Introduction to Computer Information Science

Tak Auyeung, Ph.D.

May 10, 2006

- 20060212 2324 TA: chapter 2 is almost all done. I still have to add some more about encryption and signing.
- 20060201 0104 TA: subsection 2.1.4 is done!
- 20060128 2205 TA: starting on chapter 2.
- 20060127 1037 TA: chapter 1 is finally finished! At least, I think so.
- 20060125 1037 TA: added more material, starting at 1.4
- 20060122 2347 TA: added more material, starting at 1.3.1
- 20060121 2342 TA: Added more material, starting at 1.2.3, up to and including 1.3.1
- 20060119 0035 TA: finish the first part of the first chapter (chapter 1). It was fun writing it since I get to research and learn stuff that I didn't know before. Cool. On the other hand, this chapter will be a bit longer than I originally planned. Oh, well!
- 20060118 TA: start the document. Hope the class is not canceled!

Contents

Chapter 1

Starting from Home

1.1 Personal Computers

There is no need to explain the term "personal computer" (PC) . Although there are different brands of PCs, they all share common attributes. For example, all PCs are affordable by individuals, and they are all small enough to be installed at home.

The following subsections discuss future trends of PCs.

1.1.1 More Personal

PCs will continue to become more personal. The price of a PC has dropped from US$2000 in 1980s down to about US$400 in 2006. This trend will most likely continue. For general purpose computing, it is possible that a PC can become as inexpensive as US$100 in the near future.

What this means is that in a household, every member of a family can have a PC. The configuration (hardware and software) of each PC can be fully customized to its owner.

Another aspect of becoming "more personal" is the size of a PC. Notebook computers are, generally speaking, more expensive and have less capabilities than their desktop counterparts. While this difference will continue, technology will be able to pack more processing power into notebook computers.

Combined with a concept called "thin client", PCs in the future can be devices as small as a cell phone. The screen will be integrated to a lightweight headset gear, and the "keyboard" will be a device that monitors hand/finger motion. We'll talk about thin clients at the end of the semester.

1.1.2 More Connected

Network access, particularly Internet access, is still at its infancy (as of 2006). Many home computers are either not connected, or connected via modems. High-speed and full-time Internet connection is still quite expensive, mostly because of the lack of competition and the cost of infrastructure.

This will change as wireless network technology improves. Certain carriers, such as Verizon, already offer wireless Internet service. New technologies will soon (in about 5 years of time) enable wide area (metropolitan level) high speed networking. The cost to set up a metropolitan wireless network is significantly less than that of a wired network. It is also possible to have multiple carriers offering competing services using wireless technology.

Perhaps even more importantly, wireless Internet access also means mobility. Combined with thin-client concepts, it will soon be possible to take a PC with you, and access content via the Internet no matter where you go (within the service area).

1.1.3 Where is my Computer?

PCs will soon "disappear", and you'll be left with a cell-phone sized device. The same device is your cell phone, digital camera, digital camcorder, music player, personal organizer and etc. The unification of personal electronics is inevitable. The only question is when, and how inexpensive.

1.1.4 Back to the Present

Of course, the sci-fi-like PCs mentioned in previous subsections are still years (10 years or so, I think) from now. For the time being, we have to continue our discussion based on PCs that we have today.

Bummer.

1.2 Hardware

The word "hardware" in the context of computers refer to physical components of a computer. This section is a very short description of various hardware on a PC, and how they connect/interact with each other.

1.2.1 Processing Component

The core of every PC is the "processing component". This component is responsible for the processing of data. This component can be broken into smaller components. The following is a list of these components, what they do, and how they interact with each other.

Processor

This is a single "chip" or IC (integrated circuit) in most computers. Some high-end computers do have multiple processor ICs. Examples of a processor includes Intel Pentium 4 and AMD Athlon.

The term CPU (central processing unit) is a bit outdated in PCs. This is because processor ICs include many components that, traditionally speaking, do not belong in a CPU. Such components include DMA controllers, interrupt controllers, timers and etc.

The capability of a processor is usually measured in two orthogonal properties. The first one is the "clock rate", and it is measured in GHz (giga Hertz) . One GHz means the clock of a processor cycles one billion (1,000,000,000) times per second. It is a *rough* estimate of the number of instructions that can be performed in a second. Normally, the higher the clock rate, the faster the processor. In other words, a 4GHz processor is faster than a 3GHz processor, *assuming everything else is identical.*

The second property is the width of data path in the ALU (arithmetic and logic unit) . This measures how big of a number the processor can compute for each clock cycle. As of 2006, we are currently transitioning from 32-bit processors to 64-bit processors. At the same clock rate, a 64-bit processor is *not* twice as fast as a 32-bit processor! Many factors determine the gain of a 64-bit processor compared to a 32-bit processor.

As much as a processor can crunch data (add, subtract, multiply, divide, compare, and etc.), it has very little ability to store data.

Memory

Since a processor cannot store much data, it is up to another component to do so during the execution of a program. The size of memory is often measured in MB (megabytes) . One mega byte is *roughtly* one million (1,000,000) bytes. To be more exact, one mega byte is actually 1,048,576 bytes. A byte is the size of the smallest individual addressable location in memory. A byte contains eight bits . "Bit" is derived from "binary digit".

The physical form of a piece of memory is a small printed circuit board (PCB) . Most memory modules are called DIMMs for "dual-inline memory module". Notebook computers use SODIMMs, which stands for "small outline dual-inline memory module".

The processor and memory of a computer are separate components inside a computer. Due to the physics of electricity, data cannot flow from a processor to/from memory as quickly as the processor can handle. Memory modules are rated based on how quickly they can interact with a processor. For example, a PC2100 400MHz DDR memory module can handle *up to* 400,000,000 interactions with the processor per second. DDR stands for "double data rate", which is a method that allows more data transfer between memory and processor compared to pre-DDR memory modules.

Chipset

In theory, a processor talks to memory modules directly. In reality, however, an interface sits between a processor and memory modules (and other components) in a computer. The interface is often called the "chipset" of a computer. It can consist of many ICs, or a single IC.

The use of a “chipset” permits flexibility. For example, to save money, a user can install slower memory in a PC and pay the price of performance. This is possible only because a user can change the configuration of the chipset through the BIOS (basic input/output system) settings.

The chipset of a PC is often subdivided into the “northbridge” and “southbridge”. With the processor at the north pole, the northbridge is “closer” to the processor, and it handles the high speed data pathways. The southbridge, being further away from the processor, is responsible for the lower speed data pathways.

Note that many chipsets are not merely reconfigurable and programmable pathways. Many chipsets also include integral circuits for video, sound and other traditionally individual input output devices.

1.2.2 Input/Output Component

With only the processing component of a computer, a computer is not very useful. This is because the processing component does not include any means to get data into a computer, or to get data out of a computer. The input/output component of a computer is responsible for getting data in and out of a computer.

The Input/Output (I/O) component is further broken down by connection method and purpose in the following subsections.

Connection: Chipset

As discussed earlier, the chipset of a computer can include I/O peripherals. The northbridge of a chipset, for example, often includes an integral video controller. This is because a video controller (otherwise known as the “graphics card”) requires a high speed connection to the processor.

The southbridge, on the other hand, may include an integral sound controller, an Ethernet controller and other low speed I/O controller.

When I/O peripherals are integrated into the chipset, the cost of a PC is lowered. At the same time, it is easier to assemble and test a PC. Most chipset-included peripherals can be disabled. This allows the owner of a PC to install custom I/O peripherals.

Connection: AGP

The AGP (accelerated graphics port) is a high-speed interface designed for interfacing with a graphics card. This interface, however, is currently begin phased out because of PCI express (see the next subsection).

Connection: PCI and PCI Express

The PCI (peripheral component interconnect) interface is the most standard (though slow) method of adding I/O peripherals to a PC. The main drawback of PCI is that it is a “parallel” bus. This makes it more difficult to design motherboards and maintain a high speed connection to I/O peripherals.

PCI Express is a new standard that is based on PCI from the software perspectives. However, PCI Express is a serial interface that supports 1, 4, 8 and 16 bits per clock of transfer. Due to the “serial” nature of PCI Express, it makes motherboards easier to design, and it also improves the speed of data transfer rate.

Due to the high data transfer rate of PCI Express, it is now replacing AGP as the most popular interface for connecting a video controller.

Note that PCI peripheral cards are not upward compatible with PCI Express slots in a PC. It is also true that PCI Express cards are not downward compatible with PCI slots.

Connection: USB

USB (universal serial bus) is the most popular standard for external low speed peripherals. USB 1.1 has a relatively slow maximum transfer rate at 12Mbps (mega bits per second). However, USB 2.0 has a much higher maximum transfer rate at 480Mbps.

Unlike most other connectors, USB connectors are designed to withstand a lot of insertions. These connectors are also designed to without a bit of abuse. Therefore, USB is now used for practically all external peripherals, ranging from printers, joysticks, keyboards, mice to external hard disk enclosures.

Connection: IDE

The IDE (integrated drive electronics) bus is also called the ATA (advanced technology attachment) bus. This is an older standard for connecting hard disks and CD/DVD drives to the southbridge of a chipset. IDE is a parallel interface that transmits 16 bits (2 bytes) of data for each clock cycle. Due to the parallel nature of the IDE bus, an IDE cable is usually thick and rigid. This reduces airflow inside a computer case, and it is also difficult to connect and disconnect components.

Due to the master/slave method, an IDE interface can only control up to two disk drives. Most PCs have two IDE interfaces, and hence the maximum number of disk drives in a PC is normally 4.

The IDE bus is now being phased out by the SATA bus.

Connection: SATA

The SATA (serial ATA) bus is a serialized IDE bus, but with additional enhancements. Despite the serial nature, SATA has a higher transfer rate compared to the fastest IDE standard. Furthermore, many SATA controllers also supports RAID (redundant array of inexpensive drives) for reliability.

Unlike IDE, an SATA interface is dedicated to a single disk drive. Rather than a restriction, this is actually a feature! Due to the lost cost of SATA controller ICs, connectors and cables, it is very cost effective to add ICs, connectors and cables for additional hard disks. The dedicated nature of SATA means that different disk drives in a PC *each* has a maximum transfer rate of 3Gbps for SATA II, and 1.5Gbps for SATA. This property of SATA makes it very suitable for RAID applications.

Other improvements of SATA (over IDE) also enhance the performance of SATA controllers and disk drives. A notable feature is called NCQ (native command queuing). It allows the disk drive optimize commands received from the controller, and hence improving the throughout of a hard disk. This feature is most noticable for applications that require a lot of disk accesses.

Due to the serial nature of SATA, the connection cables are fairly thin and easy to handle. This property also makes it possible to make external hard disk enclosures for SATA hard disks. At a transfer rate of 3Gbps, an SATA II external hard disk can transfer data 5 times faster than a USB external hard disk.

Device: Video Controller

The video controller of a computer is, to some people, the most important peripheral. While a low-end video controller costs as low as US$20 or so, a high-end one can cost as much as US$500 (or more!).

A video controller controls and drives the signal to display images on a monitor. The main difference between a high-end video controller and a low-end video controller is the GPU (graphics processing unit). A GPU is a highly specialized processor that accelerates many video related functions. Such functions include the following:

- 2D acceleration: moving regions of images from one location to another location.
- 2D acceleration: draw triangles (and other polygons) given only the vertices.
- 3D acceleration: general matrix computations, shading, texture mapping.
- Video acceleration: MPEG-2, MPEG-4, HDTV decoding.

Device: Sound Controller

Compared to the video controller, the sound controller of a PC is inexpensive and less demanding in terms of processing performance. Most sound controllers have jacks for microphones, line-in and speaker/line-out.

1.2.3 Storage Component

The storage component of a PC consists of electronic and mechanical parts that are used to provide non-volatile storage. This subsection describes the various parts for data storage.

Hard Disk

The most common and inexpensive storage device is a hard disk. The name hard disk refers to the rigid (hard) platters inside the device.

As of 2006, most PCs have tens to hundreds of GBs (giga bytes) of storage capacity on hard disks. The typical price point for a medium sized hard disk is about US$1 for two GBs.

The performance of a hard disk is determined by many factors. Here is a short and brief description of such factors:

- **Spindle Speed**. This is also known as the "RPM" of a hard disk. Most consumer grade desktop hard disks spin at 7200RPM, while notebook hard disks spin at 5400RPM. Some higher performance hard disks spin at 10,000RPM. The higher the spindle speed, the faster a hard disk can read all the parts of a "track".

- **Cache** . This is measured in MB (mega bytes). A typical consumer grade hard disk has 8MB of cache. The cache of a hard disk provides a mean to re-provide data that has been recently accessed, and hence save the mechanical components from performing much slower mechanical movements. As a result, a larger cache is preferred.

- **Bus**. The two consumer-grade alternatives are ATA and SATA. Low-end SATA has the same performance as high-end ATA. However, high-end SATA easily exceeds the capabilities of ATA. The main advantage of SATA (serial ATA) is the fact that SATA hard disks can optimize outstanding disk operations.

However, before you decide to upgrade your hard disk, you need to understand that the actual performance of a hard disk is limited by factors that are not hard disk related. For example, FAT32 file systems are prone to fragmentation. A badly fragmented file system slows down a system, almost independent of the performance of the actual hard disks.

As a result, instead of upgrading a hard disk for better performance, you can perform hard disk maintenance to improve hard disk performance. Defragmentation is one of the most important maintenance tasks for a computer system. In addition, an operating system may not fully recognize the abilities of a hard disk controller. This is especially the case if you put a system together. Making sure that the full capabilities of your hard disk controller is utilized is also an important factor.

CD/DVD drives

Although a hard disk is cost effective in terms of stationary non-volatile storage, CDs and DVDs are cost effective for portable data distribution. As of 2006, a 4.7GB DVD recordable disk is about US$0.30. This means for US$1, you can record about 15GB of data.

The following factors are important when you consider an optical drive:

- CD read speed: 52x is the current state of the art. 52x means an audio CD can be read at 52 times the normal play speed. It roughly translate to "a 52-minute music CD can be read in 1 minute".

- DVD read speed: 16x is the current state of the art. Needless to say, this is roughly saying "a 64-minute movie DVD can be read in 4 minutes".

