Faculty of Computer Science, Dalhousie University *13-Sep-2018*

CSCI 2132 — Software Development

Lab 2: Unix Utilities; Emacs

Location: Teaching Labs Instructor: Vlado Keselj
Time: Thursday

Lab 2: Unix Utilities and Emacs

Lab Overview

- Review of login and basic SVN
- Autocompletion in shell
- Emacs: some operations
- Some Unix utilities
  - uniq, sort, cut
- A pipe

Important Note: You should use the detailed notes to complete the lab, not the slides. The slides are meant to be used only by the TA to show on the screen while guiding you through the lab.

(You are looking at the notes here, so you should use this document.)

In this lab, you will first learn autocompletion, a feature of the Bash shell. You will then learn more about using `emacs`. In particular, we will learn some hot keys for basic common editing functionalities. Finally, you will learn three Unix filters: `uniq`, `sort` and `cut`, and get some practice on using Unix pipes.

Be sure to get help from teaching assistants whenever you have any questions.

Step 1: Logging in. First, similarly to Lab 1, you need to perform some steps to login to bluenose and create the lab directory. We will also include an SVN update step:

1-a) Login to server bluenose.cs.dal.ca via SSH from a CS Teaching Lab computer or from your own computer. If you use Windows, you can use PuTTY.

1-b) Change your current working directory to the SVN directory prepared in Lab 1. It should be the directory: `~/csci2132/svn/`*CSID* where *CSID* is your CSID.

Step 2: Check lab1 directory. First, we will run a new SVN command 'svn update':

2-a) Run the SVN update command as follows:
```
svn update
```
It is assumed here that you finished Lab 1. The command "`svn update`" will bring any possible changes in the SVN repository to your local working copy. Normally, this will not make any changes to your working copy, but it is possible that the instructor or TAs made some changes to the repository that you need to update. If you notice that a new file whose name starts with `marker` appears, you will need to check it and possibly update it.

2-b) Go to the `lab1` directory. If there is a new file named `marker-a1q1.txt` or similar, you need to check it using Emacs, or another editor. The file may contain instructions from the instructor or a TA that you should read. If the instructions ask that you make come changes to the file, you should do it, save the file, and run

the command:

```
svn commit -m'The marker file updated'
```

It is also a good idea that you check that the files in `lab1` are submitted. You can do this by checking the URL that was mentioned earlier, or using the following command:

```
svn status -v
```

The output should be similar to this:

```
     11806    11805 CSID       .
     11806    11805 CSID       HelloWorld.java
     11806    11805 CSID       HiWorld.java
```

where CSID would be your CSID, the numbers may be different, and you may have another line for the marker file if it is there.

If you use the following command:

```
svn log -v
```

the output should be similar to this:

```
------------------------------------------------------------------------
r11805 | CSID | 2018-09-12 16:44:00 -0300 (Wed, 12 Sep 2018) | 1 line
Changed paths:
    A /CSID/lab1
    A /CSID/lab1/HelloWorld.java
    A /CSID/lab1/HiWorld.java

lab1submitted
------------------------------------------------------------------------
```

Step 3: Prepare lab2 directory.

3-a) Go to the main SVN directory. If you are currently in the directory '`lab1`' you should simply run the command '`cd ..`'. If you run the command '`pwd`' you should see that your current directory ends as follows:
...CSID`/csci2132/svn/`CSID
where CSID is your CSID.

3-b) Create a directory named `lab2`

3-c) Add the directory `lab2` to SVN and submit it. If you forgot, you may need to check Lab 1 to refresh your memory about using the svn commands 'add' and 'commit'. Remember that the 'commit' command requires a log message. You can use something like:
`svn commit -m'Directory lab2 created'`

3-d) Change your current directory to `lab2`

Step 4: Autocompletion. Autocompletion is a feature of the Bash shell, which is probably your default shell on bluenose. Autocompletion is a very handy feature: it can complete a command name or a file name that you have begun typing if you have typed enough to uniquely identify it.

To try this, follow the instructions below to copy `~/csci2132/lab1/HelloWorld.java` to the `lab2` directory:

4-a) In the terminal, type: `cp ../lab1/Hel`

4-b) Press the tab key. Then, if `HelloWorld.java` is the only file in the `lab1` directory whose name begins with `Hel` the shell completes the filename on the command line so that you will see
`cp ../lab1/HelloWorld.java` in your command line.

4-c) If there is another file whose name starts with `Hel`, say, `HelloWorld.class`, in directory `lab1`, then the shell will only complete the filename until the character before the position at which these two filenames differ. That is, you will see `cp ../lab1/HelloWorld.` in your command line. In this case, type `j` and press tab again.

