Lecture 3: Interacting with a computer
Before we can program a computer...

• Before we can learn to instruct a computer, we must first learn to:
  - interact with a computer

• Devices used to interact with a computer:
  • Mouse
  • Touch pad
  • Keyboard
Before we can program a computer...
(cont.)

• For *casual* computer users, the *most important* interaction device is:
  - mouse (and a little keyboarding for chatting)...

• For *computer programmers*, the *most important* interaction device is:
  - keyboard !!!

because they must write programs in an *English-like programming language*

(Try writing a term paper using a mouse by clicking on a virtual keyboard…. Not fun!)
Suggestion…

If you currently hunt and peck,

    Start chatting using all 10 fingers on the keyboard NOW! (And keep at it until you don’t have to look at the keyboard!!)
How a human interacts with a computer

• Preliminary step:
  • We must **gain authorization** to use the computer system
  • This is usually achieved through an **authentication process** (popularly known as **log in process**)

• After **log in** is successful, we interact with a computer by:
  • Running various **computer applications**
How a human interacts with a computer (cont.)

- The **application** that you need to run depends on the **task** that you want to perform

Examples:

<table>
<thead>
<tr>
<th>Task</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Surf the Internet</td>
<td>Web browser (such as FireFox, IE, Chrome, etc)</td>
</tr>
<tr>
<td>Edit a report</td>
<td>Text editor (such as Word, gedit, vi, emacs, etc)</td>
</tr>
<tr>
<td>Email</td>
<td>Email client (such as Thunderbird, or Outlook)</td>
</tr>
<tr>
<td>Ledger</td>
<td>spreadsheet (such as Microsoft Excel)</td>
</tr>
<tr>
<td>Database application</td>
<td>Database server (such as SQL server)</td>
</tr>
</tbody>
</table>
Gaining access to a PC

- Depending on the security setting, you may need to type a user ID and the corresponding password to gain access to a PC.
- After this authentication process, you can run applications (usually through some mouse clicks).
Gaining access to the MathCS computer system: log in

• The MathCS computer lab is located on the 3rd floor of the MSC building

  The lab hours and rules can be found at this webpage:
  http://www.mathcs.emory.edu/computinglab.php

• You can log in (= gain access) to any computer in the MathCS lab using:

  • your Emory ID
  • your Emory password

• We will go over this in more detail in lab next week
Computer Operating Systems

• Every computer is controlled (= "managed") by a very complex computer program called an Operating System (OS)
• The Operating System (OS) controls:
  • The mouse
  • The keyboard
  • The monitor
  • The disks
  • ...
  • Every component of the computer !!!
Computer Operating Systems (cont.)

• The Operating System (OS) can also:

- Detect mouse clicks
- Detect key strokes
- Load a computer program from hard disk into RAM memory and execute it
Making computer do things for you

- Recall that:
  - A computer is a (dumb) machine that only executes instructions
  - If you want a computer to do a task for you, then you must:
    - Give a computer the complete list of instructions that accomplishes the task
Making computer do things for you (cont.)

• making a computer do a task can be a painful job...
• Fortunately:

  • People have written down the complete list of instructions to do frequently performed tasks
  • These lists of instructions have been saved in computer files (naturally...)
  • These files are better known as computer applications Some well known computer applications:

    • Web browser (e.g., Internet Explorer)
    • Editor (e.g., Microsoft Word)
    • and so on...
Making computer do things for you (cont.)

• In summary:

  • If you want a computer to do a common task, you must run (= "execute") the appropriate application

We will first discuss how to run (execute) an application on a computer
Executing a program through mouse clicks

- For PC users, the most common way to run a computer program is through mouse clicks.

Example:

- When you want to run the FireFox web browser, you double click on the FireFox icon:
Executing a program through mouse clicks (cont.)

