Lecture 2
Computer System Structures
OS Structures

January 30, 2002

Outline

- Announcements
  - Project groups
  - Decision about midterm date: March 6th or 18th?
  - Questions?

- Computer-system structures
  - I/O structures
  - Storage structures and hierarchy
    - memory, secondary storage, tape
  - Hardware support for protection

- Operating-system structures
  - different views: functional, components, services, structure

[ Silberschatz/Galvin/Gagne: Chapters 2-3]

The Hardware of a Modern Computer System

The Hardware of a Modern Computer System

Computer-System Structures (1): Input/Output

- Device controllers
  - special-purpose processors
  - local buffer storage
  - controllers contain registers
    - control (write-only)
    - data (read-write)
    - status (read-only)

- How do the CPU and the device controllers communicate?
  - instructions
    - read/write I/O addresses (e.g., video memory)
    - registers in I/O controllers addressed as memory
  - interrupts
    - device controllers can interrupt the CPU
Interrupt Handling

- Interrupts are “asynchronous requests for service”
  - signal on a wire connecting the devices

- When an interrupt occurs, the CPU
  - preserves the present CPU state
    - this includes its registers and program counter
  - forces execution of code at an interrupt address
    - this may be dependent on the source of the interrupt
    - typically, table-driven: a table stores addresses of interrupt handlers
      - indexed by the interrupt number (ISR)
  - interrupt handlers
    - perform the requested service
    - selective processing of other interrupts
      - e.g., only higher-priority interrupts may be handled
  - resumes the interrupted program

- Most modern OSes are interrupt-driven

Interrupt Handling (contd.)

Interruptions vs. Traps

- Interrupts
  - asynchronous
  - triggered by devices outside the CPU

- Traps
  - synchronous
  - triggered by special instructions in user program

- Other than the above, handling of interrupts and traps is identical
- Traps are the hardware mechanism for implementing system calls

I/O Operation

- Two approaches: Synchronous and Asynchronous

- Problem with the above schemes: CPU handles all I/O
  - it can spend all its time doing interrupt processing
    - disk I/O, network I/O, video I/O
Solution: Direct Memory Access (DMA)

- The main idea: add a special device to “intervene” between the device controller and the system’s memory

Operation
- the CPU tells this DMA controller
  - the “chunk” size to be transferred
    - e.g., 128 - 4096 bytes (sectors) for disks
  - the starting address in memory where this chunk ought to be stored
- the DMA controller
  - accesses the secondary device via its controller
  - transfers the chunk from the device to system memory (and vice-versa)

Benefit: Interrupts are now less frequent
- at the level of chunks of data: only to indicate completion
- hence, CPU can do a lot of work between interrupts

Memory-Mapped I/O

- Traditionally, the CPU could directly access only main memory
  - all other devices are handled via controllers
  - accessed using special I/O instructions

- In most recent systems
  - controller’s registers are mapped into RAM space
    - results in uniform treatment of the I/O devices
      - can be handled via memory management procedures
    - all addressing is to RAM space
    - DMA access, interrupt handling, polling, …
  - controller’s buffers are mapped into RAM space
    - makes sense if the I/O is to a device that is particularly fast
      - e.g., a CRT screen where each pixel is an addressable location in RAM

Computer-System Structures (2): Storage

- Primary storage: Main memory (volatile)
  - accessed directly using load/store instructions
    - 1 cycle (registers), 2-5 cycles (cache), 20-50 cycles (RAM)
    - before: only one outstanding memory operation, CPU waits for completion
    - now: several outstanding operations
- Secondary storage: Disks (non-volatile)
  - accessed using a disk controller
  - supports random access but with non-uniform cost
- Tertiary storage: Tapes (non-volatile)
  - typically used only for backup
  - very inefficient support for random access
- Organized as a hierarchy
  - small amount of faster, more expensive storage closer to the CPU
  - larger amounts of slower, less expensive storage further away

Storage Hierarchy

- Rationale
  - keep CPU busy: lots of fast memory
  - keep system cost down

- How does it work
  - caching: upon access, move datum or instruction and its neighbors into higher levels of the hierarchy
  - replacement when a level fills up
  - copies need to be kept coherent

- Why does it work
  - Real programs demonstrate locality
    - e.g.: rows and columns of a matrix
    - e.g.: sequential instructions
  - once a datum or instruction is used, things “near” them are likely to be used “soon”
Computer-System Structures (3): Protection

- **Goal:** Prevent user processes from accidentally/maliciously damaging
  - the OS structures
  - parts of other process’s memory space
  - other user’s I/O devices

