Trent University

COIS3380H

Winter 2023

Lab 0 - Doesn't count towards your grade

Due: none

By now, you should have received your credentials for the server (loki.trentu.ca) we will be using in this course. If you haven't, then go to the portal and get started.

NOTE: You are required to work on the university server for your assignments. You can actually develop elsewhere but your final submission MUST reside, work and be submitted from loki.trentu.ca.

Accessing the system:

You will be accessing the system using two different tools that we have already reviewed in class. They are ssh and scp. Both of these tools require you to provide credentials to the system.

Note: As of the Winter term of 2022, access to Loki SHOULD/must be done using ssh keys as explained in class. The objective of this lab is to ensure that you have your key pairs working and are able to connect and up/down load files to the server.

Lab 0 - Kicking the tires on Linux.

The objective of the lab is to familiarize you with the command line interface used on Linux systems. The command line is used by entering VERBS using the keyboard. Some "verbs" have parameters/modifiers, some don't. Sometimes the parameters are optional.

Using verbs in this way, you can build up sequences of commands that can be used to process a stream of data. You can also place these verb and their parameters in a flat text file and have the operating system interpret their contents. This is know as shell scripting. Using shell scripts, it is possible to use the command line verbs just like a programming language. This will allow you to automate a number of repetitive tasks.

Before you are given an assignment which counts towards your final grade, this lab is intended to allow you to explore the system and its commands before it becomes important.

Try out the various commands below. Try variations. Combine the commands and see how you can modify the results you get. Play is the best way to learn.

The LAB

Step 1: Logging in

- log into the system using a ssh terminal emulator PuTTy or TeraTerm?

Step 2: Try some simple commands:

- You are now in a shell. Commands are typed here !
- If you don't know what a command does or how its behaviour can be modified, use the "man" pages: e.g. **`man mkdir`**
- try some simple commands and look at the output:
  - **`pwd - present working directory`**
  - **`ls  - list my files (DIR in Windows/DOS)`**
  - **`who - who is currently logged into the system`**
  - **`whoami - what account am I currently using?`**
  - **`date   - what is the correct date`**
  - **`mkdir lab0  - create a directory called lab0`**
  - **`cd lab0 - change current directory to subdirectory lab0`**
  - **`pwd     - check to see where I am`**
  - **`cd $HOME - change to the dir pointer to my $HOME`**
  - **`pwd      - check to see where I am`**
  - **`man grep - check the man page for the grep command`**
  - **`ps -ef   - show which processes are running`**
  - **`cat /home/COIS/3380/lab3/small_world.txt - cat a file`**
  - **`env      - show all my environment variables`**

Adding some command line parameters to the verbs used above:

- Adding some qualifiers to how you want commands run the ls verb:
  - **`ls -lS`**
  - **`ls -i`**
  - **`ls /home/COIS/3380/`**
  - **`cd /home/COIS/3380/`**
  - **`ls -i`**
  - **`ls -1 (that's a one: 1)`**
- **`cd ~`**
- **`pwd`**
- **`grep -e laughter /home/COIS/3380/lab3/small_world.txt`**
- **`wc /home/COIS/3380/lab2_document.txt`**

The following sections are here to give you an appreciation of the "filtering" of data streams by combining commands together using PIPEs. Each line I'm asking you to type is made up of multiple commands separated by a PIPE symbol "|". To get a good idea of what the PIPE does, try these commands one section at a time and build up the command line testing each component as you add them.

Chaining commands together using PIPEs

- ls -lt /home/COIS/3380/lab3/ | head -n 5
- ps -ef | grep -e $USER
- ps -ef | grep -e `whoami` (these are back-tics no quotes)
- cat /home/COIS/3380/lab2_document.txt | grep -e blue

Environment variables: (convention dictates upper case for variable names)

- env | grep -e LOGNAME
- env | grep -i -e loki
- echo $LOGNAME
- echo $SHELL
- TIMESTAMP=`date -I` *(notice the back-ticks!)*
- echo $TIMESTAMP
- FILENAME=${USER}_${TIMESTAMP}
- echo $FILENAME

More advanced commands:

- wc -l /home/COIS/3380/lab0_ip_addresses.txt
- uniq -c /home/COIS/3380/lab0_ip_addresses.txt | head -n 20
- sort /home/COIS/3380/lab0_ip_addresses.txt | uniq -c | wc -l
- sort /home/COIS/3380/lab0_ip_addresses.txt | uniq -c | head -n 20
- sort /home/COIS/3380/lab0_ip_addresses.txt | uniq -c |head -n 20 |sort -n
- sort /home/COIS/3380/lab0_ip_addresses.txt |uniq -c| sort -r | head -n 20

Making a shell script:

First, create a directory for all of our work for lab0.

```
mkdir lab0
```

Change your shell environment to use this directory for all future work

```
cd lab0
```

Make sure you are in the proper directory:

```
pwd
```

It should come back with something like:

/home/*YOURUSERNAME*/lab0

Now, use nano (or vi if you prefer) to create a text file. call it lab0_script.sh. Even though UNIX doesn't recognize extensions, for the purpose of this course and to make things more obvious, we'll adapt the "file extension" concept (for now).

```
nano lab0_script.sh
```

The screen will blank and show you an empty file (unless it already exists of course). Type the following lines into the file:

```
#!/usr/bin/bash
echo Hello World!
```

> **NOTE: there's a syntax standard for key sequences. The caret symbol represents the Control key (ctrl). When you see it preceding another character, say C, it means to press and hold down the CTRL key while you press the C. so the ^C text means press and hold down the CTRL key, then press C and release the CTRL key.**

To save your work in nano, press ^O to save the file to disk, press enter when it shows you the filename. Press ^X to exit to the command line

Now, change the properties of the text file lab0_script.sh so that the O/S recognizes it as a script it can execute:

```
chmod u+x lab0_script.sh
```

You can now run the script: `./lab0_script.sh`

Adding variables to your work:

Edit a new script and call it: lab0_script_2.sh
Type the following lines into the file (try copy-paste!);

```
#!/usr/bin/bash
echo Hello this script is running on the server called $HOSTNAME
echo the current time is: `date`
echo
TIMESTAMP=`date -I`
MYVARIABLE=${HOSTNAME}_${USER}_${TIMESTAMP}.txt
ls -lt > $MYVARIABLE
ls -lt $MYVARIABLE
```

Again, make the file executable and run it. Have a look at the output.

Make your script more flexible by allowing parameters to be pass to it from the command line:

cp lab0_script_2.sh lab0_script_3.sh *(notice the copy keeps the execute flag!)*
nano lab0_script_3.sh

Note: for help with syntax of bash commands, look at this web site: https://tldp.org/LDP/abs/html/index.html

Change the code around to look like:

```
#!/usr/bin/bash
echo Hello this script is running on the server called $HOSTNAME
echo the current time is: `date`
echo

if [ -z $1 ]; then
   echo You forgot to supply an argument on the command line!
   exit 1
fi

TIMESTAMP=`date -I`
MYVARIABLE=$1_${USER}_${TIMESTAMP}.zip
echo Creating the backup archive: $MYVARIABLE
zip $HOME/$MYVARIABLE  $HOME/lab0/*
```

Save it (^O, ENTER, ^X), then run it at the command line. Don't forget to add the "parameter" on the command line which will be passed to the script as $1.

To check your work:

cd $HOME

unzip -l $HOME/Lab0_jacques_2022-01-17.zip

Look in your $HOME directory for the created ZIP file. Check to make sure the ZIP file contains what you think it does. For that you can use the unzip command with the -l parameter (to LIST the contents of the ZIP).

[jacques@loki ~]$ unzip -l $HOME/Lab0_jacques_2022-01-17.zip

```
Archive:  /home/jacques/1_jacques_2022-01-17.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
      228  01-13-2021 19:18   home/jacques/lab0/lab0_script_2.sh
      356  01-13-2021 19:22   home/jacques/lab0/lab0_script_3.sh
       35  01-13-2021 19:15   home/jacques/lab0/lab0_script.sh
      375  01-13-2021 19:25   home/jacques/lab0/loki.trentu.ca_jacques_2021-01-13.txt
        0  01-13-2021 19:25   home/jacques/lab0/[log_lab0]
---------                     -------
      994                     5 files
```

Logging your work

When it comes time to submit your work, you will need to supply some evidence that your code works. To do this, you will need to create a LOG of your session while you are testing.

- Use your terminal emulator to log the session
  - Since the emulator is running on your computer, the log file will be created on it and not on the loki.trentu.ca server.
  - You will need to upload the logs to the server BEFORE you create the ZIP file for submission

- use the script shell command to create a local log file
  - locally on the server, you can use the script command to create a log file of your terminal session
  - The advantage to this method is that the log file will be created on the server itself.
  - The downside: the log will collect garbage characters as you press the backspace key to delete commands or the arrow keys to scroll back to previous commands. You will need to clean them up a bit before submission.
  - use the man pages for more info on script.
    - script [log_filename]
    - run your commands here
    - type in "exit" to close the log file.
- An example session:

```
[jacques@loki lab0]$ script my_logfile.txt
Script started, file is my_logfile.txt
   [jacques@loki lab0]$ ./lab0_script_2.sh
   Hello this scrip tis running on the server called
   loki.trentu.ca
   the current time is: Sat Aug 12 11:02:04 EDT 2017

   -rw-rw-r--. 1 jacques jacques 421 Aug 12 11:02
   loki.trentu.ca_jacques_2017-08-12.txt
   [jacques@loki lab0]$ exit
exit
Script done, file is my_logfile.txt
[jacques@loki lab0]$
```

Zip command:

- use the man page for zip to find out how it works
- alternately: zip --help at the command line
- how to create a zip file: zip zip_filename path_to_file_targets/file_targets
- create a test zip
- list the content of the zip to make sure it is built properly (*unzip -l*)
- transfer the zip to the local PC using an SCP tool..