- This is what is going on inside a computer when you (double) click on an icon that is associated with a computer program:

1. You **double click** on an **icon**
2. The **Operating System** detects the **mouse clicks** on the **icon**
3. The **Operating System** locates the **program** that is associated with the **icon**, loads the program into **RAM** and execute it
Executing a program through mouse clicks (cont.)

- In Microsoft Windows, you can find out which program is associated with an icon by right-click the icon and select properties.
  Example:

  ![Mozilla Firefox Properties](image)

  - The **Target field** in the "properties" window shows the path of the program that will be **executed** when you double click the icon.
Executing a program through a keyboard

- Every computer system provides an application that allows the user to type in the path of the program that he/she wants to execute
- Application name:
  - **Microsoft Windows**: `cmd` (command line tool)
  - **UNIX**: `sh` (the original Shell), `csh` (C Shell), `ksh` (Korn Shell), `bash` (Born Again Shell),
  - **MAC OS X**: `Terminal` (it is in fact a UNIX shell program)
Executing a program through a keyboard (cont.)

- Example: running FireFox using cmd in MicroSoft Windows:

  - On your PC: click START and run cmd
  - In the cmd window, type the complete path of the FireFox application:

```
C:\Users\Cheung>C:\Program Files (x86)\Mozilla Firefox\firefox.exe
'C:\Program' is not recognized as an internal or external command, operable program or batch file.
C:\Users\Cheung>"C:\Program Files (x86)\Mozilla Firefox\firefox.exe"
C:\Users\Cheung>
```

Note: make sure you quote the entire command !!!
Storing your "stuff" away in a computer

• Today, virtually everyone has used a computer to write reports in high school
• Documents created by computer users are stored in a computer system as a:
  • compiler file
• A computer file is contained on the hard drive of the computer system
Organizing your files

• When you have a *small number* (like 10 or so) of files, you can put the files in the *same location* (e.g., the *desk top*) and you can find them easily when you need them.

• However, if you have a *large number* (like 1000 or more) of files, you will have a *hard time* locating a file if they are all put in one location.
Organizing your files (cont.)

- Directories:
  
  - A directory is a special file (yep, a directory is a file) that can contain:
    
    - Information of the locations of files
    - Information of the locations of other directories
Organizing your files (cont.)

• Organizing files and directories:

  • File and directories are organized as a file system
  • Every modern file system is organized as a logical tree structure
The (logical) tree structure

• The shape of a tree looks something like this:

• A tree structure in computer science looks like a "upside-down" tree:
The (logical) tree structure (cont.)

- What a tree structure in Computer Science look like:
The (logical) tree structure (cont.)

• Terminology:

  • The circles in the tree structure are called nodes
  • The top most node in the tree structure is called the root node
  • The lines in the tree structure are called branches or links
More terminology: up, down, parent and child

• Directions in the tree:

  • Up (or upward direction) = when you move from one node to another (connected) node that leads you closer to the root node

  • Down (or downward direction) = when you move from one node to another (connected) node that leads you further away from the root node
More terminology: up, down, parent and child (cont.)

• Example:

• Parent

  • Parent node = the first node that is located in the upward direction

  (There is only one parent node for any node in a tree - look carefully in the picture above.)
More terminology: up, down, parent and child (cont.)

- Child node:
  - Child node = any node that is located immediately in the downward direction

(There are *many* child nodes for a node in a tree - look carefully in the picture below.)

- Example:
The tree structure of the Microsoft Windows file system

• If you have worked with a PC, you should be familiar with the Microsoft Windows file system

• The tree structure of the Microsoft Windows file system is based on the disk drives that are available in "your PC"
The tree structure of the Microsoft Windows file system (cont.)

- The root node of the Microsoft Windows file system is the node "My Computer"

The first level of branches are the individual disk drives:

The Local Disk (C:) is a node under the root node (My Computer) that is commonly used to store System and users' files.
The tree structure of the Microsoft Windows file system (cont.)

- The following figure shows the nodes under the node Local Disk (C:):
The tree structure of the Microsoft Windows file system (cont.)

