Lecture 2

Introduction/Overview II

Lecture Information

Ceng328 *Operating Systems* at February 23, 2010

Dr. Cem Özdoğan
Computer Engineering Department
Çankaya University

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Contents

1 Evolution of Operating Systems, Computer-System Architecture and Op
- Mainframe Systems
- Batch and Multiprogrammed Systems
- Time Sharing
- Personnel Computers and Single-Processor Systems
- Multiprocessor Systems; Parallel Processing Systems - Tightly coupled sys
- Multiprocessor Systems; Distributed Systems - Loosely coupled systems
- Real–Time Embedded Systems
- Handheld Systems

2 Operating-System Operations
- Dual-Mode Operation
- Timer

3 Process Management

4 Memory Management

5 Storage Management
- File-System Management
- Mass-Storage Management
- I/O Systems

6 Protection

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure

Mainframe Systems

Batch and Multiprogrammed Systems

Time Sharing

Personnel Computers and Single-Processor Systems

Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems

Multiprocessor Systems; Distributed Systems - Loosely coupled systems

Real–Time Embedded Systems

Handheld Systems

Operating-System Operations

Dual-Mode Operation

Timer

Process Management

Memory Management

Storage Management

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure I

- **The Operating System Zoo**: Mainframe OSs, Server OSs, Multiprocessor OSs, Personal computer OSs, Handheld OSs, Embedded OSs, Sensor node OSs, Real-time OSs, Smart card OSs,...
- Generations:
  - First generation (1945 - 1955), vacuum tubes, plug boards
  - Second generation (1955 - 1965), transistors, batch systems
  - Third generation (1965 - 1980), ICs and multiprogramming
  - Fourth generation (1980 - present), personal computers
  - Next generation ??, personal digital assistants (PDA), information appliances

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Mainframe Systems

- The earliest computers, developed in the 1940s, were programmed in *machine language* and they used front panel switches for input.
- Room-sized metal boxes or frames (See Fig. 1). Now used to distinguish high-end commercial machines.

Figure: An IBM 704 mainframe.

- Single operator/programmer/user runs and debugs interactively. No resource coordination.
- Poor *throughput* (like amount of useful work done per hour) and poor *utilization* (keeping all devices busy). Inefficient use of hardware.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Batch and Multiprogrammed Systems I

- *Batch*. Group if jobs submitted to machine together.
- The operator grouped all of the jobs into various batches with similar characteristics (all the quick jobs might run, then the slower ones, etc.) before running them as one at a time (See Fig. 2).
- User must wait for results until batch collected and submitted.
- *Result*: Improved system throughput and utilization, but lost interactivity.

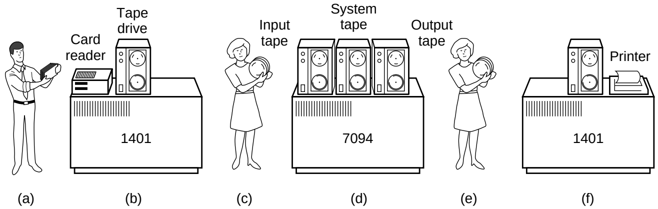
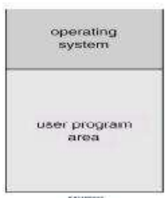


Figure: An early batch system.(a) Programmers bring cards to 1401. (b) 1401 reads batch of jobs onto tape. (c) Operator carries input tape to 7094. (d) 7094 does computing. (e) Operator carries output tape to 1401. (f) 1401 prints output.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Batch and Multiprogrammed Systems II

- *Multiprogramming*. Overlap I/O with execution by providing pool of ready jobs. A number of programs were resident in memory and the CPU could choose which one to run (see Fig. 3right).
- Multiprogramming increases CPU utilization by organizing jobs (code and data) so that the CPU always has one to execute.



