# CHAPTER 4

# COMPUTER APPLICATIONS

Applications software includes programs that users access to carry out work. This lesson examines two applications that may be of particular use to computer science student´s:

**9.1 DATABASES:** A database is a collection of information stored on one or several computers. Databases were originally developed using mainframe computers but are now common on personal computers and in networked environments. However, in the early 1980s, database technology began to move from mainframes to PCs. As a result, database management systems gradually became more powerful and easier to use. To understand a little more about databases there is a brief description in the next lines:

- **Data field:** A space allocated for a particular item of information. In a database, fields are the smallest units of information you can access. A data field contains a single piece of information (first name, family name, employee number, salary and so on).

- **Database record:** A complete set of information in a database; records are composed of fields, each of which contains one item of information. A collection of records (in this case, employee records) comprises a database. Structured databases typically store data that describes a collection of similar entities.

- **Data structure:** A scheme for organizing related pieces of information. The basic types of structures include: files, lists, arrays, records, trees, tables. Each of these basic structures has many variations and allows different operations to be performed on the data. There are three basic database models:

    - ✓ **Hierarchical databases** exhibit a branching structure, with information arranged into sets and sub-sets; An example is the process involved in finding a distant cousin on a family tree.
    - ✓ **Network databases** offer many more direct connections between files, but, similar to hierarchies, the links are predefined and are difficult to change or adjust.
    - ✓ **Object-oriented databases** link self contained entities (or objects) together. Objects can be text, a picture, a piece of film or any item that can be individually selected and manipulated.

The limitations found with these types of databases explain why most organizations have turned to relational databases. Relational databases are powerful because they require few assumptions about how data is related or how it will be extracted from the database. As a result, the same database can be viewed in many different ways.

- **Relational database:** A database that spreads information across different tables while maintaining links between them. A relational database stores facts in tables called relations. The only requirement is that the

information must be capable of being laid out in rows and columns (similar to a list of names, addresses and phone numbers).

- **Database management software**: Is used to create and manipulate databases. Data access software is usually used to search databases. The data access software understands the structure and details of the database which means the user simply has to enter his or her search specifications, using either a menu, a keyword search engine, a query language or a natural language (among others).

- **Query language:** A set of command words that can be used to direct a computer to create databases, locate information, sort records and change the data in those records.
  One query language is called SQL (structured query language). The use of query language is based on knowledge of the command word and the grammar or syntax that will let one construct valid query sentences. In more sophisticated systems, queries can also be formulated in a natural language such as standard English, French, or Japanese.

**9.2 COMMUNICATING BY COMPUTER:** Communicating via e-mail is rapidly becoming as important as telephone and fax communication and forms an important component of any office automation system.

- **Electronic mail (e-mail):** A way of sending messages between people anywhere within an organization or in the world using a *computer* that can communicate with another computer through a *computer network.* E-mail is handled by a variety of software programs such as Microsoft Outlook Express, Eudora and so on. The message originator creates a message file in the e-mail software editor. When complete, the message is posted to a message transport system that assumes the responsibility for delivering that message to its recipient 'mailbox'.
  To receive and read the message, the recipient runs a software program that retrieves incoming messages, allowing the messages to be filed, listed, forwarded or replied to.
  Generally a single user-interface program is used to send and receive messages both locally and worldwide. Users do not need to have the same e-mail software program as the person they are corresponding with. The e-mail itself may consist of simply a message or may carry with it attachments containing files created in a variety of software applications, for example word processed documents or spreadsheets.

## CLASSROOM PRACTICE

**EXERCISE A: Answer the following questions using the reading**

# CHAPTER 5

# OPERATING SYSTEMS

**INTRODUCCTION**

A modern computer consists of one or more processors, some main memory, disks, printers, a keyboard, a mouse, a display, network interfaces, and various other input/output devices. All in all, a complex system. If every application programmer had to understand how all these things work in detail, no code would ever get written. Furthermore, managing all these components and using them optimally is an exceedingly challenging job. For this reason, computers are equipped with a layer of software called the operating system, whose job is to provide user programs with a better, simpler, cleaner, model of the computer and to handle managing all the resources just mentioned. Operating systems is the program that users interact with, usually called the shell when it is text based and the GUI (Graphical User Interface)— in Fig. 1-1. Here we see the hardware at the bottom. The hardware consists of chips, boards, disks, a keyboard, a monitor, and similar physical objects. On top of the hardware is the software. Most computers have two modes of operation: kernel mode and user mode. The operating system, the most fundamental piece of software, runs in kernel mode (also called supervisor mode). The rest of the software runs in user mode, in which only a subset of the machine instructions is available.

