An Introduction to Computer Science

Jingde Cheng

Southern University of Science and Technology

An Introduction to Computer Science

- Computer Science: What Is It and Why Study It?
- Computation: What Is It and Why Study It?
- Computability
- Computational Complexity (CS101A class only)
- Algorithms
- Data, Information, and Knowledge, and Their Representations
- Data Storage
- Computer Architecture
- Data Manipulation in Computer Systems
- Programming Languages and Compilers
- Operating Systems
- System Software and Application Software
- Software Engineering (CS101A class only)
- Knowledge Engineering and Artificial Intelligence (CS101A class only)
- Information Security Engineering (CS101A class only)

Operating System: What Is It and Why Study It?

Operating System: What Is It and Why Study It ?

Operating System: What Is It and Why Study It?

♣ Operating system [A Dictionary of Computer Science (7th Edition), OUP, 2016]

- "The set of software products that jointly controls the system resources and and the processes using these resources on a computer system."

♣ Operating system [IEEE Standard Computer Dictionary]

- "A collection of software, firmware, and hardware elements that control the execution of computer programs and provides such services as computer resource allocation, job control, input/output control, and file management in a computer system."

Operating System: What Is It and Why Study It?

♣ Operating system

- An *operating system* is the software system that controls the overall operation of a computer.
- It provides the means by which a user can store and retrieve files, provides the interface by which a user can request the execution of programs, and provides the environment necessary to execute the programs requested.

♣ The necessity of operating system

- The high efficiency of CPU vs. lower efficiency of I/O devices.
- In order to provide various services for various users automatically, a control and management system (software) is indispensable.
- Without operating system, a "bare" computer is vary difficult to use, if not impossible.

Operating System [F-CS-18]

7.1.1 Operating system

An **operating system** is complex, so it is difficult to give a simple universal definition. Instead, here are some common definitions:

- An operating system is an interface between the hardware of a computer and the user (programs or humans).
- An operating system is a program (or a set of programs) that facilitates the execution of other programs.
- An operating system acts as a general manager supervising the activity of each component in the computer system. As a general manager, the operating system checks that hardware and software resources are used efficiently, and when there is a conflict in using a resource, the operating system mediates to solve it.

An operating system is an interface between the hardware of a computer and the user (programs or humans) that facilitates the execution of other programs and the access to hardware and software resources.

Two major design goals of an operating system are:

- Efficient use of hardware.
- Easy use of resources.

Operating System [DL-CS-16] [ECL-OS-10]

Operating system System software that manages computer resources and provides an interface for system interaction

Operating System (or just **System**). Although we can give different definitions based on the different views of an OS, the following informal definition is a good starting point: The OS is a collection of one or more software modules that manages and controls the resources of a computer or other computing or electronic device, and gives users and programs an interface to utilize these resources. The managed resources include memory, processor, files, input or output devices, and so on.

Early Computer Systems: Batch and Interactive Processing

Figure 3.1 Batch processing

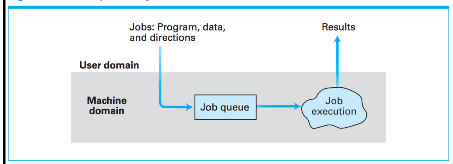
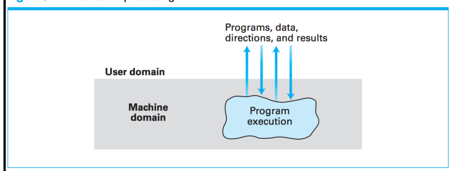


Figure 3.2 Interactive processing



Early Computer Systems: Batch Processing [DL-CS-16]

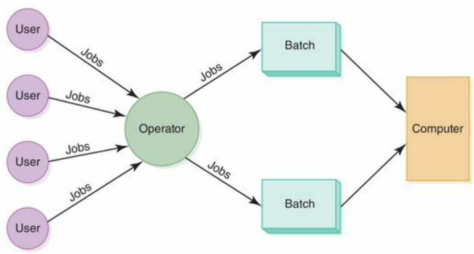


FIGURE 10.3 In early systems, human operators would organize jobs into batches

A Simple Personal Computer System [TB-OS-15]

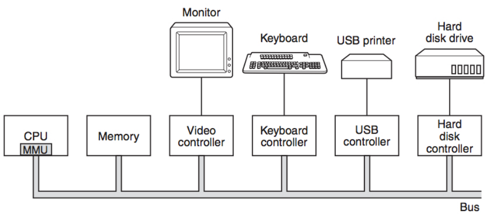


Figure 1-6. Some of the components of a simple personal computer.

A Modern Computer System [SGG-OS-11]

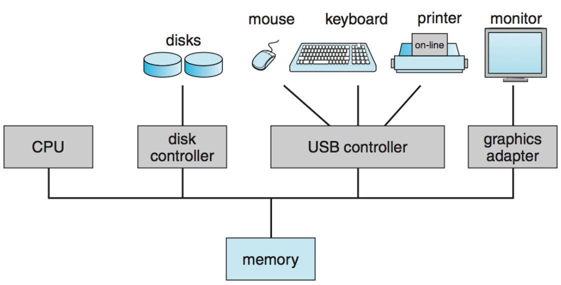


Figure 1.2 A modern computer system.

Abstract View of the Components of a Computer System [SGG-OS-11]

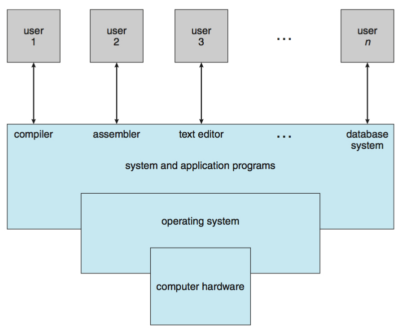


Figure 1.1 Abstract view of the components of a computer system.

Using a Computer System: The Cursor Tracking Mouse Motion [ECL-OS-10]

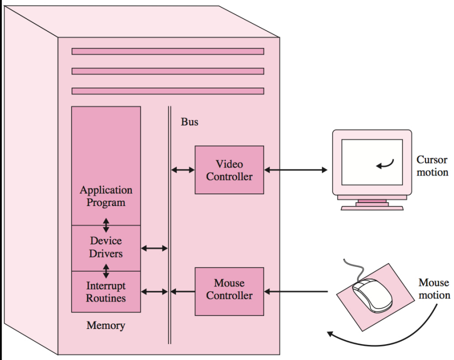
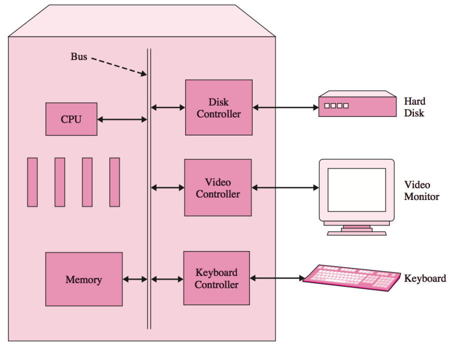


FIGURE 1.1 The cursor tracking mouse motion.

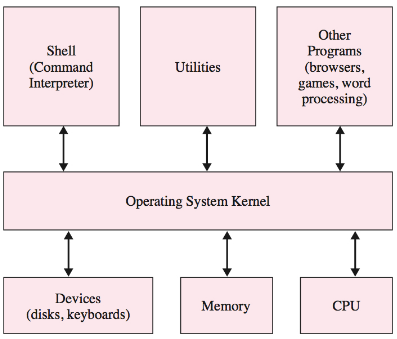
Hardware of a Small Personal Computer System [ECL-OS-10]

FIGURE 1.2 Hardware: A very simplistic view of a small personal computer.
(Note: This picture is *too* simple. In reality there are often multiple buses, say between video and memory. We will get to more detailed pictures in the Appendix.)



OS Software and Computer Hardware [ECL-OS-10]

FIGURE 1.3 A simplistic view of the OS software in relationship to hardware.



Where the OS Fits in [TB-OS-15]

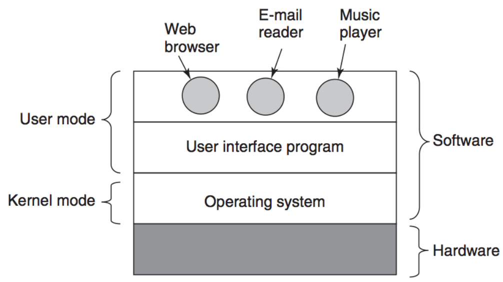


Figure 1-1. Where the operating system fits in.

OS: Interacts with Many Aspects of a Computer System [DL-CS-16]

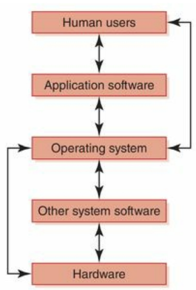
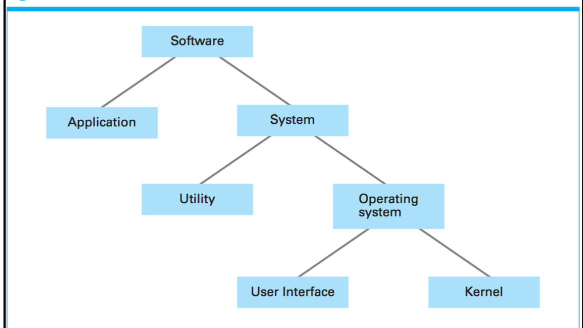


FIGURE 10.1 An operating system interacts with many aspects of a computer system.

