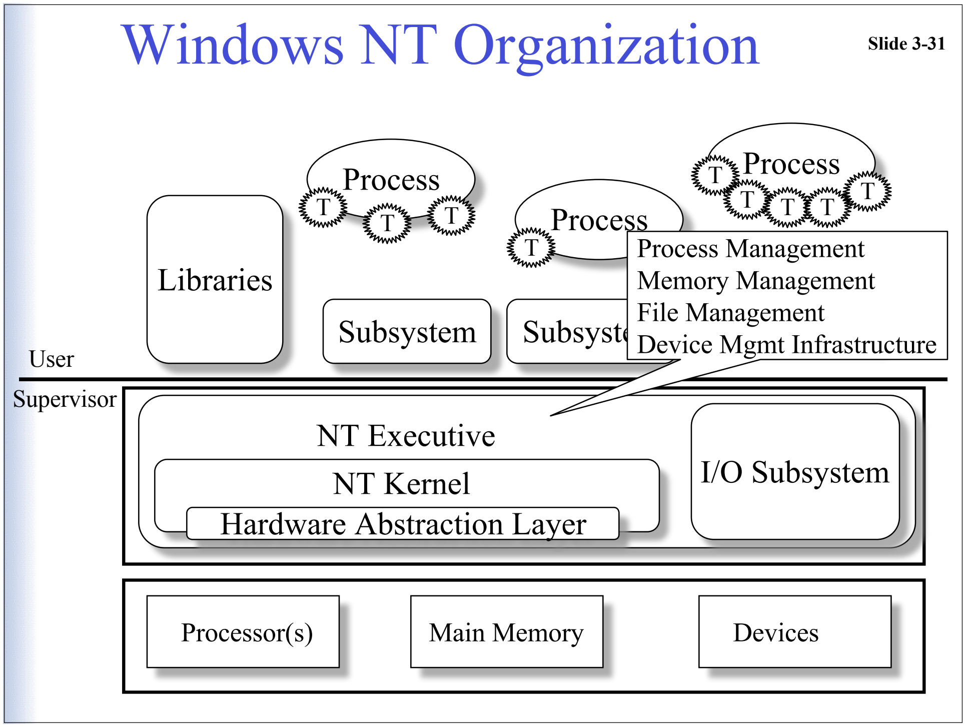
NT Design Goals

Slide 3-30

Extensibility
- configured for workstation or server
- OS uses the same source code in both
- extensible nucleus software model
  - like microkernel

