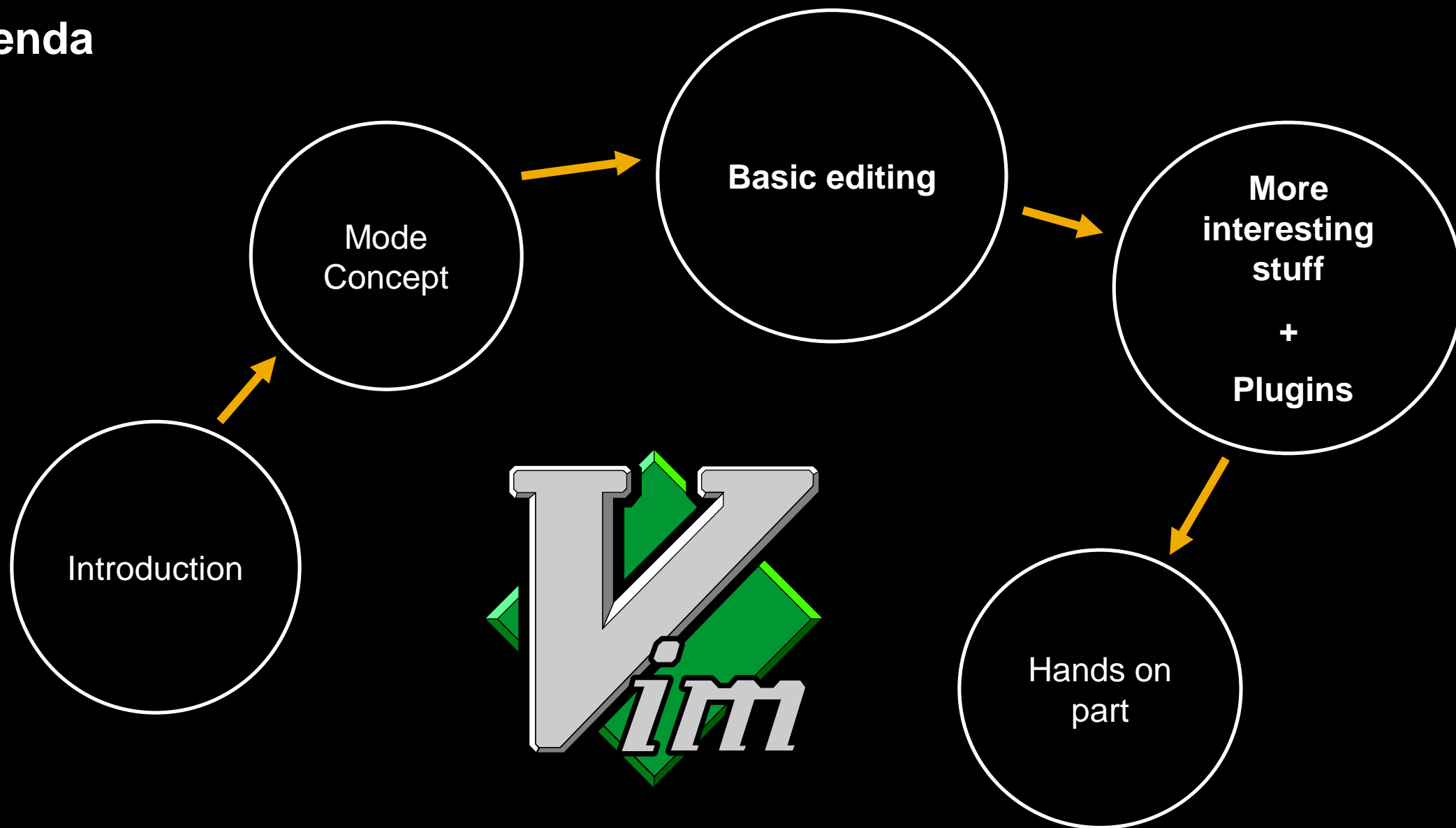


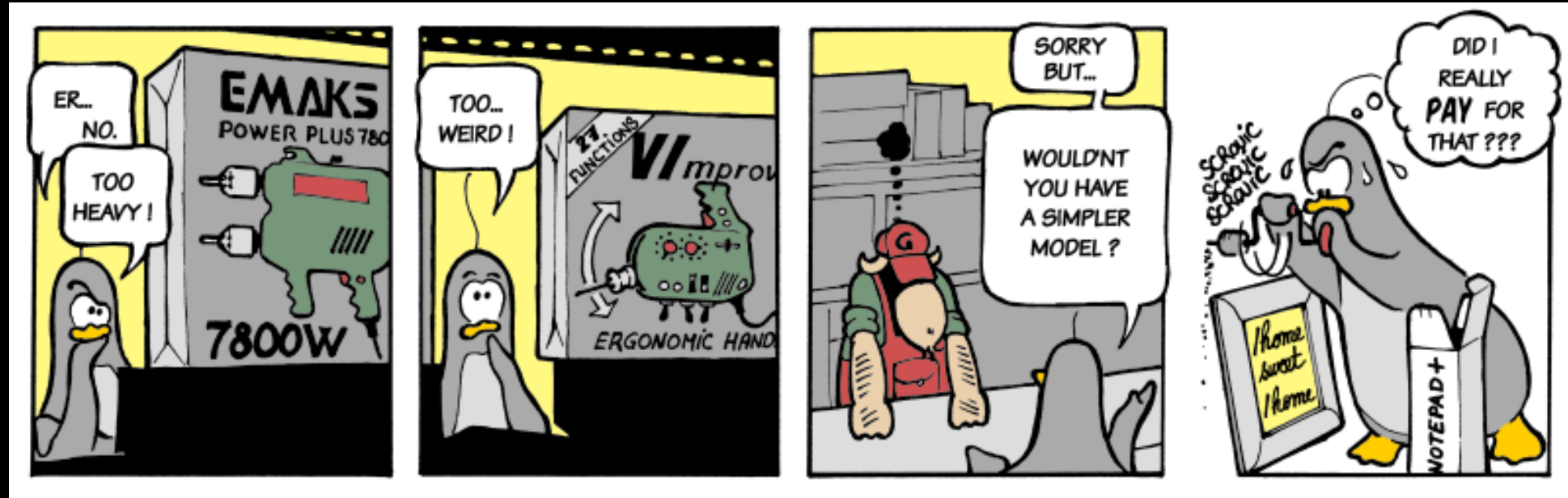


The most powerful IDE

Agenda



What is Vim? (or Vi IMproved)




“Vim is a highly configurable text editor built to enable efficient text editing.”

“Vim isn't an editor designed to hold its users' hands.





It is a tool, the use of which must be learned.”

What does the world think about Vim?

 **stackoverflow**

QuestionsDeveloper JobsTagsUsers



Search...




Log InSign Up

How to exit the Vim editor?

