

Set Yourself (and vim) Up For Success

Grace

February 2025

A post mostly for CS240 students at Purdue setting up their vim config.

Note: this is pretty out of date

While it should still work, it is pretty out of date and not the best tutorial. I intend on updating at some point

Update: February 12, 2025

Ok so I messed up with testing and didn't realize that ALE was using my clang-tidy configs from a file so if you want to use the eastwood linter, you will need to run the following command:

```
echo 'Checks: "-*,eastwood*"  
WarningsAsErrors: ""  
HeaderFilterRegex: ".*"  
FormatStyle: "file"' > ~/cs240/.clang-tidy
```

If you do this, the eastwood linter should work. Sorry for the mistake and if you run into any issues, please contact me on discord (username: qyvo).

Let's Start The Show

I have a crazy hypothetical here. Lets say that you are signed up for a C programming class that requires you to work on a remote Linux server set up by your professor. You can try to use VSCode with the Remote - SSH extension, but it eats up disk space and you only have a 5 GB quota before things stop working. This leaves you with two options - use vim or emacs. You haven't fully lost your mind yet so you use vim. But it's hard to use and doesn't have all of the creature comforts that you are used to in VSCode.

If you find yourself in this situation, you are in the right place. Here, you will be able to set up vim just the way you like it and have it be much nicer and fun to use.

Before we start, a few things

I don't really want to spend all the time to set everything up and just want a proper working vim config. Can you just give me one?

Yeah, sure. You can use the config that I use if you want to. It is available on my github page and can be installed on data with the following command:

```
curl -s https://raw.githubusercontent.com/gyoder/.vim/master/install.sh | bash

# if you are doing this on purdue data use this command
curl -s https://raw.githubusercontent.com/gyoder/.vim/master/data_install.sh | bash
```

This will give you clangd and neovim too. Just run nvim instead of vim and you are all good!

I just want the eastwood linter

Go ahead and skip to that section!

Wait how do I use vim?

I recommend checking out this site. Even if you can already kinda use vim, check it out to learn some advanced techniques.

Neovim? What's that?

Neovim is the cooler cousin of vim. It is a fork and is better supported with a few extra features (Language Server Client, Lua support, etc). I recommend checking it out.

Installing Neovim

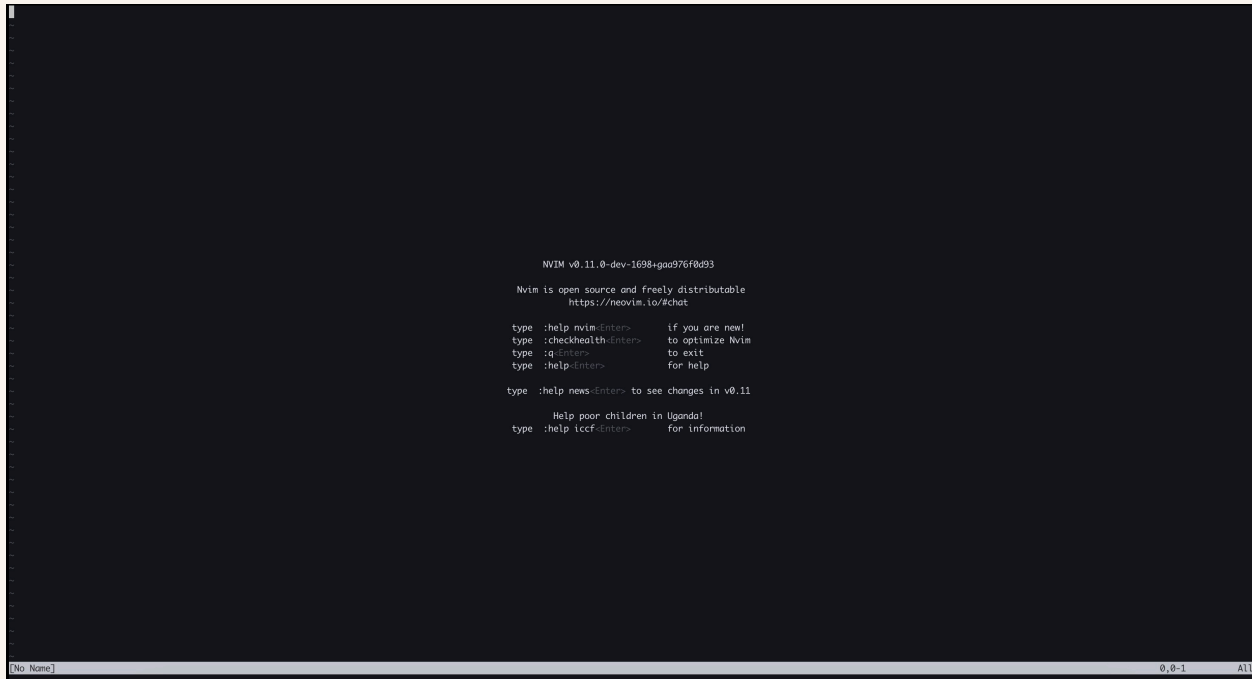
Going forward I will be assuming you are using neovim, you can use regular vim if you really want to but I recommend using neovim.

