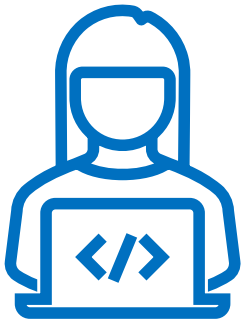


LEARN LINUX IN 28 DAYS WITH STEP-BY-STEP INSTRUCTIONS

JANET YU AND CYNTHIA ZHANG

Table of Contents

Forward

During the summer break of 2023, I was contacted by a high school graduate and a high school student regarding research-based internship opportunities. After learning about their interest and programming experience, I believe an introductory IT related topic will best fit their needs. Therefore, we started this Linux book project.

Based on the material I provided, they created this Linux tutorial for entry-level programmers. It is worth noting that this book is not about the comprehensive background knowledge of the applications. Instead, it is about how to use these applications under Linux.

This book not only can be used by self-taught programmers, but it can also be used as the lab component of a college-level system programming course. Personally, I probably will utilize some of the practices in my college teaching.

Congratulations to Janet and Cynthia for their achievements!

Liguo Yu, Project Supervisor @ South Bend, Indiana
August 2023

Preface

Linux is a multi-user operating system widely used in academia, government, and industry. Knowing how to use Linux and its applications is becoming a necessary skill for all computer science students and IT professionals. This book provides detailed instructions to learn core Linux functionalities and applications in 28 days.

Content layouts

- From Day 1 to Day 7, we practice using Linux commands.
- On Day 8 and Day 9, we learn Emacs and C++ Makefile.
- On Day 10, we learn vi and Maven.
- On Day 11, we learn HTML.
- From Day 12 to Day 14, we practice shell script programming.
- On Day 15 and Day 16, we learn network utilities.
- From Day 17 to Day 19, we practice security applications.
- On Day 20, we practice cloud computing.
- On Day 21 and Day 22, we practice version control and configuration management tools.
- On Day 23, we practice Python programming.
- From Day 24 to Day 26, we practice Perl programming.
- On Day 27 and Day 28, we learn Latex.

Prerequisites

First, you need to have a Linux machine or a Linux account of a remote Linux machine. Second, you need to install the required software programs on Linux machine. Most of the applications used in this book are in the standard installation by default. The practices given this book have been tested on Oracle Linux Server, Version 7.9.

We assume you are using a windows-based operating system, then you need the following two programs to remotely access the Linux machine.

- Putty
- WinSCP

Download the code samples

You can download the code files for this book from https://github.com/linuxbookresource/files.

Janet Yu and Cynthia Zhang
August 2023

Day 01: Introduction

The UNIX operating system was born in the late 1960s at Bell Labs. It has since grown to become one of the most widely used operating systems for mainframe computers. Since UNIX was first developed, it has gone through many different generations and mutations, which include BSD, HP-UX, MAC OS, and Linux.

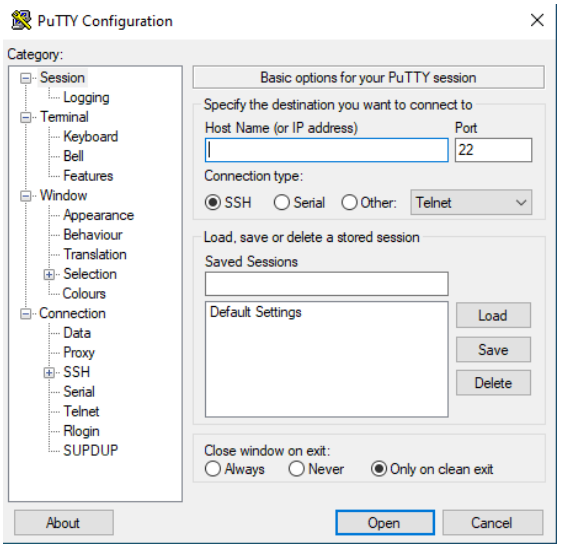
Linux is an open-source multi-user operating system originated based on the idea of UNIX. It was introduced in 1991 by Linus Torvalds at the University of Helsinki, Finland. Now, Linux is widely used by many companies, universities, and government agencies. Some of the benefits of using Linux are listed below.

- Linux kernel is free. The source code of Linux kernel is available, and anybody can write their own Linux if they include the source code in the distribution.
- Linux is considered a stable and reliable operating system.
- Linux supports multi-tasking applications and multi-user logins.
- The standard installation of Linux includes many utilities and APIs, such as the g++ compiler, OpenGL, Git, etc.
- Linux also supports many open-source free applications, such as Maven.

Therefore, knowing how to use Linux and its applications is becoming a necessary skill for computer science students and entry-level IT professionals.

Practice

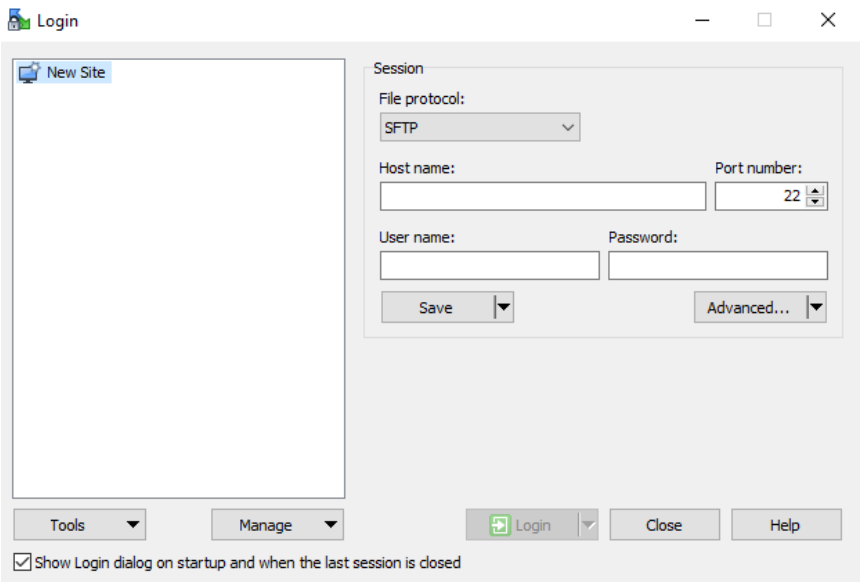
1. Assume you are using a Windows-based operating system and there is a Linux machine running remotely.
2. Download and install Putty:
   https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
3. Log onto the remote Linux machine using Putty. In Host Name, enter the Linux hostname or its IP address. Click “Open”. Then enter your username and password.



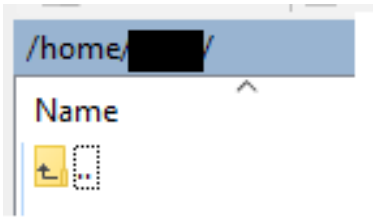
4. You will be logged in and see the command prompt.
   $

5. Log out from Putty using the following command.
   $ exit

6. Download and install WinSCP: https://winscp.net/eng/docs/guide_install
7. Log onto the remote Linux machine using WinSCP.
8. In Host name, enter the Linux hostname or its IP address. Then enter your username and password. Click “Login”.



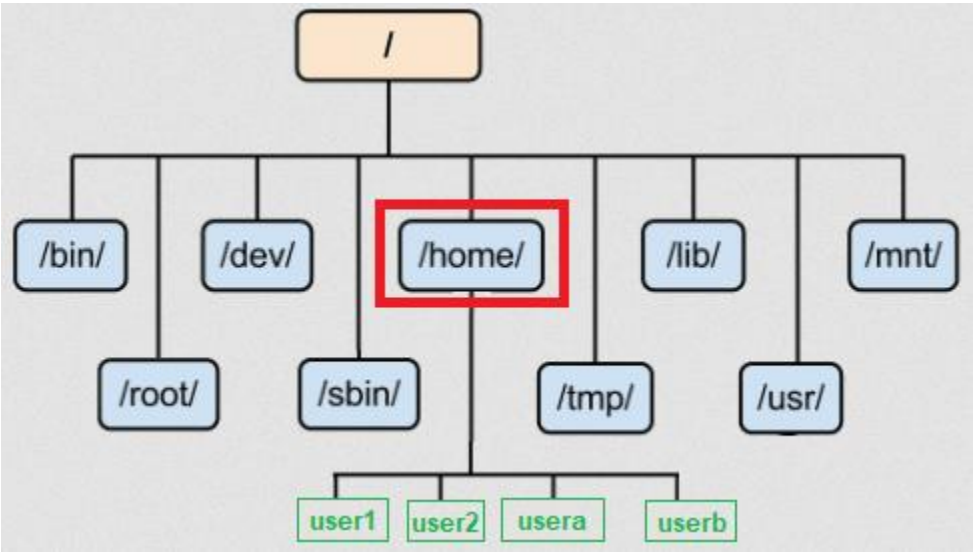
9. You are then logged onto the filesystem of the remote Linux machine. Currently, you are at your home directory.



10. Log out of WinSCP through terminating the program.

Day 02: Linux Command (Part 1)

The following chart shows the structure of Linux File System.



Some important directories are described below.

/	root directory
/bin	executable programs
/dev	devices
/home	directories for most users
/lib	shared libraries
/mnt	mountable devices

Every Linux user has a home directory that can be found under /home/username. For example, **/home/user1/** is the home directory of user1.

The environment variables in Linux are global variables that are defined within the shell and can be used for many purposes. For example, $PATH is an environment variable. To display its value, you type the following command.

$ echo $PATH

In general, a Linux command is a program or utility that runs on the command line. Here are some communally used Linux commands.

- **mkdir:** create a directory
  - Example: mkdir LinuxBook

- **rm:** delete files
  - Example: rm *.txt
- **rmdir:** delete a directory
  - Example: rmdir LinuxBook
- **ls**: list contents of a directory
  - Example: ls *# list content of current directory*
  - Example: ls /usr/ *# list content of a specific directory*
  - Example: ls -l *# list content of current directory with detailed # information*
  - Example: ls -l /bin/ *# list content of a directory with detailed # information*
- **cd:** switch working directories
  - Example: cd ./homework
- **pwd:** display current working directory
  - Example: pwd

Some special expressions in commands are given below.

Current directory: .

Parent directory: ..

The user's home directory: ~

Root directory: /

Path is the way users can access a file or directory. Basically, it tells where in the filesystem we can locate the specific file or directory. There are two kinds of path: absolute path and relative path.

Absolute path is a sequence of directories to reach a file starting from the root directory /
Example: **/home/usera/linuxbook/cat.jpg**

Relative path is a sequence of directories to reach a file starting from the current working directory **./**
Example: **./Lab1/cat.jpg**

The absolute path has the same expression no matter what the working directory is. The relative path depends on the current working directory.

Practice

1. Log onto the remote Linux machine with Putty.
2. Execute the following Linux commands one by one. Make sure you understand every command.

```
$ pwd
$ ls
$ mkdir LinuxBook
$ ls
$ cd LinuxBook
$ pwd
```

3. Start WinSCP.
4. Log onto the Linux machine with WinSCP.
5. Go to folder LinuxBook.
6. Download an online image (in jpg format) to the desktop of your local computer and rename it as "Lab02.jpg".
7. Transfer the image file from your desktop to LinuxBook folder of the Linux machine using WinSCP.
8. Under Putty, execute the following Linux commands.

```
$ ls
$ ls -l
$ cd ..
$ cd LinuxBook
$ mkdir Lab02
$ cd ./Lab02
$ ls
$ cd ~
$ pwd
$ cd /
$ ls
$ cd $HOME
$ pwd
$ cd ./LinuxBook/Lab02
$ cd ..
$ rm Lab02.jpg          # type 'y' to confirm the operation if asked
$ rmdir Lab02
$ ls
$ cd ..
$ pwd
$ ls ~/LinuxBook/
```

9. The last Linux command displays the content of directory LinuxBook, which is empty as shown below.

```
$ ls ~/LinuxBook
$
```

10. Log out Putty and WinSCP.

Day 03: Linux Command (Part 2)

Today, let's continue to work on Linux commands.

