

Text Editors

Chonnam National University
School of Electronics and Computer
Engineering

Kyungbaek Kim

Bunch of Editors

- Console-based text editors
 - vim
 - Vi IMproved
 - Improved version of “vi” editor
 - Emacs
 - Also has GUI-version
- GUI-based text editors
 - gedit
 - A GUI-text editor for GNOME
 - kedit
 - A GUI-text editor for KDE

Why do we need to learn text based editors?

- Sometimes, it is the only available editor
 - Maintenance and recovery operations almost never take place during X Window sessions.
 - When any bad thing happens, you can not log into the X Window system
- You can edit text files through remote shell session
 - Some servers will not host graphical desktops.
- Mouse movements slow down the touch-typist

VI and VIM

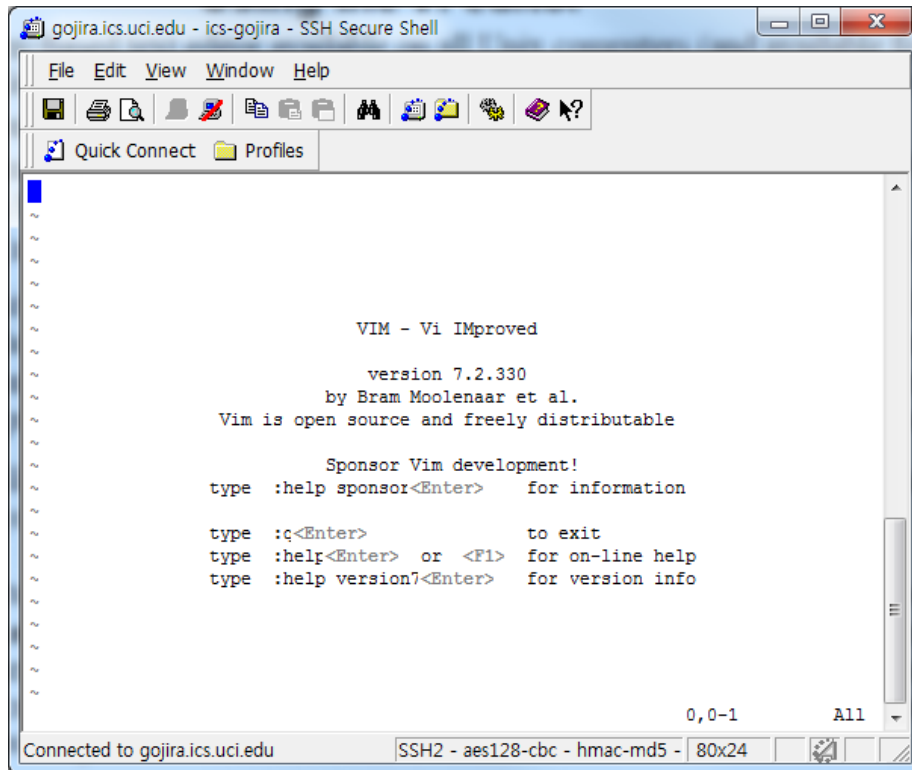
- A console-based text editor
 - Available on all UNIX/LINUX computers
 - Vim is usually aliased as vi in a shell
- How to start an editing session?

```
$ vi file.txt
```

