

1 Objectives

After completing the lab you would be able to

1. Understand the basics of LINUX directory trees and how to use them to your advantage
2. Know a few more LINUX commands: `mkdir`, `more`, `cat`, `rm`, `cd`, and `pwd`
3. Know how to edit a Java program
4. Compile and execute a Java program
5. Submit lab results electronically
6. Ask questions.

References: The following website and documents of various forms will be used throughout the lab.

- *CSCI 203 course website*: <<http://www.eg.bucknell.edu/~csci203/>>
- *The LINUX Documentation (HTML)*: <<http://www.eg.bucknell.edu/docs/linux>>
- *The LINUX Documentation (PDF)*: <<http://www.eg.bucknell.edu/docs/linux.pdf>>

2 Use the Text Editor to Set Up a Banner Template

In this course when you submit a lab or a project, it *must* have a `readme.txt` file. Every `readme.txt` file *must* start with a header containing your name, lecture section, lab section, and date. To simplify the process of providing this information every time, you will begin by creating and saving a template. When you start any lab work then you can insert this template into your file.

In this course whenever you need to do simple editing of a file, we recommend you use LINUX's text editor called **gedit**. You access **gedit** by selecting **Text Editor** on the **Accessories** menu under the **Red Hat** icon. Using Text Editor is very much like using "Notepad" on Windows or "TextEdit" on a Mac.

To-Do: Open a new File in Text Editor and Create a Banner file

1. Start the Text Editor by selecting **Text Editor** on the **Accessories** menu under the **Red Hat** icon.
2. Open a new file in Text Editor by pulling down the **File** menu and choosing the **New** option.
3. Enter the text shown in Figure 1 where you make appropriate changes in your name, professor's name, lecture and lab section number. Leave the information about date and lab numbers open.
4. Save the file by pulling down the **File** menu and choosing the **Save** option. In the new dialog window, enter the file name **banner** in the Name: field then click Save.

```
/*
*=====
* Name: Happy Student
* Course : CSCI 203 / Prof. LectureProf / MWF ?:00
* Lab Section: Monday 6-8AM
* Assignment: Lab #?, or Project #?
* Date: mm/dd/yyyy
*
*=====
*/
```

Figure 1: An Example of a Banner File

Having done this, you can use the file the rest of the semester. You will use the **banner** file later in the lab.

Note in this and the following labs, instructions for things to be done will be interspersed in the text as well as in the To-Do boxes. For example, here are two things to do. This is to force you to read the text and for you to slowly gain more independence.

Use the LINUX `ls` command to see that your **banner** is listed in your home directory.

Use the LINUX `cat` command to display the contents of a text file in the Terminal window. Try this now to display the contents of your **banner** file.

```
cat banner
```

3 Creating a `readme.txt` File

Now you should create a file called `readme.txt` for the work in this lab. Do the following.

To-Do: Creating `readme.txt` file and Copy and Paste a Banner

1. Start a new Text Editor window.
2. Create a new file named `readme.txt` by pulling down the **File** menu and choosing the **Save** option then entering the file name.
3. Copy the banner and paste it in to the `readme.txt` file. To do this drag the mouse pointer over the banner text to be selected with the left mouse button held down. The selected area should turn blue. Select **Copy** from the **Edit** menu. Move over to the `readme.txt` file and click where you want to paste the text. Now select **Paste** from the **Edit** menu and the banner text should appear. You could also use keyboard shortcuts of “Control c” and “Control v” in place of selecting **Copy** and **Paste** from the menu. Using the shortcuts can improve your productivity.
4. Edit the banner text to put in the lab number and the date of submission.
5. Save this edited file.
6. Quit both Text Editor windows.

4 A Few Words About Directory Trees

As you progress through the semester you will be creating a lot of files. As a means of keeping your account organized, LINUX provides the ability to create a special file known as a *directory* — on a PC or Mac you may know these as *folders*.

In computer file systems, directories are organized in a hierarchal fashion. Your account is itself associated with a directory that is referred to as your *home directory*. In the directory hierarchy of Bucknell’s file system there is a directory named `student` that contains 26 more directories named `a` through `z`. Each directory

named for a letter of the alphabet contains all the home directories of students whose account names begin with that respective letter. For instance, the student John Smith might have his directory in `s` if his account name is `smith`; if, on the other hand, his account name is `jsmith`, then his home directory would be in the directory `j`.

