BSCI238G - Homework 1

Skylar Chan

Due 2023-02-03

Contents

1 Introduction

This homework should be pretty easy as it is an example of homeworks operate in general. This class uses a hybrid-flipped model. In class I will walk various concepts of Bash and nontrivial commands. In homeworks you will practice what we learned and learn easier commands. There are also some **Extra Credit** questions, which you can try if you have time. Feel free to ask for help if you get stuck!

To submit your answers to this homework, write your answers in any text document and submit them via ELMS.

Every command is composed of a few parts. Let's look at an example SSH command.

```
$ ssh -Y you@grace.umd.edu
```

The dollar sign is the prompt, which is where you enter your command. The first word is the name of the command itself, in this case SSH. The rest of the command is the arguments. For example `you@grace.umd.edu` is an argument that describes who and where to login as. *Options* (aka *flags*) are special arguments, usually starting with a dash (-). In this example `-Y` means that SSH should use trusted X-server forwarding so you can use graphical programs remotely. Many commands typically have a `-h` or a `--help` command, but this is not standard.

In fact, one of the difficulties of learning Bash and the Unix commands is that the commands are *not* standard. Different commands have different ordering of arguments, different options, different behaviors, and so on. This makes sense because each command does something different, but just as in your life science classes, there's a lot of memorization and details that aren't always consistent. This has been a common criticism of Unix since the beginning (further reading), but that is just the way it is.

You can learn commands more quickly and effectively with the right technique. This homework will show you how to learn new commands. When I say to run a command, type the full command as it appears and press `enter` (do not type the special characters at the beginning such as $ or %). All commands are case-sensitive. If you make a typo when entering your command, you can press backspace and fix your mistakes. You can also press the up and down arrows to cycle through your command history. Although you can copy/paste commands from this document into the terminal (and it is safe for this assignment), I don't recommend it for reasons I will explain later. But you can copy and paste from the terminal to another document (usually with Shift+Control+C). Or you can take a screenshot of the terminal, this will be fine for the first few homeworks.

2 Installing software

Installing software can be hard. Please reach out if you get stuck.

While you can always SSH through Powershell or your terminal app, we're going to install a terminal with an X-server (explained later) so we can run graphical programs on Grace. Follow the instructions in `https://www.cs.umd.edu/~nelson/classes/resources/GraceSystem.shtml` to install and use MobaXTerm or XQuartz, depending on your OS. Make sure you can SSH into Grace this way and that you can run emacs with the GUI.

We're also going to install Bash and command line tools to use locally on your own computer so you don't always have to login to Grace to run commands.

Windows: install Git for Windows (`https://git-scm.com/download/win`). Although most people download Git for Git itself, we are more interested in the Bash and other tools which come with it. Make sure you can open Git Bash from the Start menu.

MacOS: Bash is already installed but might not used by default. I strongly recommend changing your default shell to Bash by opening a terminal and running the following command. This is so you don't have to remember to run `bash` every time you open a terminal (there are also some subtle differences between Bash and Zsh but that's an extra credit).

```
% chsh -s /bin/bash
```

You can tell if you are running Bash if the prompt has a $ instead of a %. As for the CLI tools, they should be installed by default. As mentioned in class they may have slight differences from the GNU counterparts due to their BSD roots, but hopefully that won't cause too many problems.

Now for the real homework. Make sure you are logged into Grace before starting.

3 How to learn a command

Let's use `whoami` as an example.

1. What do you get when you run `whoami`? (the exact output)
2. Let's get some help about `whoami`. Run `whoami --help`. There are two options for this command. What are they? (explain in your own words)
3. What version of `whoami` are you running? (Hint: run `whoami` using the other option (starting with `--`))

Typically `--help` will not contain everything you could possibly learn about a program. For that you will need to read the fantastic manuals. Most programs come with manual pages, or man pages for short. Man pages are sorted by sections according to their purpose. Many commands come with a man page that describes the options available to use for a program. Some man pages also come with examples.

There are three commands related to man: `man` itself, `whatis`, and `apropos`. Run the following commands and explain in your own words what each of `man`, `whatis`, and `apropos` do.

4. `whatis whatis` ?
5. `whatis man` ?
6. `whatis apropos` ?

Here's how to read a manual page.

1. `man whatis`. This opens the manual page for `whatis` using the default *pager*. A pager is a program that reads text. By default Grace uses the `more` pager.
2. Press `enter` to move the page down.
3. Press `q` to exit.

Some manual pages such as `whatis` are pretty short. Others however...