If you are on your own machine, installation is easy. Just use whatever package manager you like. If you are on a shared machine (like the purdue data server), then you will need to download the release from their github and add it to your path.

You can do this with the following commands if you are ok with it potentially being out of date by the time you read this.

```
# download the release
wget -O $HOME/nvim.tar.gz \
  https://github.com/neovim/neovim/releases/download/nightly/nvim-linux-x86_64.tar.gz
```

```
tar -xvf $HOME/nvim.tar.gz -C $HOME # untar it
mv $HOME/nvim-linux-x86_64 $HOME/nvim # put it into the right folder
rm -rf $HOME/nvim.tar.gz # remove the tar file
echo 'export PATH="$HOME/nvim/bin:$PATH"' >> $HOME/.bashrc # add it to your path
source $HOME/.bashrc # reload your bashrc
```



You can now run `nvim` to start neovim and see if it works. If you are on purdue's data server, you should make sure you don't see version 0.8.3.

Quick clangd tangent

Ok, so if you want to have autocomplete, error checking, and whatever we will eventually need clangd. If you are on the clangd data server, you can just run the following commands:

```
wget -O $HOME/clangd.zip https://github.com/clangd/clangd/releases/download/19.1.2/clangd-linux-19.1.2.zip
unzip $HOME/clangd.zip -d $HOME/clangd
mv $HOME/clangd/clangd_19.1.2/* $HOME/clangd
rm -rf $HOME/clangd.zip
echo 'export PATH="$HOME/clangd/bin:$PATH"' >> $HOME/.bashrc
echo 'export CLANG_LIB_DIR="$HOME/clangd/lib"' >> $HOME/.bashrc
source $HOME/.bashrc
```

Basic Config Stuff

First run `mkdir -p $HOME/.config/nvim` to make the config folder. Then run `nvim $HOME/.config/nvim/init.vim` to open the config file. (if you are using regular vim, run `vim ~/.vimrc` instead). This file will be ran every time you open neovim. Go ahead and add the following to it:

```
set nocompatible
filetype on
filetype plugin on
filetype indent on
syntax on
```

Now add some basic settings. I have included a few example ones below. To see the changes made, save and close the file, then reopen it.

[illegible]

These are just the settings I thought it would be worth to start with. There are many more so just go ahead and google for some.

Plugins

Plugins are great. Just like VSCode extensions, there are so many of them and they can add so many features. We will be using vim-plug to manage our plugins. To install it, run the following command:

```
# If you are using neovim, run the following command
sh -c 'curl -fLo "${XDG_DATA_HOME:-$HOME/.local/share}"/nvim/site/autoload/plug.vim
```

```
--create-dirs \
https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim'

# If you are using regular vim, run the following command instead
curl -fLo ~/.vim/autoload/plug.vim --create-dirs \
https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim
```

Now we can add plugins to the config file. Just add `call plug#begin()` above the settings and `call plug#end()` below them. Then add whatever plugins you want.

For funzies, lets install a colorscheme. you can check some out at this cool website.



vimcolorschemes

← back

view [folke/tokyonight.nvim](#) on Github

folke

tokyonight.nvim

☆ 6722
🔥 24/week

A clean, dark Neovim theme written in Lua, with support for lsp, treesitter and lots of plugins. Includes additional themes for Kitty, Alacritty, iTerm and Fish.