Each directory in the file system can contain several other directories. We refer to a directory within a directory as a *sub-directory* of the higher level directory, and the higher level directory as the *parent* of the sub-directory. For example, if your home directory was named `jsmith` it would be a sub-directory of the directory named `j`, and `j` would be the parent directory of your home directory.

When you are logged into LINUX, all references to file names are made relative to a *working directory*. When you log in, the working directory is set to your home directory. LINUX will look for any file you name in that directory. To refer to files in other directories, you must instruct LINUX to look in those directories. You will learn more about this, and about how to change your working directory, later.

Recall that a directory is just a special type of file. It should not surprise you that a directory can contain text and other types of files as well as numerous directories. In this lab, you will create and use directories and sub-directories to help you keep your account organized. While this may not seem like a major issue at the moment, by the fourth or fifth lab you will have come to appreciate this simple yet powerful concept.

5 Creating and Naming Directories

You will now create some directory structures in your account. The directory you will create in a few moments will be the home for all your work in this course including lab work, programming projects, and some other assigned work.

To-Do: Creating the Directory for Course Work

1. Open a terminal window.
2. At the LINUX prompt, issue the `ls` command. You will probably only see the files you worked with last week and the `banner` file.
3. Create a `csci203` directory by issuing the following command

```
mkdir csci203
```

4. Create a `CSCI203` directory by the following command

```
mkdir CSCI203
```

5. List your home directory using the following command.

```
ls
```

In addition to the files you created last week and `banner` file, you should see the following directories displayed in blue.

```
CSCI203 csci203
```

(They may have a `/` at the end instead of being blue depending on the systems setting. If so, the forward slash (`/`) at the end of a file name indicates that the file is a directory. **Note:** the slash character (`/`) is *not* a part of the file (directory) name! It is added to emphasize that the file is actually a directory.) Note that LINUX distinguishes between upper and lower case letters, which can be annoying if you think you typed something correctly, but you substituted a lower case letter for an upper case one or vice-versa.

Since you only need one directory for your course work, please delete the `CSCI203` one.

To-Do: Deleting CSCI203 Directory

1. Issue the command

```
rmdir CSCI203
```

In future labs we will always refer to your `csci203` directory as the directory where your work for the course is kept.

6 Some LINUX Commands

Knowing how to use directories and sub-directories and being able to manipulate the files in them will prove useful throughout this course. In this section of the lab you will get a lot of practice with files and a clear understanding of the file system. You will read about directories in the *The LINUX Documentation* but first you need to copy a couple of files so that your environment matches that expected. Copy these files to your home directory, *not* in the `csci203` sub-directory.

To-Do:	Copying Files
1. Go to your home directory by using the following command. <code>cd</code>	
2. Copy to your home directory the files <code>jack_theory</code> and <code>myhumpty</code> from the <code>lab02</code> subdirectory of the course directory. <code>cp ~csci203/2009-fall/student/labs/lab02/jack_theory .</code> <code>cp ~csci203/2009-fall/student/labs/lab02/myhumpty .</code>	
3. Read through the <i>Using Directories</i> section (Section 3.5 of Chapter 3) of <i>The LINUX Documentation</i> (HTML) and do the examples as indicated.	

As a result of doing the above exercises, a directory called `progs` will be created and used in subsequent exercises. So make sure you went through the exercises specified in the *The LINUX Documentation* (HTML). Be very careful to watch for error messages that may occur. These are an indication that things didn't go as intended. Also, make frequent use of the commands `ls`, to determine directory content, and `pwd`, to determine your current working directory.

When you have completed all the examples be sure you are in your home directory (use the `pwd` command to check). If you aren't there, issue the `cd` command with no directory argument to get to your home directory. Now that you are in your home directory do the following.

To-Do: Listing Directories Recursively

1. Issue the following command at the LINUX prompt.
`ls -R`

Cool, huh? The `-R` option instructs LINUX to *recursively* descend through the directories and perform the specified action, `ls` in this case, in each sub-directory.

Open a Text Editor window and open the `readme.txt` file by pulling down the **Open** on **File** menu.

Now you are going to copy the result of the `ls` command into your `readme.txt` file. What you will do is to copy the text from the screen and paste it into the file `readme.txt`. We will demonstrate another way to copy and paste.

