Lecture 10
Computer Software

Lecture 8 was on the binary number system and logic operations

Learning Goals for Today
1. To discuss the role of software in computing systems
2. To learn to differentiate among software belonging to the system and application categories
3. To learn about software ownership

We mentioned in Lecture 4 that at the highest level, two things are required for computing

Hardware: The physical equipment in a computing environment such as the computer and its peripheral devices (printers, speakers...)

Software: The set of instructions that operates various parts of the hardware. Also termed as “computer program”

Computer Software
- The HW needs SW to be useful; the SW needs HW to be useful
- When the user needs something done by the computer, he/she gives instructions in the form of SW to computer HW
- These instructions need to be written in a language that is readily understood by computer uP

Machine Language
A system of codes directly understandable by a computer’s CPU is termed this CPU’s native or machine language. Although machine code may seem similar to assembly language they are in fact two different types of languages. Machine code is composed only of the two binary digits 0 and 1.

Every CPU has its own machine language, although there is considerable overlap between some. If CPU A understands the full language of CPU B it is said that A is compatible with B. CPU B may not be compatible with CPU A, as A may know a few codes that B does not.

Language Translators
Human programmers write programs in a language that is easy to understand for them. They use language translators to convert that program into machine language. It converts the human understandable code in uPs understandable code, i.e. a language that is easy to understand for the uPs

Software Development
A software development process is a process used to develop computer software. It may be an ad hoc process, devised by the team for one project, but the term often refers to a standardised, documented methodology which has been used before on similar projects or one which is used habitually within an organisation.

The SW development process involves many steps, and coding, that is typing the instructions in a high-level language is only a small part of that process – taking-up only around 15% of the effort

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Operating System</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>Util</td>
<td>Language</td>
<td></td>
</tr>
<tr>
<td>Scientific</td>
<td>Business</td>
<td>Productivity</td>
</tr>
<tr>
<td>Entertainmen</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Software Development Process

Major Types of SW
- System SW
  System software is responsible for controlling, integrating, and managing the individual hardware components of a computer system. System software performs tasks like transferring data from memory to disk, or rendering text onto a display. Specific kinds of system software include loading programs, operating systems, device drivers, compilers, assemblers, linkers, and utilities. Software libraries that perform generic functions also tend to be regarded as system software. System software stored on non-volatile storage on integrated circuits is usually termed firmware. These generally perform the background tasks in a computer. These programs, many times, talk directly to the HW.
- Application SW
  Programs that generally interact with the user to perform work that is useful to the user. These programs generally talk to the HW through the assistance of system SW.

System SW are programs that ...
- Control the overall operation of the computer
  - OS
- Interact directly with HW
  - Device drivers
- Perform system management & maintenance
  - Utilities
- Are used to develop or maintain other programs
  - Language translators

Operating System
• It performs its work invisibly to control the internal functions of a computer, e.g. maintaining files on the disk drive, managing the screen, controlling which tasks the uP performs and in what order. It interacts directly with the computer HW. Other SW normally does not directly interact with the HW, e.g.

<table>
<thead>
<tr>
<th>Windows</th>
<th>Mac OS</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX</td>
<td>Solaris</td>
<td>DOS</td>
</tr>
<tr>
<td>CP/M</td>
<td>VMS</td>
<td>Firmware</td>
</tr>
</tbody>
</table>

ROM is a component of OS that permanently stored on a chip. It is a firmware program. When a computer is powered-on, it is the first program that it always executes. Firmware consists of startup and a few low-level I/O routines that assist the computer in finding out and executing the rest of the OS. On IBM-compatible PC’s, it is called BIOS

Utilities:
It is a small program that provides an addition to the capabilities provided by the operating system. In some usages, a utility is a special and nonessential part of the operating system. These are the computer programs that perform a particular function related to computer system management and maintenance

Examples:
1. Anti-virus SW
2. Data compression SW
   - Disk optimization SW
   - Disk backup SW

Language Translators
Programs that take code written in a HLL and translate it into a low-level language that is easily understood by the uP

1. Compiler translates the program written in a HLL in one go. The translated code is then used by the uP whenever the program needs to be run
2. Interpreter translates the HLL program one statement at a time. It reads a single statement, translates it into machine language and passes that machine language code to the uP and then translates the next statement, and so on …

Device Drivers
A device driver, often called a driver for short, is a computer program that is intended to allow another program (typically, an operating system) to interact with a hardware device. Think of a driver as a manual that gives the operating system (e.g., Windows) instructions on how to use a particular piece of hardware.

A device driver essentially converts the more general input/output instructions of the operating system to messages that the device type can understand.