- **date**: display or set the system date.
  - Example: date
- **echo**: display anything passed to this command as an argument on the terminal.
  - Example: echo Hello World!
  - Example: echo $PATH
  - Example: echo $HOME
- **cat**: display the entire content of a file.
  - Example: cat lab1.txt
- **more**: display the entire contents of a file one page at a time. Spacebar moves through the file one page at a time. Enter key moves through file one line at a time. You can only move forward through a file, not backward.
  - Example: more lab1.txt
  - Example: q *# terminate the more program.*
- **less**: nearly equivalent to the more command but allow you to move forward and backward in the file.
  - Example: less lab1.txt
  - Example: b *# move backward one page.*
- **man**: display manual of a command.
  - Example: man ls
- **mv**: rename or move files. It requires two parameters: name and path of the original file and new name and path of the file.
  - Example: mv ~/a.pp ./ *# move "a.cpp" from home dir to current # dir.*
  - Example: mv ~/a.cpp ~/b.cpp *# rename "a.cpp" in home directory to # "b.cpp".*
- **cp**: copy files. It requires two parameters: path and name of the original file and path and name or the new file.
  - Examples: cp ~/a.pp ./ *# copy "a.cpp" of home dir to current dir.*
  - Example: cp ~/a.pp ~/b.cpp *# copy "a.cpp" in home directory to # "b.cpp".*

Practice

1. Log onto the Linux machine with Putty.
2. Make sure there is a sub-directory LinuxBook in your home directory. If not, create one.
3. Switch into LinuxBook directory.

4. Create a new directory Lab03 inside LinuxBook.
5. Execute the following Linux commands.

```
$ date
$ echo Hello World!
$ echo $PATH
$ echo $HOME
$ echo $USER
$ echo "I'm a Linux user."
$ echo "Hello \"Hello World\""
```

6. Start WinSCP.
7. Log onto the Linux machine with WinSCP.
8. Go to folder LinuxBook, then Lab03.
9. Download hamlet.txt from the following link and save it to the desktop of your local computer and rename it as "hamlet.txt".
   https://gist.githubusercontent.com/provpup/2fc41686eab7400b796b/raw/b575bd01a58494dfddc1d6429ef0167e709abf9b/hamlet.txt
10. Transfer hamlet.txt from your desktop to LinuxBook/Lab03 folder of the Linux machine using WinSCP.
11. Under Putty, execute the following Linux commands.

```
$ cd ~/LinuxBook /Lab03
$ ls -l
$ cat hamlet.txt
$ more hamlet.txt      # press space bar to view page by page and type q to
                       # terminate
$ less hamlet.txt      # press space bar and b to view page forward and
                       # backward
$ man cat | more       # press space bar to view page by page type q to terminate
```

12. Under Putty, Execute the following Linux commands.

```
$ cd ~
$ cp ./LinuxBook/Lab03/hamlet.txt ./
$ cp ./hamlet.txt ./LinuxBook/copy.txt
$ mv ~/LinuxBook/copy.txt ./
$ mv ./copy.txt ./renamed.txt
$ ls
$ rm *.txt
$ ls -l
```

13. The last Linux command shows that the text files (.txt) are deleted from your current working directory.

14. Log out Putty and WinSCP.

Day 04: Linux Command (Part 3)

Today, let's continue to work on Linux commands.

- **head**: displays the first ten lines of a file.
- **tail:** displays the last ten lines of a file.
- **wc**: count the number of lines, words, characters, and bytes of each given file.
    - Example: wc ~/LinuxBook/lab1.txt
        - ❖ Output: 488 3932 22556
            - 488 is the number of lines.
            - 3932 is the number of words.
            - 22556 is the number of characters.
- **wc:** the options below allow you to select which counts are printed.
    - -l, Print the number of lines.
    - -w, Print the number of words.
    - -m, Print the number of characters.
- **find:** find files. The argument is a path. The name of the file must be specified with the option -name.
    - Examples:
        - find ./ -name hw1.txt *# find file with name "hw1.txt" in # current directory (including sub # directories)*
        - find / -name "*.txt" *# find file with name extension ".txt" # in root (including all sub # directories)*
- **grep**: match a regular expression against text in a file.
    - Example:
        - grep "Book Author" file1.txt *# display the lines that contain # "Book Author" in file1.txt*
- |: the pipe command allows you to pass the output of one command as the input of another command
    - Example: ls -l | more *# output of ls -l command is used as # input to more command*
- Input and output redirection.
    - Example: command < filename *# read input from a file*
    - Example: command > filename *# rewrite the file with the output of # the command*
    - Example: command >> filename *# append the output of the command # to a file*
    - Example: wc < file1.txt > file2.txt *# wc reads input from file1.txt and its # output is written to file2.txt*

Practice

1. Download the following files available online (https://github.com/linuxbookresource/files) and transfer them to your Linux home directory using WinSCP.
   - round1_p1.java
   - round1_p2.java
   - round1_p1.py
   - round1_p2.py

2. Log onto Linux machine with Putty.
3. Make sure there is a sub-directory LinuxBook in your home directory. If not, create one.
4. Switch into LinuxBook directory.
5. Create a new directory Lab04 inside LinuxBook.
6. Execute the following Linux commands.

```
$ cd Lab04
$ find ~/ -name round1_p1.py
$ cp ~/round1_p1.py ./
$ wc round1_p1.py
$ grep "Janet Yu" round1_p1.py
$ ls -l /home/ | head
$ ls -l /home/ | tail
$ ls -l /home/ > users.txt
$ wc < users.txt
$ head users.txt
$ tail users.txt
$ grep "username" users.txt          # "username" is your Linux machine user
                                     # name
```

7. The previous commands save all the usernames of the Linux machine in a text file. It then displays the line in the text file that contains your username.
8. Log out Putty.

Day 05: Linux Command (Part 4)

Pico is an easy-to-use Linux text editor. It provides a series of commands at the bottom of the screen. To start Pico, use the following command. To exit Pico, you type **Ctrl-x**.

```
$ pico file_name
```

In Linux, a process is a running program, which has an identity (PID), a parent process (PPID), an owner (UID). To view the active processes, you use command **ps**. Useful options include -A (all) and -l (long list).

To stop the execution of a program launched from a terminal you use command **Ctrl-c**. To stop a program, we can use the following command, where **pid** is the PID of a process.

```
$ kill pid
```

Example:

```
$ kill 9 1258          # a process id 1258; option 9 means terminate anyway.
```

Usually, a process is launched as a foreground task by default, and it will block the terminal until the job is done. A job could also be launched in the background using the symbol & at the end of the command line. A job can be intentionally brought to the foreground with the command fg.

Practice

1. Log onto the Linux machine with Putty. We call this Terminal 1.
2. Switch into LinuxBook directory.
3. Create a new directory Lab05 inside LinuxBook.
4. Execute the following Linux command:

   ```
   $ cd Lab05
   ```

5. Create a text file using the **Pico** editor. You can do that with the command

   ```
   $ pico lab05.txt
   ```

6. Write a few lines in this file, then save them with the command **Ctrl-o**. When the program asks you for the file name, just press enter to keep the same name. When you are done, exit the editor with the command **Ctrl-x**. Note that most useful commands for pico are displayed at the bottom of the window, where the "^" symbol means **Ctrl**.
7. Display the content of this file with the command cat.

$ cat lab05.txt

8. Launch another terminal with Putty (Log onto the same Linux machine). We call this Terminal 2.
9. From Terminal 2, switch into directory **~/LinuxBook/Lab05/**
10. In the first terminal (Terminal 1), type the command **pico**. Don't quit this application, and let it run.
11. In the second terminal (Terminal 2), find the processes running in your Linux machine with the command.

    $ ps

12. In the second terminal (Terminal 2), find out the process id of your pico process with the command.

    $ ps -Al or **$ ps -C pico**

13. Write down the pid of pico on a paper.
14. In the second terminal (Terminal 2), kill an application with the command.

    $ kill 9 pid *# pid is the process ID that you found in the previous # command*

15. Log out of Putty of both terminals.

Day 06: Linux Command (Part 5)

Today, let's continue to learn Linux commands.

- **who**: display the users currently logged onto your Linux operating system.
  - Example: who
- **ping**: check network connectivity.
  - Example: ping google.com
- **tar**: compress several files (directories) into one archive.
- **gzip**: compress/decompress a file

Suppose we have a directory (folder) called **LinuxBook** and we want to compress the folder into a single file. Use the following commands, you can then create a compressed file **book.tar.gz**.

```
$ tar –cf book.tar LinuxBook          # we get book.tar
$ gzip book.tar                       # we get book.tar.gz
```

Suppose we have a compressed file called **book.tar.gz**, use the following commands, you can decompress and extract the original files.

```
$ gzip –d book.tar.gz or $ gunzip book.tar.gz    # we get book.tar
$ tar –xf book.tar                               # we get uncompressed files
```

Practice

1. Log onto the Linux machine with Putty.
2. Make sure there is a sub-directory LinuxBook in your home directory. If not, create one.
3. Switch into LinuxBook directory.
4. Create a new directory Lab06 inside LinuxBook and switch into it.
5. On your local computer, download a compressed Linux Kernel file from: https://mirrors.edge.kernel.org/pub/linux/kernel/v1.0/linux-1.0.tar.gz
6. Use WinSCP to transfer this file to directory Lab06.
7. Under Putty, decompress the file with command:

   $ gzip -d linux-1.0.tar.gz or **$ gunzip linux-1.0.tar.gz**

8. You will get linux-1.0.tar.
9. Extract files with command:

   $ tar -xf linux-1.0.tar

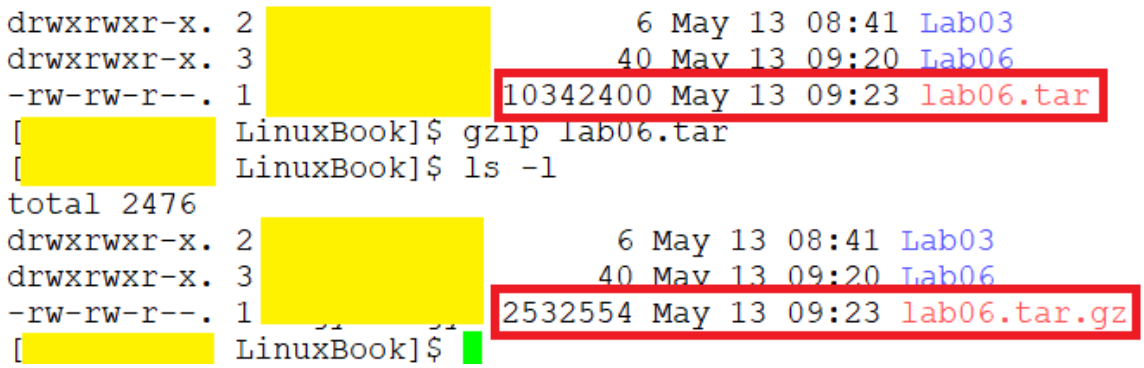
10. You will get uncompressed files (folders).
11. Execute the following commands.

