Introduction to Linux, a self-guided learning using LinkedIn Learning.

By Albert K Tai

Summary

Linux was originally developed as an open source (freely available and everyone could contribute to the code) alternatively to Unix. Yet, it becomes the most ubiquitous operating system (OS) and can be found on various electronic devices, from large computing cluster and desktop/laptop to smartphone, car/airplane/train computing system and smart device (Internet of things a.k.a. IoT)

While modern consumer-oriented Linux distributions (or distro, see video for additional explanation) contains a graphic user interface (GUI, pronounced gooey). All of them contains a command line interface, and many distros can be run on a command line only interface (For those of you old enough to remember, think of Microsoft DOS before the introduction of Windows 3.0/3.1).

Many software and tools utilized in data science (or big data), such as those for bioinformatics or artificial intelligent, are developed and run on Linux-based workstation or high-performance computing (HPC) platform. Thus, experience with Linux and its command line environment (a.k.a. shell) is a fundamental and essential skill set for data scientist.

In addition, the Linux shell contains a collection of effective tools that is extremely flexible and efficiency for working with large data file in text format. Furthermore, it allows automation of common or complex task with scripting. Thus, making Linux shell a very attractive environment for working with data.

This self-guided tour is developed as an online resource within a very short time frame. It is based on a combination of video available from LinkedIn Learning (formerly Lynda, a resource accessible to all members of Tufts University) and short supplementing note for the videos. There are 5 LinkedIn Learning courses listed in this tour and each of them takes between 1.5 to 2 hours. With the first video give you the foundation on working with Linux command line, second optional video give you some basic programming concept. The third to fifth videos demonstrate some advanced and powerful capabilities built-in to the Linux command line interface, which will require some basic training on programming.

IMPORTANT: IF you do not already have a Tufts HPC account, please obtain one before you start. Please follow the instruction of this page (https://it.tufts.edu/it-computing/technology-research/high-performance-computing-tufts-research-cluster) to get one. Once an account has been created for you, you can use the information from this page to help you with logging into the HPC (https://wikis.uit.tufts.edu/confluence/display/TuftsUITResearchComputing/Access) from a terminal (Mac OS X or Linux/Unix) or from a web browser (such as Chrome, Edge, Safari, etc, available to all platforms including Windows) following this instruction here (https://rbatorsky.github.io/intro-to-ngs-bioinformatics/lessons/01_Setup.html). If you are working from home, you will likely need to access these resources via virtual private networking (VPN). There is additional information about VPN here (https://access.tufts.edu/vpn).

Generally speaking, if you are working remotely (outside of Tufts network), you will need to first start the VPN, before you can connect to HPC via terminal or web browser. There will be one or two Zoom sessions to help with troubleshooting initially.

Finally, for all the note below, a command that you are typing is going to be displayed with a different font. For example, if you see

```
pwd ↵
```

please type the three characters pwd into the terminal, and then hit “Enter” key if it is followed by this ↵ symbol. Also, a commands / shortcuts cheat sheet is a helpful tool. I included a few for download. Hope this course becomes a good starting point for your adventure into data science, be it bioinformatics, artificial intelligent of machine learning.

The tour includes the following video courses:
Learning Linux Command Line (2h 18m) by Scott Simpson, with supplementary note.
https://www.linkedin.com/learning/learning-linux-command-line-2/

(Optional) Programming Foundations: Fundamentals (2h 4m) by Annyce Davis, with supplementary note.
https://www.linkedin.com/learning/programming-foundations-fundamentals-3/

Learning Bash Scripting (1h 25m) by Scott Simpson, with supplementary note.
https://www.linkedin.com/learning/learning-bash-scripting/

Awk essential training (2h 1m) by David D. Levine
https://www.linkedin.com/learning/awk-essential-training/

Sed essential training (2h 1m) by David D. Levine
https://www.linkedin.com/learning/sed-essential-training/

Learning Linux Command Line by Scott Simpson
https://www.linkedin.com/learning/learning-linux-command-line-2/

Additional notes/tips associate with the course, as organize by videos:

Introduction

- A very quick intro to Linux
    - Tufts University high performance cluster (HPC) uses RedHat Linux, thus the recommended free (as in free) distro is CentOS 7, or another distro that is RedHat-based may be considered, such as Fedora, Scientific Linux etc

1. Creating a Linux virtual machine

- While a Virtual machine is a somewhat convenience way and rather quick way to get a Linux (virtual) machine going, if you do not have access to one. VirtualBox requires considerable amount of resource (in term of hardware, such as CPU, memory and hard drive space) for a somewhat mediocre to poor performance. Thus, I would suggest not to go this route unless you don't have any other choice (which you do).