Ask Question


[Get started](#)

asked 5 years, 8 months ago
viewed 1,515,749 times
active 25 days ago




I'm stuck and cannot escape. It says:

2514"type :quit<Enter> to quit VIM"




But when I type that it simply appears in the object body.




vimvi

644share improve this question


edited Nov 3 '16 at 20:26


 **Peter Mortensen**
12.3k ● 18 ● 81 ● 107

asked Aug 6 '12 at 12:25

 **jclancy**
11.9k ● 5 ● 20 ● 28

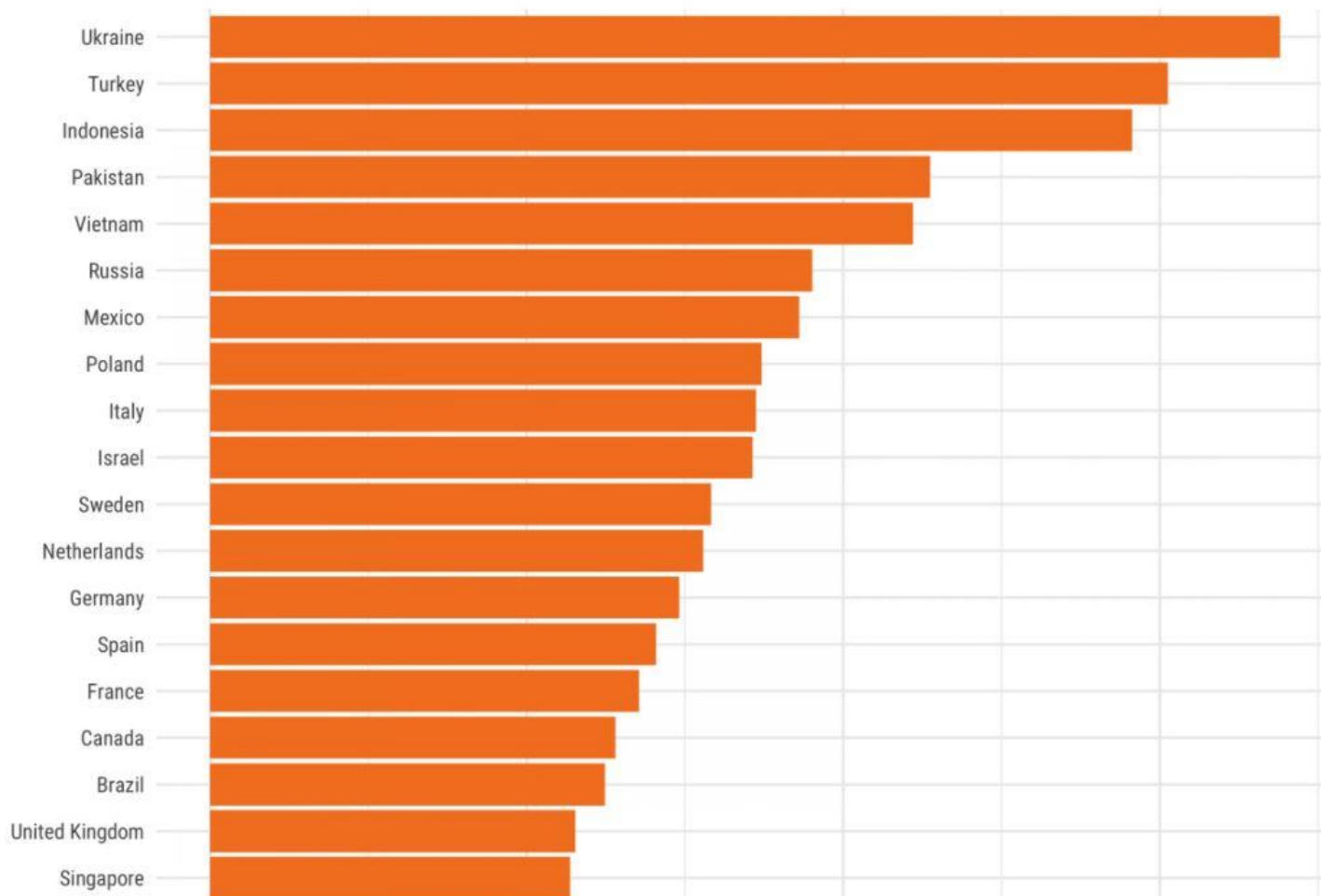
Want a C++ job?

