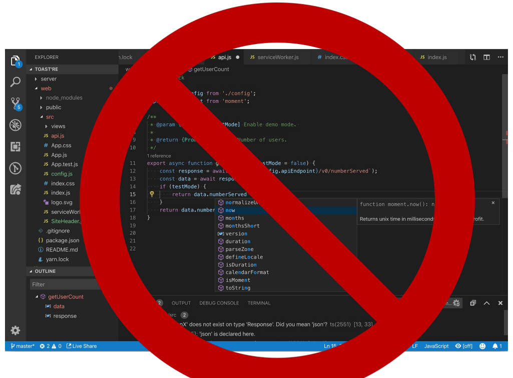
Tufts University GNU/Linux User Group (TUGSLUG)

How to not have to deal with VSCode tomfoolery

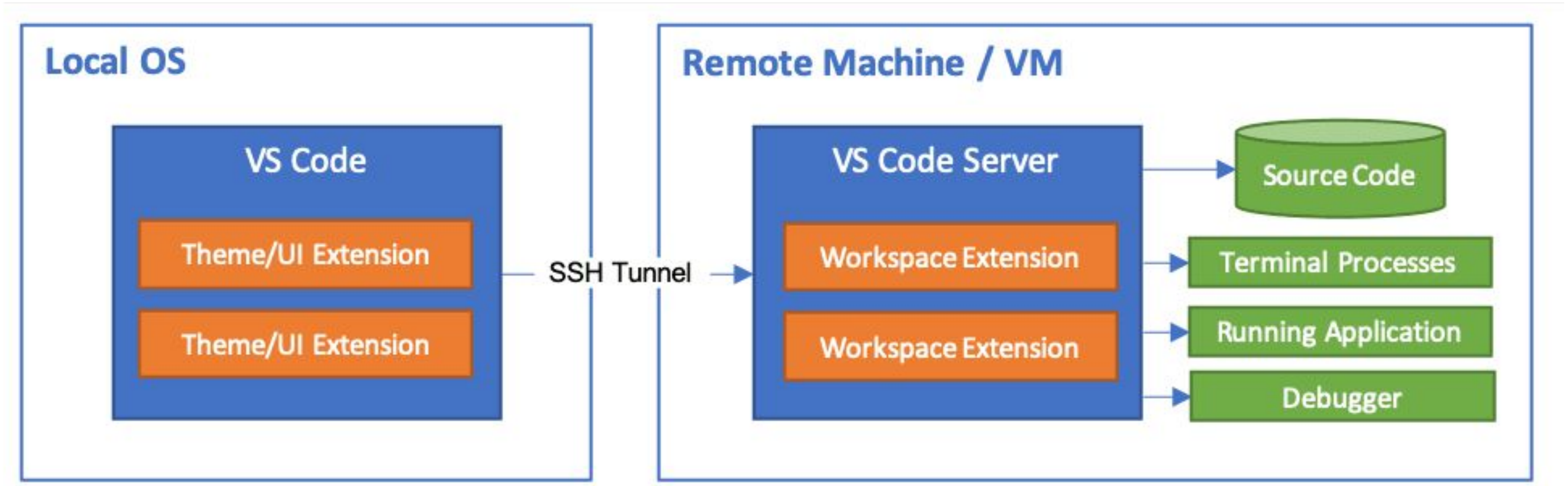
All about command-line text editing



Who we are

- Goals of the club
    - Helping with those interested in learning the basics of Linux & command-line tools.
    - Discussion of FOSS (Free and Open-Source Software).
    - Provide a community for those who use Linux as their primary OS.
- What we’re doing
    - Running workshops such as this.
        - Git workshop coming up next week! Same place, same time!
    - Conversing in our Discord server.
    - Experimenting with self-hosting FOSS tools.
    - Getting ourselves recognized as an actual club.

Why


Local OS
VS Code
Theme/UI Extension
Theme/UI Extension
SSH Tunnel
Remote Machine / VM
VS Code Server
Workspace Extension
Workspace Extension
Source Code
Terminal Processes
Running Application
Debugger

Actual image of Halligan servers as a direct result of VSCode

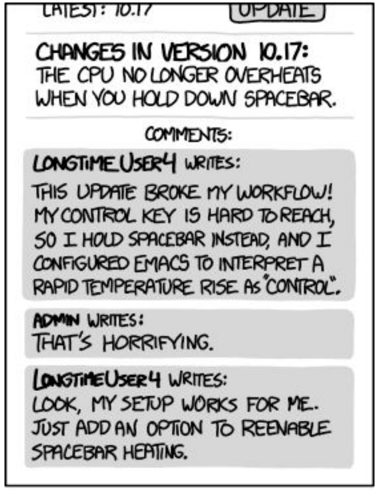
Why use a CLI editor?