4-d) Complete the command line

```
cp ../lab1/HelloWorld.java .
```

by typing one more character and pressing the Tab key again, or by typing the remaining characters. Do not forget to add period (‘.’) at the end of the command, which represents the current directory. Press enter. This will copy the file `HelloWorld.java` from directory `lab1` to `lab2`.

After you try this, you can see that this feature is quite useful when you type commands that process files with long names. This can also be used to autocomplete directory names.

SVN submission: At this point, submit the file ‘HelloWorld.java’ in the directory ’lab2’ to the SVN repository. Remember that this is a two step process, you first enter:

```
svn add HelloWorld.java
```

and then

```
svn commit -m'HelloWorld.java file submitted.'
```

Step 5: Emacs: Beginning and end of a line. We will now learn `emacs` hot keys that move cursors. You are probably already familiar with the arrow keys, which move the cursor one character in four directions, and PgUp and PgDn, which move courses one page up and down. The Home and End keys move cursor to the beginning and end of a line. Another way to move cursor to the beginning and end of a line are C-a and C-e. Remmber that according to the Emacs convention C-a is the same as Control-a (pressing Ctrl key and ’a’ key on the keyboard) and C-e is the same as Control-e.

Make sure that you are in the `lab2` directory. Use:

```
emacs HelloWorld.java
```

to open the Java source file. Hint: Use autocompletion again.

Practice the following two hot keys untill you memorize them:

- C-a Move to the beginning of a line
- C-e Move to the end of a line

You should remember the hot keys in Emacs that you learned in the last lab:

- C-x C-s Save a file
- C-x C-c Exit Emacs

Now you should exit Emacs.

Step 6: Emacs: Cut and paste. Use Emacs to edit a file named `names` in the directory `lab2`. Type the following line in Emacs and press Enter:

```
Alan Turing
```

where the rectangle (▯) shows the position of the cursor. Now, you should go up, to the beginning of the line ‘Alan Turing’, press C-k twice, and then press C-y twice. You should end up with the following contents of the Emacs buffer:

```
Alan Turing
Alan Turing
```

You can notice that C-k deletes a line to the end and stores it in an internal cut-and-paste buffer. The second C-k will delete the end-of-line character and appends it to the buffer. The hotkey C-y pastes the buffer content to the position in text. In C-k, ’k’ stands for “kill a line” and in C-y ’y’ stands for ‘yank a line’.

Add one more line with content “Ken Thompson” to the file, and make sure to finish the last line with a new-line character by pressing Enter. The content of the Emacs view should be:

```
Alan Turing
Alan Turing
```

```
Ken Thompson
```

Go to the beginning of the file, i.e., to the first character of the first line, and press C-k six times (without entering any other characters). You will see that all three lines will disappear. Now press C-y twice and the content of the file should be:

```
Alan Turing
Alan Turing
Ken Thompson
Alan Turing
Alan Turing
Ken Thompson
```

The commands that we just practiced are:

- C-k Delete to the end of line and append to buffer (cut)
- C-y Paste

Another way to cut and paste in Emacs is by marking a region, using C-w to cut it, and later C-y to paste. For example, you can change the word Turing to Byron in the second last line of the file and position the cursor as follows:

```
Alan Turing
Alan Turing
Ken Thompson
Alan Turing
Alan Byron
Ken Thompson
```

Now, enter C-@ command to "set mark" at this cursor position. At the bottom of the screen you should see a message "Mark set". Once a mark is set, all text from that position to the current position of the cursor is known as the current region in Emacs. Move the cursor to the letter 'B' of the same line, and you should see that the whole word 'Byron' is highlighted. Press C-w to delete the region and then C-y to paste it again. Now, delete Thompson at the last line as follows:

```
Alan Turing
Alan Turing
Ken Thompson
Alan Turing
Alan Byron
Ken
```

and press C-y. The word 'Byron' will be inserted. Make sure that you file ends with a new-line character, so the Emacs view should be:

```
Alan Turing
Alan Turing
Ken Thompson
Alan Turing
Alan Byron
Ken Byron
```

At this point, it is a good idea to save the file using C-x C-s hot keys. Generally, when you are editing a file, even though Emacs will usually make automatic backup saves, it is a good practice to manually save the file every so often, so that in a case of network or power outage, or a similar interruption you do not lose a lot of work.