C++ Developer for our realtime 3D rendering plugin
Enscape GmbH  Karlsruhe, Deutschland
RELOCATION
C++ user-interface

Software Developer (m/f) Strike Team
Gameforge AG  Karlsruhe, Deutschland

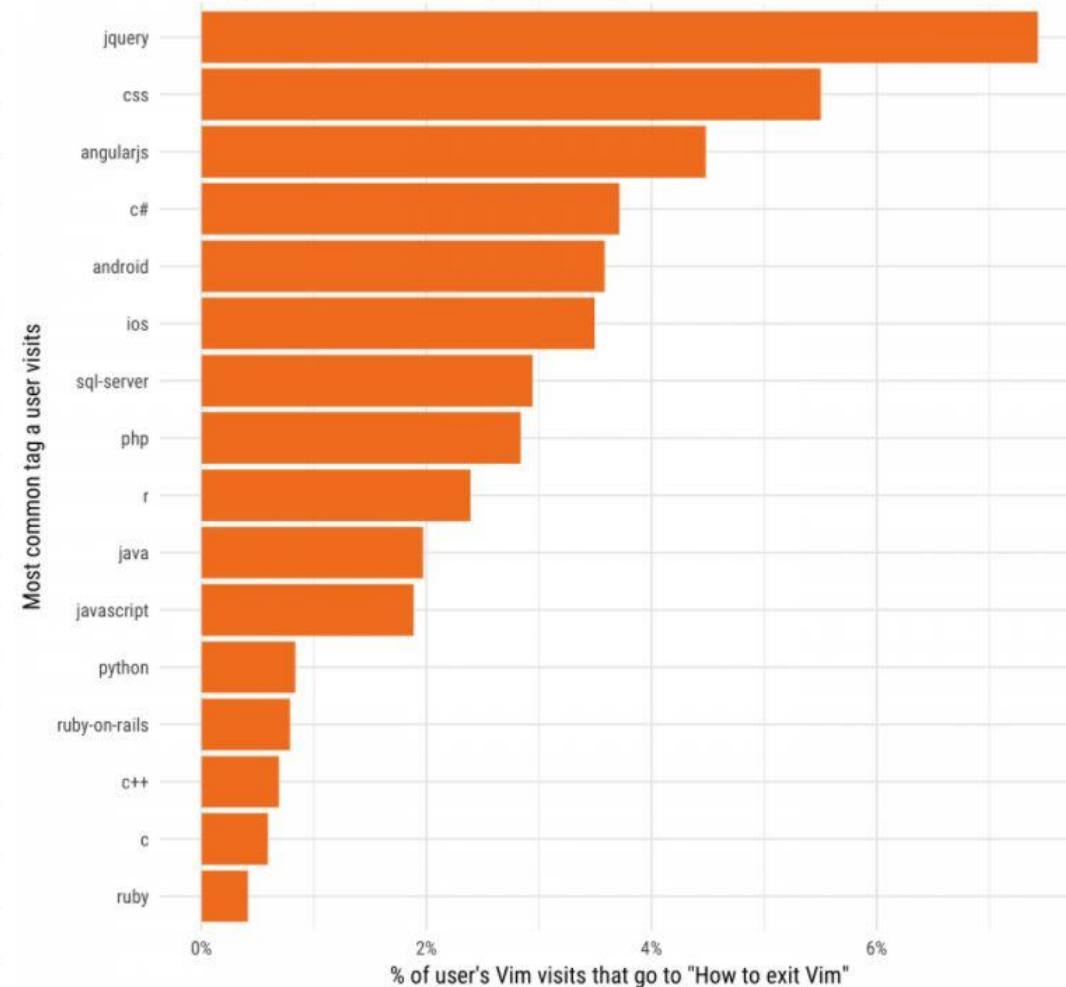
What countries are getting stuck in Vim?

Measured by a % of all Vim visits in one year of traffic. Only countries with >100 million visits in that year.



What developers are most likely to get stuck in Vim?

