

October 3, 2001

- Computer Architecture
- Operating System Functions
- Processes

- Lab 1 has been posted, due Oct 17



MULTICS (late 60s, early 70s)

- Computing as a *utility*.
- Shared computer over a vast filesystem of shared programs and data
- Virtual memory (paged segmentation)
 - *virtual address is 18 bit segment, 16 bit offset.*
 - *Second-choice page-replacement.*
- PL/1, about 300K lines of code



UNIX (1969--)

- Developed by Ken Thompson to use an otherwise-idle PDP-7, soon joined by Dennis Ritchie.
- Based on MULTICS
- Basic organization is a
 - *kernel*
 - *filesystem*
 - *command interpreter (shell)*



Personal-Computer Systems

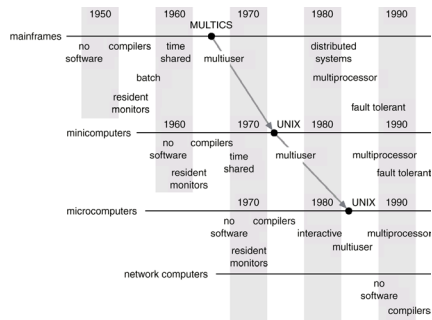
- *Personal computers* – computer system dedicated to a single user.
- I/O devices – keyboards, mice, display screens, small printers.
- User convenience and responsiveness.
- Can adopt technology developed for larger operating system' often individuals have sole use of computer and do not need advanced CPU utilization of protection features.



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

Migration of Operating-System Concepts and Features



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

Parallel Systems

- Multiprocessor systems with more than one CPU in close communication.
- *Tightly coupled system* – processors share memory and a clock; communication usually takes place through the shared memory.
- Advantages of parallel system:
 - Increased throughput
 - Economical
 - Increased reliability
 - graceful degradation
 - fail-soft systems



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

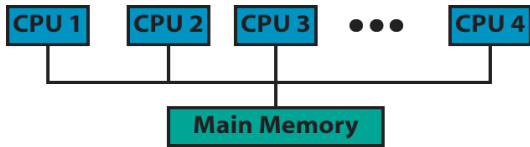
October 3, 2001

Parallel Systems (Cont.)

- **Symmetric multiprocessing (SMP)**
 - Each processor runs an identical copy of the operating system.
 - Many processes can run at once without performance deterioration.
 - Most modern operating systems support SMP
- **Asymmetric multiprocessing**
 - Each processor is assigned a specific task; master processor schedules and allocates work to slave processors.
 - More common in extremely large systems



Symmetric Multiprocessing Architecture



Real-Time Systems

- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.
- Well-defined fixed-time constraints.
- **Hard real-time system**
 - Secondary storage limited or absent, data stored in short-term memory, or read-only memory (ROM)
 - Conflicts with time-sharing systems, not supported by general-purpose operating systems.
- **Soft real-time system**
 - Limited utility in industrial control or robotics
 - Useful in applications (multimedia, virtual reality) requiring advanced operating-system features.



Distributed Systems

- Distribute the computation among several physical processors.
- *Loosely coupled system* – each processor has its own local memory; processors communicate with one another through various communications lines, such as high-speed buses or telephone lines.
- Advantages of distributed systems.
 - *Resources Sharing*
 - *Computation speed up – load sharing*
 - *Reliability*
 - *Communications*

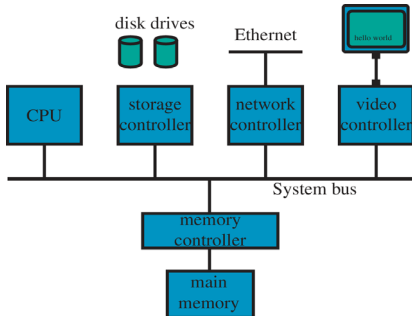


Distributed Systems (Cont.)

