modes

VimConf 2018
2018-11-24
Tatsuhiro Ujihisa
"modes" talk agenda (1/2)

- I talk about Vim modes
- I don't talk about anything other than Vim modes
  - No Vim scripting
  - No vital.vim (obviously)
  - No vim plugins
  - No asynchronous processing
  - No cooking
  - No Cities: Skylines
  - No Civilization 5 or 6
  - No Minecraft or 7 days to die
"modes" talk agenda (2/2)

● Understand what Vim's modes are
  ○ by the specification and implementation
  ○ Introduce how they are and how they work

● Tools I use today
  ○ GDB
  ○ termdebug.vim (built-in plugin)

● Target audience
  ○ Beginner and intermediate Vim users
  ○ Vim plugin authors
  ○ **Not** active Vim core developers
"modes" talk agenda

- Understand what Vim's modes are
  - by the specification and implementation
  - Introduce how they are and how they work

- Tools I use today
  - GDB
  - termdebug.vim (built-in plugin)

- Target audience
  - Beginner and intermediate Vim users
  - Vim plugin authors
  - Not active Vim core developers

Goals:
- feel more confident at Vim core
- start working on contributing vim core
Understanding specification

Let's see the doc first
$ vimtutor

- Have you done?
  - /usr/bin/vimtutor
  - 25-30 minutes
- Not good at English?
  - vimtutor ja

Welcome to the VIM Tutor - Version 1.7

Vim is a very powerful editor that has many commands, too many to explain in a tutor such as this. This tutor is designed to describe enough of the commands that you will be able to easily use Vim as an all-purpose editor.

The approximate time required to complete the tutor is 25-30 minutes, depending upon how much time is spent with experimentation.

ATTENTION:
The commands in the lessons will modify the text. Make a copy of this file to practice on (if you started "vimtutor" this is already a copy).

It is important to remember that this tutor is set up to teach by use. That means that you need to execute the commands to learn them properly. If you only read the text, you will forget the commands!

Now, make sure that your Caps-Lock key is NOT depressed and press the \textbackslash j key enough times to move the cursor so that lesson 1.1 completely fills the screen.

Lesson 1.1: MOVING THE CURSOR

**To move the cursor, press the h,j,k,l keys as indicated.**

\textasciitilde
^k
< h l >
j\textasciitilde

| 1. Move the cursor around the screen until you are comfortable. |
|-------------|-----------------|
| 2. Hold down the down key (j) until it repeats. |
| Now you know how to move to the next lesson. |
| 3. Using the down key, move to lesson 1.2. |

NOTE: If you are ever unsure about something you typed, press <ESC> to place you in Normal mode. Then retype the command you wanted.

NOTE: The cursor keys should also work. But using hikl you will be able to
$ vimtutor

- Have you done?
  - /usr/bin/vimtutor
  - 25-30 minutes
- Not good at English?
  - vimtutor ja

Welcome to the VIM Tutor — Version 1.7

Vim is a very powerful editor that has many commands, too many to explain in a tutor such as this. This tutor is designed to describe enough of the commands that you will be able to easily use Vim as an all-purpose editor.

The approximate time required to complete the tutor is 25-30 minutes, depending upon how much time is spent with experimentation.

ATTENTION:
The commands in the lessons will modify the text. Make a copy of this file to practice on (if you started "vimtutor" this is already a copy).

It is important to remember that this tutor is set up to teach by use. That means that you need to execute the commands to learn them properly. If you only read the text, you will forget the commands!

Now, make sure that your Caps-Lock key is NOT depressed and press the \texttt{j} key enough times to move the cursor so that lesson 1.1 completely fills the screen.

Lesson 1.1: MOVING THE CURSOR

** To move the cursor, press the \texttt{h}, \texttt{j}, \texttt{k}, \texttt{l} keys as indicated. **

\texttt{\textasciitilde}  \texttt{k}

Hint: The \texttt{h} key is at the left and moves left. The \texttt{k} key is at the left and moves right. The \texttt{j} key looks like a down arrow.

