Comp-206 : Introduction to Software Systems
Lecture 3

Alexandre Denault
Computer Science
McGill University
Fall 2006

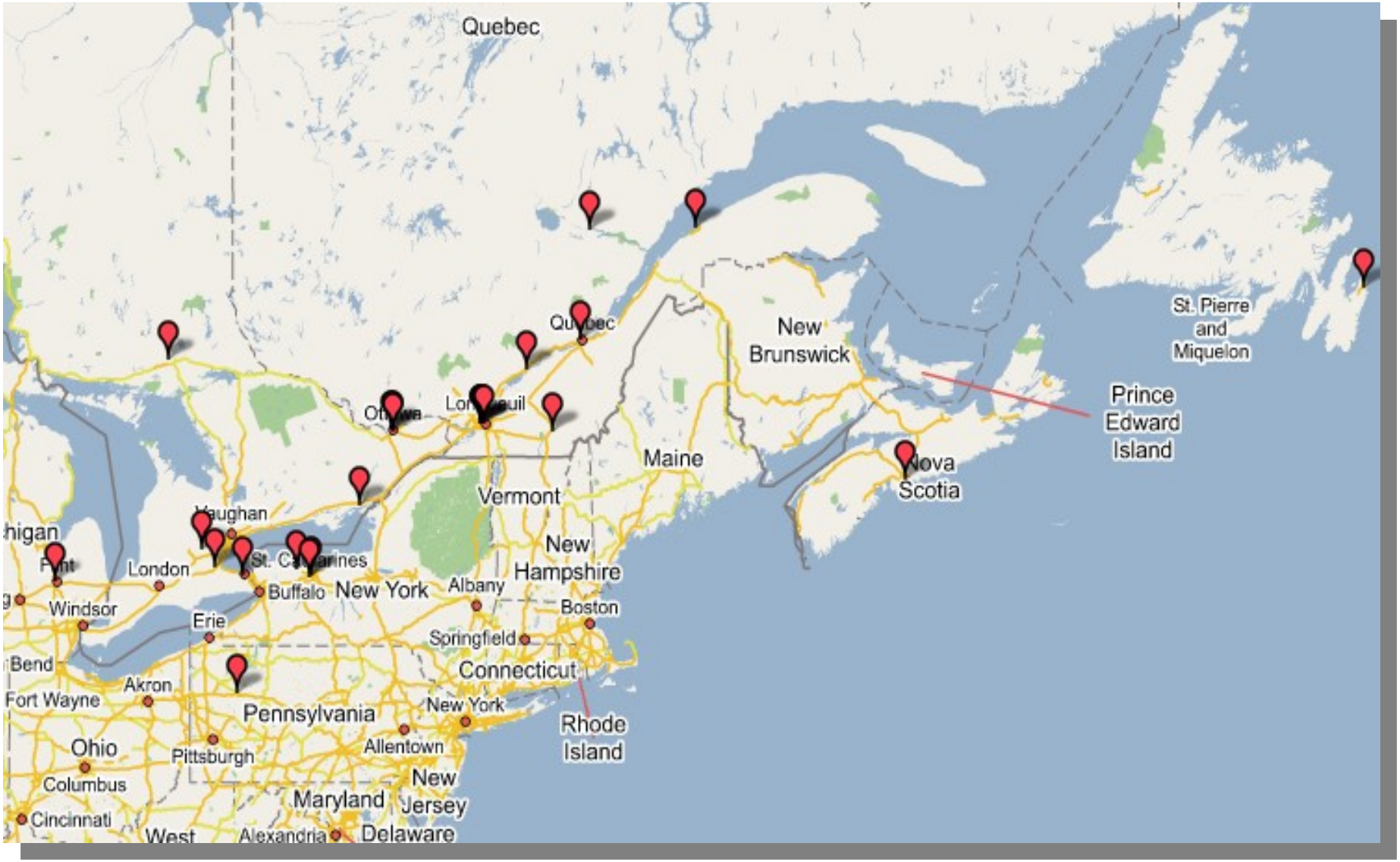
Robert Kaplow

Wednesday, 12h00 to 14h00
T.A. room, Trottier, 3rd floor

CSGames

The Computer Science Games (CS Games) is a competition for undergraduate students in computer science. For three days, participants from everywhere in North America will be challenged individually and by team in many different computer science fields. The competition covers many domains, from programming to video games without forgetting algorithm and logic.

Is CS Games a local event?



CSGames needs you!

For more information on CSGames, check out www.csgames.org.

For people interested in helping organize the event, contact Miriam Zia at mzia2@cs.mcgill.ca

Do permissions overlap?

- Given the permission "`------rwx`" of a file I own, can I read the file?
  - You will not be able to read the file.
  - People in the group owner will not be able to read the file.
  - Other people will be able to read the file.

Quiz: Can I read, write, execute?

Can user “Bob” of group “Student” read, write or execute the following files?

- `rwxr--r--  Cathy  Frosh    file1.sh`
- `r-x------  John   Student  file2.txt`
- `rwxrwxr--  Bell   Student  file3.txt`
- `rwxrwxrwx  George Teacher  file4.c`
- `rwx------  Bob    Student  file5.s`
- `rw-rw-r-x  Norm   Admin    file6.doc`
- `rwxrwx---  all    all      file7`
- `------rwx  Bob    Student  file8.doc`
- `---rwx---  Bob    Student  file9.txt`

Bonus: Which write does root have on these files?