- Network Operating System
 - *provides file sharing*
 - *provides communication scheme*
 - *runs independently from other computers on the network*
- Distributed Operating System
 - *less autonomy between computers*
 - *gives the impression there is a single operating system controlling the network.*



Computer-System Architecture



Computer-System Operation

- I/O devices and the CPU can execute concurrently.
- Each device controller is in charge of a particular device type.
- Each device controller has a local buffer.
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller.
- Device controller informs CPU that it has finished its operation by causing an *interrupt*.



Common Functions of Interrupts

- Interrupts transfers control to the interrupt service routine generally, through the *interrupt vector*, which contains the addresses of all the service routines.
- Interrupt architecture must save the address of the interrupted instruction.
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*.
- A *trap* is a software-generated interrupt caused either by an error or a user request.
- An operating system is *interrupt driven*.



I/O Structure

- After I/O starts, control returns to user program only upon I/O completion.
 - *wait instruction idles the CPU until the next interrupt*
 - *wait loop (contention for memory access).*
 - *At most one I/O request is outstanding at a time, no simultaneous I/O processing.*
- After I/O starts, control returns to user program without waiting for I/O completion.
 - *System call - request to the operating system to allow user to wait for I/O completion.*
 - *Device-status table contains entry for each I/O device indicating its type, address, and state.*
 - *Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt.*



Storage Structure

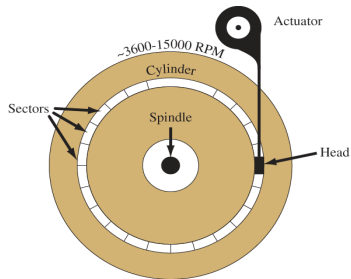
- Main memory – only large storage media that the CPU can access directly.
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity.
- Magnetic disks – rigid metal or glass platters covered with magnetic recording material
 - Disk surface is logically divided into tracks, which are subdivided into sectors.
 - The disk controller determines the logical interaction between the device and the computer.



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

Moving-Head Disk Mechanism



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

Storage Hierarchy

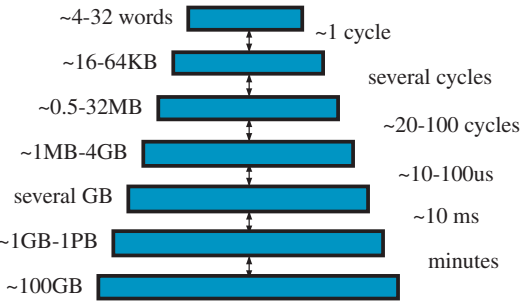
- Storage systems organized in hierarchy.
 - Speed
 - cost
 - volatility
- Caching – copying information into faster storage system; main memory can be viewed as a last cache for secondary storage.



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

Storage-Device Hierarchy



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

Hardware Protection

- Dual-Mode Operation
- I/O Protection
- Memory Protection
- CPU Protection



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

Dual-Mode Operation

- Sharing system resources requires operating system to ensure that an incorrect program cannot cause other programs to execute incorrectly.
- Provide hardware support to differentiate between at least two modes of operations.
 1. User mode – execution done on behalf of a user.
 2. Monitor mode (also supervisor mode or system mode) – execution done on behalf of operating system.

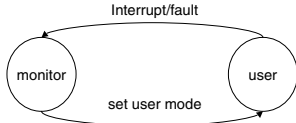


Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

Dual-Mode Operation (Cont.)

- *Mode bit* added to computer hardware to indicate the current mode: monitor (0) or user (1).
- When an interrupt or fault occurs hardware switches to monitor mode.



Privileged instructions can be issued only in monitor mode.

Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

Memory Protection

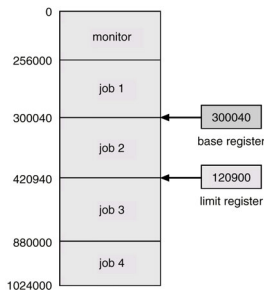
- Must provide memory protection at least for the interrupt vector and the interrupt service routines.
- In order to have memory protection, add two registers that determine the range of legal addresses a program may access:
 - *base register* - holds the smallest legal physical memory address.
 - *limit register* - contains the size of the range
- Memory outside the defined range is protected.