- Fast
- *Fast*
- The same from anywhere
    - Your configuration lives on Halligan servers
    - Configuration is one file and easily transferable

Editors

GNU emacs

- often used in GUI mode
- but it also has a TUI (text UI) mode
    - run `$ emacs -nw` in a terminal
- defaults to TUI mode when graphical environment is unavailable
- command interface is (mostly) the same as in GUI mode
    - terminal emulator (e.g. xterm) may reinterpret ^C or ^M though
    - adjust terminal emulator's config accordingly
- Based around shortcuts to perform certain actions


LATEST: 10.17
UPDATE
CHANGES IN VERSION 10.17:
THE CPU NO LONGER OVERHEATS WHEN YOU HOLD DOWN SPACEBAR.
COMMENTS:
LONGTIMEUSER4 WRITES:
THIS UPDATE BROKE MY WORKFLOW! MY CONTROL KEY IS HARD TO REACH, SO I HOLD SPACEBAR INSTEAD, AND I CONFIGURED EMACS TO INTERPRET A RAPID TEMPERATURE RISE AS "CONTROL".
ADMIN WRITES:
THAT'S HORRIFYING.
LONGTIMEUSER4 WRITES:
LOOK, MY SETUP WORKS FOR ME. JUST ADD AN OPTION TO REENABLE SPACEBAR HEATING.
EVERY CHANGE BREAKS SOMEONE'S WORKFLOW.

Emacs modifiers & notation

- Hold `<ctrl>` and press a key: `C-<key>`
- Press `<esc>` and then press a key: `M-<key>`

Very Basics

- `$ emacs <file>` open file in emacs
- Move around with arrow keys
- `C-x C-s` save file
  - You can hold down `<ctrl>` and then press `x` first then press `s`
- `C-x C-c` exit emacs
- `C-g` cancel whatever operation is currently running
  - Useful if you're stuck in some weird menu; spam it if you don't know what's happening

Movement tips

- `C-←` Move left one word
- `C-→` Move right one word
- `C-e` Move to the end of a line
- `C-a` Move to the start of a line
- `M-<` Move to the start of a file (esc can be released before pressing shift <)
- `M->` Move to the end of a file

Some more neat tips

- `C-M-→` / `C-M-←` Move to matching parenthesis

Selecting, copying, and deleting text

- `C-<space>` to start a “mark”
- Move to where you want the mark to end
- Backspace to delete it all
- `M-w` to copy
- `C-w` to cut
- You can also cut a full line of text by going to the start of a line and pressing `C-k`

Searching for text

- `C-s` and type in a string to search for
- Continue pressing `C-s` to find the instance of the string you want to start editing

File management

Emacs has its own file browser (dired) so you can access files through the emacs interface without having to type emacs <file> each time.

- `C-x d` Access the dired menu after typing the path you want to open dired in
- Move around the interface with the arrow keys
- Enter folders with the enter key
- Rename files/folders with the R(shift+r) key (basically using the mv command from bash)
- Copy files/folders with the C(shift+c) key
- Make a folder with the + key

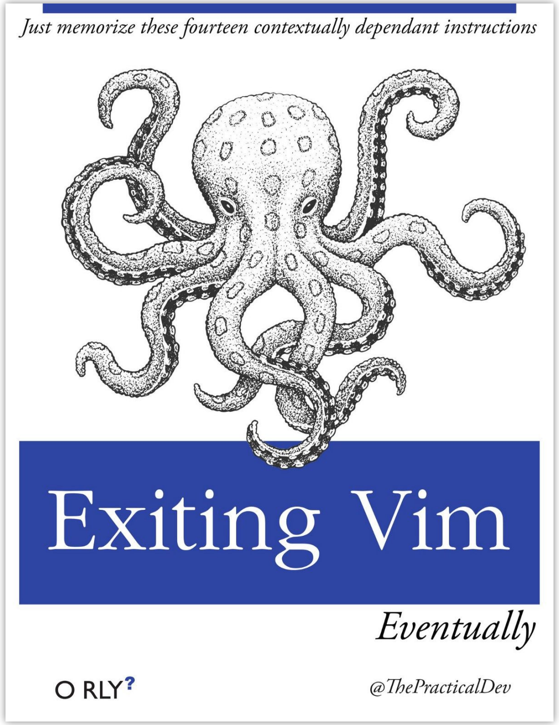
- `C-x C-f` Edit or create a new file in any location by typing the path of the file. This can be done in dired and it'll autofill the path to the current directory.
- `C-x C-b` can be used to swap back to the last open file (hint: hit tab to see all open files)