    $ ls - l

$ who
$ ping yahoo.com *# let it run for a few seconds, then interrupt it with* ***Ctrl-c***

12. Execute the following commands compress Lab06 folder.
    $ cd ~/LinuxBook/
    $ ls -l
    $ tar -cf lab6.tar Lab06
    $ ls -l
    $ gzip lab6.tar
    $ ls -l

13. Compare the size of the archive before (**lab06.tar**) and after (**lab06.tar.gz**) the compression.



14. Log out Putty.

Day 07: Linux Command (Part 6)

Alpine is a Linux-based email program. To start using it, simply type the following command.

$ alpine

You can then send an email or check incoming emails.

In Linux, there are three types of file ownership:

- Owner (creator of the file)
- Group (user group)
- Others (anyone else)

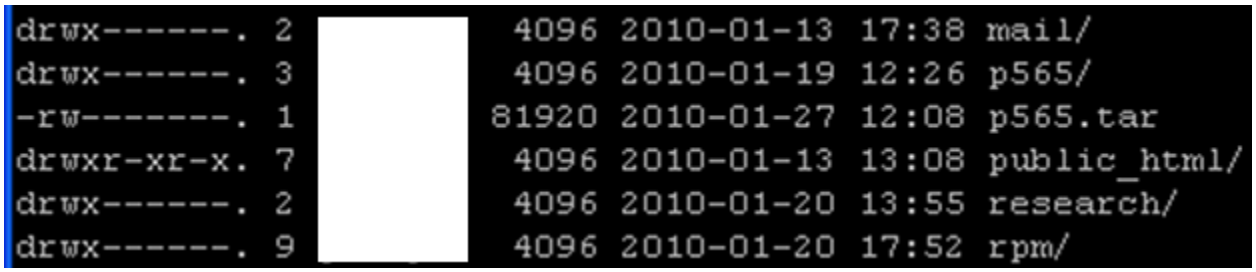
In Linux, a file has four types of permissions:

- Reading: r
- Writing: w
- Executing: x
- None: -

To check file permissions, use the following command.

$ ls -l

Look at the following example:



Column of 10 characters on left

- First character: file (-), directory (d), or link (l)
- 2nd, 3rd, and 4th characters: permissions of owner
- 5th, 6th, and 7th characters: permissions of group
- 8th, 9th, and 10th characters: permissions of all others

To change permissions, use the following command:

$ chmod mnk FileName

m, n, k are three numbers in the range of [0, 7] representing the permissions assigned to FileName.

- m: owner permission
- n: group permission
- k: anyone else permission

Permission mode values:

- Read: 4
- Write: 2
- Execute: 1
- No: 0

Permission	m/n/k
No	0
Execute	1
Write	2
Execute + Write	3
Read	4
Execute + Read	5
Write + Read	6
Execute + Write + Read	7

For example, the following command assign owner: read (4), write (2), and execute (1) permission (4+2+1=7), group: read (4) and write (2) permission (4+2=6), and anyone else: read (4) permission.

$ chmod 764 FileName

Practice

1. Log onto the Linux machine with Putty.
2. Make sure there is a sub-directory LinuxBook in your home directory. If not, create one.
3. In your home directory. Using Pico, create a file named ".signature". The "." in the file name is important and the file name should look exactly like that!

```
$ pico .signature
```

4. Include your name in the .signature file and anything else you want. Save this file and exit Pico. This file will now be used by Alpine as your signature when you send an email.
5. Switch into LinuxBook directory.
6. Create a new directory Lab07 inside LinuxBook.
7. Switch into Lab07 directory.
8. Execute the following Linux command:

```
$ date >> lab07.txt
$ who >> lab07.txt
$ chmod 777 lab07.txt
```

9. Use Alpine to send an email.

```
To: someone@someemail.com
Subject: Lab07 of LinuxBook
Message: Hello from Linux Classroom
Attachment: lab07.txt
```

10. Go to the sent mailbox.
11. Make sure the mail is sent out.
12. Terminate Alpine.
13. Log out Putty.

Day 08: Emacs and C++ Compilation

Emacs is a text editor. To start emacs, execute the following command.

```
$ emacs –nw file_name
```

To quit emacs, you use the combination of two commands: **Ctrl-x**, followed by **Ctrl-c**. To cancel a half-typed command or stop a running command: **Ctrl-g**.

There are two editing modes in Emacs: **insert** and **overwrite**. The default input mode is the insert mode. To switch to the overwrite mode, press the **[Insert]** key. To save a file, use **Ctrl-x**, followed by **Ctrl-s**.

Default Linux installation has g++ gnu compiler, which is for compiling and linking C++ programs. To compile a C++ program with just one source code file, you use the following command.

```
$ g++ source_file.cpp
```

By default, it produces an executable called **a.out**. Some options for the g++:

```
-o executable_name          # defines the name of the result
-c                          # compilation only: creates an object
```

For example:

```
$ g++ -o result source_file.cpp     # create an executable file called
                                    # result
```

To execute the result program.

```
$ ./result
```

You might need to give the “Execute” permission to the program if needed. Below is a sample C++ source code file.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string name;
    cout << "Enter your name:";
    cin >> name;
    cout << "Hello There! " << name << endl;
    return 1;
}
```

For example, the following command assigns owner: read (4), write (2), and execute (1) permissions (4+2+1=7), group: read (4) and write (2) permissions (4+2=6), and anyone else: read (4) permissions.

$ chmod 764 File_Name

Practice

1. Log onto the Linux machine with Putty.
2. Make sure there is a sub-directory LinuxBook in your home directory. If not, create one.
3. Switch into LinuxBook directory.
4. Create a new directory Lab08 inside LinuxBook.
5. Switch into Lab08 directory.
6. Use Emacs editor to create a new file Lab08.cpp by typing the command:

   $ emacs -nw Lab08.cpp

7. Type the following code in emacs editor

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string name;
```

```
        cout << "Enter your name:";
        cin >> name;
        cout << "Hello There! " << name << endl;
        return 1;
    }
```

8. Save the file: Ctrl-x then Ctrl-s.
9. Exit Emacs: Ctrl-x then Ctrl-c.
11. Compile Lab08.cpp.

    $ g++ -o hello Lab08.cpp

10. If there are errors, repeat the previous steps to fix the code until no errors are given.
12. Run program.

    $ ./hello

11. If your program can not be executed, check its permission with command: ls -l
12. Change file permission of hello if needed and run the program again.

    $ chmod 711 hello
    $ ./hello

13. Log out Putty.

Day 09: C++ Makefile

To compile a C++ program with multiple source files, it might need two steps. First, each file can be compiled separately with the option **-c** to produce an object file. Next, all the object files are linked together to create an executable.

We can simplify this process through using a **makefile**. Let's look at the following program with three source code files.

school.h

```
1   #include <string>
2   using namespace std;
3
4   class Mycourse
5   {
6   private:
7       string name;
8       int credit;
9   public:
10      Mycourse(string, int);
11      int get_credit();
12  };
```

school.cpp

```
1   #include <string>
2   #include "school.h"
3   using namespace std;
4
5   Mycourse::Mycourse(string name, int credit)
6   {
7       this->name = name;
8       this->credit = credit;
9   }
10
11  int Mycourse::get_credit()
12  {
13      return credit;
14  }
```

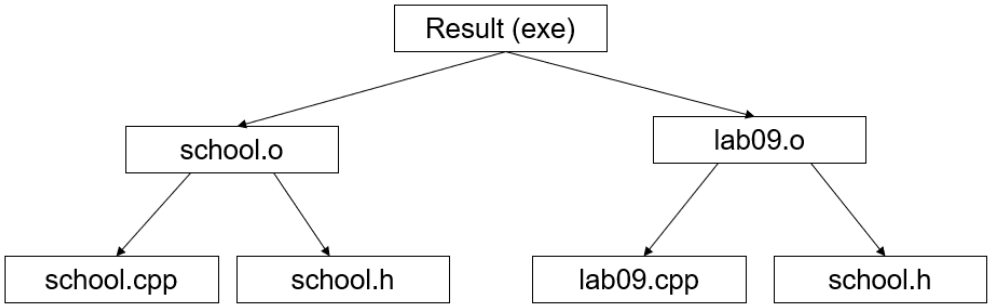
lab09.cpp

```
1   #include <iostream>
2   #include "school.h"
3   using namespace std;
4
5   int main()
6   {
7       Mycourse c1("C151", 3);
8       Mycourse c2("C201", 4);
9       Mycourse c3("C243", 4);
10
11      int total = c1.get_credit() + c2.get_credit() + c3.get_credit();
12      cout << "I am taking " << total << " credits this semester!";
13      return 1;
14  }
```

To compile this program, usually we need to execute the following commands.

```
$ g++ -c school.cpp          # create school.o
$ g++ -c lab09.cpp           # create lab09.o
$ g++ -o result school.o lab09.o   # create an executable result
$ rm –r *.o                  # remove intermediate object files
```

The chart below shows the dependencies of all the involved files.



Based on these dependencies, we can create the following Makefile and use it to compile this program.

Makefile

```
Result: lab09.o school.o
        g++ -o Result lab09.o school.o

lab09.o: lab09.cpp school.h
        g++ -c lab09.cpp

school.o: school.cpp school.h
        g++ -c school.cpp

clean:
        rm -rf *.o
```

Makefile has many components like the following.

```
target: list of dependencies
<tab> command1 to achieve target
<tab> command2
```

Before the target is built, it checks whether it is up to date by comparing time stamp of the target of each dependency; if the target file does not exist, it's automatically considered "old". If there are dependencies that are "newer" than the target, then the target is rebuilt; else untouched.

Makefile commands:

```
$ make          # create an executable (the first target)
$ make clean    # remove intermediate files
```

Practice

1. Log onto the Linux machine with Putty.
2. Switch into LinuxBook directory.
3. Create a new directory Lab09 inside LinuxBook and switch into it.
4. Download the following files given online (https://github.com/linuxbookresource/files) and transfer them to Lab09 directory.
   - school.h
   - school.cpp
   - lab09.cpp

5. Use **cat** command to look at the content of the three files.
6. Use emacs editor to create a makefile by typing the following command.

   ```
   $ emacs -nw Makefile
   ```

7. Type the following code in emacs editor (pay attention to the tab key before g++).

```
Result: lab09.o school.o
    g++ -o Result lab09.o school.o

lab09.o: lab09.cpp school.h
    g++ -c lab09.cpp

school.o: school.cpp school.h
    g++ -c school.cpp
```

```
clean:
	rm -rf *.o
```

8. Save the file: Ctrl-x, Ctrl-s
9. Exit Emacs: Ctrl-x, Ctrl-c
10. Execute the following commands. Fix errors in Makefile if there are any.

    $ make
    $ ./Result

11. Add a blank line to **school.cpp** using Emacs.
12. Use command **ls -l** to display timestamp of the last modified of all files.
13. Repeat Step 10.
14. Execute the following commands.

    $ make clean
    $ ls -l

15. Log out Putty.

Day 10: Vi and Maven

Vi is a powerful Linux text editor. Vi has two modes: insert mode and command mode. Command mode allows you to do global operations like saving the file, searching for a string and replacing it, while insert mode allows you to type in the text.

Vi starts with the command mode. To switch to insert mode, you type letter **i**, to switch back to command mode, you press the **Esc** key.