created **almost 4 years ago**

last commit **8 days ago**

```
tokyonight neovim
1 vim.o.background = "dark"
2 vim.cmd.colorscheme "tokyonight"
NORMAL init.lua unix utf-8 0% 0:2
1 " Returns true if the color hex value is light
2 function! IsHexColorLight(color) abort
3   let l:raw_color = trim(a:color, '#')
4
5   let l:red = str2nr(substitute(l:raw_color, '{0}(.{2})', '1', 'g'),
6   let l:green = str2nr(substitute(l:raw_color, '{2}(.{2})', '1', 'g'),
7   let l:blue = str2nr(substitute(l:raw_color, '{4}(.{2})', '1', 'g')
8
9   let l:brightness = ((l:red * 299) + (l:green * 587) + (l:blue * 114
10
11   return l:brightness > 155
12 endfunction
NORMAL code.vim unix utf-8 50% 6:12
```

```
tokyonight neovim
1 vim.o.background = "light"
2 vim.cmd.colorscheme "tokyonight"
NORMAL init.lua unix utf-8 0% 0:2
1 " Returns true if the color hex value is light
2 function! IsHexColorLight(color) abort
3   let l:raw_color = trim(a:color, '#')
4
5   let l:red = str2nr(substitute(l:raw_color, '{0}(.{2})', '1', 'g'),
6   let l:green = str2nr(substitute(l:raw_color, '{2}(.{2})', '1', 'g'),
7   let l:blue = str2nr(substitute(l:raw_color, '{4}(.{2})', '1', 'g')
8
9   let l:brightness = ((l:red * 299) + (l:green * 587) + (l:blue * 114
10
11   return l:brightness > 155
12 endfunction
NORMAL code.vim unix utf-8 50% 6:12
```

```

1 set nocompatible
2 filetype on
3 filetype plugin on
4 filetype indent on
5 syntax on
6
7
8 set number "Adds Line Numbers to the left hand side
9 set cursorline "Highlights the line you are on
10 set cursorcolumn "Highlights the column you are on
11 set colorcolumn=80 "Highlights the 80th column
12 set shiftwidth=2 "Set shift ammount to 2
13 set tabstop=2 "Sets tab width to 2
14 set expandtab "Use spaces for tabs
15 set scrolloff=10 "Keep the cursor at least 10 lines from the top or bottom
16 set incsearch "Highlight chars while searching
17 set ignorecase "Make case insensitive during search
18 set smartcase "Actually just kidding, not insensitive if you capitalize it
19 set showcmd "Show the last command run at all times
20 set history=500 "Keep 500 commands in history
21 set wildmenu "Command tab completion
22
23 call plug#begin()
24 Plug 'folke/tokyonight.nvim'
25 call plug#end()
26
27 colorscheme tokyonight

```

.config/nvim/init.vim 24,30 All

I like this one so I navigate to the github in the top left corner and add the username/repo to the list of plugins. I also add the colorscheme to the settings. Now I just need to close and reopen neovim and I should have a new colorscheme

```
Error detected while processing /u/riker/u99/yoder177/.config/nvim/init.vim
line 24:
E471: Argument required: Plug
line 27:
E185: Cannot find color scheme 'tokyonight'
Press ENTER or type command to continue
```

Just kidding. We will need to install it first. Go ahead and run the `:PlugInstall` command to do that and close and reopen neovim. Now you have a new colorscheme

```

1 set nocompatible
2 filetype on
3 filetype plugin on
4 filetype indent on
5 syntax on
6
7
8 set number "Adds Line Numbers to the left hand side
9 set cursorline "Highlights the line you are on
10 set cursorcolumn "Highlights the column you are on
11 set colorcolumn=80 "Highlights the 80th column
12 set shiftwidth=2 "Set shift amount to 2
13 set tabstop=2 "Sets tab width to 2
14 set expandtab "Use spaces for tabs
15 set scrolloff=10 "Keep the cursor at least 10 lines from the top or bottom
16 set incsearch "Highlight chars while searching
17 set ignorecase "Make case insensitive during search
18 set smartcase "Actually just kidding, not insensitive if you capitalize it
19 set showcmd "Show the last command run at all times
20 set history=500 "Keep 500 commands in history
21 set wildmenu "Command tab completion
22
23 call plug#begin()
24 Plug 'folke/tokyonight.nvim'
25 call plug#end()
26
27 colorscheme tokyonight

```

.config/nvim/init.vim

24,30

ALL

Linting (With the CS240 Eastwood linter)

Linting is super useful. We are going to use a vim plugin called ALE to do this. To install it, add it to the list of plugins in the config file and run `:PlugInstall`. We are also going to add some configuration to the config file to make it work with clangd and the eastwood linter.

```

" Add the ale plugin
call plug#begin()
  Plug 'dense-analysis/ale'
call plug#end()

" Not required if you are not doing CS240 Eastwood linting
let $PATH = $PATH . ':' . expand('$HOME') . '/clangd/bin:/u/riker/u98/cs240/bin'
let g:ale_linters = {
  \ 'python': ['pylint'],
  \ 'vim': ['vint'],
  \ 'cpp': ['clang'],
  \ 'c': ['clang', 'clangtidy']
\}

" Config to use the cs240 eastwood
let g:ale_cpp_clangtidy_checks = ['-*', 'eastwood*']
let g:ale_cpp_clangtidy_executable = 'clang-tidy'
let g:ale_c_parse_compile_commands=1
let g:ale_cpp_clangtidy_extra_options = '-- -I/homes/cs240/public/include/clang/
include'
let g:ale_cpp_clangtidy_options = ''