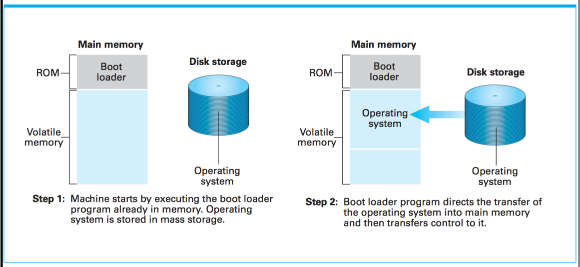
Software Classification

Figure 3.3 Software classification



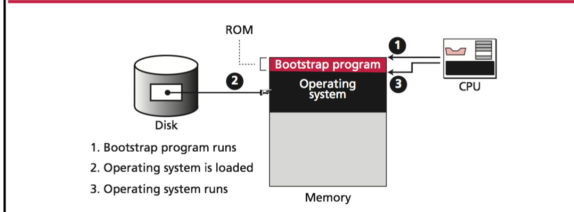
OS: The Booting Process

Figure 3.5 The booting process



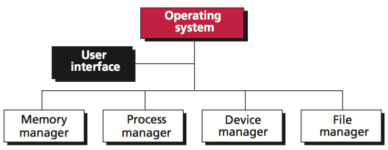
OS: The Bootstrap Process [F-CS-18]

Figure 7.2 *The bootstrap process*



OS: Components [F-CS-18]

Figure 7.3 *Components of an operating system*



OS: A Layered View of an OS [SGG-OS-11]

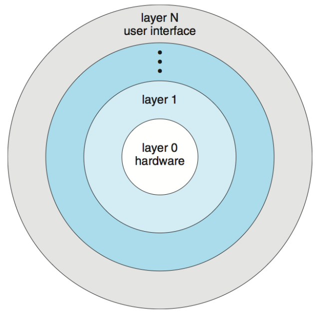


Figure 2.14 A layered operating system.

OS: A Layered View of an OS [ECL-OS-10]

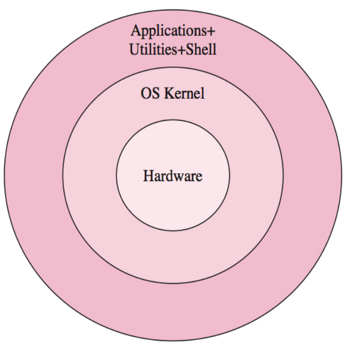


FIGURE 1.4 A layered view of an OS.

OS: The PC Model of an OS [ECL-OS-10]

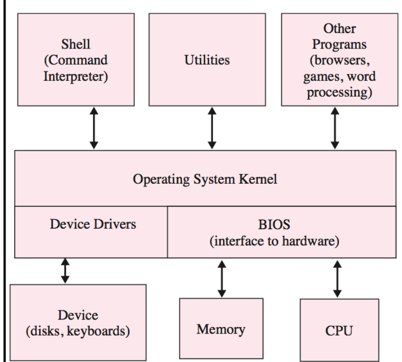


FIGURE 1.5 The PC (small system) model of an OS.

OS: The Major OS Modules [ECL-OS-10]

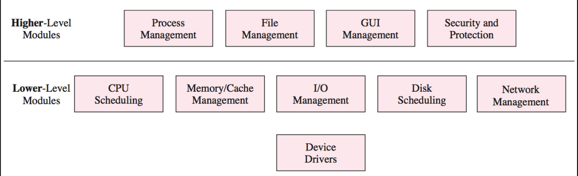
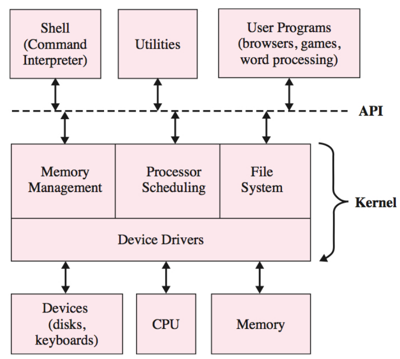


FIGURE 2.1 The major OS modules.

OS: Layered Model of an OS [ECL-OS-10]

FIGURE 2.4
Layered model of an Operating System.



OS: Microkernel Model of an OS [ECL-OS-10]

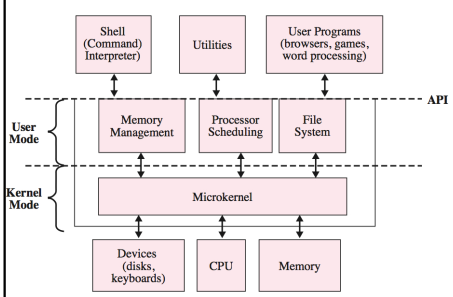


FIGURE 2.5
Microkernel model of an Operating System.

OS: A View of OS Services [SGG-OS-11]

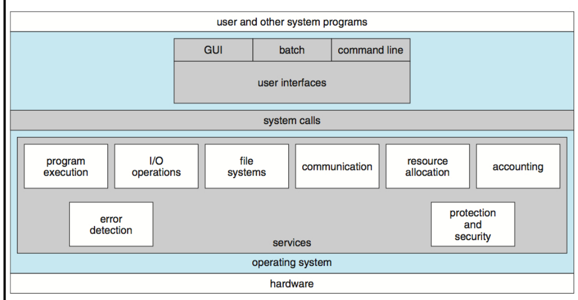


Figure 2.1 A view of operating system services.

OS: Concerns about What the OS Should Provide for [ECL-OS-10]

TABLE 1.1 Concerns of Various User Classes

End Users	Easy to use and learn Adapts to user's style of doing things Lively response to input Provides lots of visual cues Free of unpleasant surprises (e.g., deleting a file without warning) Uniform ways to do the same thing (e.g., moving an icon or scrolling down a window—in different places) Alternative ways to do one thing (e.g., some users like to use the mouse, others like to use the keyboard)
Application Programmers	Easy to access low-level OS calls by programs (e.g., reading keystrokes, drawing to the screen, getting mouse position) Provide a consistent programmer view of the system Easy to use higher-level OS facilities and services (e.g., creating new windows, or reading from and writing to the network) Portability to other platforms
Systems Programmers	Easy to create correct programs Easy to debug incorrect programs Easy to maintain programs Easy to expand existing programs
System Managers and Administrators	Easy addition or removal of devices such as disks, scanners, multimedia accessories, and network connections Provide OS security services to protect the users, system, and data files Easy to upgrade to new OS versions Easy to create and manage user accounts Average response is good and predictable System is affordable

The User Interface of OS: Shells and GUI

♣ **The user interface**

- **In order to perform the actions requested by the computer's users, an OS must be able to communicate with those users.**
- **The portion of an OS that handles this communication is often called the *user interface*.**

♣ **Shells**

- ***Shells* communicated with users through textual messages using a keyboard and monitor screen.**

♣ **GUI**

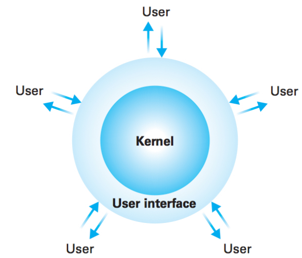
- **In a *graphical user interface (GUI)*, objects are represented pictorially on the display as icons.**
- **A GUI allows users to issue commands by using one of several common input devices.**

The User Interface of OS Acts as an Intermediary

- The user interface acts as an intermediary
  - An OS's user interface merely acts as an intermediary between the computer's user and the real heart of the OS.
  - This distinction between the user interface and the internal parts of the OS is emphasized by the fact that some OSs allow a user to select among different interfaces to obtain the most comfortable interaction for that particular user.
- Examples
  - Users of the UNIX OS can select among a variety of shells including the Bourne shell, the C shell, and the Korn shell, as well as a GUI called X11.
  - The DOS cmd.exe shell can still be found as a utility program in the latest versions of Windows OS.
  - Apple's OS X retains a Terminal utility shell that hearkens back to that system's UNIX ancestors.

The User Interface of OS Acts as an Intermediary

Figure 3.4 The user interface acts as an intermediary between users and the operating system's kernel



The Kernel of OS: File Manager

- The kernel
  - An OS's *kernel* contains those software components that perform the very basic functions required by the computer installation.
- File manager
  - The *file manager* coordinates the use of the machine's mass storage facilities.
  - More precisely, the file manager maintains records of all the files stored in mass storage, including where each file is located, which users are allowed to access the various files, and which portions of mass storage are available for new files or extensions to existing files.

The Kernel of OS: File Manager

- File manager
  - Most file managers allow files to be grouped into a bundle called a *directory* or *folder*.
  - This approach allows a user to organize files according to their purposes by placing related files in the same folder.
  - By allowing directories to contain other directories, called *subdirectories* or *subfolder*, a hierarchical organization can be constructed.
  - A chain of directories within directories is called a *directory path*.
  - Any access to a file by other software units is obtained at the discretion of the file manager.

The Kernel of OS: Device Drivers and Memory Manager

- Device drivers
  - *Device drivers* are the software units that communicate with the controllers to carry out operations on the peripheral devices attached to the machine.
  - Each device driver is uniquely designed for its particular type of device and translates generic requests into the more technical steps required by the device assigned to that driver.
- Memory manager
  - *Memory manager* coordinates the machine's use of main memory.
  - In an environment in which a computer is asked to perform only one task at a time, the program for performing the current task is placed at a predetermined location in main memory, executed, and then replaced by the program for performing the next task.

The Kernel of OS: Memory Manager

- Memory manager
  - In multi-user or multi-tasking environments in which the computer is asked to address many needs at the same time, many programs and blocks of data must reside in main memory concurrently.
  - Thus, the memory manager must find and assign memory space for these needs and ensure that the actions of each program are restricted to the program's allotted space.
  - Moreover, as the needs of different activities come and go, the memory manager must keep track of those memory areas no longer occupied.

The Kernel of OS: Scheduler and Dispatcher

- **Scheduler**
  - **In a multi-user or multi-tasking system, the scheduler determines which activities are to be considered for execution.**
- **Dispatcher**
  - **In a multi-user or multi-tasking system, the dispatcher controls the allocation of time to these activities.**

Process: What Is It ?

Process: What Is It ?