Commonly used Vi Commands

$ vi filename	*# open a file*
:w<Return>	*# save the file*
:wq<Return> or **:x<Return>**	*# save the file and exit vi*

Maven is a build tool for Java programs, especially large java projects. Building a java project usually includes downloading dependencies, compiling source code, running tests, packaging, and deploying. Maven can automate these tasks.

Practice

1. Log onto the Linux machine with Putty.
2. Make sure there is a sub-directory LinuxBook in your home directory. If not, create one.
3. Switch into LinuxBook directory.
4. Create a new directory **Lab10Part1** inside LinuxBook.
5. Switch into **Lab10Part1** directory.
6. Use vi editor to create a new file **School.java.**
   $ vi School.java

7. Its current mode (default mode) is *command mode*, let's switch to insert mode by typing letter **i**.
8. Then type the following code.

```
public class School {
    String name;
    public School(String n)
    {
        name = n;
    }
```

```
    public String getSchoolname()
    {
        return name;
    }
}
```

9. Let's switch back to command mode by pressing the **Esc** key.
10. Let's save the file and exit vi by typing

    :wq<Return>

11. Use cat command to examine the content of **School.java.**

    $ cat School.java

12. Use vi editor to create a new file **Student.java.**

    $ vi Student.java

13. Its current mode (default mode) is *command mode*, let's switch to insert mode by typing letter **i**.
14. Then type the following code.

```
public class Student {
    School s;
    String name;

    public Student(String sn, String n)
    {
        s = new School(sn);
        name = n;
    }

    public School getSchool()
    {
        return s;
    }

    public String getName()
    {
        return name;
    }
}
```

15. Let's switch back to command mode by pressing the **Esc** key.
16. Let's save the file and exit vi by typing

    :wq<Return>

17. Use cat command to examine the content of **Student.java.**

    $ cat Student.java

18. Use vi editor to create a new file **Test.java**.

    $ vi Test.java

19. Its current mode (default mode) is *command mode*, let's switch to insert mode by typing letter **i**.
20. Then type the following code.

```
public class Test {
    public static void main(String[] args)
    {
        Student s1 = new Student("Adam High School",
"John Smith");

System.out.println(s1.getSchool().getSchoolname() + "
" + s1.getName());
    }
}
```

21. Let's switch back to command mode by pressing the **Esc** key.
22. Let's save the file and exit vi by typing

    :wq<Return>

23. Use cat command to examine the content of **Test.java.**

    $ cat Test.java

24. Compile the program with command

    $ javac *.java

25. Execute the program with command

    $ java Test

26. Fix any errors if you have.
27. Switch back to **LinuxBook** directory.
28. Create a new directory **Lab10Part2** inside LinuxBook.

29. Switch into **Lab10Part2** directory.
30. Make sure Maven is installed on Linux.

    $ mvn -version

31. The command should display the Maven and Java versions installed on the system.

32. We'll use maven-archetype-quickstart plugin to create a simple java application.

    $ mvn archetype:generate -DgroupId=com.companyname.bank -DartifactId=consumerBanking -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

33. Use **ls** command, you'll see a java application project created, named consumerBanking.
34. Switch to **consumerBanking/src/main/java/com/companyname/bank** folder, delete **App.java**.
35. Copy **Test.java** you created in **Lab10Part1** to this folder and rename it as **App.java**.
36. Use vi to modify **App.java**: change class name from **Test** to **App**. Save the changes and exit vi.
37. Copy **School.java** you created in **Lab10Part1** to this folder.
38. Copy **Student.java** you created in **Lab10Part1** to this folder.
39. Save your changes.
40. Switch back to directory **~/LinuxBook/Lab10Part2/consumerbanking/**
41. Build the project with the following command.

    $ mvn clean package

42. If there are no errors, skip this step. If there are errors, use vi to modify **porm.xml** and add the following items between **url** and **dependencies**.

```
<properties>
  <maven.compiler.source>1.6</maven.compiler.source>
  <maven.compiler.target>1.6</maven.compiler.target>
 </properties>
```

43. Repeat Steps 41, 42 util program is built successfully.
44. Switch into directory **~/LinuxBook/Lab10Part2/consumerbanking/target/classes/**
45. Execute the following java command and run the program.

    $ java App

46. Log out Putty.

Day 11: HTML

Linux machines are usually installed with Apache webserver. Each user can create a folder **public_html** under their home directory to hold the webpages. We assume you have some knowledge about html and will not cover this topic here. Instead, we will focus on web hosting under Linux operating system.

Practice

1. Log onto the Linux machine with Putty.
2. Make sure there is a sub-directory LinuxBook in your home directory. If not, create one.
3. Switch into LinuxBook directory.
4. Create a new directory Lab11 inside LinuxBook.
5. Switch into Lab11 directory.
6. Download **make_page1.sh** from the supporting files (https://github.com/linuxbookresource/files) and transfer it to Lab11 folder.
7. To run this script, you need to make it executable using command

   $ chmod 700 make_page1.sh

8. Run the script.

   $ ./make_page1.sh > Lab11.html

9. You just created a html file called Lab11.html.
10. Use **cat** command to examine the content of Lab11.html.
11. Create a directory called public_html in your home directory (~).

    $ mkdir ~/public_html

12. Set the permissions (public_html) for everybody to be able to execute this directory (711).

    $ chmod 711 ~/public_html

13. If needed, make sure everybody can still execute your home (~) directory (711).

    $ chmod 711 ~

14. Create a default home page: index.html

    $ touch ~/public_html/index.html

15. Copy the html file that you have created (Lab11.html) into public_html directory.

16. Reset the permissions to all html files (index.html, Lab11.html) so that anybody can read them (644).

    $ chmod 644 ~/public_html/*

17. View your homepage, which is empty at http://your_domain_name/~your_userid/index.html.
18. View your lab 11 file online at http://your_domain_name /~your_userid/Lab11.html
19. Log out Putty.

Day 12: Shell Script (Part 1)

You can learn shell script from the following website: https://tldp.org/LDP/abs/html/.

Practice

1. Log onto the Linux machine with Putty.
2. Make sure there is a sub-directory LinuxBook in your home directory. If not, create one.
3. Switch into LinuxBook directory.
4. Create a new directory Lab12 inside LinuxBook.
5. Switch into Lab12 directory.
6. Copy **make_page1.sh** from your Lab11 directory to your current working directory (Lab12).
7. Open **make_page1.sh** using either emacs editor or vi editor and add the following functions before any of the echo commands. These functions can be used to add a list to your html page.

```
start_list()
{
  echo "<ul>"
}

end_list()
{
  echo "</ul>"
}

list_item()
{
  echo "<li>"
  echo $text
}
```

8. Add the following code to make_page1.sh (between echo "The content of your page is here." and echo "</BODY>")

```
start_list
text="cats"
list_item
text="dogs"
```

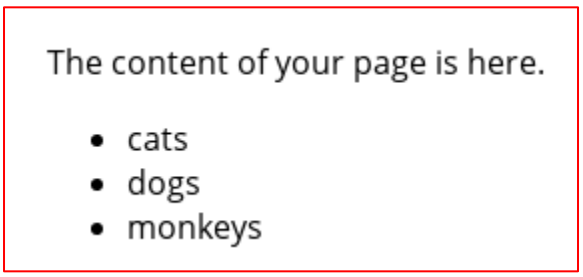
```
list_item
text="monkeys"
list_item
end_list
```

9. Run the script.
   $ ./make_page1.sh > Lab12.html

10. You will create a file Lab12.html.
11. Use **cat** command to examine the content of Lab12.html.
12. Copy the html file that you have created (Lab12.html) into **public_html** directory.
13. View your file online at the following address and make sure it is viewable by anyone.

    http://your_domain_name/~your_userid/Lab12.html

14. Your webpage should look like the following.



15. Log out Putty.

Day 13: Shell Script (Part 2)

Practice

1. Log onto the Linux machine with Putty.
2. Make sure there is a sub-directory LinuxBook in your home directory. If not, create one.
3. Switch into LinuxBook directory.
4. Create a new directory Lab13 inside LinuxBook.
5. Switch into Lab13 directory.
6. Open a new file **Lab13.sh** with emacs or vi.
   - The first line should indicate bash as the interpreter.
   - The second line display message "Hello There!"

7. Add more code to **Lab13.sh** as specified below.
   - Ask the user to enter a positive integer number.
   - Read the user input (a number).
   - Calculate the summation of numbers 1 to the number of user input (1 + 2 + 3 + ..., + user input) and display the summation. For example, if user enters 5, it displays "The summation of number 1 to 5 is 15; if the user enters 10, it displays "the summation of number 1 to 10 is 55".

8. For your reference. The solution code is given here:

```
#!/bin/bash

echo "Hello There!"

echo "Enter a number n:"
read n
sum=0

while [ $n -ge 1 ]
do
     sum=$[$sum+$n]
     n=$[$n-1]
done

echo "The sum is $sum."
```

9. Save Lab13.sh and close the editor.
10. Run the script with the command.

    $ ./Lab13.sh

11. Log out Putty.

Day 14: Shell Script (Part 3)

Practice

1. Log onto the Linux machine with Putty.
2. Download **publish.sh** given online (https://github.com/linuxbookresource/files) and transfer it to your **public_html** project folder.
3. Understand the code of **publish.sh**.
4. Download images (jpg or gif format) and transfer them to your **public_html** folder.
5. Make publish.sh executable

   $ chomd 700 publish.sh

6. Run publish.sh and you will generate one or more web pages.

   $ ./publish.sh

7. Open a web browser and access the following pages

   http://your_domain_name/~your_user_id/page1.html

8. Log out Putty.

Day 15: Computer Networks (Part 1)

You can learn basic knowledge about computer networks online. Here is a useful website: https://www.geeksforgeeks.org/computer-network-tutorials/

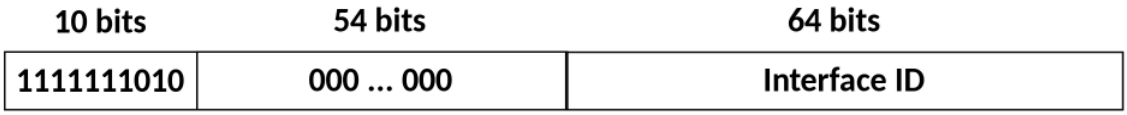
In this book, we are going to focus on using various computer network utilities/functions available in Linux.

Practice

1. Log onto the Linux machine with Putty.
2. Switch into LinuxBook directory.
3. Create a new directory Lab15 inside LinuxBook.
4. Switch into Lab15.
5. Try the following commands. Understand when and how to use them.

```
$ hostname              # display the hostname of your computer
$ ping yahoo.com        # test network connection; type Ctrl+c to terminate the
                        # process
$ wget
https://gist.githubusercontent.com/provpup/2fc41686eab7400b796b/raw/b575bd01a
58494dfddc1d6429ef0167e709abf9b/hamlet.txt    # Download a file from the Internet
$ wget -O result.txt
https://gist.githubusercontent.com/provpup/2fc41686eab7400b796b/raw/b575bd01a
58494dfddc1d6429ef0167e709abf9b/hamlet.txt    # download a file and save as a
                                              # different name
$ curl https://www.wsbt.com          # simulate a GET request for a website
$ curl -I https://www.wsbt.com       # return only the HTTP headers of a URL
$ curl -O https://www.keycdn.com/img/example.jpg # download a file
$ curl -o test.jpg https://www.keycdn.com/img/example.jpg # save as a
                                                          # different name
$ nc  198.54.116.197 21              # set up a file transfer connection with
                                     # another network device. Type "ctrl + c" to
                                     # terminate the connection
$ netstat                            # print network connections, routing tables,
                                     # etc.
$ hostname -i                        # show your IP address
$ ip addr                            # display IP Addresses and property
                                     # information
$ ifconfig -a                        # show your IP address
```

6. In the display of the last command, IPv4 address is displayed next to "**inet**". You can find **IPv6** address next to the "**inet6**" tag. For example: **fe80::7144:8b03:76d6:b3d7** means (:: means consecutive 0s)



7. Log out Putty.

Day 16: Computer Networks (Part 2)

Practice

1. Log onto the Linux machine with Putty.
2. Switch into LinuxBook directory.
3. Create a new directory Lab16 inside LinuxBook.
4. Switch into Lab16.
5. Try the following commands (Step 6 - Step 21). Understand when and how to use them.
6. Broadcast a message to all logged-in users: wall

   $ wall "Greetings from your name of C151!"

