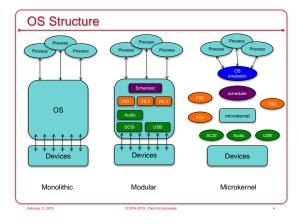
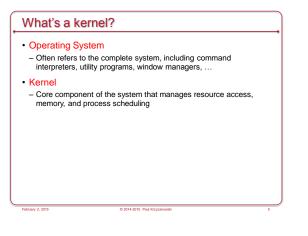


What is an operating system?

- The first program
- A program that lets you run other programs
- A program that provides controlled access to resources: – CPU
- Memory
- Display, keyboard, mouse
- Persistent storage
- Network

This includes: naming, sharing, protection, communication





Some of the things a kernel does

· Controls execution of processes

- Creation, termination, communication
- Schedules processes for execution on the CPU(s)
- Manages memory
- Allocates memory for an executing process
- Sets memory protection
- Coordinates swapping pages of memory to a disk if low on memory

© 2014-2015 Paul Krzyzanowski

- Manages a file system
 - Allocation and retrieval of disk data
- Enforcing access permissions & mutual exclusion
- · Provides access to devices
- Disk drives, networks, keyboards, displays, printers, ...
- Enforces access permissions & mutual exclusion



- Modify the memory management unit
- Set timers
- Define interrupt vectors
- Halt the processor
- Etc.
- CPU knows what mode it's in via a status register – You can set the register in kernel mode

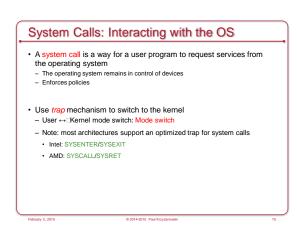
© 2014-2015 Paul Krzyzar

- OS & boot loaders run in kernel mode
- User programs run in user mode

How do you get to kernel mode? Irap: Transfer of control Like a subroutine call (return address placed on stack) Mode switch: user mode → kernel mode Interrupt Vector Table Configured by kernel at boot time Depending on architecture Code entry points Code entry points Contain a set of JMP instructions to different handlers in the kernel List of addresses Each entry contains a structure that defines the target address & privilege level Table will contain a set of addresses for different handlers in the kernel Returning back to user mode

Return from exception

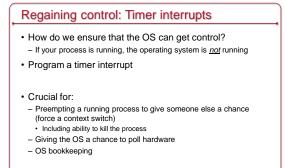
How do you get to kernel mode? Three types of traps: 1. Software interrupt – explicit instruction (Intel architecture: INT instruction (interrupt) (Intel architecture: SWI instruction (software interrupt) 2. Violation 3. Hardware interrupt Traps give us a mechanism to transfer to *well-defined* entry points in the kernel



System Calls: Interacting with the OS

- Use trap mechanism to switch to the kernel
- Pass a number that represents the OS service (e.g., read)
 System call number; usually set in a register
- · A system call does the following:
- Set the system call number
- Save parameters
- Issue the trap (jump to kernel mode)
- OS gets control
- Saves registers, does the requested work
 Return from exception (back to user mode)
- Retrieve results and return them to the calling function
- · System call interfaces are encapsulated as library functions