```

```
let g:ale_set_balloons=1
let g:ale_linters_explicit=1
let g:airline#extensions#ale#enabled=1
```

```
19 set showcmd      "Show the last command run at all times
20 set history=500   "Keep 500 commands in history
21 set wildmenu      "Command tab completion
22
23 let $PATH = $PATH . ':' . expand('$HOME') . '/clangd/bin:/u/riker/u98/cs240/bin'
24 let g:ale_linters = {
25   \ 'c': ['clang', 'clangtidy']
26 \}
27
28 let g:ale_cpp_clangtidy_checks = ['-*', 'eastwood*']
29 let g:ale_cpp_clangtidy_executable = 'clang-tidy'
30 let g:ale_c_parse_compile_commands=1
31 let g:ale_cpp_clangtidy_extra_options = '-- -I/homes/cs240/public/include/clang/include'
32 let g:ale_cpp_clangtidy_options = ''
33 let g:ale_set_balloons=1
34 let g:ale_linters_explicit=1
35 let g:airline#extensions#ale#enabled=1
36
37 call plug#begin()
38   Plug 'folke/tokyonight.nvim'
39   Plug 'dense-analysis/ale'
40 call plug#end()
41
42
43
44 colorscheme tokyonight
```

.config/nvim/init.vim [+]

44,22

Bot

You will also need to include a .clang-tidy file in you cs240 path if you want to use the linter

```
echo 'Checks: "-*,eastwood*'
WarningsAsErrors: ""
HeaderFilterRegex: ".*"
FormatStyle: "file" > ~/cs240/.clang-tidy
```

If we look at a test file, we see...

```

W 1 #include "main.h"      ■ Malformed ifndef component of include guard in header file. [eastwood-Rule8cCheck]
2 #include <stdio.h>
3
W 4 /* header comments are lame*/      ■ Malformed function header comment for function main. [eastwood-Rule7aCheck]
W 5 int main() {      ■ Exactly one empty line required after function header comment for function main. [eastwood-Rule5eCheck]
W 6     printf("wow look at this really cool code. I am so glad I can write good clean code. wow i am so good at this\n");      ■ Line length must be less
W 7     int cool;      ■ Variable declaration without definition is forbidden. [eastwood-Rule12bCheck]
W 8     cool=3;      ■ Trailing space required. [eastwood-Rule3bCheck]
W 9     int coolerVar = 0;      ■ Variable coolerVar is not all lowercase and separated by underscores [eastwood-Rule1aCheck]
10
11
W 12    if (cool > 5 || coolerVar < 4) {      ■ This sub-expression must be surrounded by parentheses [eastwood-Rule6aCheck]
E 13        printf("wow thats cool!")      ■ expected ';' after expression
14    }
15
16
W 17
W 18    cool = ++coolerVar; // yipieeeeeeeeeee      ■ Comments must appear above code except for else, case, #defines, or declarations [eastwood-Rule5bCh
W 19    return 0;
W 20 }      ■ Missing function footer comment [eastwood-Rule5dCheck]

```

cs240/random/test.c [+]

16,0-1 All

Yep, it works. Now I can feel the weight of Turkstra's judgement even when I am in the editor!

Autocomplete

All modern editors have autocomplete so we should have it too. We are going to use Neovim's built in LSP client (sorry regular vim users) with the cmp plugin to do this. We need to install 18 billion other things to do this but its all documented on their github page so we will just take that.

Add the following plugins, reload neovim, and run `:PlugInstall` to install them.

```

Plug 'neovim/nvim-lspconfig'
Plug 'hrsh7th/cmp-nvim-lsp'
Plug 'hrsh7th/cmp-buffer'
Plug 'hrsh7th/cmp-path'
Plug 'hrsh7th/cmp-cmdline'
Plug 'hrsh7th/nvim-cmp'
Plug 'hrsh7th/cmp-vsnip'
Plug 'hrsh7th/vim-vsnip'

```

We are also just going to paste the following code in. It is yonked right from the ReadMe and is verified to work.

```

lua <<EOF
-- Set up nvim-cmp.
local cmp = require'cmp'

