Working in Different Environments:
An Introduction to Linux
IT 179

Although Windows is a popular operating system, it is only one of several operating systems in common use today. The purpose of this exercise is to give you a taste of what it's like to work with Java in a different operating system, so that you can get a feel for other environments and see what does and doesn't change.

This laboratory exercise is a hands-on activity with several parts. Be sure to do everything, in order, and read all of the instructions carefully. In this lab, you'll learn to

- Log on to the School of IT Linux terminal servers
- Navigate the Linux file system
- Edit files in Linux with VS Code
- Compile and run Java programs from the command line
- Use a few built-in Linux commands
- Submit your programs

You will be submitting this for 30 points of quiz/homework credit. Make sure you answer all of the questions.

A. Logging on to the School of IT Linux machines

Access can be done in three different ways. In every case, you must log in using your ULID and your usual password for your university account.

The first two options will give you a full virtual desktop with windows, menus, etc. One option is to use any browser to go to login.it.ilstu.edu and click on the middle button (the one that says Terminal Servers). Then you will need to log in using your ULID and corresponding password. In the resulting page, expand Linux Terminal Servers under ALL CONNECTIONS, and select one of the choices there. You'll then be presented with a desktop in the browser. This option works pretty well and does not require a VPN connection when off campus. However, it is sometimes a little sluggish compared to the second option when on campus.

The second option is to use Remote Desktop on campus. You can simply put in the machine name (such as ash) for connecting to a PC. You will have to accept an issue with authentication. This method requires a VPN connection if you are off campus (see How to Use Cisco AnyConnect VPN Client | IT Help - Illinois State). I recommend sticking with the login.it.ilstu.edu method when off campus, since it tends to be less sluggish than a remote desktop connection through the VPN in my experience.

You can also connect using ssh with a program such as putty, but that will only give you a basic terminal window for command line work.

For this lab, I recommend using login.it.ilstu.edu.

Log in to one of the Linux terminal servers (ash, aspen, or bur) through login.it.ilstu.edu. Note that it does not matter which – your data will be available from any of the machines.

Once you have logged on to the system, you need to get a working environment. You'll see that you have a windowed environment that works a little like Windows. You may choose to explore this environment later, but for this lab, we're going to focus on the command-line environments.

In order to get a terminal window in which to work, you will use the Terminal Emulator. You have two choices for how to get this. You can use the Application menu in the upper left of your window, or you can click on the Terminal Emulator icon in the toolbar at the bottom of your window in the middle. Once you have a terminal window open, click on it (to make sure it is selected).
.

B. Files and Directories

pwd print working directory

In Windows, we use folders to organize our files. In Linux, we use the same idea, but we don't usually use the term *folder*; instead we call them *directories* (which is also an acceptable term to use for Windows folders). While you're working in Linux, you're always located in a particular directory: your *working directory*. To see what directory you're in, you can use the Linux command *pwd* (print working directory).

Use pwd to see what your working directory is when you first get on the system. What is the result?

/home/ad.ilstu.edu/fpayan

This is the complete path name for your home directory. This is the space set aside for you on the Linux machines. Notice that where Windows has a backslash between folder names in a path, Linux uses slashes.

cd change directory

Now that we know where we are, let's try going somewhere else. Moving around the Linux file system is done using the *cd* command (change directory). Let's do a little exploring.

Type *cd /* at the prompt. Note: there is a space between cd and /

You are now in the *root* directory—the top of the Linux file system hierarchy. Linux hides a lot of the differences between different physical disk drives from you, and has everything in the file system in one hierarchy, unlike Windows, where each disk drive has its own hierarchy.

ls list

Suppose we want to see what is in this directory. To do that, we can use the *ls* command. (ls is short for list (as in list files)).

What is the result of typing *ls* and then Enter? (list just the first 5 filenames you see)

bin elkshare initrd.img.old lost+found proc snap

This is the short form of the *ls* command. Let's add a parameter to the command.

Type *ls –l* and press Enter. How does the result differ?

I get nine columns of information. The first column mainly consists of: "drwr -xr -x". The second through fourth columns seems to get information regarding the root and the fifth through ninth columns gives a date and time along with a filename. I see a total counter with the value of 12583100.