The Concept of a Process: Process

- **The distinction between a program and its execution**
  - **One of the most fundamental concepts of modern OSs is the distinction between the instructions of a program and the activity of executing a program.**
  - **The former is a static set of directions (instructions), whereas the latter is a dynamic activity whose properties change as time progresses.**
- **Process**
  - **The activity of executing a program under the control of the OS is known as a *process*.**
  - **Associated with a process is the current status of the activity, called the *process state*.**

The Concept of a Process: Process State

- **Process state**
  - **The process state includes the current position in the program being executed (i.e., the value of the program counter PC) as well as the values in the other CPU registers and the associated memory cells.**
  - **The process state is a snapshot of the machine at a particular time.**
  - **At different times during the execution of a program (at different times in a process) different snapshots (different process states) will be observed.**

The Concept of a Process [DL-CS-16]

Process The dynamic representation of a program during execution

Process management The act of keeping track of information for active processes

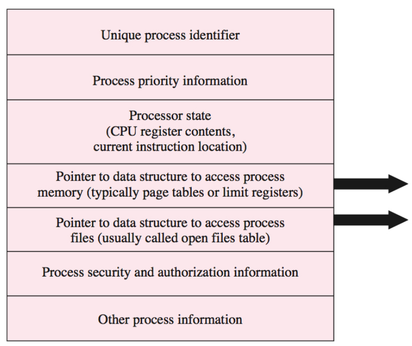
Process states The conceptual stages through which a process moves as it is managed by the operating system

Process control block (PCB) The data structure used by the operating system to manage information about a process

Context switch The exchange of register information that occurs when one process is removed from the CPU and another takes its place

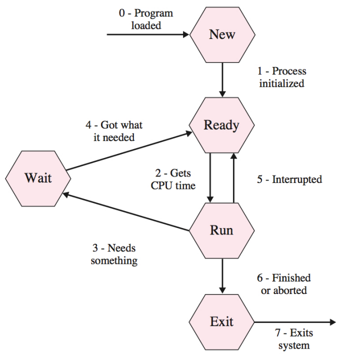
The Concept of a Process Control Block [ECL-OS-10]

FIGURE 2.3 Information the OS maintains in a process control block.



Process States and Transitions [ECL-OS-10]

FIGURE 2.2
Simplified diagram of process states and transitions.



The Process Life Cycle [DL-CS-16]

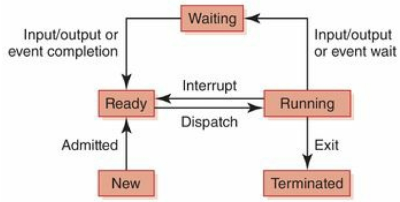


FIGURE 10.9 The process life cycle

The Concept of a Process: Process Management

- **Process management**
  - **Typical *time-sharing/multi-tasking* computers are running many processes, all competing for the computer's resources.**
  - **It is the task of the OS to manage these processes so that each process has the resources (peripheral devices, space in main memory, access to files, and access to a CPU) that it needs, that independent processes do not interfere with one another, and that processes that need to exchange data are able to do so.**

Process Administration

- **The scheduler and dispatcher**
  - **The tasks associated with coordinating the execution of processes are handled by the scheduler and dispatcher within the OS's kernel.**
- **The task of scheduler**
  - **The scheduler maintains a record of the processes present in the computer system, introduces new processes to a pool of processes, and removes completed processes from the pool.**
  - **When a user requests the execution of an application, it is the scheduler that adds the execution of that application to the pool of current processes.**
  - **To keep track of all the processes, the scheduler maintains a block of information in main memory called the *process table*.**

Process Administration: The Task of Scheduler

- **The task of scheduler**
  - **Each time the execution of a program is requested, the scheduler creates a new entry for that process in the process table.**
  - **This entry contains such data/information as the memory area assigned to the process (obtained from the memory manager), the priority of the process, and whether the process is ready or waiting.**
  - **A process is *ready* if it is in a state in which its progress can continue; it is *waiting* if its progress is currently delayed until some external event occurs, such as the completion of a mass storage operation, the pressing of a key at the keyboard, or the arrival of a message from another process.**

Process Administration: The Task of Dispatcher

- **The task of dispatcher**
  - **The dispatcher manages the execution of the scheduled processes.**
  - **In a time-sharing/multi-tasking system this task is accomplished by *multi-programming*; that is, dividing time into short segments, each called a *time slice* (typically measured in milliseconds or microseconds), and then switching the CPU's attention among the processes as each is allowed to execute for one time slice.**
  - **The procedure of changing from one process to another is called a *process switch* (or a *context switch*).**

Process Administration: Interrupt Handling

♣ **Interrupt handling**

- **Each time the dispatcher awards a time slice to a process, it initiates a timer circuit that will indicate the end of the slice by generating a signal called an *interrupt*.**
- **When the CPU receives an interrupt signal, it completes its current machine cycle, saves its position in the current process and begins executing a program, called an *interrupt handler*, which is stored at a predetermined location in main memory.**
- **This interrupt handler is a part of the dispatcher, and it describes how the dispatcher should respond to the interrupt signal.**

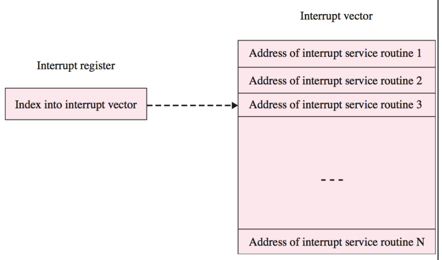
Process Administration: Interrupt Handling

♣ **Interrupt handling**

- **Thus, the effect of the interrupt signal is to preempt the current process and transfer control back to the dispatcher.**
- **At this point, the dispatcher selects the process from the process table that has the highest priority among the ready processes (as determined by the scheduler), restarts the timer circuit, and allows the selected process to begin its time slice.**
- **The key for the success of a multi-programming system is the ability to stop, and later restart, a process.**

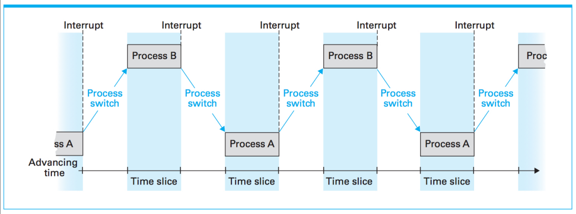
Interrupt Handling [ECL-OS-10]

FIGURE 2.6
An interrupt vector for handling interrupts.



Simplified Diagram of Process States and Transitions

Figure 3.6 Multiprogramming between process A and process B



Handling Competition Among Processes

♣ **Resource allocation**

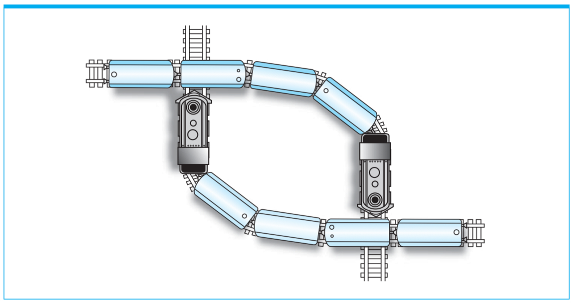
- **An important task of an operating system is the allocation of the machine's resources to the processes in the system.**
- **Here we are using the term resource in a broad sense, including the machine's peripheral devices as well as features within the machine itself.**

♣ **Resource competition**

- **Example: Deadlock (two or more processes are blocked from progressing because each is waiting for a resource that is allocated to another)**
- **To construct reliable operating systems, we must develop algorithms that cover every possible contingency, regardless of how minuscule it may appear.**

Deadlock: An Example

Figure 3.7 A deadlock resulting from competition for nonshareable railroad intersections

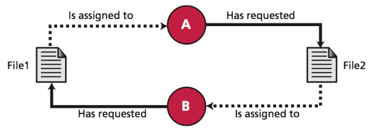


Deadlock [F-CS-18]

Deadlock

Instead of a formal definition of **deadlock**, we give an example. Assume that there are two processes, A and B. Process A is holding a file, File1 (that is, File1 is assigned to A) and cannot release it until it acquires another file, File2 (that is, A has requested File2). Process B is holding File2 (that is, File2 is assigned to B) and cannot release it until it has File1 (that is, B has requested File1). Files in most systems are not sharable—when in use by one process, a file cannot be used by another process. If there is no provision in this situation to force a process to release a file, deadlock is created (Figure 7.16).

Figure 7.16 *Deadlock*



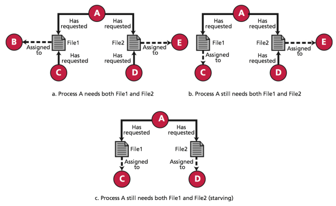
Starvation [F-CS-18]

Starvation

Starvation is the opposite of deadlock. It can happen when the operating system puts too many resource restrictions on a process. For example, imagine an operating system that specifies that a process must have possession of its required resources before it can be run.

In Figure 7.18, imagine that process A needs two files, File1 and File2. File1 is being used by process B and File2 is being used by process E. Process B terminates first and releases File1. Process A cannot be started, because File2 is still not available. At this moment, process C, which needs only File1, is allowed to run. Now process E terminates and releases File2, but process A still cannot run because File1 is unavailable.

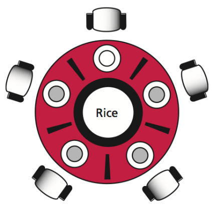
Figure 7.18 *Starvation*



Dining Philosophers Problem: Deadlock, Livelock, and Starvation

Dining Philosophers Problem [F-CS-18]

Figure 7.19 *The dining philosophers problem*



A classic starvation problem is the one introduced by Edsger Dijkstra. Five philosophers are sitting at a round table (Figure 7.19). Each philosopher needs two chopsticks to eat a bowl of rice. However, one or both chopsticks could be used by a neighbor. A philosopher could starve if two chopsticks are not available at the same time.

