Lab 01: UNIX and the Command-line

CMPT 145

CMPT 145 Laboratory Overview

- Weekly lab work, providing tools and techniques not covered in Lecture.
- Each lab has a pre-lab reading component, which you should read in advance.
- Each lab has activities scattered in the lab, marked with ACTIVITY.
- Each lab has a required hand-in, which includes the work you did for one or more of the ACTIVITY items.
- Each lab is due at the end of the week it is assigned.
- Each lab is intended to be completable in 75 minutes (i.e., one lab period), not including the pre-reading.

Laboratory 01 Overview

Part 1 : Pre-Lab Reading
Part 2 : Logging In and Becoming Familiar with the Lab
Part 3 : An Introduction to the UNIX Command-Line
Hand In : A transcript of your work with the command-line

Part I

Pre-Lab Reading

Linux Labs

Computers in S320 dual-boot into Windows and Linux. We will be using Linux. If a computer is currently running Windows, restart it, and during the boot select Linux.

- LINUX is a system built by programmers for programmers.
- Learning to use linux will make you more productive.

Windows is not based on linux/UNIX (but there are ways to get UNIX-like tools, e.g., Cygwin, WSL, MinGW).

Command-Line: Background

- Modern computer systems use graphical user interfaces (GUIs) with drop-down menus and mice, or touch-screens.
- Prior to the use of GUIs, users did everything using an application called a command-line.
- The command-line is a simple app that repeats the following steps:
  - (a) The computer shows that it is ready for a command.
  - (b) User types a command then types the RETURN key.
  - (c) Computer runs or "executes" the command.

UNIX Command-Line: What Can It Do?

- Everything you are used to doing with a GUI system can be done with a command-line, and more!
- For example, the command-line can be used as a file manager:
  - create files and folders
  - copy and move files around
  - show file and document contents
  - upload/download files from servers
  - search for information in folders, files, and documents
  - send documents to the printer
  - permanently delete files and documents

UNIX Command-Line: What Can It Do?

- Everything you are used to doing with a GUI system can be done with a command-line, and more!
- There are no menus to list the commands, so the user needs a reference manual to look-up all the commands.
- Common ones are memorized through repeated use.
- It's a tool you have to practice, and with practice, you will see its power and utility.
- This stuff is worth learning. It can save you many hours.

UNIX Command-Line: Why Bother?

- The UNIX command-line gives you access to hundreds of utility programs written by programmers for programmers and made part of the standard UNIX tool-set.
- If you know some UNIX tools, you can automate many tasks, and avoid writing Python programs.
- Being productive requires investing time to learn the tools. Learning UNIX tools is a one-time investment that pays off many times in the future.

UNIX Command-Line: The Interaction

- The command-line app (CL) is simply a loop.
- When the CL is ready to run a command, it will display a command prompt.
    - To issue a command, you type the name of the command, followed by RETURN.
    - The CL "runs" or "executes" the command. The command may produce text output to the console.
    - When it is finished, it will display the command prompt again.
- The format of the command prompt varies from system to system.
    - It might show your NSID, or some other information.
    - UNIX prompts often have the character $ or % near the end of the prompt.

UNIX Command-Line Tools: The basics

- UNIX tools are very much like functions: they take inputs and produce outputs.
- Each tool is specialized to do one limit task very well.
- Inputs are usually in the form of text typed on the command line, or stored in a file.
- Outputs are usually in the form of text displayed to the console, or written to a file.
- To use a UNIX tool you need to tell the tool exactly where to look for the inputs.
- This is unlike a GUI tool that allows you to choose a file using a file chooser, or that allows you to choose some options using a dialog or menu.

UNIX Command-Line: The Context of a Command

- This concept is very important.
- A command is almost always a single word, or an acronym, related to the purpose of the command.
- The context for a command is the environment in which the command is executed.
    - Context is not represented visually, so you have to be aware of it, and keep it in mind.

UNIX Command-Line: The Context of a Command

- This concept is very important.
- An important aspect of context for a command is the folder (or directory) in which you are working.
    - This is known as the current working directory.
- Note: The words *folder* and *directory* refer to the same thing; *folder* is more modern, but the command-line often uses the older term *directory*.

Paths, Relative Paths, and Absolute Paths

- In UNIX, a path is a sequence of folder names, separated by '/' (forwardslash), that describes the location of a file or a folder.
  - Windows uses '\' (backslash) for a similar purpose.
- A relative path is a path that starts from the current working directory.
  - The path will describe a different location if you change working directories.
- An absolute path is a path that starts from a fixed, known location called the root.
  - Does not depend on the current working directory at all.

