Preliminary

SCE 212

Spring 2019

0 들어가기 전에…

이 문서에는 SCE 212 에서 사용될 tool 들의 가이드라인 및 기초적인 리눅스 내용을 담고 있습니다.

0.1 Linux

Linux 는 간단히 설명하자면 Windows / OS X 와 같은 운영체제 중 하나라고 할 수 있습니다.. (사실 Linux 라는 명칭은 kernel 만을 의미합니다. UX 를 포함한 전 운영체제를 지칭하려면 GNU/Linux 라고 지칭해야합니다.) Linux 는 1991년 Linus Torvalds 에 의해 처음으로 개발되었습니다. 당시의 Minix 라는 교육용 운영체제를 바탕으로 Linus 는 자신이 원하는 성능 향상을 목적으로 새로운 운영체제를 개발하게됩니다. 그 운영체제가 바로 Linux 이고 이 이름은 Linux의의 이름과 Minix 가 결합된 이름입니다. (Linux 의 역사에 대해서 더 궁금하신 분들은 위키피디아를 참조하시기 바랍니다.)

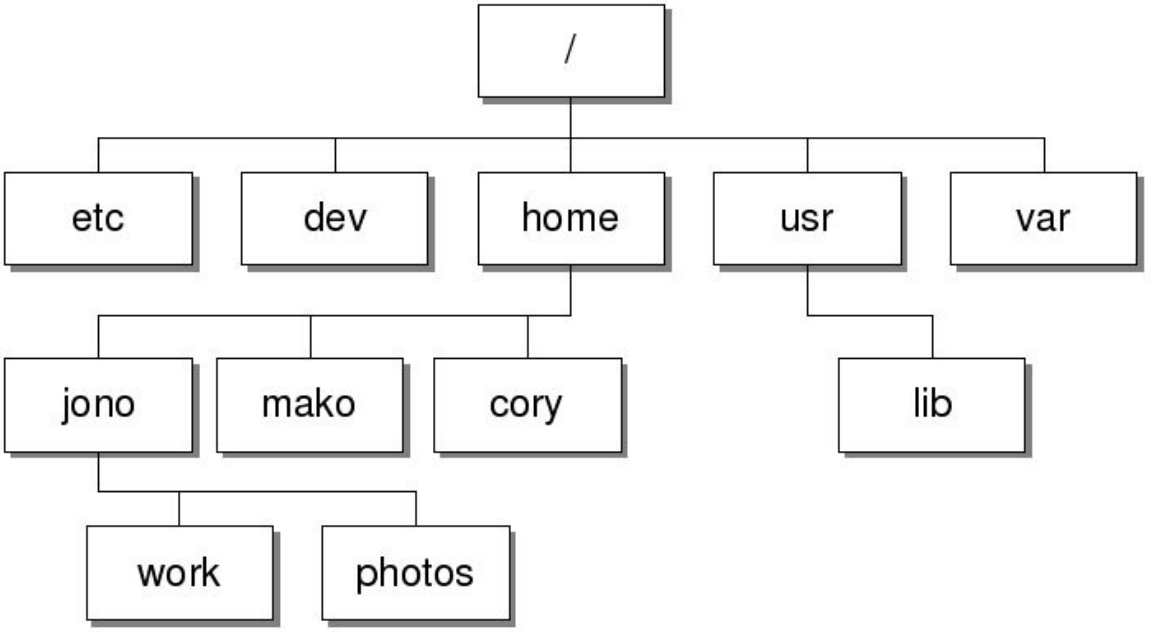
Linux 는 발표된 후 곧바로 오픈 소스로 알려지게 되었습니다. 오픈 소스란 간단히 설명하자면 누구나 읽을 수 있고 수정할 수 있는 소스입니다. (자세한 License 설명은 링크를 참조하시기 바랍니다.) Linux 가 오픈 소스로 정립되다보니 많은 사람들이 이 프로젝트에 참여하게 되었고 그 결과 현대에서 가장 획기적인 프로젝트 혹은 운영체제로 자리잡게 되었습니다.

90년대의 MS 의 Windows 출시 당시, 혹은 OS X의 근래의 눈부신 발전을 바탕으로 여전히 두 운영체제가 전 세계에서 많은 사용자 수를 가지고 있다는 것은 사실입니다. 하지만 Linux 는 현대 시대에 다른 운영체제들에게 절대로 뒤지지 않을 만큼 편리하고 효율적인 운영체제로 이미 자리를 잡고 있습니다. 많은 시장에선 서버 및 임베디드 시스템의 운영체제로 LInux를 선택하고 있고 심지어 개인 사용자들도 다른 운영체제 대신 조금 더 자유롭고 효율적인 Linux 를 많이 사용하고 있습니다. 앞으로도 Linux는 더욱 강력해질 것이고, 그 중요성 역시 나날이 커질 것입니다.

이렇듯 현대에서 Linux 는 논란의 여지 없이 가장 중요한 운영체제로서 선두에 위치하고 있습니다. 이 때문에 대학 교과과정에서 Linux 를 배워야하는 것은 정말 당연한 일입니다. 때문에 이 문서에선 학생들에게 Linux 를 간단하게 소개시켜드리고자, 깊이있는 정보까지는 아니지만 개략적인 Linux 의 정보들을 담고 있습니다.

0.1.1 Structure

Linux 는 기본적으로 user 가 볼 수 있는 모든 것부터 보이지 않는 운영체제 및 하드웨어가 구성하고 있는 모든 것까지 모두 file 의 형태로 이루어져있습니다. (Linux 자체가 file 의 총 집합체라고 말해도 과언이 아닙니다.) 아래는 Linux kernel 이 구성하고 있는 file structure 입니다.



몇 가지 생략된 directory 가 있지만 일반적으로 그림과 같은 구조를 이루고 있습니다.

- "/": root directory 입니다. Linux file structure 에서 가장 상위에 위치한 directory 입니다. 모든 시스템의 file 들은 전부 이 directory 에서 파생됩니다.
- "etc": 시스템의 설정 파일들이 담겨있는 directory 입니다.
- "dev": Hardware image file 들을 담고 있는 directory 입니다. Linux 를 실행하고 있는 혹은 부팅하고 있는 Hardware 의 모든 장치들을 거의 모두 담고 있습니다.
- "home": 사용자(user) 들의 개인 directory 가 저장되고 있는 공간입니다. 예를 들어 그림에 표시된 것처럼 "jono", "mako", "cory" 와 같은 사용자들이 각각 자신의 directory 를 home 아래에 가지고 있습니다.
- "usr": 사용자가 설치한 혹은 일반적인 명령어들을 담고 있는 directory 입니다. 특정 사용자가 아닌 일반적으로 전 사용자에게 적용되는 directory 입니다.
- "var": 시스템이 일시적으로 데이터를 저장하는 공간입니다.

이 외에 Linux distro 및 사용자의 설정에 따라 추가된 directory 가 있을 수 있습니다. 자세한 사항들은 링크를 참조하시기 바랍니다.

0.1.2 Path

Linux 에는 Windows 와 동일하진 않지만 거의 비슷한 개념으로 Path 라는 것이 있습니다. 예를 들어 Windows 에서 "C:₩Program files₩AppData₩" 로 path 를 나타낼 수 있다면 Linux 에선 비슷하게 "/home/user/application" 의 형태로 나타낼 수 있습니다. Path 를 나타내는 방식은 두 가지가 있는데 각각 표기하는 방법에 따라 불려지는 이름이 다릅니다.

Absolute Path: 직역하면 절대 경로라는 뜻으로 Root 부터 선택된 file, directory 까지의 전체 path 가 기재된 path 를 뜻합니다. 예를 들어 앞의 예제와 같은 것이 Absolute Path 입니다.

Relative Path: 직역하면 상대 경로라는 뜻으로 선택된 file 및 directory 의 시점에서 보여지는 path 를 의미합니다. 여기서는 Absolute Path 와 다르게 사용되는 개념들이 몇 가지 있습니다.

- ㅈ"." (점 하나 입니다.): 현재 위치 혹은 현재 working directory 를 뜻합니다. 예를 들어 내가 위치한 지점(working directory)이 "/home/user/app" 이라고 가정합니다. 그렇다면 "." 가 Linux 에서 읽어지는 값은 "/home/user/app" 입니다.
- ".." (점 두 개 입니다.): 현재 위치의 상위에 위치한 곳을 뜻합니다. 앞의 예를 빌어 현재 내가 위치한 곳이 "/home/user/app" 이라면 ".." 이 가리키고 있는 곳은 "/home/user" 입니다.
- "~" (Tilda): 접속한 사용자의 개인 directory path 를 뜻합니다. 접속한 사람의 이름이 "vincent" 이고 이 사람에서 Linux 가 "~" 를 인식하는 값은 "/home/vincent" 입니다.
- 예를 몇 가지 더 들어보겠습니다. 예시로 설정된 Linux 에는 "/home/vincent/foo/" 과 "/home/vincent/bar/" 라는 foo/bar directory 가 두 개 있습니다. 내가 만약 "/home/vincent/foo/" 에 위치하였다면 이 위치에서 ".." 는 "/home/vincent" 를 뜻하고 "../../" 는 "/home/" 을 뜻합니다. 마찬가지로 "../bar" 는 Absolute path 로 표현하면 "/home/vincent/bar" 를 뜻합니다.

0.1.3 User

Linux 에는 크게 두 가지의 user 가 있습니다. 먼저 시스템의 모든 파일을 읽기/수정/삭제할 수 있는 권한을 가진 super user (줄여서 su 혹은 root) 가 있고 앞서 본 것처럼 "home" 하위에 directory 를 가지고 있는 normal user 가 있습니다. 각각의 user 들은 자신의 권한들을 다르게 가지고 있습니다. 만약 normal user 가 시스템에 큰 영향을 끼치는 행위(읽기/쓰기) 들을 한다면 super user 의 권한이 필요합니다. 여기서 말하는 권한은 다음과 같이 세 가지로 나눌 수 있습니다.

- Read: file/directory 를 읽을 수 있는 권한입니다.
- Write: file/directory 를 쓸 수 있는 권한입니다.
- Execution: file/directory 를 실행할 수 있는 권한입니다.

이 권한들은 일반적으로 rwx 로 표현하며 이 값을 보통 binary 로 표현합니다. 예를 들어 내가 읽기 권한만 가지고 있다면 "r--" 로 표현되며 이 경우 "4(100)" 로 나타낼 수 있습니다. 또 다른 예로 나의 권한이 "r-x" 로 표현되어있다면 "5(101)" 로 나타낼 수 있습니다. 권한(Permission) 및 소유(Ownership) 에 관한 자세한 사항은 링크를 참조하시기 바랍니다.

0.1.4 Basic Command

Linux 에는 Windows 의 “cmd” 창과 같은 "Terminal” 이 존재하고 이곳에선 shell 이라는 공간을 활성화하여 command 를 입력할 수 있습니다. 정말 많은 Linux command 들이 있지만 이 문서에서는간단히 file/directory 에 관련된 것만 언급하도록 하겠습니다.

- `ls`: Directory 가 가지고 있는 file list 들을 출력합니다.
- `cd <path>`: 해당 path 로 이동합니다.
- `rm <file>`: 선택된 file 을 삭제합니다.
- `cp <-r> <source> <destination>`: source 있는 file 및 directory 들을 destination 으로 복사합니다.
- `mv <-r> <source> <destination>`: source 에 있는 file 및 directory 들을 destination 으로 이동시킵니다.
- `mkdir <name>`: 기재된 이름으로 directory 를 생성합니다.

위의 command 들이 기본적인 것들이고 이 외에도 linux 에서 자주 사용되는 것들이 있습니다. 각 command 에는 “option” 이 존재하는데 이 option 에 따라 command 가 실행되는 방법이 조금씩 다릅니다. command 의 자세한 사항들을 보려면 `man <command>` 를 입력하시면 됩니다. (Linux command 를 조금 더 배우고 싶다면 강의를 참조하시기 바랍니다.)

0.1.5 Distro

Linux 에는 흔히 알고 있는 Ubuntu 이 외에도 여러가지 배포판들이 있습니다. 크게 Redhat 계열의 Linux 와 Debian 계열의 Linux 로 나뉩니다. Redhat 계열엔 대표적으로 RHEL, Fedora, CentOS 가 있고 Debian 계열은 대표적으로 Ubuntu 와 Linux Mint 가 있습니다. 이 외에도 사용목적에 따라 배포판이 만들어지는 경우도 있습니다. 예를 들어 ArchLinux 의 경우 개발자에게 운영체제 개발의 자유를 보장하기 위해 만들어졌고 Kali linux 의 경우 주로 시스템 보안 이슈 분석 및 침투 모의를 위해서 만들어졌습니다.

0.1.6 Wrap up

위에서 설명드렸던 사항들 이 외에도 Linux 에 알아할 것들이 무궁무진합니다. 이 문서에서 그것들을 모두 다루기에 한계가 있습니다. 그리고 많은 것들을 단기간안에 배우기는 어렵습니다. Linux 의 모든 것들에 익숙해지기 위해선 자주 사용해야하고 혹은 더 나아가 관련된 개발을 하는 수 밖에 없습니다. 하지만 그것들에 시간을 충분히 투자한다면 분명히 가치있는 경험을 얻을 수 있을 것입니다.

- Kernel 에 관심이 있다면 이 링크에 나온 문서들을 읽어보시기 바랍니다.
- Linux GUI / UX 에 관심이 있다면 Gnome / KDE 의 문서들을 읽어보시기 바랍니다. (혹은 X windows system)
- Linux Shell 에 관심이 있다면 bash / zsh 의 차이점을 알아보고 해당 문서를 읽어보시기 바랍니다.
- 개발 automation / packaging 등에 관심이 있다면 GNU Software 의 automake, autoconf 와 같은 tool 들을 확인해보시기 바랍니다.

1 Setup

1.1 Vagrant

Vagrant는 컴퓨터구조 과제를 수행하기 위해 필요한 도구들이 미리 설정되어있는 가상 머신 이미지를 다룰 수 있는 프로그램입니다. Vagrant를 이용하면 Virtual box 내부에서 코드를 작성하지 않고, Windows나 Mac OS에서 코드를 작성하고, 컴파일은 리눅스 환경에서 할 수 있도록 도와줍니다.

Vagrant를 다운로드 받고 이용하는 방법은 다음과 같습니다.

1. Vagrant는 VirtualBox 기반으로 동작하는 프로그램입니다. 그렇기 때문에 Vagrant를 설치하기에 앞서 VirtualBox를 설치해야 합니다. VirtualBox website 에 접속하여 각 OS에 맞는 package 를 다운로드 받고 설치합니다.
2. Vagrant website 에 접속하여 Vagrant를 다운로드 받고 설치합니다. 설치후 컴퓨터를 재시작하는 것을 권장합니다.

1.1.1 Windows (OS X and Linux users can skip this section)

Note: Windows 10 을 사용하시는 분들은 bash on Ubuntu 를 사용하셔도 됩니다.

Windows 에선 Linux 에서 지원하는 Terminal 과 bash 를 일반적으로 지원하지 않기 때문에 Linux 에서 기본적으로 사용하는 명령어들을 사용할 수 없습니다. 이 때문에 Linux 의 Terminal 과 비슷한 application 인 Cygwin 을 설치해야합니다.

a. Set up Cygwin

1. Cygwin 페이지에 접속하여 setup-x86_64.exe을 다운로드합니다.(Windows 64-bit 환경)
2. 첫 화면에서 “Next”
3. “Install from Internet” 선택 후 “Next”
4. "Root Install Directory" 설정 및 install for 에서 "All Users" 선택 후 "Next".
5. "Local Package Directory" 설정 후 “Next”
6. "Select Your Internet Connection"에서 "Direct Connection" 선택 후 "Next".
7. "Choose A Download Site"에서 아무 주소나 선택하고 “Next”
8. "Select Packages" 화면이 뜨고. “View”를 눌러 “Category” 대신 “Full”을 선택합니다. 그러면 모든 패키지들이 나오는데, Search창에 Curl(7.59.0-1), bash(4.4.12-3), git(2.17.0-1)을 검색하여 “New”열의 “Skip”을 클릭하여 버전을 설정합니다. 모두 확인하고 “Next”
9. 이전 화면에서 고른 패키지들의 리스트가 있으니 다시 한번 확인해봅니다. 확인 후에 “Next”를 누르면 Install이 시작됩니다.
10. 완료되면 “Finish” 버튼을 누릅니다.
11. 설치가 완료되었다면 cygwin 을 실행하고 아래의 명령어를 입력합니다.

```
$ echo TERM=xterm-256color >> ~/.minttyrc
$ source ~/.minttyrc
```

- 주의점: Cygwin을 실행할 때 관리자 권한으로 실행해야 합니다. 그렇지 않으면 리눅스 이미지를 다운로드 받을때에 에러가 발생합니다.

1.1.2 Vagrant Setup

1. VirtualBox와 Vagrant가 성공적으로 설치가 되었다면 다음의 명령어를 터미널(windows 이용자는 cygwin 터미널)에 입력합니다.

```
$ git clone http://sce212.ajou.ac.kr/2019S-F039-2/project0.git
$ cd project0
$ vagrant up
$ vagrant ssh
```

2. `vagrant up` 명령어는 Linux image 파일을 다운로드하고 Virtual box를 실행하여 리눅스 가상머신 환경을 구축합니다. `vagrant ssh` 명령어는 앞서 만들어진 Linux 가상머신 환경에 접속합니다.
3. 가상머신을 실행하고 접속하기 위한 모든 명령어는 “project0” directory에서 실행이 되어야 합니다. “project0” directory를 지우거나 수정할 경우 vagrant가 가상머신을 설치하고 구축하거나 접속할 때 오류가 발생할 수 있으니 되도록이면 건드리지 않는게 좋습니다.
4. `vagraunt halt` 명령어는 현재 실행중인 가상 머신을 종료시킬수 있습니다. 만약 이 명령어가 작동하지 않을 경우, Linux 가상머신에 ssh로 접속중인 상태에서 명령어를 실행한 것인지 확인하십시오. 만약 그렇다면, ssh 접속을 해제하고 host로 돌아와서 다시 시도해야 합니다.
5. 이후에 다시 vagrant를 이용하여 가상머신을 이용하기 위해서는, `vagrant up` 과 `vagrant ssh` 명령어를 입력하면 Linux 가상머신을 이용할 수 있습니다.(이미지 파일을 다시 다운로드 하진 않습니다.)
6. 요약하자면 `vagrant up` 은 가상머신 구축 및 실행, `vagrant ssh`는 가상머신에 접속, `vagrant halt`는 실행 중인 가상머신을 종료시키는 명령어입니다.
7. 더 자세한 내용이 궁금하다면 Vagrant 공식가이드 문서를 참고하시기 바랍니다.

1.1.3 Git Name and Email

Git commit을 이용하려면 이름과 이메일을 초기에 설정해주어야 합니다. 다음의 명령어를 vagrant에 접속한 상태에서 입력하세요. **실제 영문 이름**과 **실제 학교 이메일**을 입력하셔야 합니다.

```
$ git config --global user.name "your_name"
$ git config --global user.email "your_email@ajou.ac.kr"
```

1.2 Editing code in your VM

제공된 VM 에는 SMB 가 설치되어있는데 이 SMB server 는 vagrant 가 가동할때 호스트 운영체제와 home directory 를 공유하도록 만들어줍니다. 이 SMB server 를 사용하면 호스트 운영체제에서 text editor 를 사용하여 VM 의 directory 에 접속할 수 있습니다. 따라서 과제를 할 때 terminal 에서 작업하는 것이 익숙하지 않은 학생들이라면 반드시 이 방법으로 작업을 하는 것을 권장합니다. (아래의 모든 경우는 `vagrant up`이 되어 있는 상태를 가정합니다.)

1.2.1 Windows

Note: 아래의 설명은 Windows 10 기준에서 설명되었습니다. Windows 7 를 사용하는 학생의 경우 아래의 가이드를 따라도 안 될 경우 조교에게 문의해주시기 바랍니다.

- SMB를 사용하기 앞서 Windows에서 SMB 기능 활성화 여부를 확인합니다.(링크 참조)
- 파일 탐색기를 열어서 Ctrl L 을 클릭하여 주소창에 커서를 위치시킵니다.
- ₩₩192.168.162.111₩vagrant 을 입력합니다. 그리고 아래의 정보들을 입력해서 접속합니다.
- username: **`vagrant`**
- password: **`vagrant`**

1.2.2 OS X

- Finder 를 실행합니다.
- 메뉴 바에서 이동 -> 서버에 연결… 을 클릭합니다. 그리고 아래의 정보들을 입력합니다.
- Server address: **`smb://192.168.162.111/vagrant`**
- Name: **`vagrant`**
- Password: **`vagrant`**

1.2.3 Linux

아래는 Ubuntu 기준(18.04 bionic) 으로 설명되었습니다.

- Files manager 을 실행합니다.
- 왼쪽 사이드바에서 맨 아래에 + Other Locations 을 클릭합니다.
- 하단의 Connect to server 에 아래의 정보를 입력하고 접속합니다.
- Server address: **`smb://192.168.162.111/vagrant`**
- username: **`vagrant`**
- password: **`vagrant`**

1.3 Shared Folders

`/vagrant` directory 는 호스트 운영체제(자신이 `vagrant up` 을 했던 운영체제) 의 `Vagrantfile` 가 위치한 directory 혹은 폴더와 공유됩니다. 따라서 이 안에 프로젝트를 추가하여 코드를 작성해도 무방합니다. 하지만 앞서 설명했던 SMB 를 사용할 것을 추천합니다.

2 Useful Tools

아래의 Tool 들은 과제를 위해 반드시 배워야하는 것들(git, make)과 그렇지 않은 것들을 설명한 내용입니다. 각 tool 들은 자세히 설명되어있지 않기 때문에 학생들은 첨부된 링크를 참조하셔서 자세한 내용을 보충하셔야 합니다.

2.1 Git

Git 은 version control system 으로 개발자가 작성한 코드를 버전별로 관리할 수 있게 도와줍니다. 예를 들어, 내가 만든 프로그램의 "a.css" 소스 코드가 할로윈 디자인을 적용한 테마였는데 고객에 의해 크리스마스 디자인으로 변경한 상황이 있다고 가정합니다. 이러한 상황에서 또 다시 고객의 변심에 의해 할로윈 디자인으로 코드를 변경해야한다면 매우 번거로운 일이 될 것입니다. 만약 이 프로젝트가 Git 의 도움을 받아 해당 코드를 각 버전별로 저장할 수 있다면 아까와 같은 상황을 모면할 수 있을 것입니다.

Git 을 Hosting 해서 사람들에게 서비스로 제공하는 곳이 있는데 그 회사가 바로 GitHub 이고 이러한 서비스를 주로 Enterprise 에게 제공하는 서비스가 GitLab 입니다.

- 만약 git 을 한 번도 사용해본 학생이 없다면 링크에서 차근히 배워보시기 바랍니다.
- git 에 이미 익숙한 학생이라면 이 슬라이드 와 이 링크를 통해 조금 더 배워보시기 바랍니다.
- 간략하게 git 의 명령어만 알고 싶으시면 이 링크를 참고해 주시기 바랍니다.

2.2 make

`make` 는 Makefile 라는 파일에 의해 프로젝트의 소스코드들을 자동적으로 빌드해주는 tool 입니다. 예를 들어, Linux 를 많이 사용해본 학생이라면 매번 `gcc -o foo foo.c` 의 귀찮음을 알 것입니다. 이것을 Makefile 에 `make` 에 정해진 문법에 맞춰서 빌드를 한다면 `make <target>` 과 같은 명령어 한 번으로 프로젝트 내에 관련된 파일들 전부를 빌드할 수 있습니다.

`make` 의 간단한 튜토리얼1/튜토리얼2을 따라해보면 어느 정도 이해할 수 있습니다. 그리고 `make` 는 GNU 에서 만든 tool 이기 때문에 공식문서 를 참조하시면 더 많은 내용을 확인할 수 있습니다.

2.3 gdb

아쉽게도 Linux 는 Visual studio 와 같은 편리한 IDE 가 없습니다. c 프로젝트를 빌드하기위해선 `gcc` 를 사용해야하고 그리고 더욱 안타깝게도 `gcc` 는 warning 및 error 만 내뱉을 뿐 Visual studio 의 꽃인 debugging tool 과 같은 것이 없습니다. 다행히도 이러한 C debugging 을 지원하는 tool 있는데 그것이 `gdb` 입니다. 만약 `gcc` 로 c 를 빌드할 때 `-g` 을 추가한다면 `gcc` 는 컴파일할 때 debug symbol 을 추가하고 이것을 통해 `gdb` 가 debugging 을 할 수 있습니다. `gdb` 를 사용한다면 내가 원하는 위치에 breakpoint 를 찍을 수 있고 오류가 발생한 지점에서 stacktrace 및 변수들의 값을 확인할 수 있습니다.

자세한 `gdb` 의 사용법은 이 문서를 참조하시고 또한 당연히 공식문서를 참조하시면 더욱 많은 것을 배울 수 있습니다.

2.4 vim

vim 은 Linux 에서 많이 사랑받는 터미널내에서 작동되는 text editor 입니다. 터미널과 친해지기위해 vim 을 배우는 것은 매우 가치있는일입니다. 더 자세히 배우길 원한다면 이 링크를 참조하시면 됩니다. (혹은 게임으로도 배울 수 있습니다!) vim 과 대등하게 많이 사용되는 editor 로 emacs 가 있는데 어떤 editor 를 사용하던 상관없지만 한 가지를 깊게 배우시는 것이 좋습니다. 이 외에도 Atom/Visual Code/Sublime text/Textmate/Notepad++ 등이 있습니다.