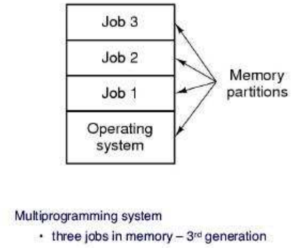


Figure: Left: Memory layout of a simple batch. Right: A multiprogramming system with three jobs in memory.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Batch and Multiprogrammed Systems III

- A single user cannot, in general, keep either the CPU or the I/O devices busy at all times (uniprogramming, see Fig. 4upper).
- One way is the CPU would move onto the next program ready to be run (pure multiprogramming, see Fig. 4lower).
- Improves throughput and utilization. New OS functionalities evolved. Buffering, Direct Memory Access (DMA), interrupt handling, job scheduling policies, memory management and protection. Still not interactive.

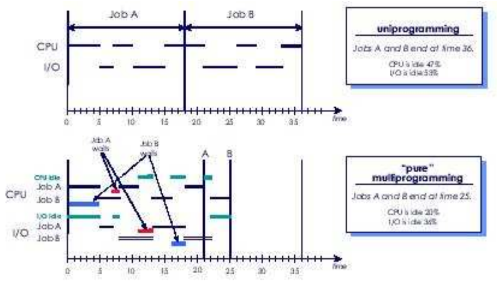


Figure: Job Interleaving

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Time Sharing I

- *Time-Sharing Systems Interactive Computing*. Time sharing (or multitasking) is a logical extension of multiprogramming.
- In time-sharing systems, the CPU executes multiple jobs by switching among them, but the switches occur so frequently that the users can interact with each program while it is running.
- As the system switches rapidly from one user to the next, each user is given the impression that the entire computer system is dedicated to his use, even though it is being shared among many users.
- The CPU switches to the next job that can be run whenever the current job enters a wait state or after the current job has used a standard unit of time.
- A time-shared OS uses *CPU scheduling* and *multiprogramming* to provide each user with a small portion of a time-shared computer.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Time Sharing II

- Time-sharing and multiprogramming require several jobs to be kept simultaneously in memory. Processes are swapped in and out of main memory to the disk.
- In effect, we are now “memory sharing” between competing users (programs). This idea leads to a mechanism called **virtual memory**.
- Virtual memory is a technique that allows the execution of a process that is not completely in memory.
- If several jobs are ready to be brought into memory, and if there is not enough room for all of them, then the system must choose among them. Making this decision is **job scheduling**.
- When the OS selects a job from the job pool, it loads that job into memory for execution. Having several programs in memory at the same time requires some form of **memory management**.
- In addition, if several jobs are ready to run at the same time, the system must choose among them. Making this decision is **CPU scheduling**.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Time Sharing III

- Time-sharing systems must also provide a file system. The file system resides on a collection of disks; hence, disk management must be provided.
- Also, time-sharing systems provide a mechanism for protecting resources from inappropriate use.
- To ensure orderly execution, the system must provide mechanisms for job synchronization and communication, and it may ensure that jobs do not get stuck in a deadlock, forever waiting for one another.
- New OS Functionalities: More complex job scheduling, memory management, concurrency control and synchronization.
- The sensible sharing of resources such as CPU time and memory must be handled by the OS. For this control program to always be in control, we require that it never be blocked from running.
- The OS will lock itself into memory and then control CPU allocation priority in order that it never be blocked from running.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Personnel Computers and Single-Processor Systems I

- Single-user, dedicated. Previously thought as individuals have sole use of computer, do not need advanced CPU utilization, protection features (see Fig. 5).
- Not still true. May run several different types of OS (Windows, Mac OS X, UNIX, Linux) which offer multitasking and virtual memory on PC hardware.