The \texttt{l} key is at the right and moves right.

Put something you typed, press \texttt{ESC} to place the cursor, and retype the command you wanted.

NOTE: The cursor keys should also work. But using \texttt{hikl} you will be able to
Lesson 1.4: TEXT EDITING - INSERTION

- **Press i to insert text.**
  - press `<ESC>` to return to Normal mode.
- Next lesson is about A
:help

- :h (without arguments)
  - It opens help.txt, the index of helps
- help.txt → (intro.txt | howto.txt) → mode-switching
- :h mode-switching
  - (or :h mode-s)
6. Switching from mode to mode

If for any reason you do not know which mode you are in, you can always get back to Normal mode by typing `<Esc>` twice. This doesn't work for Ex mode though, use `:visual`.

You will know you are back in Normal mode when you see the screen flash or hear the bell after you type `<Esc>`. However, when pressing `<Esc>` after using CTRL-O in Insert mode you get a beep but you are still in Insert mode, type `<Esc>` again.

<table>
<thead>
<tr>
<th>FROM mode</th>
<th>Normal</th>
<th>Visual</th>
<th>Select</th>
<th>Insert</th>
<th>Replace</th>
<th>Cmd-line</th>
<th>Ex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>v</td>
<td>V</td>
<td>*4</td>
<td>*1</td>
<td>R</td>
<td>gR</td>
<td>:</td>
</tr>
<tr>
<td>Visual</td>
<td>*2</td>
<td>^G</td>
<td>c</td>
<td>C</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>Select</td>
<td>*5</td>
<td>^0</td>
<td>^G</td>
<td>*6</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>Insert</td>
<td><code>&lt;Esc&gt;</code></td>
<td>--</td>
<td>--</td>
<td><code>&lt;Insert&gt;</code></td>
<td>--</td>
<td>---</td>
<td></td>
</tr>
</tbody>
</table>

**i_esc**
<table>
<thead>
<tr>
<th>FROM mode</th>
<th>TO  mode</th>
<th>Normal</th>
<th>Visual</th>
<th>Select</th>
<th>Insert</th>
<th>Replace</th>
<th>Cmd-line</th>
<th>Ex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>Normal</td>
<td>v</td>
<td>V</td>
<td>^V</td>
<td>*4</td>
<td>*1</td>
<td>R</td>
<td>gR</td>
</tr>
<tr>
<td>Visual</td>
<td>Visual</td>
<td>^G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select</td>
<td>Select</td>
<td>^0</td>
<td></td>
<td>^G</td>
<td>*6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert</td>
<td>Insert</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Replace</td>
<td>Replace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*&lt;Insert&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command-line</td>
<td>Command-line</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>:start</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ex</td>
<td>Ex</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO mode</td>
<td>Normal</td>
<td>Visual</td>
<td>Select</td>
<td>Insert</td>
<td>Replace</td>
<td>Cmd-line</td>
<td>Ex</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>---------</td>
<td>----------</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>v V ^V</td>
<td>*4</td>
<td>*1</td>
<td></td>
<td>R gR</td>
<td>: / ? !</td>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>Visual</td>
<td>*2</td>
<td>^G</td>
<td>c C</td>
<td></td>
<td></td>
<td>:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select</td>
<td>*5</td>
<td>^0 ^G</td>
<td>*6</td>
<td></td>
<td></td>
<td>:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert</td>
<td>&lt;Esc&gt;</td>
<td></td>
<td></td>
<td>&lt;Insert&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Replace</td>
<td>&lt;Esc&gt;</td>
<td></td>
<td></td>
<td>&lt;Insert&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command-line</td>
<td>*3</td>
<td></td>
<td></td>
<td>:start</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ex</td>
<td>:vi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*1 Go from Normal mode to Insert mode by giving the command "i", "I", "a", "A", "o", "O", "c", "C", "s" or S."
<table>
<thead>
<tr>
<th>FROM mode</th>
<th>Normal</th>
<th>Visual</th>
<th>Select</th>
<th>Insert</th>
<th>Replace</th>
<th>Cmd-line</th>
<th>Ex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>v V ^V</td>
<td>*4</td>
<td>*1</td>
<td>R gR</td>
<td>: / ? !</td>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>Visual</td>
<td>*2</td>
<td>^G c C</td>
<td>--</td>
<td></td>
<td>:</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>Select</td>
<td>*5</td>
<td>^0 ^G</td>
<td>*6</td>
<td></td>
<td></td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>Insert</td>
<td>&lt;Esc&gt;</td>
<td>---</td>
<td>---</td>
<td>&lt;Insert&gt;</td>
<td></td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>Replace</td>
<td>&lt;Esc&gt;</td>
<td>---</td>
<td>---</td>
<td>&lt;Insert&gt;</td>
<td></td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>Command-line</td>
<td>*3</td>
<td>---</td>
<td>---</td>
<td>:start</td>
<td></td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>Ex</td>
<td>:vi</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td>---</td>
<td></td>
</tr>
</tbody>
</table>