The first column indicates whether a file is a directory, a link (sort of like a shortcut in Windows), or an ordinary file. The next 9 columns indicate whether the file (or directory) is readable, writable, and executable by the owner, the owner's group (other users of the same type), and the world (everyone who can log on to the system). The number that comes next is the number of links to the file (not usually terribly interesting to you). Then come the username and group of the file's owner, the size of the file in bytes, the date the file was last modified, and the name of the file.

Let's move to another directory, with a plain file in it to look at.

Type *cd etc* and then Enter. Use the pwd and ls commands to determine where you are and what files are present (and write the directory name and the first 5 filenames you see below).

I am in the directory: /etc

The first five filenames I see are:

acpi hdparm.conf pm
adduser.conf host.conf

Now let's look at a file. One way to do that is to type *cat filename*. This command simply writes the contents of the file to the window with no regard to how much you can actually see of the file.

Try using the *cat* command on the file named *hosts* in the current directory. This is a very short list of different network hosts that the machine knows about.

Now that you've had a chance to explore a bit, let's go back to your home directory. This can be easily done by using the *cd* command with no arguments.

Use *cd* without arguments to go back to your home directory, and then use *pwd* to confirm that you are there.

---

mkdir make directory

Now that you've learned a bit about how to get around and explore, let's work on setting up a space in which to work. Your home directory is your space, a little like a network drive such as I: drive you might get for an IT class; but you probably don't want to do everything at the top level of your home directory, so let's make a directory for you to use for this class, named it179. To do this, use the mkdir (for make directory) command.

Type *mkdir it179*. Use ls to determine the contents of your home directory.

---

rmdir remove directory

Suppose you make a mistake and want to get rid of a directory. The command to do that is *rmdir* (for remove directory). rmdir doesn't give you a chance to change your mind if you type it by accident, but it will not remove non-empty directories.

Use cd to move into your it179 directory.