Necessary Conditions for Deadlock Occurrence [F-CS-18]

- **Four necessary conditions for deadlock occurrence**
  - **(1) Mutual exclusion. Only one process can hold a resource.**
  - **(2) Resource holding. A process holds a resource even though it cannot use it until other resources are available.**
  - **(3) No preemption. The operating system cannot temporarily reallocate a resource.**
  - **(4) Circular waiting. Some processes and resources involved form a loop (waiting cycle).**
- **Notes**
  - **All four conditions are required for deadlock to occur.**
  - **These conditions are only necessary but not sufficient to cause deadlock of themselves -- they must be present for deadlock, but they might not be enough to cause it (there may be enough number of resources).**

Memory Management: What Is It ?

Memory Management: What Is It ?

Memory Management [DL-CS-16]

Multiprogramming The technique of keeping multiple programs in main memory at the same time, competing for the CPU

Memory management The act of keeping track of how and where programs are loaded in main memory

Logical address A reference to a stored value relative to the program making the reference

Physical address An actual address in the main memory device

Address binding The mapping from a logical address to a physical address

Memory Management [DL-CS-16]

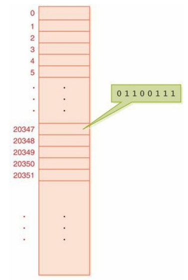


FIGURE 10.4 Memory is a continuous set of bits referenced by specific addresses

FIGURE 10.5 Main memory divided into two sections

Memory Management [DL-CS-16]

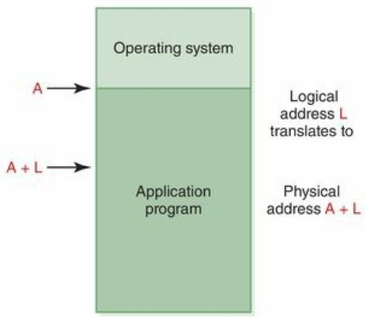


FIGURE 10.6 Binding a logical address to a physical address

Memory Management [DL-CS-16]

Single contiguous memory management The approach to memory management in which a program is loaded into one continuous area of memory

Fixed-partition technique The memory management technique in which memory is divided into a specific number of partitions into which programs are loaded

Dynamic-partition technique The memory management technique in which memory is divided into partitions as needed to accommodate programs

Base register A register that holds the beginning address of the current partition

Bounds register A register that holds the length of the current partition

Memory Management [DL-CS-16]

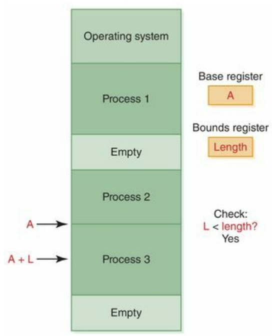


FIGURE 10.7 Address resolution in partition memory management

File System: What Is It ?

File System: What Is It ?

File Systems [DL-CS-16]

File A named collection of data, used for organizing secondary memory

File system The operating system's logical view of the files it manages

Directory A named group of files

Text file A file that contains characters

Binary file A file that contains data in a specific format, requiring a special interpretation of its bits

File type The specific kind of information contained in a file, such as a Java program or a Microsoft Word document

File extension Part of a file name that indicates the file type

File Types [DL-CS-16]

Extensions	File type
txt	text data file
mp3, au, wav	audio file
gif, tiff, jpg	image file
doc, wp3	word processing document
java, c, cpp	program source files

FIGURE 11.1 Some common file types and their extensions

File Systems: Files [DL-CS-16]

- **Files**
  - **From the user's point of view, a *file* is the smallest amount of data that can be written to secondary memory.**
  - **Organizing everything into files presents a uniform view for data storage.**
- **Contents of files**
  - **A file may be considered a sequence of bits, bytes, lines, or records, depending on how you look at it.**
  - **As with any data in memory, you have to apply an *interpretation* to the bits stored in a file before they have meaning.**
  - **The creator of a file decides how the data in a file is organized, and any users of the file must understand that organization .**

File Systems: Text Files and Binary Files [DL-CS-16]

- **Text files and binary files**
  - **All files can be broadly classified as either text files or binary files.**
  - **In a *text file*, the bytes of data are organized as characters from the ASCII or Unicode character sets.**
  - **A *binary file* requires a specific interpretation of the bits based on the data in the file.**
- **"text file" and "binary file"**
  - **The terms "text file" and "binary file" are somewhat misleading. They seem to imply that the data in a text file is not stored as binary data.**
  - **Ultimately, however, all data on a computer is stored as binary digits. These terms refer to how those bits are formatted: as chunks of 8 or 16 bits, interpreted as characters, or in some other special format.**

File Systems: File Types [DL-CS-16]

- **File types**
  - **Most files, whether they are in text or binary format, contain a specific type of data. The kind of data contained in a document is called the *file type*.**
  - **Most operating systems recognize a list of specific file types.**
- **File extension**
  - **A common mechanism for specifying a file type is to indicate the type as part of the file's name.**
  - **File names are often separated, usually by a period, into two parts: the main name and the *file extension*. The extension indicates the type of the file.**
  - **File types allow the operating system to operate on the file in ways that make sense for that file. They also usually make life easier for the user.**

File Systems: File Operations [DL-CS-16]

- **With the help of the operating system, you might perform any of several operations to and with a file:**
  - **Create a file**
  - **Delete a file**
  - **Open a file**
  - **Close a file**
  - **Read data from a file**
  - **Write data to a file**
  - **Reposition the current file pointer in a file**
  - **Append data to the end of a file**
  - **Truncate a file (delete its contents)**
  - **Rename a file**
  - **Copy a file**

File Systems: File Access [DL-CS-16]

File access

- The data in a file can be accessed in several different ways.

Sequential file access

- The most common access technique, and the simplest to implement, is *sequential file access*, which views the file as a linear structure.
- It requires that the data in the file be processed in order.
- Read and write operations move the current file pointer according to the amount of data that is read or written.
- Some systems allow the file pointer to be reset to the beginning of the file and/or to skip forward or backward by a certain number of records.

File Systems: File Access [DL-CS-16]

Direct file access

- Files with *direct file access* are conceptually divided into numbered logical records.
- Direct access allows the user to set the file pointer to any particular record by specifying the record number.
- Therefore, the user can read and write records in any particular order desired.

Sequential file access The technique in which data in a file is accessed in a linear fashion

Direct file access The technique in which data in a file is accessed directly by specifying logical record numbers

File Systems: File Access [DL-CS-16]

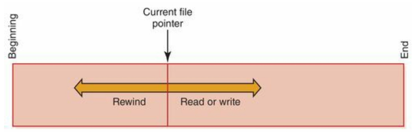


FIGURE 11.2 Sequential file access

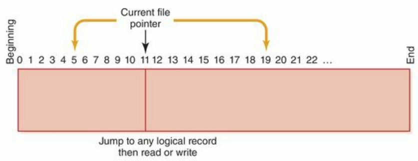


FIGURE 11.3 Direct file access

File Systems: File Protection [DL-CS-16]

File protection

- In multi-user systems, *file protection* is of primary importance. That is, we don't want one user to be able to access another user's files unless such access is specifically allowed.
- It is the operating system's responsibility to ensure valid file access. Different operating systems administer their file protection in different ways.

File protection mechanism

- A *file protection mechanism* determines who can use a file and for what general purpose.

File Systems: File Protection [DL-CS-16]

File protection in UNIX

- A file's protection settings in the UNIX operating system are divided into three categories: Owner, Group, and World.
- Under each category you can determine whether the file can be read, written, and/or executed.
- Under this mechanism, if you can write to a file, you can also delete the file.

	Read	Write/Delete	Execute
Owner	Yes	Yes	No
Group	Yes	No	No
World	No	No	No

File Systems: Directories [DL-CS-16]

Directories

- A *directory* is a named collection of files.
- It is a way to group files so that you can organize them in a logical manner.
- The operating system must carefully keep track of directories and the files they contain.

Directory files

- A directory, in most operating systems, is represented as a file. The *directory file* contains data about the all files in the directory.
- For any given file, the directory file contains the file name, the file type, the address on disk where the file is stored, the current size of the file, the protection data, and data describing when the file was created and when it was last modified.

File Systems: Directories [DL-CS-16]

♣ **Directory trees**

- **A directory of files can be contained within another directory.**
- **The directory containing another directory is usually called the *parent directory*, and the one inside is called a *subdirectory*.**
- **To visualize hierarchy, a file system is often viewed as a *directory tree*, showing directories and files within other directories.**
- **The directory at the highest level is called the *root directory*.**

Directory tree A structure showing the nested directory organization of the file system

Root directory The topmost directory, in which all others are contained

File Systems: Directory Tree [TB-OS-15]

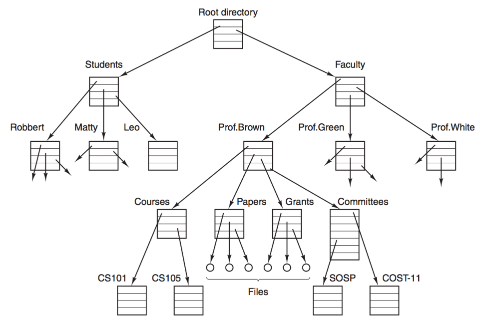


Figure 1-14. A file system for a university department.

