Lab 9.1: Navigating with vim

Scenario: You wish to familiarize yourself with **vim** by navigating a copy of a familiar document, `/etc/passwd`.

Instructions:

1. Log in as user `student` with the password *`student`*. If you are using the graphical environment, start a terminal by clicking Applications->Accessories->Terminal

2. Create your own copy of `/etc/passwd` in your home directory:

   ```
   [student@stationX ~]$ cp /etc/passwd ~
   ```

   Note that if you intended to actually modify this file you would have to be logged in as `root` and it would be considered best-practice to use the **vipw** command instead of running **vim** directly. **vipw** uses prevents other instances of **vipw** from opening the file at the same time. That way two administrators cannot accidentally overwrite each other's changes.

3. Open your copy of `passwd` in **vim**

   ```
   [student@stationX ~]$ vim ~/passwd
   ```

4. First, try moving around using standard keys such as the arrows, *PgUp*, *PgDn*, *Home* and *End*. They should all work as expected. Note, however, that on older systems they might not.

5. Now try moving from word to word with the **w** and **b** keys. Note that the keys for moving by sentence (the parentheses) and paragraph (the curly braces) simply move from one end of the file to the other. Why is this?

   ---

6. Try combining numbers with movement keys. For example:

   5w
   2*Down Arrow*

7. Next you will enter insert mode and make some changes. Remember that since you are working as a non-root user on a copy of the real `passwd` file, there is no danger of actually damaging the system.

   Press the **i** key. Note that the word INSERT appears at the bottom of your screen. Note that the arrow keys still move your cursor, but the other keys now modify the document, as they would in an ordinary text editor.

8. Once you have changed some text, exit insert mode by pressing ***Esc***. Note that the Insert disappears from the bottom of your screen.

9. Try pressing the **u** key. This will undo each change that you made. Changes can be re-done with *Ctrl-r*

10. Now change **vim**'s behavior by altering some settings. To do this, you will need to enter ex mode. Remember that ex mode is invoked by typing `:` while in command mode. Do this now.

11. At the ex prompt (:) enter `set nu`. This is short for `set number`, which turns on **vim**'s line numbering feature. You should now see line numbers displayed on the left side of your screen.

12. Try entering ex mode again and disabling line numbering by typing `:set nonu`. Repeat the previous command to turn it back on.

13. Try jumping to line 5 by typing `5G`. You can jump to the end of the file by typing `G` by its self. How might you jump to the first line of the file?

    ---

14. Try jumping to the first instance of the word root by using **vim**'s search feature: `/root`. Note that every instance of root is now highlighted. What happens when you press the `n` key? What about `N`?

    ---

15. Next, we will introduce some more options relevant to the search feature. Suppose you do not want search results to be highlighted. Try typing `:set nohls`. What happens to the highlighting? How could you re-enable this feature? To turn off the current highlights (but leave highlighting on for the next search), use `:nohl`

    ---

16. It is time to finish up and quit **vim**. Try running `:q`. This will fail. Note the error message. How could you change the command so that **vim** is forced to quit, abandoning changes? Once you have figured this out, run the command.

Lab 9.2: Configuring vim

Scenario: In this sequence, you will create a `~/.vimrc` file to practice document creation with **vim**. This file is also used to set your preferred settings for **vim**.

Instructions:

1. Start by creating a new file:

   ```
   [student@stationX ~]$ vim ~/.vimrc
   ```

2. Remember that before you can type anything, you will need to enter insert mode. To do this, press:

   ```
   i
   ```

   You can now type as you would in an ordinary text editor. The next exercise will explore some of what you can do in command mode.

3. Type the following as the first line in your document:

   ```
   :set nu
   ```

   This will cause line numbering to be turned on by default.

4. Press ***Enter*** to start a new line and type the following:

   ```
   :set wrapmargin=10
   ```

   This will cause lines to automatically "wrap" when they come within 10 characters of the edge of **vim**'s screen, instead of being treated like one long line, which is **vim**'s default behavior.