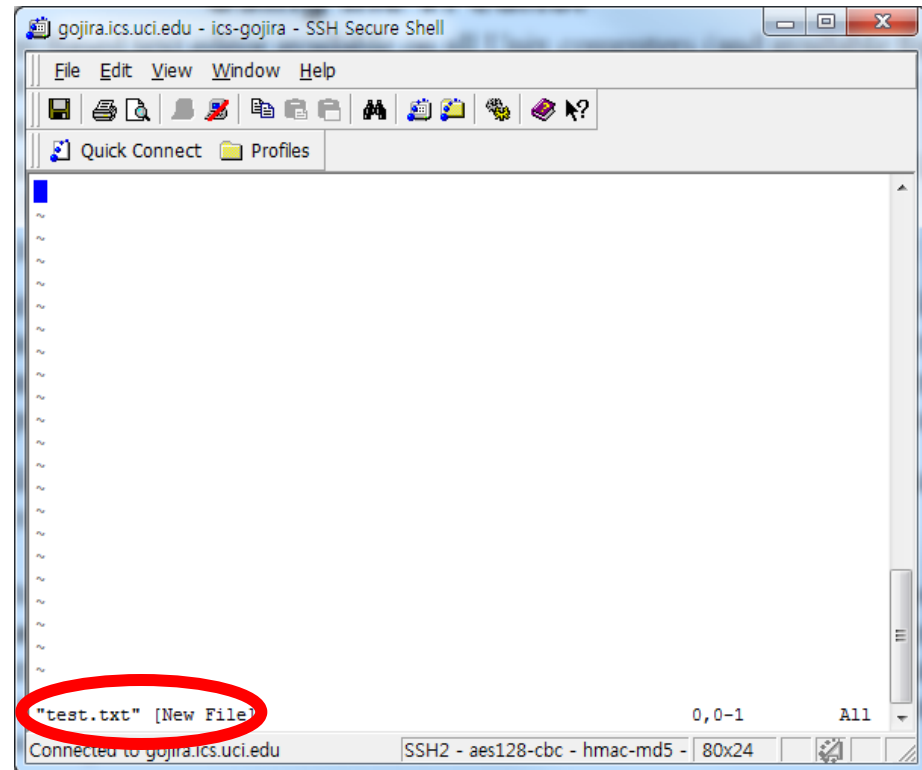
- If there is the file, it is opened based on the permission of the file
- If there is not the file, virtually opened first
 - Later the file can be created when vi save the file

Example

- -R option : for read only



Without filename :
Empty view



Filename :
view with the filename

Other ways to open

- vi + filename
 - Place the cursor on last line of file
- vi +n filename
 - Place the cursor on line "n" of file.
- vi +/pat filename
 - Place cursor on line with first occurrence of "pat"tern

Getting out from VI

- You need to save the file or exit from VI
 - First, switch to command mode
 - Then there are many options (in line mode)
 - **:w filename**
 - To save the edited file to a new file “filename” and quit
 - **:w!**
 - Write the file to disk even if read/only
 - **:q**
 - To quit VI, only if the file is already saved
 - **:wq**
 - To save the edited file and quit
 - **:q!**
 - To quit VI without any saving changes
 - **ZZ**
 - Same command to “:wq”

Recovery of file

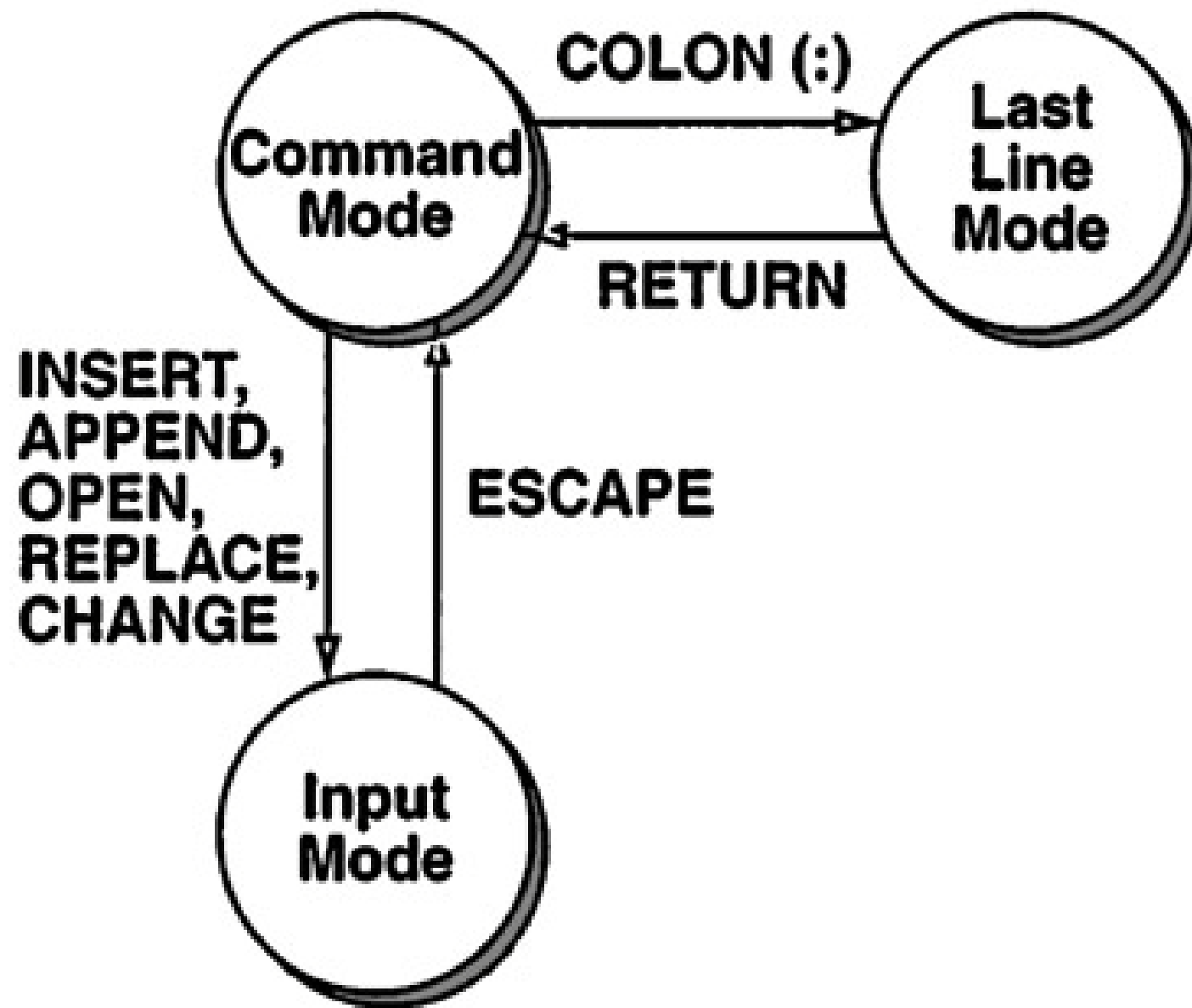
- Sometimes, vi is interrupted and could not save the file properly
- Swap file
 - vi stored the things you changed in a swap file
 - A hidden file starting with “.” in the same directory where the edited file locates
- With “-r” option
 - Case sensitive
 - C.f.) -R option : read only