Figure: IBM PC XT.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Personnel Computers and Single-Processor Systems II

- PCs contain a microprocessor in the keyboard to convert the keystrokes into codes to be sent to the CPU.
- A disk-controller microprocessor receives a sequence of requests from the main CPU and implements its own disk queue and scheduling algorithm.
- This arrangement relieves the main CPU of the overhead of disk scheduling.
- The use of special-purpose microprocessors is common and does not turn a single-processor system into a multiprocessor.
- If there is only one general-purpose CPU, then the system is a single-processor system.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems I

- The desire for increased throughput has led to system designs in which multiple streams of processing occurs in parallel.
- Tightly coupled systems have two or more processors in close communication, sharing the computer bus and sometimes the clock, memory, and peripheral devices.
- Communication usually takes place through the shared memory. (see Fig. 6left)

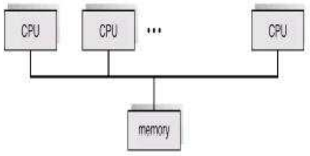


Figure: Left: Tightly coupled system. Right: The Cray-2, the world's fastest computer from 1985 to 1989.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems II

- Multiprocessor systems have three main advantages:
  1. *Increased throughput.* By increasing the number of processors, we expect to get more work done in less time. When multiple processors cooperate on a task, a certain amount of overhead is incurred in keeping all the parts working correctly. This overhead, plus contention for shared resources, lowers the expected gain from additional processors.
  2. *Economy of scale.* Multiprocessor systems can cost less than equivalent multiple single-processor systems, because they can share peripherals, mass storage, and power supplies.
  3. *Increased reliability.* If functions can be distributed properly among several processors, then the failure of one processor will not halt the system, only slow it down. If we have ten processors and one fails, then each of the remaining nine processors can pick up a share of the work of the failed processor. Thus, the entire system runs only 10 percent slower, rather than failing altogether.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems III

- The multiple-processor systems in use today are of two types.
  1. Some systems use *asymmetric multiprocessing*, in which each processor is assigned a specific task. A master processor controls the system. This scheme defines a master-slave relationship.
  2. The most common systems use *symmetric multiprocessing* (SMP), in which each processor performs all tasks within the OS. SMP means that all processors are peers.

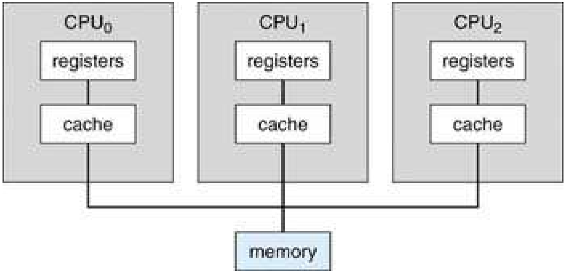


Figure: Symmetric multiprocessing architecture.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems IV

- Virtually all modern OSs-including Windows, Windows XP, Mac OS x, and Linux-now provide support for SMP.
- The difference between symmetric and asymmetric multiprocessing may result from either hardware or software.
- A recent trend in CPU design is to include multiple compute cores on a single chip (see Fig. 8). In essence, these are multiprocessor chips.

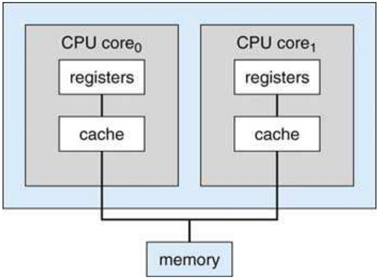


Figure: A Dual-Core Design.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Multiprocessor Systems; Distributed Systems - Loosely coupled systems I

- While these tightly coupled systems require specialized hardware support in order that the CPUs can share the common memory system, another approach is to use a network to join together more conventional systems into what is termed a *distributed system*.
- A distributed system is a collection of physically separate, possibly heterogeneous computer systems that are networked (LAN or WAN) to provide the users with access to the various resources that the system maintains.
- Access to a shared resource increases computation speed, functionality, data availability, and reliability.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Multiprocessor Systems; Distributed Systems - Loosely coupled systems II

- Multicomputers. They do not share memory and clock. May be either client-server or peer-to-peer systems.
- Many of today's systems act as **server systems** to satisfy requests generated by **client systems**. This form of specialized distributed system, called client-server system, has the general structure depicted in Fig. 9left.
- A server running a database that responds to client requests for data is an example of such a system.