7. To broadcast a multi-line message, invoke the command without arguments.

   $ wall

8. The wall command will wait for you to enter text. When you're done typing the message, press "**Ctrl+D**" to end the program and broadcast the message.
9. Use vi or emacs to create a file "**mymessage.txt**". Type the following message inside it:

   Hello, my name is **your_name**.
   I am learning Linux now.
   Bye

10. Save and close the file.
11. Broadcasting a Message From a File. **Note: this can only be done by a root user.** Regular users can only see the file name.

    $ wall mymessage.txt

12. Display logged-on users.

    $ who

13. The output will be something like the following.

    user1 pts/1 2023-11-20 11:59 (:0)
    user2 pts/7 2023-11-20 13:09 (:0)

14. Pick one user and write a message to him/her.

    $ write user1 pts/1

15. Type any message you want to send. To exit from typing message, use "**ctrl + z**".

16. When you receive a message from another user, please identify his/her username and write back a message saying "10-4", which means “message received” in radio communications.
17. To create a simple chat with netcat, first you need to find a partner who is currently logged on to the same Linux machine to form a group of two members.
18. (Done by Group member 1. Group member 2, please ignore this step)
    a. Use putty to open another terminal of this Linux machine and run commands:
       $ ifconfig -a
    b. Let your partner (member 2) know your **ip address**.
    c. Start chat.
       $ nc -nvlp 4-digit-random-port-number (example: nc -nvlp 1234).

19. (Done by Group member 2): Connect to the chat server:
    $ nc <IP Address of group member 1> 4-digit-port-number-provided-by-member-1 (example: nc 149.161.65.101 1234)

20. A chat channel is created and both of you can type message like the following format so that you know the author of the message:
    (your name): your message

21. After you send four or more messages, you can leave the chat channel by typing "**ctrl + c**".
22. Logout of Putty.

Day 17: Computer Security (Part 1)

In this book, we are going to focus on using various computer security utilities/functions available in Linux.

Practice

1. Log onto the Linux machine with Putty.
2. Switch into LinuxBook directory.
3. Create a new directory Lab17 inside LinuxBook.
4. Switch into Lab17.
5. Create a text file, say Lab17.txt using vi or emacs. Type any message inside it. save the file and terminate the text editor.
6. Use SSH to remote logon to another Linux computer

   $ ssh your_username@address

7. Enter your password. Now you are logged onto another computer.
8. Play around with Linux commands, such as ls, pwd, etc.
9. Logout of the remote computer.

   $ exit

10. Now, you are back at the first Linux machine.
11. Use SSH to remote logon to a free public SFTP Server (server name: test.rebex.net; username: demo; password: password).

    $ ssh demo@test.rebex.net

12. Enter the password. Now you are logged onto another computer.
13. Play around with Linux commands, such as ls, pwd, etc.
14. Logout of the remote computer.

    $ exit

15. Use SFTP protocol to transfer files.

    $ sftp demo@test.rebex.net

16. Enter the password. Now you are logged onto another computer using SFTP protocol.
17. Execute the following commands.

    $ ls -l
    $ cd pub
    $ ls -l
    $ cd example

```
$ ls -l
$ get pocketftp.png
$ put Lab17.txt
$ exit
```

18. Transfer pocketftp.png to your desktop using WinSCP.
19. View the image. It looks like the following.

20. Logout Putty.

Day 18: Computer Security (Part 2)

Practice

1. Log onto the Linux machine with Putty.
2. Switch into LinuxBook directory.
3. Create a new directory Lab18 inside LinuxBook.
4. Switch into Lab18.
5. Create a text file, say **Lab18.txt** using vi or emacs. Type any message inside it. save the file and terminate the text editor.
6. Run the file through a hash algorithm MD5.

   $ md5sum Lab18.txt

7. Notice the resulting checksum (hashed) value. Let's store it into a file.

   $ md5sum Lab18.txt > hashes.txt

8. Copy Lab18.txt and create two new files.

   $ cp Lab18.txt ./Lab18copy1.txt
   $ cp Lab18.txt ./Lab18copy2.txt

9. Use any text editor to modify Lab18copy2.txt. Let's just change one character.
10. Display the hash values of copied files.

    $ md5sum Lab18copy1.txt
    $ md5sum Lab18copy2.txt

11. To make it easy to compare the values, let's store them in a file.

    $ md5sum Lab18copy1.txt >> hashes.txt
    $ md5sum Lab18copy2.txt >> hashes.txt

12. Now you can compare the checksums of three files.

    $ cat hashes.txt

13. From the hash values, you can tell which two files are same, which one is different.

```
1426fd3c567fbfba8102a6c16914c149  Lab18.txt
1426fd3c567fbfba8102a6c16914c149  Lab18copy1.txt
efea6406ab5af4d0f8137b453a4069df  Lab18copy2.txt
```

14. Log out Putty.

Day 19: Computer Security (Part 3)

Practice

1. Log onto the Linux machine with Putty.
2. Switch into LinuxBook directory.
3. Create a new directory Lab19 inside LinuxBook.
4. Switch into Lab19.
5. Download a file online.

   $ wget https://gist.githubusercontent.com/provpup/2fc41686eab7400b796b/raw/b575bd01a58494dfddc1d6429ef0167e709abf9b/hamlet.txt

6. Rename the file.

   $ mv hamlet.txt hamlet

7. Generate a key pair.

   $ gpg --gen-key

   - Then it asks what kind of keys you want. Select (4) RSA (sign only).
   - GPG prompts you for the key size. Press Enter again to accept the default value of 2,048 bits.
   - GPG asks you when the keys expire. Select the default (never expire).
   - When GPG asks whether you really want the keys to never expire, press the 'y' key to confirm.
   - GPG prompts you for your name, your email address, and a comment to make it easier to associate the key pair with your name.
   - When GPG gives you a chance to change the information or confirm it, confirm by typing the letter 'O' and pressing Enter.
   - GPG prompts you for a passphrase that protects your private key. Type a long phrase and then press Enter.

8. To let others verify your digital signature, you have to give them your public key.
9. GPG keeps the public keys in your key ring (storage). To list the keys in your key ring.

   $ gpg --list-keys

10. Export the key to a file (replace email@youremail.com with the email listed in previous command; replace yournamekey.asc with your name (one word).

    $ gpg --armor --export email@youremail.com > yournamekey.asc

Here is an example:

$ gpg --armor --export someone@mail.com > someone.asc

11. Now, share (through email or copy) this public key file (.asc) with another user.
12. When you receive a public key file from someone, import it to your key ring (storage).

$ gpg --import keyname.asc

13. Verify that the key is in your key ring.

$ gpg --list-keys

14. Sign a document with your private key.

$ gpg -o hamlet.sig -s hamlet

15. Send this signed document (hamlet.sig) to the other user that has your public key.
16. Create a new folder and switch into it.

$ mkdir receiveddoc
$ cd receiveddoc

17. After you receive a signed document from another user, transfer it to this new folder.
18. List the public key of the other user.

$ gpg --list-keys

19. Check its fingerprint (replace user@email.com with the other user's email).

$ gpg --fingerprint user@email.com

20. Verify the signature. See if the result matches the fingerprint.

$ gpg --verify hamlet.sig

21. Get back the original document.

$ gpg -o hamlet --decrypt hamlet.sig

22. List all documents in this folder.

$ ls -l

23. This is what should see.

```
-rw-rw-r--. 1 [redacted] 191726 Mar 27 16:10 hamlet
-rw-r--r--. 1 [redacted]  79433 Mar 27 16:08 hamlet.sig
```

24. Log out Putty.

Day 20: Cloud Computing

You should have an Azure account to work on this practice.