Basic UNIX command-line commands

- `pwd` ("print working directory.") displays the path from the root to your working directory.
- `ls` lists the contents of a folder with the command.
- `mkdir` creates a new folder in the current working directory.
- `cd` ("change directory") allows you to change (or "move to") your working directory.
- `more` will display the contents of a named document to the command-line window.

A few other UNIX command-line commands

- `clear` clears the command-line window.
- `date` displays the date in the command-line window.
- `whoami` displays the user's login name.
- `!!` repeats the previous command, exactly.
- `python3.6` starts up a Python interactive session in the command window.

Part II

Logging In and Becoming Familiar with the Lab

Linux Apps You'll Find Useful for CMPT 145

- **PyCharm** (Python Environment)
- **Firefox** (web browser)
- **Chrome** (web browser)
- **Kate** (programming editor)
- **Vim** (programming editor)
- **Nano** (programming editor)
- **Terminal** (command line interface)

Software You Don't Need in CMPT 145

- **Microsoft Word**
  - Word documents are full of junk that we don't want in our Python programs
  - Write programs with PyCharm or any other text editor mentioned peviously
  - Write simple explanations or discussions using any program that saves to .txt

Part III

An Introduction to the UNIX Command-Line

UNIX Command-Line: Getting Started

- On Mac and Linux, the command-line interface is essentially the same, as both systems are UNIX variants.
- **Mac**: Open Finder, go to "Applications", "Utilities", and run "Terminal".
- **Linux**: Right-click on the background, choose "Konsole."

ACTIVITY: Open your command-line!

Note: Keep the Terminal window open until you are completely done with the lab. You'll copy/paste all the text in the window and upload it to Moodle. Further instructions at the end of the lab slides.

Determining the Working Directory with `pwd`

ACTIVITY: To find out the directory (or folder) in which you are currently working, type `pwd` in the command prompt, followed by the RETURN key.

```
% pwd
/home/abc123
%
```

- The command `pwd` abbreviates "print working directory."
- The command displays the path from the root to your working directory.
- This is the context for the commands you type.
- In the Spinks labs, you will see a path that has your NSID on it. It may appear different on your own Mac or Linux computer.

Listing the Contents of a Directory with `ls`

- On the command-line, you can list the contents of a folder with the `ls` command.
- ACTIVITY: Use the `ls` command by typing it into the command-line.
- By default, `ls` lists the contents of the current working directory. Depending on your context, you will see different contents.

Options!

- Most commands have a default behaviour. You can modify behaviours by adding "options" to the command.
- For example, `ls` does not show hidden files[1] by default.
- ACTIVITY: Type `ls -la` to reveal hidden files and extra information
  - Notice the special folders whose names are '.' ("dot") and '..' ("dot dot").

[1] In UNIX, a file is hidden only for tidiness, not for secrecy! Hidden files in UNIX are typically configuration files, and other meta-information.

Creating Folders with `mkdir`

- The command `mkdir` creates a new folder in the current working directory.
- ACTIVITY: Type `mkdir cmpt145` to create a new folder named "`cmpt145`".
- ACTIVITY: Use `ls` to check if the folder was created!
- Note: Spaces are meaningful to the command-line. If you type the command `mkdir cmpt 145`, you'll get two new folders (`"cmpt"` and `"145"`), not one with a space!
- `mkdir` is an example of a command that requires an argument.

Changing Folders

- An important aspect of a command's context is the folder in which the command is issued.
- It is possible to "move" to a different folder, and the new folder will be the context of commands that follow.
- On the command-line, this can be done with the command `cd`.
    - The command is an acronym for *change directory*.
- ACTIVITY: In the Terminal, change your working directory to the folder you created earlier, by typing `cd cmpt145`.
    - This is another example of a command that can take an argument.
- Type `pwd` to verify that it changed successfully.

Creating New Files

ACTIVITY:

(a) Open the text editor (TextWrangler)

(b) Type some text into the editor window. It doesn't matter what you type here!

(c) Save the text as a file named "`lab1file.txt`" in your "`cmpt145`" folder.

(d) On the command-line, use the command `ls` to verify that it is there.

...and more!

We can scroll through a text file with the `more` command.

ACTIVITY:

(a) Typing `more lab1file.txt` into the command -line will display the file you created earlier in the command window.

(b) You should see all the text you typed.

(c) If you typed more than can be seen in a single window, `more` will limit the display to what fits in the window. To see more[2] of the file, press the SPACE BAR.

[2]Computer science jokes, what fun!

More on Context

- When we typed `more lab1file.txt`, we only referred to the name of the text file.
- That's because the command-line's context includes your current working directory.
- If we want to access a file outside the current working directory, we need to know where the file is (more on this later).

