COMP 9 Lab 1: Meeting Kate

Out: Thursday, January 27th, 1:15 PM

Due: Thursday, February 3rd, 1:15 PM

1 Overview

This assignment introduces you to the Linux operating system, the ruby interpreter, and the Kate text editor. You will use these tools to write a Ruby program and run it.

2 Navigating Linux and Kate (Lab part 1)

The computers should already be on; if the screen is blank, simply hit a key on the keyboard and wait a few seconds. You should be able to log in with your recently created CS account. Enter your CS user name and password to log in. Right-click on the desktop and select "Terminal" from the popup menu. A window will appear. This window is running a *Linux shell*. The shell takes commands typed on the keyboard and executes them. You should see a *prompt* that looks something like this:

```
lab116s{ndaniels}:
```

This prompt indicates that Linux is ready to execute your next command. You'll use the prompt to launch the Kate text editor, navigate the *filesystem*, and submit your completed assignments. The *filesystem* is the hierarchical directory ("folder") system of Linux. Unlike Windows, in which the file hierarchy starts with a drive letter and `:\` (for example, `C:\`), the Linux file system always starts with a `/`, which is called the *root* of the file system.

When you first start the Terminal, you are in your *home* directory. Now, create a new directory within your home directory (a *subdirectory*) to store all of your Comp 9 files for the semester. To create this new directory, type the following at the prompt:

```
mkdir comp9
```

The command `mkdir` is short for "make directory." Next, you will "move into" the new comp9 directory, much as if you had double-clicked a folder on Mac OS or Windows. Type:

```
cd comp9
```

The `cd` command stands for "change directory"

Kate is the text editor you will use to create and edit your Ruby programs. Kate looks a little like a word processor, except you'll notice that there are no text formatting options. Kate creates plain text, which the *source code* to any program really is. At the terminal command prompt, type:

```
kate &
```

This opens a new window running the Kate program. The `&` is there so that you get your command prompt back immediately; otherwise, you would not be able to issue new commands in that same window until you quit Kate. If Kate prompts you about opening a session, just click the 'new' button. Kate has already created a new file for you to start working in, but it hasn't yet saved it and doesn't know what you want to call it. Let's start with a lab report file for today's lab. Click on the Save button in the toolbar, and in the 'Location' field, type `report.txt` and click save.

In your new file, type the following:

```
# File: report.txt
# Name:
# Date: 01/27/2011
# Lab 1
```

(Fill in your name where appropriate). When you are finished editing the file, you can save it by clicking the Save button, or hitting `ctrl-s`.

For the rest of the lab, you will need to switch back and forth between the terminal window and Kate. For this section, refer to the *summary of commands* at the end of this document. You'll be entering some commands, and writing your answers in the file `report.txt`.

1. What is your current working directory?
2. What are the contents of your current directory?
3. What are the contents of your home directory?
4. Make a new directory called `backup`. Make a copy of the file `report.txt` and move it to `backup`. What is the sequence of commands you used?

5. What are two ways (two sequences of commands) to list the files in the backup directory?

6. What happens when you try to delete the backup directory and its contents? How did you complete this task?

7. What does the command `cal` do?

8. How do you use the `-y` option to cal?

2.1 Handing in your solution

You will submit all your work at the end of the lab. Make sure you have saved the file `report.txt`; you will not need it for a while but you'll have to submit it later.

3 Computing the hypotenuse of a right triangle (Lab part 2)

In this part of the lab, you will write your first Ruby program, and you will run it. Ruby is an *interpreted* language, which means that unlike some other languages, there is no separate *compiler* that you need to run to produce a usable program. To start, create a new file called `hypotenuse.rb` in Kate. Begin with the boilerplate we discussed in class:

```
#!/usr/bin/env ruby

# File: hypotenuse.rb
# Name:
# Date: 01/27/2011
# Lab 1
```

Save the file. Though it doesn't actually contain any Ruby code yet, this file can be run by the ruby interpreter.

3.1 Running the program

Switch back to your Terminal window, and type `chmod 755 hypotenuse.rb` and hit enter. This tells Linux that the file should be treated as an *executable program* which means you'll be able to run it from the command line. Then, simply type `./hypotenuse.rb` and hit enter. You should not see any error messages; indeed, you should just see the Linux prompt again because your program produced no output. So, let's fix that!

3.2 Editing the program

Now it's time to fill in the code to make your program work. Initially, get the part that computes the hypotenuse of a right triangle to work. So, as we did in class, define `a` and `b` to be specific values of your choosing (you might wish to choose easy numbers like 3 and 4, so that you know the correct answer will be 5). Your program should `puts` the answer to the screen. Refer to lecture notes and your text; don't be afraid to ask for help, as well. When you are done, save the program and run it just as you did before. It should produce the correct output.

3.3 Asking for input

So far, we've produced output, but we have not asked for input. Just like `puts` prints output to the screen, `gets` takes a line of input. Your program can `puts` instructions to ask for input, and then `gets` the input you type in the terminal. However, `gets` always treats whatever you type as a `String`, so if you want to turn the string `'12'` into the integer `12`, you would use the `to_i` (to-integer) conversion method.

```
'12'.to_i    => 12
```

So, edit your program `hypotenuse.rb` so that it asks for the sizes of the two known sides. It should then compute the hypotenuse associated with those input lengths.

Remember, $c^2 = a^2 + b^2$

3.4 Optional (extra credit)

Extend the hypotenuse calculation program to request the name of the user before getting the sides of the triangle, and greets the user by name before producing an answer. Do not modify your `hypotenuse.rb` file; instead, make a copy to a new file named `hypotenuse_name.rb`. This program should produce output that looks like this:

```
Noah, the hypotenuse of a 3,4 triangle is 5.0
```

Hint: when you read a line of input using `gets`, it includes a special character at the end, called a `newline` (you can think of it as a 'return' character; it's usually represented as `\n`). To remove this character, you'll have to use the `chomp` method, which removes the last character of a string if and only if it is a `newline` character.:

```
name = gets
name = name.chomp
```

If you don't do this, you'll find you get slightly different output from what you want:

```
Noah
, the hypotenuse of a 3,4 triangle is 5.0
```

Another hint is that just as you can turn a `String` into an `Integer` with `to_i`, you can turn an `Integer` or `Float` into a `String` with `to_s`. You have to do this to concatenate that `Integer` onto a `String`.

3.5 Handing in your solution

When you are satisfied that your program works perfectly with a variety of inputs, submit and your lab report to be graded using the following terminal command:

```
provide comp9 lab1 report.txt hypotenuse.rb
```

If you also did the extra credit, make sure you include that as well:

```
provide comp9 lab1 report.txt hypotenuse.rb hypotenuse_name.rb
```

When you are done in the lab, make sure you *log off* from your account; otherwise someone else could gain access to your account.

4 Common Linux commands

Command	Description	Example
`pwd`	print the current working directory	
`ls`	List the files and subdirectories in the current directory or specified directory	`ls` `ls backup`
`cd`	Change the working directory	`cd /h/ndaniels/comp9`
`cd ..`	Change the working directory to the parent directory.	
`cd ~`	Change to your home directory.	
`mkdir`	Create a new directory	`mkdir comp9`
`cp`	Copy a file from one place to another.	`cp test.rb test2.rb` `cp test.rb /h/ndaniels/comp9/`
`mv`	Move or rename a file	`mv tset.rb test.rb`
`rm`	Delete a file. There is no trash can or recycle bin. It's gone!	
`rmdir`	Delete a directory. This command only works for directories that are empty.	`rmdir my_empty_directory`
`man`	Display the "help" (manual) page for a particular command	`man cal`

Table of common Linux commands