CSCI 3753
Operating Systems

Design Issues

Lecture Notes By

Shivakant Mishra

Computer Science, CU-Boulder

Last Update: 08/31/2017

Recap …

- An operating system is a layer of software between applications and hardware that provides useful services to applications
  - Extended machine view resource manager view
- Batch processing
- Multiprogramming
  - Context switch
- Multitasking
  - Preemptive and non-preemptive multitasking
- Readings: Please read Chapters 1 and 2

Three Design Issues

- System Boot
  - What happens when you switch on or reset a computing system
- Protecting OS from applications
  - How can we prevent an application program from corrupting the operating system
- System Call API
  - How is the system call interface that enables application programs to access OS services implemented

System Boot

- Operating system manages all programs: where they are stored, when to run them, etc.
- But how does the system know where the operating system is or how to load the kernel?
- *Booting* the system: Procedure of starting a computer by loading the operating system
- Bootstrap program (also called bootstrap loader)
  - Locates the OS kernel, loads it into main memory, and starts its execution
  - Typically a 2-step process: a simple bootstrap loader fetches a more complex boot program from disk, which in turn loads the kernel

System Boot

- When CPU receives a reset event (powered/reboot)
  - IR is loaded with a predefined memory location that contains the initial bootstrap program
  - In ROM: needs no initialization and cannot easily be infected
- Bootstrap program
  - Run diagnostics to determine the state of the machine
  - Initialize registers, main memory, device controllers, etc.
  - Start OS
- Smaller systems: store entire OS in ROM or EPROM (firmware)

System Boot (Larger Systems)

- Multi-stage procedure:
  1. Power On Self Test (POST) from ROM
     - Check hardware, e.g. CPU and memory, to make sure it’s OK

  2. BIOS (Basic Input/Output System) looks for a device to boot from…
     - May be prioritized to look for a USB flash drive or a CD/DVD-ROM drive before a hard disk drive
     - Can also boot from network

System Boot (Larger Systems)

- Multi-stage procedure: (continued)

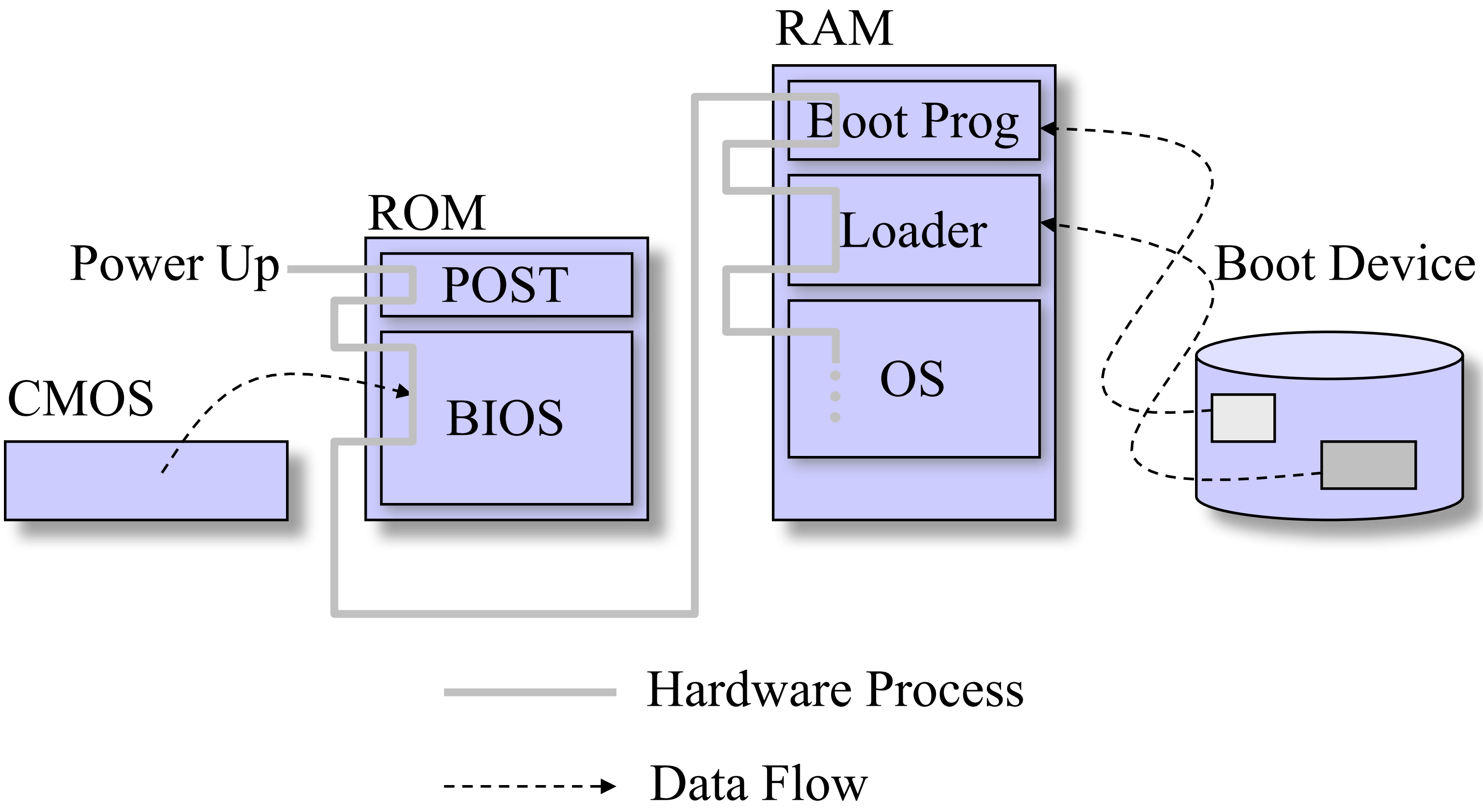
  3. BIOS finds a hard disk drive to boot from
     - Looks at Master Boot Record (MBR) in sector 0 of disk
     - Only 512 bytes long (Intel systems), contains primitive code for later stage loading and a partition table listing an active partition, or the location of the bootloader