- Windows Subsystem for Linux on Windows 10
    - This is a relative recent feature available on Windows 10. However,
- Following along on a Mac
    - Technically speaking, the MacOS X is Unix-based rather than Linux. However, the Bourne-again shell (Bash), the same command line interface, is the default shell for many Linux distro, including Redhat Enterprise, Centos, Ubuntu, etc. Thus, many commands and functions are identical or very similar. Although some software packages are somewhat unique to one or the other, e.g. "say" command on MacOS X. For those of you who are interested in learning more about MacOS X command line, another LinkedIn Learn video of interest is Unix for Mac OS X Users (https://www.linkedin.com/learning/unix-for-mac-os-x-users/, 6h 35m). And I used to recommend this course for learning about BASH before this course comes along. As a matter of fact, I will be using a few videos from the Unix for Mac OS X Users to supplement this course (please look for ☑ symbol)
- Following along on Linux
    - This is the recommended approach for this course. When you sign up for this course, A Tufts High Performance Computing (HPC) account should have been created for you. If not, please use this link (http://research.uit.tufts.edu/) to create one. Please following this instruction () to log onto the HPC and continue with the video.
- Using a cloud provider
    - Unless you have a specific reason and access to a cloud provider, this is not a recommended approach, at least not for this course.

2. Command-Line Basics

- What is the command line?
    - Bash (Bourne-again shell) is also used on Tufts HPC and Mac OS X (Terminal), so you are good.
- How commands are structured
    - This part is extremely important to familiarize with the commands structure. Please review this video and make sure the structure make sense to you.
    - IMPORTANT: Linux (and Unix) is a **case-sensitive** operating system (as supposed to Windows, which is not). Hence, you can have a file named "Test.docx" and another named "test.docx" co-exist within the same directory (a.k.a. folder) on a Linux system. You can even have "tEst.docx", "teSt.docx" in additional to the previous two, and they will be treated as 4 different files. All commands, options and arguments are case-sensitive as we!! **Space** (i.e. pressing space bar) in between commands option and argument is also important. Most of the common errors for beginner users are caused by wrong case or wrong space usage.
- Write commands in a shell at the prompt

- The terminal application is also available on Centos, Ubuntu and MacOS X (search via Spotlight), and many other Linux distros. If you follow the previous instruction and logged onto the Tufts HPC, you are already seeing a shell prompt and you can skip the "terminal" step and type along the commands.

- Helpful keyboard shortcuts in the terminal
  - There are two commands that is good for
  - Tab completion is extremely useful, especially dealing with long file names and to avoid typo. It is also useful to find a file that you don't remember the exact file or command name.
  - Up and down arrows are also used frequently!
- Finding help for commands
  - "man" (stand for manual) is your best friend when you are exploring available option or learning a new command, so is "Google".
  - Press q to quit, or ZZ, :q or Q (remember Linux is a case sensitive environment)
  - For many commands (but not all), you can also use "command -h" or "command --help", to call out an abbreviated manual straight from the prompt, but it is command specifics.

☑ Before proceeding to the next video of this course, please pause and watch the following short videos from "Unix for Mac OS X Users" (around ~15 minutes)

- https://www.linkedin.com/learning/unix-for-mac-os-x-users/the-working-directory
- https://www.linkedin.com/learning/unix-for-mac-os-x-users/listing-files-and-directories
- https://www.linkedin.com/learning/unix-for-mac-os-x-users/moving-around-the-filesystem
- https://www.linkedin.com/learning/unix-for-mac-os-x-users/filesystem-organization?u=2193697

☑ Now you can resume now (https://www.linkedin.com/learning/learning-linux-command-line-2/files-folders-and-navigation)