- CD-R and CD-RW. Both types have capacities up to 700MB. CD-R is one-time recordable, while CD-RW is eraseable. 40x is a reasonable speed for recording CD-R, while 16x is a reasonable speed for recording CD-RW.

- DVD-R, DVD+R and DVD-RW. These types have capacities up to 8.7GB for double-layer capable drives, and 4.7GB for single-layer capable drives. A reasonable recording speed for DVD-R and DVD+R is 16x.

In additional to playing DVDs and CDs, recordable optical drives are also useful for "burning" CDs and DVDs.

1.3 Software

A PC with superior hardware components is useless unless it also has useful software. This section explores the various types of software that make a PC useful.

1.3.1 System Software

The term "system software" is defined rather loosely. Basically, it consists of "a collection of programs that provide a common foundation of services for all application software". In more specific terms, system software include the following subcomponents.

- Operating system (kernel).
- Modular Hardware driver.
- Modular services.
- Modular File systems.

Operating system kernel

The "kernel" of an operating system is, as the name implies, the core of an operating system. The design of an operating system kernel has impact to the security, reliability, efficiency and flexibility of an operating system.

It is well beyond the scope of this class to discuss the services provided by a kernel. It suffices to say, however, that the ability of an operating system to run multiple programs at the same time, and the ability to allocate memory resources are both responsibilities of the kernel. It is also up to the kernel to protect concurrent programs from each other.

Modular hardware driver

The kernel of an operating system is specific only to the processor of a computer and some of the other core components (such as memory and sometimes chipset). It does *not* include components to utilize I/O peripherals.

Software that interface the kernel of an operating system to I/O peripheral devices consists of device drivers. Device drivers are dependent on the periperals installed in a PC. Some common and basic device drivers are often "bundled" with an operating system to provide very basic services. Such device drivers include the drivers of the ATA bus, ATA hard disks, optical disks, basic video controllers and basic sound controllers.

Because the developer (and distributor) of an operating system cannot predict and account for all the possible peripherals that are and will be available, it is up to peripheral vendors to supply device drivers for their own products. Such devices drivers must be installed before the hardware devices can be used.

Modular services

A "service" is neither a device driver nor a part of the kernel of an operating system. A service assumes some basic functionalities to be provided by one or more device drivers, and a service provides an interface to the kernel.

Here is an example. When you install an Ethernet NIC (network interface card) in your PC, you may need to install a device driver. This device driver understands the integrated circuits used on the NIC, and provides basic abilities to receive and transmit data.

However, this does not mean that you can use this NIC to network with other computers. A device driver provides very low level functionalities. It does not provide high level functionalities, such as the ability to communicate with another computer using TCP/IP (transport control protocol/internet protocol). TCP/IP support is provided by a service.

A TCP/IP service, bound to the device driver of a NIC, provides an interface to the operating system kernel. This interface is in the form of special and ordinary file operations. To the kernel, the connection of a computer to another computer is viewed as a file that it can open, close, write to and read from.

It is important to understand that services are modular. In other words, it is possible to add additional services to an operating system. For example, most operating systems include IPv4 (Internet protocol, version 4) services. When the time comes, these operating systems can install a new service and support IPv6 (Internet protocol, version 6). Adding a new service should have no impact to the kernel nor device drivers.

Modular file systems

A file system is a collection of programs and libraries that provide the illusion of files and directories from physical mass storage devices or other devices. Most operating systems come preinstalled with one or more file systems. For example, Linux usually supports extension 2 (ext2) out of the box. Windows 98 supports FAT32 out of the box, while Windows XP supports FAT32 and NTFS out of the box.

Most modern operating systems, such as Windows and Linux, are modular enough to support additional file systems without having to patch the operating system. This modularity makes it possible for vendors to add support for additional file systems after an operating system is released.

1.3.2 Application Software

Application software provides value to users of a computer system. Although system software determines many attributes of a computer system, such as reliability and efficiency, it is up to application software to provide value. Without application software, even the best computer hardware and system software are useless.

Application software may, or may not, be published by the same vendors as system software. In the case of Microsoft, it develops and distributes both system software and application software. This practice, however, raises suspicion of internal collaboration. In other words, the system software branch of Microsoft can collaborate with the application software branch, and offer features only known internally. This makes it unfair for third party vendors who are not aware of the internally circulated system software features.

There many types of application software. The following subsubsections describe some of the more important ones.

Word Processor

A word processor is a program that “turns a computer into a fancy typewriter”. Of course, this is an understatement. Modern word processors can be used to author and format all kinds of document, from personal letters to resumes, term papers and even short books.

The most used word processor is probably Microsoft Word, which is a component of the Microsoft Office suite. An older popular word processor is WordPerfect. The original word processor that made the term “word processing” famous was Word Star. OpenOffice Writer is an open source word processor that has many of the same features as Microsoft Word. OpenOffice is an open source office suite, which means it is free to download, and free to install on any number of systems.

It should also be noted that OpenOffice is “ported” to many platforms. Windows, Mac OS X, Linux, FreeBSD are only a few operating systems that can run OpenOffice.

Spreadsheet

A spreadsheet is an electronic table, in which a cell can contain a formula that refers to the value of other cells. As a result, a spreadsheet is well suited for computation tasks of all kinds. The use of a spreadsheet requires almost no training of programming. In addition, a spreadsheet can also include charts of all kinds to display data graphically.

The first spreadsheet program was VisiCalc, but it was Lotus 1-2-3 that made spreadsheet a common application on PCs. Microsoft’s Excel (also a part of the Microsoft Office suite) is probably the most used spreadsheet program. Calc of OpenOffice is an open source alternative that shares many features with Excel.

Presentation Software

Although a word processor can produce well formatted printed document, it is not well suited for presentations. Presentation software is a class of programs that make computer-based presentations (using an A/V projector) functional, attractive and versatile.

Microsoft’s PowerPoint is the most popular presentation software. The Impress component of OpenOffice offers similar features as an open source program.

Database Frontend

A database front-end is a GUI (graphical user interface) program that makes accessing data from a database easy. A database frontend lets a user create database tables, formulate queries, design forms and generate reports. Most of these tasks can be done without any programming, using only visual tools.

Microsoft's Access is the most popular database frontend. Only recently does OpenOffice include its own database frontend called Base.

Email Client

An email client is a program that lets a user read and send email. A "pure" email client program is one that only lets a user read and send email. This kind of program do not exist anymore. Instead, most email clients also support newsgroup access.

Mozilla Thunderbird is an open source email client.

Communication Software

Email client programs are gradually replaced by communication software. A communication program can perform all the functions of an email client. However, a communication program can also handle many other tasks, such as scheduling, calendar and task tracking.

Microsoft Outlook is the most popular communcation program for personal and business use. It allows users schedule meetings and perform other "organizer" functions in the context of business. Evolution is an open source alternative to Outlook, providing many of the same features.

Photo Editor

Digital cameras have become more practical and less expensive than film cameras. As a result, photo editing programs become important tools for business and personal use. A photo editor provides a long list of features to edit pictures (mostly photographs).

The most popular photo editor (that has a "lite" version bundled with digital cameras) is Adobe's Photoshop. For personal use (as well as professional use), an open source alternative is the GIMP (GNU Image Manipulation Program).

Desktop Publishing

Although a word processor *can* be used to author a brochure, or a magazine, such tasks expose limitations of word processors. For professional publishing work, most professionals rely on desktop publishing software. A desktop publishing program offers a wide selection of tools to perform formatting, layout control, page control and other flexibilities required in paper-based publications.

The standard tool for desktop publishing used to be Adobe's PageMaker (originally developed by Aldus, which merged with Adobe). However, the current standard is Adobe's InDesign. For those who are short of money or gungho for open source solutions, Scribus is an open source alternative that works in Windows and many other platforms.

Illustration software

An illustration program is fundamentally different from a photo editing program. A photo editing program works on rasterized (pixel-based) images. On the other hand, an illustration program works on stroke-based art work. You can, in fact, see an illustration program as an electronic canvas. Art work produced by an illustration program are "scaleable", which means it can be enlarged to any size and still retain a smooth and detailed rendering.

There are many popular illustration programs. Adobe's Illustrator, for example, is widely used. Its popularity is followed by Freehand (also published by Aldus, but developed by Altsys, then merged with Macromedia). On the open source side, OpenOffice draw provides seamless integration with other OpenOffice components. For more serious art work, however, Sodipodi and its competing sibling, Inkscape, are more powerful and use SVG (scalable vector graphics) as their output format.

For those who are not very talented in art, there are many commercial web sites that sell art work. On the other hand, you can also use open sourced clip art from sources such as http://www.openclipart.org.

1.4 The Internet

One increasingly important resource of a home computer is the connection to the Internet. This section briefly describes the Internet, and some of the more useful tasks one can perform using the Internet. Last, but not least, I will also mention certain things to watch out for when your computer is connected to the Internet.

1.4.1 The Term

"Internet" means the interconnection among computer networks. This is, essentially, what we know as the Internet (or "web"). Without explaining how, it suffices to say that the Internet permits a computer connect to another computer, and the two connected computers can exchange data.

1.4.2 Resources

The Internet, or rather computers connected to the Internet, provide a vast amount of resources. One can write a *thick* book to describe all the resources available through the Internet. In this class, however, we'll focus on just a few of the more important ones. This subsection introduces these useful resources. We'll discuss some of these topics later in a different light.

The "Web"

The "Web", in a more restrictive scope, refers to resources that are available via HTTP (hypertext transport protocol) servers. Essentially, it means the computers that can provide content to a "web browser". In more technical terms, a computer that can provide such content is called an HTTP server, whereas a computer that requests, save and/or display such content is called an HTTP client.

It is relatively inexpensive (try "free") and easy (try "WYSIWYG") to author and publish content to the web. As a result, the amount of content accessible through the web is simply enormous. Locating a web page can be a very difficult task. Fortunately, there are search engines to help us with web searching. A search engine is a specialized web site that automatically "crawl" through the web, and index all the web pages, images and video clips that it encounters.

Google, Yahoo and Microsoft Network (MSN) all have a search engine interface. Learning how to use a search engine effectively is a topic that deserves its own book. Nonetheless, one can begin with very simple search queries, and gradually start to learn the advanced options.

Another important use of the web is eCommerce. eCommerce, in general, simply means conducting business "online". This is usually means, for the average user, is that we can purchase (and sell) by using a web browser to visit certain web sites.

Famous (or sometimes infamous) web sites for purchasing and selling include Amazon, eBay, buy.com and many other online merchants. For those who want to invest, it is also quite easy to access a bank, investment accounts and etc. via a web browser.

Email

For some people, this is actually a "web" topic. This is because most email services (gmail, yahoo, hotmail and etc.) are now accessible through a web interface (using a web browser). However, some people still prefer to access email using dedicated email clients.

Regardless of the access method, email is just what the name implies: electronic mail. It means the sender and recipient of a message do not need to be online at the same time. A sender can go online, send a message, and shut down the computer. A recipient can then turn on a computer, connect to the Internet, and receive the message at any point of time.

Email is widely used for personal and business purposes. While email does not replace telephone, fax and other conventional means of communication, it does offer a more convenient method for people to communicate.

Peer-to-peer

Peer-to-peer (P2P) is a method to distribute lots of content efficiently using voluntary and non-dedicated machines. A P2P client (to download content) is also a P2P server (to provide content). The interesting part of some P2P clients, such as BitTorrent, is summarized as follows:

- Via a "tracker" server, a client knows which other computers are close to it in network sense.
- A client can download different pieces of the same file from different servers.
- A client can be a server at the same time, and serve only the parts of a file that it has already downloaded.
- Clients and servers are anonymous. Except for the IP address, clients and servers do not have additional information about each other. There is no need to set up login accounts.

- Servers are decentralized and dynamic. In other words, at any moment, a server can be taken off line, can get online.

Various attributes of P2P makes it a efficient tool for distributing large files, such as that of an image of copyrighted content (CD, DVD and etc.). This is why P2P is a controversial topic. However, P2P is also used frequently in legitimate applications. For example, many free operating systems (Linux, FreeBSD and etc.) are distributed in CDs and DVDs. The content of such CDs and DVDs are GPLed (General Public Licenced). In layman's term, this means such CDs are not subjected to the same copyright rules as music CD and DVDs.

IRC/IM

Instant messaging and internet relay chat (IRC) are methods by which people can type and read text messages in "real time". A chatter (one who chats) needs to sign up for an account with a chat room engine, then use a chat client program to connect to the chat room engine. After that, the chatter can join a chat room with many chatters, or have a private conversation with another chatter.

While IRC/IM is typically used by younger people for casual communication, it is gradually being used by businesses and other organizations. For example, an online photography store may have sales people participate chats with potential customers, and answer questions via the online interface.

Many open source software vendors utilize IRC/IM to let users self support a product. For example, Debian (a distribution of GNU/Linux) has many chatrooms for its distributions. Such chatrooms usually have expert users and novice users. The expert users answer questions from novice users when they have time. The cool part is that there is always some expert user logged in 24/7/365.

Warning!!!

No matter how you use the Internet, watch out for malware! Malware means "malicious software", and a malware program does *something* that is bad.

For more details about malware and how to prevent and repair damage, please take a CISS (security) class.

In short, you should do the following to prevent malware infection:

- Do not visit web sites that you are not familiar with. Or, at least try to avoid this!
- Keep your OS up-to-date. This can be done by installing the latest updates and patches from the vendor of the OS that you use.
- Install a virus checker, and periodically scan the entire system for malware.
- Keep the virus database of your virus checker up-to-date. Most virus checkers let you set up an interval to check and update the database.
- Install a spyware checker, and periodically scan the entire system for spyware.
- Keep the spyware database up-to-date.

Chapter 2

Servers on the Net

In the previous chapter, we discuss some tasks you can perform on the Internet. For example, you can shop, bank, communicate and get entertainment using the Internet. This chapter picks up from those topics. However, instead of just mentioning *what* services are available, we will discuss *how* things happen, at least from the networking perspectives.