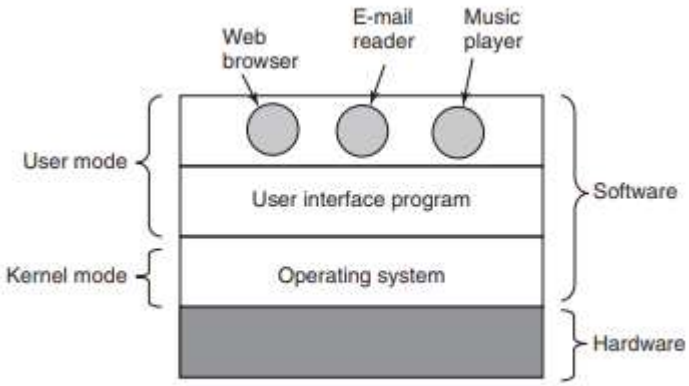


**Figure 1-1.** Where the operating system fits in.

The user interface program, shell or GUI, is the lowest level of user-mode software, and allows the user to start other programs, such as a Web browser, email reader, or music player. These programs, too, make heavy use of the operating system.

Operating systems differ from user (i.e., application) programs in ways other than where they reside. In particular, they are huge, complex, and long-lived. The source code of the heart of an operating system like Linux or Windows is on the order of five million lines of code or more. It should be clear now why operating systems live a long time—they are very hard to write.

Windows 95/98/Me was basically one operating system and Windows NT/2000/XP/Vista/Windows 7 is a different one. They look similar to the users because Microsoft made very sure that the user interface of

Windows 2000/XP/Vista/Windows 7 was quite similar to that of the system it was replacing, mostly Windows 98. Besides Windows, the other main example we will use throughout this book is UNIX and its variants and clones.

**FUNCTIONS OF AN OPERATING SYSTEMS**

The architecture (instruction set, memory organization, I/O, and bus structure) of most computers at the machine-language level is primitive and awkward to program, especially for input/output.

But even this level is much too low for most applications. For this reason, all operating systems provide yet another layer of abstraction for using disks: files. Using this abstraction, programs can create, write, and read files, without having to deal with the messy details of how the hardware actually works. This abstraction is the key to managing all this complexity.

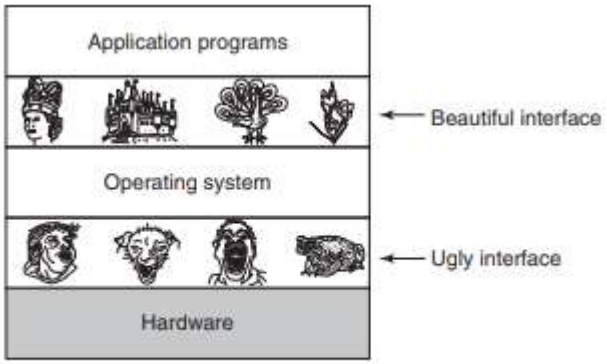


**Figure 1-2.** Operating systems turn ugly hardware into beautiful abstractions.

One of the major tasks of the operating system is to hide the hardware and present programs (and their programmers) with nice, clean, elegant, consistent, abstractions to work with instead. Operating systems turn the ugly into the beautiful, as shown in Fig. 1-2.

It should be noted that the operating system's real customers are the application programs (via the application programmers, of course). They are the ones who deal directly with the operating system and its abstractions. In contrast, end users deal with the abstractions provided by the user interface, either a command-line shell or a graphical interface.

**HISTORY OF OPERATING SYSTEMS**

Operating systems have been evolving through the years. In the following sections we will briefly look at a few of the highlights. Since operating systems have historically been closely tied to the architecture of the computers on which they run, we will look at successive generations of computers to see what their operating systems were like. The progression given below is largely chronological, each development did not wait until the previous one nicely finished before getting started. There was a lot of overlap.

The first true digital computer was designed by the English mathematician Charles Babbage (1792–1871). Although Babbage spent most of his life and fortune trying to build his ‘‘analytical engine,’’ he never got it working properly because it was purely mechanical, and the technology of his day could not produce the required wheels, gears, and cogs to the high precision that he needed. Needless to say, the analytical engine did not have an operating system.

**First generation from 1945 to 1955 (Vacuum tubes)**

After Babbage’s unsuccessful efforts, little progress was made in constructing digital computers until the World War II period. In 1944, the Colossus was built and programmed by a group of scientists (including Alan Turing), and the ENIAC was built by William Mauchley at the University of Pennsylvania. Some were binary, some used vacuum tubes, some were programmable, but all were very primitive and took seconds to perform even the simplest calculation.


By the early 1950s, the routine had improved somewhat with the introduction of punched cards. It was now possible to write programs on cards and read them in instead of using plugboards; otherwise, the procedure was the same.