vi -r → list all the files which can be recovered

vi -r filename → open vi and recover the previous contents

Three modes in VI

- VI has three modes
 - **Insert (editing) mode**
 - Editing texts
 - **Command (or viewing) mode**
 - Performing special functions
 - Initial mode
 - **Line mode**
 - Special mode to execute more complicated functions
- How to switch.
 - To switch to insert mode:
 - Press “i” or “a” or “o”
 - To switch to command mode:
 - Press ESC key
 - To switch to line mode :
 - In command Mode, press “:” or “/” or “?”



Moving around

- There is a blinked cursor
- In command mode, you can move the cursor
 - By using arrow keys
 - Or other keystrokes

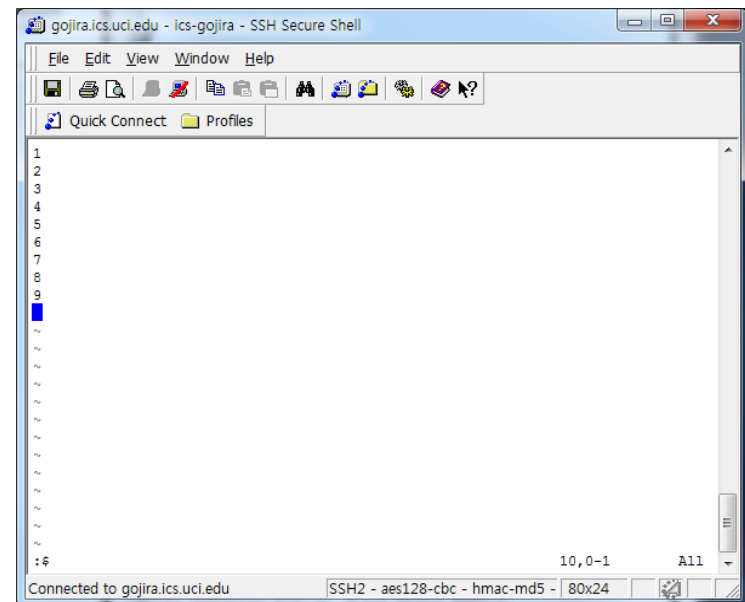
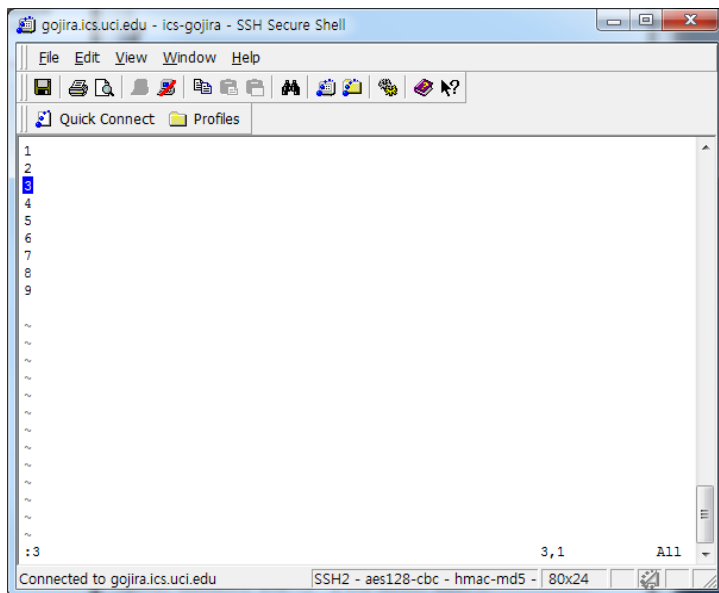
h → left one character
l → right one character
k → up one line
j → down one line
b → back one word
w → forward one word