3. Files, Folders and Permissions

- Files, folder and navigation
  - From the additional videos from another course, you should familiar yourself with the commands: pwd (stands for print working directory) and cd (change directory), as well as some understanding of the file structure, which will be discussed further here in this video.
  - There is no graphical user interface (GUI) on Tufts HPC Bash. Thus, you can
  - ~ symbol (or tilde), refers to the home (default) directory / folder of the current user.
  - As you can see the folder name "Exercise Files", space is troublesome to deal as a folder or file names in Bash. Thus, some experienced users prefer to use underscore instead of space in file and folder names! Pressing "Tab" could be useful here.
  - .. refers to previous level and each directory level is separated by a slash (/)
  - The highest level of directory (folder) is called a root directory and is presented by a single slash (/). There is no directory above root directory!
- A little more about ls: No additional note.
- Create and remove folders
  - Remove directory with care!
- Copy, move, and delete files and folders
  - cp = copy
  - mv = move, also use to rename a file in additional to move a file. Thing of moving the content from one file to another file with a different name.
  - rm = remove, use with care, especially with -r (recursion) and/or -f (force) option!!!!!! If you want to remove file using wild-card (* or ?), use ls with the wild-card combination first and exam the list of files are indeed what you want to delete first!
    IMPORTANT: There is no recycle bin in command line. If rm remove a file, the file is considered gone. (There are ways to recover deleted files… but they are tedious and/or difficult)
  - rmdir could be tedious when removing a directory, but it is a safer option
- Find files from the command line: No additional note
- User roles and sudo
  - Most users on Tufts HPC are general users
  - I do not believe a general user can use "sudo" on Tufts HPC and I would recommend AGAINST trying it.

    - If you need help install software or packages on HPC (which require sudo permission a some of the time), please contact IT for help.
- File permissions
    - Important to understand and remember! Especially when sharing/transferring files between users.
    - You can only run chmod on files / directories that you have permission "write" permission to.
    - I found octal value easier to work with
- Create hard and symbolic links
    - Symbolic links is basically a shortcut
    - Convenience if you don't want to make multiple copy of a large file.
- The Linux filesystem
    - If I were, I won't poke around on the HPC, even if it should be relative harmless
    - If you are lost, use cd ~ (i.e. change directory back to your home directory)

4. Common Command-Line Tasks and Tools

- The Unix philosophy: No additional note
- Use pipes to connect commands together
    - pipe (|) takes the output from preceding command as the input of the following command
    - multiple pipes can be chained together
- View text files with cat, head, tail and less
    - Also, zcat, which is basically cat that can be used on text file that is compressed using gzip (.gz extension) without the need of decompressing the file first.
- Search for text in files and streams with grep
    - No additional note.
    - A little more about grep (5m), https://www.linkedin.com/learning/cert-prep-red-hat-certified-system-administrator-ex200/use-grep-and-regular-expressions-to-analyze-text?
- Manipulate text with awk, sed, and sort
    - Please take a look at the sort command manual and check out the available options, especially -n, -k -u and -r
    - If you have the energy to, please look at commands tr, cut and uniq
    - Awk essential training (2h 1m): https://www.linkedin.com/learning/awk-essential-training/
    - Sed essential training (2h 1m): https://www.linkedin.com/learning/sed-essential-training/
- Edit text with vim
    - You DO NOT need to install vim in Tufts HPC. Vim is already installed. Also, "apt-get" is not the installation command used for RedHat-based distro. Finally, you need to be an administrator to install package normally (there are exceptions, but not the scope of discussion here).
    - vim commands cheat sheet (https://github.com/Praful/vim-cheatsheet/blob/master/vim-cheatsheet.pdf)
- Edit text with nano: No additional note
- Working with TAR and ZIP archives
    - TAR is commonly used with Gzip and is time widely in used as a mean to package and distribute data or software. It is also convenience and economic for sharing results.
    - Zip file is available and is better if you intend to share you results to a Windows or Mac OS user.
- Output redirection
    - For redirect output to a text file using 1>filelist.txt, please remember that there is NO space between 1 and >, but there could be a space between > and filelist.txt. Thus, when you use the shortcut as > alone, you can have a space between the > and filelist.txt. Similarly, there cannot be a space between 2 and > when redirecting the standard error to a file.
    - echo prints something to screen (a.k.a. standard out)
- Exploring environment variables and PATH
    - Be wary when modifying PATH variable, it can cause some damage to your working environment. Sometime, by closing the terminal and restarting a new one (or logging in again if you are working with the HPC can reset your PATH variable to default.
    - Any change made to the .profile (or .bashrc for Centos) will be applied every time you start a new Bash session, i.e. login via a new terminal (with or without closing the existing terminal).

- Challenge: Extract information from a text file: No additional note
- Solution: Extract information from a text file
    - No additional comment. But if you work on a personal Linux or MacOS and found attempt to hack into your computer. You may want to be careful that you use a hard to guess username and password combination.