File Systems: Windows Directory Tree [DL-CS-16]

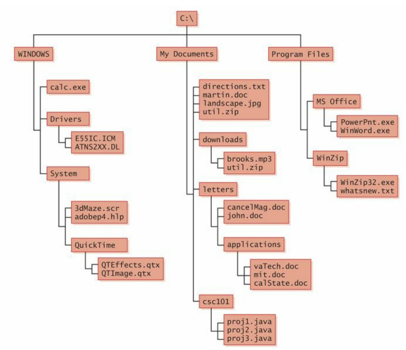


FIGURE 11.4. A Windows directory tree

File Systems: UNIX Directory Tree [DL-CS-16]

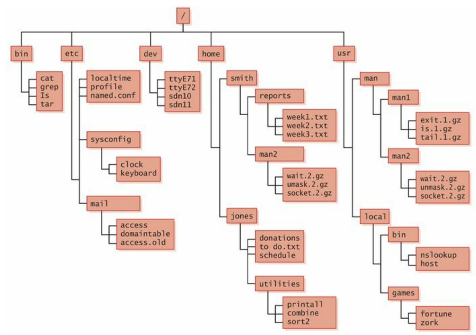
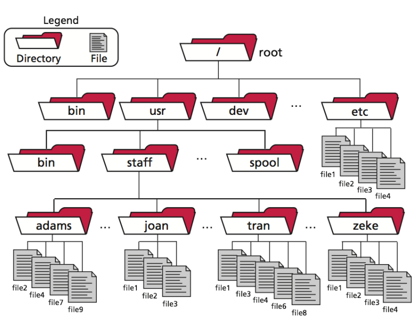


FIGURE 11.5 A UNIX directory tree

File Systems: Directory Tree [F-CS-18]

Figure 13.14 *An example of directory system in UNIX*



Three Major OSs: Unix, Mac OS X, and Windows

Three Major Operating Systems: Unix, Mac OS X, and Windows

The UNIX Operating System [F-CS-18]

7.4.1 UNIX

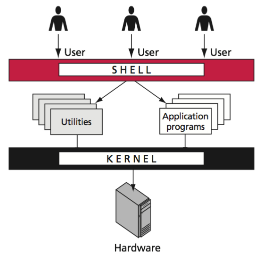
UNIX was originally developed in 1969 by Thomson and Ritchie of the Computer Science Research Group at Bell Laboratories. UNIX has gone through many versions since then. It has been a popular operating system among computer programmers and computer scientists. It is a very powerful operating system with three outstanding features. First, UNIX is a portable operating system that can be moved from one platform to another without many changes. The reason is that it is written mostly in the C language (instead of a machine language specific to a particular computer system). Second, UNIX has a powerful set of utilities (commands) that can be combined (in an executable file called a *script*) to solve many problems that require programming in other operating systems. Third, it is device-independent, because it includes device drivers in the operating system itself, which means that it can be easily configured to run any device.

UNIX is a multiuser, multiprocessing, portable operating system designed to facilitate programming, text processing, communication, and many other tasks that are expected from an operating system. It contains hundreds of simple, single-purpose functions that can be combined to do virtually every processing task imaginable. Its flexibility is demonstrated by the fact that it is used in three different computing environments: stand-alone personal environments, time-sharing systems, and client–server systems.

**UNIX is a multiuser, multiprocessing, portable operating system.
It is designed to facilitate programming, text processing, and communication.**

The UNIX Operating System [F-CS-18]

Figure 7.20 *Components of the UNIX operating system*



The UNIX Operating System [SGG-OS-11]

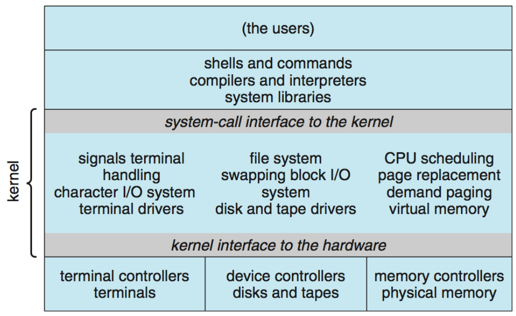


Figure 2.13 Traditional UNIX system structure.

The Mac OS X Operating System [SGG-OS-11]

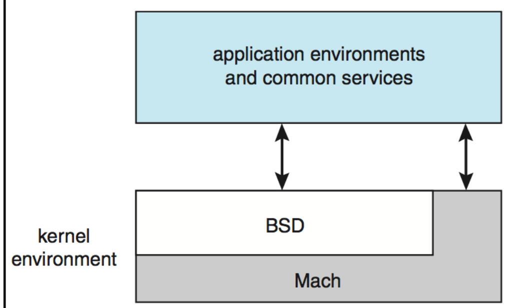
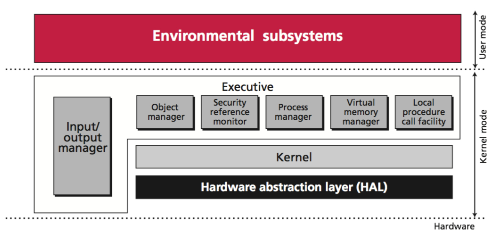


Figure 2.16 The Mac OS X structure.

Windows Operating System [F-CS-18]

Figure 7.21 *The architecture of Windows*



An Introduction to Computer Science

- **Computer Science: What Is It and Why Study It?**
- **Computation: What Is It and Why Study It?**
- **Computability**
- **Computational Complexity (CS101A class only)**
- **Algorithms**
- **Data, Information, and Knowledge, and Their Representations**
- **Data Storage**
- **Computer Architecture**
- **Data Manipulation in Computer Systems**
- **Programming Languages and Compilers**
- **Operating Systems**
- **System Software and Application Software**
- **Software Engineering (CS101A class only)**
- **Knowledge Engineering and Artificial Intelligence (CS101A class only)**
- **Information Security Engineering (CS101A class only)**

System Software and Application Software

System Software and Application Software

System Software

- System software [A Dictionary of Computer Science (7th Edition), OUP, 2016]
- "The totality of software required to produce a system acceptable to end users."
- System software [IEEE Standard Computer Dictionary]
- "Software designed to facilitate the operation and maintenance of a computer system and its associated programs; for example, operating systems, assemblers, utilities."

System software Programs that manage a computer system and interact with hardware

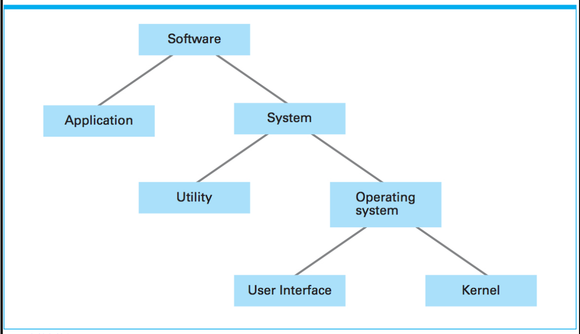
Application Software

- Applications software [A Dictionary of Computer Science (7th Edition), OUP, 2016]
  - "Collective term for applications programs."
- Applications program [A Dictionary of Computer Science (7th Edition), OUP, 2016]
  - "Any program that is specific to a particular role and makes a direct contribution to performing that role."
  - Application software [IEEE Standard Computer Dictionary]
  - "Software designed to fulfill specific needs of a user; for example, software for navigation, payroll, or process control."

Application software Programs that help us solve real-world problems

Software Classification

Figure 3.3 Software classification



System Software vs. Application Software

- The distinction between system software and application software
- The distinction between system software and application software is not so explicitly and strictly.
- Example: MS internet explorer
- Is MS IE an application software or a part of Windows OS?
- If it is the former, then MS should sell IE as a separate product.
- If it is the latter, then MS may sell IE within Windows OS.
- This distinction is very important in determining whether MS violated antimonopoly law or not.

System Software

- The features of system software
- Control and manage resources.
- Interact with hardware.
- Provide common services to ALL users and application software.
- System software we have learned
- Assemblers, Compilers, Interpreters, Operating Systems (including loaders, device drivers, etc.).
- Other system software
- Editors, linkers, debuggers, storage-media formaters, network software, and various utilities.

Application Software

- The features of application software
  - Each has a SPECIAL purpose; Each solves a SPECIAL problem; Not serve for all users but a SPECIAL group.
- Application software
  - Scientific computing software, Engineering computing software, Business computing software, CAD software, Database management systems (DBMS), etc.
  - Office software (calendar, scheduler, note pad, calculator, reader, word processor, presentation, table calculation, drawing, dictionaries, etc.).
  - Internet software (chat, e-mail, browser, download, terminal, etc.).
  - Multimedia software (photo processing, music player, video player, voice recorder, voice editor, video editor, etc.).
  - Entertainment software (games, etc.).

Database System: What Is It ?

Database System: What Is It ?

Database System: What is It?

- Database System (DBS) [A Dictionary of Computer Science (7th Edition), OUP, 2016]
  - "Normally and strictly, a body of information held within a computer system using the facilities of a database management system. All accessing and updating of the information will be via the facilities provided by this software as will be the recording of information on the log file, database recovery, and multi-access control."
- Database management system (DBMS) [A Dictionary of Computer Science (7th Edition), OUP, 2016]
  - "A software system that provides comprehensive facilities for the organization and management of a body of information required for some particular application or group of related applications."

Database System: What is It?

- Database (DB) [IEEE Standard Computer Dictionary]
  - "A collection of logically related data stored together in one or more computerized files."
- Database system (DBS) [IEEE Standard Computer Dictionary]
  - "A software system that supports multiple applications using a common database."
- Database management system (DBMS) [IEEE Standard Computer Dictionary]
  - "A computer system involving hardware, software, or both that provides a systematic approach to creating, sorting, retrieving and processing information stored in a database. A DBMS acts as an interface between computer's programs and data files as well as between users and the database. It may include backup/recovery, checkpoint processing, and ad-hoc query capability."

Database System: What is It?

- Database (DB)
  - A *database* is a collection of logically related data stored structurally, organizationally, and together in a computing system.
  - Notes: "data" but not "information", "logically related", "stored structurally, organizationally, and together", and "in a computing system".
- Database management system (DBMS)
  - A *database management system* (*DBMS*) is a software system running on a computing system that provides a systematic approach to manage data stored in a database and control access to the database.
  - The DBMS is a application-independent (general-purpose) software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications.

Database System: What is It?

- Database system (DBS)
  - A *database system* consists of a database and a database management system managing it.
  - Management of data involves both defining structures for storage of data and providing mechanisms for the manipulation of data.
  - In addition, the database system must ensure the safety and security of the data stored, despite system crashes or attempts at unauthorized access.
  - Note: Using a database management system to manage different databases may create and maintain different database systems.
- Database application (DB application, Not DBA!)
  - A *database application* is simply a program that interacts with a database system at some point in its execution.