To-Do: Copying Contents from Terminal to `readme.txt` File

1. Use Text Editor to open your `readme.txt` file if you haven't done so.
2. Insert into `readme.txt` the heading for this problem shown in Figure 2.
3. Left-click so the mouse is pointing to the screen where the result of `ls -R` was displayed.
4. Use the **left** mouse button to select (highlight) the text.
5. Left-click at the correct location in your Text Editor window where you would like the copied text to be inserted.
6. Click the **scrolling-wheel** mouse button to paste a copy of the selected text into your Text Editor file.
7. Save the `readme.txt` file and quit Text Editor by closing the window.

What you need to copy begins with:

```
[your-name@host-name ~]$ ls -R
```

And ends with:

```
[your-name@host-name ~]$
```

(host-name is the name of the computer on which you are working.) Having copied the data to the `readme.txt` file, you should put a header on the copied data. Skip a couple of blank lines after the banner, then add the five lines in Figure 2 **before** your data: If you are unsure that things have gone as they should, have

```
//#####  
//  
//   Exercise 1 in Lab 2  
//  
//#####
```

Figure 2: Exercise One Sub-heading

your professor or TA verify that you have done this correctly. Now save your `readme.txt` file.

Retain this text editor window in the corner of your workspace so that when instructed you can copy data into it, thus gradually building up the file, which you will hand in electronically later.

The time will come when you want to delete a directory for one reason or another. You learned how to remove (delete) files with `rm` in previous exercises. Using this same command you can also remove directories, including all the sub-directories and files they may contain. Can you guess why `rm` should be used with caution? Try the following.

<p>To-Do: Attempting to Delete a Directory</p> <p>1. Issue the command <code>rm progs</code></p>

Note that the directory `progs` should have been created when you went through the directory exercises in the The LINUX Documentation .

The message you see informs you that `progs` is a directory and just using `rm` will not delete a directory. The `rm` command used in conjunction with the option `-r` instructs the system to delete all the files in a directory (including any sub-directories) and the directory itself. Do the following.

<p>To-Do: Correct Way to Delete a Directory</p> <p>1. Issue the command <code>rm -r progs</code></p>

The system asks you if you want to look into the directory `progs`. Confirm the operation (y). Next the system asks you if you want to delete the file named `jack_theory`. Respond with no (n). Because you instructed the system not to remove the file `jack_theory`, the directory is not empty and the system will not delete it. This helps to guard against deleting files and directories accidentally. Now do the following.

<p>To-Do: Actually Deleting Files in a Directory</p> <ol style="list-style-type: none">1. Issue the command again <pre>rm -r progs</pre> <p>only this time answer y to all prompts.</p> <ol style="list-style-type: none">2. Now remove the <code>labs</code> directory and all its contents (this only requires one command).3. Next remove the file <code>myhumpty</code> from your home directory.

Next you will create some more directories for the course work. Make sure you are in your `csci203` directory. Use the command `pwd` to check. If you are in the wrong place, use the `cd` command to move there. Using your newly gained knowledge to do the following.

To-Do:	Creating More Directories
<ol style="list-style-type: none">1. Create two sub-directories in your <code>csci203</code> directory with the names <code>lab01-loginName</code> and <code>lab02-loginName</code> where you replace “loginName” with your login name, e.g., <code>lab01-xyz03</code>.2. Move the files you worked with last week into your directory <code>csci203/lab01-loginName</code>3. Move the file <code>banner</code> into your <code>csci203</code> directory. It will always be there to use when creating a new <code>readme.txt</code> file.4. Move your <code>readme.txt</code> file into your <code>csci203/lab02-userName</code> directory.5. Now use <code>cd</code> to move into that <code>lab02-userName</code> directory and use Text Editor to open the <code>readme.txt</code> file that is there. You will be adding material to it as the lab proceeds.	

If you are unsure, have your professor or TA check what you have done.

7 Your First Java Program

Today you get your first CSCI 203 experience with a Java program. The first step of the process is to create a Java program in plain text following specific Java syntax using an editor such as Text Editor. Then save the program in a file following the naming conventions, e.g., the file extension must be `.java`. Finally you can compile and run the Java program.