2.1 TCP/IP and the Internet

The Internet is, litarally, the interconnection of computer networks. Physically, computers (DTE , or data *terminal* equipment) are connected by a network of routers, bridges, repeaters, cables and other infrastructure devices. However, to make the Internet useful, all the connected computers must use a common language and protocol to communicate with each other. Such a protocol is TCP/IP , which is actually two protocols, TCP is transport control protocol, whereas IP is Internet protocol.

The following subsections discuss more details about TCP/IP. If you are interested in these topics, you should definitely consider taking more CISN classes.

2.1.1 Addresses

Each computer in a IP network has an address. This address is a quad octet. In other words, it is a sequence of four numbers, with each number ranging from 0 to 255. Numbers in this quad octet are separated by periods. For example, 192.168.0.1 is a valid IP address.

Note that this quad octet notation is only useful for IP version 4. IP version 6 has a different notation for IP addresses.

A “host” is usually a computer. It is fairly intuitive to understand a host address is simply an address by which other computers can address that host.

A network address, on the other hand, is a “family” of addresses. In other words, a network address is the common part of the host addresses of all the computers in that network. Here is an example.

Let’s assume a home has a local area network (LAN). Three computers are connected in this network. The connection method is not important. Let’s name these hosts Tom, Dick and Harry. Because these three computers are physically connected to the same network, their addresses must have common part.

For most residential gateways, it is common to use 192.168.0.**0** as a network address. Here, the boldface **0** simply means that it is the part that uniquely distinguishes a host from another in the same network. So, we can assume that Tom has an IP address of 192.168.0.100, Dick has an IP of 192.168.0.101 and Harry has an IP address of 192.168.0.102. Certain numbers are reserved, and never be used for a host. In this case, 192.168.0.0 identifies the entire network, whereas 192.168.0.255 identifies a “broadcast address”, which means it cannot be assigned to a host.

2.1.2 Ports and Sockets

Each IP address identifies a host. This allows a computer on a network identifies and communicate with another computer. However, if we see an IP address as a phone number, then each computer can only communicate with one other computer, and can only carry on one conversation at any particular time. In reality, however, TCP/IP is far more flexible.

One feature that makes TCP/IP very flexible is the concept of a “port”. A “port” is not a physical entity. Instead, it is a logical entity that exists in software. A port has a number associated with it, ranging from 0 to 65535. Furthermore, when a computer initiates communicate with another computer, it must specify a port number. This means for any communication request, *both* an IP and a port number must be specified.

This is “kind of” like having 65536 individual extensions for a telephone number. Using the extension analogy, however, we are still limited to having up to one conversation per extension, giving us a total of 65536 parallel conversations for each phone number.

The concept of a socket makes this scheme even more flexible. It is difficult to find an analogy in reality to describe what a socket is. Essentially, it makes it possible to have an infinite (in theory) number of parallel conversations for each port for each IP address. In other words, an IP address and a port number are needed to desinate the recipient of a conversation. However, once the request to communicate is accepted, a computer creates a new virtual phone line for *that* conversation. This leaves the IP address and port number ready to receive the next request to communicate.

2.1.3 Inter-network Connectors

Let us continue the LAN set up from the previous subsubsection. Now, assume Tom can also connect to the Internet. The technology is not even important (cable modem, DSL, fiber optics or even dial-up). Many modern operating systems let Tom share this internet connection with Dick and Harry. In essence, Tom serves as a “gateway” for Dick and Harry. In network terms, Tom acts as a router.

Here comes the interesting part. Because Tom connects to the Internet directly, it is given a unique IP address that is known to the rest of the Internet. Let’s say this address is 52.69.12.7. This means Tom is known to the Internet as 52.69.12.7. However, to Dick and Harry, Tom is known as 192.168.0.100.

What is going on here?

As it turns out, the rule is not that every computer must have one (and only one) IP address. The rule specifies that each *interface* must have one-and-only-one IP address. Tom has two interfaces. One interface for the LAN (to communicate with Dick and Harry), and that interface has an IP address of 192.168.0.100. However, the other interface, be it a modem or another network interface card, is used to connect to the Internet. *That* interface has the address of 52.69.12.7.

As a result, the Internet knows Tom has 52.69.12.7, but Dick and Harry knows Tom has 192.168.0.100.

Because Tom is in the unique position of being between the Internet and the LAN, it can do some pretty interesting tricks.

2.1.4 Network Address Translation

NAT (network address translation) is an interesting trick that allows Dick and Harry connect to the Internet through Tom in our example.

An IP connection connects from a source port from a requesting computer to a destination port of the destination computer. This means that when Harry wants to connect to yahoo.com, it picks a source port, say 15923, and attempts to connect to a destination port 80 of the IP address 66.94.234.13.

This is where it gets interesting. The following is the sequence of events:

- Harry tries to connect from its IP address, 192.168.0.102 and from port 15923, to port 80 of 66.94.234.13.
- Since 66.94.234.13 is not in the LAN, Harry decides to ask Tom to “forward” the request. When Harry talks to Tom, Tom’s IP address is 192.168.0.100.
- Here comes the cool part. Tom receives the request from Harry. Tom records both the port number (15923) and the IP address (192.168.0.102) of Harry.
- Tom proceeds to forward the request (to connect) to the internet via it’s IP address 52.69.12.7, and port that Tom chooses, say 5829.
- Yahoo.com receives the request *as if it originated from Tom.* Yahoo replies to Tom as 52.69.12.7 via port 5829.
- Tom recognizes that port 5829 corresponds to a forwarded request from Harry via port 15923.
- Tom forwards the reply from yahoo.com (66.94.234.13) to port 15923 of Harry (192.168.0.102).
- Harry receives the reply *as if it received it directly from yahoo.com.*

The key of this trick is that Tom tracks the the source port from Harry, and translate that port to one of its own ports. This way, all activities of Tom's exterior port (5829) are relayed to Tom's interior port (15923), and vice versa.

Note that there are different variants of NAT. For the purposes of this class, however, we will not get into variants of NAT.

2.2 Email

Email is a service that has existed since the early days of the Internet. Despite more modern electronic means of communication, email is still very popular. This is because email is *not* a real-time method of communication. In other words, the sender and recipient of a message do not need to be online at the same time.

This section discusses email from a network technical standpoint. We'll explore the general networking protocols used for email purposes.

2.2.1 The Overall Picture

Sending an email message is, as it turns out, a somewhat complicated procedure. Here is the sequence of events:

- A sender writes a message on his/her computer, and press the "Send" button.
- The sender's computer sends the message using SMTP (simple mail transport protocol) to an SMTP server owned by the sender's ISP (internet service provider). Let's call this SMTP Server 1.
- SMTP Server 1 sends the message via SMTP to an SMTP server owned by the ISP of the recipient, call that SMTP Server 2.
- The message stays on SMTP Server 2.
- The recipient goes online, and retrieves the message from SMTP Server 2. The protocol for this purpose can be POP3 (post office protocol 3) or IMAP (internet message access protocol).

Although this scheme seems too complicated, it is necessary. The two SMTP servers are necessary. This way, in case SMTP Server 2 is down for any reason, the message stays on SMTP Server 1 as it reattempts to deliver the message to SMTP Server 2.

2.2.2 POP3 versus IMAP

POP3 is an older protocol for message retrieval, whereas IMAP is a newer protocol. IMAP is a more flexible protocol, allowing on-server message management. This makes it possible to check email from multiple computers (home, work, school).

POP3, on the other hand, usually removes a message from the server once it is retrieved. You can set up an email program *not* to remove messages from a server and just retrieve them. However, a POP3 server cannot track which message has been read, unlike an IMAP server.

2.3 HTTP (Web) Serving

By comparison, HTTP serving is a bit easier. Well, sort of.

2.3.1 Basic Web Serving

Basic web serving is fairly straight forward. When you click a hyperlink or the ENTER key after keying in an URL (universal resource locator).

This is the sequence of events that happen:

- The requesting computer, called the "client", sends request to the destination computer, called the "server".
- The server computer receives the request, looks for the requested file, and replies back to the client.
- The client receives the document, displays it on the screen, and the transaction is complete.

Simple, eh?

2.3.2 Scripting

What was described in the previous section was, well, an over simplication of what actually happens for many HTTP resources. The middle step, which read "looks for the requested file, and replies back to the client" can be much more complicated.

Although an URL can specify a file, it can also specify the invocation (execution) of a script. In addition, an URL can specify pieces of information, called parameters, for a script to execute.

This means that when a server receives a request that refers to a script, the HTTP server (as a program) triggers the execution of a program, called a CGI (common gateway interface) program. The output of this program, in return, becomes the "web page" to be sent from the server to the client, and be eventually displayed on a "browser".

This may sound simple, but the deceptive simplicity is due to the simple words of "output of a CGI program". In reality, a CGI program can be very complex. If you are interested in writing CGI programs, you may want to consider first taking CISW300, followed by CISW410 or CISW420.

2.3.3 HTTP Thin Client

Before we talk about thin clients , we should first talk about thick or fat clients. A fat client is a dedicated program that is used for some form of data entry or retrieval. For example, a program that permits a real estate agent look up a database for houses. Or, a program that permits a financial advisor to pull up the performance profiles of mutual funds for a side-by-side comparison.

In the past, such programs are "fat clients" because they are specialized and dedicated. This means that a real estate agent or a personal financial officer need to install such fat client programs on all computers that he or she may use for the job. Not only is this approach inconvenient, but it also passes the responsibilities of upgrades and proper installation to the user.

However, with CGI programming and capable browsers, it is now possible to use web browsers as "thin clients". Instead of having a real estate agent or a financial advisor install specialized programs, they can rely on a web browser on any computer to do their jobs.

That's right!

The use of web-based (web-browser based) thin clients is fairly common now. Not only does it remove the responsibility to install and upgrade software from the users, but it also makes the release of new version easier for developers.

2.4 eCommerce

eCommerce is the act of conducting commerce via computers and computer networking.

This is a fairly vague description. It includes the sales of items. For example, in the case of amazon.com and many online merchants, it is the sales of actual products and goods.

For others, eCommerce is online banking. This allows clients of a bank perform operations such as balance checking, bill pay and investment online.

Yet, for some others, eCommerce is the sales of items through companies such as eBay.

2.5 Privacy and Security

Privacy and security are not terms invented after the Internet. However, they become more important and more controversial in the age of computers and computer networking. Before computers are available, evedropping and wiretapping can only be done manually. This means it was practically almost impossible to evedrop or wiretap *everyone*.

Thanks to computers, however, it is now possible to evedrop and wiretap every packet ever transmitted on the Internet. The implication is significant.

2.5.1 Privacy

One definition of "privacy" is simply "the quality of being secluded from the presence or view of others". In a more restricted sense, "privacy" also means "where only the intended recipients can read a message".

In the context of online (networked) activities, privacy mostly translates to content exclusion with the only exception of the sender and intended recipient. However, privacy also includes the exclusion of any data and information from others unless there is prior and explicit approval from the owner of such data and information.

There are clear cases in which privacy should be respected. For example, let's say you are done with your taxes using TurboTax of a comparable program. Is it okay for a third party to look around in your computer, locate the TurboTax data file, and upload it to the Internet?

Or, is it okay for your friends to publish what they know about you on the Internet, without prior concent from you? Your friends may not post any false statements, but some of those statements can still be very personal that you may not want to share with the whole world. For some people, it can be their weight, for others, it can be their religious belief. It can be something that can cause unnecessary embarrassment, or something that can cause bodily harm in a not-so-democratic society.

2.5.2 Security

There are many "general" definitions of security. For this class, however, the best one is "Freedom from risk; safety" (as quoted from 1913 Webster).

In the limited context of computing, security means safety from having a computer compromised, and freedom from risks of data loss and information theft. In a wider context, however, security also means freedom from *all* risk.

Here is a question. If Jack is suspected of participating in a plan to cause harm to many people, for the maintenance of security, is it okay to read all his email messages and even remotely access his computer to look for evidence?

The answer is clearly yes.

However, in a democratic society with means for checks and balances, any activity to invade Jack's privacy requires the prior authorization of a court. In other words, the police, FBI or any department of the *executive* branch needs to get the authorization of the *judiciary* branch, based on laws that are maintained by the *legislative* branch. The executive branch is responsible for the execution of the invasion of privacy. The judiciary branch is responsible for the interpretation of the law, and determine whether the submitted evidence is strong enough to warrant invasion of privacy. The legislative branch is responsible for the maintenance of laws, creating new ones and abolishing old ones as necessary.

This approach may be seen by some people as cumbersome, inefficient and even risky. Why can't we permit a department of the executive branch carry out evedropping or wiretapping based on evidence that it has collected, and based on its assessment of the potential level of harm?

Any means to circumvent the judiciary and legislative branches will eventually lead to the abuse of power because there is no way to cross check whether the invasion of privacy is justified or not.

2.5.3 Encryption

Encryption is a technique that ensures a certain degree of privacy. Most encryption schemes are based on "key-pair" encryption. In key-pair encryption, there are two "keys". One key is called a private key, and other key is called a public key. Content encrypted by one key can be deciphered only by *the other key*.

Key pair encryption makes it possible to freely distribute public keys. This is because all parties with only a public key cannot decipher each other's encrypted content. Only the party with a matching private key can decipher contents encrypted by a public key.

TLS (transport layer security) is a protocol that permits the encrypted "tunneling" of other protocols. It is widely used by online banks and other web resources that require privacy. HTTP can tunnel through TLS, and the result is often called HTTPS. In fact, to see if you are connected to a web site using a secure connection, you only have to check to see if the URL begins with HTTPS instead of HTTP.

Encryption is also extremely useful for email. This is because the operator of a email server (such as Hotmail, Yahoo! and etc.) can see the content of every message exchanged at the server. In order to preserve privacy, one can use end-to-end encryption. This makes sure that only the intended recipient can view the content of a message.

Unfortunately, end-to-end encryption requires that the recipient first disclose his/her public key to all potential senders, and a sender needs to look up the encryption (public) key for a recipient before sending a message. Some email clients, such as Thunderbird, has extensive support for end-to-end encryption that makes key management relatively easy.

Encryption forms the basis of other security related measures such as signatures, authentication and certificates.

2.5.4 Digital Signature

How do you know a message was really sent by the claimed sender? This is a difficult question to answer. As it turns out, it is fairly easy to create a message, and make it look like it was sent by a particular sender.