5. If there are any other settings you have decided that you prefer, add them now.

6. Exit insert mode by pressing ***Esc***, then enter ex mode and instruct **vim** to write (save) the file and quit:

   ```
   :wq
   ```

7. Open `~/passwd` again. Observe that line numbering, along with the other settings you added to `~/.vimrc` are now automatically enabled. When you are finished exploring, quit **vim**.

Lab 9.3: Configuring basic sudo privileges with vim

Scenario: So far, you have been using **su** to become root. Hopefully, you have not been logging in directly as root! This lab has you make a simple modification to **sudo** to allow `student` to execute individual commands as `root`.

Instructions:

1. To make the initial modification, you will first need to become `root`.

   ```
   [student@stationX ~]$ su -
   ```

2. **sudo** gets its configuration information from the file `/etc/sudoers`. However, *this file should never be modified directly*. Instead, the command **visudo** opens the file in **vim** and then runs a syntax checker to catch any mistakes you make before they can cause problems. Try running **visudo** now.

   ```
   [root@stationX ~]# visudo
   ```

   You should find yourself looking at the contents of `/etc/sudoers` in **vim**.

3. While details of this file are outside the scope of RH033 (see `RH133: Red Hat Enterprise Linux System Administration` and/or **man sudoers** for more), for this exercise you will need to understand one line, which you will then use **vim** to copy and modify.

   The line appears about halfway down the file and looks like this:

   ```
   root    ALL=(ALL) ALL
   ```

   Try reaching it efficiently using **vim**'s search feature. Type:

   ```
   /root
   ```

   This will take you to the first instance of the word `root`, which is probably not the one you were looking for. Repeat the search by pressing **n** until your cursor lands on the correct line.

4. Simply put, this line says:

   "Allow root, on any system using this `sudoers` file, to run any command as any user"

   You want to create a line that says:

   "Allow student, only on the system called station*X*, to run any command as any user"

   The distinction between systems can be important since under some circumstances a single `sudoers` file may be shared between multiple systems.

   Start by creating a copy of the current line. In **vim** type:

   `yyp`

   `yy` "yanks" (copies) the current line and **`p`** "pastes" it one line down, taking your cursor with it.

   Your cursor should now be on the second of two identical lines.

5. Next, replace the first word root with student. Start by typing:

   `cw`

   The word root should disappear and you should find yourself in insert mode.

   Type **`student`** and exit insert mode by pressing *Esc*.

6. Now you can move to the next word, ALL and change it to station*X*, where *X* is your station number.

   Move to the next word by pressing

   `w`

   and then prepare to replace it by typing

   `cw`

   Type **`station`*`X`***

   The line should now read:

   `student stationX=(ALL) ALL`

7. Exit insert mode by pressing *Esc*.

8. You are done! Time to save your changes and quit.

```
:wq
```

If you have made any mistakes, **visudo** will catch them and give you the option to go back and correct them. Work until you have a syntactically correct file. Ask your instructor if you need assistance.

To review, the complete set of keystrokes needed to find the line you were looking for, duplicate it and make your changes was:

```
/root
nnn
yyp
cwstudentEsc
w
cwstationxEsc
```

You could have gone straight into insert mode (or used a simpler editor), but the same task would have taken a lot more work!

9. You can now elevate your privileges on a per-command basis. Leave your root shell by running **exit** and attempt to access a restricted directory once as your non-root user and once using **sudo**. When you are prompted for a password, enter `student`'s password, not `root`'s. This is just to prove that you are the legitimate owner of the account using **sudo**.

```
[student@stationX ~]$ ls /etc/pki/CA
ls: /etc/pki/CA: Permission denied
[student@stationX ~]$ sudo ls /etc/pki/CA
Password:
private
```

10. Try running the **sudo** command again. Note that you are not prompted for a password. **sudo** will remember that you have already authenticated for five minutes before prompting you again.