The hot-key Emacs commands that we covered are:

- C-@ Set mark
- C-w Cut the region (delete and store in the cut-and-paste buffer)

Step 7: Emacs: Copy and paste. The copy-and-paste process is quite similar to cut-and-paste: we mark a region using C-@ and a cursor position, then copy using M-w, and then paste using C-y. Remember that M-w is obtained by pressing 'Alt' and 'w' keys on the keyboard, or by pressing 'Esc' and 'w' keys *one after another.*

To try copy-and-paste, go with the cursor to the letter 'B' of the word 'Byron' in the last line and press C-@. Move cursor to the right until you pass the letter 'n' at the end of the word. Now press M-w and you will copy the marked region. Go to the last line, type 'Ada', a space, C-y to paste 'Byron' and press Enter. Your file should look as follows:

```
Alan Turing
Alan Turing
Ken Thompson
Alan Turing
Alan Byron
Ken Byron
Ada Byron
```

Similarly as C- stands for Ctrl and another key in Emacs, M- stands for Alt and another key. The M- combination can also be obtained by pressing Esc, letting it go, and then pressing the other key in succession. This combination uses M- since there is sometimes a Meta key on some older keyboards.

Step 8: Emacs: Undo changes. The `emacs` hot key for undo is 'C-x u'. You can try typing something and reverting it using this hot key.

Save the file `names`, and exit Emacs.

SVN Submission: Submit the file `names` to the course SVN.

The Emacs hot keys that we covered in detail in the previous and this lab are:

C-x C-s	Save a file
C-x C-c	Exit Emacs
C-a	Move to the beginning of a line
C-e	Move to the end of a line
C-k	Delete to the end of line and append to buffer (cut)
C-y	Paste
C-@	Set mark
C-w	Cut the region (delete and store in the cut-and-paste buffer)
M-w	Copy the region
C-x u	Undo the last change

Step 9: Filters: uniq. A *filter* is a program that gets most of its data from the standard input (stdin) and writes its results to the standard output (stdout). The concept of filter is a part of an important software architectural pattern called 'pipe-filter'. By connecting filters into a pipe we create more complex filters.

In this lab, we will learn some filters. The first filter that we will learn here is the command `uniq`. It can be used to display a file with all of its identical **adjacent** lines replaced by a single occurrence of the repeated line.

We will start with the file 'names' created earlier in this lab.

Use the `wc` command to verify that this file has exactly 7 lines.

You can read about the command `wc` in its manual page ('`man wc`'). The command 'wc' stands for 'word count'. It prints the number of characters, words, and lines in a file. You can print only one of these numbers by choosing

an appropriate option: '`-c`', '`-w`', or '`-l`'. You can learn about these options by reading the man page.

The syntax of `uniq` is `uniq [inputfile [outputfile]]`. The square brackets mean that parameters are optional. When they are not present, `uniq` reads from `stdin` and writes to `stdout`. The nesting of brackets means that if you only supply one file name, it will be used as the input file.

Perform the following tasks:

9-a) Type `uniq names`. The output should contain 6 lines. Only the first line is not printed since it is followed by exactly the same line.
9-b) Type `uniq -c names`. This -c option causes each line of output to be preceded by the number of occurrences that were found.
9-c) Type `uniq -f 1 names`. The output should be four lines only. Use the Unix `man` command to find out why. This is equivalent to `uniq -1 names`. Read page 88 of the textbook to see what it means.
9-d) Save the outputs of the above three commands in the files `step9a.out` `step9b.out` and `steb9c.out` You can do it using the following commands:
```
uniq names > step9a.out
uniq -c names > step9b.out
uniq -f 1 names > step9c.out
```
This feature in which we use the special character '>' (greater than) to save the output of a command into a file is called the *standard output redirection.*

SVN submission: Submit these three files to the SVN repository.

Make sure that you understand what this command does before moving to the next question.

Step 10: Filters: sort. We have seen the command `sort` briefly in class before. Let's learn more about it.

This command sorts the lines of a file based on one or more sort fields (keys).

The basic usage is `sort [filename]`. This uses the entire line as the sort key.

10-a) Try this out by typing `sort names`. By default, this sorts all lines in lexicographic order (i.e., the order in which words appear in dictionaries).
This command can also be used to sort a file by a field in each line. In the example above, each line of the file has a set of words (fields) separated by white spaces. There can be multiple space characters between fields. The fields of each line are numbered 1, 2, ..., from left to right. The `-k` option can be used to sort by a particular field.
10-b) To sort the names in `names` by first names, type `sort -k 1 names`.
10-c) To sort by last names, try `sort -k 2 names`.
10-d) To sort by last name, and then by first name, type: `sort -k 2 -k 1 names`
10-e) `sort -k n,m names` can be used to sort using fields between (and including) fields n and m as sort keys. Try `sort -k 1,2 names`
10-f) In (d), when `,m` is absent, sort will use fields `n`, `n+1`, ..., until the end of the line as sort key. Thus, for file `names`, `sort -k 1 names` is equivalent to `sort -k 1,2 names` since this file has two fields. If it has more fields, and you would like to perform the task of step (d) (to sort by last name, and then by first name), you are expected to use: `sort -k 2,2 -k 1,1 names`
Try these commands and understand what they do.