How to change permissions

To use the chmod command, you need to know the following abbreviations:

Who

- u : The user who owns the file (this means “you.”)
- g : The group the file belongs to.
- o : The other users
- a : all of the above (an abbreviation for ugo)

Permission

- r : Permission to read the file.
- w : Permission to write (or delete) the file.
- x : Permission to execute the file, or, in the case of a directory, search it.

chmod

- The syntax of the command is as follows:

```
chmod who=permission files
```

- Here are a few examples of the chmod command:

Give read permission to group

```
chmod g=r file.txt
```

Give read/write/execute permission to you (user)

```
chmod u=rwx file2.txt
```

Remove all permissions from others

```
chmod o= file3.txt
```

Give read/write permission to user and group

```
chmod ug=rw file4.txt
```

Directories

- Directories use the same permission system as files.
- The root directory is denoted by a slash (/).
- As already mentioned, the home directory of a user is denoted by the tild (~).
- A common UNIX system has the following base directory structure.
  - /etc : Contains configuration files and passwords
  - /bin : Contains executable related to the OS
  - /usr : Installation directory for application
  - /opt : Other installation directory for application
  - /dev : Contains the device files for all the hardware connected to the computer.
  - /var : Contains system files that change a lot (logs, printer spool, mail spool, etc).

Relative Paths vs Absolute Paths

- Absolute : An absolute path contains all the directory structure starting from the root to the file.
  - For example, the path to the file containing user information is /etc/passwd
- Relative : A relative path to a file changes depending on the current directory.
  - If my current directory is /etc, then the relative path is ./passwd
  - If my current directory is the root (/), then the relative path is etc/passwd or ./etc/passwd
  - If my current directory is /usr, then the relative path is ../etc/passwd
- When we take a closer look at File I/O, we will compare the advantages and disadvantages of relative vs absolute paths.

Relative or Absolute

Find the corresponding absolute path:

- “adenau/file1.txt” if my current directory is “/home”
- “file2.sh” if my current directory is “/usr/lib”

Find the corresponding relative path if my current path is “/home/adenau”:

- “/home/adenau/test/file3.c”
- “/etc/config/file4.sh”

Shell

- A shell is an application that allows you to relay commands to the Operating System.
- These commands are executed by the Operating System and the output is sent back to the shell.
- Most often, shells are used to execute commands.
- Most shells also include a simple programming language, often known as a scripting language.
  - The first popular shell was the Bourne shell (sh). It remains at the core of most UNIX OS.

Script (Batch)

- Scripts are collections of commands, grouped in a file and sequentially execute.
- Scripts are not compiled programs, they are interpreted.
- Scripts run from top to bottom, with little control flow (does not have the complexity of a piece of software).
- System administrators use script to automate complex tasks.

Command line and parameters

- The command line is the prompt that allows you to enter text commands.
- You can enter simple commands.

```
ls
```

- You can allow use switches (additional parameters) to alter the behavior of a command.

```
ls -l
```

- The number of parameters you can use depends of the command itself.

```
ls -l -a -F
ls -laF
```

Current Directory

- At any moment, a shell always has a current directory.
- This current directory is used to resolve relative paths.
- After login in, the current directory is usually the user's home directory.
- The `pwd` (print working directory) displays the current directory.
- The `cd` (change directory) allows a user to change their current directory.
  - The current directory is local to the instance of the shell.
  - Changing the current directory in one shell will not affect other shells.

Directory Content

- The `ls` command is used to display the content of a directory.
- To get more information, use the `-l` argument (`ls -l`).
- By default, `ls` does not display hidden files. To include hidden files in the listing, use the `-a` argument (`ls -a`).
  - File names and directory names that start with a period ¨.¨ are considered hidden under Unix.
- The `ls` command displays the complete list of files, without stopping. To display a page by page list, you need another command.

Man Pages

- The `man` command allows you to access the on-line manual pages of the various command available on the shell.
  - These pages are often referred to as Man pages.
- The man pages are your first source of information when working in the shell.
- To access a man page, simply type man and the name of the command.

```
man ls
```

Redirection

- Previously, we mentionned that normal output goes to the STDOUT channel.
  - When the ls command is executed, its output is sent to STDOUT.
  - STDOUT is then displayed by the shell.
- • Output sent to STDOUT can be redirected.
  - By using the > sign, STDOUT can be redirected to a file.
    Ex: `ls -la > list.txt`
  - By using the >> sign, STDOUT can be appented to a file.
    Ex: `ls -la >> list.txt`
  - By using the | sign, STDOUT can be redirected to another application.
    Ex: `ls -la | more`

cat, tail, more or less

- cat [options]
  - file concatenate (list) a file
- echo [text string]
  - echo the text string to stdout
- head [-number] file
  - display the first 10 (or number of) lines of a file
- more [options] file
  - page through a text file
- less [options] file
  - browse (navigate) through a text file
- tail [options] file
  - display the last few lines (or parts) of a file

Redirection for ls

- The previous commands can easily be combined with the ls command.
  - `ls -la | more` will present a paginate list of files.
  - `ls -la | head` will present only the first 10 files.
  - `ls -la | tail` will present only the last 10 files.
  - `cat `ls *.log | tail -n5` >> text.out` concatenate the last 5 log files in the current directory and write them to the text.out file.