Vi IMproved Basics

Marek Kozłowski

Faculty of Mathematics and Information Sciences
Warsaw University of Technology

Vi IMproved

- *Vim* (*Vi Improved*) is a *vi* clone with numerous extensions including:
  - portability to almost all platforms,
  - GUI interface using GTK+ library (*gvim*),
  - multi-language support,
  - smart syntax highlighting and spell checking,
  - support for extended regular expressions,
  - almost unlimited *undo* (up to 1000 commands),
  - mouse support (with and without GUI),
  - support for compressed files,
  - high extensibility via plug-ins,

  and many others.

Vim Operating Modes

- *Vim* modes of operation include:
  - *command mode* (default, can be accessed by pressing <Esc> anytime):
    - buffer edition commands,
    - file management and configuration commands (start with :),
    - shell commands (start with :!),
  - *insert mode* (editing a text),
  - *insert (paste) mode* (for copying a pre-formatted text from external sources),
  - *visual mode* (selecting characters),
  - *visual line mode* (selecting lines).

Ex Commands

- By default *vim* is started in command mode.
- Pressing the `<Esc>` key always switches to the command mode.
- Commands that start with `:` are so-called *ex* commands; they are usually used for file management, changing *vim* settings or displaying help.
- After typing `:` the up and down keys can be used for history.
- `%` in *ex* commands denotes currently edited (*this*) file name.
- Most *ex* commands are full words. If there is no ambiguity they can be abbreviated. For example: the command `:q` is a shortcut for `:quit`.

File Menu Commands

- Square brackets below denote optional parts (full command name).
- For practical reasons short forms are typically used.

`:q[uit]`	– quit (close current window)
`:w[rite]`	– save
`:w[rite]` *file_name*	– save as
`:wq`	– save and quit
`:q!`	– don't save and quit
`:e[dit]` *file_name*	– edit
`:e[dit] %`	– reload the file

Getting Help

- For better clarity we use the long form on this slide.
- Getting help:

`:h[elp]` – display help
`:help` *`command`* – help on a *vim* command

- Examples:

`:help :q` – help on `:q` command
`:help i` – help on `i` command
`:help :set` – help on `:set` command
`:help number` – help on `number` setting (see the next slide)

- For closing a help window press `:q`

Configuring *Vim*

- We mix both short and long settings' names here (as most *vim* users do).

`:set`	– display all settings
`:set spell`	– enable spell checking
`:set nospell`	– disable spell checking
`:set spelllang=pl`	– change spell checker language (to Polish);
`:set nu[mber]`	– enable line numbering
`:set nonu[mber]`	– disable line numbering
`:set enc=`*`encoding`*	– encoding for viewing; requires file reloading to take effect
`:set fenc=`*`encoding`*	– encoding for file saving (`fileencoding`)
`:set ts=`*`n`*	– tab size (`tabsize`)
`:set sw=`*`n`*	– indent size (`shiftwidth`)

Executing Shell Commands

- Commands that start with `:!` are executed by a shell.
- Examples:

`:!date`	– what time is it?
`:!./%`	– run *this* script from the working directory
`:!gcc %`	– compile *this* file using GNU C compiler
`:!man 3 printf`	– display man page on *printf*

Entering the Insert (Edition) Mode

- Any of the following commands switches to the insert mode:

`i` – insert before the cursor
`I` – insert at the beginning of the line
`a` – append after the cursor
`A` – appending at the end of the current line
`o` – append (open) a new line below the current line
`O` – append (open) a new line above the current line

- For returning to the command mode press `<Esc>`.

Basic Editing Commands

`x`	– delete a character
`r`	– replace a character
`.`	– repeat the latest command
`u`	– undo
`Ctrl-r`	– redo

- Numerous commands can be preceded by a number n to be executed n times. For example: `4x` (delete 4 characters), `2u` (undo twice).

Search Commands

- Whenever possible *vim* reuses commands form other utilities. Searching works exactly the same way is in `less`:

/pattern	– search forward for *pattern*
?pattern	– search backward for *pattern*
`n`	– next occurrence
`N`	– previous occurrence

- Regular expressions can be used to specify search patterns.
- Special characters in search patterns must be escaped with `\` to protect their literal meaning.
- After `/` or `?` up and down cursor keys can be used for history.
- Found items are highlighted. Turn off highlighting by `:nohl`.

