*Lu Sun, and many more.*

# *A Notebook on Linux Operating System*

*To all family members, friends and communities members who have been dedicating to the presentation of this notebook, and to all students, researchers and faculty members who might find this notebook helpful.*

# Contents

**Foreword** **vii**

**Preface** **ix**

**List of Figures** **xi**

**List of Tables** **xiii**

## I General Introduction 1

**1 General Introduction to Linux** **3**
- 1.1 General Introduction to Operating System . . . . . . . . . . 3
- 1.2 A Brief History of Linux . . . . . . . . . . . . . . . . . . . . . 3
- 1.3 Key Features of Linux . . . . . . . . . . . . . . . . . . . . . . 3
  - 1.3.1 Filesystem Hierarchy . . . . . . . . . . . . . . . . . . . 3
  - 1.3.2 Access Control Lists . . . . . . . . . . . . . . . . . . . 4

**2 Linux Installation** **5**
- 2.1 Different Flavours of Linux . . . . . . . . . . . . . . . . . . . 5
- 2.2 Linux Installation . . . . . . . . . . . . . . . . . . . . . . . . 5
- 2.3 Linux Basic Configuration . . . . . . . . . . . . . . . . . . . 5
- 2.4 Useful APPs and Tools . . . . . . . . . . . . . . . . . . . . . 5

## II Linux Basics 7

**3 Shells** **9**
- 3.1 Brief Introduction to Linux Shells . . . . . . . . . . . . . . . 9
- 3.2 Basic Grammar . . . . . . . . . . . . . . . . . . . . . . . . . 9
- 3.3 Useful Commands . . . . . . . . . . . . . . . . . . . . . . . . 9

**4 Text File Editing** **11**
- 4.1 Text File Editing Environment in Linux . . . . . . . . . . . . 11
- 4.2 Vim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
  - 4.2.1 General Introduction to Vim . . . . . . . . . . . . . . 12
  - 4.2.2 Configure Customizable User Profile . . . . . . . . . . 13
  - 4.2.3 Commonly Used Operations in Normal Mode . . . . . 15

**5 Files Management** **17**
5.1 Filesystem Hierarchy Standard . . . . . . . . . . . . . . . . . . 17
5.2 File Management . . . . . . . . . . . . . . . . . . . . . . . . . 17

**6 Software Management** **19**
6.1 Linux Kernel Management . . . . . . . . . . . . . . . . . . . . 19
6.2 General Introduction to Linux Package Management Tools . 19
6.3 Installation of Software . . . . . . . . . . . . . . . . . . . . . . 19
6.4 Software Upgrade . . . . . . . . . . . . . . . . . . . . . . . . . 20
6.5 Uninstallation of Software . . . . . . . . . . . . . . . . . . . . 20
6.6 Software Management Using Git . . . . . . . . . . . . . . . . . 20

**7 Process Management** **21**
7.1 General Introduction to Process . . . . . . . . . . . . . . . . . 21
7.2 Running Process Management on Linux . . . . . . . . . . . . 21

**III Linux Advanced** **23**

**IV Linux Server Management** **25**

**V Linux Security** **27**

**VI Linux on Cloud** **29**

**Bibliography** **31**

# *Foreword*

If a piece of software or an e-book can be made completely open source, why not a notebook?

This brings me back to the summer of year 2009, when I just started my third year as a high school student in Harbin No. 3 High School. In around August and September of every year, that is, when the results of Gaokao (National College Entrance Examination of P. R. China, annually held in July) are released, you would find people selling notebooks photocopies claimed to be collected from the top scorers of the exam. Much as I was interested in what these notebooks look like, I myself was not expecting to actually learn anything from them, mainly for the following three reasons.

First of all, some (in fact many) of these notebooks were more difficult to understand than the textbooks. I guess we cannot blame the top scorers for being too smart and make things sometimes extremely brief, or otherwise overwhelmingly complicated.

Secondly, why would I want to adapt to notebooks of others when I have my own? And by the way, I was positive that mine would be as good as theirs, given that I had been putting the same time (three years of high school, only for 6 modules!) and effort learning the courses and preparing the notebooks.

And lastly, as a student in Harbin No. 3 High School, I knew that the top scorers of the coming year would probably be a schoolmate next door, perhaps even a good friend of mine. Why would I want to pay a great amount of penny to a complete stranger in a photocopy shop for his or her notebook, rather than ask from him or her directly?

However, things have changed later on after entering a university as an undergraduate student. I think the main cause of the change is that, since in the university there are so many modules and materials to learn, students are often distracted from digging into one book or module very deeply. (For those who still can concentrate, you have my highest respect.) The situation becomes even worse as I become a Ph.D. student, this time due to that I have to concentrate on one subject entirely, and can hardly split much time on other irrelevant but still important and interesting contents.

This motivates me to start reading and taking notebooks for selected books and articles such as journal papers and magazines. I have a bunch of notebooks with me, most of them are physical. My very first notebook is on *Numerical Analysis*, an entrance level module for engineering background students. Till today I have on my hand dozens of notebooks. One day it suddenly came to me: why not digitalize them, and make them accessible online and open source, and let everyone read and edit it?

—

As majority of open source software, this notebook (and it applies to the other notebooks in this series) does not come with any "warranty" of any kind, meaning that there is no guarantee for the statement and knowledge in this notebook to be exactly correct as it is not peer reviewed. **Do NOT cite this notebook in your academic research paper or book!** Of course, if you find anything here useful with your research, please trace back to the origin of the citation, and read it yourself, and on top of that determine whether or not to use it in your research.

This notebook is suitable as:

- a quick reference guide;
- a brief introduction to the subject;
- a "cheat sheet" for students to prepare for the exam (Don't bring it to the exam unless it is allowed by your lecture!) or for lectures to prepare the teaching materials.

This notebook is NOT suitable as:

- a direct research reference;
- a replacement to the textbook;

because as explained the notebook is NOT peer reviewed and it is meant to be simple and easy to read. It is not necessary brief, but all the tedious explanation and derivation, if any, shall be "fold into appendix" and a reader can easily skip those things without any interruption to the reading.

—

Although this notebook is open source, the reference materials of this notebook, including many textbooks, journal papers, conference proceedings, etc., may not be open source. Very likely many of these reference materials are licensed or copyrighted. Please legitimately access these materials and properly use them if necessary.

# *Preface*

Some references of this notebook are the Linux Bible (10th edition) that I borrowed from National Library Singapore, and also many Bilibili and YouTube videos, which I will cite as I go through the notebook.

# *List of Figures*

4.1 Mode switching between normal mode and insert mode, and basic functions associated with the modes. . . . . . . . . . . . . 13
4.2 A flowchart for simple creating, editing and saving of a text file using *Vim*. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

# *List of Tables*

4.1 Commonly used modes in *Vim*. . . . . . . . . . . . . . . . . . . 13

## Part I

# General Introduction

# 1

# *General Introduction to Linux*

**CONTENTS**

1.1 General Introduction to Operating System ..................... 3
1.2 A Brief History of Linux ..................... 3
1.3 Key Features of Linux ..................... 3
1.3.1 Filesystem Hierarchy ..................... 3
1.3.2 Access Control Lists ..................... 3

“nobreak

## 1.1 General Introduction to Operating System

“nobreak

## 1.2 A Brief History of Linux

“nobreak

## 1.3 Key Features of Linux

“nobreak

### 1.3.1 Filesystem Hierarchy

“nobreak

### 1.3.2 Access Control Lists

# 2

# *Linux Installation*

**CONTENTS**

2.1 Different Flavours of Linux ..................................... 5
2.2 Linux Installation ............................................... 5
2.3 Linux Basic Configuration ...................................... 5
2.4 Useful APPs and Tools ........................................ 5

“nobreak

## 2.1 Different Flavours of Linux

“nobreak

## 2.2 Linux Installation

“nobreak

## 2.3 Linux Basic Configuration

“nobreak

## 2.4 Useful APPs and Tools

# Part II

# Linux Basics

# 3

# *Shells*

**CONTENTS**

3.1 Brief Introduction to Linux Shells .............................. 9
3.2 Basic Grammar .............................................. 9
3.3 Useful Commands ............................................ 9

“nobreak

## 3.1 Brief Introduction to Linux Shells

“nobreak

## 3.2 Basic Grammar

“nobreak

## 3.3 Useful Commands

# 4

# *Text File Editing*

## CONTENTS

4.1 Text File Editing Environment in Linux ........................ 11
4.2 Vim ........................................................ 11
  4.2.1 General Introduction to Vim ............................ 12
  4.2.2 Configure Customizable User Profile .................... 13
  4.2.3 Commonly Used Operations in Normal Mode .......... 15

“nobreak

## 4.1 Text File Editing Environment in Linux

“nobreak

## 4.2 Vim

*Vim* is a free and open-source software initially developed by Bram Moolennar, and has become the default text editor of many Unix/Linux based operating systems.

Some people claim *Vim* to be the most powerful text file editor as well as integrated development environment for programming on a Linux machine (and potentially on all computers and servers). The main reasons are as follows.

- *Vim* is usually built-in to Linux during the operating system installation, making it the most available and cost-effective text editor.
- *Vim* can work on machines where graphical desktop is not supported.
- *Vim* is light in size and is suitable to run even on an embedded system.

- *Vim* operations are done mostly via mode switch and shortcut keys, so that **the brain does not need to halt and wait for the hand to grab and move the mouse** which slows down the text editing and interrupts the logic flow.
- *Vim* is highly flexible and can be customized according to the user's habit (for example, through `~/.vim/vimrc`), and it allows the users to define shortcut keys.
- *Vim* can automate repetitive operations, such as by using macros.
- *Vim* can be integrated with third-party tools for useful functions such as browsing project folders.

The above reasons have their point, and it is true *Vim* can be come very powerful and convenient for the user if he is very familiar with it and is very used to it. On the other hand, however, *Vim* is not as intuitive as other text editors such as *gedit* and *notepad++*, and may require a learning curve for beginners.

In this section, *Vim* is introduced as the text editor that will be used for viewing and editing text files, either being configuration files or programming codes.

### 4.2.1 General Introduction to Vim

Different from other text editors, *Vim* defines different "modes" during the operation, each mode has some unique features. For example, in the *insert* mode, *Vim* takes in the keyboard inputs and put them into the text file. In this concept, many other editors can be taken as a slim version of *Vim* where there is only one mode, the *insert* mode.

In the case of *Vim*, however, there are other equally useful modes that eventually make it unique and powerful. For example, in the *normal* mode (this is the default mode when opening *Vim*), *Vim* uses useful and customizable shortcut keys to quickly navigate the document and perform operations such as cut, copy, paste, replace, search, and macro functions. In the *virtual* mode, *Vim* allows the user to select partial of the document for further editing. In the *cmdline* mode, *Vim* takes order from command lines and interact with Linux to perform tasks such as save, quit or even navigating folders.

The following Table 4.1 summarizes the commonly used modes in Vim.

As a start, the following basic commands can be used to quickly create, edit and save a text file using vim. In home directory, start a shell and key in

```
$ vim testvim
```

to create a file named "testvim" and open the file using *Vim*. Notice that in some Linux versions, *vi* might be aliased to *vim* by default.

In the opened file, use `Esc` and `i/a` to switch between normal mode and

**TABLE 4.1**
Commonly used modes in *Vim*.

| Mode | Description |
|---|---|
| Normal | Default mode. It is used to navigate the cursor in the text, search and replace text pieces, and run basic text operations such as undo, redo, cut (delete), copy and paste. |
| Insert | It is used to insert keyboard inputs into the text, just like commonly used text editors today. |
| Visual | It is similar to normal mode but areas of text can be highlighted. Normal mode commands can be used on the highlighted text. |
| Cmdline | It takes in a single line command input and perform actions accordingly, such as save and quit. |

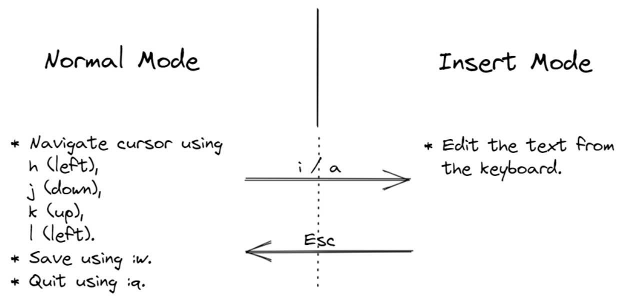

**FIGURE 4.1**
Mode switching between normal mode and insert mode, and basic functions associated with the modes.

insert mode. In the normal mode, use `h`, `j`, `k`, `l` to navigate the position of the cursor. Finally, in the normal mode, use `:w` to save the file, and `:q` to quit *vim*, or use `:wq` to save and quit *Vim*.

The above basic commands and their relationships are summarized in Fig. 4.1. A flowchart to create/open, edit, save, and quit a text file using the aforementioned commands are given in Fig. 4.2.

### 4.2.2 Configure Customizable User Profile

With the basic operations introduced in Section 4.2.1, we are able to create and edit a text file as we want to, just like using any other text editor. Though at this point the advantages of using *Vim* over other text editors are not obvious yet, the *Vim* editor is finally useful now.

Before introducing more advanced features of *Vim* for more convenient user experience, we can now customize user profile to suit our individual habit.

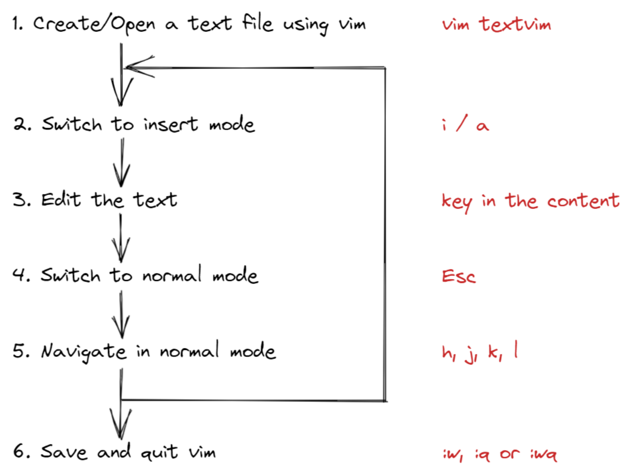

**FIGURE 4.2**
A flowchart for simple creating, editing and saving of a text file using *Vim*.

Notice that the customization is completely optional and personal. This section only introduces the ideas and basic methods of such customization, such as re-mapping keys and create user-defined shortcuts. Everything introduced here are merely examples and it is completely up to the user how to design and implement his own profile.

In Linux, navigate to home directory. Create the following path and file `~/.vim/vimrc` or `~/.vimrc`. Open the *vimrc* file as a blank file using *Vim*. The individual user profile can be customized here.

### Mapping of Keys

It is desirable to re-map some keys to speed up editing. For example, people may want to map `jj` to `Esc` in insert mode for more convenient mode switching to normal mode (consequent "jj" is rarely used in English). Other people may feel like mapping `j`, `k`, `i` to `h`, `j`, `k` respectively in normal and visual modes, making the navigation more intuitive. In that case, a different key needs to be mapped for `i` since it is an important key for switching to insert mode.

It is possible re-map certain key (or keys combination) in selected modes. The following configuration in *vimrc* file re-maps the aforementioned keys.

```
inoremap jj <Esc>
noremap j h
noremap J H
noremap k j
noremap K J
noremap i k
noremap I K
noremap h i
noremap H I
```

where `inoremap` is used to map keys (combinations) in insert mode, and `noremap` in normal and visual modes.

The upper case letter `S` and lower case letter —s— in control mode are originally used to delete and substitute texts. They may be not so important in practice as there functions are overlapped by another shortcut key `c`, which is powerful in replacing characters and is more frequently used. We can re-map `S` for saving the text, and disable `s` to prevent mis-touching. Similarly, upper case letter `Q` is mapped to quit *Vim*.

```
noremap s <nop>
map S :w<CR>
map Q :q<CR>
```

where `<nop>` stands for "no operation" and `CR` stands for the "enter" key on the keyboard. The keyword `map` differs from `noremap` in the sense that `map` is for recursive mapping.

#### Syntax Highlight, Color Scheme and Others

By default *Vim* displays white color contents on black background. Use the following command in *vimrc* to enable syntax highlighting or change color scheme. Use `:colorscheme` in normal mode in *Vim* to check for available color schemes.

```
syntax on
colorscheme default
```

### 4.2.3 Commonly Used Operations in Normal Mode

# 5

# *Files Management*

**CONTENTS**

5.1 Filesystem Hierarchy Standard ........ 17
5.2 File Management ........ 17

“nobreak

## 5.1 Filesystem Hierarchy Standard

“nobreak

## 5.2 File Management

# 6

# *Software Management*

**CONTENTS**

6.1 Linux Kernel Management .................................... 19
6.2 General Introduction to Linux Package Management Tools ..... 19
6.3 Installation of Software ........................................ 19
6.4 Software Upgrade .............................................. 19
6.5 Uninstallation of Software ...................................... 20
6.6 Software Management Using Git ............................... 20

“nobreak

## 6.1 Linux Kernel Management

“nobreak

## 6.2 General Introduction to Linux Package Management Tools

“nobreak

## 6.3 Installation of Software

“nobreak

## 6.4 Software Upgrade

“nobreak

## 6.5 Uninstallation of Software

“nobreak

## 6.6 Software Management Using Git

# 7

# *Process Management*

**CONTENTS**

7.1 General Introduction to Process ................................ 21
7.2 Running Process Management on Linux ........................ 21

“nobreak

## 7.1 General Introduction to Process

“nobreak

## 7.2 Running Process Management on Linux

# Part III

# Linux Advanced

# Part IV

# Linux Server Management

# Part V

# Linux Security

# Part VI

# Linux on Cloud

# *Bibliography*