

System Programming & OS 실습

7. Vim&Shell

정지현, 안석현, 김선재

Dankook University

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

❖ Vim Plugin

- Vim Vundle

❖ Shell

- Shell Script

❖ Vim Vundle

- > 일반적으로 IDE에서 사용하는 기능들 사용 가능

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

[illegible]

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('~/.vim/bundle')

" 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 - Vi IMproved
      version 8.1.3741
      by Bram Moolenaar et al.
Modified by team+vim@tracker.debian.org
Vim is open source and freely distributable
```

```
      Help poor children in Uganda!
type  :help iccf<Enter>      for information

type  :q<Enter>              to exit
type  :help<Enter> or <F1>    for on-line help
type  :help version8<Enter>  for version info
```

```
:PluginInstall
```

```
" Installing plugins to /home/ec2-user
/.vim/bundle
Plugin 'VundleVim/Vundle.vim'
Plugin 'tpope/vim-fugitive'
! Plugin 'git://git.wincent.com/command-
t.git'
! Plugin 'file:///home/gmarik/path/to/pl
ugin'
Plugin 'rstacruz/sparkup'
Plugin 'scrooloose/nerdtree'
* Helptags
```

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

❖ Colors schem : jellybeans

<https://github.com/nanotech/jellybeans.vim>

- 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 }
```

```
[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
```

```
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, _) => println!(  
                        "The %s command is an alias for the %s program.",  
                        command.cmd, prog),  
                    => ()  
                }  
            }  
            match command.usage_full {  
                UsgStr(msg) => println!("%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);  
        }  
    }  
}
```

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])
61             # user not found in db
```

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

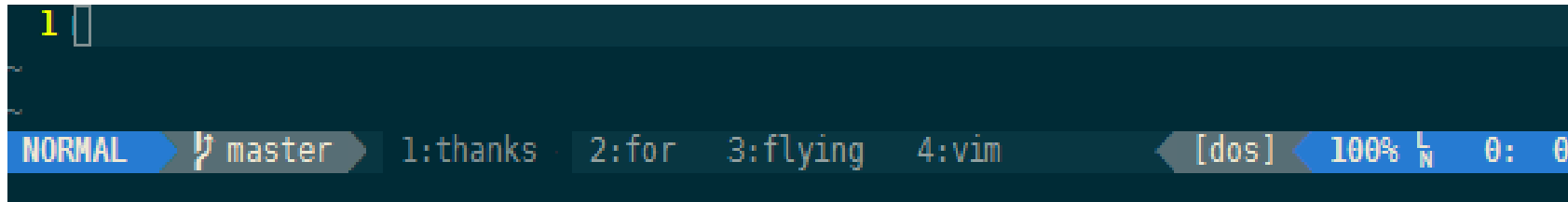
Vim Plugin

❖ Airline

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

- Plugin 'vim-airline/vim-airline'
- Vim 실행
- Vim command : PluginInstall

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



The screenshot shows a Vim editor window with a dark background. The status bar at the bottom is divided into several sections: 'NORMAL' in a blue box, a branch indicator 'master' with a fork icon, '1:thanks' in a grey box, '2:for' in a grey box, '3:flying' in a grey box, '4:vim' in a grey box, '[dos]' in a grey box, '100%' in a blue box, 'L N' in a blue box, and '0: 0' in a blue box. The line number '1' is visible in the top left corner of the editor area.

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 - Vi IMproved
|~
|~          version 8.2
|~          by Bram Moolenaar et al.
|~  Vim is open source and freely distributable
|~
|~  type :h :q<Enter>          to exit
|~  type :help<Enter> or <F1>  for on-line help
|~  type :help version8<Enter> for version info
|~
+-----+
| A | B |                                C                                X | Y | Z | [...] |
+-----+
```

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 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, vim에서는 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 찾는 데 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
 - Cscope -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