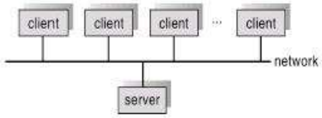


Figure: Left: Client-Server. Right: An example of a computer cluster – this is a Silicon Graphics Cluster-SGI.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Multiprocessor Systems; Distributed Systems - Loosely coupled systems III

- Another example of a client-server system is the file-server system on campus. Here, a central server provides file access to authenticated users at client machines.
- Peer-to-Peer Systems are another form of distributed system in which the participating computer systems are similar.
- Some OSs have taken the concept of networks and distributed systems further than the notion of providing network connectivity.
- A network OS is an OS that provides features such as file sharing across the network and that includes a communication scheme that allows different processes on different computers to exchange messages.
- A distributed OS provides a less autonomous environment: The different OSs communicate closely enough to provide the illusion that only a single OS controls the network.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Real–Time Embedded Systems

- Embedded computers. These devices are found everywhere, from car engines and manufacturing robots to VCRs and microwave ovens. They tend to have very specific tasks.
- Embedded systems almost always run real-time OSs.
- A real-time system is used when rigid time requirements (time critical) have been placed on the operation of a processor or the flow of data; thus, it is often used as a control device in a dedicated application.
  - *hard* real-time (must react in time), the real-time system absolutely must complete critical tasks within a guaranteed time.
    - Contrast this system with a time-sharing system, where it is desirable (but not mandatory) to respond quickly, or a batch system, which may have no time constraints at all.
  - *soft* real-time (deal with failure to react in time), the real-time system can satisfy its performance criteria by running any critical task at a higher priority (of CPU access).
    - Useful in applications (multimedia, virtual reality) requiring advanced operating-system features.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Handheld Systems

- Handheld systems include personal digital assistants (PDAs), such as Palm and Pocket-PCs, and cellular telephones, many of which use special-purpose embedded OSs.
- Hand-held systems must also deal with limited resources although their screens have recently become more substantial. Because of their size, most handheld devices have a small amount of memory, slow processors, and small display screens.
- Generally, the limitations in the functionality of PDAs are balanced by their <u>convenience</u> and <u>portability</u>.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Operating-System Operations

- Modern OSs are **interrupt driven**.
- If there are no processes to execute, no I/O devices to service, and no users to whom to respond, an OS will sit quietly, waiting for something to happen.
- Events are almost always signaled by the occurrence of an interrupt or a **trap**.
    - A trap (or an exception) is a software-generated interrupt caused either by an error (for example, division by zero or invalid memory access) or by a specific request from a user program that an operating-system service be performed.
- With sharing, many processes could be adversely affected by a bug in one program. For example, if a process gets stuck in an infinite loop, this loop could prevent the correct operation of many other processes.
- More subtle errors can occur in a multiprogramming system, where one erroneous program might modify another program, the data of another program, or even the OS itself.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Dual-Mode Operation I

- At the very least, we need two separate modes of operation: user mode and kernel mode (also called supervisor mode, system mode, or privileged mode).
- A bit, called the mode bit is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1).
- User mode: a *subset* of instructions. Limited set of hardware and memory available.
  - *I/O protection*, all I/O operations are privileged; so user programs can only access I/O by sending a request to the (controlling) OS.
  - *Memory protection*, base/limit registers (in early systems), memory management unit, (MMU, in modern systems); so user programs can only access the memory that the OS has allocated.
  - *CPU control*, timer (alarm clock), context switch; so user programs can only read the time of day, and can only have as much CPU time as the OS allocates.
- The dual mode of operation provides us with the means for protecting the OS from errant users-and errant users from one another.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Dual-Mode Operation II

- If an attempt is made to execute a **privileged instruction** in user mode, the hardware does not execute the instruction but rather treats it as illegal and traps it to the OS.
- When a user application requests a service from the OS (via a system call, treated by the hardware as a software interrupt), it must transition from user to kernel mode to fulfil the request (see Fig. 10).
- Control passes through the interrupt vector to a service routine in the OS, and the mode bit is set to kernel mode.