Note that the textbook uses a different syntax for sorting with a particular filed. It is however obsolete, even though it still works on bluenose.

The `-r` option can be used to sort in reverse order. Try `sort -r -k 2 names`.

Save the outputs of the following commands as described above:
(a) `sort names`
(b) `sort -k 1 names`
(c) `sort -k 2 names`

(d) `sort -k 2 -k 1 names`
(e) `sort -k 1,2 names`
(f) `sort -k 2,2 -k 1,1 names`
(g) `sort -r -k 2 names`
into the files:
`step10a.out`, `step10b.out`, `step10c.out`, `step10d.out`, `step10e.out`, `step10f.out`, and `step10g.out` using the standard output redirection.

SVN Submission: Submit the above files to the SVN repository.

Step 11: Sort with CSV. Fields in a file can be separated by other characters other than spaces. For example, in a comma-separated values (CSV) file the data items are separated by a comma. Create a file named `names.csv` which contains the following 7 lines:

```
Alan,Turing
Alan,Turing
Ken,Thompson
Alan,Turing
Alan,Byron
Ken,Byron
Ada,Byron
```

Verify that this file has 7 lines.

To make the `sort` command work with the above file, the option `-t` can be used to specify the deliminator, i.e. the character that separates the fields in each line. The deliminator should be provided immediately after `-t` in the command line.

Try the command: `sort -t, -k 2 -k 1 names.csv`
What does this command do?

Save the output of the above command into the file '`step11.out`'. Using emacs, open the file, go the the end of it, add an empty line, and then a brief textual note explaining the output saved in this file.

SVN Submission: Submit the file `step11.out`, to the SVN repository.

Step 12: More about sort. We are not done with the command `sort` yet. Recall that this command sorts keys in lexicographic order. However, sometimes we wish to sort a key in numerical order, or even sort month names (say, Jan is before Dec). The -n and -M options are used for the above purposes.

Create a file named `holidays` whose content is (make sure that there is only one space character between any two consecutive fields in the same line; this is required to use the `cut` command in a later question):

```
Dec 26 Boxing Day
Dec 25 Christmas Day
Jan 1 New Year
```

Verify that this file has 3 lines.

Now let's sort the lines by date. This can be done by using the following command:

```
sort -k 1M -k 2n holidays
```

Try the command and read about the options. Save the output of the command in the file '`step12.out`'.

SVN Submission: Submit the file `step12.out` to the SVN repository.

Step 13: The command `cut`**.** The `cut` command can be used to extract fields from each line of the input.

The syntax is `cut -d deliminator -f fields [filename].`

Here deliminator is a single character (this can be put inside double quotes, especially if it is a space character). The fields part can be a list of field numbers separated by commas (this is different from `sort`, as commas here are not used to specify a range of field numbers).

Enter the command: `cut -d " " -f 2,4 holidays`
What does this command do?

(You can answer this question for yourself, to understand the command, but you do not need to record the answer.)

Save the output of the command in the file `step13.out`.

SVN Submission: Submit the file `step13.out` to the SVN repository.

Step 14: A pipe for practice. We will used the file `names` created earlier. Remember that the file has the following contents:

```
Alan Turing
Alan Turing
Ken Thompson
Alan Turing
Alan Byron
Ken Byron
Ada Byron
```

Develop a single command line that prints the number of distinct last names in the above file on the screen. The output should be 3 in this example. Once you get the command line ready, copy it into the file '`cmd14`'. **Note** that this file must contain the actual command line, not the output of the command.

In order to write a pipeline command (or 'pipe' for short) have in mind that this is done by connecting several commands with the pipe symbol '`|`'. This was mentioned in class. The first command should probably include the file name since it will be reading from the file and writing to the standard output. The next command will read this output as its input, so it does not need a file to specified. Each next command will also read output of the previous command and will not need a file name to be specified.

If you want to test the command that you wrote in the file, you can do it using the command:
`source cmd14`

SVN Submission: Submit the file `cmd14` to the SVN repository.

Step 15: The lab ends. By now, you have finished the required work of this lab. Make sure that all required files are submitted in SVN.