csce215 — UNIX/Linux Fundamentals
Spring 2022 — Lecture Notes: Making Changes

This document contains slides from the lecture, formatted to be suitable for printing or individual reading, and with some supplemental explanations added. It is intended as a supplement to, rather than a replacement for, the lectures themselves — you should not expect the notes to be self-contained or complete on their own.

(2.1) *Last time*

Assignment 1 grades will likely be posted tomorrow, January 25.

Last time we learned some commands for **looking around** at files and directories:

- `ls`
- `cd`
- `cat`
- `less`

And we saw some special ways to refer to certain directories:

`..`	parent directory
`.`	current directory
`~`	home directory
`/`	root directory

Today, we will learn some commands for **creating**, **deleting**, **modifying** things.

(2.2) *Editing files*

We'll want to be able to **create new files** and **edit existing files**.

`vim`
Edit a file using 'Vi IMproved', a programmer's text editor.

Vim has been around since 1991, and is an expanded clone of the `vi` ('vee eye') editor, which was first released in 1976. It continues to be updated and expanded to this day.

(2.3) *Why vim?*

Vim is quite different from other 'normal' editors, and takes some practice to learn.

Two main reasons learning `vim` is worthwhile:

- It's available **everywhere**.
- It can be **extremely powerful** with some practice.
- It can be **customized** to match your own style.

(2.4) *Two modes*

Vim has two primary **modes**:

- **Normal mode**, you give commands: move around, copy/paste, search, save, quit, etc.
- **Insert mode**, you type text into the document.

Changing between modes is easy.

- From normal mode, press **i** to get to insert mode.
- From insert mode, press **Escape** to get back to normal mode.

(2.5) *Vim normal mode: Essentials*

Things to do in vim's normal mode:

arrow keys	move around
i	insert before the current character
dd	cut the current line
yy	copy the current line
p	paste
u	undo
Ctrl-R	redo
nG	go to line n

Certain vim commands start with a colon:

:w	save ('write')
:q	quit

(Press Enter to finish these.)

(2.6) *Vim normal mode: A little more*

A few more normal mode commands:

h, j, k, l	move around like a pro
x	cut the current character
I	insert at the start of the current line
A	insert at the end of the line
G	go to the end of the file

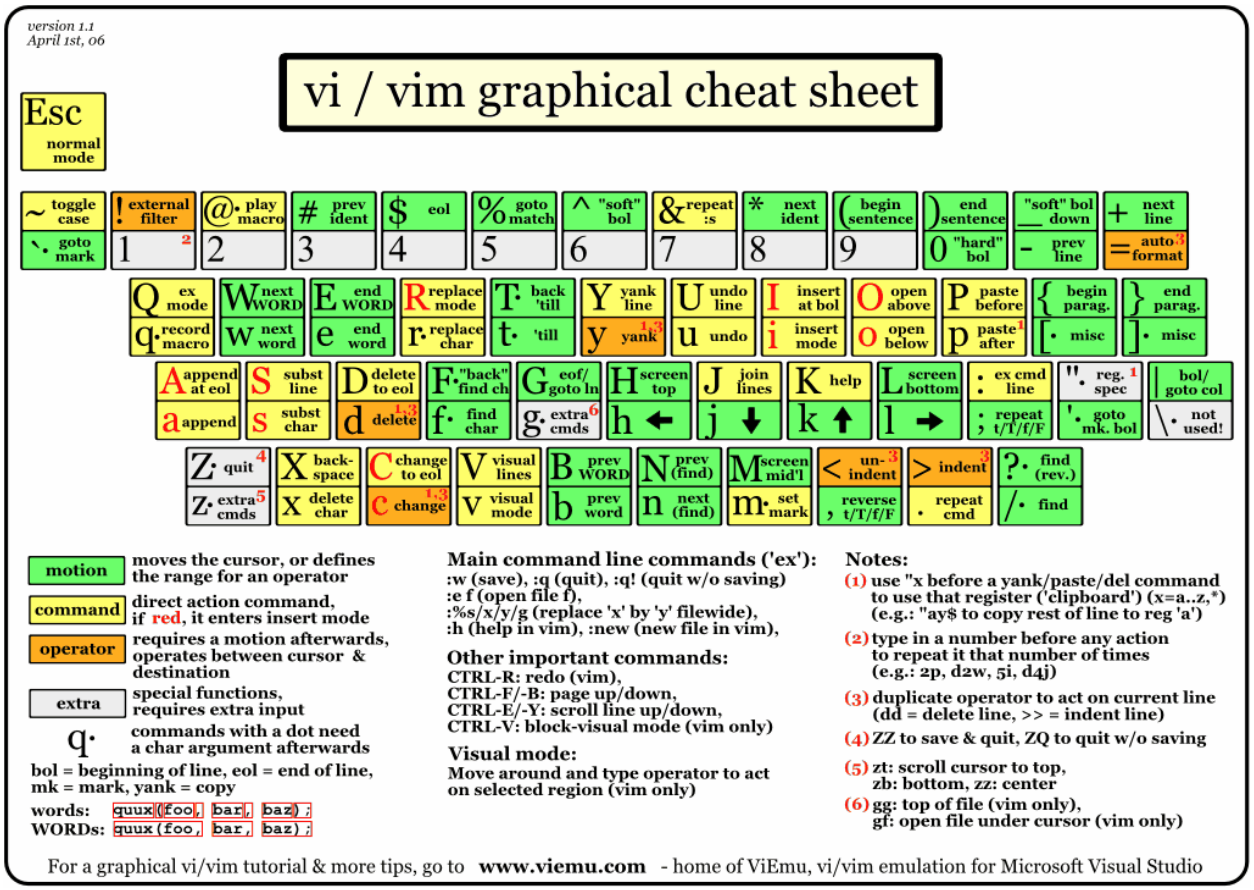
And with colons:

:wq	save and quit
:q!	quit without saving

(Press Enter to finish these.)

(2.7) *Going deeper*

Vim 'cheat sheet':



http://www.viemu.com/vi-vim-cheat-sheet.gif

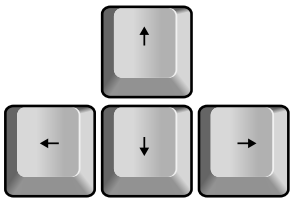
https://vim-adventures.com/

(2.8) *Some shell keys*

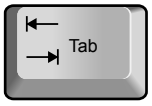
Moving on from `vim`, back to the shell.

A few keys are extremely helpful:

- Use the **left** and **right** arrow keys to edit the current command.
- Use the **up** and **down** arrow keys to go back to previous commands.



- Use the **tab** key for auto-complete.



- Use **Ctrl-C** to (try to) kill a command.

(2.9) *Sending command output to a file*

We can **redirect** output from a command into a file instead of our terminal window.

> and >>

Send the standard output of a command to a file.

> If the file exists, replace it.

>> If the file exists, add to the end.

Normal redirection:

```
$ date
Thu 03 Mar 2022 09:31:25 AM EST
$ date > log
$ cat log
Thu 03 Mar 2022 09:31:28 AM EST
$ date > log
$ cat log
Thu 03 Mar 2022 09:31:31 AM EST
```

Appending redirection:

```
$ date >> log
$ cat log
Thu 03 Mar 2022 09:31:31 AM EST
Thu 03 Mar 2022 09:31:46 AM EST
$ date >> log
$ cat log
Thu 03 Mar 2022 09:31:31 AM EST
Thu 03 Mar 2022 09:31:46 AM EST
Thu 03 Mar 2022 09:31:52 AM EST
```

(2.10) *Why redirection is cool*

Key idea: This output redirection works to capture the standard output of

EVERY SINGLE LINUX PROGRAM EVER.

For example, a program that you might have written:

```
$ javac Hello.java
$ java Hello
hello, world
$ java Hello > output.txt
$ cat output.txt
hello, world
```

(2.11) *Showing output*

echo

Print things on standard output.

```
$ echo hello, world
hello, world
```

```
$ echo Leonardo > tmnt.txt
$ echo Donatello >> tmnt.txt
$ echo Raphael >> tmnt.txt
$ echo Michelangelo >> tmnt.txt
$ cat tmnt.txt
Leonardo
Donatello
Raphael
Michelangelo
```

(2.12) *Creating directories*

Directories can help us keep files organized.

mkdir

Create a new directory.

```
$ ls
log
tmnt.txt
$ mkdir stuff
$ ls
log
stuff
tmnt.txt
```

(2.13) *Copying files*

We can make a copies of files.

cp

Copy files.

Copying one file in the same directory:

```
$ ls
log
stuff
tmnt.txt
$ cp tmnt.txt copy.txt
$ ls
copy.txt
log
stuff
tmnt.txt
```

Copying one file to another directory.

```
$ ls stuff
$ cp tmnt.txt stuff
$ ls stuff
tmnt.txt
```

Copying multiple files to another directory.

```
$ ls stuff
tmnt.txt
$ cp tmnt.txt copy.txt stuff
$ ls stuff
copy.txt
tmnt.txt
```

Remember: The last argument to cp is the destination; everything before that is a list of what to copy.