```

```

cmp.setup({
  snippet = {
    -- REQUIRED - you must specify a snippet engine
    expand = function(args)
      vim.fn["vsnip#anonymous"](args.body) -- For `vsnip` users.

    end,
  },
  window = {
    -- completion = cmp.config.window.bordered(),
    -- documentation = cmp.config.window.bordered(),
  },
  mapping = cmp.mapping.preset.insert({
    ['<C-b>'] = cmp.mapping.scroll_docs(-4),
    ['<C-f>'] = cmp.mapping.scroll_docs(4),
    ['<C-Space>'] = cmp.mapping.complete(),
    ['<C-e>&t;'] = cmp.mapping.abort(),
    ['<CR>'] = cmp.mapping.confirm({ select = true }),
  }),
  sources = cmp.config.sources({
    { name = 'nvim_lsp' },
    { name = 'vsnip' }, -- For vsnip users.
    -- { name = 'luasnip' }, -- For luasnip users.
    -- { name = 'ultisnips' }, -- For ultisnips users.
    -- { name = 'snippy' }, -- For snippy users.
  }, {
    { name = 'buffer' },
  })
})

-- To use git you need to install the plugin petertriho/cmp-git and uncomment
lines below
-- Set configuration for specific filetype.
--[[ cmp.setup.filetype('gitcommit', {
  sources = cmp.config.sources({
    { name = 'git' },
  }, {
    { name = 'buffer' },
  })
})
require("cmp_git").setup() ]]-

-- Use buffer source for '/' and '?' (if you enabled `native_menu`, this won't
work anymore).
cmp.setup.cmdline({ '/', '?' }, {
  mapping = cmp.mapping.preset.cmdline(),
  sources = {
    { name = 'buffer' }
  }
})

```

```

-- Use cmdline & path source for ':' (if you enabled `native_menu`, this won't
work anymore).
cmp.setup.cmdline(':', {
  mapping = cmp.mapping.preset.cmdline(),
  sources = cmp.config.sources({
    { name = 'path' }
  }, {
    { name = 'cmdline' }
  }),
  matching = { disallow_symbol_nonprefix_matching = false }
})

-- Set up lspconfig.
local capabilities = require('cmp_nvim_lsp').default_capabilities()
-- Replace YOUR_LSP_SERVER with each lsp server you've enabled.
require('lspconfig')['clangd'].setup {
  capabilities = capabilities
}
EOF

```

Now, it should work.

```

W 1 #include "main.h"      ■ Malformed ifndef component of include guard in header file. [eastwood-Rule8cCheck]
2 #include <stdio.h>
3
4 /* header comments are lame*/ ■ Malformed function header comment for function main. [eastwood-Rule7aCheck]
W 5 int main() {           ■ Exactly one empty line required after function header comment for function main. [eastwood-Rule5eCheck]
6   printf("wow look at this really cool code. I am so glad I can write good clean code. wow i am so good at this\n"); ■ Line length must be less
7   int cool;             ■ Variable declaration without definition is forbidden. [eastwood-Rule12bCheck]
8   cool=3;               ■ Trailing space required. [eastwood-Rule3bCheck]
9   int coolerVar = 0;    ■ Variable coolerVar is not all lowercase and separated by underscores [eastwood-Rule1aCheck]
10
11
W 12 if (cool > 5 || coolerVar ≤ 4) { ■ This sub-expression must be surrounded by parentheses [eastwood-Rule6aCheck]
13   printf("wow thats cool!");
14 }
15
16 f
17 for~      Snippet      (init-statement; condition; inc-expression) {statements}
W 18 fclose~  Function      (FILE *stream)
19 fdopen~   Function      (int fd, const char *modes)
W 20 } feof~   Function      (FILE *stream)
   feof_unlocked~ Function (FILE *stream)
   ferror~    Function      (FILE *stream)
   ferror_unlocked~ Function (FILE *stream)
   fflush~    Function      (FILE *stream)
   fflush_unlocked~ Function (FILE *stream)
   fgetc~     Function      (FILE *stream)
   fgetc_unlocked~ Function (FILE *stream)
   fgetpos~   Function      (FILE *restrict stream, fpos_t *restrict pos)
   fgets~     Function      (char *restrict s, int n, FILE *restrict stream)
   fileno~    Function      (FILE *stream)
   fileno_unlocked~ Function (FILE *stream)
   flockfile~ Function      (FILE *stream)
   fmemopen~  Function      (void *s, size_t len, const char *modes)
   fopen~     Function      (const char *, const char *)
cs240/r
-- INSERT --

```

Conclusion

Hey, look at that. you have a crazy powerful editor. If you want to do more with it, you are sure able to. I have 20 plugins that I use between file explorers, minimaps, fuzzy

finders, and more. You can just google for cool things and you'll find them. If you want inspiration, my vim config is on my github here