*2 Go from Visual mode to Normal mode by giving a non-movement command, which causes the command to be executed, or by hitting `<Esc>` "v", "V" or "CTRL-V" (see v_v), which just stops Visual mode without side effects.
<table>
<thead>
<tr>
<th>FROM mode</th>
<th>Normal</th>
<th>Visual</th>
<th>Select</th>
<th>Insert</th>
<th>Replace</th>
<th>Cmd-line</th>
<th>Ex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>v V ^V *4 *1</td>
<td>R gR : / ? ! Q</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visual</td>
<td>*2</td>
<td>^G c C</td>
<td></td>
<td></td>
<td></td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Select</td>
<td>*5</td>
<td>^0 ^G *6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert</td>
<td>&lt;Esc&gt;</td>
<td>--</td>
<td>--</td>
<td></td>
<td>&lt;Insert&gt;</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>Replace</td>
<td>&lt;Esc&gt;</td>
<td>--</td>
<td>--</td>
<td></td>
<td>&lt;Insert&gt;</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>Command-line</td>
<td>*3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>:start</td>
<td></td>
</tr>
<tr>
<td>Ex</td>
<td>:vi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*3 Go from Command-line mode to Normal mode by:
- Hitting <CR> or <NL>, which causes the entered command to be executed.
- Deleting the complete line (e.g., with CTRL-U) and giving a final <BS>.
- Hitting CTRL-C or <Esc>, which quits the command-line without executing the command.

In the last case <Esc> may be the character defined with the 'wildchar' option, in which case it will start command-line completion. You can ignore that and type <Esc> again. (Vi: when hitting <Esc> the command-line is executed. This is unexpected for most people; therefore it was changed in Vim. But when the <Esc> is part of a mapping, the command-line is executed. If you want the Vi behaviour also when typing <Esc>, use ":cmap ^V<Esc> ^V^M".)
More modes (:h vim-modes)

- Normal
- Visual
- Select
- Insert
- Replace
- Command-line (Cmdline)
- Ex
- Terminal-Job
- Operator-pending
- Virtual Replace
- Insert Normal
- Terminal-Normal
- Insert Visual
- Insert Select
Short summary

- vimtutor
- :h
- 7 modes + 7 additional modes
- Many ways switch between modes
- Each key behaviour depend on the current mode
Vim

Insert mode
Normal mode
Visual mode
Operator-pending mode

Key mapping
Vim script
Command functions

Vim.h
normal.c
vim.c
edit.c
globals.h

Implementation
Bundled plugins
Syntax
Vital VIM

Contribution
Development

(modify model) to
:CTRL-D
or
not

(add jobs)
:term
Vim

Insert mode
Normal mode
Visual mode
Operator pending mode
Window
Command functions
Command mapping
Vim script
Vim.h
Vim.c
Global.h
Editor
Syntax
Implementation
Bundled plugins
Vim
Vim.c
Vim.h

