

CIS241

System-Level Programming and Utilities

Viewing and Editing Files // vim and nano

Erik Fredericks, frederer@gvsu.edu

Fall 2025

Based on material provided by Erin Carrier, Austin Ferguson, and Katherine Bowers



First...

We will need some files to play with!

In your home directory

- `cd ~`

Download the following files:

- `wget` is a handy tool for downloading files, with lots of options
- <https://github.com/cis241-gvsu/w23-classmaterial/tree/master/misc-files/week01>

Click each of the CSV files in GitHub, then click 'Raw', copy the URL, and then use `wget` to pull down both files:

- `wget <URL>`

If you type `ls` afterwards you should see the two CSV files!

A man with dark hair and a light-colored shirt is shown from the chest up. He has a wide-eyed, open-mouthed expression of shock or surprise, with his hands pressed against his cheeks. The background is a solid dark grey. Overlaid on the image is the text 'How do we view files *in the terminal*?'.

How do we view files *in the terminal*?

Viewing commands

cat	list entire file in terminal
less	interactively view file
more	interactively view file
head	print first N lines ($N = 10$ by default)
tail	print last N lines ($N = 10$ by default)
uniq	removes adjacent repeated lines
sort	prints file with lines sorted

Oops the file was too big and `cat` doesn't work well

Can pipe! We'll get into that more later, but

```
cat superbigfile | more
```

Sends output of the first command into the input of the second!



How do we edit files *in the terminal?*

We'll need some form of editor

Depending on your Linux distribution, some may not come pre-installed!

If you don't have vim locally, and you're using a Debian-based system, that will be

```
sudo apt install vim
```

That brings us to a good point!

You should update your system often (unless if you're using Arch, then that may be different...)

- ```
sudo apt update && sudo apt upgrade
```

  - You can't do this on EOS? Why?

# OK then editors now

**We'll use them for a lot! Writing C programs, bash scripts, etc.**

- Similar to Notepad / Notepad++
- Use whichever editor you're most comfortable with!
  - I pretty much exclusively use vim