Encryption is not going to be helpful here because the public key of a recipient is, afterall, public. This means anyone could have used the public key to encrypt a faked message to the recipient.

Signing a message, however, is the answer to this problem. A message, regardless of its length, can be "summarized" by a fixed-length digest. The function to generate a digest for a message is public knowledge.

When a sender sends a message, he/she can generate a digest for a message. This digest is not useful by itself, as an imposter can generate a digest just as easily. However, a real sender can also encrypt the digest using his/her *private* key. This encrypted digest is sent along with the message itself.

When a recipient receives this message, the recipient can compute the digest. Then, the recipient can decipher the encrypted digest using the public key of the sender. If the deciphered digest matches the computed digest, then the message is authenticated.

2.5.5 Certificates and Authentication

A certificate is a file that contains important information for security purposes. It has the following pieces of information:

- Subject: this identifies the entity that uses the certificate. For a client-side user certificate, it is usually the email address of the client user. A server certificate lists the domain name of the server in the subject.
- Valid time frame: this defines when a certificate becomes valid, and when it becomes in valid.
- Public key: this is the key that can be distributed freely.
- Private key: this is the key that should be kept private.
- Issuer: this identifies who creates the certificate. A certificate should be signed by an issuer.

Note that not everyone can just make a certificate for oneself. For a certificate to be meaningful, it should be created by a Certificate Authority (CA) and signed by the same entity. Most browsers recognize a list of default CAs such as AOL, Thawte, VeriSign and others. A CA signed certificate means that an Authority has verified the identity of the subject of a certificate.

A certificate is very useful for authentication purposes. Let's take a look at the following conversation (A is trying to verify the identity of B):

- A: Who are you?
- B: I am B.
- A: How do I know you are really B?
- B: Here is a certificate, the subject is B, myself.
- A: How do I know that this certificate is not faked?
- B: The certificate is issued and signed by Thawte. You can get the public key of Thawte from their website.
- A: (Use the public key of Thawte to decipher the encrypted digest of B's certificate, and it matches the compute digest of the same certificate) Okay, you are B.

Note that in this example, B can be a client or a server. It does not really matter.

After authenticateion, a certificate is also frequently used to provide a public key for encryption purposes. Note that when a certificate is sent from the owner, the private key is *never* included.

Personal certificates (for email and client purposes) are often free, even from big names like Thawte and Verisign. However, server certificates (for web servers and other servers) can be quite expensive. If drtak.org was to get a certificate from Thawte, it would have costed US$150 annually.

Chapter 3

Content

This chapter discusses various file formats for different kinds of content.

3.1 Audio

3.1.1 What formats

There are many formats for audio contents. The following is a list of common formats, and a brief description for each format.

WAV

The WAV format is one of the most commonly used file format for audio content. WAV also has many variants, each has it own method of encoding audio data electronically.

The most commonly used encoding method is pulse-code modulation (PCM) . This encoding method is loseless. This means that a PCM encoded file reproduce audio signals exactly as before PCM encoding. However, PCM encoding requires a lot of storage space.

A less commonly known encoding method is adaptive differential pulse-code modulation (ADPCM) . ADPCM is used in situations where it is important to keep the size of files small. ADPCM is used often in handheld recording devices. Unlike PCM, ADPCM is a lossy encoding method. This means that in order to maintain the size requirement, the encoded data loses some of details of the original data.

Nowadays, the most common use of WAV files is for mastering CDs and other high quality sound sources.

WMA

Windows Media Audio (WMA) is a Microsoft proprietary compressed file format for audio content. Like the WAV format, WMA also has many variants. Currently, WMA supports both lossy and lossless compression, as well as specialized compression algorithms just for voice encoding.

The original WMA format has inferior sound quality compared to riveling standards like MP3 and Ogg Vorbis. Microsoft has been pushing a new standard, called WMA pro. WMA pro has comparable quality compared to riveling standards, even compared to Apple's AAC.

MP3

MPEG Audio Layer 3 (MP3) was the first successful lossy high compression ratio audio format. MP3 permits an encoder to choose a “bit-rate”, which determines how many bits each second of music should encode to (on the average). A user can, therefore, trade storage space for quality, or vice versa.

MP3 is, perhaps, still one of the most popular audio formats for general music distribution. However, MP3 encoding is patented. This means *any* encoding software needs to get a license from the patent owner. Microsoft invented WMA because of this patent issue.

Nonetheless, there are plenty of free MP3 encoders. The open source community use LAME (LAME ain't MP3 encoder), and there many other shareware or freeware alternatives.

Ogg Vorbis

Ogg Vorbis actually refers to two related concepts. Ogg is a bitstream *container* format. It is a file specification used for encapsulating multiple types of content into one file. For example, a movie has a sound track, a video content, and possibly a caption channel. All three “channels” are bitstreams, and Ogg specifies how the bitstreams are encoded into a single file.

Vorbis, on the other hand, specifies how audio is encoded in a bitstream. It is an audio compression algorithm that is comparable to MP3, but does not infringe the patent of MP3. The quality of Vorbis is considered very high even among proprietary formats like WMA pro and AAC.

It should be noted that Vorbis is an open standard. This means the encoding method is published (not proprietary like WMA and AAC), and the method is not patented (like MP3). The result is countless refinement of the encoding method by enthusiasts, which contributes to the high quality of encoded music even at a modest bitrate.

In the past, Vorbis was only supported by PCs running Linux and FreeBSD. However, nowadays, Vorbis is also supported by Samsung, Rio, Neuros, Cowon and iRiver. It is also used for encoding music in many commercial video games.

FLAC

Free Lossless Audio Codec (FLAC) is exactly what its name implies. For audiophiles who cannot tolerate the artifacts of lossy compression methods, FLAC offers a free and open compression alternative to MP3, WMA and even Vorbis.

An audio file can usually be compressed to about 10% of the original size with a lossy method without any detectable distortion to an average listener. By comparison, FLAC can only compress an audio file to about 50% of the original size.

FLAC is not the only lossless audio compression method. WMA Pro has an option for lossless compression. However, regardless of the format, lossless compression typically can only compress to about one half of the size of the original file.

Speex

Speex is a lossy compression method that is specifically designed for encoding voice (not music or general audio). When used for voice compression, Speex can achieve a very high compression ratio while preserving the audibility of the encoded signal. As an example, 75 minutes of lecture usually compress to less than 5MB of file. This translates to about 1.2kBps (1.2 kilo bytes per second), which is about 1/4 of the data transmission speed of a modem.

Like FLAC and Vorbis, Speex is an open standard.

3.1.2 Editing

Audacity is an open source audio editor/recording program. It is available for Windows systems as well as many other operating systems. Go to http://audacity.sourceforge.net for more information about and download Audacity.

For editing purposes, it is best to use uncompressed formats. This is because each time a lossy audio file is saved and reopened, more information is lost in the process. The WAV format is commonly used for this purpose. It does generate rather big files, but even such files are relatively small compared to the capacity of a modern hard disk.

3.1.3 Streaming

Audio files tend to be relatively large. It is inconvenient to have to transmit an entire file because an audio source can be played. This is why many audio providers use a technique called streaming.

Streaming is a technique by which the content of a file is sent serially. However, the receiver does not wait until the entire file is received before playing the content. Streaming requires both the server (content provider) and client (content player) understand the same streaming protocol.

Several protocols are commonly used for content streaming. The Real-time Transport Protocol (RTP) uses both user datagram protocol (UDP) and transport/control protocol (TCP) to accomplish streaming. However, the Real Time Streaming Protocol (RTSP) and the Real Time Control Protocol (RTCP) only use UDP.

Because UDP is connectionless, streaming protocols using UDP are often blocked by firewalls, including common residential gateways. That’s why some streaming servers use standard HTTP for streaming. Icecast is a open source implementation of a streaming protocol. It is, in part, a reimplementation of a proprietary streaming server called SHOUTcast .

Audio streaming is commonly used by online ”radio stations” to broadcast live programs and recorded programs. It can also be used for schools to broadcast lectures online.

3.2 Video

3.2.1 What formats

AVI

Microsoft’s Audio Video Interleave (AVI) is not a video format, but rather a container format. In other words, it is similar to OGG in terms of functionality.

DV

DV is a video encoding method used by Mini-DV and Digital8 camcorders. Video data is not really compressed in this format, resulting in huge files. For example, 10 minutes of video is encoded as 2GB files. This is rather inconvenient, as FAT32 has a file size limit of 2GB. Consequently, video editing software divide a long video stream into chunks of 2GB, and logically connect the chunks together as a video clip.

One nice property of DV is that it is lossless. In other words, video data encoded in DV represents the original video stream without losing any information. As a result, DV is commonly used for video editing.

MPEG-2

MPEG-2 (Motion Picture Expert Group 2) is a group of video and audio coding standards. MPEG-2 is the encoding standard used on DVDs. Compared to DV, MPEG-2 is much more space efficient. However, MPEG-2 is a lossy encoding method. This means the video of a DVD does not contain as much details as the original video source.

The compression of MPEG-2 is quite impressive. 2 hours of video data (production movie) compresses to 6 to 8 GB. 2 hours of video encoded by DV requires 48GB. As a result, MPEG-2 has a compression ratio of about 6:1 to 8:1 compared to DV.

MPEG-4

MPEG-4 is a successor of MPEG-2. One main feature of MPEG-4 is the advanced simple profile (ASP) compression.

Note that MPEG-4 include many “codecs”. In other words, there are various codecs (coder/decoder) software that implements various aspects of MPEG-4. Each codec does not have to be compatible with other codecs. The most notable codecs are DivX and XviD.

Note that MPEG-4 is, like, MPEG-2, a lossy video encoding method. Different MPEG-4 codecs, however, produce video in different levels of quality given the same compression ratio. XviD, for example, has been shown to be better than DivX at the same compression level.

MPEG-4 has at least two encoding algorithms. The first one, MPEG4-2 is easier in terms of processing power (for encoding), but the quality is only average. The new standard, MPEG4-10, requires 10 times more processing power for encoding, but the quality at the same compression ratio is considerably better.

Many solid-state camcorders now encode video into MPEG-4 directly because MPEG-4 is much more space efficient compared to MPEG-2. Because of the lack of processing power, such camcorders use MPEG4-2 as an encoding algorithm, which means the quality of the resulting video files is only average.

XviD and DivX

XviD and DivX are two video encoding methods that have a common root. Both are MPEG4-2 type encoding methods, which makes them more efficient than many other methods (such as MPEG-2).

In the beginning, there was a project called OpenDivX, which was an open source project to implement an MPEG-4 codec. As the project progressed, the hosting company decided to close the source code. This act upset many volunteer developers, which started the project XviD (DivX spelled backwards). XviD has since been an open source project.

3.2.2 Transcoding

Given video content in a particular format, it is possible to convert the format (or encoding method) to another format. This is called transcoding. For example, in order to make a DVD out of a tape from a camcorder, the encoding method needs to changed from DV to MPEG-2. For online distribution, however, it is more common to convert to MPEG-4 using DivX or XviD codecs because the resulting file size is much smaller.

An open source (and therefore free) transcoding software suite is "Gordian Knot" (http://www.autogk.me.uk).

3.2.3 Editing

Video editing is a type of highly specialized software. You can visit stores and find some commerical (proprietary) programs for video editing. Such programs are often quite expensive for home use.

For frugal producers, there are some free choices. For example, Zwei-Stein (http://www.zs4.net is a video editor that is available for Windows or Linux.

There are some options that are X11 (Linux, FreeBSD and MacOS X) only. Cinelerra (http://heroinewarrior.com/cinelerra.php3) is a mature GPLed program. It includes capabilities that rival some commercial video editors, such as H.264 (MPEG4-10) encoding. Another mature free video editing program is avidemux (http://www.avidemux.org).

3.2.4 Streaming

Video streaming is similar to audio streaming, with the exception of speed requirement. Because video files tend to be larger, video streaming typically requires more "bandwidth" (number of bits transmitted per second).

Most video file formats, such as AVI and MPEG, are designed for streaming. In other words, such file formats are designed so that it is possible to start playing a video file even though only a part of it is available.

3.3 HTML

HyperText Markup Language (HTML) is a plain text syntax for web documents. HTML is an important standard for the world wide web because HTML documents act as the glue for many files (video, audio and etc.) that are distributed over the Internet.

3.3.1 What formats

HTML has many versions and related languages.

HTML

The current version of HTML is 4.01. It is a subset of Standard Generalized Markup Language (SGML). The language is maintained by the World Wide Web Consortium (W3C).

XHTML

XHTML is Extensible HyperText Markup Language. The current version (as of 2006/3/6) is 1.1, although version 2.0 is currently in draft stage.

The main objective of XHTML is to enable mobile devices (such as portable digital assistant (PDA) and cell phone devices) to display hypertext documents. As such, the XHTML standard is less complex

XHTML is more strict compared to HTML. In other words, an XHTML interpretor assume full compliance to the syntax specified. Most browsers allow HTML documents not to be "well formed".

XML

Extensible Markup Language (XML) is not exactly in the same family as HTML and XHTML. However, it is worth mentioning here because it can be interpreted by many browsers. Unlike HTML and XHTML, XML does not specify *how* content is displayed, it only specifies the structure and data of some "content".

The proper structure of an XML document can be described by a "schema", which specifies how various elements of an XML document can relate to each other. An older (but still widely in use) standard for specifying the structure of an XML document is Document Type Definition (DTD). Another standard to specify the proper structure of an XML document is called "RELAX NG", which is also widely in use.

Because an XML document only contains data, but not *how* the data should be rendered (displayed), an XML document requires an Extensible Stylesheet Language (XSL) document to describe the rules to display data in an XML document. The transformation of the data of an XML document into another form (for displaying on screen or in print) is called XSL Transformation (XSLT).

The significance of XML is huge. It can be used to encode the data of practically anything, from word processing documents (open document format, ODF) to hand-drawn graphics (scalable vector graphics, SVG). When a browser is equipped with the ability to interpret a "schema" and an XSL file, a browser becomes a flexible front-end to display web distributed data that does not need to be limited by the capabilities of HTML or XHTML.

