Text Editors

Handout 1
COP 3363 Spring 2024

2024-01-31

Text Editors

- Files in Unix are usually one of 2 types:
  1. Text Files - when opened, they are intelligible for humans, that is, we can read and understand their contents.
  2. Binary Files - These files are usually executable are not in machine-readable form, so they might not be human readable.
- Special files like directories are intelligible by both humans and computers due to some special formatting. They are the exception, not the rule.
- Irrespective of the file type, Unix files can be opened by a text editor. Text editors are special programs that can work with files, especially text files.
- For this course, we prefer the vim text editor, which should come with your shell account.
- We will aslo look at some other text editors like vi and emacs.
- Some other programs, usually mail composers can also be used for text editing - like nano or pico, These are not really text editors, as they do not offer the full functionality or power of vim or emacs.

the cat command

- The `cat` command is a way to quickly look at the contents of a file.
- Syntax: `cat filename`
- This is just to **display** the contents of the file on the terminal, and cannot be used to **edit** the file.
- For file editing, we would have to use one of the many text editors.

The vim editor

We will be using the vim text editor for the course.
This is a quick way to get started.

- **Creating a file**: type `vim filename`
  This will create a file if it doesn't exists and open the empty file in vim.
- **Opening an existing file**: Type `vim filename`
  This will open an existing file in vim with all of it's previous contents.
- vim has 2 modes:
  - `command mode`: in this mode characters you type are interpreted as commands
  - `insert mode`: characters you type are inserted as part of the text
- vim starts in command mode.
- Typing "i" switches to insert mode.
- The ESC key gets you back to command mode.
- vim is case sensitive in command mode. Uppercase and lowercase commands do different things.

Some basic vim settings

vim settings are written into the `vimrc` file.

- To open the file, **In Your Home Folder** open the file by typing "vim ~/.vimrc" and Enter.
- In that file, please make sure you have the following exactly:

```
set expandtab
set tabstop=4
set softtabstop=4
set shiftwidth=4
set textwidth=80
syntax on
set wrap
set laststatus=2
set showmode
set showcmd
set number
set matchpairs+=<:>
```

- Save and Quit vim.

Starting up vim

- Open a file with vim - `vim filename`
  - If the file doesn't exist, it will be created.
  - If the file exists, vim will open the file and display the current contents.
- Open a file with the cursor at the beginning of line *n*: `vim +n filename`
- Open a file with the cursor at the last line of the file: `vim + filename`
- Recover a file after a system crash: `vim -r filename`
- The file will be opened in a **buffer**. This is like a working copy of the file.

vim - Saving Files

vim commands are like a language - most commands will read like a shorthand of English.

- To save a file, we need to **Write** to a buffer
  :w
  This will overwrite the exiting file.
- To save the file as another file (Save-As)
  :w newname
  The old file will be closed, unchanged and the new file will now be open in the buffer.
- To save the current file over an existing file
  :w! otherfile
  The exisitng file will be deleted.
- Save the current file and open another file for **editing**:=
  :e file
  The current file will be saved and closed, and the new file will be opened in the editor.

Quitting vim

- To quit vim
  :q
- Save the current file and quit
  :wq
- To quit without saving
  :q!
- Save the file if it has changed and quit
  :x

Moving the cursor in vim

- Left one character - h
- Right one character - l
- Up one line - k
- Down one line - j
- Left one word - b
- right one word - w
- Start of current sentence - (
- End of current sentence -)
- Start of current paragraph - {
- End of current paragraph - }

More Navigation

- Top of the file - 1G
- Line n of the file - nG
- End of the file: G
- First character of insertion: <Ctrl>W
- Up half a screen - <Ctrl>U
- Down half a screen - <Ctrl>D
- Up one screen - <Ctrl>B
- Down one screen - <Ctrl>F