To-Do:	Creating a Java Program
<ol style="list-style-type: none">1. Open a new Text Editor window.2. Start a new file by selecting New on File menu.3. Now type in the program shown in Figure 3.4. Select Save from File menu. Click on “Browse for other folders”. In the dialog box, select your <code>lab02-userName</code> directory by double clicking. In the Name: field enter <code>Hello.java</code> then press Save.	

```
// Program to print "Hello World!"

// Class heading
public class Hello {

    // Method heading for main
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Figure 3: First Java Program

Now that you just created and saved a Java program, you can compile and run the program at a LINUX terminal. You will first use `javac` to compile the Java source code `Hello.java` into Java byte code `Hello.class`. Compiling translates a Java program from plain text into byte code so that the computer can execute it. You then will use `java` to actually execute or run the program. The command `java` runs the Java Virtual Machine (JVM) on the byte code created by the command `javac`. (We also call this executing the Java program.) Java byte code is machine-independent, in other words, you can use this byte code on any computer, be it a LINUX computer, a UNIX computer, or a Windows computer, as long as the computer has the JVM on it.

While completing the following task, you may encounter errors at the compiling stage or at the execution stage. If you see error messages while you compile the program, compare what you typed to the program listed in Figure 3 make any corrections, save the file, and recompile it. If you see errors at execution stage, that means your program has semantic errors (or logical errors).

Your professor or TA can assist you if things get a bit confusing. Do the following.

To-Do: Compiling and Executing a Java Program

1. Make the working directory for your terminal window
`lab02-userName`.
2. Compile the file you typed in from Figure 3 by typing the following at the LINUX prompt:
`javac Hello.java`
and press enter.
3. Type the following on the command line.
`java Hello`
you should see the words
`Hello World!`
appear on the screen.

Now complete the following activities using `Hello.java` as a base.

To-Do: Modifying a Java Program

1. Modify the program so that, in addition to the message `Hello World!`, the program also displays your name, with a space before and after it. Your name should appear on the same line as `Hello World!`.
2. Compile and run the modified program.
3. Now make active the `readme.txt` edit window you opened earlier and create a new header (actually, just copy and paste the exercise 1 header and change the name). Label this new header `Hello.java`.
4. Copy and paste this modified “Hello World” program code into your `readme.txt` file.
5. Copy and paste the output from a test run of the program.

8 What to Hand In Electronically

Save your `readme.txt` file at this point. You might want to look it over to make sure that what is there is in a nice format and that nothing is missing. The contents

of `readme.txt` should contain the following.

1. The result of recursively listing the contents of your home directory using the list command `ls -R`.
2. The source code and the result of a test run from the `Hello.java` program.

9 Setup the CSCI203 Lab and Project Drop Boxes

To setup your CSCI203 Lab and project drop boxes, you need to do the following steps only once this semester.

1. Double click on the icon that looks like a folder with a house. This opens the File Browser.
2. Click on the icon that looks like a sheet of paper and a pencil that is found on the left side of File Browser. Fill in the **Location:** field with

```
smb://netspace.bucknell.edu/computer_science$
```

and press **Enter** key. You will be prompted for your netspace password. Then double click to enter `public` folder.

3. Select `CSCI203 Lab drop_box` folder and drag it into the panel on the left.
4. Select the `CSCI203 Wittie drop_box` or `CSCI203 Zaccione drop_box` or `CSCI203 Guattery drop_box` folder depending on your **lecture** instructor. Drag it into the panel on the left. This is the folder where you will drop your programming projects later in the semester.
5. Close the File Browser.

10 How to Hand In Electronically

In order to receive a grade on a lab you must name the lab directory properly, including lab number and your login name, e.g., `lab02-xyz03`,

Assuming your `readme.txt` file is in your `lab02-userName` directory, do the following:

1. Double click on the icon that looks like a folder with a house. This opens the File Browser.

2. In the File Browser, find your folder `lab02-userName` and drag it to `CSCI203 Lab drop_box` in the left pane. Be sure to put it in the *lab* drop box. If you have logged off since your last use of netSPACE, you will be prompted for your netSPACE password.
3. **Important:** Check with the TA to make sure that the drop worked.

If later, you decide you want to submit a revised version, just drag over your folder to `CSCI203 Lab drop_box` again. The system will ask if you want to replace the old folder and you answer yes.

Note that you can not see what is inside the `CSCI203 Lab drop_box` nor can you replace someone else's folder.