Database System: What is It?

Database (DB) (application-dependent)

+

Database Management System (DBMS) (application-independent)

=

Database System (DBS)

A Simplified Database System Environment [EN-DBS-16]

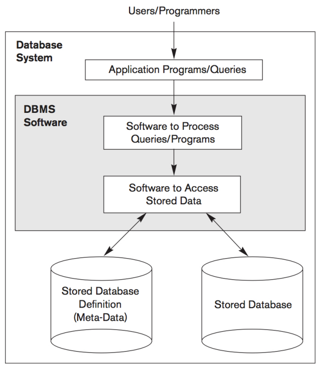


Figure 1.1
A simplified database system environment.

A Database Example [EN-DBS-16]

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

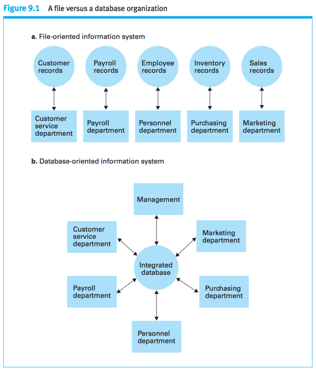
Figure 1.2
A database that stores student and course information.

Database System: Why Study It ?

Database System: Why Study It ?

File Organization vs. Database Organization

Figure 9.1 A file versus a database organization



Database System: Why Study It?

♣ **File-based systems**

- **A collection of application programs that perform services for the end-users.**
- **Each program defines and manages its own data.**

♣ **Disadvantages of data processing by files**

- **Data redundancy and inconsistency; Difficulty in accessing data; Data isolation; Integrity problems; Atomicity problems; Concurrent-access anomalies; Security problems**

♣ **The necessity of database systems**

- **The disadvantages of data processing by files prompted the development of database systems.**
- **The concepts and algorithms of database systems enable them to solve the problems with file-processing systems.**

Example: Data Processing by Files [CB-DBS-15]

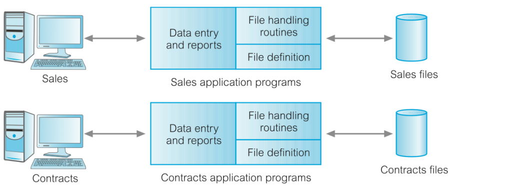


Figure 1.5 File-based processing.

Sales Files
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)
PrivateOwner (ownerNo, fName, lName, address, telNo)
Client (clientNo, fName, lName, address, telNo, prefType, maxRent)

Contracts Files
Lease (leaseNo, propertyNo, clientNo, rent, paymentMethod, deposit, paid, rentStart, rentFinish, duration)
PropertyForRent (propertyNo, street, city, postcode, rent)
Client (clientNo, fName, lName, address, telNo)

Disadvantages of Data Processing by Files

♣ **Data redundancy (duplication) and inconsistency**

- **Since different programmers create the files and application programs over a long period, the various files are likely to have different structures and the programs may be written in several programming languages.**
- **Moreover, the same data may be duplicated in several places (files).**
- **This redundancy leads to higher storage and access cost.**
- **In addition, it may lead to data inconsistency; that is, the various copies of the same data may no longer agree.**

Disadvantages of Data Processing by Files

♣ **Data separation and isolation**

- **The conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner.**
- **Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.**
- **When data is isolated in separate files, it is more difficult to access data that should be available. The application developers must synchronize the processing of two or more files to ensure that the correct data is extracted.**

Disadvantages of Data Processing by Files

♣ **Data dependence**

- **The physical structure and storage of the data files and records are defined in the application code.**
- **This means that changes to an existing structure are difficult to make.**
- **The programmers needs to identify all the affected programs, modify them, and then retest them.**
- **Clearly, this process could be very time-consuming and subject to error.**
- **This characteristic of file-based systems is known as *program–data dependence*.**

Disadvantages of Data Processing by Files

♣ **Integrity problems**

- **The data values stored in the database must satisfy certain types of consistency constraints.**
- **Developers enforce these constraints in the system by adding appropriate code in the various application programs.**
- **However, when new constraints are added, it is difficult to change the programs to enforce them.**
- **The problem is compounded when constraints involve several data items from different files.**

Disadvantages of Data Processing by Files

♣ **Atomicity problems**

- **A computer system, like any other device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure.**
- **That is, the failed operation must be atomic, i.e., it must happen in its entirety or not at all.**
- **It is difficult to ensure atomicity in a conventional file-processing system.**

Disadvantages of Data Processing by Files

♣ **Concurrent-access anomalies**

- **For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously.**
- **In such an environment, interaction of concurrent updates is possible and may result in inconsistent data.**
- **To guard against this possibility, the system must maintain some form of supervision.**
- **But supervision is difficult to provide because data may be accessed by many different application programs that have not been coordinated previously.**

Disadvantages of Data Processing by Files

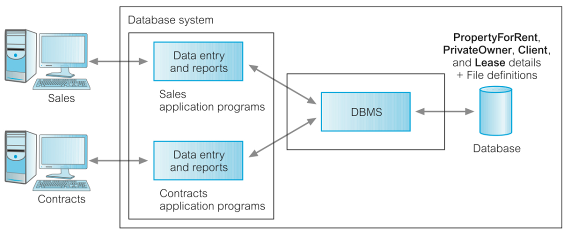
♣ **Security problems**

- **Not every user of the database system should be able to access all the data.**
- **But, since application programs are added to the file-processing system in an ad hoc manner, enforcing such security constraints is difficult.**

♣ **Two intrinsic factors of the file-based approach**

- **The definition of the data is embedded in the application programs, rather than being stored separately and independently.**
- **There is no control over the access and manipulation of data beyond that imposed by the application programs.**

Example: Data Processing Database System [CB-DBS-15]

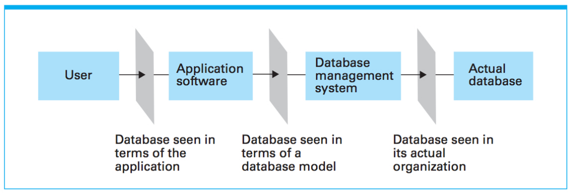


PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)
PrivateOwner (ownerNo, fName, lName, address, telNo)
Client (clientNo, fName, lName, address, telNo, prefType, maxRent)
Lease (leaseNo, propertyNo, clientNo, paymentMethod, deposit, paid, rentStart, rentFinish)

Figure 1.7 Database processing.

The Conceptual Layers of a Database Implementation

Figure 9.2 The conceptual layers of a database implementation



Database Management System: What Is It ?

Database Management System: What Is It ?

DBMS as an Interface between Databases and Users

♣ **DBMS as an interface**

- **The users and application programs do not directly manipulate the database.**
- **The actual manipulation of the database is performed by the DBMS.**

♣ **Benefits of dichotomy between the application programs and the DBMS**

- **It allows for the construction and use of abstract tools. If the details of how the database is actually stored are isolated within the DBMS, the design of the application software can be greatly simplified.**
- **Such an organization provides a means for controlling access to the database.**

Functions of DBMS

♣ **Services for users**

- Allow users to create/define new DBs and specify their schemas (logical structure of the data), usually through a specialized *Data-Definition Language* (*DDL*).
- Allow users to insert, delete, update, and retrieve data from the DB, usually through a *Data Manipulation Language* (*DML*).
- Having a central repository for all data and data descriptions allows the DML to provide a general inquiry facility to this data, called a *query language*.

♣ **Managements for DBs**

- Support the storage of very large amounts of data over a long period of time, allowing efficient access to the data for queries and DB modifications.

Functions of DBMS

♣ **Managements for DBs**

- Enable durability, the recovery of the DB in the face of failures, errors of many kinds, or intentional misuse.
- Control access to data from many users at once, without allowing unexpected interactions among users (called isolation) and without actions on the data to be performed partially but not completely (called atomicity).
- For example, it may provide:
  a security system, which prevents unauthorized users accessing the DB;
  an integrity system, which maintains the consistency of stored data;
  a concurrency control system, which allows shared access of the DB;
  a recovery control system, which restores the DB to a previous consistent state following a hardware or software failure;
  a user-accessible catalog, which contains descriptions of the data in the DB.

Components of a DBMS and Their Interactions [EN-DBS-16]

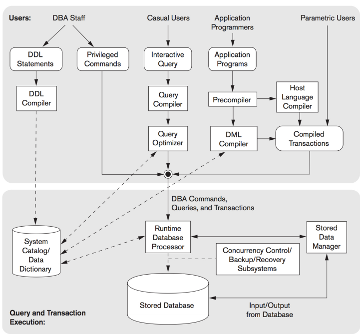


Figure 2.3
Component modules of a DBMS and their interactions.

Database System Structure [S-DBS-10]

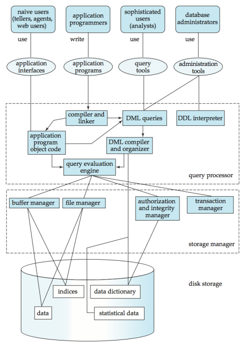


Figure 1.5 System structure.

DBMS: Advantages vs. Disadvantages [CB-DBS-15]

TABLE 1.2 Advantages of DBMSs.

Control of data redundancy	Economy of scale
Data consistency	Balance of conflicting requirements
More information from the same amount of data	Improved data accessibility and responsiveness
Sharing of data	Increased productivity
Improved data integrity	Improved maintenance through data independence
Improved security	Increased concurrency
Enforcement of standards	Improved backup and recovery services

TABLE 1.3 Disadvantages of DBMSs.

Complexity
Size
Cost of DBMSs
Additional hardware costs
Cost of conversion
Performance
Greater impact of a failure

TABLE 1.1 Limitations of file-based systems.