Divided by tag the user visited most frequently, showing the 16 most common such tags.



▲
3534

Hit the `Esc` key to enter "Normal mode". Then you can type `:` to enter "Command-line mode". A colon (`:`) will appear at the bottom of the screen and you can type in one of the following commands. To execute a command, press the `Enter` key.



- `:q` to quit (short for `:quit`)
- `:q!` to quit without saving (short for `:quit!`)
- `:wq` to write and quit
- `:wq!` to write and quit even if file has only read permission (if file does not have write permission: force write)
- `:x` to write and quit (similar to `:wq` , but only write if there are changes)
- `:exit` to write and exit (same as `:x`)
- `:qa` to quit all (short for `:quitall`)
- `:cq` to quit without saving and make Vim return non-zero error (i.e. exit with error)

You can also exit Vim directly from "Command mode" by typing `zz` to save and quit (same as `:x`) or `zq` to just quit (same as `:q!`). (Note that case is important here. `ZZ` and `zz` do not mean the same thing.)

Vim has extensive help - that you can access with the `:help` command - where you can find answers to all your questions and a tutorial for beginners.

But who the hell had this idea?



Bill Joy created Vi

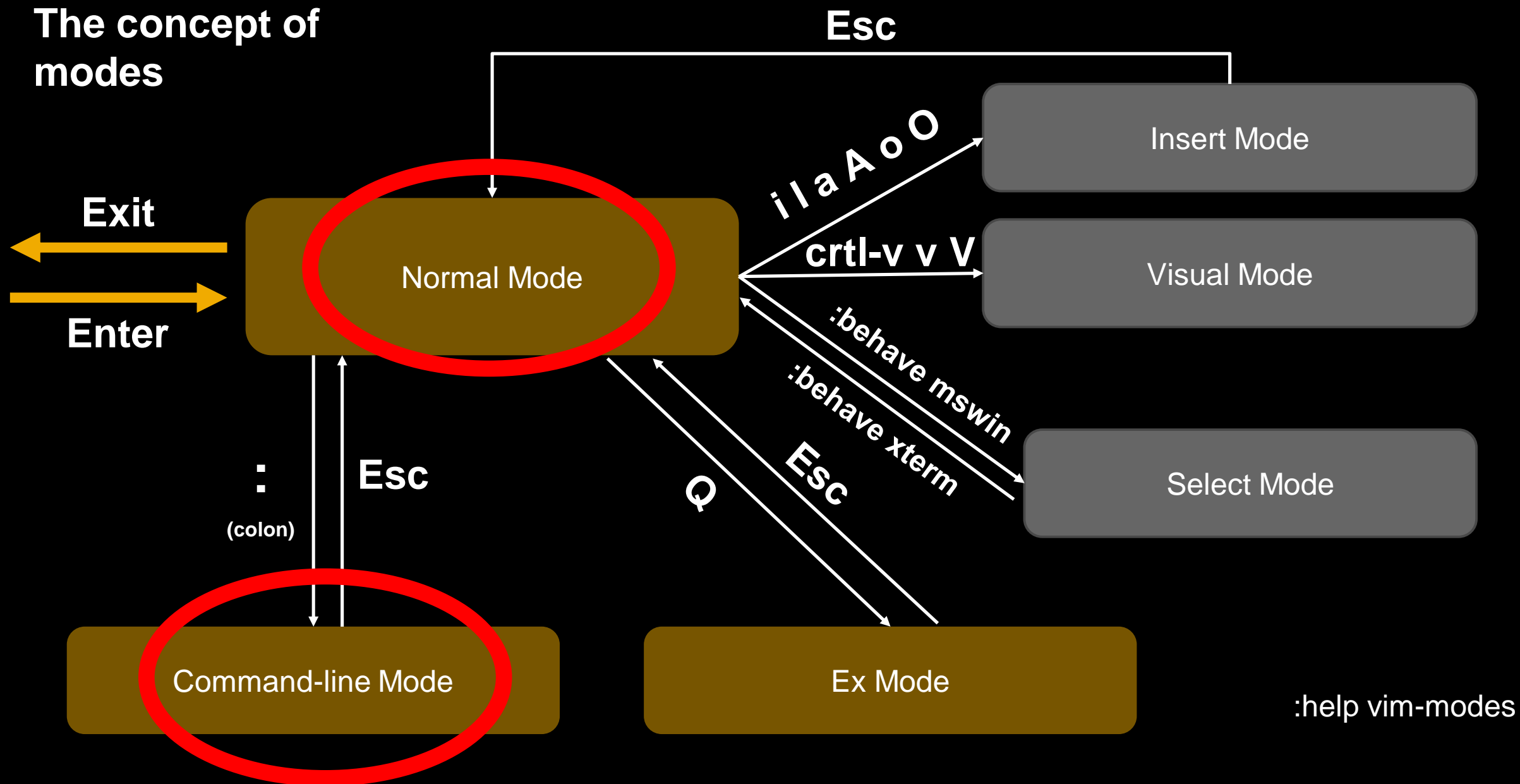
1976



Bram Moolenaar created Vim

1991

The concept of modes

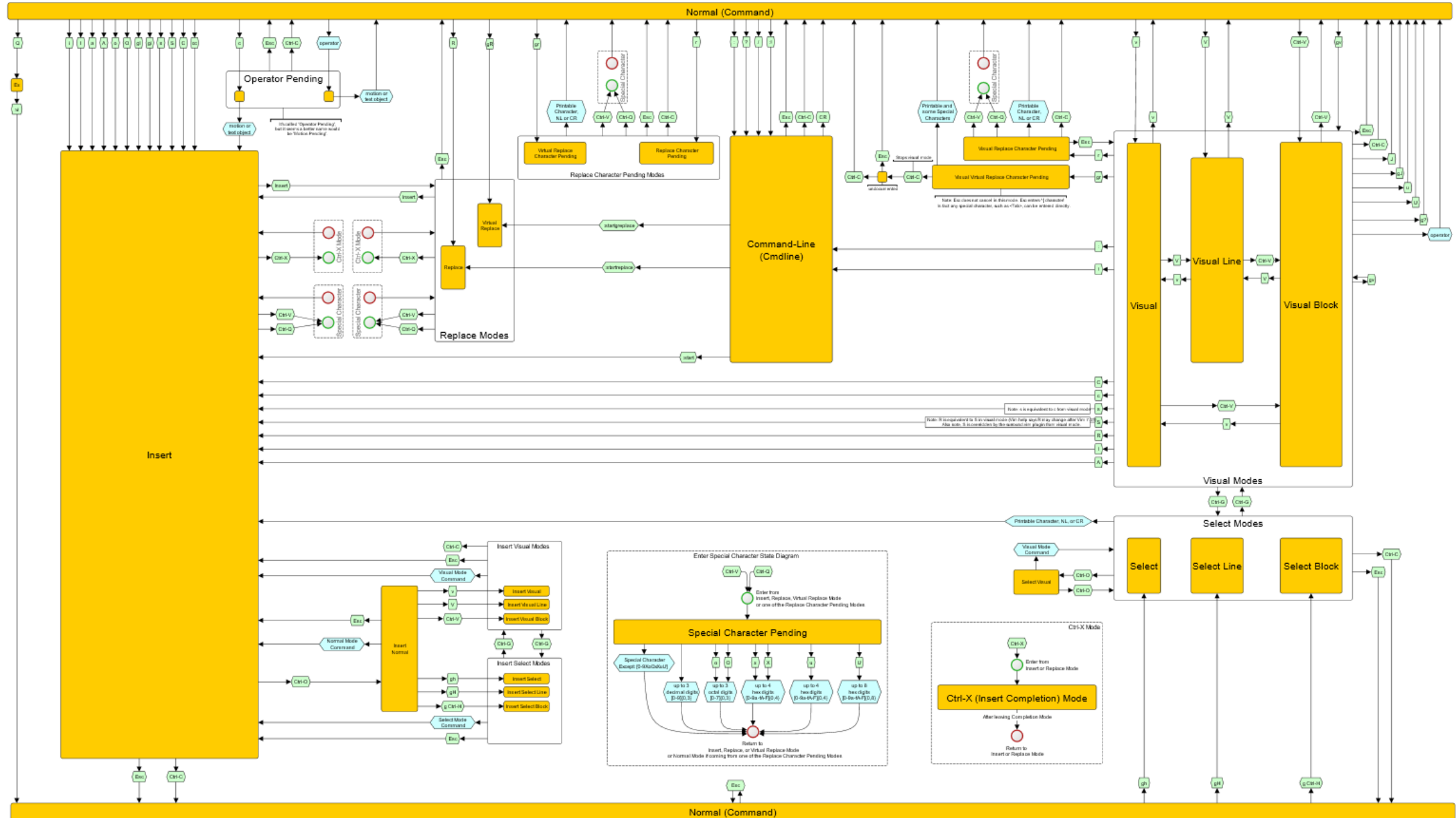


VIM Modes Transition Diagram

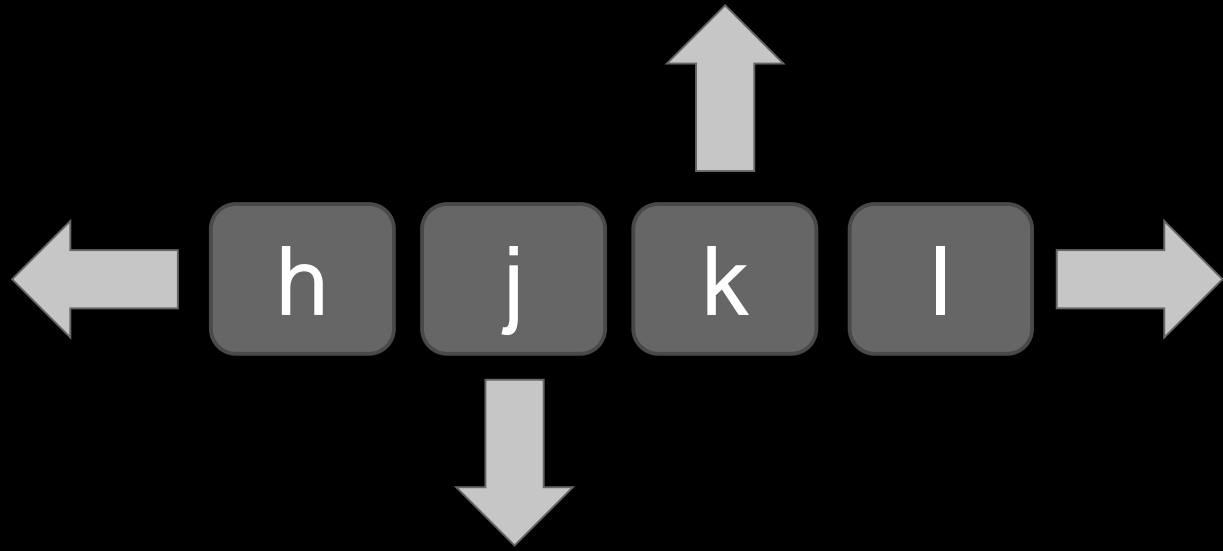
Note(s)

Exc — Equivalent — CH₂

By darcyparker@gmail.com Draft (3/19/2012. But not released as a new draft) Feedback welcome



In Normal Mode



w	- to the beginning of the next word
e	- to the end of the word
b	- to the beginning of the previous word
0	- beginning of the Line
\$	- end of the Line
1G	- first Line
#G	- Line Number #
G	- Last Line

[changing-command] [number / option] move-command

d - Delete

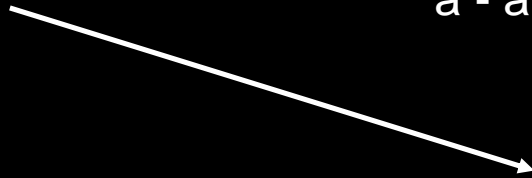
y - Yank

c - change

i - inner

a - all

p P - paste



Vim is a **programmers** editor

: 5, 10 fo - fold from line 5 to 10

za - toggle fold

>> << - indent line

=i{ - auto indentation of the inner {...} part

% - jump to corresponding brackets

* # - find symbol under curser

gd - goto definition (of variable etc.)

/sometext - search for some Text

:%s/old/new/gc - replace global in this file with confirmation

:find filename - fuzzy finder for files

ctags

[Generates index file of names found in source code]

<C-]> goto definition (push on stack)