(2.14) *Copying entire directories*

cp
Copy files.
-r copy *recursively*, i.e. copy any subdirectories and their contents
-v show details about what is being copied

```
$ ls
copy.txt
log
stuff
tmnt.txt
$ cp -rv stuff more_stuff
'stuff' -> 'more_stuff'
'stuff/tmnt.txt' -> 'more_stuff/tmnt.txt'
'stuff/copy.txt' -> 'more_stuff/copy.txt'
$ ls
copy.txt
log
more_stuff
stuff
tmnt.txt
```

(2.15) *Moving and renaming*

Use `mv` to move and/or rename things.

`mv`
Move files or directories.

Renaming a file:

```
$ mv copy.txt tmnt2.txt
$ ls
log
more_stuff
stuff
tmnt2.txt
tmnt.txt
```

Moving to a new directory:

```
$ mv log stuff
$ ls
more_stuff
stuff
tmnt2.txt
tmnt.txt
$ ls stuff
copy.txt
log
tmnt.txt
```

(2.16) *Deleting files*

`rm`
Delete ('remove') files.

Be careful! There is no 'un-delete'!
No 'trash can'! No 'recycle bin'!

Deleting a single file:

```
$ ls
more_stuff
stuff
tmnt2.txt
tmnt.txt
$ rm tmnt2.txt
$ ls
more_stuff
stuff
tmnt.txt
```

Deleting multiple files:

```
$ ls more_stuff
copy.txt
tmnt.txt
$ rm more_stuff/*.txt
$ ls more_stuff
```

(2.17) *Deleting empty directories*

`rmdir`

Delete a directory, but only if it's empty.

```
$ rmdir stuff
rmdir: failed to remove 'stuff': Directory not empty
$ rmdir more_stuff
```

(2.18) *Deleting entire directories*

`rm`

Delete ('remove') files.

`-r` delete *recursively*, i.e. remove any subdirectories and their contents

`-v` show details about what is being deleted

Be careful! A single `rm -r` can delete a **lot** of files!

```
$ ls
stuff
tmnt.txt
$ rm -rv stuff
removed 'stuff/copy.txt'
removed 'stuff/tmnt.txt'
removed 'stuff/log'
removed directory 'stuff'
$ ls
tmnt.txt
```

(2.19) *Sample final exam questions*

1. In `vim`, to go from insert mode to normal mode, press the _____ key.

   A. Control
   B. Escape
   C. Tab
   D. Shift

2. In `vim`, which command is used to save the file?

   A. `:w`
   B. `:x`
   C. `:s`
   D. `:q`

3. In `vim`, which command is used to copy the current line?

   A. `Ctrl-R`
   B. `yy`
   C. `Ctrl-C`
   D. `u`

4. In `vim`, which command is used to go to the 15th line?

   A. `15G`
   B. `15g`
   C. `15l`
   D. `15L`

5. Suppose the current directory contains files called `foo` and `bar` along with a subdirectory called `quux`. What effect would the command `cp bar quux` have?

   A. It would copy the file `bar` into the subdirectory `quux`.
   B. It would copy the each file in `quux` to the current directory.
   C. It would move the file `bar` into the subdirectory `quux`.
   D. It would generate an error, because `quux` already exists.

6. In `vim`, to go from normal mode to insert mode, press the _____ key.

   A. `j`
   B. `k`
   C. `i`
   D. `h`

7. Which of these commands will rename a file from `old.txt` to `new.txt`?

   A. `ren old.txt new.txt`
   B. `cp old.txt new.txt`
   C. `mv old.txt new.txt`
   D. `mv new.txt old.txt`

8. Which option can be used with the `cp` command to copy any subdirectories and their contents?

   A. `-H`
   B. `-r`
   C. `-v`
   D. `-i`

9. Which of these commands will *not* remove an empty directory called `delete_me`?

A. `rm -r delete_me`

B. `rmdir delete_me`

C. `rm -v delete_me`

D. `rm -rv delete_me`

10. Which of these commands will create a directory called `tasty`?

A. `mkdir tasty`

B. `redir tasty`

C. `less tasty`

D. `vim tasty`

11. The command whose purpose is to edit files is _____.

A. `vim`

B. `less`

C. `man`

D. `cat`