Instead of performing XSLT on the browser side, which requires quite a bit of processing, it is often done on the server side. In other words, data or documents are not stored in HTML or XHTML, but in XML. When a document is requested, the server locates the schema file to validate the XML document, and then use an XSL file to transform the XML document into an HTML or XHTML document. This HTML or XHTML document is, then, replied to the requesting client.

3.3.2 Editing

HTML document editing can be done in many ways. Because an HTML document is actually just a text document, one can use plain text editors such as Notepad. However, a plain text editor lacks some important features. This is why HTML editing is often done with an editor that specializes in HTML editing.

There are several free editors that are capable of HTML editing. Mozilla (now known as SeaMonkey, http://www.mozilla.org/pro
comes with an integrated HTML editor called "Composer". If you do not use Mozilla, you can install "Nvu" instead at http://www.nvu.com. "Nvu" is based on "Composer", and shares many common features.

Yet another free HTML editor is "Amaya". Amaya is developed and published by W3C (World Wide Web Consortium), which is the governing body overseeing the HTML and XHTML standards. Amaya differs from Nvu and Composer because it include support for XHTML, XML, SVG and MathML. It is, therefore, possible embed all kinds of math symbols and scalable graphics in a document. Go to http://www.w3.org/Amaya to download it. Amaya, like Nvu and Composer, is available for many platforms, including Windows.

Though not open sourced like Nvu, Composer and Amaya, another free HTML editor is "Arachnophilia", which is available at http://www.arachnoid.com/arachnophilia/. Arachnophilia is available for any platform that runs Java. This means it is available for practically all computer systems.

On the commercial side, the most commonly used HTML editor is "Dreamweaver", published now by Adobe. "Frontpage" is a Microsoft HTML editor that is bundled with some versions of Microsoft Office.

Note that Dreamweaver is a rather expensive, big and slow program compared to the open source alternatives. On the other hand, Dreamweaver can embed server-side code as well as client-side code. Dreamweaver is designed to handle entire web sites, not individual web pages. Furthermore, Dreamweaver is also designed to handle server-side scripting and database access.

3.3.3 Publishing

In order to publish an HTML document (so the rest of the world can read it), you must have a "web space" first. Note that a "web space" is not the same as a domain name. A web space is simply some storage space that is web accessible, but you have little control over the name (URL) of the storage space.

For beginners who just want to experiment with HTML documents, there is free web space hosting. For example, http://www.freewebs.com is a company that provides basic web space for free. The company does charge monthly fees when an account is upgraded to have more features. Nonetheless, the basic account is free, and it is more than sufficient for beginners to experiment with.

Once a web space is established (through registration with a web space hosting company), HTML documents can be uploaded to the web space. Uploading (transferring) files used to done only with an FTP (file transfer protocol) client program. However, most web hosts now use a web-based interface. A web-based interface provides better security for the web host, and it is also easier for a beginner.

3.4 Copyright and Copyleft

The web (Internet, in general) revolutionizes methods of distribution content that can be digitized, such as music, movies, documents and etc. As a result, laws, ethics and methods of "traditional" content distribution are being challenged on a daily basis.

3.4.1 What is Copyright?

Copyright is literally "the right to copy". It was a system designed to *restrict* the publication of content. An author grants copyright to a publisher so that the publisher has the proper right, often exclusive, to publish content. The concept of copyright has a long history, all the back to Handel (the baroque composer).

In the U.S., the copyright clause gives the congress the power "to promote the progress of science and useful arts, by securing for limited times to authors and inventors the exclusive right to their respective writings and discoveries."

The main purpose of copyright is for the author to have control over the distribution of contents. In many cases, this translates to monetary gain. In some cases, copyright can lead to artificial hike of books, CDs, DVDs and etc.

This is evident in case of textbooks. The cost to print textbooks is no more than that of printing novels and other less expensive books. However, textbooks are expensive because publishers know that customers (students) do have have alternatives. Up to 15% of the price of a textbook translates to income of the original author. However, there is no royalty for the author from the sales of used books.

In poorer countries, the same textbook often is sold for much less than the price in a rich country. This is evidence indicating textbooks can be less expensive and still be profittable. In other words, copyright makes it possible to set the price based on demand (or how much people are willing and can pay). In return, the ability to set a high price to textbooks forms an artificial barrier for low income students to succeed in school.

3.4.2 Copyleft/Open License

Copyleft (as opposed to copyright) states that a work (program code, document, music and etc.) has the following properties:

- the work is freely distributed
- everyone who has a copy of the work can use, modify and redistribute the work

To prevent content theft for profit, most copyleft licenses state clearly that the acknowledgement of all preceding authors must be preserved in the work, and that a notice must be included in the work to indicate where it can be found for free. In addition, most copyleft licenses also require that all redistributed work include the original copyleft license so that anyone who receives a copy understand the copyleft concept.

Some copyleft licenses are "viral". This means that any derivation of copyleft work is *automatically* copyleft as well. This means if you take a copyleft document and makes changes to it, your modified version is automatically copyleft. You can keep it private for your own use with most copyleft licenses. However, as soon as you publish the document, it must conform with all the rules stated in the copyleft license.

The primary motivation of copyleft licenses is to make content (documents, music, video, program code and etc.) freely accessible to all so that the distributed work can be improved by anyone who wants to contribute. A prime example of this philosophy is Wikipedia (http://en.wikipedia.org. At this web site, documents (encyclopedia type) are authored, modified and maintained by anyone who wants to contribute.

Chapter 4

Overview of System Development

In the previous chapters, we explore topics as end users. Starting with this chapter, we will start to explore topics as developers, or people who design and develop information systems so that end users can use them.

This chapter is an introduction to system development. It introduces the concept of the “macro” and “micro” view of an information system (IS). There is no much technical detail in this chapter. Such details are deferred to the respective macro and micro view chapters.

4.1 Introduction

4.1.1 Information System

What exactly is an information system? There are many definitions. Generally speaking, an information system consists all necessary components to collect, sort, store, retrieve, display and report information. Such components include, but are not limited to individual computers, backup devices, networks, printers, policies, processes as well as human components.

In the context of this course, however, an IS excludes all human components. In other words, we focus only on the portion of an IS that is computer-based.

4.1.2 Examples

An example is the information system used by ARC to allow students to register, enroll and manage other course related activities. Such a system consists of the following components, at a minimum:

- Servers at the Los Rios District
- Workstations at various locations
- Policies regarding course enrollment and other related activities
- Personnel to maintain the computers and programs
- Personnel to assist students to use the online system
- Computer network to connect all the computers together

Note that students are not the only end users of this system. Administrators, for example, also use the same system to survey enrollment trends. Instructors use this system to print rosters and manage online grading. Counselors use this system to retrieve information about students before advice is formulated.

Another example is the information system used by a bank. Such an information system maintains information for all clients of a bank (such as balance, activities, etc.). However, such a system also serves tellers, bankers and many other types of end users. The information system, again, consists of many different components.

4.2 The Macro View

In the macro view of an IS, an IS is seen as a black box. The "macro" view of an IS concentrates on how the IS interacts and serves its intended end users. This section discusses some of the topics related to the macro view of an IS.

4.2.1 Users

"Users" refer to agents who are external to the information system who may interact with an IS. The identification of users of an IS is extremely important, as it is a starting point to define what an IS does.

In the case of a bank IS, users may include the following groups of agents.

- Customers. A customer is a person who users some services of a bank.
- Tellers. A teller is a person who needs limited access to the information of customers in order to provide basic face-to-face services.
- Bankers. A banker is a person who needs more access to the information of customers in order to provide advanced face-to-face services.
- Decision makers. A decision maker is a person who is in a position to decide the directions and business strategies for a bank. Such a person requires statistic information regarding the business of a bank, as well as detailed information about branches, personnel and customers of a bank.

The classification of users depends on many factors. First, it depends on the business and whom it serves. In the case of a bank, customers may further be classified into "home customer", "small business" and "large business". In the case of an enrollment management system, users may include "first-time students" and "re-entry students".

Second, the classification of users also depends on the internal operation of a business. For example, in the case of a bank, the classifications of "teller", "banker", "branch manager", "CEO" and etc. reflects the internal structure of the organization that uses an IS. In the case of a course enrollment management system, the internel user classifications may include "instructor", "dean", "counselor" and "administrator".

Third, the classification of users also includes agents who are not human. For example, in the case of a bank, a "user classification" may include "tax report system" to include computer systems operated by the Internal Revenue Services and the Franchise Tax Board.

4.2.2 Functions

After users are identified, the macro view of an IS also includes the identification of functions of the IS with respect to each classification of users. It is important to understand that at this level, we are only interested in "what" the IS does, but not "how" it does it.

For example, in the case of a "home customer" of a bank, the following functions must be provided by the IS:

- Balance display.
- Activity report.
- Transfer of money between accounts.
- Online bill pay.
- Support request.

As another example, in the case of a "teller" of a bank, the following functions must be provided the IS:

- Accept deposit.
- Process withdrawal.
- Report balance.
- Report activities.

- Issue bank checks.

In some cases, a classification of users may perform everything that another classification can, but also some additional tasks. In the case of a "banker", he/she expects all the functions provided to a "teller", but also the additional functions as follows:

- Open account.
- Close account.
- Change address.
- Add co-owner.
- Add power of attorney.

The identification of functions of an IS can be a very tedious process.

4.2.3 User Interfaces

The "macro view" of an IS also includes the design of user interfaces. An user interface can be a dialog box, the look of an application, or the design of web pages if the system is accessible via the web.

The design of an interface depends on many factors. The function for which it serves determines minimum features of an interface. However, ergonomics and other human factors differentiate a good design from bad designs.

4.2.4 Business Logic

The "macro view" of an IS includes the specification of business logic. Business logic specifies how to make decisions and how data flows between different steps of a function.

For example, in the case of a bank IS, the business logic may specify the following (in English): "if a customer attempts to pay more than 2 bills that are more than $200 each, freeze web access and send the customer a letter to contact the bank."

4.2.5 Other Specifications

The key of the "macro view" is that it consists of many "specifications". For the scope for this chapter, we will stop the discussion of the macro view here.

4.2.6 The Importance of Specifications

The importance of specificaitons is that these specifications are shared by different groups in a systems development team. The specifications keep everyone in the team "on the same page".

Groups in a systems development team include the following:

- Clients: a client is essentially the future owner of an IS. It is, therefore, particularly important to make sure the IS to be delivered is the IS that a client wants. A set of good specifications describe an IS sufficiently. Unfortunately, most clients are not technical enough to understand the diagrams and documents used in systems specifications. As a result, a systems analyst often needs to translate specifications back to English, let the client read, approve and sign off.

- Systems analysts: these people talk to the "clients" of an information system, and talk to the rest of the development team. Systems analysts understand the business of a client, then in return translate the requirements, business logic and etc. into specifications. Most systems analysts also understand some programming so they can also talk to programmers to figure out technical details or difficulties of an IS.

- Programmers: these people write programs so that a computer behaves as specified by the system behavior specifications. Computer programs are written in programming languages, which are, typically, unlike natural languages.

- Quality assurance: these people take the systems specifications, and derive test plans and check lists to make sure the implementation of an IS meets all the specifications.
- Technical writers: these people take the systems specifications, and create manuals and other documents for end users and other non-technical people.

4.2.7 Systems Analysts

The "macro view" is mostly how a systems analyst sees an IS. As mentioned already, a systems analyst talks to the "client" of an IS to figure out the requirements and expected behavior of an IS. Based on the information collected from a client, a systems analyst create systems specifications to describe, in technical and unambiguous teams, the IS.

Consequently, a systems analyst must have some (broad) knowledge of how particular businesses operate. For example, a systems analyst working on a bank IS should know the business of banking. Furthermore, a systems analyst must have the skills to acquire information from clients. People skills are a good starting point. However, it is equally important to have organization skills as well as logical thinking skills.

Recall that a systems analyst needs to create specifications for an IS. In most cases, such specifications are in the form of technical diagrams and documents. As a result, a systems analyst needs to specialized tools to create such diagrams and documents.

One of the most important tasks of a systems analyst is to specify the structure and behavior of an IS to the point that there is no ambiguation. This is necessary because the specifications of an IS is distributed to many parties, and such specifications (collectively) describe what the client is going to get. Any ambiguation in the specifications results in disagreement among a development team, as well as failure to meet the expectations of a client.

4.3 The Micro View

The "micro" view of an information system refers to the view of programmers (also known as developers or coders). A programmer specifies how a computer operates at the lowest level. In the context of systems development, a behavior of a program must meet the specifications of the IS.

Note that a "program" is often a very little piece of the overall IS. Most complex ISs require many programs deployed on each computer in the IS, and often multiple computers are needed to implement an IS.

In the context of systems development, the task of programming is essentially the translation of systems behavior specifications into the code written in a particular programming language. This section discusses several main topics related to programming.

4.3.1 Programming Languages

A program must be written in a particular programming language. Common programming languages include Visual Basic, C, C++, Java, Perl, PHP and Python.

A programming language often does not resemble natural languages. This is necessary to make programs more concise and less ambiguous. In addition, a computer requires a program to be written strictly according to the "syntax" of a programming language. The "syntax" of a programming language is, essentially, then grammar of the language. Unlike people reading natural language paragraphs and sentences, a computer cannot handle *any* grammatical mistakes.

4.3.2 Programmers

The tasks of a programmer are significantly different from those of a systems analyst. A programmer seldom needs to talk to a client, or to understand business logic. However, a programmer needs to be proficient at programming languages, and be efficient at translating systems behavior specifications into actual program code.

Chapter 5

The Macro View

This chapter discusses the macro view of an information system. If you are interested in the material in this chapter, you should consider taking CISP457 (Systems Analysis). This chapter only covers the material in a superficial way.

5.1 The UML

The Unified Modeling Language (UML) is a "language" used to specify various aspects of an IS (or any computer system). It was originally created by Rational Inc. with their Rational Rose product. However, the UML is now used as the defacto standard for systems specification.

The UML consists of many types of diagrams, each type to represent an aspect of systems specifications. We'll discuss some of the elementary ones in this chapter. As a "language", the UML uses very little text, which makes it somewhat easier to understand.