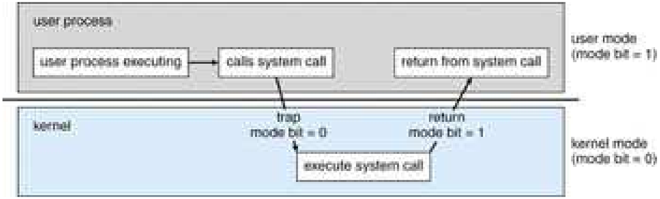


Figure: Transition from user to kernel mode.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Timer

- We must ensure that the OS maintains control over the CPU. We must prevent a user program from getting stuck in an infinite loop or not calling system services and never returning control to the OS.
- To accomplish this goal, we can use a **timer**. A timer can be set to interrupt the computer after a specified period.
- Thus, we can use the timer to prevent a user program from running too long. A simple technique is to initialize a counter with the amount of time that a program is allowed to run.
- When the counter becomes negative, the OS terminates the program for exceeding the assigned time limit.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Process Management I

- A key concept in all OSs is the **process**. A program does nothing unless its instructions are executed by a CPU.
- A process needs certain resources-including CPU time, memory, files, and I/O devices-to accomplish its task.
- When the process terminates, the OS will reclaim any reusable resources.
- Associated with each process is its **address space**, a list of memory locations from 0 to some maximum, which the process can read and write (executable program, program's data and its stack).
- Also associated with each process is a set of resources, commonly including registers (program counter, stack pointer, ..), a list of open files, outstanding alarms, lists of related processes, and all the other information needed to run the program.
- All these information about each process is stored in a table called the **process table**, which is an array (or linked list) of structures, one for each process currently in existence.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Process Management II

- if a process can create one or more other processes (referred to as **child processes**) and these processes in turn can create child processes, we quickly arrive at the process tree structure of Fig. 11.

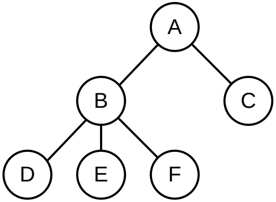


Figure: A process tree. Process A created two child processes, B and C. Process B created three child processes, D, E, and F.

- Related processes that are cooperating to get some job done often need to communicate with one another and synchronize their activities. This communication is called **interprocess communication** (IPC).

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Process Management II

- A single-threaded process has one **program counter** specifying the next instruction to execute. The execution of such a process must be sequential.
- A multithreaded process has multiple program counters, each pointing to the next instruction to execute for a given thread.
- The OS is responsible for the following activities in connection with process management:
  - Creating and deleting both user and system processes,
  - Suspending and resuming processes,
  - Providing mechanisms for process synchronization,
  - Providing mechanisms for process communication,
  - Providing mechanisms for deadlock handling.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Memory Management

- In a very simple OS, only one program at a time is in the memory. To run second program, the first one has to be removed and the second one placed in memory. **Single Tasking System**.
- More sophisticated OSs allow multiple programs to be in memory at the same time. To keep them from interfering with one another (and with OS), some kind of protection mechanism is needed. **Multi Tasking System**.
- For a program to be executed, it must be mapped to <u>absolute addresses</u> and loaded into memory.
- The OS is responsible for the following activities in connection with memory management:
    - Keeping track of which parts of memory are currently being used and by whom,
    - Deciding which processes (or parts thereof) and data to move into and out of memory,
    - Allocating and deallocating memory space as needed.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

File-System Management I

- A major function of the OS is to hide the peculiarities of the disks and other I/O devices and present the user/programmer with a nice, clean abstract model of device-independent files.
- To provide a place to keep files, most OS have the concept of a **directory** as a way of grouping files together.
- Every file within the directory hierarchy can be specified by giving its **path name** from the top of the directory hierarchy, the **root directory**.
  - Absolute path
  - Relative path
- Before a file can be read or written, it must be opened, at which time the permissions are checked. If the access is permitted, the system returns a small integer called a **file descriptor** to use a subsequent operations.
- Each medium is controlled by a device, such as a disk drive or tape drive, that also has its own unique characteristics. These properties include access speed, capacity, data-transfer rate, and access method (sequential or random).