7. Run `man bash | less`. Here we use a *pipe* (vertical bar, typically Shift+Backslash (\)). Don't worry about what it does yet, for now just know that it lets us use the `less` pager, which is an improvement over `more` (because `less` is more!). Scroll to the end of the man page using the down arrow or the page down key. Look at the bottom of the terminal window. How many lines long is it?

When you are done, press `q` to exit. We will configure our shell to use `less` by default later.

In contrast to man pages, the info pages tend to be more detailed, and are essentially digital textbooks. Not as many programs come with info pages due to historical reasons, however most GNU programs come with info pages, which is great for our course. Info can also read man pages (try `info man`) which can come in handy if a man page references many other man pages. Navigating an info page is very different from navigating a man page.

To use info:

1. Run `info`, which will show a menu of all installed info manuals.

2. Press `/` (backslash), type `bash`, and press `enter`. This searches for the word "bash" in the info page, and the cursor should jump to the bash info entry.

3. When a cursor is over a line starting with a `*` that means it's a cross-reference (a link) to another page. Press `enter` to jump to the Bash info page.

4. Press the down key to scroll down the manual until you reach "Introduction", then press `enter`. Use the the square brackes (`[` and `]`) to change pages (aka nodes), and use the arrow keys to move your cursor.

5. Press `q` to exit.

Now run `info bash` and read sections 1.1 and 1.2.

8. According to section 1.1, what two shells does Bash incorporate useful features from?

9. What are the commands section 1.2 lists as shell builtins?

When you are done, press `q` to exit.

There are many, many shortcuts for using less and info, which you can explore on your own. Here are some other manuals you might want to read.

- `man intro`
- `info coreutils`

Man and info pages are also available online, use your favorite search engine to search 'man/info page for X program' or similar. Speaking of search engines, the web can be a great resource if the man/info pages are too difficult to understand.

4 Investigating the system

We already saw `whoami` as an example command. Let's learn some more commands that print system information (in fact these are the basis of so-called "fetch" scripts like Neofetch). Tip: use the scroll wheel in the terminal window to view the output of long commands.

10. `uname` prints certain information about the operating system. An operating system is a program that runs other programs.

    (a) What is the output of `uname -a` on Grace?

    (b) Based on the output, name two differences between the operating system of Grace and your local computer.

11. `lscpu` lists CPU information. The CPU (central processing unit) is the brain of the computer.

(a) Run `lscpu` on Grace and report the following:

- Architecture
- Number of CPUs (the `nproc` command should return the same number)
- Vendor
- Model name
- Hypervisor Vendor (meaning that the OS is *virtualized*, or running atop another OS)

(b) Compare Grace's CPU with your computer's CPU using `https://cpu.userbenchmark.com/`. Provide the link for the CPU comparison. Which computer has a more powerful CPU?

12. `free -h` lists the amount of memory (RAM) currently used by the system. Run `free -h` on Grace. How much total memory does Grace have? Ignore the "Swap" row. Does your computer have more or less RAM?

There are many other interesting commands to analyze the system:

- `uptime`, lists how long the computer has been powered on and the load average, or the average CPU demand of a computer over time.
- `who`, `finger`, and `pinky` are more programs for seeing information about logged-in users.
- For an interactive task manager, try `top` (press `q` to exit).

5 Time and date

There are some fun commands such as `cal` but a more interesting one is `date`. By default it prints the current time and date, but the output can be controlled using various sequences starting with `%`. Each instance of a sequence is substituted with something else. For example, `date +%s` uses the format string `%s`. This `%s` is substituted with is the number of seconds since the (Unix) Epoch (1970-01-01 00:00 UTC). Therefore `date +%s` will return the number of seconds since 1970 (over 1.6 billion!)

13. **Extra credit:** Create your own custom `date` command using a format string containing at least 2 different sequences. Report the output of your command and explain in your own words how your format string prints the time.

6 X-server forwarding

So why did you login using MobaXTerm, or XQuartz with `ssh -Y`? Because we are going to use trusted X-server forwarding. Basically, the X-server (aka Xorg or simply X) has been the standard display server on Linux/BSD systems for decades and will not be going away any time soon. As its name implies, the X server has a client/server model. One computer

runs an X server and another computer (maybe the same one) connects to the server, either locally or remotely. In our case, Grace runs an X server, and you are the client. This means we can run graphical Linux applications from Grace on our local computer. Not all servers run an X server or have graphical apps available, so consider this a luxury!

14. Submit a screenshot where you are running `xeyes`, `xclock`, and `xlogo` at the same time. You must put your mouse inside the `xeyes` window (use a keyboard shortcut to take a screenshot). You'll need to open three terminals and ssh into Grace with each of them. Be sure to clearly show each terminal with the full text visible.