:stag <C-W]> show definition in split

g] show definitions

<C-t> pop from tag stack

:tags show list of used tags

Handle multiple files in windows

Ctrl-w v

- vertical split

Ctrl-w s

- horizontal split

Ctrl-w h / j / k / l

- move between splits

Ctrl-w H / J / K / L

- move splits

Ctrl-q

- quit a split

:tabnew

- create new tab

gt

- go to next tab

gT

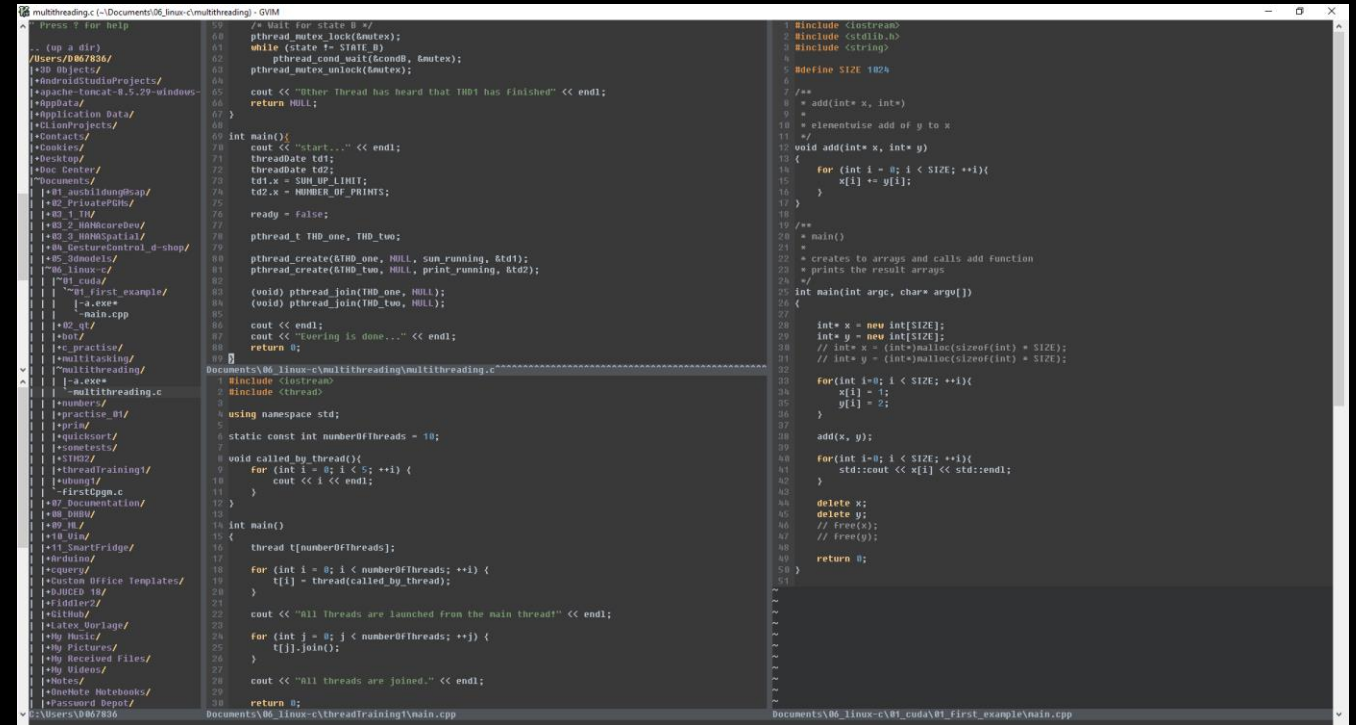
- go to previous tab

:Ex

- file tree

:e filename

- open file



```
multithreading.c
1 #include <iostream>
2 #include <thread>
3 using namespace std;
4 static const int numberOfThreads = 10;
5 void called_by_thread(){
6     for (int i = 0; i < 5; ++i) {
7         cout << i << endl;
8     }
9 }
10 int main()
11 {
12     thread t[numberOfThreads];
13     for (int i = 0; i < numberOfThreads; ++i) {
14         t[i] = thread(called_by_thread);
15     }
16     cout << "All threads are launched from the main thread!" << endl;
17     for (int j = 0; j < numberOfThreads; ++j) {
18         t[j].join();
19     }
20     cout << "All threads are joined." << endl;
21     return 0;
22 }
```

```
main.cpp
1 #include <iostream>
2 #include <stdlib.h>
3 #include <string>
4 #define SIZE 1024
5
6 /**
7  * add(int* x, int*)
8  * elementwise add of y to x
9  */
10 void add(int* x, int* y)
11 {
12     for (int i = 0; i < SIZE; ++i){
13         x[i] += y[i];
14     }
15 }
16
17 /**
18  * main()
19  * creates arrays and calls add function