Technically speaking, one can draw UML diagrams with any tools, or no tools, at all. In reality, however, it is much more efficient to draw UML diagrams with specialized tools. Some of these tools are extremely expensive. However, there are free alternatives that are suitable for students. ArgoUML is an open source product, and Poseidon has a free version.

ArgoUML can be downloaded from http://argouml.tigris.org, and Poseidon can be downloaded from http://www.gentleware.com Note that both products require that you have a relatively recent version of Java already installed. For most computers, you can download and install Java from http://www.java.com.

5.2 The Use-Case Diagram

A use-case diagram documents both classifications of users as well as functions of the system. In a use-case diagram, a classification of user appears as a match-stick figure, while a function appears as an oval. Lines are drawn between functions and classifications of users to make associations.

In the UML, a classification of users is called an "actor" , while a function is called a "use case" (hence the name use-case diagram). It is important to distinguish a user from a classification of users. John Smith the student is a user, but he is only an *example* of the classification of "students".

5.3 State diagrams

A state diagram in the UML specifies the various states of objects, and how to transition from one state to another. This sounds pretty abstract, but we can illustrate the concepts of states using real-life examples.

Let's consider the example of a letter, a personal one. As soon as you pick up a piece of paper, and *decide* to write a letter on it, the blank piece of paper becomes a letter in its initial (beginning) state.

The next state is "work in progress", which reflects the fact that you are writing the letter. The letter stays in this state until you are ready to mail it. As soon as you fold up the letter, put it in an envelop and seal it, it is no longer "work in progress", but rather "in transit".

Of course, you can break up the "in transit" state to even more states. "On its way to a mailbox" and "processed by USPS" are two possible states. For our discussion, we'll just have one "in transit" state.

Once the letter is delivered at the recipient's address, it is no longer "in transit", but "delivered". The letter can stay in this state for days, weeks or months. As soon as it is opened and read, its state changes to "read".

If you want to make this analogy complete, you can add the states "archived" and "recycled" (or "shredded").

In a state diagram of the UML, you can represent each state of a letter as a bubble. Bubbles are connected by transitions. A transition is graphically represented as an arrow. A transition presents a few concepts.

First, a transition states an event that is external to the object that triggers the transition. In our example, the trigger for the transition from "initial" to "work in progress" is "start writing". Note that this event "start writing" is external to the letter.

Next, a transition can specify an action to be performed by the object as the transition occurs. In our example, a letter is an inanimate object, so it does not have an action associated with any of its transitions. In the case of an active object, such as a security alarm, the transition from "armed" to "tripped" can specify an action of "dial monitoring station" in response to the trigger "window sensor opened".

The key concept to remember here is that we are looking at the states of an object *from the perspectives of an object.* A trigger has an origin that is *external to the object.* However, an action is *performed by the object.*

State diagrams in the UML can get far more complicated. For the purpose of this class, however, we'll conclude the discussion of state diagrams here.

Chapter 6

The Micro View

In this chapter, we'll discuss the micro view of an information system. This is how programmers (also known as developers and coders) view an information system.

6.1 Activity Diagrams

Activity diagrams of the UML is particularly important to programmers. This is because an activity diagram captures the logic of how a task is done. A programmer reads an activity diagram, and translate the represented logic into code written in a particular programming language.

In an activity diagram, an activity is represented by a rounded rectangle. An activity is also known as a "state", but I use the term "activity" to avoid any confusion with a state in a state diagram.

An activity can lead to another activity using an arrow. The direction of the arrow indicates the order of activities. Note that the proper name of such an arrow is "transition", which makes it easy to be confused with a transition in a state diagram.

It is common to have to represent decisions in an activity diagram. A decision is represented by a diamond in an activity diagram. A diamond has one incoming arrow and two outgoing arrows. Each outgoing arrow of a decision is labeled to indicate in what condition that arrow will be chosen. Note that at a decision, one and only one outgoing arrow should be followed.

There are many other advanced concepts related activity diagrams. However, for the purpose of this course, we'll stop here. If you want to learn more about activity diagrams (or other UML diagrams), consider taking CISP457.

6.2 Basic Programming Concepts

6.2.1 Data: Variables and Values

In order for a program to process data, data must be stored in "variables" first. A variable has a name and a value. The value of a variable can change during the execution of a program.

Depending on the "type" of a variable, it can store different sorts of information. For example, if a variable is of "integer" type, it can only be used to store integer values. Some languages, such as Visual Basic, has a lot of different types. Others, such as Perl, has very few (but flexible) types.

However, variables are not the only sources of values. Values used in a program can be constants. Unlike that of a variable, the value of a constant cannot change. Constants can be named, or they can be "literal". The number 5234 is a literal integer constant.

6.2.2 Operators

An operator takes one, two or sometimes three values, and produce a single result. For example, the multiplication operator takes two numerical values, and produce a product, which is also a numerical value. We are all familiar with basic operators like addition, subtraction, multiplication and division.

There are also operators that do not produce a numerical answer. For example, the `>` (greater than) operator takes two numerical values, but produces a result of the nature of "yes or no". This type of result has a specific name: Boolean.

In other words, the *expression* "`x > y` " compares value `x` and value `y`, then produces a result to indicate whether `x` is really greater than `y`.

In general an expression is a construct that results in some kind of values. In our previous example, "`x > y` " is a Boolean expression. However "`x * y` " is a numerical expression. Boolean expressions are particularly interesting, as they are used in many control structures.

6.2.3 Assignment Statement

An assignment statement has two sides. The right hand side (RHS) is an expression that produces a value. The left hand side (LHS) specifies a variable. An assignment statement first evaluates the RHS, then overwrite the LHS with the result of the RHS.

In most programming languages, as assignment statement looks like this:

```
x = y * z
```

This means the program first evaluates and figure out the product of variable `y` and variable `z`, then the product overwrites the value of variable `x`. Note that a variable can only contain one value. Therefore, after this statement, the old value of `x` is lost, forever.

It should be noted that the previous statement is not stating that `x` equals `y * z`, it is copying the product, `y * z` to the variable `x`. In other words, it is not a comparison, it is a copy operation.

Assignment statements are the most basic statements because they cannot contain other statements.

6.2.4 Sequences

In most programming languages, the *order* of statements is important. Like most western languages, a program should be read from left to right, then top to bottom. A sequence is simply a collection of statements in which the order is specified.

However, in a sequence, a step can only follow the previous one. In other words, a sequence is essentially a straight line of steps to perform.

For example, the following statements form a sequence:

```
x = 3
y = x * 3
x = y + 2
```

Note that in this example, the order of the three assignment statements is significant. Any change of the ordering affects the result of the sequence.

6.2.5 Branches

Sequences are assignment statements form the basis of programs. However, such a program can only execute steps in a "straight line" fashion. A program can become smarter if it can perform different operations depending on some factors. In other words, it is better to have programs that can differentiate and handle different cases.

This is why most programming languages have branch constructs. A branch construct is also commonly known as a "conditional statement" . At the simplest level, a conditional statement permits a program evaluate a Boolean expression, then follow one of two possible paths depending whether the expression produces a value of true or false.

For example, the following conditional statement makes sure that variable `m` is the maximum of `x` and `y`:

```
if x > y then
  m = x
else
  m = y
end if
```

Even with conditional statement, a program can only move forward, and cannot go back to "do something again". This is why we have loops.

6.2.6 Loops

A loop in a programming language specifies a sequence of steps that can potentially be performed repeatedly. As with any repetition, a program must specify the following:

- What action is to be performed repeatedly?
- How do we stop the repetition and continue the next step?
- When do we check to see if we can exit a loop?

In Visual Basic, the most general form of a loop is as follows:

```
do
  x = x + 1
loop
```

This is an infinite loop, as it only specifies that we need to add one to x for each iteration, but it does not specify when we can stop doing this.

Except for *very* few special cases, loops should not be infinite. To make a loop *finite*, we can specify when a loop should continue. Or, we can also specify when to exit a loop. Furthermore, we can specify whether the condition should be checked before we perform an iteration, or checked after we perform an iteration.

To be consistent with other languages like C, C++ and Java, we only specify when to *stay in a loop*. This is done by the reserved word `while`.

For example, if we want to check the stay-in-loop condition before each iteration, we can change the loop to the following:

```
do while x < 5
  x = x + 1
loop
```

This means we check if `x` is really less than 5 before each iteration. As soon as `x` is no longer less than 5, the loop terminates. Note that in this case, if `x` has a value of at least 5 before the loop, *not a single iteration* will be performed.

Here is a loop that performs an iteration before checking the stay-in-loop condition:

```
do
  x = x + 1
loop while x < 5
```

The main difference of this program and the previous one is that in this case, even if `x` is at least 5 before the loop, it still gets incremented once. This is because an iteration is performed first, then the program checks the stay-in-loop condition.

6.3 Testing and Debugging

During the development of a program, it must be tested and debugged. This section discusses what it means to test and debug a program.

6.3.1 Activity Diagrams, Again

The behavior of a program is specified, in large, by activity diagrams. As long as activity diagrams are verified by a systems analyst, they describe the correct behavior of an information system. Consequently, activity diagrams form the basis of any tests of components of an information system.

6.3.2 Test Plans

From an activity diagram, test plans should be derived. Each step of a test plan has the following components:

- Test pattern.
- Proper response.

A test pattern is a specific input stimulus. It can vary from a specific value in a text box, to a specific network packet transmitted to an information system. For every specific test pattern, an information system (or program) should respond in an observable way. The proper response defines how a program should respond.

A test plan is essentially a sequence of test patterns and proper responses, interleaved. The concept of a test plan is relatively simple, the difficult part is the derivation of *good* test plans. A good set of test plans exercises most, if not all, of the logic of a program, and catches most, if not all, possible defects of a program.

There are programs that read an activity diagram, then *automatically generates* test plans. However, test plans that are generated tend to be tedious and wasteful of resources. This is especially the case as software becomes more and more complex.

For large programs that are not designed to be tested in a structured way, there is no practical way to automatically generate test plans. This applies to programs that are written 10 to 20 years ago. Many of these programs are extended time and again to the complexity that makes automatically generated test plans impractical.

This is why most programs are not tested by such test plans. This partially explains why programs still have flaws.

Most programs are tested with hand crafted test plans. Therefore, the success of test plans depend on the people who craft them.

6.3.3 Debugging

Debugging is "the process to identify and remove flaws in a program". This is, perhaps, the true black art of programming. Many programmers dread debugging because there is no known procedure to debug programs. The debug effectiveness of a programmer depends on many fuzzy factors such as experience, personality and etc.

A "bug" is, essentially, a flaw in a program. Although most people associate a "bug" with a system or program crashing, or hanging up, the true definition of a "bug" is simply "deviation from specified behavior". In other words, whenever a program does not behave as specified (by flowcharts or activity diagrams), it has a bug.

This means a well constructed set of test plans should at least help the identification of bugs. Once a bug is identified by quality assurance, programmers will locate the faulty code and fix it.

Chapter 7

Database

Database is an important concept in any information system. This chapter describes the nature of a database, and how to use one.

7.1 Tables

A database has any number (including zero!) of tables. A table, as the name implies, can be visualized in a tabular form. In this form, each column is called a “field”, while each row is called a “record”.

Each table can have any number of rows, but a table typically has at least one column.

Note that although a database table looks similar to a spreadsheet, it is not. The *implementation* of a database table is completely different from that of a spreadsheet. As a result, it is generally not a good idea to use a database table to represent a spreadsheet, or vice versa.

Let us consider a few tables. First, let us consider a table that stores information about students:

lastname	firstname	studentID	phone
Smith	John	0283934	372-4782
Peterson	Peter	2893911	657-2738
George	Georgia	7857292	628-2837

In this case, the “studentID” field is called a primary key. A primary key is a field in a table such that it is not empty and unique for each row. In this case, each student must have one student ID, and no two students may share the same student ID.

Next, let us consider a table to represent courses:

courseID	courseNum	title
58273	CISC310	Introduction to Computer Information Science
72838	CISP300	Algorithm Design and Problem Solving
82738	CISW420	Server Side Scripting

For enrollment purposes, we also need a table for individual class sections:

sectionID	courseID	classRm	days	start	end
18278	82738	129	TuTh	1730	2015
10283	72838	129	MW	1730	1850
12903	72838	121	TuTh	1100	1215

Here, the column “sectionID” is a primary key. However, the column “courseID” is called a foreign key. A foreign key is a key that is used to look up a row in another table. In this example, class section 10283 refers to course ID 72838. This means that 10283 is an offering of CISP300.

Last, we have a table to represent enrollments:

studentID	sectionID
7857292	10283
0283934	10283
0283934	18278

In this table, no column is a primary key. The first row means that student “Georgia George” is enrolled in the evening section of CISP300. The second row means that John Smith is also enrolled in the same class. However, John Smith is also enrolled in the evening class of CISW420.

7.2 Relational Databases

In the previous section, we saw a glimpse of the power of a relational database. In a relational database, a table can "link" to another table by a field. In our example, the enrollment table links to the student table using student ID, and it also links to the section table using the section ID. The section table, in return, links to the course table using the course ID.

This kind of linking is important from many perspectives. From the data storage point of view, this means we can save lots of space. In our example, there is no need to store the duplicate title of CISP300 for both sections. There is also no need to store the duplicate name of (and other information about) John Smith.

Most databases are relational databases.

7.3 SQL

Structured Query Language (SQL) is a common language used by almost all relational databases. It is not exactly a programming language, as it is incapable of specifying control structures. Nonetheless, it is a computer language that is very important.

The simplest query asks about information in a table:

```
select * from student;
```

This query selects "everything" (denoted by the asterisk symbol) from the table called "student". It prints every row stored in the table called "student".

We can refine what to retrieve using a "where" clause:

```
select * from student where student.lastname = "Smith";
```

In this example, the database only retrieves records for those students with a lastname of "Smith".