Practice

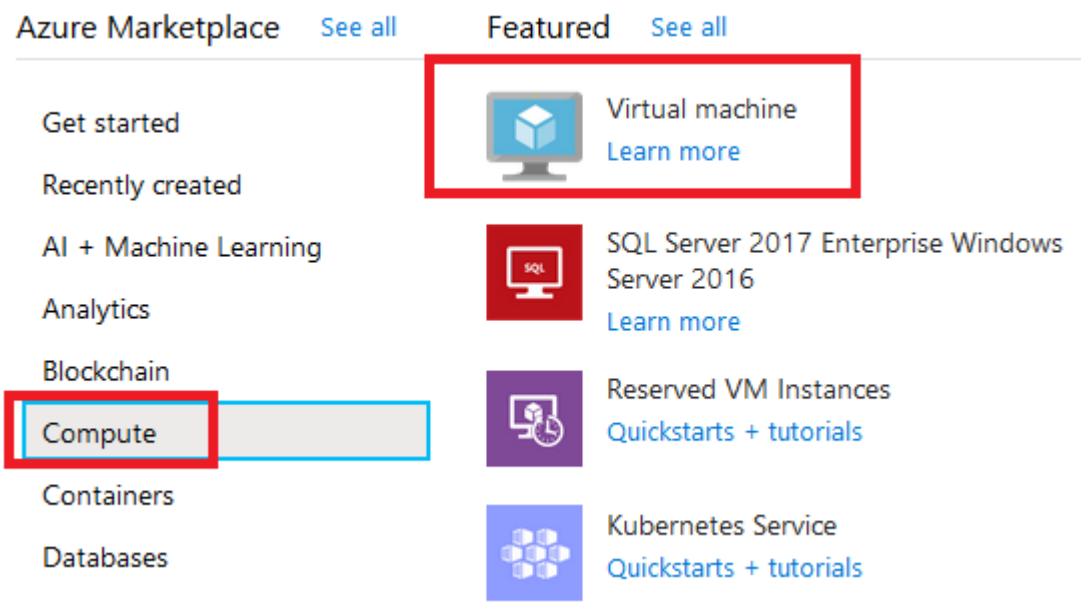
Part 1. Logon to Azure

1. Go to https://portal.azure.com/ to login.

Part 2: Linux Virtual Machine

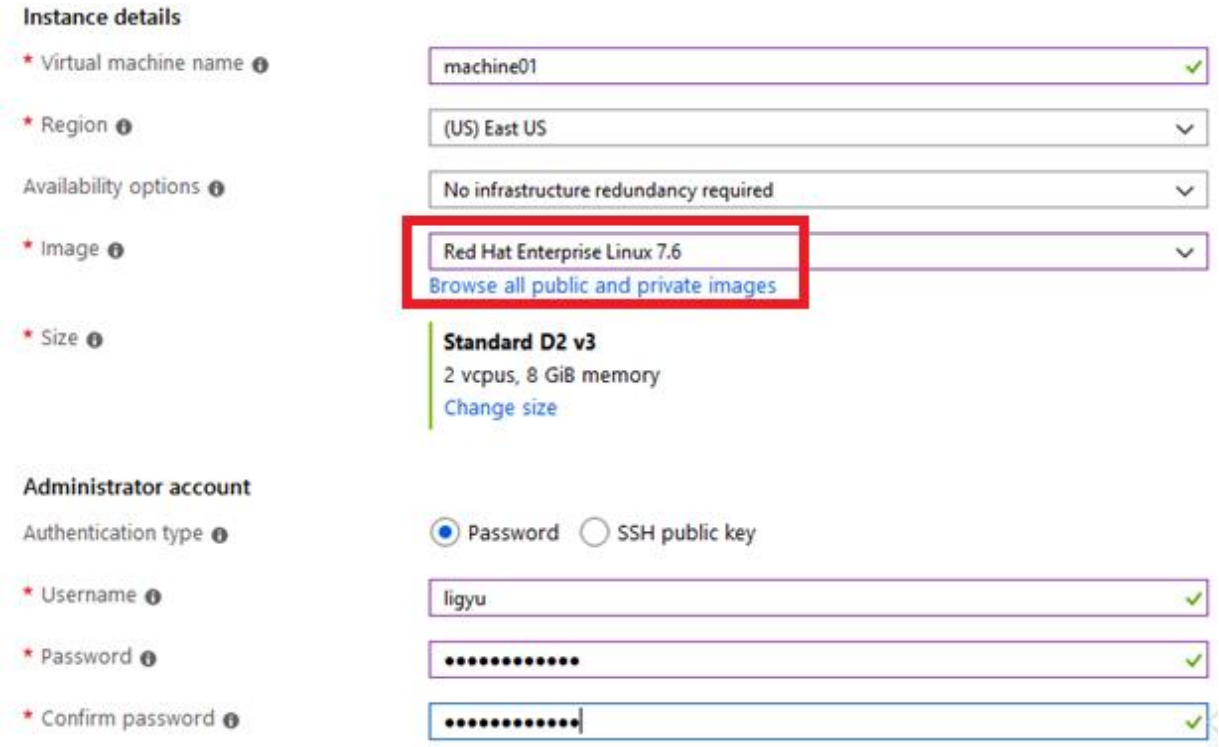
2. Click on “Create a resource”.

3. Under “Compute”, click on “Virtual machine”.

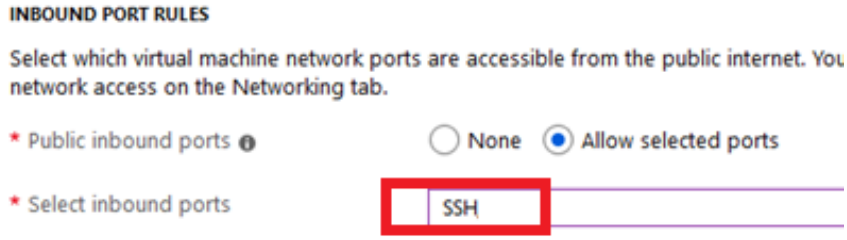


4. Create a “Resource group” and enter other information.

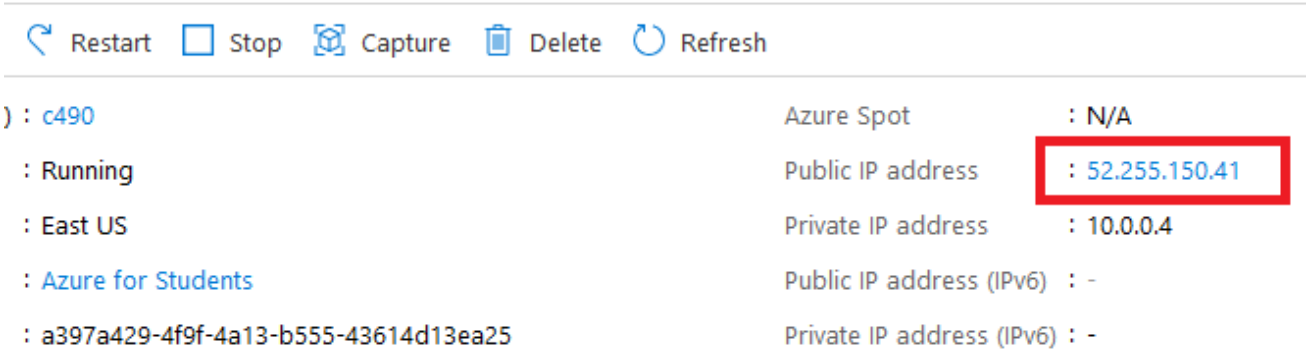
5. Make sure Image is a Linux machine.
6. For Authentication type, select Password. Choose a username and password. Write them down. You will need to use them later.



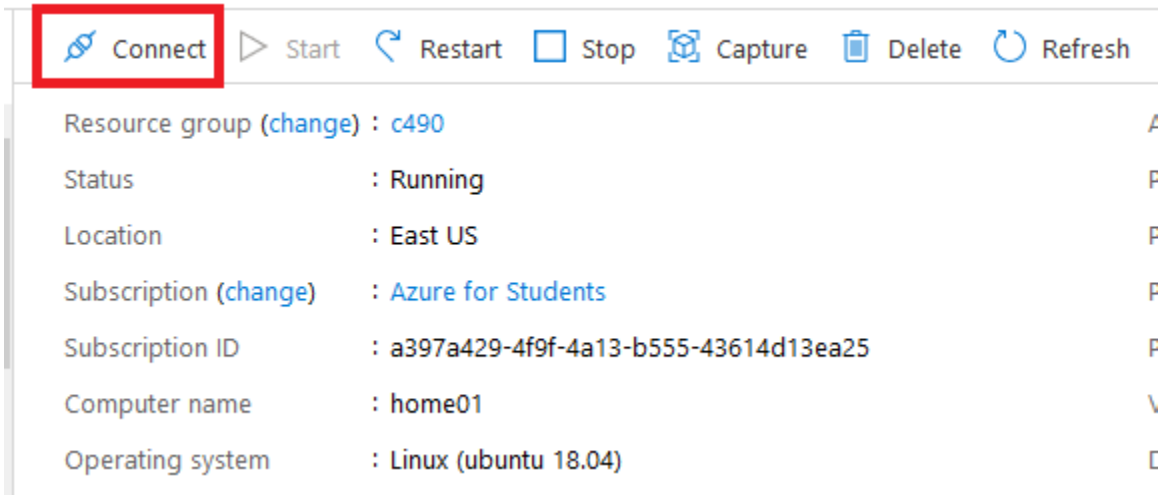
7. For INBOUND PORT RULES, please select SSH.



8. Complete the rest of selections based on your preference
9. Click on “Review + Create”, and finally, click Create. The machine will be deployed in several minutes.
10. Click on “Go to Resources” when the deployment is completed.
11. Find the IP address of this machine. Copy it.

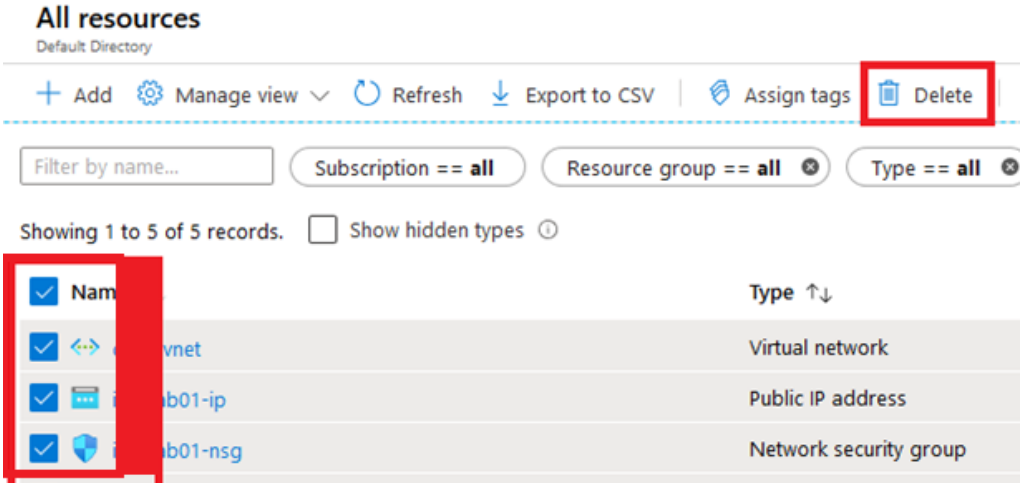


12. Click on "Connect" to see how to remotely connect to this machine.

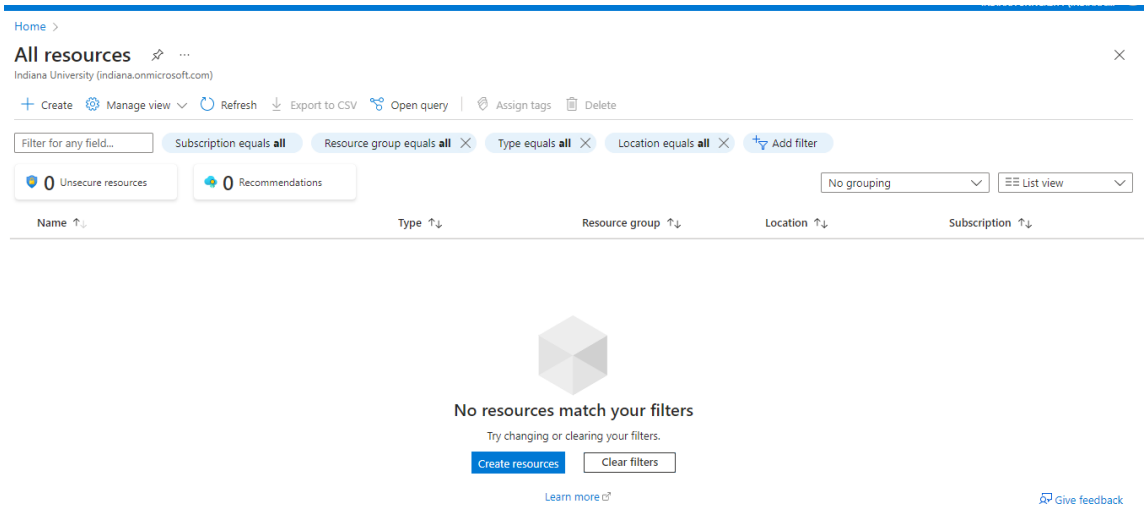


13. Use both Putty and WinSCP to remote logon to this virtual machine with your copied IP address, your username and password.
14. Play around on your virtual machine.
15. Screen shot the virtual machine (Putty and WinSCP) and save the image.
16. Logout Putty and WinSCP.
17. Go back to your Azure Portal home, click "All resources"

18. Select all the running resources. Then, click “delete”. Follow the instructions to delete these resources.



19. Wait for about 3-5 minutes to see if these resources are deleted or not. If not, repeat Step 18. It should look like the following.



20. After all running resources disappear, you are assured your account will not be changed.

Day 21: Subversion

Subversion is a version control and configuration management tool.

Practice

Note:

1. If this assignment is done by a team of two users, **User 1** needs to log onto and carry out the operations on **machine1** and **machine3**. **User 2** needs to log onto carry out the operations on **machine2**.
2. If this this assignment is done by one user, **he/she** needs to log onto and carry out the operations on **machine1, machine2**, and **machine3**.