\$ → to end of the line
^ → to beginning of the line
{ → up one paragraph
} → down one paragraph
^b or ^u → back one page
^f or ^y → forward one page

G → to the last line
1G → to the first line
17G → to line #17

Moving with Line mode

- Move commands
 - `:n` → move to nth line of the file
 - `:$` → move to last line of the file
 - `:$=` → print total number of line of the file
 - `:.=` → print the number of the current line



Inserting Text

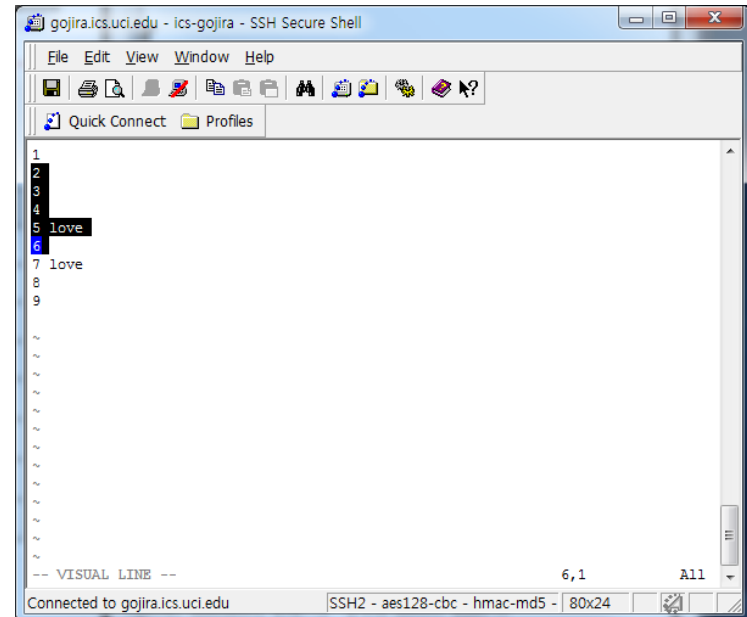
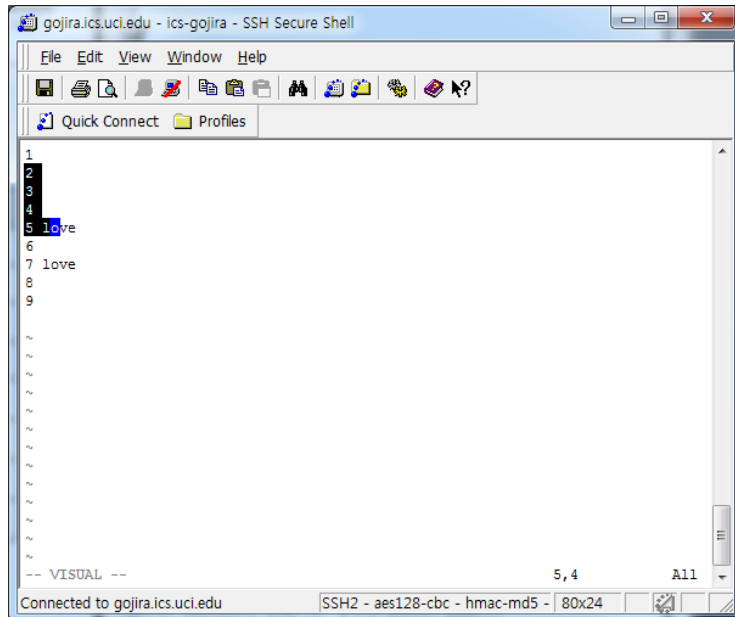
- Options of switching insert mode
 - i → just before the current cursor position
 - a → just after the current cursor position
 - o → into a new line below current cursor
 - I → at the beginning of the current line
 - A → at the end of the current line
 - O → into a new line above current cursor