5. A Peek at Some More Advanced Topics

- Find distro and kernel information
    - Some software packages are distro lineage or kernel version specific, or with distro lineage specific installation instruction (for instance, Debian-based vs RedHat-based, apt-get vs yum install).
- Find system hardware and disk information
    - Try to find hardware information on Tufts HPC may not be very meaningful. When you login the HPC, which is a cluster of computers connected by high speed network, the terminal interacting with you is computer commonly refers as to head node. Most of the time computing intensive job is submitted to another computer (or computers) refer as to daughter node via a job scheduler (SLURM for Tufts HPC). Any commands to find hard drive space, cpu or memory information is mostly concerning the head node.
- Install and update software with a package manager
    - Just remember apt-get is for Debian-based distro, and both Tufts HPC and Centos are both RedHat-based distro.
    - Unless you have sudo (sometime refer as to sudoer) or administrator permission. Chances are you cannot install package on any computer, less so on Tufts HPC.

Conclusion

- The Tufts HPC current runs on RedHat Linux, which requires a paid commercial license to run. The freely available CentOS distro contains the packages for corresponding version. RedHat is also commonly used for server of many large organization. Thus RedHat/CentOS are probably the best ones to dive into further. I am personally using CentOS release 7.7 as my main work computer operating system.

Optional: Programming Foundations: Fundamentals by Annyce Davis
https://www.linkedin.com/learning/programming-foundations-fundamentals-3/

Introduction

- The fundamental of programming: No additional note
- Following along with the course: No additional note

1. Programming Basics

- What is programming?
  - A bug is a coding error, and hence removing error from a code is called "debugging". This term has an intriguing origin that involves real insect. Do a Google search if you are interested. :)
- What is a programming language?
  - The important things about this video, and as a matter of fact this entire optional course, is not for you to learn a specific programming language. The key learning point of this optional course, is the concept of the program structure and flow, e.g. repeating a task (e.g. for and while loop), making decision (e.g. if..then..else, case, etc).
  - It is also important to notice that even most high level programming languages resemble human language, each has it own syntax (i.e. grammar)
- Writing source code
  - If you are working on Tufts HPC, you can only use vim or nano as text editor for coding on HPC. If you need refresher please watch (https://www.linkedin.com/learning/learning-linux-command-line-2/edit-text-with-vim) for vim or (https://www.linkedin.com/learning/learning-linux-command-line-2/edit-text-with-nano) for nano.
  - If you are using vim or nano, you don't need to worry about rich text format, which is NOT available.
  - Please make sure you are in your intended working directory (`pwd`↵) before running text editor and save your first code.
- Running your code
  - If you are on HPC, Python should be installed by default.
- Using an IDE
  - IDE stands for Integrated Development Environment
  - I have not used Xcode or VScode, however, you would want an IDE that can works with more than one programming language. Other example including Eclipse. However, most of code you are writing at the beginning probably won't need an IDE.
  - R Studio has some features similar to that of an IDE, but specific for development us R language. (yes, R is also a programming language, specialized for data and statistic-related computation)
- Chapter Quiz

2. Programming Syntax

- Why Python?
  - Python is definitely one of the best choices for a beginner to learn at the moment
  - There are many different top five or top ten lists of programming language. Depending on the source and the rating methodology and metric, but Python is always
- Installing Python on a Mac
  - If you are not planning to use the Tufts HPC (which I strongly recommend you use HPC). You don't need to install Python on you own computer.
  - If you would like to install the VSCode IDE (for later), you WILL need a functional Python installed on your computer. MacOS X (10.8 or above) should have a version 2 installed by default.
  - At this moment, there should not be a big difference between using Python version 2.x.x or 3.x.x.
- Installing Python on Windows
  - If you would like to install and try out the Visual Studio Code IDE (see below), you will need to install a version of Python. Windows family of OS does not have Python pre-installed. Please follow the video instruction to install one.
- Running Python on the command line

    - If you are using HPC, Python 3 is NOT available by default. There are multiple versions of Python available on HPC as modules and you will learn about HPC and modules in a different course. To load Python 3 (version 3.6.0), try typing:
      `module load python/3.6.0` ↵
      `python3 --version` ↵
    - You can follow along with Python 2 (without loading a module) or Python 3 (with Python 3.6.0 module loaded). Make sure you type
      `python` ↵
      or, after loading the python module
      `python3` ↵
      and see the ">>>" prompt before you start.
    - You can type `quit()` ↵ to get out of the Python prompt and back to Bash
    - At this moment, there should not be a big difference between using Python version 2.x.x or 3.x.x.
- Installing Visual Studio Code on a Mac
    - Once again, a version of Python 2 should be included with Mac OS X by default. Installing the VS Code alone should work.
- Installing Visual Studio Conde on Windows:
    - Please make sure you have a working Python installed before installing the VS Code IDE.
- Running Python in an IDE
    - Please download the Excise files as you see fit.
- Basic statements and expression: no note
- Troubleshooting issues
    - Using Python version 2 (python) or 3 (python 3) won't make a different for this video. But you may want to be consistent about the version of python you use at this point.