Part 1. Create a repository and start the server

1. Log onto **machine1** with Putty.
2. Create a repository:
   $ svnadmin create LBookrep

3. Initialize an SVN server.
   $ svnserve -d

4. Create a folder **research** in your home directory. Switch into it and create a file named **myfile.txt** and add some text into it.
5. Change the permission of **myfile.txt** to **777**.
   $ chmod 777 myfile.txt

6. Switch back to your home director.
   $ cd ~/

7. Copy the file to the repository and make it under version control.
   $ svn import ~/research
   file:///home/machine1username/LBookrep/myproject -m "initial import"

8. To allow clients to access the server repository
   $ chmod -R 777 LBookrep

Part 2. Access repository by clients

9. Use Putty to log onto **machine2** as a client.

10. **Create a folder** in **LBOOK**, say **Lab21D1** and **switch into** it.
11. Checkout the repository from server (**machine1**) to local computer (**machine2**).

```
$ svn checkout file:///home/machine1username/LBookrep/myproject
```

12. Switch into the project.

```
$ cd myproject
```

13. Modify **myfile.txt**, add a new line of text. Save the changes.
14. Commit the changes.

```
$ svn commit myfile.txt -m "changed."
```

Part 3. Add more files to the repository for version control

15. Use Putty to log onto **machine3** as another client.
16. **Create a folder** in **LBOOK**, say **Lab21D2** and **switch into** it.
17. Checkout the project

```
$ svn checkout file:///home/machine1username/LBookrep/myproject
$ ls
$ cd myproject
```

18. Use cat to display the new content of **myfile.txt**.
19. Create a new file inside this folder, say **file2.txt** and type some text inside. Save the changes.
20. Add the new file to the repository

```
$ svn add file2.txt
```

21. Make the file under version control

```
$ svn commit -m "message"
```

Part 4. Coedit a file without conflict

22. On **machine2**, inside myproject directory, get the latest version from the server repository.

```
$ ls
$ svn update
$ ls
```

23. On **machine3**, inside myproject directory, make sure you have the latest version from the server.

```
$ svn update
$ ls
```

24. Now, two clients checked out the same version of **myproject**. Let's modify them concurrently.
    - Make changes to myfile.txt on **machine2**. Add a new line of text "Hello" at the beginning of the file.
    - Make changes to myfile.txt on **machine2**. Add a new line of text "BYE BYE" at the end of the file.

25. Let's try to commit these changes.

```
$ svn commit myfile.txt -m "changed."   # On machine2 (this will be OK, and
                                        # it will generate a new version.)
$ svn commit myfile.txt -m "changed."   # On machine3 (this will not be
                                        # successful, and it gives a message
                                        # "file is out of date".)
```

26. Then, **on machine3**, we need to do the following to update its local copy.

```
$ svn update
```

27. Because there is no conflict of coediting, local copy will be combined with the server copy and becomes a new version.
28. Make the local copy available to the server repository and both the local copy and server copy become a new version.

```
$svn commit -m "message"
```

29. Look at the content of myfile.txt.

```
$ cat myfile.txt
```

30. **On machine2**, to get the latest version.

```
$ svn update
$ cat myfile.txt
```

Part 5. Coedit a file with conflict

31. Now, both two clients (**machine2**, **machine3**) have the latest version of **myproject**. Let's modify them concurrently.
    - Make changes to myfile.txt on **machine2**. Add " **LINUX**" to the end of the first line. It now reads: "**Hello LINUX**".
    - Make changes to myfile.txt on **machine3**. Add " **Windows**" to the end of the first line. It now reads: "**Hello Windows**".

32. Let's try to commit these changes.

```
$ svn commit myfile.txt -m "changed."   # On machine2 (this will be OK, and
                                        # it will generate a new version.)
$ svn commit myfile.txt -m "changed."   # On machine3 (this will not be
                                        # successful, and it gives a message
                                        # "file is out of date".)
```

33. Then, **on machine3**, we need to do the following to update its local copy.

    $ svn update

34. Because there is conflict between the two coediting, it provides us the following options:

```
Conflict discovered in 'myfile.txt'.
Select: (p) postpone, (df) diff-full, (e) edit,
      (mc) mine-conflict, (tc) theirs-conflict,
      (s) show all options:
```

35. Make a selection (mc) or (tc). Let's select (**mc**). The conflict is resolved.
36. Make the resolved solution available to the server repository and both the local copy and server copy get a new version.

    $ svn commit -m "message"

37. On **machine3**, look at the content of the latest version of myfile.txt

    $ cat myfile.txt

38. On **machine2** (inside **myproject** folder), retrieve the latest version of myproject from server and look at the content of myfile.txt.

    $ svn update
    $ cat myfile.txt

39. Log out all machines.

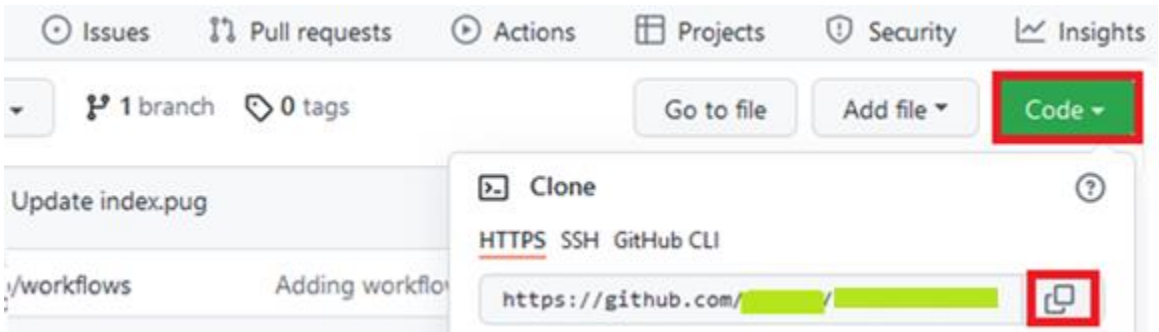
Day 22: Git and GitHub

Git is a distributed version control and configuration management tool. GitHub is an online repository supporting Git.

Practice

Part 1. Create a GitHub account and a project

1. Go to https://github.com/
2. Create a GitHub account if you do not have one.
3. Create a **public** project/repository (LB) and add a text file (lab22.txt) with any information inside.
4. In your GitHub account, go to the project folder. Click "**Code**" and copy the **Clone URL**.



Part 2. Use Git from Linux

5. Use Putty to log onto the Linux machine.
6. Create a new folder Lab22 under LinuxBook and switch into it.
7. Set your account's default identity by running the following git commands (for first time user only). Replace **blue text** with your own information.

```
$ git config --global user.email you@youremail.com
$ git config --global user.name your_github_username
$ git config -l
```

8. To clone the GitHub repository, execute the following command (replace **url** with the one you copied at Step 4):

```
$ git clone url
```

9. The project (**LB**) on GitHub is copied to your local repository.
10. Switch into the local repository.

    $ cd LB

11. Open File: **Lab22.txt**.
12. Add a new line of text to the end of the file.
13. Save the changes.
14. Execute the following commands.

$ git add -A	(select the files for commit of changes)
$ git commit -m "change a file"	(the changes are made to the repository)
$ git push	(push the local changes to GitHub)

15. Enter username and password if asked.
16. If authentication fails, please do Steps 17-19. Otherwise go to Step 20.
17. Create Access Token on GitHub.
    - Click on your **GitHub profile icon** in the top right corner.
    - Click **Settings.**
    - From the menu shown on the left, click **Developer Settings.**
    - Click **Personal access tokens.**
    - Click **Generate new token.**
    - Add a note that will help you identify the scope of the access token to be generated.
    - **Choose the Expiration period** from the drop-down menu.
    - Finally, **select the scopes you want to grant the corresponding access to the generated access token**. I will check **all of them**.
    - Finally click **Generate Token.**
    - Copy the token and save it on your desktop computer.
18. On Linux, execute the following command to reset authentication method.

    $ git remote set-url origin
    https://<token>@github.com/<user>/<repository>.git

    In the above command make sure to replace:
    - **<token>** with the personal access token you have copied in the previous step
    - **<user>** with your GitHub username
    - **<repository>** with the name of your GitHub repository (**LB**)
19. Execute the following command to push the change again.
    $ git push

20. In your web browser, go to GitHub and check file **Lab22.txt** to see if it is updated.

Part 3. Coediting under Git

21. Log onto GitHub and create a new public project/repository, say Lab22P3.
22. Add a file readme.txt (with any info) to this project.
23. Copy this project's URL.
24. Use Putty to log onto the Linux machine and switch to **LinuxBook** directory.
25. Configure your Git client (replace bule text with your own information).

```
$ git config --global user.email you@youremail.com
$ git config --global user.name your_github_username
$ git config -l
```

26. Clone your GitHub repository to your local computer. Replace copied_url with the one you got at Step 23.

```
$ git clone copied_url
```

27. The project/repository is cloned locally. Switch into it.

```
$ cd Lab22P3
```

28. Create a new file say **L22P3.txt** and add several lines of messages and save it.
29. Commit the changes (make the change available to the repository for version control)

```
$ git add L22P3.txt
$ git commit -m "some message"
```

30. Push the changes to GitHub.

```
$ git push
#enter username if asked
#enter password if asked
```

31. If authentication fails. Please follow the instructions on Lab 22 (Step 17) to reset authentication with Access Token.
32. Push the change to GitHub again if Step 31 failed.

```
$ git push
```

33. Check the changes on Git Hub.
34. Make change to **L22P3.txt** on GitHub (for example, add a new line of text). Remember to commit the changes**.**
35. Check out the latest version of your GitHub repository to your Linux computer.

```
$ git pull
```

36. Examine the changes locally.
    $ cat L22P3.txt

37. Now, let's create a case of co-editing without conflict.
    37.a. On **GitHub**, add a new line of text "HELLO THERE" **to the beginning** of L22P3.txt. Remember to commit the change.
    37.b. On your Linux machine, modify L22P3.txt and add a new line of text "BYE BYE" **to the end** of L22P3.txt.

38. Commit the local change on Linux (make the changed file available to the repository for version control).
    $ git add L22P3.txt
    $ git commit -m "some message"

39. Push the local change on Linux onto GitHub .
    $ git push

40. You will see a message saying the version of document you checked out and modified has been updated. You have to merge the changes of co-editing.
41. Let's solve the problem: pull the remote file (on GitHub) to local (Linux) to merge two changes.
    $ git pull

42. Git found no conflicts of co-editing. The changes will be merged. You need to write a commit message for this merge operation (next step).
43. Vi editor is open automatically. Type a message and exit vi. (i: insert mode; Esc: command mode; :x<return>: exit)
44. Push the new version (merged version) on Linux onto GitHub.
    $ git push

45. Check L22P3.txt on GitHub to see if it is updated. You can also check the content of the latest version of L22P3.txt on Linux using cat command.
46. Now, let's create a case of conflict in co-editing.
    46.a. On **GitHub**, add text " **ENGINEER**" **to the end of first line** of L22P3.txt. So, the first line becomes "**HELLO THERE ENGINEER**". Commit the change.
    46.b. On **your Linux machine**, open L22P3.txt and add text " **PROGRAMMER**" **to the end of first line**. So, the first line becomes "**HELLO THERE PROGRAMMER**".

47. Now, try to commit the Linux local change and push it to GitHub.
    $ git add L22P3.txt
    $ git commit -m "some message"
    $ git push

48. You will see some messages saying your local changes cannot be pushed to the remote repository.
49. Let's solve the problem.
    $ git pull # pull the remote file locally to resolve conflicts

50. Because the two changes have conflicts, they cannot be merged automatically. We must resolve the conflicts manually (next step).
51. Open L22P3.txt with a text editor (Pico, Emacs, Vi). Make your decision, suppose you delete the online change and keep your local change.
52. Save and close L22P3.txt.
53. Commit the change and push it to GitHub.
    $ git add L22P3.txt
    $ git commit -m "some message"
    $ git push

54. Check L22P3.txt on GitHub to see if it is updated. You can also check the content of the latest version of L22P3.txt on Linux using cat command.
55. Log out all machines.