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

File-System Management II

- Another important concept in UNIX is the **special file**. Special files are provided in order to make I/O devices look like files.
- That way, they can be read and written using the same system calls as are used for reading and writing files.
  - **Block special files** are used to model devices that consist of a collection of randomly addressable blocks, such as disks.
  - **Character special files** are used to model printers, modems, and other devices that accept or output character stream.
  - */dev* is the directory. */dev/lp* might be the printer (once called the line printer).
- The OS is responsible for the following activities in connection with file management:
  - Creating and deleting files,
  - Creating and deleting directories to organize files,
  - Supporting primitives for manipulating files and directories,
  - Mapping files onto secondary storage,
  - Backing up files on stable (nonvolatile) storage media.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Mass-Storage Management

- The proper management of disk storage is of central importance to a computer system.
- The OS is responsible for the following activities in connection with disk management:
  - Free-space management,
  - Storage allocation,
  - Disk scheduling,
- The entire speed of operation of a computer may hinge on the speeds of the disk subsystem and of the algorithms that manipulate that subsystem.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

I/O Systems

- One of the purposes of an OS is to hide the peculiarities of specific hardware devices from the user.
- For example, in UNIX, the peculiarities of I/O devices (including keyboards, monitors, printers, and so on) are hidden from the bulk of the OS itself by the I/O subsystem.
- The I/O subsystem consists of several components:
  - A memory-management component that includes buffering, caching, and spooling,
  - A general device-driver interface, that is, applies to many or all I/O devices equally well.
  - Drivers for specific hardware devices.
- Only the device driver knows the peculiarities of the specific device to which it is assigned.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Protection I

- **File Protection.** It is up to the OS to manage the system security so that files are only accessible to authorized users.
- Files in UNIX are protected by assigning each one a 9-bit binary protection code.
  - Three bit fields, one for owner, one for other members of the owner's group, and one for everyone else.
  - Each field has a bit for **read** access, a bit for **write** access, and a bit for **execute** access.
  - $(d)rwxr-x--x$
- **OS Protection**. OS must protect itself from users; reserved memory only accessible by OS.
- The OS is responsible for allocating access to memory space and CPU time and peripherals etc., and it will control dedicated hardware facilities:
  - The memory controller, control to detect and prevent unauthorized access.
  - A timer will also be under OS control to manage CPU time allocation to programs competing for resources.

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management

Protection II

- **User Protection**. OS may protect users from another user. A fundamental requirement of multiple users of a shared computer system is that they do not interfere with each other.
- This gives rise to the need for separation of the programs in terms of their resource use and access:
    - If one program attempts to access main memory allocated to some other program, the access should be denied and an exception raised.
    - If one program attempts to gain a larger proportion of the shared CPU time, this should be prevented.
- One approach to implementing resource allocation is to have at least two modes of CPU operation (see 1)

Evolution of Operating Systems, Computer-System Architecture and Operating-System Structure
Mainframe Systems
Batch and Multiprogrammed Systems
Time Sharing
Personnel Computers and Single-Processor Systems
Multiprocessor Systems; Parallel Processing Systems - Tightly coupled systems
Multiprocessor Systems; Distributed Systems - Loosely coupled systems
Real–Time Embedded Systems
Handheld Systems
Operating-System Operations
Dual-Mode Operation
Timer
Process Management
Memory Management
Storage Management