You can also specify just a few columns to print:

```
select student.firstname from student where student.lastname = "Smith";
```

Now, let's move on to some more interesting example. Let's say we want to find out who are enrolled in section 10283, and print out the firstname and lastname of those students. This can done with the following query:

```
select student.firstname, student.lastname
  from student INNER JOIN enrollment
  on student.studentID = enrollment.studentID
  where enrollment.sectionID = 10283;
```

Isn't this cool? But wait, we can make this even better. We can include the course table so that we can just refer to a section by its course number:

```
select student.firstname, student.lastname
  from student INNER JOIN enrollment INNER JOIN course
  on section.courseID = course.courseID AND
     student.studentID = enrollment.studentID
  where course.courseNum = "CISP300";
```

Chapter 8

Open Source Software

This chapter compares open source software to proprietary software. The concept of open source software is not, by any means, a recent development. However, it is only recently that open source software gain more visibility from general users.

8.1 What is Open Source?

The best definition can be found at http://www.opensource.org/docs/definition.php. In essense, open source means the source code of a (computer) program is available, for free.

What is the source code of a program?

In a nutshell, the source code of a program is like a detailed blueprint of a machine. Given the blueprint of a machine, the machine can be fabricated from individual components (or even raw materials). Likewise, given the source code of a program, a program can be created.

In the machinary analogy, it is necessary to have tools like a lathe, a mill and etc. However, in the case of software, the tools required are often available for free as well.

This means that *most* open source programs are available for free from the perspectives of an end user.

Note that open source software is not public domain. Although open source programs, and their source code, are available for free, certain restrictions still apply to the redistribution. For example, one restriction is that the authors' contribution notice be respected.

8.2 End User Oriented Open Source Programs

In the past, the types of open source programs were somewhat restricted to applications that did not concern end users. However, many open source programs are now end user oriented. This section attempts to enumerate a few categories of open source programs that benefit end users.

8.2.1 Office Productivity

Open Office, available at http://www.openoffice.org, is no doubt the most end-user oriented open source program. The current version (as of 2006/04/16) has the following main components:

- Writer: word processor.
- Calc: spreadsheet.
- Impress: presentation.
- Base: database frontend.

Open Office, in many ways, is comparable to Microsoft Office. This is especially the case for the average end users who do not need to use macros or Visual Basic for Applications (VBA). Many documents saved in Microsoft Office formats can be opened in Open Office.

8.2.2 The GIMP

The GIMP (GNU Image Manipulation Program) is a bitmap (image) editor. It has many features that are also found in Adobe Photoshop. While the GIMP is not suitable for professional photographers, it is suitable for the average end users.

You can download the GIMP (for Windows) at http://gimp-win.sourceforge.net/.

8.2.3 Audacity

Just as the GIMP is for image editing, Audacity is for sound editing and conversion. While Audacity is not intended for recording studio use, it can be used to record and combine multiple tracks.

For more information about the capabilities of Audacity and to download it, visit http://audacity.sourceforge.net/.

8.2.4 Sodipodi

Sodipodi (and its twin, Inkscape) is a program that allows you to draw artwork that is "scalable". A scalable artwork is one that can be enlarged to any size without getting pixelized, or becoming fuzzy.

For more information about Sodipodi and download it, visit http://www.sodipodi.com/index.php3.

8.2.5 Personal Finance

If you want to balance your check book, or to visualize spending habits, you can use Grisbi. It is an open source project originated from France, but translation of the software has been done for English users.

Visit http://www.grisbi.org/ for more information and to download it. An alternative program is GFP, which is available from http://gfd.sourceforge.net/. Note that GFP requires Java.

8.3 Infrastructure Open Source Programs

"Infrastructure" in this context simply means not intended for end users, and needs to be configured by someone other than an end-user. Many open source programs fall into this category. However, the most well known programs are known as LAMP. "L" for Linux, "A" for Apache, "M" for MySQL and "P" for either PHP or Perl.

8.3.1 Linux

Linux is a free operating system (OS). Although you can use it as an end-user desktop, it is mostly used on a "server" computer that is usually locked in a room somewhere.

A lot of infrastructure open source programs run on Linux. This is why it is important to know about Linux. A discussion of how to use Linux is way beyond the scope of this class. If you are interested in learning Linux, please take CISC323 and CISC324.

8.3.2 Apache

Apache is an HTTP (hypertext transport protocol) server program. It receives and interpret HTTP requests from a client computer, then locate a file on the server computer (or otherwise create an HTML document), and reply with an HTML document.

Apache is also an important architectural component of many infrastructure programs. This is because many infrastructure programs now rely on web browsers as thin clients. This means infrastructure programs are designed to use any web browser as the interface.

8.3.3 MySQL

MySQL is a database server program. It is used to store, index and retrieve information. Although MySQL is useless by itself, it serves as the "backend" of many infrastructure programs.

8.3.4 PHP

PHP is a programming language that is specifically designed for Common Gateway Interface (CGI) programming. This is the most popular computer language used in infrastructure programs. Note that the interpretation engine of PHP is not exactly open source, but it is available for free.

8.3.5 Perl

Perl is older than PHP, and is also often used as the programming language of infrastructure programs. Unlike PHP, Perl is completely open sourced.

8.4 Web Oriented Programs

Many open source programs rely on the LAMP architecture, and provide services through a “web based” interface. This simply means that an end user only needs to use a browser (like Internet Explorer and Mozilla).

Note that these open source programs need to be set up and configured by system administrators, or other people how have access to server computers. Generally speaking these programs cannot run on a single personal computer.

8.4.1 WebCalendar

This is an infrastructure program that maintains calendars for individuals as well as groups. Although Personal Digital Assistants (PDAs) are common these days, it is more convenient to use a web-based calendar system, especially when it is necessary to coordinate with a group of people. WebCalendar is well suited to applications for families, as it provides a means to let “assistants” schedule events for others.

8.4.2 SQL Ledger

This infrastructure program is intended for accountants and book keepers. It permits the use of a remote server to track accounting related data using only a web browser as the frontend.

8.4.3 Moodle

You should know this one already. Modular Object-Oriented Dynamic Learning Environment (that’s Moodle!) is a infrastructure program that provides a online learning environment.

8.5 Why open source?

Open source programs are free, and that attribute raises many questions about open source software. This section answers some of those questions.

8.5.1 Profit

Compared to proprietary software, which is sold for profit, open source software is free. The obvious question is: why would any one write programs for free? No matter how fun it is to write programs, a developer still needs to feed his/her family.

How can people make money from free software?

The answer is “service”. Instead of making money from selling licenses to use a program, open source developers make money from servicing. For example, let’s say that Peter is a main developer of the GIMP (photograph editing software). If a photographer wants to add a particular feature, Peter can charge the photographer for the time that he has to invest into developing new code. The feature, once added, will be available to everyone.

For infrastructure software, such as MySQL, charged services include the installation of the database engine, security and performance auditing, database engine maintenance, training, troubleshooting and etc.

As of now (Spring of 2006), many venture capitalists are looking into funding open source projects. Note that not all open source projects are successful in terms of profit, but many are at least self sustaining.

For open source projects that are purely for fun, there is still “profit” to be made. In the open source community, developers specialize in different areas. For example, a photo editing expert may spend some free time to add features to the GIMP, which seems to be a loss of time and money. However, another developer may spend his/her free time to maintain Audacity. As a result, both developers can use the product of the other party, and hence saving expenses related to software.

8.5.2 Quality

Many people think that open source products have inferior quality compared to proprietary software. The basis of this argument is that open source projects do not have the resources of a large company to test software before releases.

First of all, there are high quality open source programs, and there are low quality open source programs. Likewise, there are high quality proprietary programs, and there are low quality ones, too. The development model does not directly lead to high or low quality.

Many open source programs have release candidates (RC) before a "stable release". An RC is, essentially, a beta release. Users who download and install RCs know that there can be potential problems. However, most RCs also have enhancements over current versions, which helps to encourage experienced users to install and use RCs.

However, even before RCs, most open source programs have "nightly makes". These are releases based on the most recent source code, and most likely contain defects. However, there are developers and users who must use the latest and greatest features, and not worry about potential defects. Afterall, it is simple to roll back the version of an open source program to a stable point.

Even after a stable version is released, most open source programs continue to release "subversions" based on feedback and defect reports. Because the source code of an open source program is accessible by everyone, an advanced programmer-user can identify and even propose a fix for a defect.

8.5.3 Progress

Does open source promote or stagnate progress?

On one hand, it seems that open source stagnate progress because there is "no money in it". In the realm of proprietary programs, progress is made to compete for end users. With open source programs, the drive to make more money is often downplayed, or simply does not exist.

However, due to the nature of available source code, if a developer feels that a product can be improved, he/she has the freedom to take the existing source code, and modify/enhance it. Many times, an open source project is cloned, and another project is spawned. This is also called "forking".

8.5.4 Security

If the source code of a program is available, doesn't that make it easier for hackers to locate vulnerabilities, and come up with viruses and other malware to attack the program?

In addition, if everyone has access to the source code, doesn't that make it trivial for a hacker to put in malicious code right into the source code of an open source product?

First of all, historically, proprietary software like Windows and Internet Explorer have been known to be more vulnerable compared to their open source counterparts. Some can argue that this is only because it is not worth the time of a hacker to attack 1 to 2 percents of all computers. Nonetheless, it proves that the unavailability of source code does not prevent a program from being compromised.

With the source code available, most open source programs can be audited either manually or automatically for security vulnerabilities. Any security expert can look into the most critical part of an open source program, and try to identify code that may lead to exploits. This is not true for proprietary software.

Although all open source programs have the source code available to everyone, *changes* to the source code are still managed. In other words, anyone can *suggest* a change to the source code, but it is up to a core team of developers to decide what proposed changes actually make its way into the released programs. In addition, because the update process is transparent, even developers who are not a part of the core can still audit changes, and identify attempts to compromise security by writing vulnerabilities into the source code.

Chapter 9

Data Representation

This chapter discusses data representation. We'll begin with binary numbers and base conversion, than proceed to discuss how numbers and other types of data are represented.

9.1 Base Conversion

We ordinarily write numbers in decimal. This means we use base-10 numbers. A digit in a base-10 number can range from 0 to 9. There is really not much that we need to explain, or is there?

Computers, on the other hand, inherently use binary numbers. This is because a transistor in a computer is either on or off. In other words, the most fundamental storage unit in a computer can only represent either 0 or 1. This is not a big deal, as it turns out. We can convert any value from one base to another.

9.1.1 Revisit Decimal Numbers

Let's just revisit decimal numbers, but in a slightly analytical way.

What is 752_{10}? The subscript 10 means the number is a base-10 number. Let's consider what this actually means:

$$\begin{aligned}
752_{10} &= 700_{10} + 50_{10} + 2_{10} && (9.1)\\
&= 7_{10} \times 100_{10} + 5_{10} \times 10_{10} + 2_{10} \times 1_{10} && (9.2)\\
&= 7_{10} \times 10^2 + 5_{10} \times 10^1 + 2_{10} \times 10^0 && (9.3)\\
& && (9.4)
\end{aligned}$$

Alright, big deal. However, this representation will help us understand how number of other bases work.

9.1.2 Base-2

Now, let's take a look at base-2 numbers. For example, let us consider 10110_2. What value does it represent?

$$\begin{aligned}
10110_2 &= 1 \times 10000_2 + 1 \times 100_2 + 1 \times 10_2 && (9.5)\\
&= 1 \times 10_2^4 + 1 \times 10_2^2 + 1 \times 10_2^1 && (9.6)\\
&= 2^4 + 2^2 + 2^1 && (9.7)\\
&= 16 + 4 + 2 && (9.8)\\
&= 22 && (9.9)
\end{aligned}$$

What just happened? It should be fairly obvious that there is a 10000_2 in 10110_2. But how come 10000_2 is 10_2^4? This is true for all bases. The number of zeros in such a number indicates the power of the base. Next, how come 10_2 is just 2_10?? For this class, we'll just accept this as a fact.

Oh, by the way, we have just illustrated how to convert from base-2 to base-10!

9.1.3 Base-10 to Base-2

The method to convert from base-10 to base-2 is similar to how we figure out changes. For example, in order to get $21.36 in cash, it'd be unwise to use 2,136 pennies. The most optimal method is as follows:

- one 20-dollar bill
- one 1-dollar bill
- one quarter
- one dime
- one penny

No, problem, right? Well, base conversion from base-10 to base-2 is actually similar to coming with the properly amount of cash. Only that in this case, we only have the following at our disposal:

- one 1-cent
- one 2-cent
- one 4-cent
- one 8-cent
- one 16-cent
- one 32-cent
- one 64-cent
- one 128-cent
- one 256-cent
- one 512-cent
- one 1024-cent
- one 2048-cent
- ...

However, the method that we use is the same:

- is there a 2048-cent in $21.36? There is one, so we use one 2048-cent coin. The remaining amount is 88 cents.
- there is no 1024-cent in 88 cents
- there is no 512-cent in 88 cents
- there is no 256-cent in 88 cents
- there is no 128-cent in 88 cents
- there is a 64-cent in 88 cents, the remainder is 24 cents
- there is no 32-cent in 24 cents
- there is a 16-cent in 24 cents, the remainder is 8 cents
- there is a 8-cent in 8 cents, the remainder is 0, but we are not done yet!
- there is no 4-cent in 0 cents

- there is no 2-cent in 0 cents
- there is no 1-cent in 0 cents

How does that convert to a binary number? Starting with the largest value coin, we do the following, from left to right:

- if a coin is used, write a 1
- if a coin is not used, write a 0

In this case, we have one 2048-cent, which means we start with a 1 on the leftmost side, then followed by a 0 for not using a 1024-cent coin and etc. The binary number is, then, 100001011000_2!

9.2 Data Representation

9.2.1 Integers

It is easy to represent integers. Well, sort of. Because we already know how to convert from base-10 to base-2, we already know how a decimal number is stored in a computer as a binary (base-2) number. The only remaining issue is how big of a number can we represent?

Given n binary digits, the largest value that can be represented is $2^n - 1$. In other words, given 12 binary digits for each number, we can represent values from 0 to $2^{12} - 1 = 4095$. Let us not worry about negative values in this course.

For most computers, the number of binary digits to represent a number is a power of 2, starting with 8. In fact, we have special names for basic units. A binary digit is called a "bit", where as 8 bits is called a "byte". Using our equation, a byte can represent numbers from 0 to 255.

The next size up is called a "word", which consists of 16 bits. A word can represent values from 0 to 65535. The next size up is called a long-word, which consists of 32 bits. A long-word cna represent values from 0 to approximately 4 billion.