modify model to "CTRL" or not

return :CTRL

add job

Contribution/Development

These
24

we

finished
Understanding specification implementation

Let's see the Vim C code now
https://github.com/vim/vim

- `src/**/*.c`
- everything is there
https://github.com/ujihisa

- Vancouver, Canada
- Tokyo, Japan
- Vim for about 20 years
- Ruby on Rails, Scala for distributed systems, Clojure, Haskell for myself, and Vim script
- VimConf founder
- ujihisa.vim
(unorganized) 115 plugins I use

aggit.vim
aldmeris
alt
ansible-yaml
asterisk
autodirmake.vim
calendar.vim
caw.vim
coffee-script
colors-pencil
colors-solarized
Colour-Sampler-Pack
concealedyank.vim
context_filetype.vim
cpp
cruby
cursorword
deolete.nvim
elixir
filetype-haskell
fontzoom
ft-clojure
ft-cmake
ft-mongo
game-code-break
game_engine.vim
ghcmod
gina.vim
go

groovyindent
haskell.vim
hug-neovim-rpc
ifer/vim
incsearch.vim
J6uil.vim
javaclasspath
jplus
kotlin
lexima.vim
linediff.vim
ltsv
mario.vim
metaffer
monokai
nclipper.vim
neco
neco-ghc
neco-syntax
.neobundle
neobundle.vim
neobundle.vim
neobundle-vim-recipes
neochat.vim
neocljoure.vim
neoinclude.vim
neomru.vim
neopairs.vim
neosnippet
neosnippet-snippets
neverland-vim-theme
nvim-yarp
open-browser.vim
operator-replace
operator-user
paper-color-theme
perfect.vim
postlist
prettyprint
puyo.vim
quicklearn
quickrun
Rainbow-Parentheses-Bundle
ref
ref-hoogle
ref-ri
rengbang
reversi.vim
ruby
scala
sexp
showtime
smartch
sudo.vim
surround	apagebuffer.vim
tabpaged

textobj-syntax
textobj-user
textobj-wiw

themis
translua
typescript
typedef
typescript
unite
unite-build
unite-colorscheme
unite-giti
unite-haskellimport
unite-help
unite-history
unite-included
unite-javaimport
unite-locate
unite-outline
unite-ruby-require.vim
unite-ssh
unite.vim
viml
vimlinter
vimlparser
vimimport
vimproc
vimshell
vimshell-ssh
vital.vim
zenesque.vim
(unorganized) 115 plugins I use

https://github.com/ujihisa/config

vimrc: 2772 lines (not organized at all)
(unorganized) 115 plugins I use

https://github.com/ujihisa/config

vimrc: 2772 lines (not organized at all)
working at quipper (2018-09 ~)

- Ruby on Rails
- React and Redux
- https://www.quipper.com/
- Education service
https://github.com/ujihisa

- Vancouver, Canada
- Tokyo, Japan
- Vim for about 20 years
- **Ruby** on Rails,
  **Scala** for distributed systems,
  **Clojure, Haskell** for myself,
  and **Vim script**
- VimConf founder / staff
  ujihisa.vim
Thanks a lot for Vim and its development ecosystem. I've been living with the Vim community.
https://github.com/ujihisa

- Vancouver, Canada
  Tokyo, Japan
- Vim for about 20 years
- **Ruby** on Rails,
  **Scala** for distributed systems,
  **Clojure, Haskell** for myself,
  and **Vim script**
- VimConf founder / staff
  ujihisa.vim
How to start

- Read C code from top to bottom
- Find a specific function, and read it carefully
- Run, and see what debugger shows
  - GDB
Understanding implementation

Let's see the Vim C code now
GDB
See the behaviour dynamically
GDB

- The GNU Project Debugger
- "go inside a running program"
- Run Vim with GDB
  - The Vim needs to be compiled with some special flags
Build a GDB-Friendly Vim