**Second generation from 1955 to 1965 (Transistors and Batch Systems)**

The introduction of the transistor in the mid-1950s changed the picture radically. Computers became reliable enough that they could be manufactured and sold to paying customers with the expectation that they would continue to function long enough to get some useful work done. For the first time, there was a clear separation between designers, builders, operators, programmers, and maintenance personnel. These machines, now called mainframes, were locked away in large, specially air-conditioned computer

rooms, with staffs of professional operators to run them. Only large corporations or major government agencies or universities could afford the multimillion-dollar price tag. Given the high cost of the equipment, it is not surprising that people quickly looked for ways to reduce the wasted time. The solution generally adopted was the batch system. The idea behind it was to collect a tray full of jobs in the input room and then read them onto a magnetic tape using a small inexpensive computer, such as the IBM 1401, which was quite good at reading cards, copying tapes, and printing output, but not at all good at numerical calculations. Other, much more expensive machines, such as the IBM 7094, were used for the real computing. This situation is shown in Fig. 1-3.

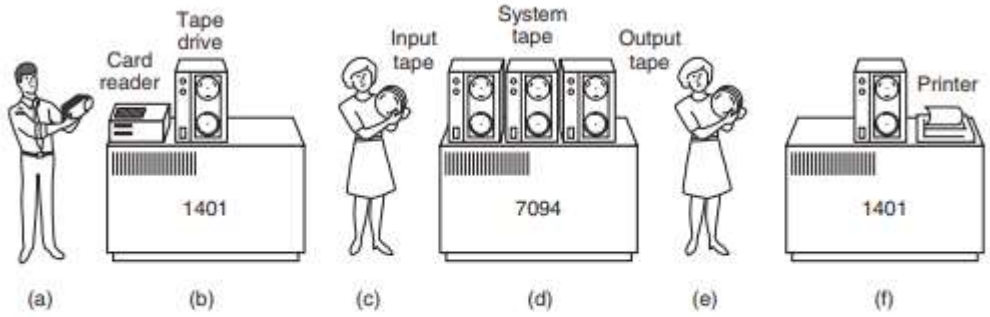


**Figure 1-3.** An early batch system. (a) Programmers bring cards to 1401. (b) 1401 reads batch of jobs onto tape. (c) Operator carries input tape to 7094. (d) 7094 does computing. (e) Operator carries output tape to 1401. (f) 1401 prints output.

The structure of a typical input job is shown in Fig. 1-4. It started out with a $JOB card, specifying the maximum run time in minutes, the account number to be charged, and the programmer's name. Then came a $FORTRAN card, telling the operating system to load the FORTRAN compiler from the system tape. It was directly followed by the program to be compiled, and then a $LOAD card, directing the operating system to load the object program just compiled. Next came the $RUN card, telling the operating system to run the program with the data following it. Finally, the $END card marked the end of the job. These primitive control cards were the forerunners of modern shells and command-line interpreters.

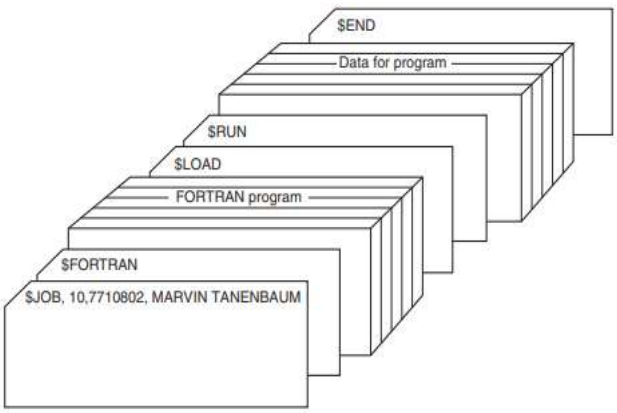


**Figure 1-4.** Structure of a typical FMS job.

**Third Generation from 1965 to 1980 (Multiprogramming)**

By the early 1960s, most computer manufacturers had two distinct, incompatible, product lines. On the one hand, there were the word-oriented, large-scale scientific computers, such as the 7094, which were used for industrial-strength numerical calculations in science and engineering.

Developing and maintaining two completely different product lines was an ex□pensive proposition for the manufacturers.

The IBM 360 was the first major computer line to use (small-scale) ICs (Integrated Circuits), thus providing a major price/performance advantage over the second-generation machines, which were built up from individual transistors. It was an immediate success, and the idea of a family of compatible computers was soon adopted by all the other major manufacturers. The descendants of these machines are still in use at computer centers today. Nowadays they are often used for managing huge databases (e.g., for airline reservation systems) or as servers for World Wide Web sites that must process thousands of requests per second.

Another major feature present in third-generation operating systems was the ability to read jobs from cards onto the disk as soon as they were brought to the computer room. Then, whenever a running job finished, the operating system could load a new job from the disk into the now-empty partition and run it. This technique is called spooling (from Simultaneous Peripheral Operation On Line) and was also used for output. With spooling, the 1401s were no longer needed, and much carrying of tapes disappeared.

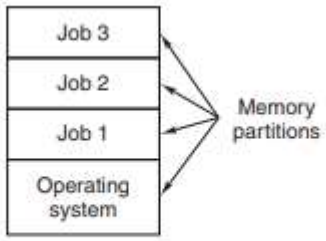


**Figure 1-5.** A multiprogramming system with three jobs in memory.