Because we keep increasing the number of bit per integer, it is better to use a more technical term: n-bit integer. You can, therefore, call a byte an "8-bit integer", a word a "16-bit integer", a long-word a "32-bit integer" and so on. Modern processors can natively handle up to 64-bit integers.

Don't worry about negative values and fractional values. We are not going to talk about those in this course.

9.2.2 Text

Now that we know how to represent numbers in a computer, how about text information? It is one thing to be able to store a value of 2732, it is another thing to store names, addresses and etc.

Chapter 10

Ethics

This chapter discusses topics related to the ethics of using computers. There are many topics that affect hackers, developers, publishers, patent holders and end users.

10.1 The Frontier

A long time ago, computers were merely tools used by the military and large organizations to process data. As such, few ethical issues needed to be considered. However, computers are no longer limited to data processing. In fact, computers touch our daily life more frequently than people realize. This quickly expanding frontier touches many new ethical issues.

10.2 The Business

Computer hardware and software are *big* businesses, lots of money can be made in either area. However, even more importantly, the *content* that can be distributed and rendered by computers is even more lucrative.

As long as there is money to be made, there is always the possibility that greed will lead to ethical issues. This is true about agriculture, energy (oil), as well as computer related fields.

At this point (2006), the most debated topic is "who owns a computer, who owns the content?" Corporations are deriving strategies to increase their control over personal computers and even how contents are rendered on personal computers. It becomes easier for content authors to control the use of contents.

10.3 Privacy and Security

Privacy and security have always been challenged, not only by criminals, but also by authoritative and totalitarian governments. Before computers are widely used, intrusion to privacy had to be performed manually, which put an limit to how much intrusion can be done. The same applies to security breaches.

With computers, however, both privacy and security can be challenged by computer running special programs. This means the limits based on available manpower no longer exists. With computers, it is even easier for hackers and oppressive governments to achieve their objectives.

10.4 Hacking

As publishers charge for licenses to install software, it generates interests to "hack" copyrighted software so that people can install and use software without having to pay (the full amount). Ever since the infancy of personal computers (remember the Apple 2 computers?), hackers have engaged publishers in a tug war to see who is smarter.

Note that copyright laws also apply to books, and photocopies can be made from books. However, the difference between books and software is the methods of replication and distribution. With books, the replication and distribution methods are physical (well, until recently). This means the extent of copyright infringement is limited by physical capabilities of distribution.

With software and networking, however, there is little limitation. Replicating a program is a trivial operation, and it can be easily be combined with the distribution method. Even more importantly, some distribution methods are stealthy and it is difficult for content copyright holders to identify infringement.

10.5 Open Source

Although the open source movement started with software source code, it expands to include the distribution of various contents. The open source movement presents challenges to corporations who subscribe to the "proprietary" philosophy.

The open source movement started as a grassroot movement, involving mostly devoted developers. However, this movement has now expanded to impact major businesses. For example, IBM, HP and many other companies have already embraced open source (Linux) in their businesses. Venture capitalists have also started to look into investment in open source project teams.

10.6 Software Patents

10.6.1 Background

The original intent of a patent is, believe it or not, to encourage innovation. Without patents, innovators are reluctant to share their ideas because others may "steal" the ideas and leave the original inventor pennyless. Furthermore, an innovator may try every possible way to preserve the secrets of his/her invention.

With the patent system, however, innovators can disclose their inventions and still protect their (financial) interests. Once an invention is granted a patent, the inventor has exclusive rights to utilize the invention for a fixed period of time.

Anyone who attempts to capitalize a patented invention without getting a license from the inventor can be sued for infringement. This system curbs competition for a fixed period of time to permit an inventor to capitalize from the invention. However, at the same time, it also permits the invention to be disclosed so that others can study the invention, and possibly improve upon it. When a patent expires, the invention becomes public domain.

Up until very recently, patents are only granted to physical inventions. Mathematical ideas, for example, are not patentable. This is not an oversight. Rather, it was an intentional decision. Mathematical theorems are not considered inventions, but rather discovered truths.

10.6.2 Software Patents

Where does this leave us with software?

Should concepts and ideas used in software be patentable?

It all depends on whether a program is considered sythetic or discovered. In other words, for a given problem that has a solution, does the solution exist already all along, or is it invented?

Computer scientists, such as Dr. Knuth, consider software methods a branch of mathematics. In other words, a software method (also called an algorithm) that solves a particular problem is no different from a theorem in mathematics.

Chapter 11

The Future

This last chapter discusses the future of computer and its related technologies. While some of the materials in this chapter may seem far fetched, we have to remember what we use now on a daily basis was considered far fetched!

11.1 The Near Future

What holds in the near future for us? Material in this section will impact us in the next 5 to 10 years. Consequently, this section may be useful when you need to determine what you need to learn in order to stay competitive at work.

11.1.1 Wearable Computers

Transistors in computers are shrinking, and now to the point that a PDA (personal digital assistant) sized machine can have plenty of processing power. Combined with improved battery technology and display technology, many computers are now wearable. For example, PDAs and cellular phones are basically wearable computers.

Because storage density is also improving very rapidly, it will become possible to bring your own personal computer with you, to school, to work, to LAN parties. There are devices like this already, although the processing power of such computers are still relatively weak. The impact of such portable devices is that we can bring *everything* with us, not just the data on a flash memory stick.

11.1.2 Wifi Network

There are a few emerging competing wireless networking technologies. All of these emerging technologies claim less expensive hardware and much improved capacities. When these technologies mature in the near future, they will help build inexpensive but high speed metropolitan wireless networks.

Wireless high speed metra wireless networks will change the way we connect to the Internet. Combined with wearable computers, the Internet will be accessible all the time and everywhere. If you think driving and talking on a cell phone is distracting, try replying email and driving at the same time!

In order to implement effective wireless networks, the industry will require electrical engineers who specialize in analog and microwave circuits. This may quickly drain the available pool of university graduates. Currently, most graduates specialize in digital circuit design, and most people believe that “analog is dead”.

11.1.3 Thin Clients

Most of us know that software, especially the operating system, requires constant updates to improve security and privacy. The level of expertise required to maintain a computer is quickly escalating to a level that is beyond most end users. Combined with the installation and update of office productivity software, games and etc., the amount of administrative work involved in computing is just too much for most.

This is why the concept of “thin clients” is coming back. With sufficient networking resources and wearable computers, this concept is more attractive than ever. A thin client is, essentially, a computer with minimal resources just to handle the input (keyboard, mouse) and output (graphics and sound). However, all the actual processing is performed on a remote computer (a server) that has lots of processing power and storage capacity.

Using the thin client concept, each individual (wearable) computer only needs to have very simple software installed. Hopefully, this greatly reduces the need to patch and update the software. Most of the maintenance responsibility is pushed to the administrator of the server. However, this is relatively easy to do because all thin clients share the same operating system and application programs.

The only concern of thin client computing for the masses is privacy. Because storage and processing are both centralized in a managed system, it is difficult to ensure end user privacy and security. A possible solution is to use a proxy server that end users may purchase. The proxy server serves as a middleman. On one hand, it communicates with an ISP to updates its application programs and operating system. On the other hand, it communicates with thin clients (as wearable computers).

This way, both storage and processing are handled by a computer that end users own, but still retain the ability to minimize thin client hardware and maximize ease of system maintenance.

11.1.4 Security

Computer security will become increasingly important because of improved accessibility of the Internet. There will be more opportunities for criminals (black hats) to exploit others. This automatically generates more demand for computer security experts (white hats) to protect end users from black hats.

The use of thin clients will greatly improve computer security for end users, but only when it is done correctly. Otherwise, with every thin client running the same faulty software, it actually makes it easier for black hats to exploit computers!

The current method of virus and malware scanning will become nearly impossible as the number of malware programs continue to increase. I can see two possibilities. One, computers can utilize a specialized processor just for virus scanning. This frees out the main processor for regular computation work. Two, *all* communication will require some degree of authentication. This way, if a source does not have the proper authentication, a connection (and its content) will simply be ignored.

11.1.5 Cluster Computing

Clustering is the grouping of individual computers to solve a difficult problem collectively. This really is not in the future. Distributions of Linux and Windows (server) can already perform clustered computing.

However, with the availability of faster networking, cluster computing becomes possible and practical on a much wider scale. Currently, cluster computing is only practical in a local area network. However, with improved Internet connection speed, it will soon be practical to form global clusters.

The importance of clustering is that many problems that cannot be practically solved with one computer can now be solved collectively by many computers. Particle simulation and ray tracing (for computer generated graphics) are two common applications that benefit from cluster computing.

There are other applications of cluster computing that are, well, less legitimate. For example, the cracking of encryption is an application that can also benefit from cluster computing.

11.1.6 Media Distribution

I think we'll see a dramatic change of how media (music, video and etc.) are distributed in the near future. The current CD/DVD methods will be replaced by online download. To fight piracy, media content providers (members of the RIAA and MPAA) will most likely resort to encryption and "trusted computing".

We can also expect to see new bill and legislatures designed specifically to limit what end users can do, and give media content providers more control over computers. For example, someone has already proposed that the RIAA and MPAA should regulate what kind of consumer video and audio devices will be legal based on conformance to their rules.

The basic idea behind this is simple: don't let consumer purchase anything that can potentially be used to pirate music and movies. We'll most likely start to see digital headphones and speakers that decode a stream of music right at the earpiece or speaker. This way, music can stay encrypted all the way until sound is actually produced.

11.2 Within Our Lifespan

This section discusses advances that will most likely happen in our lifespan (in the next 10 to 20 years).

11.2.1 Quantum and Photonic Computers

Our existing computer processor architecture is, inherently, restrictive. This is because all transistors of an integrated circuit exist on the same plane (surface), and they need to be connected to each other using a small number of planes. This technology also limits that our processor be "sequential". This means our processors can only do one thing at a time. Although recent developments have been able to make many processing occur in parallel inside a processor, the amount of parallelism is still extremely limited.

With quantum computers, however, "processing" can be performed in massive parallelism. Some experts believe that many "intractable" problems that our sequential (von Neumann) machines cannot solve will be handled by quantum computers.

A photonic computer is a computer that operates on light. Unlike electronics in a circuit, photons do not have dimensional limitations. Furthermore, instead of just using on and off states, photos also have amplitude and frequency properties that can be used to encode and store data. Combining all of this with the ability to build three dimensional circuits, photon computers have the potential to out perform electronic computers by huge magnitudes in terms of processing power and data storage density.

11.2.2 AI

Artificial intelligence was a *hot* field in computer science in the 80's when the U.S. was competing with Japan to see who can come up with the first intelligent system. However, much of the AI related research has died off since early 90's. The main reason was that there simply was not enough processing power for artificial intelligence programs.

With dramatically faster computers (such as quantum and photon computers), many artificial intelligence related topics may get a second life.

11.2.3 Direct Nervous System Interface

Engineers and scientists have already successfully interfaced silicon devices with neurons. However, there is much research to be done if we are to project images directly into the brain, or have the brain to interface with a database directly to access information.

Much of the research in this area advances as nanotechnology matures. This is because nanotech devices can be made *much* smaller than that of a neuron, and require very little signal to work.

Once artificial devices and natural neurons can interact with ease, cybernetics will become commonplace. The big question is, then, at which point is a person no longer a person, but rather a mechanical being?

11.3 And Beyond

What will happen 20 years later? It is increasingly difficult to predict the future because science and technology advance at an increasing speed. However, it is still kind of fun to imagine how things will be in the future!

11.3.1 Thought Sharing

With neurons interfacing with artificial devices, and fast networks, it'll be possible for people to share their thoughts. We already do that to a certain extent. Every time you call someone on the phone, or write someone an email, it is already sharing thoughts.

However, sharing thoughts at the neuron level is fundamentally different from sharing thoughts at a language (spoken or written) level. First of all, some translation still needs to occur. No two individual store concepts and ideas in the same way. This translation, unlike lingual translation, does not involve grammar as we know it. It'll involve a basic "mapping" of the nervous system by subjecting it to various stimuli, and recording how the system responses to each stimulus.

Assuming idea and concept representation can be mapped, thought communication will be, by today's standard, an interesting experience. I am *guessing* that it'll feel like split personality, with one personality conversing with the other personality all in the mind. Except, of course, that the "other" personality is *really* another person.

11.3.2 Collective Consciousness

With thought sharing, it is not difficult to imagine a collective consciousness. Although each person may still have an individuality, the network of individual consciousnesses may give rise to a new level of consciousness and awareness.

Collective consciousness is, obviously, just fictional at this point. However, many science fictions have already incorporated this concept into stories. Most notably, Star Trek (the Next Generation) has the Borg. Many Japanese anime also explore this idea, such as “Ghost in the Shell”.

11.4 Earth to Tak...

What is the bottomline for now?

For an “average” end user, I think the following topics will have significant impact:

- Open source. Open source software has matured to the point to be utilized by an average end user. An open source program is free, and it can usually replace a proprietary program that costs much more.
- Alternative operating systems. Linux has always been considered a backroom operating system. However, it is now *almost* ready for adventurous end users. Distributions like Knoppix make it easier for end users to experiment with Linux without a lot of commitment.
- Computer security. Keep an eye on computer security news. Attempts to exploit and hack into your computer will only increase in the near future.
- Privacy. Learn about encryption and digital signatures. Start to question the validity of email senders as well as contents.
- Web-based applications. With recent developments like AJAX, it is possible for a web application to appear seamless and efficient. Both Google and Microsoft are investing resources in their own web-based office productivity services.

For those of you who intend to get into a computer related field, I think the following topics will become increasingly important.

- Open source. Many people have already mentioned that we may get another tech. bubble fueled by open source software. If you want to become a developer, learn about the languages and tools commonly used by open source developers.
- Security. More black hats will attempt to compromise computers due to the proliferation of computers. More importantly, more businesses are depending on computers and networks. As a result, the demand of white hats can only increase in the near future. Due to the sensitivity, such jobs are unlikely to be off-shored.
- Computer ”handyman”. As more end users own computers, the demand of computer maintenance and repair increases. Note that these jobs require technical skills, experience and people skills. Furthermore, such jobs cannot be off-shored.

Index