Cutting, Copying, Pasting

- In command mode
 - x → delete(cut) character
 - 24x → delete(cut) 24 characters
 - dd → delete(cut) current line
 - 4dd → delete(cut) four line
 - D → delete(cut) to the end of line from the cursor
 - dw → delete(cut) to (remainder of) the current word
 - yy → copy (without cutting) current line
 - 5yy → copy (without cutting) 5 line from the current line
 - p → paste after current cursor position/line
 - P → paste before current cursor position/line

Visually Copying

- v → select text for copying (unit of character)
- V → select text for copying (unit of line)



After pressing v or V, selecting text by moving cursor with h,j,k and l.
Pressing y to finish selecting and copying the selected text

Replacing Text

- This combines two steps : deleting then inserting text
 - r → replace 1 character (under the cursor) with a given character
 - 8r → replace each of 8 character with a given command
 - R → overwrite, replace text with typed input, ended with ESC key
 - C → replace from cursor to end of line, with typed input (ended with ESC key)
 - S → replace entire line with typed input (ended with ESC key)
 - 4S → replace 4 lines with typed input (ended with ESC key)
 - cw → replace (remainder of) word with typed input (ended with ESC key)

Replacing text in line mode

- `:[begin,end]s/pattern1/pattern2/flag`
 - Change pattern1 to pattern2
 - `[begin,end]`
 - Define the range of lines for applying the changes
 - “%” represent the entire file
 - “\$” represent the last line
 - “.” represent the first line
 - Without `[begin,end]`, the command is applied in the current line
 - Pattern1, Pattern2 → follows regular expression
 - Flag
 - `g` → every cases
 - `c` → interactive

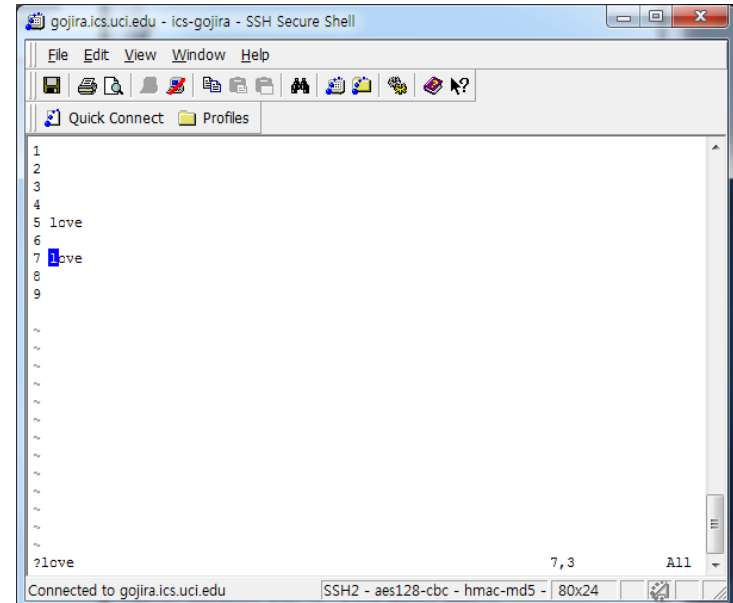
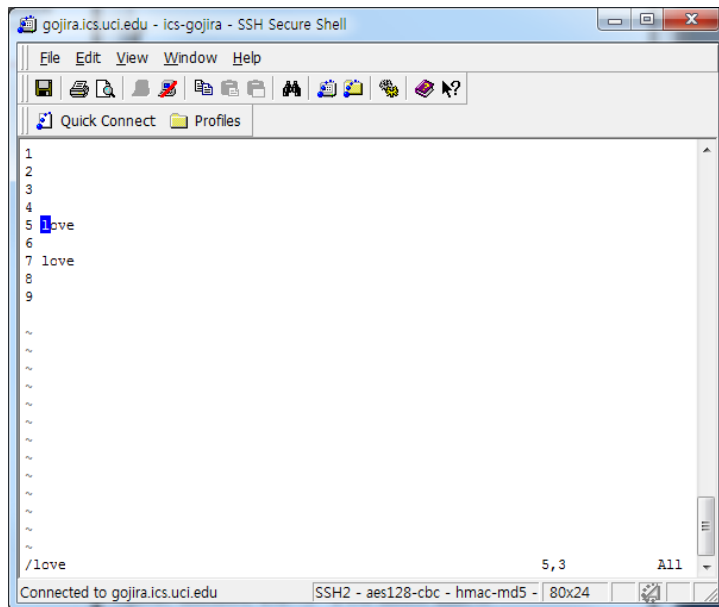
```
:1,10s/peterpan/hook/g  
:%s/love/hate/g
```