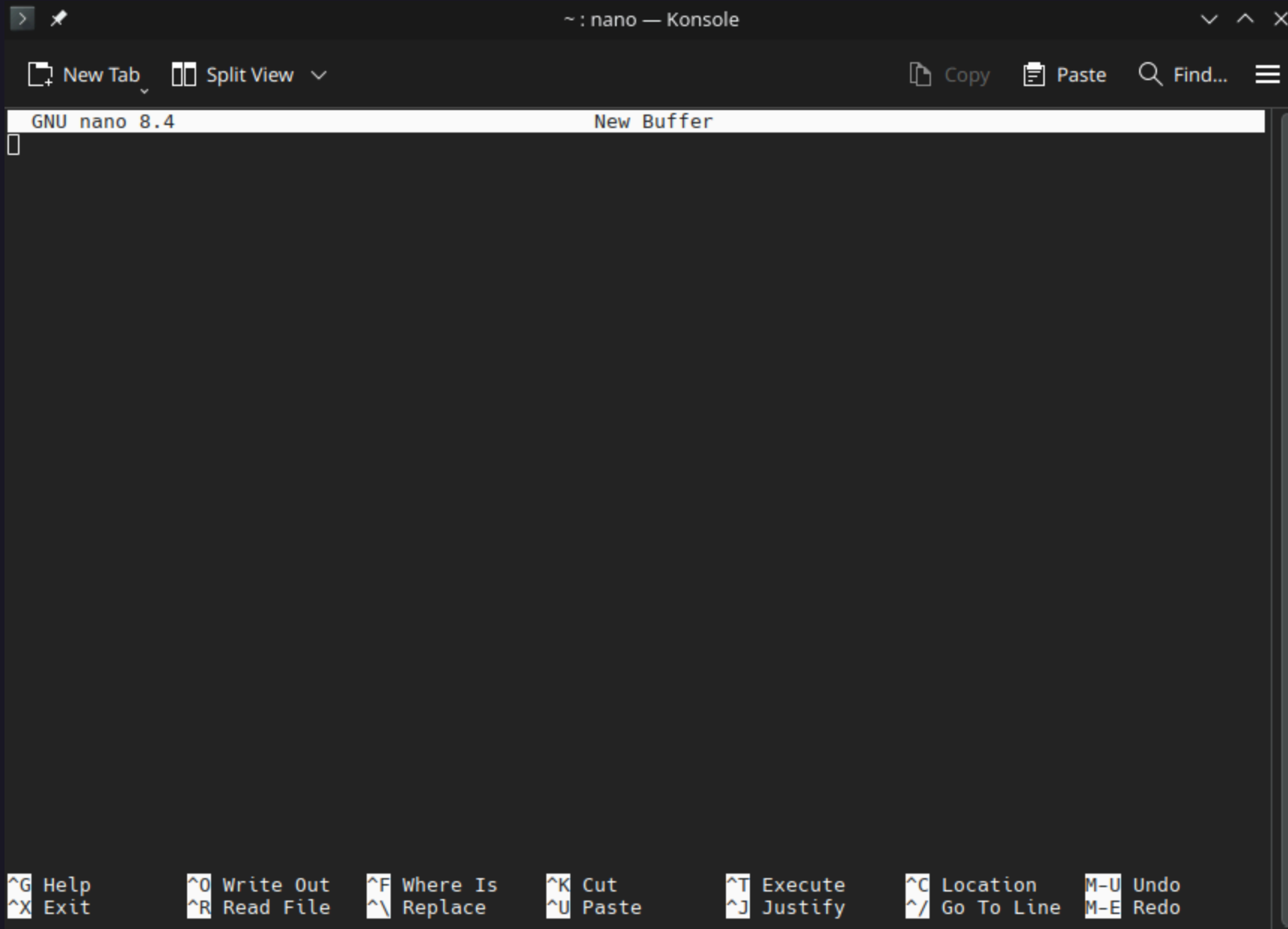
**Many are out there!**

- Common ones: nano, vim, emacs, gedit
  - gedit - very easy to use
  - nano - easier to use
  - vim - very keystroke-oriented
  - emacs - very shortcut-oriented

**Holy wars:** [https://en.wikipedia.org/wiki/Editor\\_war](https://en.wikipedia.org/wiki/Editor_war)



# nano



The screenshot shows the nano text editor running in a terminal window titled "~ : nano — Konsole". The interface includes a top toolbar with icons for "New Tab", "Split View", "Copy", "Paste", "Find...", and a menu icon. Below the toolbar, the text "GNU nano 8.4" and "New Buffer" are displayed. The main editing area is empty, with a cursor at the top left. At the bottom, a status bar lists various keyboard shortcuts: ^G Help, ^O Write Out, ^F Where Is, ^K Cut, ^T Execute, ^C Location, M-U Undo, ^X Exit, ^R Read File, ^\ Replace, ^U Paste, ^J Justify, ^/ Go To Line, and M-E Redo.

```
> ✂ ~ : nano — Konsole
New Tab Split View Copy Paste Find...
GNU nano 8.4 New Buffer
[]

^G Help ^O Write Out ^F Where Is ^K Cut ^T Execute ^C Location M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo
```

# nano shortcuts

Write normally and use shortcuts!

| Shortcut | Meaning                                            |
|----------|----------------------------------------------------|
| Ctrl + G | Display this help text                             |
| Ctrl + X | Close the current file buffer / Exit from nano     |
| Ctrl + O | Write the current file to disk                     |
| Ctrl + J | Justify the current paragraph                      |
| Ctrl + R | Insert another file into the current one           |
| Ctrl + W | Search for a string or a regular expression        |
| Ctrl + Y | Move to the previous screen                        |
| Ctrl + V | Move to the next screen                            |
| Ctrl + K | Cut the current line and store it in the cutbuffer |
| Ctrl + U | Uncut from the cutbuffer into the current line     |
| Ctrl + C | Display the position of the cursor                 |
| Ctrl + T | Invoke the spell checker, if available             |