System Boot (Larger Systems)

- Multi-stage procedure: (continued)

  4. Primitive loader then loads the secondary stage bootloader
     - Examples of this bootloader include LILO (Linux Loader), and GRUB (Grand Unified Bootloader)
     - Can select among multiple OS's (on different partitions) – i.e. dual booting
     - Once OS is selected, the bootloader goes to that OS's partition, finds the boot sector, and starts loading the OS's kernel

Intel System Initialization



-

Protecting OS from applications

- In early CPUs, there was no way to differentiate between the OS and applications:
  - Want to protect OS from being overwritten by app's
  - Want to prevent applications from executing certain privileged instructions, like resetting the time slice register, resetting the interrupt vector, etc.
- Processors include a hardware *mode* bit that identifies whether the system is in *user* mode or *supervisor/kernel* mode
  - Requires extra support from the CPU hardware for this OS feature

Processor mode

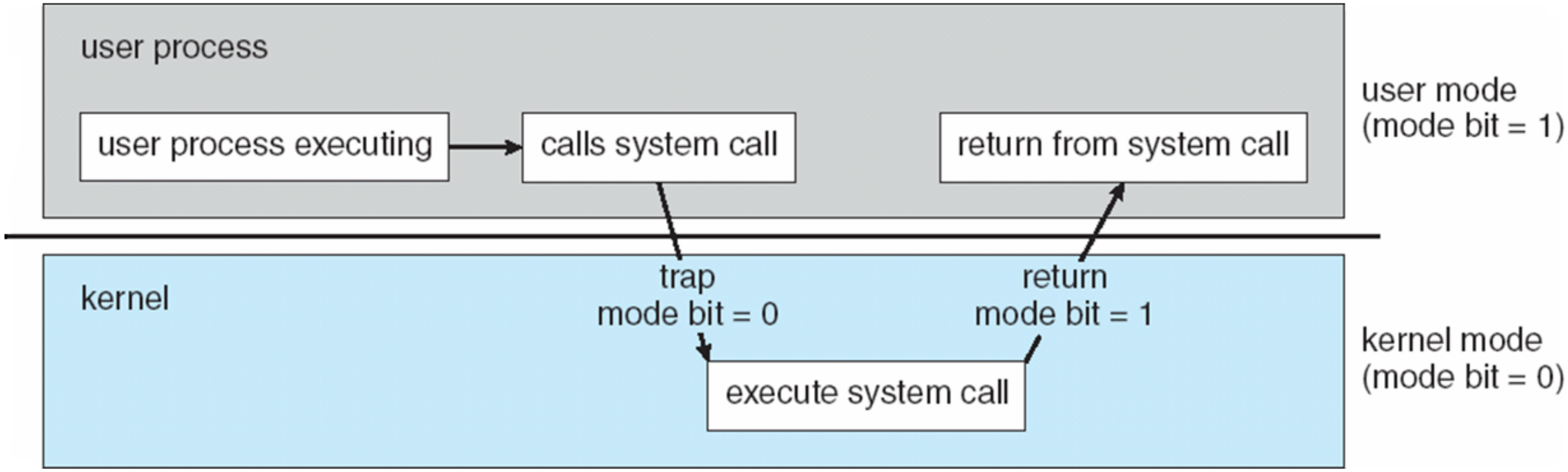
- Supervisor mode or user mode:
  - Supervisor mode (mode bit = 0): processor can execute every instruction available in the instruction set.
  - User mode (mode bit = 1): processor can execute only a subset of instructions available in the instruction set.
- Privileged (protected) instructions:
  - Instructions that can be executed only in supervisor mode.
  - I/O instructions
  - Protection and security: privileged load and store instructions
- Used to define two classes of memory space: user space and system space.
- Execution in kernel: supervisor mode (mode bit = 0)
- Execution in user space: user mode (mode bit = 1)

Mode Bit

- Key question
  - How is the mode bit changed from 0 to 1 or from 1 to 0?