Entering insert mode

- Insert after cursor - a
- Insert after the last character on the line - A
- Insert before the cursor - i
- Insert before the first character on the line - I
- Open a line below the current line - o
- Open a line above the current line - O

vim - changing and replacing text

- Change or replace current word - cw
- Change or replace k words at the cursor - kcw (eg. 3cw)
- Change or replace current line - cc
- Change or replace n lines around the cursor - ncc (eg. 6cc)
- Replace current character only - r
- Replace current character and those to its right - R
- Another way to replace current character - s
- Change or replace current line (another way) - S
- Switch cases (lower and upper case, like Caps Lock) - $\sim$

vim - deleting text

- Delete character under the cursor - x
- Delete n characters - nx (eg. 12x)
- Delete character to the left of the cursor - X
- Delete the current word - dw
- Delete k words - kdw (eg. 7dw)
- Delete current line - dd
- Delete from cursor tot he beginning of the line - d0
- Delete from cursor to the end of the line - d$
- Delete n line - ndd (eg. 8dd)
- Delete to the beginning of the paragraph - d{
- Delete to the end of the paragraph - d}
- Delete from cursor to the beginning of the line - :1,.d
- Delete from the cursor to the end of the file - :.,$d
- Delete entire file contents - :1,$d

vim - Searching

- Find the next occurrence of “word” - /word
- find the previous occurrence of “word” - ?word
- Find the next line that starts with “word” - /^word
- Find the next line that ends with “word” - /word$
- find the next occurrence of “word” or “Word”: /[wW]ord
- Repeat the most recent search in the same direction - n
- Repeat the most recent search in the opposite direction - N

emacs

- emacs is another powerful text editor. It can be used instead of vim, though it has been getting somewhat heavy, and thus falling out of favor, of late.
- Using vim or emacs is a matter of preference. In this course, we will show you a short introduction to emacs.
- If you need a longer user-guide, please contact the instructor.
- You might need to enable tunnelling on your shell account to access emacs.

Starting emacs

- The command “emacs” will start the “emacs” text editor in “scratch” mode, with an empty buffer
- “Scratch” mode is a pain to use, will not warn you about saving your work, and will cause various other grief
- Specifying a file name will have “emacs” open that file (or start a new file).
- `emacs` - will open a nameless buffer - (avoid this)
- `emacs <filename>` - preferred

emacs - basic commands

- Arrow keys are used to navigate around document
- If configured, the mouse can work, but you will learn to work without it
- The caret symbol ($\wedge$) indicates you must press and hold the control key first, then press the key for the command.
- **Undo** - $\wedge$x u or $\wedge$- will undo the most recent command (one of the only places in UNIX where you can undo something)
- **Saving** - $\wedge$x $\wedge$s saves the buffered text to the currently specified file
- **Quit** - $\wedge$x $\wedge$c exits emacs

Some other file related commands

- wc- word count
  - Counts the characters, lines, or words in a file
  - Syntax: `wc [options] filename`
- more
  - Simple text viewer for large files. Page through with space bar
  - Syntax: `more filename`
- less
  - Better text viewer than `more`. Move with up/down arrows.
  - Exit with "Shift - z - z "
  - Syntax: `less filename`

Running c++ programs on Unix

Follow these steps to compile and run a C++ program on Unix.

1. Write your program using a text editor of your choice.
2. Make sure your file is saved with a ".cpp" extension
3. **To compile** `g++ -c program.cpp`
   This will compile your file and generate an object file - a file with a .o extension. If nothing is printed, compilation was successful. Otherwise, errors will be displayed.
4. **To link into an executable** `g++ -o executableName program.o`
   Here, program.o is from the previous step.
   If there are errors, they will be displayed. Otherwise, the executable was built successfully.
5. **To run** `./executableName`
   Here, executableName is the file from the previous step.

Running c++ programs on Unix

- There is a **shortcut for compiling and linking**.
- We can combine the compiling and linking steps into one
  `g++ -o` *`executableName`* `program.cpp`
  This will not produce the intermediary object file, but the rest of the steps are the same.
- **Shortcut to the Shortcut**: We can skip the executable name.
  `g++ program.cpp`
  This will produce an executable under the default name `a.out`.
  If there was any precious a.out file in the same directory, it would be deleted.