# Try it out!

`nano` brings up an empty file and prompts you to save

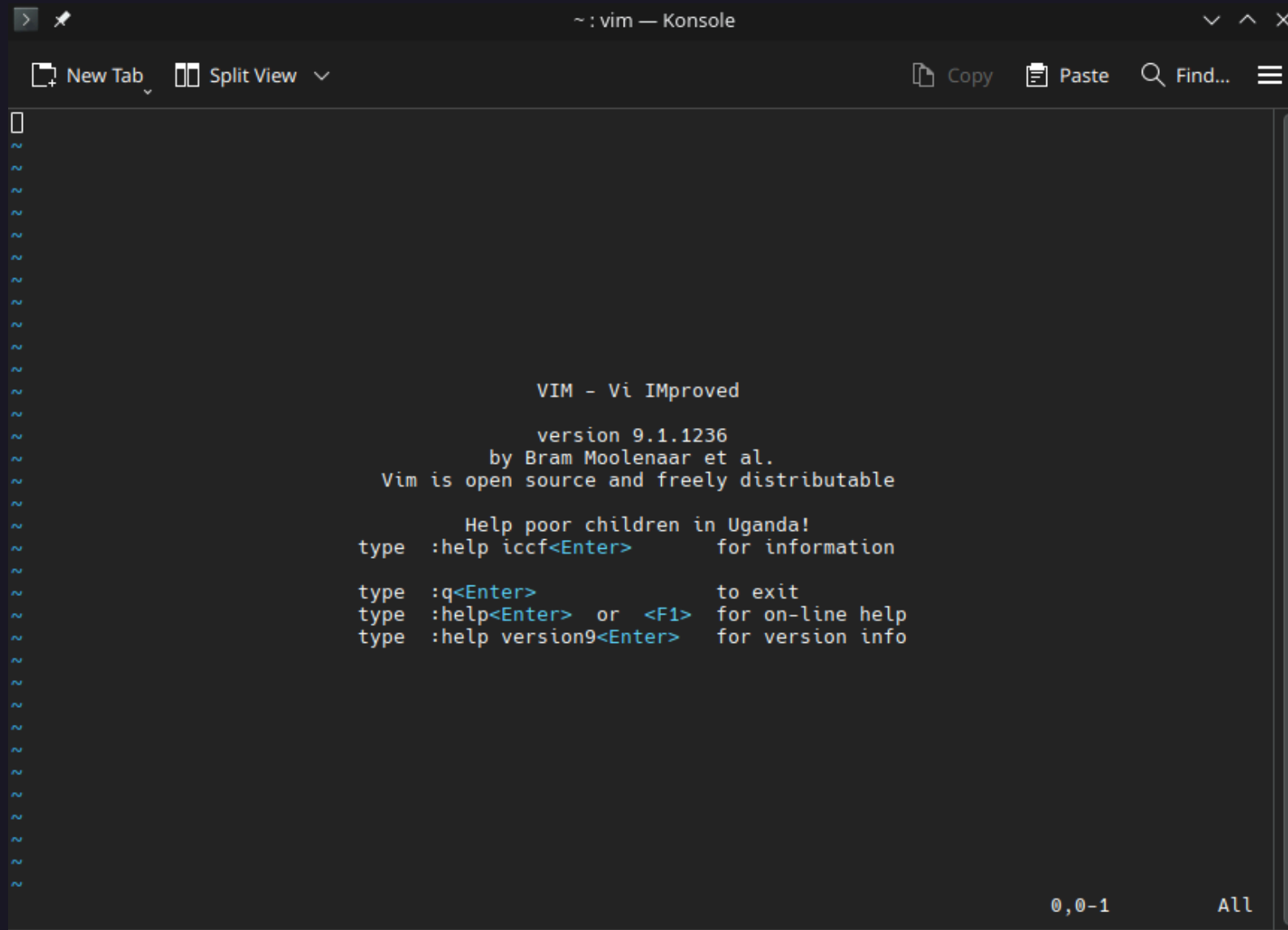
`nano filename` creates (or opens) a file named `filename`

# A permissions concern!

What happens if you try to work on a file to a protected area? (Note - this will apply to vim too)

- All your hard work will disappear
- Save your buffer locally (`~/mynewfile`)

# vim



The screenshot shows the Vim editor interface. At the top, there is a title bar with a window icon, a close button, and the text '~: vim — Konsole'. Below the title bar is a menu bar with 'New Tab', 'Split View', 'Copy', 'Paste', 'Find...', and a hamburger menu icon. The main editing area displays the Vim startup screen. On the left side of the editing area, there is a vertical list of tilde (~) characters representing line numbers. The startup screen text is as follows:

```
VIM - Vi IMproved

 version 9.1.1236
 by Bram Moolenaar et al.
Vim is open source and freely distributable

 Help poor children in Uganda!
type :help iccf<Enter> for information

type :q<Enter> to exit
type :help<Enter> or <F1> for on-line help
type :help version9<Enter> for version info
```

At the bottom right of the editing area, the status line shows '0,0-1' and 'All'.