© 2014-2015 Paul Krzyzanowski



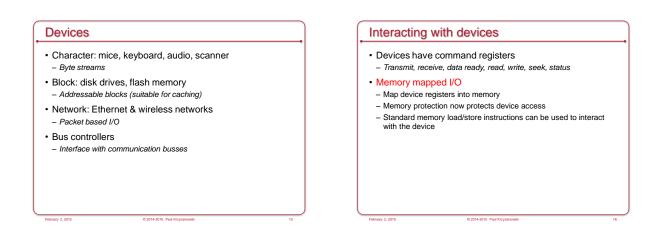
© 2014-2015 Paul Krzyzanowski

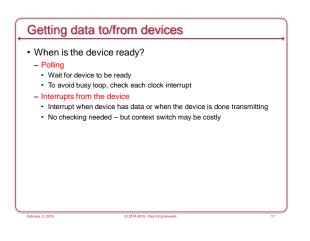
Timer interrupts

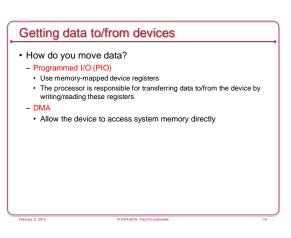
- · Windows
- Typically 64 or 100 interrupts per second
- Apps can raise this to 1024 interrupts per second
- Linux
 - Interrupts from Programmable Interval Timer (PIT) or HPET (High Precision Event Timer) and from a local APIC timer (one per CPU)
- Interrupt frequency varies per kernel and configuration
- Linux 2.4: 100 Hz
- Linux 2.6.0 2.6.13: 1000 Hz
- Linux 2.6.14+ : 250 Hz
- Linux 2.6.18 and beyond: aperiodic tickless kernel
 PIT not used for periodic interrupts; just APIC timer interrupts

Context switch & Mode switch An interrupt or trap results in a *mode switch*. An operating system may choose to save a process' state and restore another process' state → preemption Context switch Save all registers (including stack pointers, PC, and flags) Load saved registers (including SP, PC, flags) To return to original context: restore registers and return from exception South to kernel mode Save state so that it can be resored later Load another process' saved state

- Return (to the restored process)

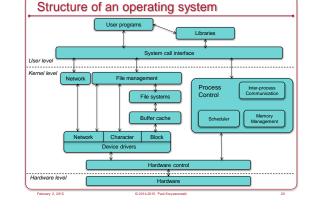


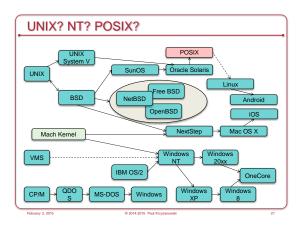


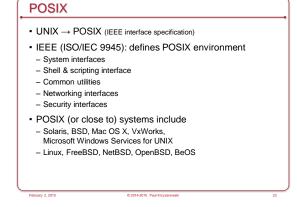




- Persistent storage of data
- Handle allocation of disk space
- Provide user-friendly names to identify the data
- Associate attributes with the data
- Create time, access time, owner, permissions, ...
 Device or data file?









| OS Mechanisms & Policies | |
|--|----|
| Mechanisms: | |
| Presentation of a software abstraction: | |
| Memory, data blocks, network access, processes | |
| Policies: | |
| Procedures that define the behavior of the mechanism | |
| Allocation of memory regions, replacement policy of data blocks | |
| Permissions | |
| Enforcement of access rights | |
| | |
| Keep mechanisms, policies, and permissions separate | |
| | |
| | |
| February 2, 2015 © 2014-2015 Paul Krzyzanowski | 24 |

Processes

· Mechanism:

- Create, terminate, suspend, switch, communicate
- Policy
 - Who is allowed to create and destroy processes?
- What is the limit?
- What processes can communicate?
- Who gets priority?
- · Permissions
- Is the process making the request allowed to perform the operation?

Threads

- · Mechanism:
 - Create, terminate, suspend, switch, synchronize
- Policy
 - Who is allowed to create and destroy threads?
 - What is the limit?
 - How do you assign threads to processors?
 - How do you schedule the CPU among threads of the same process?

Virtual Memory

· Mechanism:

- Logical to physical address mapping

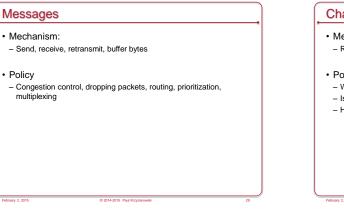
Policy

Policy

- How do you allocate physical memory among processes and among users?
- How do you share physical memory among processes?
- Whose memory do you purge when you're running low?

File Systems

- Mechanism:
- Create, delete, read, write, share files
- Manage a cache; memory map files
- Policy
- What protection mechanisms do you enforce?
- What disk blocks do you allocate?
- How do you manage cached blocks of data (Per file? Per user? Per process?)



Character Devices

- Mechanism:
 - Read, write, change device options
- Policy
 - Who is allowed to access the device?
 - Is sharing permitted?
 - How do you schedule device access?