If you have any questions, feel free to reach out to me on discord (username: qyvo) as I am always happy to help. Anyway here are some screenshots of my neovim setup :)

```

" Press ? for help
94 set foldmethod=indent          "Fold based on indent
95 set foldmethod=10             "Deepest fold is 10 levels
96 set nofoldenable              "Don't fold by default
97 set foldlevel=1               "This is just what I use
98
99 " PLUGINS ----- {{{
100
101 call plugbegin('~/.vim/plugged')
102
103   Plug 'dense-analysis/ale'
104   Plug 'preservim/nerdtree'
105   Plug 'tjdevries/vim-solarized8'
106   Plug 'sainmh/everforest'
107   Plug 'craigjennery/vim-autotag'
108   Plug 'tjdevries/vim-fugitive'
109   Plug 'vim-airline/vim-airline'
110   Plug 'vim-airline/vim-airline-themes'
111   let g:airline_theme='base16_solarized_dark'
112   Plug 'junegunn/fzf'
113   Plug 'junegunn/fzf.vim'
114   Plug 'terryms/vim-smooth-scroll'
115   Plug 'f-person/git-blame.nvim'
116   Plug 'Yggdrasil/indentline'
117   Plug 'deoplete-plugins/deoplete-clang'
118   Plug 'Shougo/deoplete.nvim'
119   Plug 'nvim-lua/plenary.nvim'
120   Plug 'emilienemore/clang-tidy.nvim'
121   Plug 'neovim/nvim-lspconfig'
122   Plug 'nvim-lua/completion-nvim'
123   Plug 'hrsh7th/cmp-nvim-lsp'
124   Plug 'hrsh7th/cmp-buffer'
125   Plug 'hrsh7th/cmp-path'
126   Plug 'hrsh7th/cmp-cmdline'
127   Plug 'hrsh7th/vim-cmp'
128   Plug 'hrsh7th/cmp-vsnip'
129   Plug 'hrsh7th/vim-vsnip'
130   Plug 'Raimondi/delimitMate'
131   if (g:hostname =~# 'cs.purdue.edu$')
132     Plug 'neoclides/coc.nvim', {'branch': 'release'}
133     Plug 'github/copilot.vim'
134     Plug 'wfxr/minimap.vim'
135     let g:minimap_width = 10
136     let g:minimap_auto_start = 1
137     let g:minimap_highlight_search = 1
138     let g:minimap_git_colors = 1
139   endif
140
141 call plugend()
142
143 " }}}
144
145
/Users/scie/.vim/.vimrc          vim utf-8[unix]

```

```

1 #!/bin/bash
2
3 wget -O $HOME/clangd.zip https://github.com/clangd/clangd/releases/download/19.1.2/clangd-linux
4 unzip $HOME/clangd.zip -d $HOME/clangd
5 mv $HOME/clangd/clangd_19.1.2/ $HOME/clangd
6 rm -rf $HOME/clangd.zip
7
8 wget -O $HOME/nvim.tar.gz https://github.com/neovim/neovim/releases/download/nightly/nvim-linux
9 tar -xvf $HOME/nvim.tar.gz -C $HOME
10 mv $HOME/nvim-linux-x86_64 $HOME/nvim
11 rm -rf $HOME/nvim.tar.gz
12
13 echo 'export PATH="$HOME/clangd/bin:$HOME/nvim/bin:$PATH"' >> $HOME/.bashrc
14 echo 'export CLANGD_LIB_DIR="$HOME/clangd/lib"' >> $HOME/.bashrc
15
16 source $HOME/.bashrc
17
18 # ask if they would like to install the .vim folder
19
20 echo "Would you like to install the .vim folder from gyoder? (y/n)"
21 read install_vim
22 if [ $install_vim = "y" ]; then
23   curl -s https://raw.githubusercontent.com/gyoder/vim/master/install.sh | bash
24 fi
25
26 You • Thu Feb 6 16:18:33 2025 • data install script

```

```

" Press ? for help
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <math.h>
4
5 typedef struct node node_t;
6 struct node {
7   int num;
8   node_t* next;
9 };
10
11
12 int main() {
13   int a = 5;
14   int b = 8;
15
16   int c = pow(a, b);
17
18   node_t* head = (node_t*) malloc(sizeof(node_t));
19   head->num = c;
20
21   for (int i = 0; i < 5; i++) {
22     head->next =
23   }
24
25 }