Day 23: Python

We assume you know the basics of Python programming. Let's work on some examples under Linux.

Practice

1. Log onto the Linux machine with Putty.
2. Switch into LinuxBook directory.
3. Create a new directory Lab23 inside LinuxBook.
4. Switch to Lab23.
5. Download **lab23.py** given in the supporting files (https://github.com/linuxbookresource/files) and transfer it to your **Lab23** folder.

Part 1. Interactive Python

6. Start Python

```
$ python
```

7. Execute the following statements (one line at a time)

```
>>> data = range(0, 100, 5)
>>> print data
>>> for x in data:
...     print x
...
>>> input_file = open("lab23.py")
>>> output_file = open("lab23copy.py", "w")
>>> for line in input_file.readlines():
...     output_file.write(line)
...
>>> exit()
```

8. Use cat command to look at the content of "**lab23copy.py**" and "**lab23.py**".

Part 2. Modular Python

9. Use your favorite text editor to open **lab23.py**.
10. Complete Function isleapYear(n). Here is the description of the function.

• Leap Years are any year that can be exactly divided by 4 (such as 2012, 2016)
• except if it can be exactly divided by 100, then it isn't (such as 2100, 2200)
• except if it can be exactly divided by 400, then it is (such as 2000, 2400)

11. This function takes a year (n) as input and returns true or false indicating whether the input is a leap year.
12. For your reference, the code is given here.

```
def isLeapYear(n):
    if n % 4 != 0:
        return False
    elif n % 100 != 0:
        return True
    elif n % 400 == 0:
        return True
    else:
        return False
```

13. Execute the program:
    $ python lab23.py

14. Sample input and output of running the program.
    Input: 1999
    Output: no

    Input: 2012
    Output: yes

    Input: 2100
    Output: no

    Input: 2400
    Output: yes

15. Log out Putty.

Day 24: Perl (Part 1)

Perl language reference is available online, such as https://en.wikipedia.org/wiki/Perl and http://www.troubleshooters.com/codecorn/littperl/perlreg.htm
In this book, we are going to focus on running Perl programs under Linux.

Practice

1. Log onto the Linux machine with Putty and switch to LinuxBook directory.
2. Create a new directory Lab24 inside LinuxBook and switch into it.
3. Download **Lab24.pl** and **numbers.txt** given in the supporting files (https://github.com/linuxbookresource/files) and transfer them to your **Lab24** folder.
4. Use cat command to examine the content of **numbers.txt**.
5. Use your favorite text editor to open **Lab24.pl** and understand the structure of the program.
6. Make Lab24.pl executable, run it, and see the output.

   $ chmod 711 Lab24.pl
   $ ./Lab24.pl

7. Add code to **Lab24.pl** to complete the subroutine **min**.
8. Run the program again until it works correctly. The correct program should display:

   10 5.6 209 -98 67 3.1415926 -7.8 3 78 -90 34.55 8

   The summation of these numbers is 222.4915926
   The smallest number is -9

9. For your reference, the solution code is given here.

```
sub min
{
      my @numbers = @_;
      my $smallest =$numbers[0];

      foreach $n (@numbers)
      {
            if($smallest > $n)
            {
                  $smallest = $n;
            }
      }
```

```
    return $smallest;
}
```

10. Log out Putty.

Day 25: Perl (Part 2)

Today, let's continue to learn Perl language.

Practice

1. Log onto the Linux machine with Putty.
2. Switch to LinuxBook directory.
3. Create a new directory Lab25 inside LinuxBook.
4. Switch to Lab25.
5. Download **Lab25.pl** given in the supporting files (https://github.com/linuxbookresource/files) and transfer it to your **Lab25** folder.
6. Download **numbers.txt** given in the supporting file (https://github.com/linuxbookresource/files) and transfer it to your **Lab25** folder.
7. Use cat command to examine the content of **numbers.txt**.
8. Use your favorite text editor to open **Lab25.pl** and understand the structure of the program.
9. Make Lab25.pl executable, run it, and see the result.

   $ chmod 711 Lab25.pl
   $ ./Lab25.pl

10. Add code to **Lab25.pl** to split lines into numbers and merge them into one array.
11. Run the program again until it works correctly.
12. The correct program should display:

```
10 5.6 209
-98 67 3.1415926
-7.8 88 3 78 -200 -90
34.55 8

The summation of these numbers is 110.4915926
The smallest number is -200
```

13. For your reference, the solution code is given here.

```
foreach $line (@input_lines)
{
     @new_numbers = split(/\s/, $line);
     @all_numbers = (@all_numbers, @new_numbers);
}
```

14. Log out Putty.

Day 26: Perl (Part 3)

Today, let's continue to learn language Perl.

Practice

1. Log onto the Linux machine with Putty.
2. Switch to LinuxBook directory.
3. Create a new directory **Lab26** inside LinuxBook and switch into it.
4. Download **Lab26.pl** given in the supporting files (https://github.com/linuxbookresource/files) and transfer it to your **Lab26** folder.
5. Use your favorite text editor to open **Lab26.pl** and understand the structure of the program.
6. Make **Lab26.pl** executable.
7. Add code to **Lab26.pl** to analyze the script "hamlet".
8. Run the program again until it works correctly.
9. Your output of running the program (**hamlet_analysis.txt**) should look like the following screenshot.

```
The number of words is 19715
The number of unique words is 2976
The occurrence of each words is listed below
****************************************
      1
0     113
003399      2
0px   4
1     34
10    4
100   5
102   1
103   2
105   3
106   2
107   1
109   2
11    70
110   5
111   1
113   1
115   1
12    7
```

10. For your reference, the solution code is given here.

```
# save the content to file hamlet.html
open(output1,">hamlet.html");
print output1 $whole_script;
close output1;

$whole_script =~ tr/A-Z/a-z/;

# split the whole scripts into words
 @words = split(/\W+/, $whole_script);

# count the words, %wordcount is a hashtable
foreach $word (@words)
{
   $wordcount{$word} = $wordcount{$word} + 1;
}

# @unique_word is an array that contains all the unique words
@unique_word = keys %wordcount;

$num_words = $#words + 1;
$uni_words = $#unique_word + 1;

open(output2,">hamlet_analysis.txt");

# print out the analysis to file
print output2 "The number of words is $num_words\n";
print output2 "The number of unique words is $uni_words\n";
print output2 "The occurrence of each words is listed
below\n";
print output2 "****************************************\n";

foreach $word (sort (keys %wordcount))
{
   print output2 "$word\t$wordcount{$word}\n";
}

print output2 "\n\nThe occurrence of each words is listed
again in increasing order counts\n";
print output2 "****************************************\n";

foreach $word (sort{$wordcount{$a}<=>$wordcount{$b}} (keys
%wordcount))
{
   print output2 "$word\t$wordcount{$word}\n";
```

```
}

close output2;
```

11. Log out Putty.

Day 27: Latex (Part 1)

You can learn Latex through
https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes.
In this book, we are going to focus on how to compile Latex source code and create a pdf file in Linux.

Practice

1. Log onto the Linux machine with Putty.
2. Switch into LinuxBook directory.
3. Create a new directory Lab27 inside LinuxBook and switch into it.
4. Download **lab27.tex** given online (https://github.com/linuxbookresource/files) and transfer it to your **Lab27** folder.
5. Download **Figure1.JPG** given online (https://github.com/linuxbookresource/files) and transfer it to your **Lab27** folder.
6. Use your favorite text editor to edit **lab27.tex**:
   a. Change "Your Name" to **your name**
   b. Change "Figure Name" to **Figure1.JPG**.
7. Run the following command.
   $ pdflatex lab27

8. You will generate a file called **lab27.pdf**. Transfer it to your desktop and view it. It should look like the following.

My Study of Linux Kernel

[redacted]

April 24, 2023

1 Introduction

To be done!

2 Result

Result is shown in Figure 1.

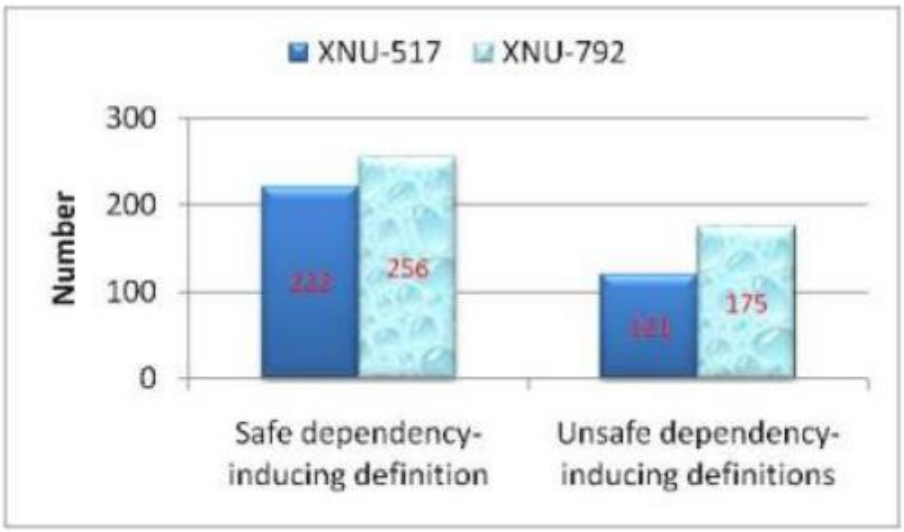


Figure 1: Depiction of the production of kernel-based software

3 Conclusion

Latex is not user friendly.

9. Log out Putty.

Day 28: Latex (Part 2)

Let's continue to learn how to use Latex under Linux.

Practice

1. Start Putty.
2. Log onto the Linux machine with Putty.
3. Switch to LinuxBook directory.
4. Create a new directory Lab28 inside LinuxBook.
5. Switch to Lab28.
6. Download **lab28.tex** given online (https://github.com/linuxbookresource/files) and transfer it to your **Lab28** folder.
7. Download **lab28.bib** given online (https://github.com/linuxbookresource/files) and transfer it to your **Lab28** folder.
8. Examine **lab28.tex** and **lab28.bib** with cat command or a text editor. Understand the structure of each reference item in .bib file. Please also understand how to cite a reference item in .tex file.
9. Repeat the following commands 3 rounds:
   $ latex lab28
   $ bibtex lab28
   $ pdflatex lab28

10. Transfer the output file **lab28.pdf** to your desktop and view it. It should look like the following file.
    https://raw.githubusercontent.com/linuxbookresource/files/main/Day28/lab28.pdf

11. Log out Putty.

Online Resources

Linux Command: https://ubuntu.com/tutorials/command-line-for-beginners#1-overview

Linux Networking: https://www.freecodecamp.org/news/linux-networking-commands-for-beginners/

Linux Security: https://www.linuxtopia.org/LinuxSecurity/

Makefile: https://makefiletutorial.com/

Maven: https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

HTML: https://html.com/

Shell Scripting: https://www.shellscript.sh/

Subversion: https://www.tutorialspoint.com/svn/index.htm

Git: https://www.atlassian.com/git/tutorials

Python: https://docs.python.org/3/tutorial/index.html

Perl: https://www.perltutorial.org/

LaTex: https://latex-tutorial.com/