20  * prints the result arrays
21  */
22 int main(int argc, char* argv[])
23 {
24     int* x = new int[SIZE];
25     int* y = new int[SIZE];
26     // int* x = (int*)malloc(sizeof(int) * SIZE);
27     // int* y = (int*)malloc(sizeof(int) * SIZE);
28     for (int i=0; i < SIZE; ++i){
29         x[i] = 1;
30         y[i] = 2;
31     }
32     add(x, y);
33     for (int i=0; i < SIZE; ++i){
34         std::cout << x[i] << std::endl;
35     }
36     delete x;
37     delete y;
38     // free(x);
39     // free(y);
40     return 0;
41 }
```

You want to know where you are:

Ctrl-g

:help <whatever>

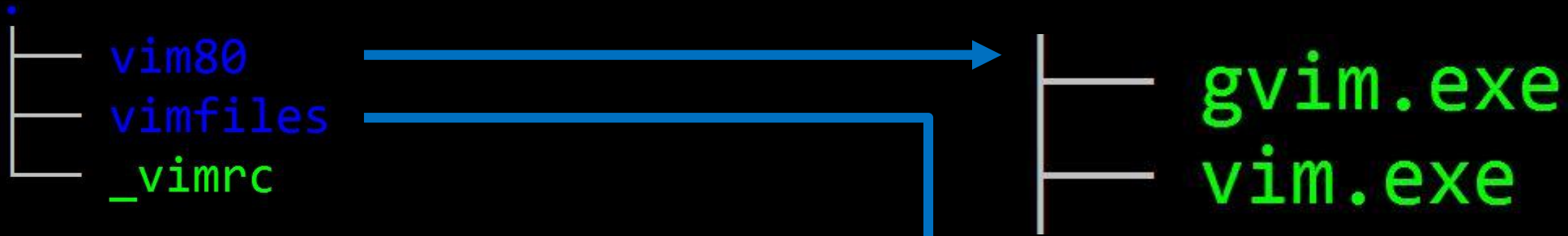
:help tutor



Let's make
Vim an IDE.

Folder Structure

```
d067836@WDFN33986757A:/mnt/c/Program Files (x86)/Vim$ tree -L 1
```



Vim Version 8.0 → **vim80** folder contains the tools

vimfiles folder contains all the plugins and addons

_vimrc file contains all your settings and modifications

```
dr-xr-xr-x 2 root root 0 Apr 17 18:16 autoload
dr-xr-xr-x 2 root root 0 Apr 23 09:26 bundle
dr-xr-xr-x 2 root root 0 Apr 17 16:22 colors
dr-xr-xr-x 2 root root 0 Apr 17 16:22 compiler
dr-xr-xr-x 2 root root 0 Apr 25 12:04 doc
dr-xr-xr-x 2 root root 0 Apr 17 16:22 ftdetect
dr-xr-xr-x 2 root root 0 Apr 17 16:22 ftplugin
dr-xr-xr-x 2 root root 0 Apr 17 16:22 indent
dr-xr-xr-x 2 root root 0 Apr 17 16:22 keymap
dr-xr-xr-x 2 root root 0 Apr 17 16:22 plugin
dr-xr-xr-x 2 root root 0 Apr 17 16:22 syntax
```