# vim (necessities)

## Exit vim:

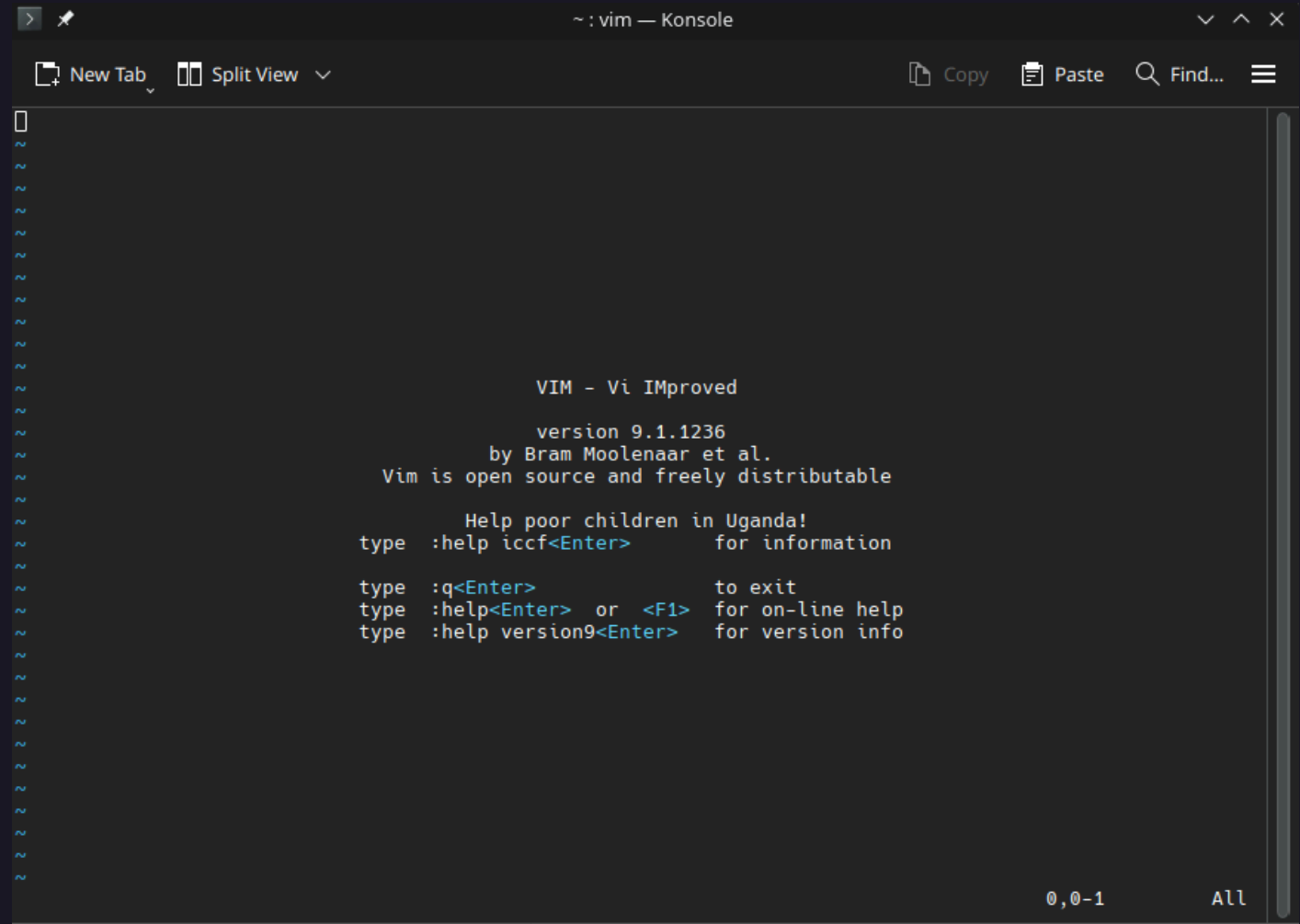
- `:q <enter>`

## Save:

- `:w <enter>`
- `:wq <enter>` (save and quit)

## Quit without saving

- `:q!`



The screenshot shows the Vim editor interface in a terminal window. The title bar reads '~ : vim — Konsole'. The interface includes a top menu bar with 'New Tab', 'Split View', 'Copy', 'Paste', 'Find...', and a hamburger menu. The main area displays the Vim startup screen with the following text:

```
VIM - Vi IMproved
 version 9.1.1236
 by Bram Moolenaar et al.
Vim is open source and freely distributable

 Help poor children in Uganda!
type :help iccf<Enter> for information

type :q<Enter> to exit
type :help<Enter> or <F1> for on-line help
type :help version9<Enter> for version info
```

At the bottom right, the status line shows '0,0-1' and 'All'.