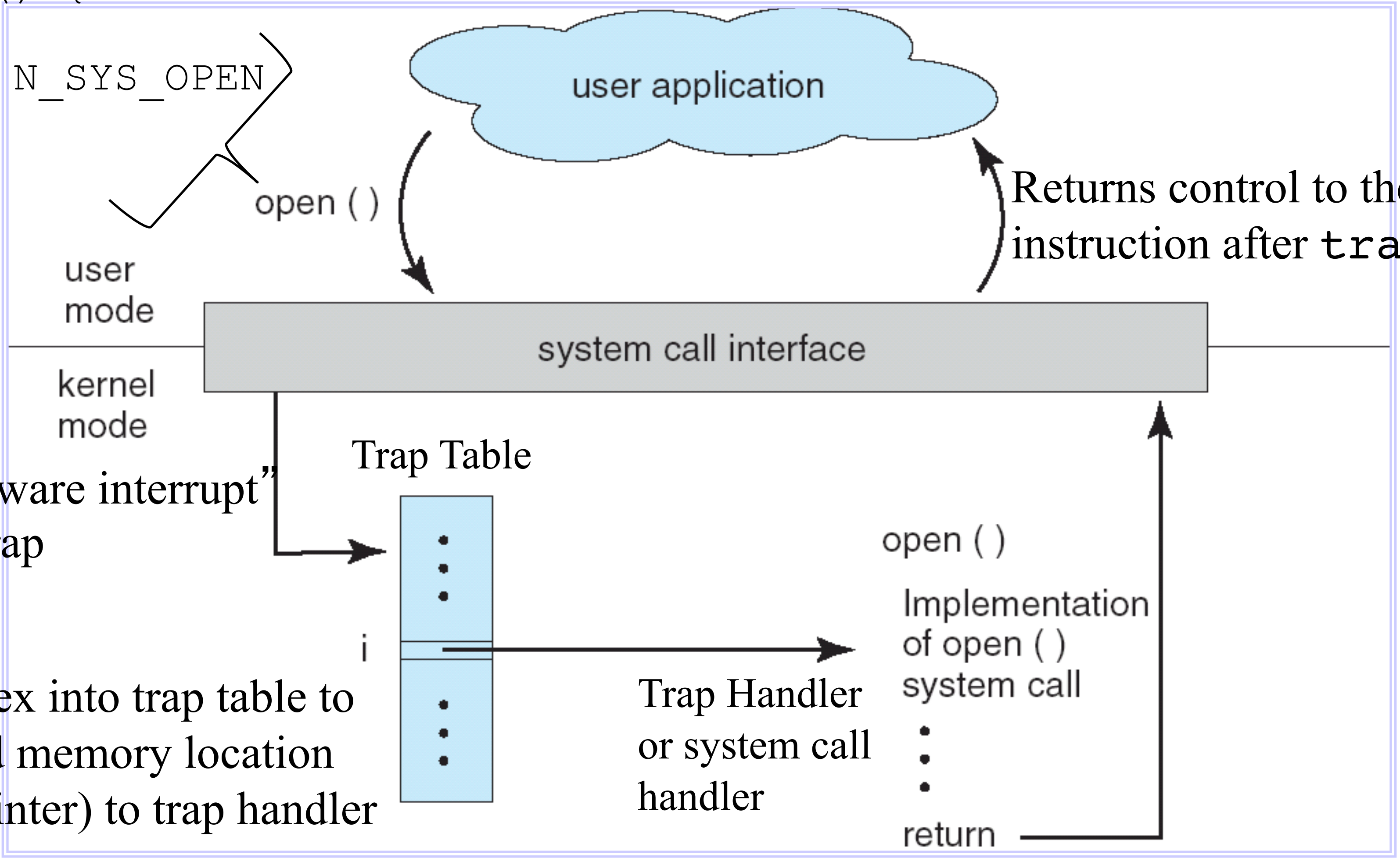
trap Instruction

- The trap instruction is used to switch from user to supervisor mode, thereby entering the OS
  - Also called syscall in MIPS
  - trap sets the mode bit to 0
  - mode bit set back to 1 on return
- Any instruction that invokes trap is called *a system call*
- trap indexes into a *trap table* (stored in kernel) and runs the function pointed to from that location



API – System Call – OS Relationship

```
open() {
…
trap N_SYS_OPEN
…
}
```



Returns control to the next instruction after `trap`

“software interrupt” = a trap

Trap Table

Trap Handler or system call handler

Index into trap table to find memory location (pointer) to trap handler

Trap Table

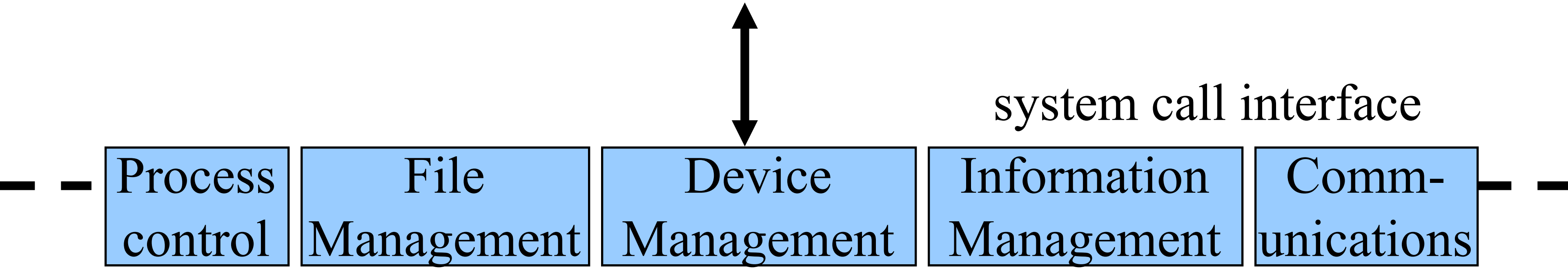
- The process of indexing into the trap table to jump to the trap handler routine is also called dispatching
- The trap table is also called a *jump table* or a *branch table*
- “A trap is a software interrupt”
- Trap handler (or system call handler) performs the specific processing desired by the system call/trap

System Call Parameter Passing

- Often, more information is required than simply the identity of desired system call
  - 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* in memory, and block address 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

Classes of System Calls Invoked by `trap`



system call interface

Process control

File Management

Device Management

Information Management

Comm-unications

- end, abort
- load, execute
- fork, create, terminate
- get attributes, set
- wait for time
- wait event, signal event
- allocate memory, free

- create, delete
- open, close
- read, write, reposition
- get attributes, set

- request device, release
- read, write, reposition
- get attributes, set
- logically attach or detach devices

Note Similarity

- get time/date, set
- get system data, set
- get process, file, or device attributes, set

- create connection, delete
- send messages, receive
- transfer status info
- attach remote devices, detach