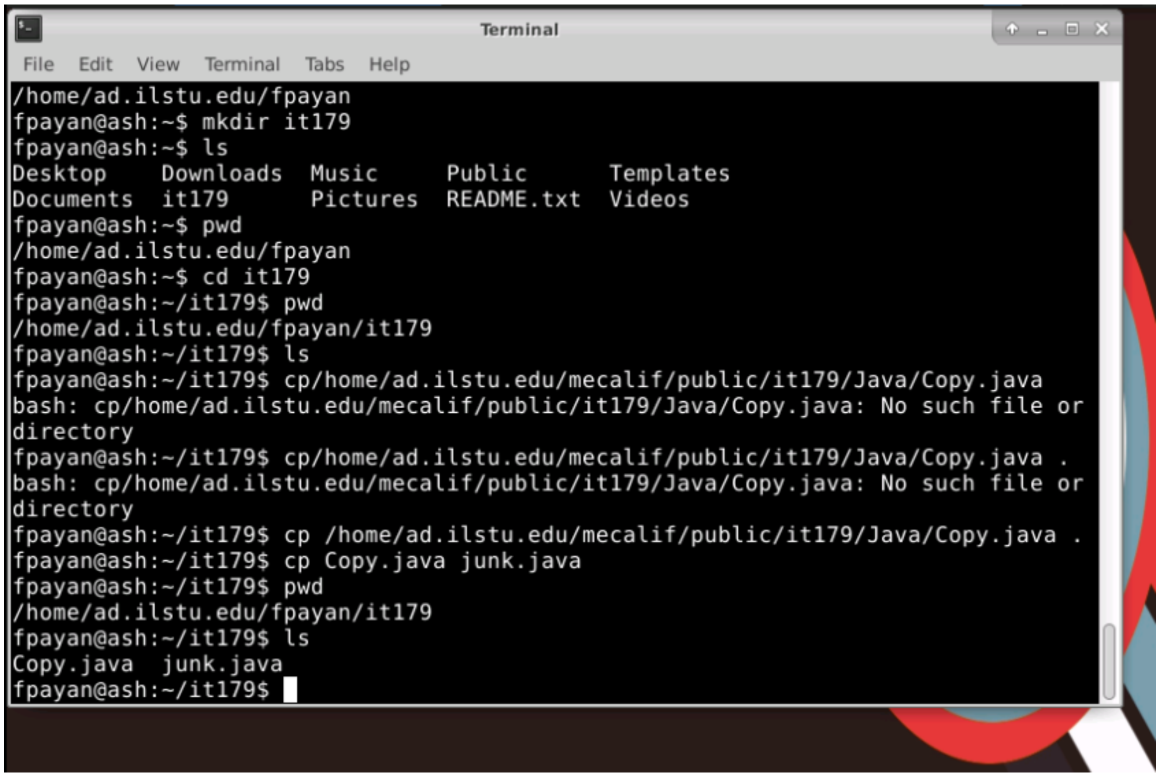
---

cp copy

Now we need to get copies of some example files to work with. We'll do that using the *cp* (copy) command. All of the files to be copied will come from the directory /home/ad.ilstu.edu/mecalif/public/it179.

Handy note: at a prompt, hitting the up arrow causes whatever you entered at the last prompt to be entered automatically.
First, use
cp /home/ad.ilstu.edu/mecalif/public/it179/Java/Copy.java .
to copy the file *Copy.java* to the current working directory(itk179). Note: There is a space between .java and the dot at the end .
The . at the end simply stands for the current directory. Linux also uses .. to refer to the parent directory.

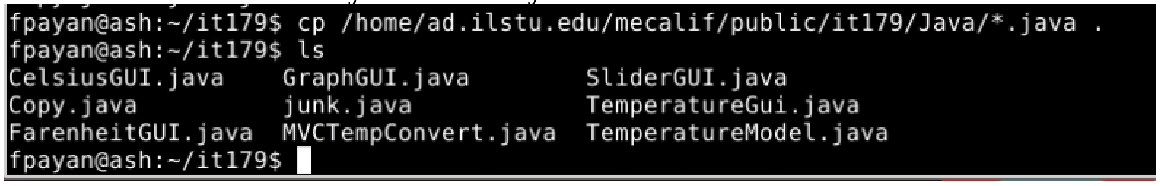
In copying a file, you can also change the name of the file. Try the following:
cp Copy.java junk.java
Confirm that both files are in your directory and have the same contents.



```
/home/ad.ilstu.edu/fpayan
fpayan@ash:~$ mkdir it179
fpayan@ash:~$ ls
Desktop    Downloads  Music     Public      Templates
Documents  it179      Pictures  README.txt  Videos
fpayan@ash:~$ pwd
/home/ad.ilstu.edu/fpayan
fpayan@ash:~$ cd it179
fpayan@ash:~/it179$ pwd
/home/ad.ilstu.edu/fpayan/it179
fpayan@ash:~/it179$ ls
fpayan@ash:~/it179$ cp/home/ad.ilstu.edu/mecalif/public/it179/Java/Copy.java
bash: cp/home/ad.ilstu.edu/mecalif/public/it179/Java/Copy.java: No such file or
directory
fpayan@ash:~/it179$ cp/home/ad.ilstu.edu/mecalif/public/it179/Java/Copy.java .
bash: cp/home/ad.ilstu.edu/mecalif/public/it179/Java/Copy.java: No such file or
directory
fpayan@ash:~/it179$ cp /home/ad.ilstu.edu/mecalif/public/it179/Java/Copy.java .
fpayan@ash:~/it179$ cp Copy.java junk.java
fpayan@ash:~/it179$ pwd
/home/ad.ilstu.edu/fpayan/it179
fpayan@ash:~/it179$ ls
Copy.java  junk.java
fpayan@ash:~/it179$ 
```

When copying files, we often want to copy several files at a time. Linux provides a nice way for us to do this using a wildcard character (*).

cp /home/ad.ilstu.edu/mecalif/public/it179/Java/*.java   .
What are the contents of your directory now?



```
fpayan@ash:~/it179$ cp /home/ad.ilstu.edu/mecalif/public/it179/Java/*.java .
fpayan@ash:~/it179$ ls
CelsiusGUI.java    GraphGUI.java       SliderGUI.java
Copy.java          junk.java           TemperatureGui.java
FarenheitGUI.java  MVCTempConvert.java  TemperatureModel.java
fpayan@ash:~/it179$ 
```

The command should have copied all eight files ending in *.java* from the original directory to yours.