# vim usage

Different than nano - you start in command (i.e., normal) mode

- You have to 'enter' editing mode
  - **Great** for power users
  - **Annoying** for new users

# vim usage

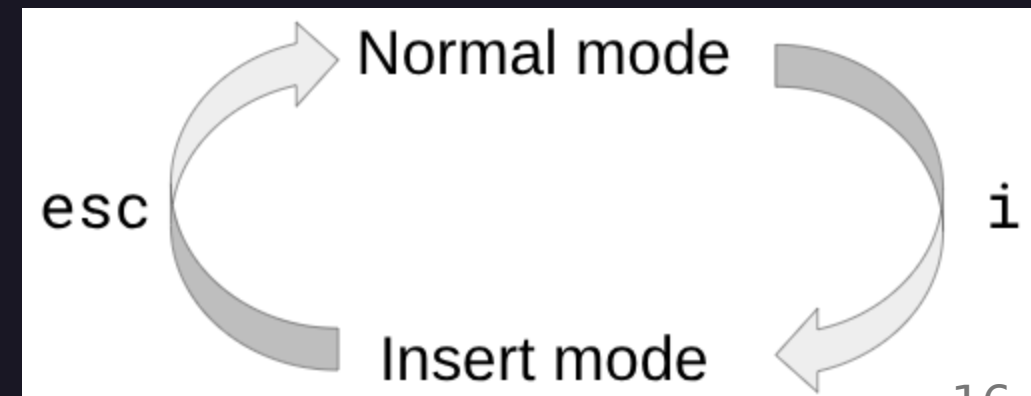
vim has three modes of operation: normal, insert, and visual

By default, vim starts in normal mode. It doesn't let you insert text, but rather interprets keystrokes as commands.

Insert mode will allow you to modify the file

Visual mode allows you to highlight text in the file

- To switch to insert mode, type `i`
- To switch to visual mode, type `v`
- To switch to normal mode, type `Esc`





# vim usage - moving cursor

Can use standard arrow keys

As a shortcut, can use letters as arrow keys (in normal mode):

- h: ←, j: ↓, k: ↑, l: →

More shortcuts:

- `PageDown` (or `Ctrl+F`) to move forward one screen of data
- `PageUp` (or `Ctrl+B`) to move backward one screen of data
- `G` to move to the last line in the buffer
- `num G` to move to the line number num in the buffer
- `gg` to move to the first line in the buffer

# vim shortcuts

To get to command line mode, press `:` key in normal mode

More shortcuts:

- `q` to quit if no changes have been made to the buffer data
- `q!` to quit and discard any changes made to the buffer data
- `w filename` to save the file under a different filename
- `wq` to save the buffer data to the file and quit

# vim shortcuts

- `2dd` (or other number) will delete 2 lines starting at the line of the cursor position
- `5x` (or other number) will delete 5 characters starting at the cursor position

vim cheat sheet: <https://vim.rtorr.com/>

vim cheat sheet: <https://devhints.io/vim>

## 15-131 : Great Practical Ideas for Computer Scientists

### VIM cheat sheet

Allison McKnight (aemcknig@andrew.cmu.edu)

#### Navigation

|     |                                                |   |                             |
|-----|------------------------------------------------|---|-----------------------------|
| h   | Move left                                      | H | Top line on screen          |
| j   | Move down                                      | M | Middle line on screen       |
| k   | Move up                                        | L | Bottom line on screen       |
| l   | Move right                                     |   |                             |
| 10j | Move down 10 lines                             |   |                             |
| gg  | First line of the file                         | e | The end of the current word |
| G   | Last line of the file                          | b | Beginning of current word   |
| :20 | Line 20 of the file                            | w | Beginning of next word      |
| 0   | Beginning of current line                      |   |                             |
| ^   | First non-whitespace character of current line |   |                             |
| \$  | End of current line                            |   |                             |
| %   | Move to matching parenthesis, bracket or brace |   |                             |

The left, right, up and down arrow keys can also be used to navigate.