- git clone from github.com/vim/vim
- edit src/Makefile manually
  - CFLAGS=-ggdb3 to include debug info
    - -O0 to disable optimization
    - -g to include debug info
    - -g2 for more
    - -g3 for even more
    - -ggdb3 for even more just for gdb
    - (See `man gcc` for details)
  - STRIP=/bin/true not to strip (/usr/bin/true for mac)
Build a GDB-Friendly Vim

```bash
diff --git a/src/Makefile b/src/Makefile
index 5b25e033f..0e3051a26 100644
--- a/src/Makefile
+++ b/src/Makefile
@@ -591,7 +591,7 @@ CLink = $(CC)
 # When using -g with some older versions of Linux you might get a
 # statically linked executable.
 # When not defined, configure will try to use -O2 -g for gcc and -O for cc.
-#CFLAGS = -g
+CFLAGS = -ggdb3
+CFLAGS = -0

 # Optimization limits - depends on the compiler. Automatic check in configure
@@ -1005,7 +1005,7 @@ TOOLS = xxd/xxd$(EXEEXT)
 ### prefix the top directory for the data (default "/usr/local")
 #
 # Uncomment the next line to install Vim in your home directory.
-#prefix = $(HOME)
+prefix = $(HOME)/git/vim/local

 ### exec_prefix is the top directory for the executable (default $(prefix))
 #
@@ -1156,7 +1156,7 @@ INSTALL_DATA_R = cp -r

 ### Program to run on installed binary. Use the second one to disable strip.
 #STRIP = strip
-#STRIP = /bin/true
+STRIP = /bin/true

 ### Permissions for binaries {{{
 BINMOD = 755
```
:Termdebug

- Vim wrapper for GDB
  - powered by :terminal
  - made by Bram Moolenaar
  - (He introduced at his talk)
- Use vim to see the source code
- Use :terminal to run a debuggee program (i.e. Vim)
Termdebug

- Vim wrapper for GDB
  - Powered by :terminal
  - Made by Bram Moolenaar (introduced at his talk)

- Use vim to see the source code
- Use :terminal to run a debuggee program (i.e., Vim)

```
593 #ifdef USE_ON_FLY_SCROLL
594     dont_scroll = FALSE;    /* allow scrolling here */
595 #endif

597 #ifdef FEAT_EVAL
598     /* Set v:count here, when called from main() and not a stuff
599     * command, so that v:count can be used in an expression mapping
600     * when there is no count. Do set it for redo. */
601     if (toplevel && readbuf1_empty())
602         set_vcount_ca(&ca, &set_prevcount);
603 #endif

605     /* Get the command character from the user.
606     */
607     c = safe_vgetc();
608     c = safe_vgetc();
```
Short summary

- GDB: step execution / view code
- Termdebug: Vim in different window / code in Vim
- main() ->vim_main2()->main_loop()
- normal_cmd()
  - safe_vgetc()
- find_command() / nv_cmds[idx]
- nv_edit()
- edit()
- restart_edit
Contribution / Development

Let me show my example quickly
● Make pull requests to vim/vim
  ○ see existing issues first
  ○ vim-jp also has some
● ujihisa's contributions (vim-core)
  ○ https://github.com/vim/vim/search?q=author%3Aujihisa&unscoped_q=author%3Aujihisa&type=Issues
Provide a way to tell if a command is executed from i_CTRL-O or not #3000

ujihisa wants to merge 6 commits into vim:master from ujihisa:mode

Problem

A user-defined command cannot tell if it's executed as an Ex command, or as i_CTRL-O from insert mode. This makes it hard to implement some Vim plugins, such as one that inserts text into the buffer with adjusting contents depending on its context.

Example use case: https://github.com/koron/iferr :IfErr command inserts a Go code snippet at the
f_mode()

void

put values into rettv

char_u buf[4]

used for return value

rettv->vval.v_string = vim_strsave(buf);

buf[0]

'n', 'i', ...

mode() uses just this part

buf[1]

NULL, 'i', ...

buf[2]

NULL, 'I', 'R', or 'V'

currently only for C-o
https://github.com/ujihisa

- https://twitter.com/ujm

↑ I'll share my talk slides today

- Tatsuhiro Ujihisa
- Tokyo, Japan
  and
- Vancouver, Canada