Application SW
Application SW are programs that interact directly with the user for the performance of a certain type of work
• Scientific/engineering/graphics SW
   – Mathematica; AutoCad; Corel Draw
• Business SW
   – The billing system for the mobile phone company
• Productivity SW
Another way of classifying SW

- Shrink-Wrapped SW
  - You can just go to a shop and buy it

- Custom-built SW
  - You cannot just go to a shop and buy it; you have to find someone who can develop it for you

Shrink-Wrapped SW

SW built in such a way that it is useful for many different users in many different ways.
Example: MS Word. Individuals use it and so do many large corporations. It is used for writing one-page letters and also to typeset books

Custom-Built SW (1)

These SW are built for a particular organization to fulfill the needs of that particular organization. This type of SW is expensive because the builder has to recoup costs and make a profit from a single sale
Example: A system for predicting the preferences of the Northwest Airline pilots

Custom-Built SW (2)

This is another type of custom built SW. The delivery time is longer. Customers get more productivity out of it because it is built according to their exact specifications – just like a custom-built shoe fits better, but generally is more expensive, and requires a longer period for delivery

Who Owns Software?

- Generally, although a piece of SW that is being used by millions, it is not owned by any of them! Instead, it is owned by the maker of the SW

- The makers let us use their SW but keep the ownership to themselves. When we buy a SW package, we do not really buy it – we just buy a license that allows us to use it, the ownership stays with the maker

- However, there are variations on this theme …

Main types of SW licensees

- Proprietary – Most software on a Windows PC or a Macintosh belongs to this category
• Freeware – Most software on a Linux PC belongs to that category
• Shareware – the category which lies between the above two categories

**Proprietary SW License**

Proprietary software, as defined by the Free Software Foundation, means any software that is not free software or is only partially free. The modification, use and redistribution is prohibited, or requires express permission the originator. The user needs to pay the maker of the SW for buying a license that allows the user to use the SW. The license, generally, does not transfer the ownership of the SW; it just allows the user to use it. The user is legally barred from making copies of the licensed SW. Generally, the license is for the personal use only. Most SW in use in the world is of this type. Examples: Windows, Mac OS, MS Word, Adobe Photoshop, Norton Antivirus

**Types of Proprietary Licenses**

• Single-user license
• Multi-user license
• Concurrent-user license
• Site license

**Freeware SW License**

It is also known as "Public Domain SW". It allows the user to free use of the SW. The author, however, generally retains ownership. It can usually be downloaded from various Web sites.
Examples: Linux; LaTeX; Netscape Web browser – the Navigator; MS Web browser – the Internet Explorer

**Open-Source SW License**

Some authors give away the machine code only, which is extremely difficult to modify, if at all. Others even give away the high-level language source code so that users can make changes according to their own requirements. The later practice is called open-source licensing. Generally is any computer software whose source code is either in the public domain or, more commonly, is copyrighted by one or more persons/entities and distributed under an open-source license. Such a license may require that the source code be distributed along with the software, and that the source code be freely modifiable, with at most minor restrictions, such as a requirement to preserve the authors' names and copyright statement in the code.
Examples: Linux; Netscape Navigator

**Shareware SW License**

Shareware is software that is distributed without payment ahead of time as is common for proprietary software. Typically shareware software is obtained free of charge by downloading, thus allowing one to try out the program ahead of time. A shareware program is accompanied by a request for payment, and often payment is required per the terms of the license past a set period of time. shareware are similar in that they can be obtained and used without monetary cost. Usually shareware differs from open source software in that requests of voluntary "shareware fees" are made, often within the program itself, and in that source code for shareware programs is generally not available in a form that would allow others to extend the program. A shareware's program source, maintenance and extensibility can sometimes be negotiated for a licensing fee with the author(s) similar to standard proprietary software.
Examples: WinZip, Download Accelerator
**Trialware**
It is similar to shareware but difference is that the SW is usable for a short period only. When the period is expired, it is no more in use until the license is not purchased. The trial period may vary according to its developer. This period may range from a week to a few months.

It can be downloaded from the Internet or alternatively.

**What have we learnt today?**
1. We have found out about the role software plays in a computing environment
2. We also learned to distinguish between software belonging to the system and application categories
3. We also discussed the different types of software licenses

**Topics of some of the future lectures**
- Operating system
- Application SW
  - Productivity SW
    - Word processor
    - Spreadsheets
    - Presentation making
    - Databases
- Programming Languages
- The SW development process
- The Web development series of lectures is clearly focused on developing SW

**Focus of the Next Lecture**
- The role of the OS in a computing environment
- The various functions that an OS performs
- The main components of an OS
- Various types of OSes