Undo and redo and .

- u → undo last changes in the last insert mode
- ^r → redo last changes which were undone.
- . → repeated the last command
 - Every command is applicable

Searching for Text in Line mode

- Use “/” or “?” character
 - /love → jump forward to the next occurrence of the string “love”
 - ?love → jump backward to the previous occurrence of the string “love”
 - n → repeat the last search given by “/” or “?”



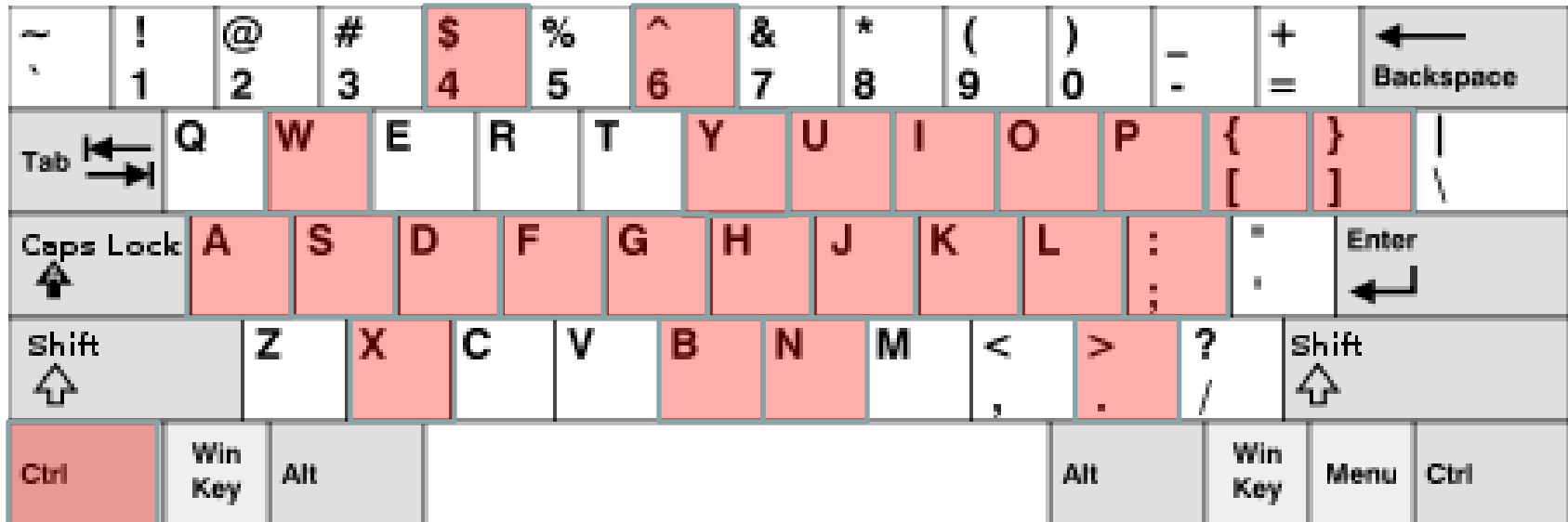
More option for saving in line mode

- **:begin,end w filename**
 - saving the text from begin line to end line into the given filename
- **:1,. w filename**
 - saving the text from 1st line to the current line into the given filename
- **:1,. w >> filename**
 - appending the text from 1st line to the current line into the given filename
- **:3,\$ w >> filename**
 - saving the text from 3rd line to the last line into the given filename

Other linemode commands

- `:r filename`
 - Read a given filename into the current position of cursor
- `:e filename`
 - Finish editing the current file (should be saved) and open a given file
- `:set number / :set nonumber`
 - Show line number / hide line number (default)
- `:! command`
 - Execute shell command
 - E.g.) `:! ls`
- `:r! command`
 - Execute shell command and put the result to the file

Why is vi addictive...