#### Editing

|        |                                                              |
|--------|--------------------------------------------------------------|
| i      | Insert before current character                              |
| a      | Insert after current character                               |
| I      | Insert at the first non-whitespace character of the line     |
| o      | Insert a line below the current line, then enter insert mode |
| O      | Insert a line above the current line, then enter insert mode |
| r      | Overwrite one character and return to command mode           |
| U      | Undo                                                         |
| Ctrl+R | Redo                                                         |

#### Opening, closing and saving files

|              |                                                                                      |
|--------------|--------------------------------------------------------------------------------------|
| :w           | Save the current file                                                                |
| :wq          | Save the current file and close it; exits vim if no open files remain                |
| :w newname   | Save a copy of the current file as 'newname,' but continue editing the original file |
| :sav newname | Save a copy of the current file as 'newname' and continue editing the file 'newname' |
| :q!          | Close a file without saving                                                          |
| :e somefile  | Opens file in the current buffer                                                     |
| :x           | Write any changes to the file and close the file                                     |

#### Mode switching

|         |                                                   |
|---------|---------------------------------------------------|
| i       | Enter insert mode                                 |
| :       | Enter command mode                                |
| R       | Enter replace mode                                |
| v       | Enter visual mode (highlighting)                  |
| V       | Enter visual line mode (highlighting lines)       |
| esc     | Return to normal mode from insert or replace mode |
| esc+esc | Return to normal mode from command or visual mode |

#### Copy/pasting

##### Within vim

|   |                                       |
|---|---------------------------------------|
| y | Yank                                  |
| c | 'Change'; cut and enter insert mode   |
| C | Change the rest of the current line   |
| d | Delete; cut but remain in normal mode |
| D | Delete the rest of the current line   |
| p | Paste after the cursor                |
| P | Paste before the cursor               |
| x | Delete characters after the cursor    |
| X | Delete characters before the cursor   |

Copy/paste commands operate on the specified range. If in visual mode, that range is the highlighted text. If in normal mode, that range is specified by a series of modifiers to the commands:

|     |                                         |
|-----|-----------------------------------------|
| cw  | Change one word                         |
| c4w | Change four words                       |
| c4l | Change four letters                     |
| cc  | Change current line                     |
| 4x  | Change four characters after the cursor |
| 4p  | Paste five times after the cursor.      |

Modifiers work similarly for cut, delete, yank and paste.

##### From system clipboard

|              |                                                            |
|--------------|------------------------------------------------------------|
| :set paste   | Enter paste mode                                           |
| :set nopaste | Exit paste mode                                            |
| Ctrl+Shift+V | Paste into file, if in paste mode; Command+Shift+V for Mac |

# vim copy/paste

- Cut + paste: use `dd` to cut the line, then `p` to paste it at the cursor position
- Copy: `y` is the copy command (yank), and works similar to `d:` `yw` is yank word, `y$` to yank to end of line
  - `yy` is also yank the full line
  - Yanking can be tricky because can't see specifically what you're copying
- Visual mode lets you highlight text. Move cursor to where you want to start yanking, and press `v`. When you've highlighted the text you want to copy, press `y`. Then, move to where you want to paste, then press `p`.

# vim find



- In normal mode, press the forward slash ( / ) key.
  - Type the text you want to find, then press Enter.
- If the word appears after the current cursor location, it jumps to the first location where the text appears
- If the word doesn't appear after the current cursor location, it wraps around the end of the file to the first location in the file where the text appears (and indicates this with a message).
- It produces an error message stating that the text was not found in the file.
- To continue searching for the same word, press / and then press Enter , or use n (next)

# vim replace

## Must be in command line mode

- General format: `s/old/new/`
- `:s/old/new/g` to replace all occurrences of old in a line
- `:n,ms/old/new/g` to replace all occurrences of old between line numbers n and m
- `:%s/old/new/g` to replace all occurrences of old in the entire file
- `:%s/old/new/gc` to replace all occurrences of old in the entire file, but prompt for each occurrence

# vim configuration

Perhaps you don't want to reconfigure it each time

- e.g., the color scheme, or number of spaces for a tab
  - or spell checking...

Create a `.vimrc` file in your home directory

- `~/.vimrc`
  - (Your first dot file!)

For example, in the file:

```
colorscheme morning
set spell
set spelllang=en_us
```



# Colors?

Check available:

```
:colorscheme <space> Ctrl+d
```

or

```
:colorscheme <space> <tab>
```

## All the options:

<https://vimdoc.sourceforge.net/html/doc/options.html>