```

```
Press ? for help      1.*wikipedia.org
                      2.medium.com
                      3.*mastodon.*
                      4.*google.*
                      5.*youtube.*
scd/Documents/webmapping/blacklist.txt 6.*youtube.*
classification_server.py 7.*reddit.*
data.csv 8.*twitch.*
graph.graphml 9.*discord.*
path.png 10.*instagram.*
scrape.py 11.*facebook.*
testdata.csv 12.*github.com.*
             13.*gitlab.*
             14.*nexus.*
             15.*creativecommons,* You * Wed Jan 8 23:34:02 2025 * init
             16.*bsky.app.*
             17.*tiktok.*
             18.*linkedin.*
             19.*microsoft.*
             20.*linux.*
             21.*apple.*
             22.*plannedparenthood.*
             23.*mozilla.*
             24.*gnu.*
             25.*archlinux.*
             26.*matrix.*
             27.*steamcommunity.*
             28.*librewolf.*
             29.*nginx.*
             30.*dell.*
             31.*rust.*
             32.*apache.org.*
             33.*docker.*
             34.*digitalocean.*
             35.*aws.*
             36.*readthedocs.io.*
             37.*w3.org.*
             38.*ansible.*
             39.*hubnettes.*
             40.*rpm.*
             41.*nodejs.*
             42.*archive.org.*
             43.*.edu.*
             44.*.gov.*
             45.*ide.org.*
             46.*pine64.*
             47.*raspberrypi.*
             48.*arduino.*
             49.*adafruit.*
             50.*ingun.*
             51.*canonical.*
             52.*ubuntu.*
             scments/webmapping/blacklist.txt tex_

35 |         continue
36 |     elif splink[Z] == url:
37 |         continue
38 |     elif len(splink) > 3 and re.match("(next|previous)?host", splink[3]):
39 |         try:
40 |             links.add(requests.get(link, allow_redirects=True).url.split('/')[2])
41 |         except requests.exceptions.RequestException as e:
42 |             pass
43 |     elif any([re.search(b, splink[Z]) for b in blacklist]):
44 |         continue
45 |     links.add(splink[Z])
46 | return links
47 |
48 | with open('blacklist.txt','r') as f:
49 |     blacklist = f.read().splitlines()
50 |
51 | if __name__ == '__main__':
52 |     try:
53 |         #print(get_links("grace.pink"))
54 |         if not len(sys.argv) == 2:
55 |             print("Usage: python scrape.py <iterations>")
56 |             sys.exit(1)
57 |         DG = nx.DiGraph()
58 |         queue = [Link("grace.pink")]
59 |         for i in range(int(sys.argv[1])):
60 |             link = queue.pop(0)
        You * Wed Jan 8 23:34:02 2025 * init
61 |             print(f"Scanning '{link.url}'")
62 |             if not link.scanned:
63 |                 if link.scanned = True
64 |                     DG.add_node(link)
65 |                 try:
66 |                     links = get_links(link.url)
67 |                 except:
68 |                     print(f"Failed to scan", link.url)
69 |                     time.sleep(2)
70 |                     continue
71 |                 for l in links:
72 |                     if DG.has_node(Link(l)):
73 |                         DG.add_edge(link, Link(l))
74 |                     else:
75 |                         queue.append(Link(l))
76 |                         DG.add_edge(link, Link(l))
77 | except KeyboardInterrupt:
78 |     print("\nProcess interrupted by user")
79 | finally:
80 |     print("Nodes:")
81 |     [print(str(n)) for n in DG.nodes]
82 |     nx.write_graphml(DG, "graph.graphml")
83 |
84 |
85 |
```

```