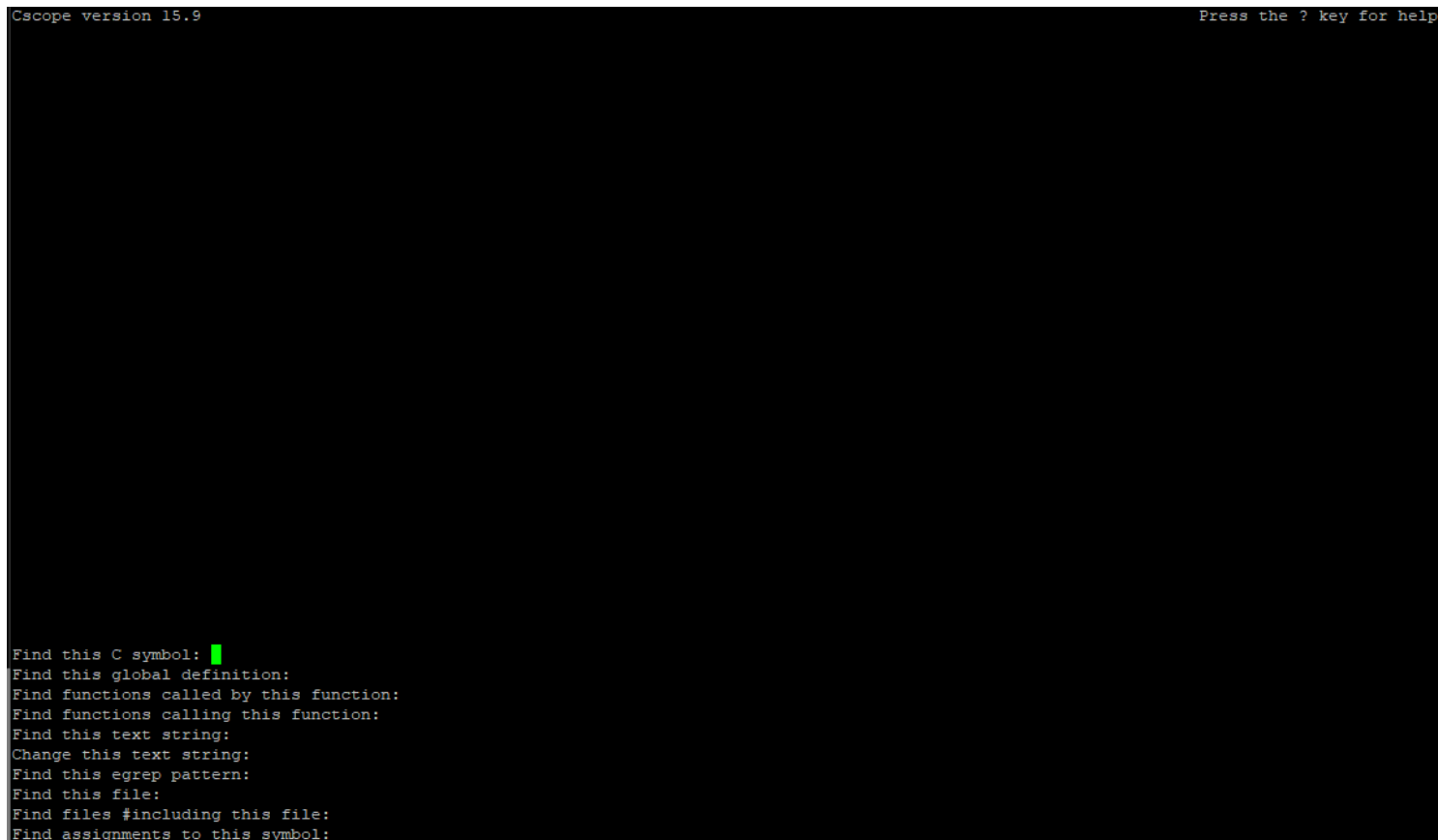
```
[ec2-user@ip-172-31-8-194 ~]$ cscope -i cscope.files
[ec2-user@ip-172-31-8-194 ~]$ ls
cscope.files  cscope.out  test2.c  test.c
```

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
```




```
Cscope version 15.9                                     Press the ? key for help


Find this C symbol: █
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
Find assignments to this symbol:
```

Vim Plugin

❖ cscope

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

```
Find this C symbol:   
Find this global definition:  
Find functions called by this function:  
Find functions calling this function:  
Find this text string:  
Change this text string:  
Find this egrep pattern:  
Find this file:  
Find files #including this file:  
Find assignments to this symbol:
```