Separation and isolation of data
Duplication of data
Data dependence
Incompatible file formats
Fixed queries/proliferation of application programs

DBMS: Advantages vs. Disadvantages [CB-DBS-15]

表 1-2　DBMS 的优点

受控的数据冗余	经济合算的规模
数据一致性	平衡各种需求冲突
相同数据量表示更多信息	增强的数据可访问性和响应性
数据共享	提高的生产率
增强的数据完整性	通过数据的独立性增强可维护性
增强的安全性	提高的并发性
强制执行标准	增强的备份和恢复服务

表 1-1　基于文件系统的局限性

数据被分离和孤立
数据存在冗余
数据存在依赖性
文件格式不相容
查询一成不变 / 应用程序需不断翻新

表 1-3　DBMS 的缺点

复杂性高
规模大
DBMS 的费用高
需要附加的硬件费用
转化费用大
性能相对较低
故障带来的影响较大

Database System Applications ?

Database System Applications ?

Database System Applications in Enterprises

♣ Enterprise data/information management

- Sales: For customer, product, and purchase information.
- Accounting: For payments, receipts, account balances, assets and other accounting information.
- Human resources: For information about employees, salaries, payroll taxes, and benefits, and for generation of paychecks.
- Manufacturing: For management of the supply chain and for tracking production of items in factories, inventories of items in warehouses and stores, and orders for items.
- Online retailers: For sales data noted above plus online order tracking, generation of recommendation lists, and maintenance of online product evaluations.

Database System Applications in Enterprises

♣ Banking and Finance

- Banking: For customer information, accounts, loans, and banking transactions.
- Credit card transactions: For purchases on credit cards and generation of monthly statements.
- Finance: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds; also for storing real-time market data to enable online trading by customers and automated trading by the firm.

♣ Universities

- For student information, course registrations, and grades (in addition to standard enterprise information such as human resources and accounting).

Database System Applications in Enterprises

♣ Airlines

- For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner.

♣ Telecommunication

- For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.

♣ Universality of database applications

- As the list illustrates, databases form an essential part of every enterprise today, storing not only types of data that are common to most enterprises, but also data that is specific to the category of the enterprise.

Database System Applications in Our Daily Life

♣ Purchases from the supermarket

- When you purchase goods from your local supermarket, it is likely that a database is accessed.
- The checkout assistant uses a bar code reader to scan each of your purchases.
- This reader is linked to a database application that uses the bar code to find out the price of the item from a product database.
- The application then reduces the number of such items in stock and displays the price on the cash register.
- If the reorder level falls below a specified threshold, the database system may automatically place an order to obtain more of that item.

Database System Applications in Our Daily Life

♣ Purchases using your credit card

- When you purchase goods using your credit card, the assistant normally checks whether you have sufficient credit left to make the purchase.
- This check may be carried out by telephone or automatically by a card reader linked to a computer system.
- In either case, there is a database somewhere that contains information about the purchases that you have made using your credit card.
- To check your credit, there is a database application that uses your credit card number to check that the price of the goods you wish to buy, together with the sum of the purchases that you have already made this month, is within your credit limit.
- When the purchase is confirmed, the details of the purchase are added to this database.
- The database application also accesses the database to confirm that the credit card is not on the list of stolen or lost cards before authorizing the purchase.

Database System Applications in Our Daily Life

♣ Booking a vacation with a travel agent

- When you make inquiries about a vacation, your travel agent may access several databases containing vacation and flight details.
- When you book your vacation, the database system has to make all the necessary booking arrangements.
- In this case, the system has to ensure that two different agents do not book the same vacation or overbook the seats on the flight.
- The travel agent may have another, usually separate, database for invoicing.

Database System Applications in Our Daily Life

♣ Using the local library

- Your local library probably has a database containing details of the books in the library, details of the readers, reservations, and so on.
- There will be a computerized index that allows readers to find a book based on its title, authors, or subject area.
- The database system handles reservations to allow a reader to reserve a book and to be informed by mail or email when the book is available.
- The system also sends reminders to borrowers who have failed to return books by the due date.
- Typically, the system will have a bar code reader, similar to that used by the supermarket described earlier, that is used to keep track of books coming in and going out of the library.

Database System Applications in Our Daily Life

♣ Taking out insurance

- Whenever you wish to take out insurance, your agent may access several databases containing figures for various insurance organizations.
- The personal details that you supply, such as name, address, age, and whether you drink or smoke, are used by the database system to determine the cost of the insurance.
- An insurance agent can search several databases to find the organization that gives you the best deal.

Database System Applications in Our Daily Life

♣ Using the Internet

- Many of the sites on the Internet are driven by database applications.
- Google
- Facebook
- Twitter
- Amazon
- Baidu
- QQ
- Weixin (WeChat)

Database Examples

Figure 9.3 A relation containing employee information

Empl Id	Name	Address	SSN
25X15	Joe E. Baker	33 Nowhere St.	111223333
34Y70	Cheryl H. Clark	563 Downtown Ave.	999009999
23Y34	G. Jerry Smith	1555 Circle Dr.	111005555
•	•	•	•
•	•	•	•
•	•	•	•

Database Examples

Figure 9.5 An employee database consisting of three relations

EMPLOYEE relation

Empl Id	Name	Address	SSN
25X15	Joe E. Baker	33 Nowhere St.	111223333
34Y70	Cheryl H. Clark	563 Downtown Ave.	999009999
23Y34	G. Jerry Smith	1555 Circle Dr.	111005555

JOB relation

Job Id	Job Title	Skill Code	Dept
S25X	Secretary	T5	Personnel
S26Z	Secretary	T6	Accounting
F5	Floor manager	FM3	Sales
•	•	•	•
•	•	•	•
•	•	•	•

ASSIGNMENT relation

Empl Id	Job Id	Start Date	Term Date
23Y34	S25X	3-1-1999	4-30-2010
34Y70	F5	10-1-2009	*
23Y34	S26Z	5-1-2010	*
•	•	•	•
•	•	•	•
•	•	•	•

Database Examples

Figure 9.6 Finding the departments in which employee 23Y34 has worked

EMPLOYEE relation

Empl Id	Name	Address	SSN
25X15	Joe E. Baker	33 Nowhere St.	111223333
34Y70	Cheryl H. Clark	563 Downtown Ave.	999009999
23Y34	G. Jerry Smith	1555 Circle Dr.	111005555
• • •	• • •	• • •	• • •

JOB relation

Job Id	Job Title	Skill Code	Dept
S25X	Secretary	T5	Personnel
S26Z	Secretary	T6	Accounting
F5	Floor manager	FM3	Sales
• • •	• • •	• • •	• • •

are contained in the personnel and accounting departments.

ASSIGNMENT relation

Empl Id	Job Id	Start Date	Term Date
23Y34	S25X	3-1-1999	4-30-2010
34Y70	F5	10-1-2009	*
23Y34	S26Z	5-1-2010	*
• • •	• • •	• • •	• • •

The jobs held by employee 23Y34

Historical Development of Database Systems ?

Historical Development of Database Systems

Historical Development of Database Systems [CB-DBS-15]

TIMEFRAME	DEVELOPMENT	COMMENTS
1960s (onwards)	File-based systems	Precursor to the database system. Decentralized approach: each department stored and controlled its own data.
Mid-1960s	Hierarchical and network data models	Represents first-generation DBMSs. Main hierarchical system is IMS from IBM and the main network system is IDMS/R from Computer Associates. Lacked data independence and required complex programs to be developed to process the data.
1970	Relational model proposed	Publication of E. F. Codd's seminal paper "A relational model of data for large shared data banks," which addresses the weaknesses of first-generation systems.
1970s	Prototype RDBMSs developed	During this period, two main prototypes emerged: the Ingres project at the University of California at Berkeley (started in 1970) and the System R project at IBM's San José Research Laboratory in California (started in 1974), which led to the development of SQL.
1976	ER model proposed	Publication of Chen's paper "The Entity-Relationship model—Toward a unified view of data." ER modeling becomes a significant component in methodologies for database design.
1979	Commercial RDBMSs appear	Commercial RDBMSs like Oracle, Ingres, and DB2 appear. These represent the second generation of DBMSs.
1987	ISO SQL standard	SQL is standardized by the ISO (International Standards Organization). There are subsequent releases of the standard in 1989, 1992 (SQL2), 1999 (SQL:1999), 2003 (SQL:2003), 2008 (SQL:2008), and 2011 (SQL:2011).
1990s	OODBMS and ORDBMSs appear	This period initially sees the emergence of OODBMSs and later ORDBMSs (Oracle 8, with object features released in 1997).
1990s	Data warehousing systems appear	This period also see releases from the major DBMS vendors of data warehousing systems and thereafter data mining products.
Mid-1990s	Web–database integration	The first Internet database applications appear. DBMS vendors and third-party vendors recognize the significance of the Internet and support web–database integration.
1998	XML	XML 1.0 ratified by the W3C. XML becomes integrated with DBMS products and native XML databases are developed.

Figure 1.10 Historical development of database systems.

Historical Development of Database Systems [KA-DBS-12]

Figure 1-25 Database History

Era	Years	Important Products	Remarks
Predatabase	Before 1970	File managers	All data were stored in separate files. Data integration was very difficult. File storage space was expensive and limited.
Early database	1970–1980	ADABAS, System2000, Total, IDMS, IMS	First products to provide related tables. CODASYL DBTG and hierarchical data models (DL/I) were prevalent.
Emergence of relational model	1978–1985	DB2, Oracle	Early relational DBMS products had substantial inertia to overcome. In time, the advantages weighed out.
Microcomputer DBMS products	1982–1992+	dBase-II, R:base, Paradox, Access	Amazing! A database on a micro. All micro DBMS products were eliminated by Microsoft Access in the early 1990s.
Object-oriented DBMS	1985–2000	Oracle ODBMS and others	Never caught on. Required relational database to be converted. Too much work for perceived benefit.
Web databases	1995–present	IIS, Apache, PHP, ASP.NET, and Java	Stateless characteristic of HTTP was a problem at first. Early applications were simple one-stage transactions. Later, more complex logic developed.
Open source DBMS products	1995–present	MySQL, PostgresQL, and other products	Open source DBMS products provide much of the functionality and features of commercial DBMS products at reduced cost.
XML and Web services	1998–present	XML, SOAP, WSDL, UDDI, and other standards	XML provides tremendous benefits to Web-based database applications. Very important today. May replace relational databases during your career. See Chapter 12.
The NoSQL movement	2009–present	Apache Cassandra, dbXML, MonetDB/XQuery, and other products	The NoSQL movement is really a NoRelationalDB movement that replaces relational databases with nonrelational data structures. The NoSQL approach, which is used by Facebook and Twitter, often is based on XML. See Chapter 12.