What command could you use to copy all files from the /home/mecalif/public/itk169/Java directory to the current working directory?

cp /home/ad.ilstu.edu/mecalif/public/it179/Java/* .

Use it to copy all of those files across now.

We need to learn two other commands for working with files: a way to move files without having to copy them, and a way to get rid of files we no longer need.

To move a file without copying it, you use the command *mv* (for move). The command has two forms:
mv fileName newFileName
mv fileName(s) directoryName
You can use this command to move a file from one place to another or to change the name of a file. Note that you must *mv* a file in order to change its name.

Try changing the name of the file sample.txt to myfile.txt

To delete a file, you use the *rm* command (for remove). Note that you can remove more than one file at a time if you choose. On our system, the *rm* command automatically requests confirmation. You'll need to answer y for each file you actually want to delete. Any other response will keep the file from being deleted.

Use the *rm* command to delete the file junk.java.

C. Editing Files

There are several choices for editing on the Linux terminal servers, but I recommend that you use VS Code.

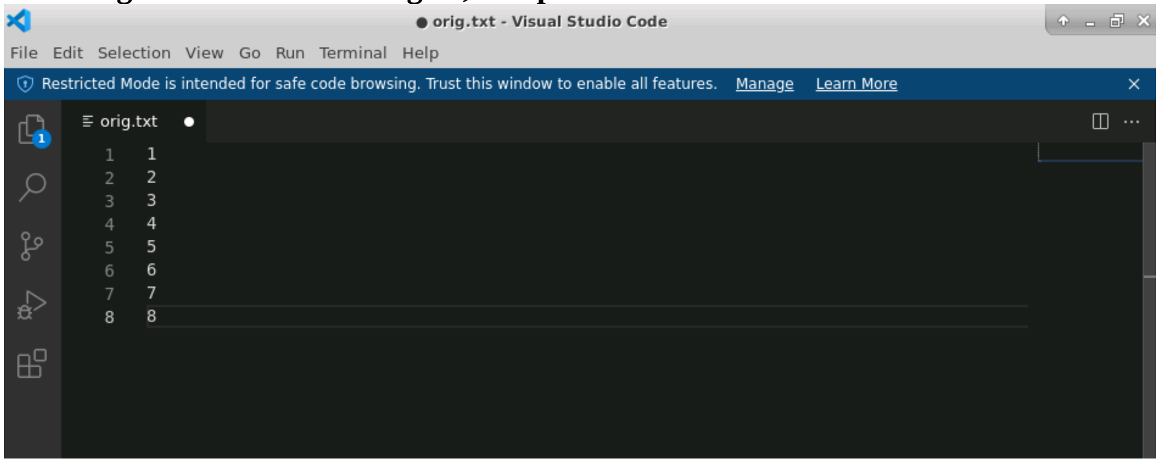
You can run VS Code from the Applications menu, but we're going to do it from the command line.

First, let's use it to create a file like this:

At the command line, type
code orig.txt

This will bring up a VS Code window with an file orig.txt open in the editor.

Edit orig.txt to contain 8 integers, one per line.


● orig.txt - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Manage
Learn More
orig.txt ●

Restricted Mode is intended for safe code browsing. Trust this window to enable all features.

```
1
2
3
4
5
6
7
8
```

We can also open folders/directories in VS Code, just as we do on Windows. To open your current working directory in VS Code, you will type

code .

at the command line.

Do that now and look at the Copy.java file.

D. Compiling and Executing Java Programs

We do have VS Code (and Eclipse) available to us on the Linux terminal servers. However, this semester we are going to run some programs where we are using command line parameters, where we are giving the program information when we run it. Therefore, we want to be able to compile and run programs from the command line.

We're going to work with Copy.java, which is a fairly straightforward program to copy a file of integers.