Ideas for moving forward with emacs

- `M-x` Open the command menu, which lets you run different tools and configure settings
- Pressing `M-x` and then typing `customize-themes` will let you pick from a couple of pre-installed themes for emacs
- Pressing `M-x` and then typing `column-number-mode` and pressing enter will show how long a certain line is. This is a pretty important setting because of the dreaded 80 character limit
- You can configure your default emacs experiencing by editing your .emacs.d/init.el file Which should be created in your home directory
  - A good way to start using this is to make emacs always have “column-number-mode” on when you start it. You can go to “.emacs.d/init.el” (you can create it if it’s not there) and at the end of it, put “(setq column-number-mode t)”
- `C-x 2` and `C-x 3` can be used to open multiple buffers, `C-x o` can be used to swap between them

Vim

- Philosophy: editor for editing text files
- Not based around shortcuts
- Three primary modes
  - Normal
    - Movement, entering other modes, the funky stuff
  - Insert
    - Typing
  - Visual
    - Selecting
  - Command
    - Save, quit


Just memorize these fourteen contextually dependant instructions
Exiting Vim
Eventually
O RLY?
@ThePracticalDev

Goals of this tutorial

- Introduce commands in tiers
- Don’t worry about remembering all of them
- If you learn the basics, you can pick up more over time and gradually get faster

The very basics: viewing files

- `$ vim [file]` opens a file in vim
- `h`, `j`, `k`, `l` move cursor left, down, up, or right
- `:q` exit the file "quit"

Making minor edits

- `i` enter insert mode
- `<esc>` exit insert mode
- `:w` save "write"
- `:wq` save *and* quit

Finding your way around faster

- `w` move to the beginning of the next word
- `b` move backwards to the start of the word

- `gg` go to top of file
- `G` go to bottom of file

- `/` begin searching
- `n` next instance of search
- `N` previous instance of search

Visual, yank and put

- V visual by line
- v visual by character

- y yank (copy)

- p put after (paste)
- P put before

- yy copy single line

Speedy edits

- `A` insert at the end of a line “append”
- `I` insert at the beginning of a line

- `o` open a new line below and enter insert mode
- `O` open a new line above and enter insert mode

- `dd` delete a line

- `dw` delete one word to the right “delete word”
- `cw` delete one word to the right and enter insert mode “change word”

Extra helpful stuff

- `^`, `$` go to beginning or end of line
- `%` go to matching parenthesis/bracket

- `D`, `C` delete to end of line, or change to end of line

- `ci<something>` “change inner”

- `x` delete single character
- `r` replace single character

- `:noh` stop highlighting search results “no highlight”

- `:e`, `:tabe`, `gt`, `gT` tools for editing multiple files

Configuration

Configuration lives in ~/.vimrc

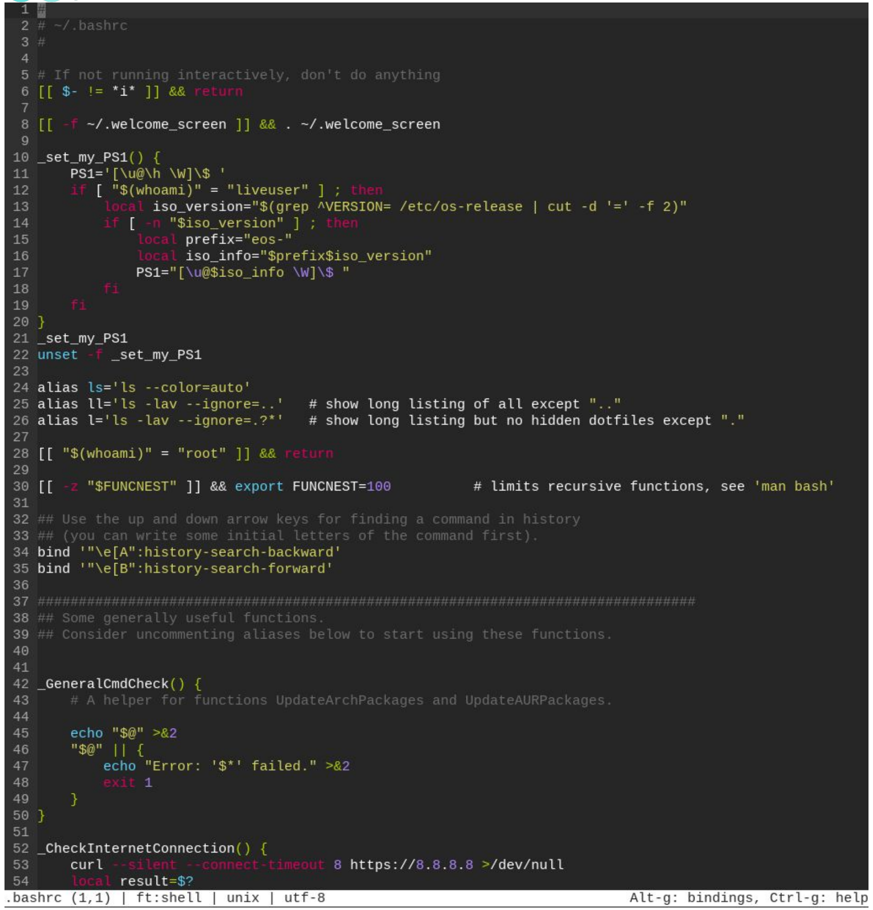
```
set tabstop=4 shiftwidth=4 softtabstop=4 expandtab
set autoindent
set colorcolumn=81
set number
set wildmenu
set ignorecase
set scrolloff=5
set title
```