Data Abstraction: What Is It and Why Study It ?

Data Abstraction: What Is It and Why Study It ?

Data Abstraction

- **A major purpose of a DBS: providing abstract view of the data**
  - **A major purpose of a DBS is to provide users with an abstract view of the data.**
  - **That is, the system hides certain details of how the data are stored and maintained.**
- **Data abstraction at several levels**
  - **For the DBS to be usable, it must retrieve data efficiently.**
  - **The need for efficiency has led designers to use complex data structures to represent data in the DB.**
  - **Since many DBS users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users' interactions with the system.**

The Three Levels of Data Abstraction [S-DBS-10]

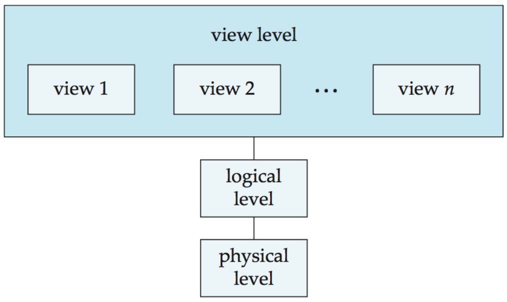


Figure 1.1 The three levels of data abstraction.

Data Abstraction

♣ Physical level

- The lowest level of abstraction describes how the data are actually stored.
- The physical level describes complex low-level data structures in detail.

♣ Logical level

- The next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data.
- The logical level thus describes the entire database in terms of a small number of relatively simple structures.

Data Abstraction

♣ View level

- The highest level of abstraction describes only part of the entire database.
- Even though the logical level uses simpler structures, complexity remains because of the variety of data stored in a large database.
- The view level of abstraction exists to simplify their interaction with the system.
- The system may provide many views for the same database.

The ANSI-SPARC Three-level Architecture

♣ The DBTG two-level architecture

- An early proposal for a standard terminology and general architecture for database systems was produced in 1971 by the DBTG appointed by CODASYL in 1971.
- The DBTG recognized the need for a two-level approach with a system view called the schema and user views called subschemas.

♣ The ANSI-SPARC three-level architecture

- The American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC), or ANSI/X3/SPARC, produced a similar terminology and architecture in 1975 (ANSI, 1975).
- The ANSI-SPARC architecture recognized the need for a three-level approach with a system catalog.

The ANSI-SPARC Three-level Architecture [S-DBS-10]

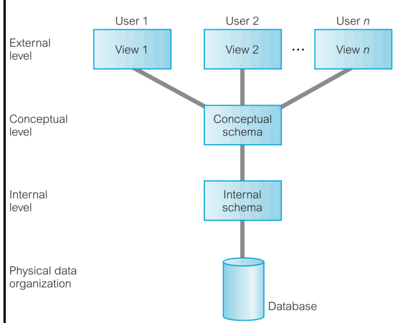


Figure 2.1 The ANSI-SPARC three-level architecture.

The ANSI-SPARC Three-level Architecture

♣ The ANSI-SPARC three-level architecture

- The levels form a three-level architecture comprising an external, a conceptual, and an internal level.
- The way users perceive the data is called *the external level*.
- The way the DBMS and the operating system perceive the data is *the internal level*, where the data is actually stored using the data structures and file organizations.
- *The conceptual level* provides both the mapping and the desired independence between the external and internal levels.

The ANSI-SPARC Three-level Architecture

♣ The external level

- The users' view of the database.
- This level describes that part of the database that is relevant to each user.
- The external level consists of a number of different external views of the database.
- Each user has a view of the "real world" represented in a form that is familiar for that user.
- The external view includes only those entities, attributes, and relationships in the "real world" that the user is interested in.
- Other entities, attributes, or relationships that are not of interest may be represented in the database, but the user will be unaware of them.

The ANSI-SPARC Three-level Architecture

♣ The conceptual level

- The community view of the database.
- This level describes what data is stored in the database and the relationships among the data.
- This level contains the logical structure of the entire database as seen by the database application.
- It is a complete view of the data requirements of the organization that is independent of any storage considerations.
- The conceptual level represents: all entities, their attributes, and their relationships; the constraints on the data; semantic information about the data; security and integrity information.

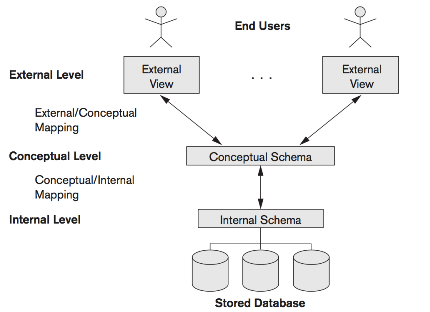
The ANSI-SPARC Three-level Architecture

♣ The internal level

- The physical representation of the database on the computer.
- This level describes how the data is stored in the database.
- The internal level covers the physical implementation of the database to achieve optimal runtime performance and storage space utilization.
- It covers the data structures and file organizations used to store data on storage devices.
- It interfaces with the operating system access methods to place the data on the storage devices, build the indexes, retrieve the data, and so on.

The Three-schema Architecture [EN-DBS-16]

Figure 2.2
The three-schema architecture.



[9]This is also known as the ANSI/SPARC (American National Standards Institute/ Standards Planning And Requirements Committee) architecture, after the committee that proposed it (Tsichritzis & Klug, 1978).

Components of a DBMS and Their Interactions [EN-DBS-16]

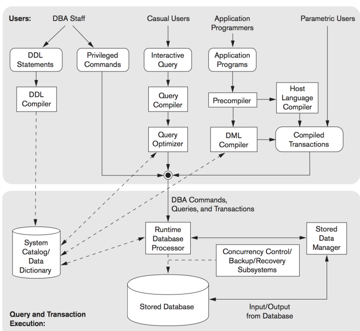


Figure 2.3
Component modules of a DBMS and their interactions.

A Physical Centralized Architecture [EN-DBS-16]

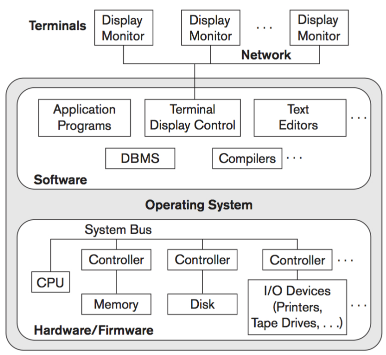
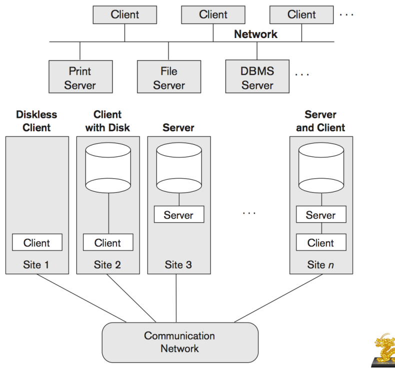


Figure 2.4
A physical centralized architecture.

Logical Two-tier Architecture and Physical Two-tier Architecture [EN-DBS-16]

Figure 2.5
Logical two-tier client/server architecture.

Figure 2.6
Physical two-tier client/server architecture.



Two-tier and Three-tier Architectures of Database Applications

- **Clients and servers**
- **Most users of a database system today are not present at the site of the database system, but connect to it through a network.**
- **We can therefore differentiate between *client machines*, on which remote database users work, and *server machines*, on which the database system runs.**
- **The two-tier architecture**
- **The application resides at the client machine, where it invokes database system functionality at the server machine through query language statements.**

Two-tier and Three-tier Architectures of Database Applications

- **The three-tier architecture**
- **The client machine acts as merely a front end and does not contain any direct database calls.**
- **Instead, the client end communicates with an application server, usually through a forms interface.**
- **The application server in turn communicates with a database system to access data.**
- **The business logic of the application, which says what actions to carry out under what conditions, is embedded in the application server, instead of being distributed across multiple clients.**
- **Three-tier applications are more appropriate for large applications, and for applications that run on the WWW.**

Two-tier and Three-tier Architectures of Database Applications [S-DBS-10]

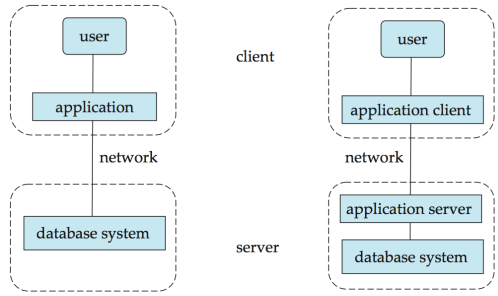


Figure 1.6 Two-tier and three-tier architectures.

An Introduction to Computer Science

- **Computer Science: What Is It and Why Study It?**
- **Computation: What Is It and Why Study It?**
- **Computability**
- **Computational Complexity (CS101A class only)**
- **Algorithms**
- **Data, Information, and Knowledge, and Their Representations**
- **Data Storage**
- **Computer Architecture**
- **Data Manipulation in Computer Systems**
- **Programming Languages and Compilers**
- **Operating Systems**
- **System Software and Application Software**
- **Software Engineering (CS101A class only)**
- **Knowledge Engineering and Artificial Intelligence (CS101A class only)**
- **Information Security Engineering (CS101A class only)**