Basic *Go To* Commands

`1G` – first line (see: `less`)
`G` – last line (see: `less`)
n`G` – line number *n* (see: `less`)
`0` – beginning of the line
`^` – first non-blank in the line (see: *regexp*)
`$` – end of the line (see: *regexp*)
`%` – matching parenthesis

- Instead of the cursor arrows the keys: `h`, `j`, `k` and `l` can be used. Those as well as navigation keys can be preceded by a number.

Other *Go To* Commands

- Some other useful *go to* commands include:

`b` – beginning of the current word
`B` – beginning of the current word with punctuation
`e` – end of the current word
`E` – end of the current word with punctuation
`w` – beginning of the next word
`W` – beginning of the next word with punctuation
`f`n – next occurrence of the character n
`t`n – 1 char before next occurrence of the character n
`F`n – previous occurrence of the character n
`T`n – 1 char after previous occurrence of the character n

Operations on Blocks

`d` – delete (cut) block
`dd` – delete (cut) current line
`y` – copy (yank) block
`yy` – copy (yank) current line
`p` – paste block (after the cursor)

- The commands `d` and `y` must be followed by some *go to* marker, for example: `d$` , `dG` , `d%` etc.
- `dd` and `yy` may be preceded by numbers.
- Note that *vim*'s clipboards (*registers*) are independent of desktop environments' clipboard! Use toolbar buttons for exchange with your desktop.

Operations on Blocks – Examples

`d$`	– delete form the cursor to the end of line
`d/sometext`	– delete to the next occurrence of *sometext*
`d%`	– delete text between this and matching parenthesis/brackets
`dG`	– delete all lines from the current one to the end of file
`ddp`	– swap lines
`yy10p`	– duplicate the current line 10 times
`dBd4W`	– delete four words (including the current one)

Visual and Visual Line Modes

- Blocks can be also selected by using visual and visual line modes:
  - `v` – start highlighting characters (enter visual mode),
  - `V` – start highlighting lines (enter visual line mode).
- Text for highlighting is selected by pressing navigation keys and/or 'go to' commands.
- Text can be highlighted by keeping the left mouse button pressed.
- In *gvim* double click allows normal/visual mode switching.
- If some block is highlighted then `d` and `y` operate on this block.

Operations on Blocks (2)

`~` – change case
`>` – shift (indent) `shiftwidth` right
`<` – shift (indent) `shiftwidth` left

- By default `shiftwidth` is equal to tab size, that is: 8.
- If `shiftwidth` is a multiple of tabs size then only tabs are inserted. Otherwise tabs are complemented with spaces.

String Substitution

`:`*ranges*`/`*text1*`/`*text2*`/`*options*

– substitute *text1* with *text2* in the *range* according to *options*

- The range may be specified as:
  - empty – current line only,
  - line number,
  - `%` – whole file.
- Allowed options (can be combined) are:
  - `g` – replace all occurrences (default: only the first one),
  - `c` – confirm each substitution,
  - `i` – ignore case for search pattern.

Notes on Copy&Paste

- Before copying a formatted text from an external source (using terminal's *paste*) switch to the so-called insert paste mode:

  ```
  :set paste
  i
  ```

- Default configuration in our labs disables the *copy* menu option in terminal for highlighted text in vim. You may change this behavior by executing:

  ```
  :set mouse-=a
  ```

Other *Vim* Features

- Recording macros.
- Custom markers.
- Using programming tags.
- Working with multiple files.
- Defining custom commands.
- Defining default actions.
- Working with numerous clipboards (registers).
- Spreadsheet mode.
- ...and many many others.

Vim as an IDE

`:!./%` – execute the script being editing

`:!gcc -Wall -pedantic %` – compile the C source being edited

`:!pdflatex %` – compile the LaTeX source being edited

- *taglist* (`http://vim-taglist.sourceforge.net`) plug-in implements source code browser for *vim*