Let's talk about Plugins...

```
|~Documents/      1 #include <iostream>
| |+01_ausbildung@sap/ 2 #include <stdlib.h>
| |+02_PrivatePGMs/ 3 #include <string>
| |+03_1_TM/         4
| |+03_2_HANACoreDev/ 5 #define SIZE 1024
| |+03_3_HANASpatial/ 6
| |+04_GestureControl_d-shop/ 7 /**
| |+05_3dmodels/      8 * add(int* x, int*)
| |+06_linux-c/       9 *
| |~01_cuda/          10 * elementwise add of y to x
| |~01_first_example/ 11 */
| | |~-a.exe*         12 void add(int* x, int* y)
| | |~-main.cpp       13 {
| |+02_qt/            14     for (int i = 0; i < SIZE; ++i){
| |+bot/              15         x[i] += y[i];
| |+c_practise/       16     }
| |+multitasking/     17 }
| |+multithreading/   18
| |~-a.exe*          19 /**
| |~-multithreading.c 20 * main()
| |+numbers/          21 *
```

NERDTree

A more advanced
directory tree

```
1 # On branch master
2 # Your branch is up-to-date with 'origin/master'.
3 #
4 # Untracked files:
5 #   (use "git add <file>..." to include in what will be committed)
6 #
7 #       prototypes/diyDashboard_proto1/src/.main.cpp.swp
8 #
Documents\GitHub\diydrive\.git\index [Preview][R0]^000000000000000000000000
1 #include <QGuiApplication>
2 #include <QQmlApplicationEngine>
3
4 int main(int argc, char *argv[])
5 {
6     QGuiApplication app(argc, argv);
7
8     QQmlApplicationEngine engine;
9     engine.load(QUrl(QStringLiteral("qrc:/main.qml")));
10    if (engine.rootObjects().isEmpty())
11        return -1;
12
13    return app.exec();
Documents\GitHub\diydrive\prototypes\diyDashboard_proto1\src\main.cpp
:Gstatus
```

Vim-fugitive

Git integration

```
25 int main(int argc, char* argv[])
26 {
27
28     int* x = new int[SIZE];
29     int* y = new int[SIZE];
30     i|
~ int text
~ iswcntrl function
~ int_fast32_t class
~ isunordered function
~ intmax_t class
```

Vim-lsp and the language
server protocol

Autocompletion etc.

Some extras

```
6 #include <iostream>
7
8 /**
9  * main function
10 */
11 int main(int argc, char* argv){
12
13     std::cout << "Hello Vim World.\n" << std::endl;
14
15     return 0;
16 }
```

C:\Program Files (x86)\Vim\vim80\vimrun.exe

C:\WINDOWS\system32\cmd.exe /c (g++ Desktop\nix2\test.cpp ^&^& a.exe ^<

Hello Vim World.

Hit any key to close this window...

→ Snippets

→ Compile and run with one key press

Why you should use vim?

```
gvim -d pgm1.cpp pgm2.cpp
```

record with q<letter> and apply recording with @<letter>

:earlier 21m

...or use vim in intellij (_ideavimrc)

Thank you and happy vimming.

Contact information:

Lars Hübner

Autocompletion in vim

Language server protocol

- Install Language server
 - pip install python-language-server
- Install vim language server client
 - cd ~/path/to/vimfiles/bundle
 - git clone <https://github.com/prabirshrestha/async.vim.git>
 - git clone <https://github.com/prabirshrestha/vim-lsp.git>

git clone <https://github.com/vim-scripts/AutoComplPop.git>

```
if executable('pyls')  
    " pip install python-language-server  
    au User lsp_setup call lsp#register_server({  
        \ 'name': 'pyls',  
        \ 'cmd': {server_info->['pyls']},  
        \ 'whitelist': ['python'],  
        \ })  
  
    autocmd FileType python setlocal omnifunc=lsp#complete  
endif
```


C/C++ language server

git clone <https://github.com/cquery-project/cquery> --single-branch --depth=1

Install <https://www.visualstudio.com/downloads/#build-tools-for-visual-studio-2017>

Install python 3

Install 7zip

cd cquery

git submodule update --init

python waf configure build

add \cquery\build\release\bin to environmental variables

Sources:

<https://upload.wikimedia.org/wikipedia/commons/9/9f/Vimlogo.svg>

<https://rawgit.com/darcyparker/1886716/raw/eab57dfe784f016085251771d65a75a471ca22d4/vimModeStateDiagram.svg>

<https://stackoverflow.blog/2017/05/23/stack-overflow-helping-one-million-developers-exit-vim/>

https://de.wikipedia.org/wiki/Bill_Joy

https://en.wikipedia.org/wiki/Bram_Moolenaar

<https://www.karlsruhe.dhbw.de>