- The following figure shows 2 more levels of the file system tree structure:
The tree structure of the UNIX file system

- You will be using the UNIX Operating System in college

Reason: UNIX is more stable (crashes less often) (from answer.com - see: http://wiki.answers.com/Q/What_is_the_difference_between_Windows_and_Unix)

Windows and Unix

As far as operating systems go, to some it would seem as if UNIX has a clear advantage over Windows. UNIX offers greater flexibility than Windows operating systems; furthermore, it is more stable and it does not crash as much as much as Windows. To some, UNIX is just as easy to use as Windows, offering a GUI interface as well as command line. But there are users out there that believe UNIX is for only for computer gurus only, claiming that the fragmentation of the UNIX GUI is its greatest competitive weakness.

One thing that has been established though, UNIX is quite a bit more reliable than Windows, and less administration and maintenance is needed in maintaining a UNIX system. This is a huge cost saver for any organization.
The tree structure of the UNIX file system (cont.)

- The tree structure of the *UNIX* file system:

- The tree structure of the UNIX file system is based on functionality/purpose:
  - Files used for a specific function/purpose is stored in the same directory
The tree structure of the UNIX file system (cont.)

- The root node of the UNIX file system is the node "/"
- The second level (immediately under '/') usually contains the directories:

<table>
<thead>
<tr>
<th>Directory name</th>
<th>Function of files/directories contained in directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>home</td>
<td>home directories of users</td>
</tr>
<tr>
<td>bin</td>
<td>binary files: files that contain machine instructions</td>
</tr>
<tr>
<td>dev</td>
<td>devices: contains files that link to disk drives, keyboard and other devices</td>
</tr>
<tr>
<td>lib</td>
<td>library files: support files for various high level programming languages</td>
</tr>
<tr>
<td>tmp</td>
<td>temporarily files: scratch area for some running programs</td>
</tr>
<tr>
<td>etc</td>
<td>miscellaneous files and directories</td>
</tr>
</tbody>
</table>
Home directory of UNIX users

• Unlike a PC ("Personal Computer") which is mainly used by 1 person, the UNIX operating system is usually used to:
  
  • manage a large computer system that is used by many different users (at the same time)
Home directory of UNIX users (cont.)

- **Unlike** a PC ("Personal Computer") where you can store your files *more or less anywhere* in the file system, you are assigned a *specific location* in the UNIX file system:

  - Every user is assigned a **home directory**
  - You can create (make) new **directories** and **files** inside your **home directory**
  - You **cannot** create anything in **someone else's home directory**

    Nor can you create files/directories in the **system's directory** (such as: **bin**, **dev**, and so on)
The home directories of all users in the UNIX system are contained in the home directory under the root directory:
Home directory of UNIX users (cont.)

• Example:

- The directory km under home is the home directory of Prof. Mandleberg
- The directory jlu under home is the home directory of Prof. Lu
- And so on...

• Each user can create more directories to organize his/her files inside his/her home directory

In the figure, you see some of the directories that Prof. Cheung has created inside his home directory
Interacting with a UNIX computer
Concept: the home directory

- Each user in UNIX is assigned a unique user ID
- Each user in UNIX is assigned a home directory (folder) where he/she can store his/her files.
- The (absolute) directory path of the user with the user ID "X" is:
  - /home/X

Example: The home directory of Prof. Swenson whose user ID is mswens2 is:
  /home/mswens2
Tasks that are performed very frequently

• Frequently performed tasks: (you will practice these in Lab 1)

  • Operations on directory (folder):
    • Print the path of the current (working) directory
    • List the content of a directory
    • Navigate the directory tree
    • Create a directory
    • Delete a directory
    • Rename a directory
    • Move a directory to somewhere else
Tasks that are performed very frequently (cont.)