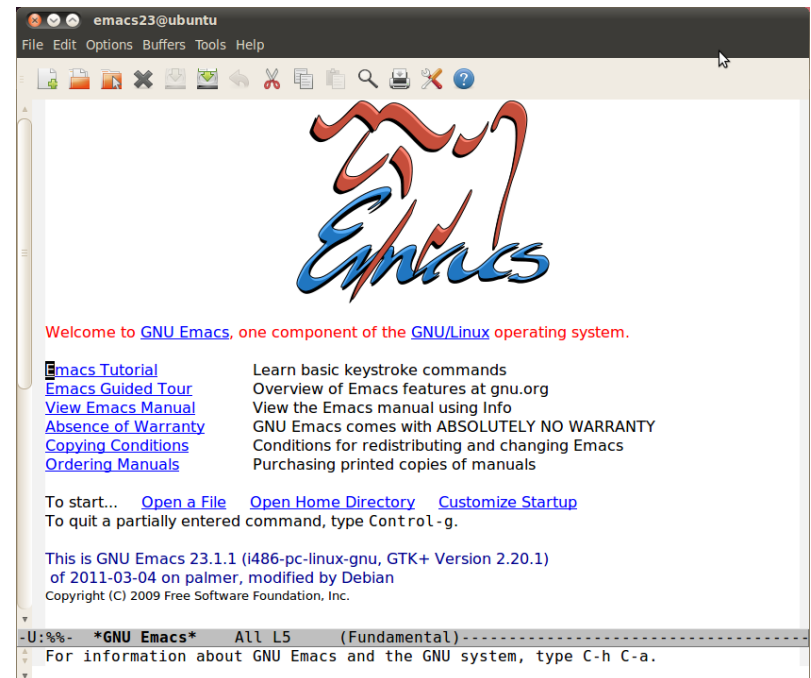
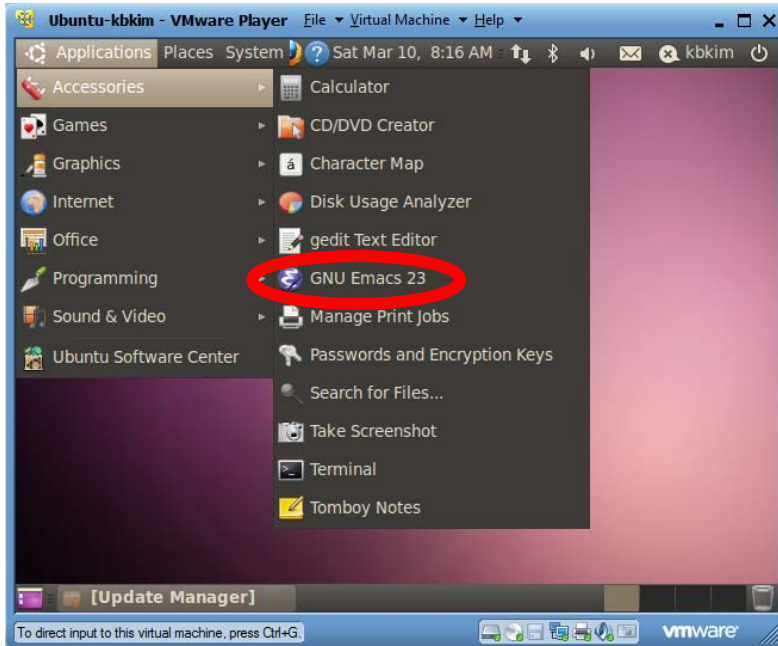
During editing or even commanding, the movement of your hand will be limited.

Worst case



Emacs

- A class of text editor, usually characterized by their extensibility.
- Over 2000 built-in command
- Users can combine these commands into macros to automate work



Emacs as a platform

- Emacs is a platform rather than a simple text editor
- You can plot a graph, use calendar or playing game in Emacs
- Lots of utilities can be programmed in Emacs
 - Emacs Lisp programming language
 - Customizability and extensibility