A challenger approaches…

Tufts University GNU/Linux User Group (TUGSLUG)

micro

- modern interpretation of nano
- sane keybindings visible
  on-screen and integrated help
- mouse support
- available on the halligan servers



```
#
# ~/.bashrc
#

# If not running interactively, don't do anything
[[ $- != *i* ]] && return

[[ -f ~/.welcome_screen ]] && . ~/.welcome_screen

_set_my_PS1() {
    PS1='[\u@\h \W]\$ '
    if [ "$(whoami)" = "liveuser" ] ; then
        local iso_version="$(grep ^VERSION= /etc/os-release | cut -d '=' -f 2)"
        if [ -n "$iso_version" ] ; then
            local prefix="eos-"
            local iso_info="$prefix$iso_version"
            PS1="[\u@$iso_info \W]\$ "
        fi
    fi
}
_set_my_PS1
unset -f _set_my_PS1

alias ls='ls --color=auto'
alias ll='ls -lav --ignore=..'   # show long listing of all except ".."
alias l='ls -lav --ignore=.?*'   # show long listing but no hidden dotfiles except "."

[[ "$(whoami)" = "root" ]] && return

[[ -z "$FUNCNEST" ]] && export FUNCNEST=100          # limits recursive functions, see 'man bash'

## Use the up and down arrow keys for finding a command in history
## (you can write some initial letters of the command first).
bind '"\e[A":history-search-backward'
bind '"\e[B":history-search-forward'

################################################################################
## Some generally useful functions.
## Consider uncommenting aliases below to start using these functions.


_GeneralCmdCheck() {
    # A helper for functions UpdateArchPackages and UpdateAURPackages.

    echo "$@" >&2
    "$@" || {
        echo "Error: '$*' failed." >&2
        exit 1
    }
}

_CheckInternetConnection() {
    curl --silent --connect-timeout 8 https://8.8.8.8 >/dev/null
    local result=$?
.bashrc (1,1) | ft:shell | unix | utf-8                                    Alt-g: bindings, Ctrl-g: help
```

Open a file

- `$ micro [FILES]` - open files for editing
- `C-c` - copy selected text
- `C-v` - paste clipboard
- `C-x` - cut selected text
- `C-a` - select all
- `C-z` - undo change
- `C-y` - redo change
- `C-s` - save the current file
- `C-q` - quit micro
- `C-←` - move left one word
- `C-→` - move right one word
- `C-/` - comment/uncomment code
- `C-f` - find (with regex!)
- `C-p`/`C-n` - find previous/next

Tabs and Panes

- `C-t` - open new tab
- `C-o` - open file in current tab
- `C-q` - close current tab
- `M-,`/`M-.` to move between tabs
- `C-e` - open execution prompt
  - `> vsplit [FILE]` - vertically split and open FILE
  - `> hsplit [FILE]` - horizontally split and open FILE
- `C-w` - Switch between panes
- Or just use the mouse to move between panes and tabs

Integrated terminal

- `C-e` + `> term` - open a terminal in the current tab
- `C-e` + `> term [PROGRAM]` - execute PROGRAM and show results in the current pane

And more…

- plugin system
- code highlighting
- completely reconfigurable keybindings
- full documentation available on their GitHub
    - https://github.com/zyedidia/micro/

Shameless plug