3. Variables and Data Types

- Introduction to variables and data types: no additional note
    - Many programming languages use equal sign "=" to assign a value to a variable, a few have other means. The one you are going to encounter is R, which can use an arrow "->" to assign a variable, and to make things interesting, the arrow can be from left-to-right (similar to "=" in term of variable being on the left and value being on the right of the assignment), or right-to-left (reverse).
- Variables across languages
    - It is a bad idea to use same word but different cases as two variables. Avoid if possible.
    - Keywords sometime refer as to reserved words
- Working with numbers
    - Other than basic arithmetic symbols, different language could use different symbol or command for representing/calculating exponential, modulus, scientific notation, etc. You may want to double check before you start (for instance, Bash)
- Working with strings
    - Quote and parentheses mispairing are common syntax errors is coding.
- Properly using whitespace: no additional note
    - Just in case you are wondering "=" and "==" means different things in Python. In this case, it is testing if the answer variable contains the text string "Yes". (Is the answer case sensitive? Test it out if you are running the code).
    - Wrong spacing is yet another common error in coding or in command line interface
- Working with comments
    - Comment usually start with # for most language, but not all! Some languages allow block comment as well.
    - Comment out a statement is commonly used for troubleshooting. Print the value contains within a variable during intermediate step is also useful.
- Challenge: What's the output?: no additional note
- Solution: What's the output?: no additional note

4. Conditional Code

- Making decisions in code: no additional note
- Exploring conditional code: no additional note
- Working with simple conditions
    - Always save your code to commit a change before running / testing it!
- Conditionals across languages
    - For reference, in Bash scripting the basic if statement is

```
if [ ] ;
then
Statement1
else
Statement2
fi
```

- Challenge: Guessing game
- Solution: Guessing game

5. Modular Code

- Introduction to functions: no additional note
- Creating and calling functions: no additional note
- Setting parameters and arguments: no additional note
- Returning values from functions
    - The order of values/arguments passed into a function have to be the same as the parameters defined by the function. In the example, value for current_balance HAS to be the first value, and the amount HAS to be the second.
- Functions across languages
    - For reference, in Bash scripting the basic function assumes the structure

```
function_name () {
Local variable
Statement
return variable  # if returning a value from the function
}
```

- Challenge: Favorite cites: no additional note
- Solutions: Favorite cites: no additional note

Conclusion

- Exploring languages: no additional note
- Next steps: no additional note

☑ Just a little bit more. Please also watch (https://www.linkedin.com/learning/learning-python-2/loops , 7 min) to learn another commonly used program control, the loops, the "WHILE loop" and the "FOR loop". This video is based on Python still, and uses the VS Code as an IDE also.

☑ An array js just a collection of variables.

☑ If thing doesn't quite make sense, don't worry too much. I just want to introduce the concept of loop, in additional to conditional IF as a tool for writing code. Loops are particular important when you want to process a text file line-by-line, as well as doing a certain step a given time based on another parameters (e.g. based on number of rows or columns a table has).

Learning Bash Scripting (1h 25m) by Scott Simpson
https://www.linkedin.com/learning/learning-bash-scripting/

Introduction

- Welcome: no additional note
- What you need to know before continuing:
  - Don't worry about Ubuntu (a Debian-based Linux distribution), you should be able to continue on with Tufts HPC (RedHat), Centos or any other distribution. As long as the shell is some version of BASH.
  - I will recommend using the "ondemand" interface Tufts HPC for this course, especially if you are using Windows.
- Using the exercise file: no additional comment