- Operations on files
  - Print a file out to the terminal
  - Delete a file
  - Rename a file
  - Move a file from one directory (folder) to another directory (folder)

Creating a file can be done through an editor application which will be explained in a later lab.
Tasks that are performed very frequently (cont.)

- Remember:
  - You will be executing a *computer program* to perform each of the tasks above.
  - Some times, you will be executing the *same program* to do different tasks.

(Because some programs are like a *Swiss army knife* that can do many different tasks)

- We will *also* need to learn some *concepts* (and terminology) associated with these *tasks*.
Task: List the content of a directory

• The command (= application) that is used to list the content of a directory:

  `ls DIRECTORY-PATH`

  (`ls` is an abbreviation of the word `list`)

The `ls` command (application) will list the names of the files and the directories stored in the specified directory in alphabetical order.

Note:

• The `ls` command will give you the same information as a file browser (such as "Window Explorer" on a PC)
Task: List the content of a directory (cont.)

• Example 1: "ls /" --- give the list of directory and files stored in the root directory /

Output of the ls command

Browsing the directory with a File Explorer
Task: List the content of a directory (cont.)

- Example 2: "ls /etc" --- give the list of directory and files stored in the directory /etc

Output of the ls command

Browsing the directory with a File Explorer
Listing vs. Browsing a directory:

- **Browsing is slow**
  (You need to click a lot before you can reach the desired directory that you want to browse)
  I can type in a directory path a lot a faster than I can click...

- **Browsing produce a nicer output**
  (If you become a Computer Scientist, you will learn that only information matters, not a nicer look)

- Bottom line: geeks prefer the speedier directory listing method
Convenience of the *current* directory

- When working with some *file(s)*, it would be nice if we could access the file with the *least number of key strokes* possible
- Fact in UNIX:
  - *All applications* in UNIX can access a file by its *name* when the file is in the *current directory*
  (That's because the *filename itself* is the *relative path* to the file - which will be explained later)
Convenience of the *current* directory (cont.)

- In order to take *take advantage* of this convenience, we must *change the current directory* to the directory that contains the desired file(s)
Task: Changing the current (working) directory / Navigate the directory tree

• The command (= application) that is used to change the current directory to another directory is:

   `cd NEW-Current-DIRECTORY-PATH`

`cd` is an abbreviation of the words change directory

The New-Current-DIRECTORY-PATH can be an absolute directory path or a relative directory path
Concept: Special directories

- There are a number of special directories in UNIX that have special notations
- Special directories:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Directory represented by the symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>The root directory</td>
</tr>
<tr>
<td>~</td>
<td>The home directory of the user</td>
</tr>
<tr>
<td>.</td>
<td>The current (working) directory</td>
</tr>
<tr>
<td>..</td>
<td>The parent directory of the current directory</td>
</tr>
</tbody>
</table>

The parent directory is the directory that is immediately above the current directory in the directory tree.
The parent directory of the root directory / is itself (\/)
Concept: Special directories (cont.)

- Examples
  - Suppose the home directory is /home/cheung and
  - current (working) directory is /home/cheung/cs170
  then

<table>
<thead>
<tr>
<th>Command</th>
<th>Directory represented by the symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>ls /</td>
<td>root directory</td>
</tr>
<tr>
<td>ls ~</td>
<td>/home/cheung (= home directory of the user)</td>
</tr>
<tr>
<td>ls .</td>
<td>/home/cheung/cs170 (= current (working) directory)</td>
</tr>
<tr>
<td>ls ..</td>
<td>/home/cheung (= the parent directory of the /home/cheung/cs170)</td>
</tr>
<tr>
<td>ls ../..</td>
<td>/home</td>
</tr>
<tr>
<td>ls ../..</td>
<td>/</td>
</tr>
<tr>
<td>ls ../../../..</td>
<td>/ (because the parent directory of the root directory / is itself)</td>
</tr>
</tbody>
</table>
Creating and running java programs

• See diagram in section 1.8
• Demo using Hello.java program from Friday