Unix Workbench Notes

Haggai Liu

April 24, 2018

1 Unix and Command Line Basics

1.1 What Is Unix?

Unix is an operating system and a set of tools. one particular tool is known a a **shell**, which is a computer program that provides a command line interface. Using the command line interface lets you enter lines of code into a shell (also called a console) and that code instructs your computer to perform a specific task. The terms **command line**, **shell**, and **console**, are often used interchangeably.

The shell is a very direct and powerful way to manipulate a computer. We can produce wonderful creations that help thousands of people, or we can wreak havoc on ourselves and on others.

There are several popular shell programs but Bash is the default shell program on Mac and Ubuntu.

1.2 Getting Unix

Mac and Ubuntu users can simply open up the Terminal. Windows 10 users can enable and install Bash on Ubuntu on Windows. Users of earlier Windows versions can use VirtualBox.

1.3 Command Line Basics

1.3.1 Hello Terminal!

Once we fire up a Terminal there will be a prompt like the following:
`<machine-id>:~<user>$`

Commands are typed right after the prompt. Enter is pressed after the command to execute it. If no command is typed before pressing enter, nothing happens and a new prompt appears directly below the old prompt. All those prompts appearing on the screen after several enter presses may begin to look messy. Those prompts can be cleared using the `clear` command.

Every command line command is actually a little computer program, even commands as simple as clear. These commands all tend to have the following structure:
`[command] [options] [arguments]`

Some simple commands like clear don't require any options or arguments. Options are usually preceded by a hyphen (-) and they tweak the behaviour of the command. Arguments can be names of files, raw data, or other options that the command requires. A simple command that has an argument is echo. The echo command prints a phrase to the console. An example is given below. Surround the message with single or double quotes.

```
:~$ echo "Hello World"
Hello World
:~$
```

To see the last command press the Up arrow key. We can press Up and Down in order to scroll through the history of commands that were entered. If we want to re-execute a past command, we can scroll to that command then press Enter.

Summary:

- Commands are typed after the prompt.
- `clear` cleans up the terminal screen.

- `echo` prints text to the terminal.
- Can scroll command history via Up and Down arrows.

Exercises:

1. Print you name to the console.

   ```
   echo "Haggai Liu"
   ```

2. Clear your terminal after that.

   ```
   clear
   ```

1.3.2 Navigating the Command Line

We now discuss how files and folders are organized on the computer.

Computers are organized in a hierarchy of folders, where a folder can contain many folders and files. People who use Unix often refer to folders as directories and these terms are interchangeable. This directory hierarchy forms a tree. The command line can be used to navigate these trees on your computer.

There are a few special directories that we should take note of. The directory at the top of the tree is called the root directory. The root directory contains all other directories, and is represented by a slash (/).

The home directory is another special directory that is represented by a tilde (~). The home directory contains personal files, like photos, documents, and the contents of the desktop. When we first open up the shell we usually start off in the home directory. Imagine tracing all of the directories from your root directory to the current working directory. This sequence of directories is called a path. Below gives an example of the path from a hypothetical root directory to the home directory.

```
/home/hliu
```

Whatever directory the shell is in is called the working directory. The `pwd` command prints the working directory. The working directory can be changed via the `cd` command. Using it without arguments changes the working directory to the home directory.

To change the working directory to a directory other than the home directory, we need to provide `cd` with the path to another directory as an argument. We can specify a path as either a path that is relative to your current directory, or you can specify the absolute path to a directory starting from the root of your computer. If we simply want to change the working directory to one of the folders that is inside our home directory, then first we need to be able to see which folders are in our working directory. We can list the files and folders in a directory using the `ls` command.

Exercise:

1. Set your working directory to the root directory.

   ```
   cd /
   ```

2. Set your working directory to your home directory using three different commands.

   ```
   cd
   cd ~
   cd /home/hliu/
   # can also use relative paths as appropriate
   ```

3. Find a folder on your computer using your file and folder browser, and then set your working directory to that folder using the terminal.

   ```
   cd /abs/path/to/desired/folder
   ```

4. List all of the files and folders in the directory you navigated to in 3.

   ```
   ls
   ```

1.3.3 Creation with Inspection

- `mkdir` command
  - Creates a new directory.
  - Takes, as argument, the path of the directory
- `touch` command
  - Creates a new blank file
  - Takes the name of the new file as an argument
- `ls` command alone does not differentiate between files and folders
- Need to run `ls` with the `-l` option to get a long listing of files in a directory.
  - one line per file
  - Each line begins with a string of 10 characters, each of which is a letter or a dash.
  - If first character is `d` , then it's a directory. Otherwise it's an ordinary file.
  - The other 9 characters represent permissions.

2 Working with Unix

3 Bash Programming

4 Git and GitHub