System Programming & OS 실습

A3. Vi & Vim

최민국, 정지헌, 안석현, 김선재

Dankook University

{mgchoi, wlgjsjames7224, seokhyun, rlatjswo0824}@dankook.ac.kr

Index

❖Vi & Vim

- 사용법
- Plugin
- 실습

Vi editor

❖ Vi 시작

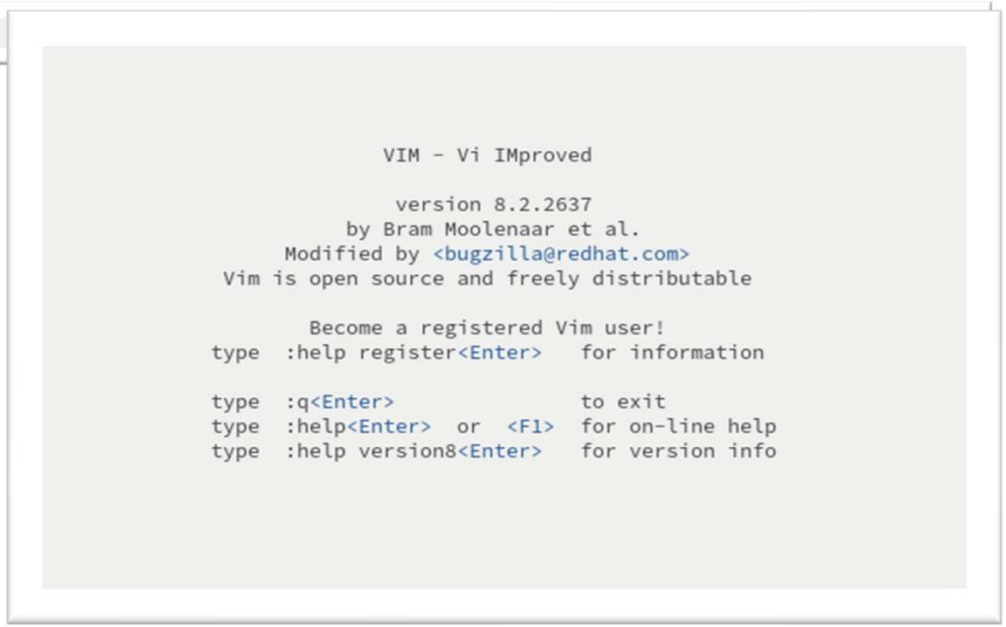
- 파일을 지정할 경우 : 해당 파일이 있으면 파일의 내용이 보이고, 없는 파일이면 빈 파일이 열린다.

```
sunjae@localhost ~]$ vi text.txt
```

- 파일을 지정하지 않을 경우 : 그냥 빈 파일이 열린다(파일명은 저장할 때 지정 가능)

```
sunjae@localhost ~]$ vi
```

❖ 초기 화면



Vi editor

❖ Vi 종료

- 명령모드나 마지막행 모드에서 저장하고 종료 가능

구분	명령키	기능
마지막 행 모드	:q	Vi에서 작업한 것이 없을 때 그냥 종료한다
	q!	작업한 내용을 저장하지 않고 종료한다.
	:w [파일명]	작업한 내용을 저장만 한다, 파일명을 지정하면 새 파일로 저장한다
	:wq, :wq!	작업한 내용을 저장하고 Vi를 종료한다
명령 모드	ZZ(shift + zz)	작업한 내용을 저장하고 vi를 종료한다

Vi editor

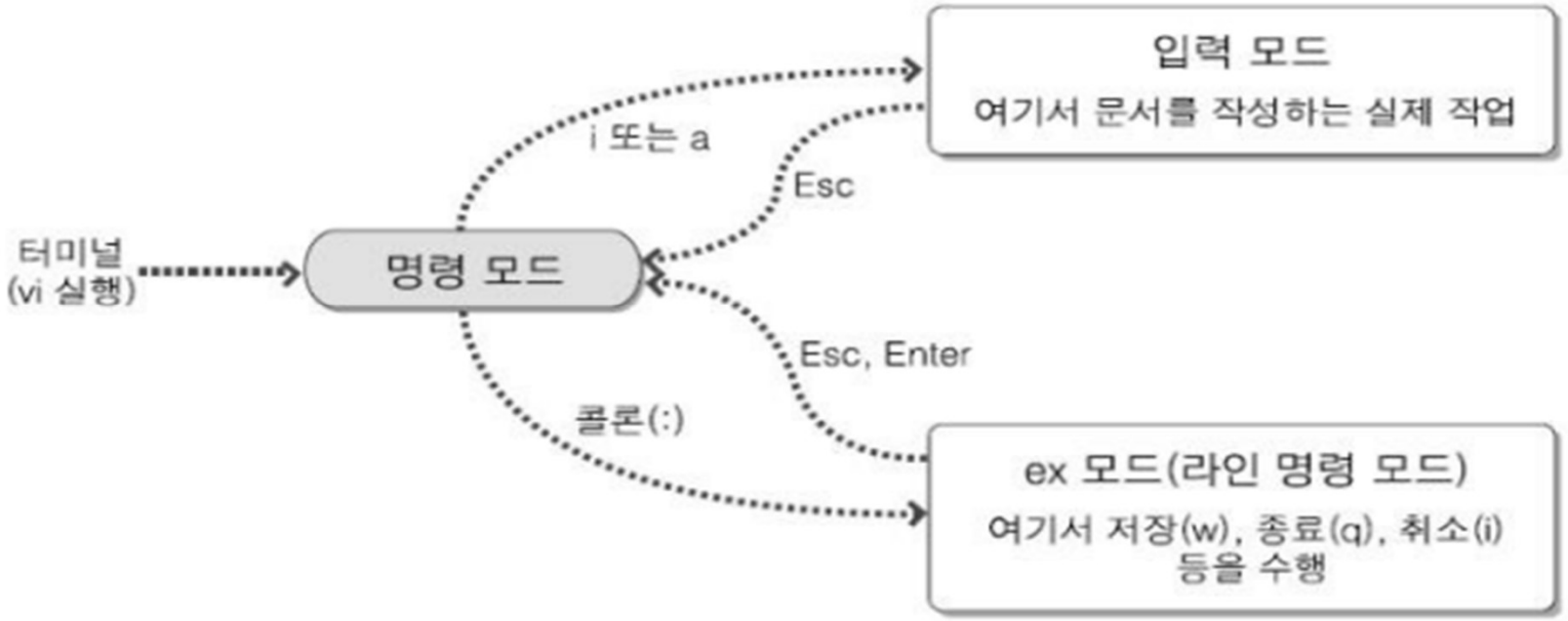
❖ 입력 모드로 전환

명령 키	기능
i	커서 앞에 입력한다(현재 커서 자리에 입력)
a	커서 뒤에 입력한다(현재 커서 다음 자리에 입력)
o	커서가 위치한 행의 다음 행에 입력한다
I	커서가 위치한 행의 첫 칼럼으로 이동하여 입력한다
A	커서가 위치한 행의 마지막 칼럼으로 이동하여 입력한다
O	커서가 위치한 행의 앞 행에 입력한다

Vi editor

❖ 실습

- Vi의 기본적인 사용법을 익혀보자



Vi editor

❖ 실습

- 삽입
- 커서 이동

① 삽입 명령			
a	커서 뒤에 입력	A	라인 끝에 입력
i	커서 앞에 입력	I	라인 시작 부분에 입력
o	커서 있는 라인 밑에 입력	O	커서가 있는 라인 위에 입력

② 커서 이동 명령			
h	왼쪽으로 커서 한칸 이동	H	화면의 처음으로 이동
l	오른쪽으로 한칸 이동	L	화면 끝으로 이동
e	다음 단어의 마지막으로 이동	E	커서를 공백으로 구분된 다음 단어 끝으로 이동
b	한 단어 뒤로 이동	B	커서를 공백으로 구분된 이전 단어로 이동
w	커서를 한 단어 뒤로 이동	W	커서를 공백으로 구분된 다음 단어로 이동
k	커서를 한 라인 위로 이동	j	커서를 한 라인 아래로 이동
0	커서를 라인의 시작으로 이동	$	커서를 라인의 끝으로 이동
Enter	커서를 다음 라인 시작으로 이동	-	커서를 전 라인의 시작으로 이동
Ctrl + F	다음 화면으로 이동	Ctrl + D	화면의 반만 앞으로 이동
Ctrl + B	전 화면으로 이동	Ctrl + U	화면의 반만 뒤로 이동
G	커서를 텍스트의 마지막 라인으로 이동	숫자G	커서를 숫자 라인만큼 이동
M	커서를 화면 중간 라인으로 이동	''	커서를 전 위치로 이동
(	문단의 시작으로 이동	{	문단의 시작 위치로 이동
)	문장 끝으로 이동하여 다음 단어의 시작으로 커서 이동	}	문단 끝으로 이동

Vi editor

❖ 실습

- 삭제
- 바꾸기

③ 삭제 명령			
x	커서가 있는 문자 삭제	X	커서가 있는 문자 앞의 문자 삭제
dw	커서가 있는 단어 삭제	db	커서 앞에 있는 단어 삭제
dW	공백으로 구분된 뒷 단어 삭제	dB	공백으로 구분된 앞 단어 삭제
dd	커서가 있는 라인 삭제	D	커서가 있는 라인의 나머지 삭제
d)	문장의 나머지 삭제	d}	문단의 나머지 삭제
dG	파일의 나머지 삭제	dH	화면의 시작까지 삭제
dL	화면의 나머지 삭제	J	커서와 다음 단어의 공백을 모두 삭제

④ 바꾸기 명령			
r	커서가 있는 문자 대치	R	입력 모드로 한 문자씩 덮어씀
s	커서가 있는 문자 삭제 후 입력 모드로 전환	S	커서가 있는 줄을 삭제한 후 입력 모드로 전환
cb	커서가 있는 앞 문자 삭제 후 입력 모드	cW	공백으로 구분된 뒷 단어를 삭제한 후에 입력 모드
cB	공백으로 구분된 앞 단어 삭제 후 입력 모드	cc	커서가 있는 라인을 삭제하고 입력 모드
C	커서가 있는 라인의 나머지를 삭제하고 입력모드로 전환	c0	커서에서부터 라인의 시작까지 텍스트 바꾸기
c	특정 텍스트 바꾸기	c)	문장의 나머지 바꾸기
c}	문단의 나머지 바꾸기	cG	파일의 나머지 바꾸기
cm	표시까지 모든 것 바꾸기	cL	화면의 나머지 바꾸기
cH	화면의 시작까지 바꾸기		

Vi editor

❖ 실습

- 이동
- 복사

①텍스트 이동			
p	삭제나 복사된 텍스트를 커서가 있는 문자라 라인 뒤에 삽입	P	삭제나 복사된 텍스트를 커서가 있는 문자라 라인 앞에 삽입
dw p	커서가 있는 단어를 삭제한 후 이를 원하는 곳 커서 뒤로 삽입	dw P	커서가 있는 단어를 삭제한 후 이를 변경한 커서가 있는 곳 앞으로 삽입
d p	지정한 다음 텍스트로 삭제한 후 커서가 가리키는 곳으로 이동	d) P	문장의 나머지로 이동
d} p	문단의 나머지로 이동	dG p	파일의 나머지로 이동
dH p	화면 시작 부분으로 이동	dL p	화면의 나머지를 이동

복사			
yw	커서가 있는 단어를 복사	yb	커서가 있는 앞 단어를 복사
yW	공백으로 구분된 뒷 단어 복사	yB	공백으로 구분된 앞 단어를 복사
y	특정한 다음 텍스트 복사	yy	커서가 있는 라인을 복사, 커서가 가리키는 곳으로 라인을 이동
y)	문단의 나머지 복사	y}	문단의 나머지 복사
yG	파일의 나머지 복사	yH	화면 시작까지 복사
yL	화면의 나머지 복사		

Vi editor

❖ 동작모드 예 – i, I, a, A, o, O

a 명령은 커서가 있는
x 문자 뒤에 커서가 입력
모드로 대기

➔ x문자 뒤에서부터 문자 입력

i 명령은 커서가 있는
x 문자에서 커서를 입력
모드로 대기

➔ x문자 앞에서부터 문자 입력

Vi editor

❖ 동작모드 예 – i, I, a, A, o, O

A 명령은 커서가 있는 줄의 마지막 칸에 커서가 입력모드로 대기

➔ 첫 번째 라인 아무 곳에서 A명령 시 x문자 뒤에서부터 입력

I 명령은 커서가 있는 줄의 처음 시작 칸에 커서가 입력 모드로 대기

➔ 첫 번째 라인 아무 곳에서 A명령 시 L문자 앞에서부터 입력

Vi editor

❖ 동작모드 예 – i, I, a, A, o, O

O 명령은 커서가 있는 줄의
첫 칸에 커서가 대기

o 명령은 커서가 있는 줄의
아래 줄 첫 칸에 커서가 위치

Vi editor

❖ Vi editor setting

- 색상 변경, 자동 들여쓰기…

.vimrc 파일

```
syntax enable
syntax on                                   ## 하이라이트
filetype on                                 ## 파일종류 자동인식
set autoindent                              ## 자동 들여쓰기
set background=dark                         ## 배경 컬러
set cindent                                 ## C언어 자동들여쓰기
set history=100                             ## 명령어 기록
set hlsearch                                ## 검색어 강조
set number                                  ## 행 번호 표시
set paste!                                  ## 계단현상 제거 (붙여넣기)
set shiftwidth=4                            ## 들여쓰기 설정
set showmatch                               ## 괄호의 짝을 표시해주는 기능
set statusline=%h%F%m%r%=[%l:%c(%p%%)]          ##상태정보라인 구성
set smartindent                             ## 스마트한 들여쓰기
set tabstop=4                               ## 탭(tab) 간격
set textwidth=80                                ## 자동줄 바꿈 길이
set title                                       ## 현재 수정중인 파일명 표시
set ruler                                       ## 좌표 표시
colo koehler                                ## 색상 테마출처
```

```
[root@ip-172-31-15-105 ec2-user]# yum install vim
```

1. Vim 패키지 설치

```
[root@ip-172-31-15-105 ec2-user]# vi ~/.vimrc
```

2. .vimrc 파일 생성 및 내용 추가

Vi editor

❖ Vi editor setting

- 색상 변경, 자동 들여쓰기…

```
[root@ip-172-31-15-105 ec2-user]# vi ~/.bashrc
```

3. vi ~/.bashrc 파일 수정

```
1  # .bashrc
2
3  # Source global definitions
4  if [ -f /etc/bashrc ]; then
5      . /etc/bashrc
6  fi
7
8  # User specific environment
9  if ! [[ "$PATH" =~ "$HOME/.local/bin:$HOME/bin:" ]]
10 then
11     PATH="$HOME/.local/bin:$HOME/bin:$PATH"
12 fi
13 export PATH
14
15 # Uncomment the following line if you don't like systemctl's auto-paging feature:
16 # export SYSTEMD_PAGER=
17
18 # User specific aliases and functions
19
20 alias rm='rm -i'
21 alias cp='cp -i'
22 alias mv='mv -i'
23 alias vi='vim'
```

4. alias vi='vim' 내용 추가

```
[root@ip-172-31-15-105 ec2-user]# source ~/.bashrc
```

5. 저장 후 설정한 부분 적용

Vi editor

❖ 파일 생성

- 김소월 '가늘 길' 작성

```
[ec2-user@ip-172-31-15-105 taba1]$ vi poem
```

```
[ec2-user@ip-172-31-15-105 taba1]$ cat poem
```

```
 1 가 늘   길
 2                              김 소 월
 3 그 립 다
 4 말 을   할 까
 5 하 니   그 리 워 .
 6
 7 그 냥   갈 까
 8 그 래 도
 9 다 시   더   한   번 …….
10
11 저   산 에 도   까 마 귀 , 들 에   까 마 귀 ,
12 서 산 에 는   해   진 다 고
13 지 저 귑 니 다 .
14
15 앞 강 물 , 뒷 강 물 ,
16 흐 르 는   물 은
17 어 서   따 라 오 라 고   따 라 가 자 고
18 흘 러 도   연 달 아   흐 릅 디 다 려 .
```

Vim Plugin

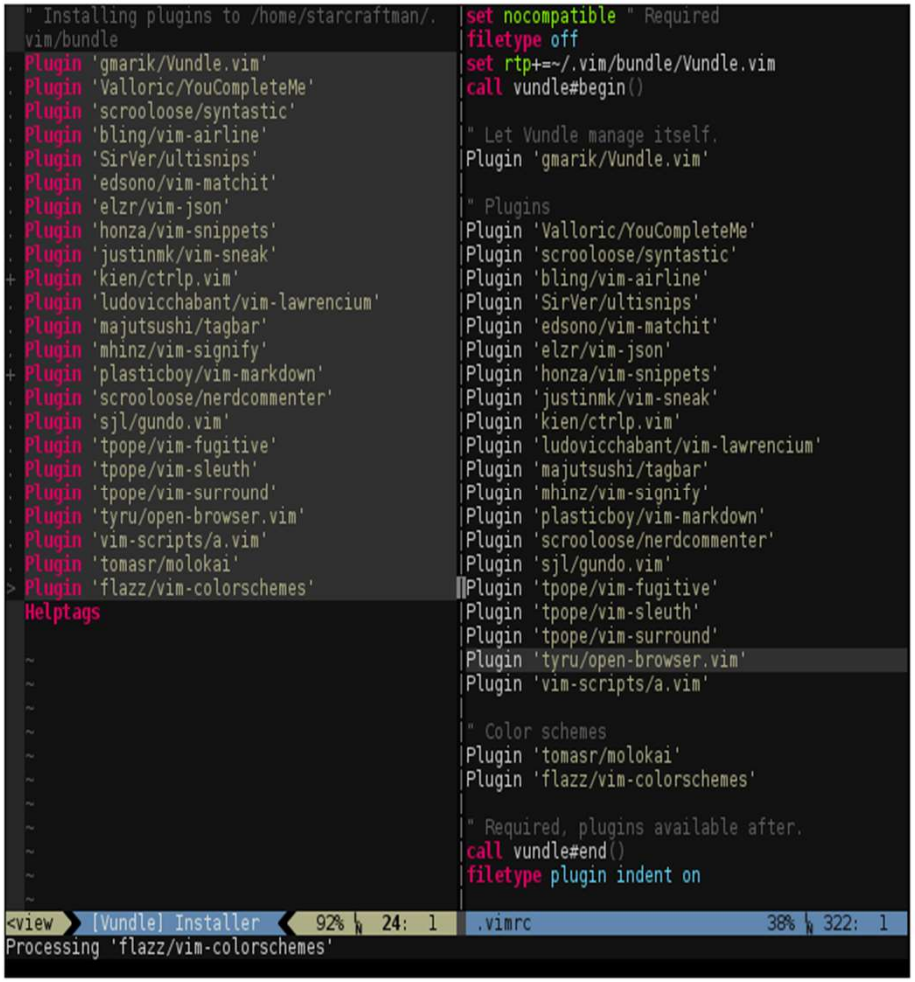
❖ Vim Vundle

- Vim plugin 관리
- 코드 자동 완성, 함수 호출 등 다양한 기능 사용이 가능

-> 일반적으로 IDE에서 사용하는 기능들 사용 가능

- 온라인으로 목록을 불러와서 설치 및 설정이 가능

링크 : https://github.com/VundleVim/Vundle.vim



Vim Plugin

❖ Vim Vundle

- Vundle git 페이지에서 명령어를 수행

-> git clone https://github.com/VundleVim/Vundle.vim.git ~/.vim/bundle/Vundle.vim

```
[ec2-user@ip-172-31-5-168 ~]$ sudo yum install git
Last metadata expiration check: 0:08:19 ago on Wed 14 Aug 2024 03:42:48 PM UTC.
Dependencies resolved.
```

- git 명령어 다운

-> sudo yum install git

Vim Plugin

❖ Vim Vundle

- .vimrc 생성
- Vim 설치 위치와 동일한 위치에 생성

-> vim .vimrc

- 옆의 내용 복사 붙여넣기

-> git 에서 해당 내용 복사

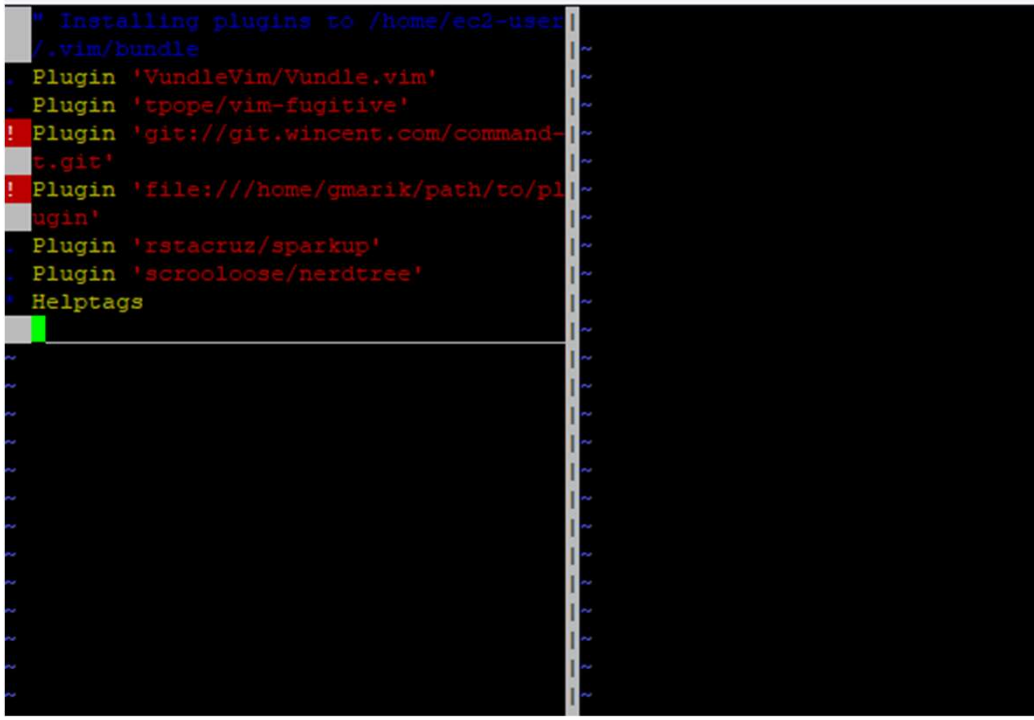
```
set nocompatible              " be iMproved, required
filetype off                  " required

" set the runtime path to include Vundle and initialize
set rtp+=~/.vim/bundle/Vundle.vim
call vundle#begin()
" alternatively, pass a path where Vundle should install plugins
"call vundle#begin('~/some/path/here')

" let Vundle manage Vundle, required
Plugin 'VundleVim/Vundle.vim'

" The following are examples of different formats supported.
" Keep Plugin commands between vundle#begin/end.
" plugin on GitHub repo
Plugin 'tpope/vim-fugitive'
" plugin from http://vim-scripts.org/vim/scripts.html
" Plugin 'L9'
" Git plugin not hosted on GitHub
Plugin 'git://git.wincent.com/command-t.git'
" git repos on your local machine (i.e. when working on your own plugin)
Plugin 'file:///home/gmarik/path/to/plugin'
" The sparkup vim script is in a subdirectory of this repo called vim.
" Pass the path to set the runtimepath properly.
Plugin 'rstacruz/sparkup', {'rtp': 'vim/'}
" Install L9 and avoid a Naming conflict if you've already installed a
" different version somewhere else.
" Plugin 'ascenator/L9', {'name': 'newL9'}

" All of your Plugins must be added before the following line
call vundle#end()            " required
filetype plugin indent on    " required
" To ignore plugin indent changes, instead use:
"filetype plugin on
"
" Brief help
" :PluginList       - lists configured plugins
" :PluginInstall    - installs plugins; append `!` to update or just :PluginUpdate
" :PluginSearch foo - searches for foo; append `!` to refresh local cache
" :PluginClean      - confirms removal of unused plugins; append `!` to auto-approve removal
"
" see :h vundle for more details or wiki for FAQ
" Put your non-Plugin stuff after this line
```

Vim Plugin

❖ Vim Vundle

- Vim 실행 후 명령 모드

-> :PluginInstall



Vim Plugin

❖ .vimrc

- 위치 : home
- .vimrc 필요한 plugin 및 옵션 설정

```
42 " see :h vundle for more details or wiki for FAQ
43 " Put your non-Plugin stuff after this line
```

- 파일 아래 위와 같은 문구가 존재
- 이 문구 아래에 설치할 plugin 작성

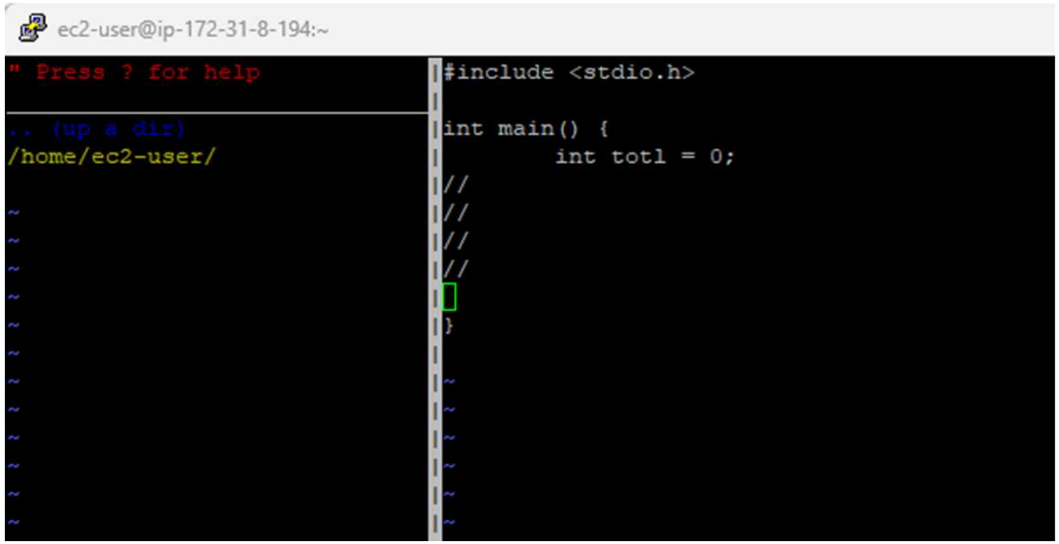
Vim Plugin

❖ Nerdtree

- 복잡한 디렉토리 계층 구조 확인 가능
- 파일 탐색, 읽기, 편집 가능
- Plugin ‘scrooloose/nerdtree’
- Nmap <F9> : NERDTreeToggle<CR>

```
48 Plugin 'scrooloose/nerdtree'
49 nmap <F9> :NERDTreeToggle<CR>
```

- Ctrl+w+w



Vim Plugin

❖ Colors schem : jellybeans
https://github.com/nanotech/jellybeans.vim

```
[ec2-user@ip-172-31-8-194 ~]$ sudo mkdir -p ./vim/colors
[ec2-user@ip-172-31-8-194 ~]$ ls -al
total 32
drwx------. 5 ec2-user ec2-user  148 Aug 16 13:23 .
drwxr-xr-x. 3 root     root       22 Aug 14 01:17 ..
-rw-------. 1 ec2-user ec2-user  397 Aug 16 00:57 .bash_history
-rw-r--r--. 1 ec2-user ec2-user   18 Nov 24  2022 .bash_logout
-rw-r--r--. 1 ec2-user ec2-user  141 Nov 24  2022 .bash_profile
-rw-r--r--. 1 ec2-user ec2-user  492 Nov 24  2022 .bashrc
drwx------. 2 ec2-user ec2-user   29 Aug 14 01:17 .ssh
drwxr-xr-x. 3 ec2-user ec2-user   20 Aug 14 19:45 .vim
drwxr-xr-x. 3 root     root       20 Aug 16 13:23 vim
-rw-------. 1 ec2-user ec2-user 8593 Aug 16 13:23 .viminfo
-rw-r--r--. 1 ec2-user ec2-user 1797 Aug 14 19:47 .vimrc
```

- Vim UI 커스텀 가능
- mkdir –p /.vim/colors
- cd /.vim/colors
- curl -O https://raw.githubusercontent.com/nanotech/jellybeans.vim/m

```
45 colorscheme jellybeans
46 syntax on
```

```
1  #include <stdio.h>
2
3  int main() {
4          int total = 0;
5
6          printf("hello, world\n");
7          for(int i=0; i<10; i++) {
8                  total += i;
9          }
10
11         return 0;
12 }
~
```

```
1  #include <stdio.h>
2
3  int main() {
4          int total = 0;
5
6          printf("hello, world\n");
7          for(int i=0; i<10; i++) {
8                  total += i;
9          }
10
11         return 0;
12 }
~
~
```

Vim Plugin

❖ Rainbow

https://github.com/frazrepo/vim-rainbow

- Plugin ‘frazrepo/vim-rainbow’
- let g:rainbow_active=1

```
51 Plugin 'frazrepo/vim-rainbow'
52 let g:rainbow_active = 1
```

```
    rustc::usage(os::args()[0].clone())
}

fn find_cmd(command_string: &str) -> Option<Command> {
    do COMMANDS.iter().find |command| {
        command.cmd == command_string
    }.map_move(|x| *x)
}

fn cmd_help(args: &[~str]) -> ValidUsage {
    fn print_usage(command_string: ~str) -> ValidUsage {
        match find_cmd(command_string) {
            Some(command) => {
                match command.action {
                    CallMain(prog, _) => printfln!(
                        "The %s command is an alias for the %s program.",
                        command.cmd, prog),
                    _       => ()
                }
                match command.usage_full {
                    UsgStr(msg) => printfln!("%s\n", msg),
                    UsgCall(f)  => f(),
                }
                Valid(0)
            },
            None => Invalid
        }
    }

    match args {
        [ref command_string] => print_usage((*command_string).clone()),
        _                    => Invalid
    }
}

fn cmd_test(args: &[~str]) -> ValidUsage {
    match args {
        [ref filename] => {
            let p = Path(*filename);
            let test_exec = p.filestem().unwrap() + "test~";
            invoke("rustc", &[~"--test", filename.to_owned(),
                              ~"-o", test_exec.to_owned()], rustc::main_args);
NORMAL  master  /usr/local/src/rust/src/librust/rust.rs                    rust  utf-8[unix]  50% ℄ 129
```

Vim Plugin

❖ Indent-guides

https://github.com/vim-airline/vim-airline

- cd ~/.vim/bundle
- git clone git://github.com/preservim/vim-indent-guides.git

- Plugin ‘nathanaelkane/vim-indent-guides’
- let g:indent_guides_enable_on_vim_startup=1

```
54 Plugin 'nathanaelkane/vim-indent-guides'
55 let g:indent_guides_enable_on_vim_startup = 1
```

- “IndentGuidesEnable”로 실행

```
40  # Enable user to choose a new password
41  def lost_password
42    redirect_to(home_url) && return unless Setting.lost_pass
43    if params[:token]
44      @token = Token.find_by_action_and_value("recovery", pa
45      redirect_to(home_url) && return unless @token and !@to
46      @user = @token.user
47      if request.post?
48        @user.password, @user.password_confirmation = params
49        if @user.save
50          @token.destroy
51          flash[:notice] = l(:notice_account_password_update
52          redirect_to :action => 'login'
53          return
54        end
55      end
56      render :template => "account/password_recovery"
57      return
58    else
59      if request.post?
60        user = User.find_by_mail(params[:mail])
```

```
colorscheme bclear
set ts=2 sw=2 et
let g:indent_guides_start_level = 2
```

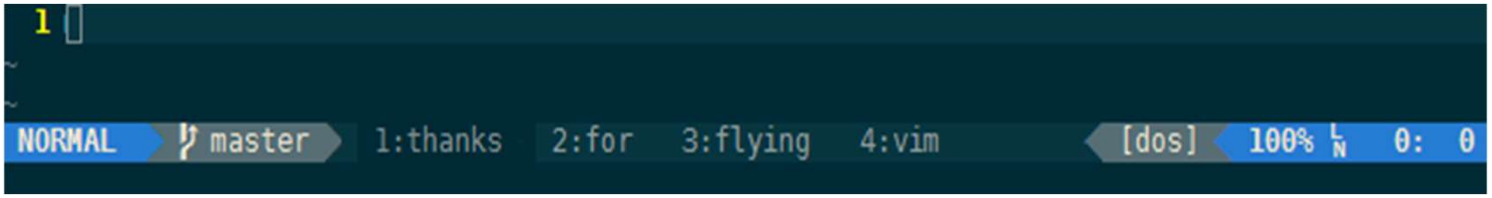
Vim Plugin

❖ Airline

https://github.com/vim-airline/vim-airline

```
Plugin 'vim-airline/vim-airline'
```

- Plugin ‘vim-airline/vim-airline’
- Vim 실행
- Vim command : PluginInstall



Vim Plugin

❖ Airline

Section	meaning
A	현재 디스플레이 모드(입력 모드(insert), 명령 모드(command), 일반 모드(normal), 비주얼 모드(visual)
B	현재 환경 상태(VCS(버전관리시스템) 코드 수정 내용 및 현재 branch 이름 등 표시 ※ B Section을 보기 위해서는 tpope/vim-fugitive 플러그인 있어야 화면에 표시된다. (당연히 git은 설치되어 있어야 하며, git 저장소가(.git) 생성되어 있어야 표시가 된다.)
C	파일 이름 + read-only flag 표시
X	파일 타입
Y	파일 인코딩 형식 표시(utf-8[unix[)
Z	현재 파일내 커서 위치 표시 [ex) 10% ☰ 10/100 ln : 20] 10% - 현재 커서 위치가 파일의 첫 라인으로부터 10% 밑에 있다. ☰ 10/ - 현재 커서 위치가 10번째 라인이다. 100 ln - 파일내 총 라인 수 : 20 - 현재 커서의 20번째 열에 위치한다.

Vim Plugin

❖ Vim 옵션

- set laststatus = 2 : statusline을 보여주는 값

(0 : 출력안함, 1 : 창 2개 이상 일때, 2 : 항상)

- set showmatch : 닫는 괄호 입력 시, 짝이 되는 괄호를 표시

-> 한 눈에 알아보기 편한 기능

- set ruler : cursor 위치의 줄 번호와 행 번호 출력
- set fileencodings = utf8, euc-kr : 인코딩 형식 지정

- set nu : 라인 넘버 출력

-> 프로그래밍 할 때, 에러와 함께 에러난 코드의 라인 출력 가능

Vim Plugin

❖ Vim 옵션

- Smart setting : 검색, 탭, 들여쓰기를 자동으로 설정
  - set smartcase
  - set smaarttap
  - set smartindent

- set ignorecase : 검색 시 대소문자 무시
- set incsearch : 단어 검색 시 글자 입력할 때마다 검색

Vim Plugin

❖ Vim 창 분할

- SP : 가로 창 분할

- 이동
  - Crtl + w, w : 다음 창으로 커서
  - Crtl + w, [H, J, K, L] : 해당 방향으로 커서 이동, vim 에서는 h, j, k, l 이 방향키 대체 가능
- Crtl + w, r : 다음 화면과 위치 전환
- 종료 : q , 모든 창 종료 : qa

Vim Plugin

❖ ctags(red hat linux < 10.2)

- 소스 파일 내의 정의된 전역 변수, 함수, 매크로, 구조체의 정보를 데이터베이스 파일로 생성하는 툴
- 특정 심벌을 찾는데 용이
- 소스 코드 분석

- 설치
  - Yum install ctags
- 데이터베이스 생성
  - Ctags-s
- 경로 설정

```
set tags=/home/choi_gunhee/MyProject/Linux_Kernel_study/linux-5.16.13/tags
```

  - 위치 : .vimrc
  - Set tags=/home/user/workspace/project/tags
  - Set tags =./tags, /usr/src/linux/tags -> 콤마 여러 개 추가 가능

Vim Plugin

❖ ctags(red hat linux < 10.2)

- 소스 파일 내의 정의된 전역 변수, 함수, 매크로, 구조체의 정보를 데이터베이스 파일로 생성하는 툴

```
choi_gunhee@gunhee-linux-94:~/MyProject/Linux_Kernel_study/linux-5.16.13$ ls
arch      CREDITS       Documentation  init     kernel       Makefile  samples   tools
block     crypto        drivers        ipc      lib          mm        scripts   usr
certs     cscope.files  fs             Kbuild   LICENSES     net       security  virt
COPYING   cscope.out    include        Kconfig  MAINTAINERS  README    sound
choi_gunhee@gunhee-linux-94:~/MyProject/Linux_Kernel_study/linux-5.16.13$ ctags -R
choi_gunhee@gunhee-linux-94:~/MyProject/Linux_Kernel_study/linux-5.16.13$ ls
arch      CREDITS       Documentation  init     kernel       Makefile  samples   tags
block     crypto        drivers        ipc      lib          mm        scripts   tools
certs     cscope.files  fs             Kbuild   LICENSES     net       security  usr
COPYING   cscope.out    include        Kconfig  MAINTAINERS  README    sound     virt
choi_gunhee@gunhee-linux-94:~/MyProject/Linux_Kernel_study/linux-5.16.13$
```

Vim Plugin

❖ cscope

- Ctags로 찾기 힘든 전역 변수나 함수의 호출 등을 볼 때 사용
- 대용량 프로젝트의 소스코드 분석에 용이

- C언어로 작성된 소스코드 분석이 주 목적
  - *.c, *.h 등

- C++, java로 작성된 소스코드에도 적용이 가능

Vim Plugin

❖ cscope

- ls 명령어로 현재 디렉토리의 모든 파일 확인

```
[ec2-user@ip-172-31-8-194 ~]$ ls
test2.c  test.c
```

- find ./ -name “*.[파일확장자]” > cscope.files
  - 원하는 파일확장자만 리스트화 시킴

```
[ec2-user@ip-172-31-8-194 ~]$ find ./ -name "*.[Cc]" > cscope.files
[ec2-user@ip-172-31-8-194 ~]$ ls
cscope.files  test2.c  test.c
```

- cscope.files 확인

```
[ec2-user@ip-172-31-8-194 ~]$ cat cscope.files
./test.c
./test2.c
```

Vim Plugin

❖ cscope

- Cscope.files를 이용해 실제 데이터베이스 생성
  - Csope –b –q –k
  - b : for Build, -q : quick symbol lookup , k : for kernel mode

- cscope –i cscope.files
  - Cscope.files를 직접 입력하는 방법
  - Cscope.files에 있는 소스코드 리스트만을 분석하여 데이터베이스로 만듦

```
[ec2-user@ip-172-31-8-194 ~]$ cscope -i cscope.files
[ec2-user@ip-172-31-8-194 ~]$ ls
cscope.files  cscope.out  test2.c  test.c
```

Vim Plugin

❖ cscope initial GUI

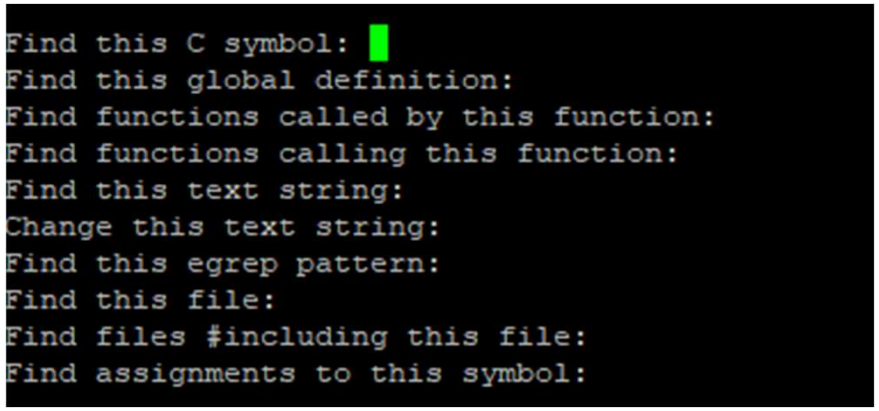
- Cscope 데이터베이스가 정상적으로 생성된 이후 동일한 폴더 내에서 cscope를 실행하게 되면 다음과 같은 실행창이 발생

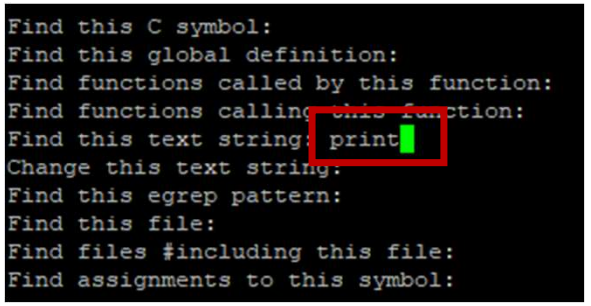
```
[ec2-user@ip-172-31-8-194 ~]$ cscope
```

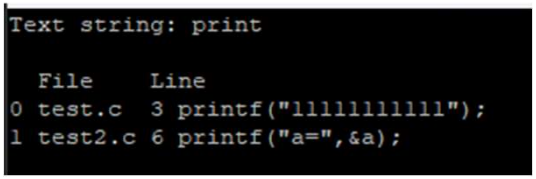
Vim Plugin

❖ cscope

- 키보드 방향키를 이용하여 원하는 input field로 이동
- 검색하고 싶은 내용 입력
- 검색결과가 상단에 출력







Vim Plugin

❖ cscope

- 파일명, 라인, 내용 출력
- 0, 1 번호를 입력하여 파일 수정이 가능하며, enter를 눌러도 바로 수정이 가능

```
#include <stdio.h>
int main(){
        int a = 0;
        int b, c ;

        printf("a=",&a);
}
```

```
Text string: print

  File     Line
0 test.c   3 printf("11111111111");
1 test2.c  6 printf("a=",&a);
```

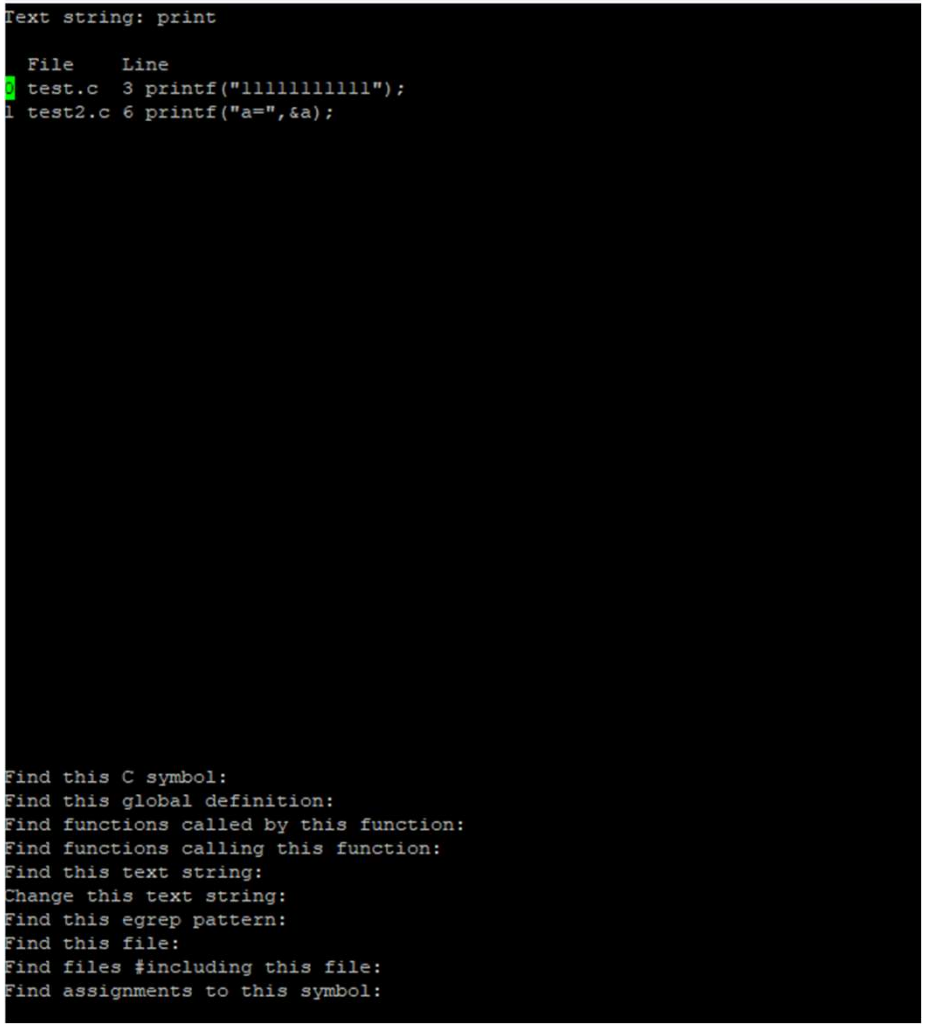
```
Text string: print

  File     Line
0 test.c   3 printf("11111111111");
1 test2.c  6 printf("a=",&a);
```

Vim Plugin

❖ cscope

- Tap으로 input field 전환이 가능
- 각 input field의 기능은 직관적으로 이해하기 쉽게 구성

되어있음



Vim Plugin

❖ cscope main feature

- Find this C symbol
  - int, pointer와 같은 변수뿐 아니라 함수 등을 의미
  - 해당 Symbol의 선언, 초기화, 호출 등 논리적으로 사용된 모든 라인 호출

- Find this global definition
  - 검색한 symbol의 전역 선언문을 선택
  - 함수의 경우 함수 호출문이 아닌 함수 선언문을 출력
  - 변수의 경우 메인 함수를 포함하여 지역 변수가 아닌 전역 변수 부분을 출력

Vim Plugin

❖ cscope main feature

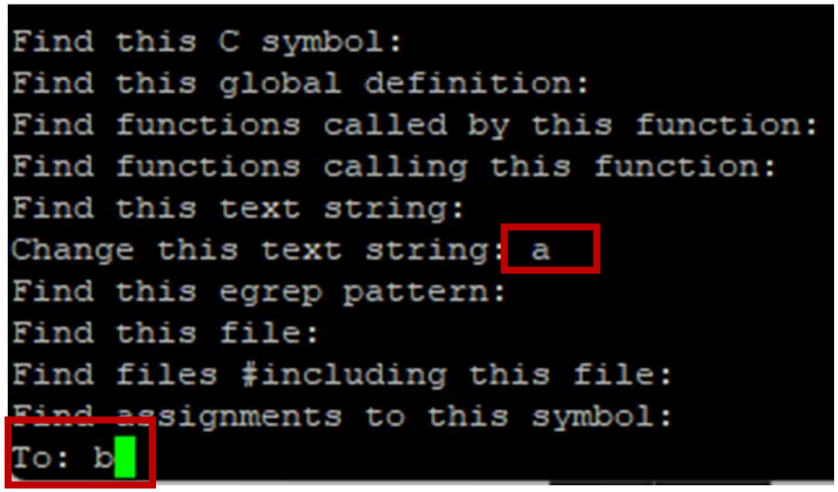
- Find functions called by this function
- Find functions calling this function
  - 검색한 텍스트의 이름을 갖는 함수가 호출하고 있는 함수
  - 해당 함수를 호출하고 있는 함수

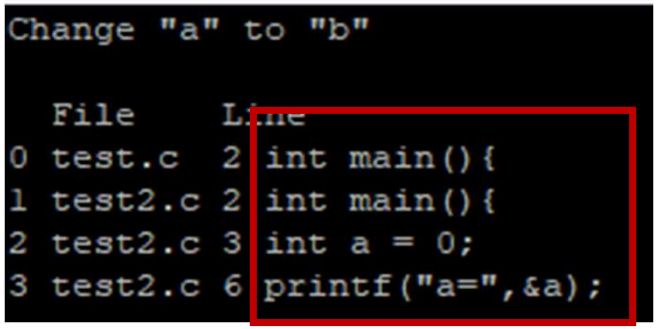
- Find this text string
  - 입력한 텍스트를 cscope DB 전체에서 검색
  - 해당 메소드는 DB에 입력된 소스들 중 입력된 텍스트를 단순히 검색한 결과를 출력

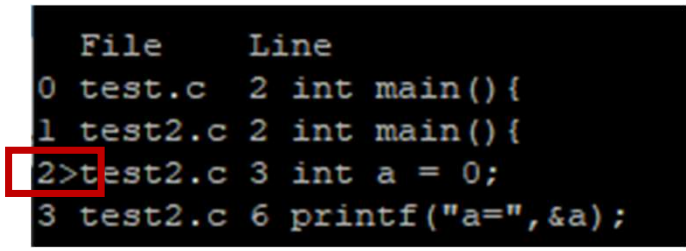
Vim Plugin

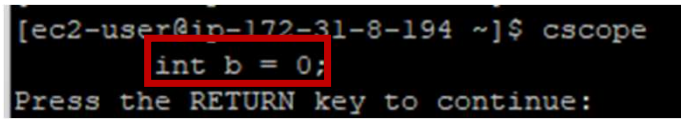
❖ cscope main feature

- Change this text string
  - 검색 결과는 find this text string과 동일
  - 검색된 text 중 일부만 선택하여 출력이 가능









Vim Plugin

❖ cscope main feature

- Find this egrep pattern
  - 메타문자를 이용해 표현되는 패턴을 검색
  - Egrep : grep으로 커버할 수 있는 정규 표현식 규칙과 or 연산자와 같은 몇 가지 기능이 추가된 표현
- Find this file
  - 입력한 텍스트를 파일 이름의 일부 혹은 전체를 탐색하여 출력
- Find file #including this file
  - 입력한 텍스트를 include 하는 파일을 검색하여 출력
  - 라이브러리 또는 직접 작성한 헤더파일 명시하기 때문에 파일들의 관계 분석에 용이

Vim Plugin

❖ cscope

- 쉘 스크립트를 통한 cscope 사용(mkcscope.sh)
  - #!/bin/bash
  - rm -rf cscope.files cscope.out
  - Find . `pwd` \(-name '*.c’ -o -name '*.cpp' -o -name '*.cc' -o -name '*.h' -o -name '*.s' -o -name '*S' \) -print > cscope.files
  - cscope -i cscope.files

```
[ec2-user@ip-172-31-8-194 ~]$ cat /usr/local/bin/mkcscope.sh
#!/bin/bash
rm -rf cscope.out cscope.files
find . 'pwd' \( -name '*.c' -o -name '*.cpp' -o -name '*.cc' -o -name '*.h' -o -name '*.s' -o -name '*.S' \) -print > cscope.files
cscope -i cscope.files
```

Vim Plugin

❖ cscope

- 쉘 스크립트를 통한 cscope 사용

- chmod 755 mkcscope.sh (권한 설정)
- mv mkcscope.sh /usr/local/bin
  - 언제나 실행할 수 있도록 bin 으로 이동

- 프로젝트 최상위에서 스크립트 실행
  - mkcscope.sh cscope.files cscope.out
  - 생성이 완료되면 ctrl+d 입력 후 종료

```
[ec2-user@ip-172-31-8-194 ~]$ mkcscope.sh cscope.files cscope.out
find: 'pwd': No such file or directory
[ec2-user@ip-172-31-8-194 ~]$ ls
cscope.files  cscope.out  test2.c  test.c  test.c~
```

Vim Plugin

❖ cscope

- 쉘 스크립트를 통한 cscope 사용

```
set csprg=/usr/bin/cscope
set csto=0
set cst
set nocsverb

if filereadable("./cscope.out")
        cs add home/ec2-user/cscope.out
endif
set csverb
```

- .vimrc 에 다음과 같은 내용 입력
  - set csprg=/usr/bin/cscope
  - set csto=0
  - set cst
  - set nocsverb

  - if filereadable("./cscope.out")
    - cs add home/ec2-user/cscope.out (프로젝트 경로)
  - Endif
  - set csverb

Vim Plugin

❖ cscope

- 쉘 스크립트를 통한 cscope 사용

- 사용법
  - Cs find [유형] 검색어

Ex) cs find 4 cscope -> “cscope”를 포함하고 있는 문자열 검색

0 또는 s	C 심볼 검색
1 또는 g	정의 검색
2 또는 d	호출되는 함수 검색
3 또는 c	호출하는 함수 검색
4 또는 t	텍스트 문자열 검색
6 또는 e	확장 정규식을 이용한 검색
7 또는 f	파일 이름 검색
8 또는 i	본 파일을 include하는 파일 검색