Current Folder and Parent Folder

- Every UNIX folder contains two special folders, ‘.‘ (dot) and ‘..’ (dot dot).
    - ‘.‘ (dot) refers to the current folder, (e.g. “this folder”).
    - ‘..’ (dot dot) refers to the parent folder, (e.g., “the folder this folder is in").

Activities

- ACTIVITY: On the command-line, type `cd .` (dot), then check the path. There should be no change!
- ACTIVITY: On the command-line, type `ls .` (dot). You should see the files in the folder. The default behaviour of `ls` with no argument is the same as `ls .`
- ACTIVITY: Now type `cd ..` (dot dot), then check the path again. List the contents of the current directory.
- Summary: to enter a folder, use `cd` with the folder's name; to go back, use `cd ..`

Paths, Relative Paths, and Absolute Paths (Recap)

- In UNIX, a path is a sequence of folder names, separated by '/' (forwardslash), that describes the location of a file or a folder.
- A relative path is a path that starts from the current working directory.
    - The path will describe a different location if you change working directories.
- An absolute path is a path that starts from a fixed, known location called the root.
    - Does not depend on the current working directory at all.

Paths, Relative Paths, and Absolute Paths

ACTIVITY:

(a) Open Finder, go to your "`cmpt145`" folder.

(b) Make a new folder by right-clicking inside the "`cmpt145`" folder, selecting "New Folder", and naming it "`lab1`".

(c) Right-click the "`lab1`" folder, select "Get Info" to check its path (under "General" > "Where").

(d) On the command-line currently examining the "`cmpt145`" folder, switch context to the new "`lab1`" folder by using a path relative to the current one: `cd lab1`.

(e) This is possible because the "`lab1`" folder is directly inside the "`cmpt145`" folder.

Paths, Relative Paths, and Absolute Paths

ACTIVITY:

(a) You should be in `lab1`.

(b) Type `cd ..`. You should be in `cmpt145`.

(c) Type `cd ..` again.

(d) Type `ls`. You should see the folder `cmpt145`, and perhaps some other files.

(e) Type `cd cmpt145/lab1`. The path `cmpt145/lab1` tells the `cd` to follow the whole (relative) path.

(f) Type `cd ../..`, then `ls`. Paths can go “up” as well as “down.”

History of Commands

- Typing lots of commands can become tedious.
- Modern UNIX command-line interfaces save previously entered commands, and allow you to reuse them quickly.
- ACTIVITY: Press the UP-ARROW and DOWN-ARROW keys to cycle through your history. From there, it is possible to edit or use a previous command. Experiment with the LEFT-ARROW and RIGHT-ARROW keys as well!

There is a lot more to learn about command-lines, but it's easy and once you master it, it is very powerful.

About Spaces in Document Names, and Folder Names

- Names of files and folders are allowed to contain spaces.
- On the command-line, spaces are used to separate different parts (“arguments”) of commands.
  - The UNIX system treats a space as a separator, *unless the space is preceded by a backslash* ‘\’.
  - The backslash[3] tells UNIX that the space is not a separator, so it can be part of a file or folder name.

[3] Careful! The backslash and the forwardslash mean very different things!

Actitivy

- ACTIVITY: Make a new folder called "`test folder`" and change to it. Don't forget the backslash (we didn't put it in yet; you have to do it yourself)!
- Generally, until you are reasonably familiar with the command-line interface, it's wise to avoid files and folders with spaces in the names.
- Use the underbar character '_' instead!

Summary

We will introduce other commands as we need them.
In this part, we managed files using the command-line:

(a) In the default directory (home directory), we created a folder named "`cmpt145`"
(b) Within folder "`cmpt145`", we created a file (using a text editor) named "`lab1file.txt`"
(c) We created a folder named "`lab1`" under "`cmpt145`"

Commands used:

- `pwd`
- `mkdir`
- `cd`, Variations: `cd .`, `cd ..`
- `ls`, Variations: `ls -l`, `ls -la`
- `more`

Part IV

Hand In

What To Hand In

(a) Open TextWrangler (or any text editor you want)

(b) Keep the Terminal (command-line) window open. Select all the text (COMMAND-A) and copy it (COMMAND-C).

(c) Paste (COMMAND-V) the lab work you did into TextWrangler.

(d) Save your work as “`Lab01_transcript.txt`”.

(e) Upload “`Lab01_transcript.txt`” to Moodle.

Grading

- If you've performed all the commands marked as ACTIVITY in this lab, you'll get full marks.
- If you hand nothing in, you'll get zero marks.
- Your transcript will probably show evidence of commands being used incorrectly if you've misunderstood something. That's perfectly fine; no marks will be deducted.