To compile our programs, we'll use *javac,* the java compiler. To compile the Copy class, we type:
javac Copy.java

The program should compile without error. This will create a file named Copy.class, which is your compiled code. To run a Java program, you run java on the class that contains the main. Note that you use the class name, not the file name, so you would type:
java Copy

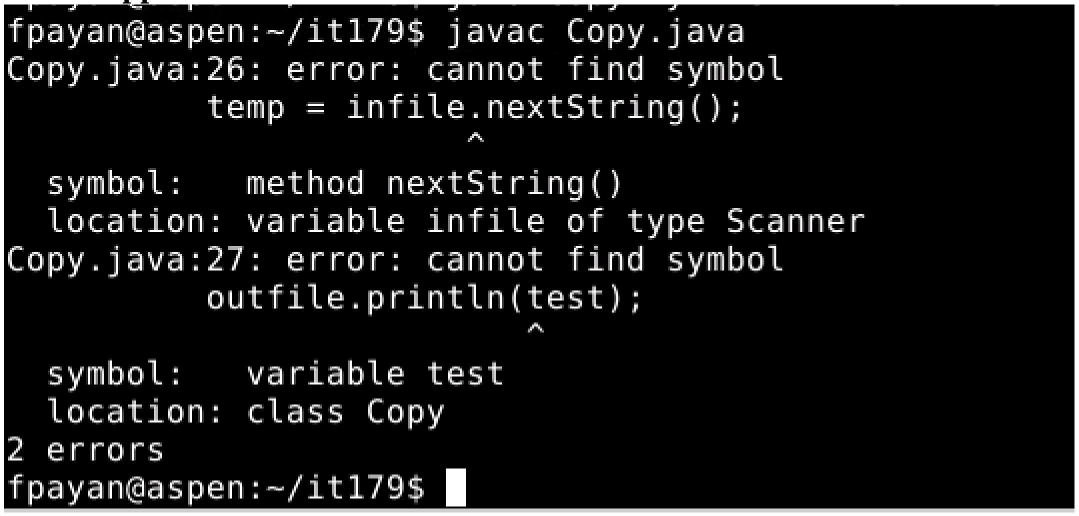
You'll notice that the program ran, but it didn't do much. That's because you didn't give the program all of the information it needed. Take a look at the source code again. You'll notice that the main function takes two parameters. These allow the program to take information from the command line, so that you give parameters to the program when you run it. This particular program takes the names of the input file and output file from the command line, making it more flexible.

Try this:
java Copy myfile.txt newfile.txt

Examine myfile.txt and newfile.txt to see whether or not the program worked.

Now that we've successfully compiled and executed a program, let's see what happens when compilation does not work.

Modify the file Copy.java to introduce a syntax error. Try compiling the program. What happens?



```
fpayan@aspen:~/it179$ javac Copy.java
Copy.java:26: error: cannot find symbol
          temp = infile.nextString();
                       ^
  symbol:   method nextString()
  location: variable infile of type Scanner
Copy.java:27: error: cannot find symbol
          outfile.println(test);
                          ^
  symbol:   variable test
  location: class Copy
2 errors
fpayan@aspen:~/it179$ 
```

It returns to the terminal the errors found in the program when compiling it.

Look at the other .java files you copied. Try compiling and executing the other program.

E. Practice

Write a program Average.java that accepts any number of command line arguments and prints their average (assuming that the arguments are integers). If the user does not supply any arguments on the command line, the program should print a message stating that. Use the parseInt method of the Integer class to extract the integer values form the Strings. Use exception-handling to deal with any non-integer values that are passed in ziand print an appropriate error message.

You can zip files on Linux from the command line by doing the following command:

zip *zipfilename filenames*

Zip Average.java into a file named hwork.zip. Submit your zip file to ReggieNet along with a PDF version of this document with your answers.

F. Submitting Your Work

To attach your files to ReggieNet, you'll need to run a web browser. That can be found in the toolbar at the bottom of the screen or in the Applications menu. Log in to ReggieNet (through my.illinoisstate.edu or by going to reggienet.illinoisstate.edu directly). Attach your zip file to the assignment, but don't submit yet, since you also need to attach the

PDF file of this document. Make sure you get out of the assignment on the Linux machines before you try to log in to ReggieNet from your Windows or Mac environment.