1. Working with the Command Line

- What's Bash? : no additional note
- Reviewing common Bash commands
  - You have learned most of these commands in previous videos!!
  - `more` and `less` are two commands performing similar function. You will find that a number of tasks in bash can be done by more than one command.
- Tilde and brace expansion
  - `~` is a shortcut to your home directory
  - Brace expansion is a powerful mean for pattern matching as well, when you are looking through a large number of files. When use in conjunction with `grep`, brace expansion can also be used to look for certain combination of text with a huge text file (as in 10,000s to 100,000,000s of lines).
- Changing where things go with pipes and redirection
  - Once again, `|` (pipe) uses the output from the commands to the left of the `|` as input for the commands to the right of the `|`
  - Only redirect the output to `/dev/null` when you don't need the output of the results (e.g. log or run status). Sometimes you want to do this because displaying large amount of text on screen could slow down a script in a significant manner. However, if you want to save this information, you should redirect with `1>`, `2>` or `&>` instead.
- Manipulating output with `grep`, `awk` and `cut`
  - If you want to learn more about `awk`, please see the optional course on this command.
- Understanding Bash script syntax
  - `#!/bin/bash` should always be there as the first line of a bash script.
  - `nano` text editor can also recognize the .sh extension as a bash script and apply color scheme to the text. For example, comment in light blue. Color scheme could be different.
- Creating a basic Bash script
  - You can use `bash my.sh` or `sh my.sh` to run a shell script, both should work.

2. Building Bash Scripts

- Displaying text with `echo`: no additional note
- Working with variables
  - NO space between variable, equal sign and the value !!!!!
- Command substitution
  - The output of a command or piped command chain in between the symbol `$(` and `)` will be stored in to the valuable. Some scripts use two backward single quotes `` ` `` instead (above the tab key). For example, instead `d=$(pwd)`, it can be `d=`pwd``. These two commands should produce the exact same result, storing the current working directory (output of `pwd`) into the variable `d`. The former is easier to read!
- Working with numbers
  - Backward single quote `` ` `` CANNOT replaces `$(` and `)` in `$((` and `))` for arithmetic operation!

- Comparing values
  - Space is important in between `[[` and `"cat"`, as well as `"cat"` and =, = and `"dog"`, and `"dog"` and `]]`
- Working with strings
  - For string substitution, you may want to do a Google search for a more comprehensive review
- Coloring and styling text
  - To call/use/return the value of a variable, append a `$` sign in front variable name! Hence, if you assign a value to variable `d`, to print the value to screen, you use `echo $d`
  - Don't worry too much about this section unless you are very into formatting.
- Exploring some handy helpers: `date` and `printf`: no additional note
- Working with arrays
  - Zero-based means the first element is assign an index number of 0, second element has an index of 1, and third element has an index of 2 (banana)
  - Some language us one-base array, meaning that the first element is assign an index number of 1 and so on.
- Reading and writing text files
  - `cat file.txt` works the same as `cat < file.txt` and can be used as a shot cut.
  - Using `cat < file.txt` is a proper syntax but also to contrast with the command showed in the next video
  - Don't worry if you cannot fully grasp the ftp command options.
- Using here documents: not additional note
- Challenge: Make a script that generates a system report
- Solution: Make a script that generates a system report
- Chapter Quiz

3. Control Structure

- Testing truth conditions with the `if` keyword
  - By adding a `;` you can keep the `then` with the same line as the `if` statement. Or you can add the then to the next line. You can write the statement as

```
if [[ $a =~ [0-9]+ ]] ; then
    echo "There are number in the string: $a"
else
    echo There are no numbers in the string: $a
fi
```

Or

```
if [[ $a =~ [0-9]+ ]]
then
    echo "There are number in the string: $a"
else
    echo "There are no numbers in the string: $a"
fi
```

  - Similar concept applies to the loops below.
- Working with `while` and `until` loops: no additional note
- Introducing `for` loop: no additional note
- Selecting behavior using `case`: no additional note
- Using functions: no additional note
- Chapter Quiz: no additional note

4. Interacting with the User

- Working with arguments: no additional note
- Working with flags: no additional note

- Getting input during execution: no additional note
- Ensuring a response: no additional note
- Challenge: Make a script that uses input: no additional note
- Solution: Make a script that uses input: no additional note

Conclusion

- Next steps
    - If you are interested, please continue to the video for awk and sed, advance features that you can use in bash to display and extract information from large table (awk) or manipulate large text file.

Awk essential training by David D. Levine
https://www.linkedin.com/learning/awk-essential-training/
Additional notes/tips associate with the course, as organize by videos:

Introduction

- Welcome
  - No additional note
- What you should know before watching this course
  - If you feel you lack basic programming concept or need a refresher. Please watch the optional video **Programming Foundations: Fundamentals (2h 4m)** by Annyce Davis
  - Text editor, such as vim or nano from course before
- Using the excise files
  - Lydna.com is now LinkedIn Learning

5. What is AWK?

- AWK should already been installed on Tufts HPC, you don't need to install on your own computer, if you prefer not to. I would strongly recommend learning AWK on Tufts HPC