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

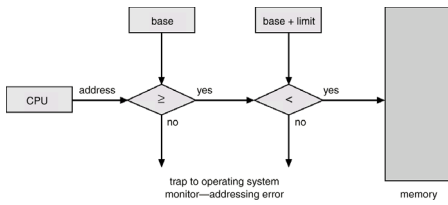
A Base And A limit Register Define A Logical Address Space



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

Protection Hardware



- When executing in monitor mode, the operating system has unrestricted access to both monitor and user's memory.
- The load instructions for the *base* and *limit* registers are privileged instructions.



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

General-System Architecture

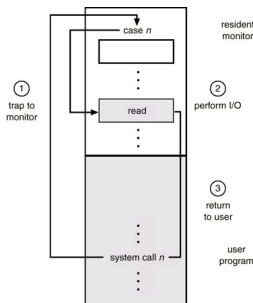
- Given the I/O instructions are privileged, how does the user program perform I/O?
- System call – the method used by a process to request action by the operating system.
 - Usually takes the form of a trap to a specific location in the interrupt vector.
 - Control passes through the interrupt vector to a service routine in the OS, and the mode bit is set to monitor mode.
 - The monitor verifies that the parameters are correct and legal, executes the request, and returns control to the instruction following the system call.



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

Use of A System Call to Perform I/O



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

Common System Components

- Process Management
- Main Memory Management
- Secondary-Storage Management
- I/O System Management
- File Management
- Protection System
- Networking
- Command-Interpreter System



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

Process Management

- A *process* is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.
- The operating system is responsible for the following activities in connection with process management:
 - *Process creation and deletion.*
 - *process suspension and resumption.*
 - *Provision of mechanisms for:*
 - *process synchronization*
 - *process communication*



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

Main-Memory Management

- Memory is a large array of words or bytes, each with its own address. It is a repository of quickly accessible data shared by the CPU and I/O devices.
- Main memory is a volatile storage device. It loses its contents in the case of system failure.
- The operating system is responsible for the following activities in connections with memory management:
 - *Keep track of which parts of memory are currently being used and by whom.*
 - *Decide which processes to load when memory space becomes available.*
 - *Allocate and deallocate memory space as needed.*



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

Secondary-Storage Management

- Since main memory (*primary storage*) is volatile and too small to accommodate all data and programs permanently, the computer system must provide *secondary storage* to back up main memory.
- Most modern computer systems use disks as the principle on-line storage medium, for both programs and data.
- The operating system is responsible for the following activities in connection with disk management:
 - *Free space management*
 - *Storage allocation*
 - *Disk scheduling*



I/O System Management

- The I/O system consists of:
 - *A buffer-caching system*
 - *A general device-driver interface*
 - *Drivers for specific hardware devices*



File Management

- A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data.
- The operating system is responsible for the following activities in connections with file management:
 - *File creation and deletion.*
 - *Directory creation and deletion.*
 - *Support of primitives for manipulating files and directories.*
 - *Mapping files onto secondary storage.*
 - *File backup on stable (nonvolatile) storage media.*



Protection System

- *Protection* refers to a mechanism for controlling access by programs, processes, or users to both system and user resources.
- The protection mechanism must:
 - *distinguish between authorized and unauthorized usage.*
 - *specify the controls to be imposed.*
 - *provide a means of enforcement.*



Command-Interpreter System

- Many commands are given to the operating system by control statements which deal with:
 - *process creation and management*
 - *I/O handling*
 - *secondary-storage management*
 - *main-memory management*
 - *file-system access*
 - *protection*
 - *networking*



Command-Interpreter System (Cont.)

- The program that reads and interprets control statements is called variously:
 - *control-card interpreter*
 - *command-line interpreter*
 - *shell (in UNIX)*

Its function is to get and execute the next command statement.