- **Mechanisms address different ways in which protection breaks down**
  1. **dual-mode operation**
     - Prevent user process taking over part of the OS and using this to overwrite other processes or even modify the OS itself (as in MS-DOS)
  2. **privileged instructions**
     - Prevent user process intervening in I/O of another process via control of the I/O handlers and indirectly causing damage
  3. **memory protection**
     - Prevent user process directly accessing another user process’ storage
  4. **CPU protection via timers**
     - Prevent hanging the OS – e.g., via an infinite loop

Protection Mechanisms (1): Dual-mode Operation and Privileged Instructions

- **Dual-mode operation**
  - *supervisor* and *user* modes
  - system starts off in supervisor mode and reenters it for interrupt processing
  - operating system gains control in supervisor mode

- **Privileged instructions**
  - restrict use of certain instructions to supervisor mode
  - I/O, including interrupt control
  - exception is instructions which generate interrupts
  - may be done by memory mapping
  - affect memory mapping
  - affect CPU mode (user/supervisor)
  - hardware support crucial for performance and for atomicity

Protection Mechanisms (2): Memory Protection

- **Basic method:** Memory is divided into segments

- **Furthermore**
  - logical addresses are mapped to physical addresses
  - provides sharing, etc.
  - hardware support for address mapping
  - a memory protection violation is detected
    - *user process traps to (interrupts) the OS*

Protection Mechanisms (3): Timers

- **OS code can enforce policies only if it gets a chance to run**

- **Timers maintain a count of elapsed (system) clock ticks**
  - when timer expires, the CPU is interrupted → run the OS code

- **Used for**
  - interrupting hung processes
  - context switching in time-shared systems

- **Access to timers is (usually) privileged**
  - WHY?
Outline

- Announcements
  - Project groups
  - Questions?
- Computer-system structures
  - I/O structures
  - Storage structures and hierarchy
    - memory, secondary storage, tape
  - Hardware support for protection
- Operating-system structures
  - different views: functional, components, services, structure

[Silberschatz/Galvin/Gagne: Chapters 2-3]

Hardware and OS Structures

OS Views (1): Functional View

- What are the functions performed by an OS?
  - Explicit operations
    - program execution and handling
    - I/O operations
    - file-system management
    - inter-process communication
    - exception detection and handling
      - e.g., notifying user that printer is out of paper
  - Implicit operations
    - resource allocation
    - accounting
    - protection
      - e.g., maintaining data integrity, logging invalid login attempts

OS Views (2): Components View

- Processes: run-time representations of user programs
  - create, terminate, suspend, resume
  - access to shared resources (e.g., printers)
- Storage
  - allocation of memory among resident processes
  - disk management (e.g., scheduling of disk accesses)
- I/O
  - device drivers, handling of device interrupts
  - files and directories
- Protection
  - user access to system resources

Course organization follows this view

User Applications

Support Applications
Compilers, Linkers, Windowing Systems, …

User-mode
Kernel-mode

Protection and Security
Storage Management
Process Management
Networking

Software
Hardware

Storage
memory
disk, tape
naming, caching

Devices
controllers
interrupts
DMA

CPU
dual-mode
priv. instructions
memory protection
timers
OS Views (3): Services View

- Two issues
  - What services does an OS provide? (same as functional view)
  - How do users and user programs access these services?

- Interface between the user and the OS: Command Interpreter
  - typical commands
    - process creation and (implicitly) destruction
    - I/O handling and file system manipulation
    - communication: interact with remote devices
    - protection management: changing file/directory access control, etc.
  - different varieties
    - the interpreter contains the code for the requested command (e.g., delete)
    - the interpreter calls a system routine to handle the request
    - the interpreter spawns new process(es) to handle the request
  - process lookup through some general procedure

  ➤ you will implement a simple shell in Nachos Lab 5

OS Views (3): Services View (contd.)

- Interface between a user program and the OS: System Calls
  - arguments passed in registers, a memory block, or on the stack
  - entry into the kernel using the trap mechanism

- Standard system calls
  - process control
  - file manipulation
  - device manipulation
  - information maintenance
    - get/set system data (time, memory/cpu usage), process and device attributes
  - communications

OS Views (4): Structure View

- How to structure OS functionality
  - Layering
  - Microkernels
  - Virtual machines

- Designing and implementing an OS
  - Read Sections 3.5-3.9, Silberschatz, Galvin, and Gagne
  - Look at Nachos source code
    - Thomas Narten’s roadmap

Lecture Summary

- Computer-system structures (Chapter 2)
  - I/O structures
  - storage structures
  - support for protection

- Operating-system structures (Chapter 3)
  - different views: functional, components, services, structure

- Next lecture
  - Processes (Sections 4.1 – 4.5)