Portability

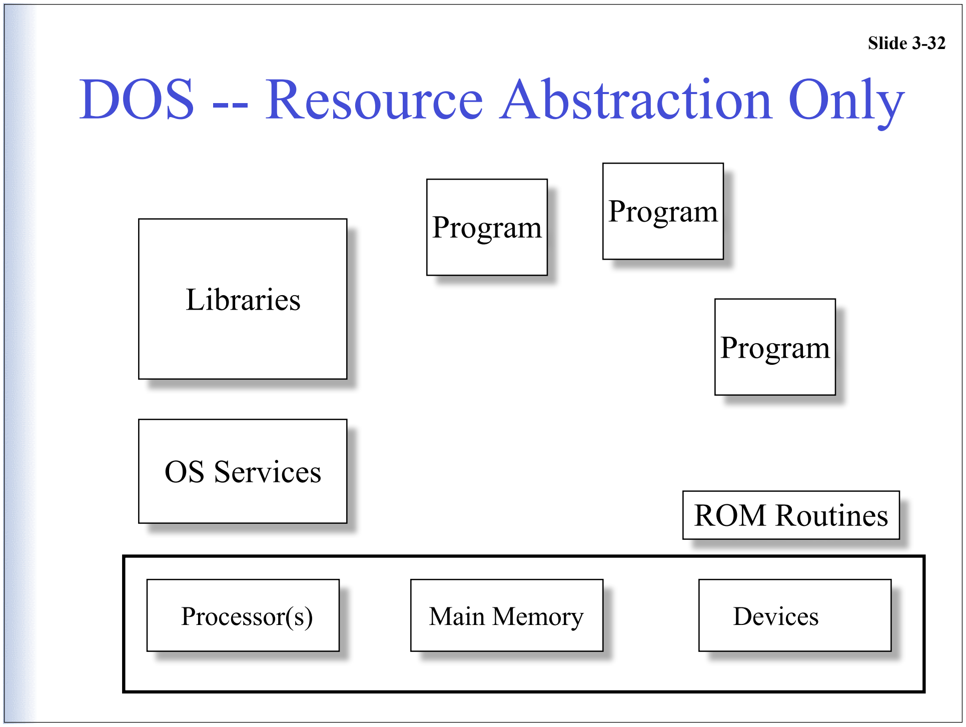
Reliability and Security



Windows NT Organization

Slide 3-31

Process Management
Memory Management
File Management
Device Mgmt Infrastructure

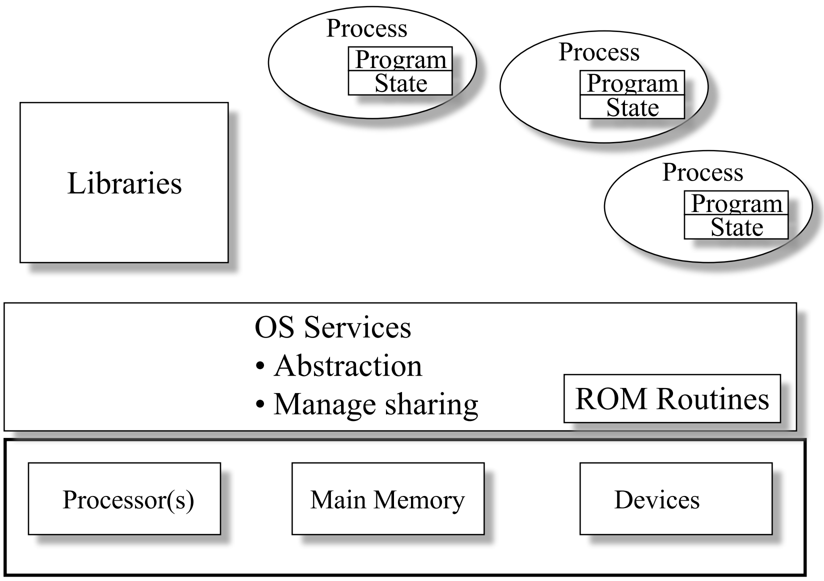


Slide 3-32

DOS -- Resource Abstraction Only

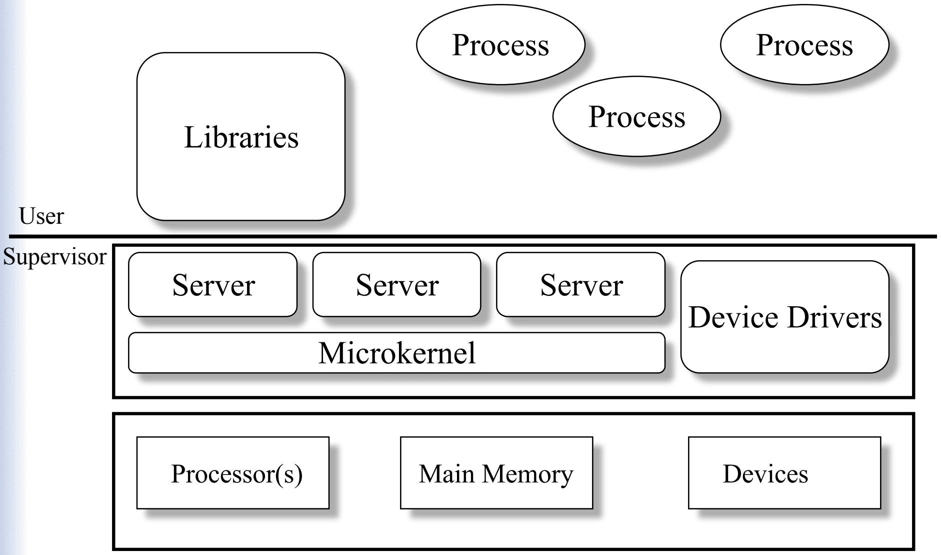
Slide 3-33

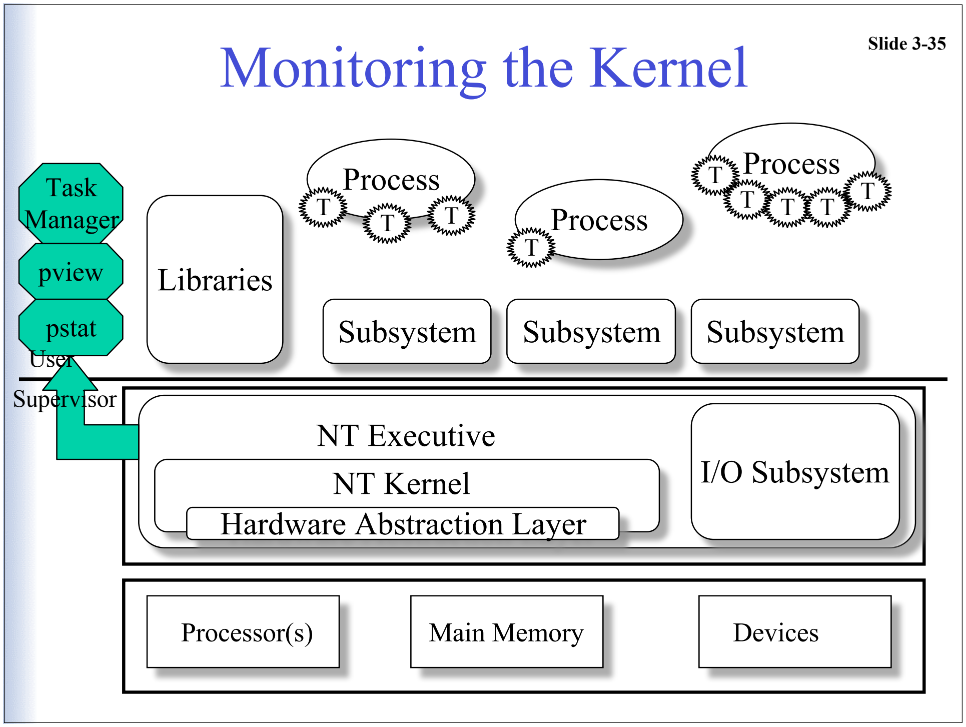
Abstraction & Sharing


Process
Program
State
Process
Program
State
Process
Program
State
Libraries
OS Services
- Abstraction
- Manage sharing
ROM Routines
Processor(s)
Main Memory
Devices

Slide 3-34

Microkernel Organization


Process
Process
Process
Libraries
User
Supervisor
Server
Server
Server
Device Drivers
Microkernel
Processor(s)
Main Memory
Devices



Monitoring the Kernel

Slide 3-35

Task Manager

pview

pstat

User

Supervisor

Libraries

Process

T T T

Process

T

Process

T T T T T

Subsystem

Subsystem

Subsystem

NT Executive

NT Kernel

Hardware Abstraction Layer

I/O Subsystem

Processor(s)

Main Memory

Devices