# AP_EFT/      118 }
# AP_ESC_Telem/ 119 I2c_master_stop(cmd);
# AP_ExternalAHBS/ 120
# AP_ExternalControl/ 121 TickType_t timeout = 1 + 10L * (send_len + recv_len) * 1000 / bus.bus_clock / portTICK_PERIOD_MS;
# AP_ETEScheduler/ 122 for (int i = 0; result != -result; i++) {
# AP_FileSystem/ 123     result = (I2c_master_cmd_begin(bus.port, cmd, timeout) == ESP_OK);
# AP_FlashIface/ 124     if (result) {
# AP_FlashStorage/ 125         I2c_reset_tx_fifo(bus.port);
# AP_Flow/ 126         I2c_reset_rx_fifo(bus.port);
# AP_Frsky_Telem/ 127     }
# AP_Generation/ 128 }
# AP_GPS/
# AP_Grip/
ESP32_REV_MIN_0 modules/esp_idf/components/esp_hw_support/port/esp32/Kconfig.hw.support-
ESP32_REV_MIN_0 build/esp32buzz/esp-idf_build/bootloader/config/sdkconfig.json />
ESP32_REV_MIN_0 build/esp32buzz/esp-idf_build/bootloader/config/sdkconfig.json /> ..
ESP32_REV_MIN modules/esp_idf/components/esp_hw_support/port/esp32/Kconfig.hw.support-
ESP32_REV_MIN modules/esp_idf/components/esp_hw_support/port/esp32/Kconfig.hw.support-
ESP32_REV_MIN build/esp32buzz/esp-idf_build/config/sdkconfig.json /> ESP32.R..
ESP32_REV_MIN build/esp32buzz/esp-idf_build/bootloader/config/sdkconfig.json />
ESP32_REV_MIN modules/esp_idf/components/bootloader-support/bootloader_flash/src/boo-
ESP32_REV_MIN build/esp32buzz/esp-idf_build/bootloader/config/sdkconfig.json /> ..
boards modules/esp_idf/components/esp_hw_support/port/esp32/Kconfig.hw.support-
hdfef build/esp32buzz/esp-idf_build/config/sdkconfig.json /> "ESP32.R..
profil build/esp32buzz/esp-idf_build/bootloader/config/sdkconfig.json />
ESP32_FIXED_SIZE modules/esp_system/port/soc/esp32/Kconfig.im-
ESP32_FIXED_SIZE modules/esp_idf/components/esp_system/port/soc/esp32/Kconfig.memory--
util/l modules/esp_idf/components/bootloader-support/bootloader_flash/src/boo-
ESP32_MEMMAP_TRACEMEM_TWOBANKS modules/esp_idf/components/esp_system/port/soc/esp32/Kconfig.tracemem-
ESP32_MEMMAP_TRACEMEM modules/esp_idf/components/esp_system/port/soc/esp32/Kconfig.tracemem-
Analog ESP32_IRAM_AS_8BIT_ACCESSIBLE_MEMORY modules/esp_idf/components/esp_system/port/soc/esp32/Kconfig.im-
Analog ESP32_FIXED_SIZE modules/esp_idf/components/esp_system/port/soc/esp32/Kconfig.memory--
Device modules/esp_idf/components/esp_system/port/soc/esp32/Kconfig.system-
Device modules/esp_idf/docs/conf_common.py //ESP32_DOCS = ["api-reference..
ESP32_DOCS modules/esp_idf/docs/conf_common.py //ESP32_DOCS = ["api-reference..
GPIO_C modules/esp_idf/components/esp_system/port/soc/esp32/Kconfig.system-
GPIO_H build/esp32buzz/esp-idf_build/bootloader/config/sdkconfig.json /> ESP32.D..
HAL_ES modules/esp_idf/build/bootloader/config/sdkconfig.json /> ..
ESP32_DISABLE_BASIC_ROM_CONSOLE build/esp32buzz/esp-idf_build/bootloader/config/sdkconfig.json /> ..
HAL_ES modules/esp_idf/components/bootloader-support/bootloader_flash/src/boo-
HAL_ES libraries/AP_HAL_ESP32/AnalogIn.cpp //#define ESP32_ADC_MVLINK_DE-
I2C_SW ESP32_3.. modules/esp_idf/docs/static/chip-esp32-e3.svg /> id="ESP32_3..>S>".
I2C_SW modules/esp_idf/docs/static/chip-esp32-e3.svg /> id="ESP32_16..>S>"/>
I2CDRV modules/esp_idf/docs/static/chip-esp32-e3.svg /> id="ESP32_1..
I2CDrv modules/esp_idf/docs/static/chip-esp32.svg /> id="ESP32_11..>S>"/>
OSD_cp modules/esp_idf/docs/static/chip-esp32.svg /> id="ESP32_10..>S>"/>
Profil T234M451PS488 (0)
Profil Taggs ESP32
RCInput RCInput.h 160 return AP_HAL::OwnPtr<AP_HAL::I2CDevice>(nullptr);
RCOutput.cpp 161 }
RCOutput.h 162 auto dev = AP_HAL::OwnPtr<AP_HAL::I2CDevice>(NEW_THROWN I2CDevice(bus, address, bus_clock, use_smbus, timeout_ms));
README.md 163 return dev;
README_e3_debug_howto.md 164 }
RttStgReader.cpp 165
RttStgReader.h 166 */
Scheduler.h 167 get_max_bus_numbers for all configured I2C buses
Scheduler.cpp 168 */
SoCond.cpp 169 uint32_t I2CDeviceManager::get_bus_mask(void) const
TERMINAL terminal /Users/scie/Documents/airplane/ardupilot/libraries/AP_HAL_ESP32/I2CDevice.cp
```