Operating System Services

- Program execution – system capability to load a program into memory and to run it.
- I/O operations – since user programs cannot execute I/O operations directly, the operating system must provide some means to perform I/O.
- File-system manipulation – program capability to read, write, create, and delete files.
- Communications – exchange of information between processes executing either on the same computer or on different systems tied together by a network. Implemented via *shared memory* or *message passing*.
- Error detection – ensure correct computing by detecting errors in the CPU and memory hardware, in I/O devices, or in user programs.



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

Additional Operating System Functions

Additional functions exist not for helping the user, but rather for ensuring efficient system operations.

- *Resource allocation* – allocating resources to multiple users or multiple jobs running at the same time.
- *Accounting* – keep track of and record which users use how much and what kinds of computer resources for account billing or for accumulating usage statistics.
- *Protection* – ensuring that all access to system resources is controlled.



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

System Calls

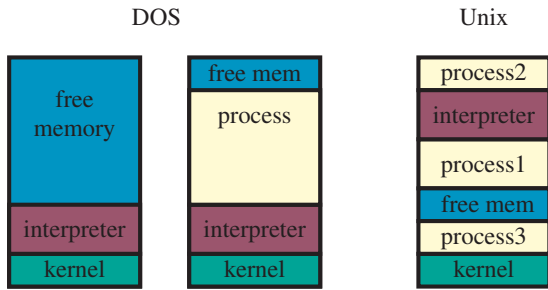
- System calls provide the interface between a running program and the operating system.
 - Generally available as assembly-language instructions.
 - Languages defined to replace assembly language for systems programming allow system calls to be made directly (e.g., C, Bliss, PL/360)
- Three general methods are used to pass parameters between a running program and the operating system.
 - Pass parameters in registers.
 - Store the parameters in a table in memory, and the table address is passed as a parameter in a register.
 - Push (store) the parameters onto the stack by the program, and pop off the stack by operating system.



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

MS-DOS/Unix Execution



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

System Programs

- System programs provide a convenient environment for program development and execution. They can be divided into:
 - File manipulation
 - Status information
 - File modification
 - Programming language support
 - Program loading and execution
 - Communications
 - Application programs
- Most users' view of the operation system is defined by system programs, not the actual system calls.



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

System Structure - Simple Approach

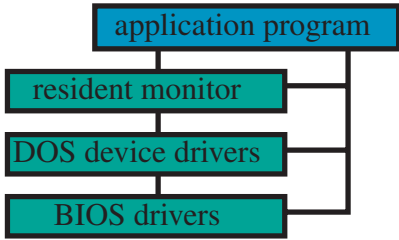
- MS-DOS - written to provide the most functionality in the least space
 - not divided into modules
 - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

MS-DOS Layer Structure



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

System Structure - Simple Approach (Cont.)

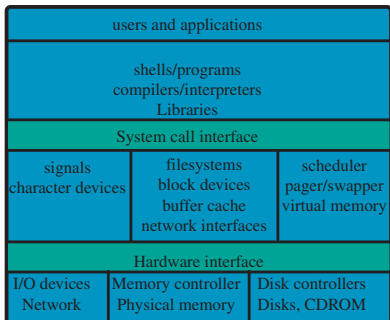
- UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring. The UNIX OS consists of two separable parts.
 - *Systems programs*
 - *The kernel*
 - *Consists of everything below the system-call interface and above the physical hardware*
 - *Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level.*



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

UNIX System Structure



Computing Systems
<http://www.cs.caltech.edu/cs134/cs134a>

October 3, 2001

System Structure - Layered Approach

- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers.



Layered Structure of the THE OS

- A layered design was first used in THE operating system. Its six layers are as follows:
 - layer 5: user programs
 - layer 4: buffering for input and output
 - layer 3: operator-console device driver
 - layer 2: memory management
 - layer 1: CPU scheduling
 - layer 0: hardware



System Design Goals

- User goals - operating system should be convenient to use, easy to learn, reliable, safe, and fast.
- System goals - operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient.