ENSE 370 Lab1 Software Tools

Trevor Douglas
Software Systems Lab Instructor
University of Regina

1 Background

1.1 Command Line Background

This lab gives a refresher on Unix-like Command Line Interfaces, and teaches you how to copy and submit your files to Snoopy. This is a requirement to submit all your labs and assignments for this class.

The command line may feel slow and unintuitive at first, but with practice it can become quicker and more convenient than working in a GUI for programming tasks.

1.2 SSH

Secure Shell SSH is a protocol for securely connecting to a remote computer. Functionally, this is similar to opening up a shell inside of a shell, but instead you are opening a shell on a different computer.

Basic usage:

See the following command :

```
$ ssh username@server.com
```

You will then be prompted for your password.

If this is your first time connecting, you may also need to confirm the identity of the server.

Some ssh servers require key-based authentication instead of a password. To use this, you will generate a public and private key files, storing the public key on the server and the private key on the client. When you authenticate you will provide the private key in lieu of a password. You'll learn more about authentication in other SSE courses.

You log in to uregina.ca using your novel credentials. (The same as URCourses.)

```
$ ssh username@uregina.ca
```

You log in to snoopy using your snoopy credentials. (Provided by Dr. Yow)

```
$ ssh username@snoopy.engg.uregina.ca
```

when you are finished with your session, you may use the command

```
$ exit
```

1.3 Secure Copy

Secure Copy (scp) is an application for copying files to and from a remote server over the ssh (or sftp) protocol. It works like copy, but additionally requires parameters specifying the remote location.

Basic usage: “‘ scp source destination “‘ where local files are described the same way they would be on a local disk, but remote files are described with the username@server.com syntax, a colon :, and then the file or directory location on the remote server.

We can also use tunneling with scp as well, to copy a file in two jumps.

Here are four common use cases which you can use as a basis for constructing your own commands:

1. Local file to remote file:

   ```
   $ scp file.txt username@server.com:/remote/directory
   ```

2. Remote file to local file:

   ```
   $ scp username@server.com:file.txt /local/directory/
   ```

3. Local directory to remote:

   ```
   $ scp -r /local/directory/ username@server.com:/remote/directory/
   ```

4. Remote directory to local:

   ```
   $ scp file.txt username@server.com:/remote/directory
   ```

You can also use a graphical application such as WinSCP, or on Linux connect to a remote directory through some graphical file browsers.

—

1.4 In terminal editors

So you successfully copied your files to the server. However, all our code will be run on snoopy so best to develop there but if not, then you must at least test your code there. How can we edit it directly on the server?!

Terminal editors to the rescue! These are text editors which run right in the shell.

Here are some in-terminal editors, from historical to modern

- ‘ed‘ - The original. Primitive, but technically functional.
- ‘vi‘ - visual editor. Thanks, I hate it.
- ‘vim‘ - a improved drop in replacement for vi. Steep learning curve.
- ‘emacs‘ - Powerful editor with macros. Veteran of the editor wars.
- ‘pico‘ - pine composer.
- ‘nano‘ - a better drop-in replacement for pico, so named because nano

1.5 Visual Editors

I have installed a visual editor named Kate. You will need an X server.

Mac - Xquartz https://www.xquartz.org Windows - Xming https://sourceforge.net/projects/xming/

```
$ ssh -Y [username]@snoopy.engg.uregina.ca
$ kate &
```

** also use -Y when logging into @uregina.ca

2 Procedure

- Login to snoopy.
- Create directories and files and move around them.
- Copy files to and from the server.
- Complete Phase 1 and Phase2.
- You must show Phase 1 to your lab instructor before leaving the lab.

3 Phase 1 - Hello World

- Login to snoopy and create a java Hello World application. This is to help you understand the process to edit your code.
- Compile your code and run this application.

  ```
  $ javac HelloWorld.java
  ```

- Run:

  ```
  $ java HelloWorld
  ```

4 Phase 2 - Hangman

- Create an application that has the following features:
- Allow the user to enter a secret word.
- The other user will guess the secret word by guessing letters.
- The program will prompt the user by asking them to pick a letter or guess the word.
- Each round print the secret word with the letters that have been guessed. With the unknown letters use "*".
- If the user guesses a letter that is not part of the word, notify the user. The user is only allowed 5 bad guesses during the game otherwise it is game over.
- If the user guesses the wrong word then it is game over.

4.1 Hints

Hints: You may wish to make use of the following utilities:

- Java 'Scanner'
- Java 'StringBuilder'
- Java 'Console' readPassword() method
- String Class Functions, such as '.toUpperCase', '.contains', '.charAt', '.equals'

5 Phase 3 - Submission

Assignments and labs for this course are submitted directly on Snoopy. You must submit a readme.txt explaining how to run your code.

Here is how to submit phase 2:

Log into Snoopy. From Snoopy, submit your Hangman.java file using the following command:

```
$ ~ense370/bin/submit L01 Hangman.java readme.txt
```

You may overwrite this file up until your due date.

You may check that you have submitted your assignment correctly by typing

```
$ ~ense370/bin/submit --check
```