```
Find this C symbol:  
Find this global definition:  
Find functions called by this function:  
Find functions calling this function:  
Find this text string: print   
Change this text string:  
Find this egrep pattern:  
Find this file:  
Find files #including this file:  
Find assignments to this symbol:
```

Text string: print

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

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("111111111111");
1	test2.c	6	printf("a=", &a);

Text string: print

	File	Line	
0	test.c	3	printf("111111111111");
1	test2.c	6	printf("a=", &a);

Vim Plugin

❖ cscope

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

```
Text string: print

File      Line
test.c    3 printf("lllllllllll");
test2.c   6 printf("a=", &a);

Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
Find assignments to this symbol:
```

❖ 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

❖ 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

```
Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string: a
Find this egrep pattern:
Find this file:
Find files #including this file:
Find assignments to this symbol:
To: b
```

Change "a" to "b"

	File	Line
0	test.c	2 int main(){
1	test2.c	2 int main(){
2	test2.c	3 int a = 0;
3	test2.c	6 printf("a=", &a);

	File	Line
0	test.c	2 int main(){
1	test2.c	2 int main(){
2	test2.c	3 int a = 0;
3	test2.c	6 printf("a=", &a);

```
[ec2-user@ip-172-31-8-194 ~]$ cscope
int b = 0;
Press the RETURN key to continue:
```

❖ 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

- 셸 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

- 셸 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

- 셸 스크립트를 통한 cscope 사용
- .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

```
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
```


Vim Plugin

❖ cscope

- 셸 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

Shell Script

❖ Shell Script

- 유닉스(Unix), 리눅스(Linux) 등의 운영체제에서 사용되는 스크립트 언어
- 일반적인 명령어들을 모아서 하나의 파일로 작성한 것
-> 명령어들을 순차적으로 실행시켜주는 인터프리터
- 반복적인 작업을 자동화하거나 여러 명령어를 순차적으로 실행하기 위해 사용



Shell Script

❖ Shell Script 기초 문법

- 제일 상단에 `#!/bin/bash` 입력
 - `#!` : 쉼뱅(shebang)으로 스크립트가 실행될 때 사용할 셸 지정
 - `/bin/bash` : bin 하위 폴더에 bash 셸로 실행 지정
- `#!/bin/bash` vs `#!/bin/sh`
 - bash : sh의 개선된 버전으로 기능이 많고 확장성이 뛰어남
 - sh : 가장 기본적인 셸로 최소한의 기능만 제공

bash	sh
<code>#!/bin/bash</code>	<code>#!/bin/sh</code>
더 많은 기능	최소한의 기능
작업 제어 지원	작업 제어 지원하지 않음
유효한 POSIX 셸이 아님	유효한 POSIX 셸
사용하기 편리	bash에 비해 사용하기 어려움
확장된 언어	오리지널 언어

Shell Script

❖ Shell Script 기초 문법

- echo : 문자열을 컴퓨터 터미널에 출력하는 명령어

```
[ec2-user@ip-172-31-8-194 ~]$ vim myshell.sh
[ec2-user@ip-172-31-8-194 ~]$ cat myshell.sh
#!/bin/bash

echo "hello, world"
```

- vim myshell.sh

```
#!/bin/bash
```

```
echo "hello, world"
```

- chmod +x myshell.sh

-> execute 권한 추가

- ./myshell.sh

```
[ec2-user@ip-172-31-8-194 ~]$ chmod +x myshell.sh
[ec2-user@ip-172-31-8-194 ~]$ ls
cscope.files  cscope.out  day5  myshell.sh  test2.c  test.c  test.c~
[ec2-user@ip-172-31-8-194 ~]$ ./myshell.sh
hello, world
```

Shell Script

❖ Shell Script 기초 문법

- 변수 사용
 - 변수 선언 : language="hello world"
 - 변수 사용 : \$language

#!/bin/bash

language="hello world"

echo "\$language"

```
[ec2-user@ip-172-31-8-194 ~]$ cat myshell.sh
#!/bin/bash
language="hello world"
echo "$language"
[ec2-user@ip-172-31-8-194 ~]$ ./myshell.sh
hello world
```

Shell Script

❖ Shell Script 기초 문법

- function 이용

```
#!/bin/bash
```

```
function Taba() {
```

```
    echo "hello, Taba"
```

```
    echo $1
```

```
}
```

```
Taba "hello, $1 print"
```

```
[ec2-user@ip-172-31-8-194 ~]$ cat myshell.sh
#!/bin/bash

function Taba() {
    echo "hello, Taba"
    echo $1
}

Taba "hello, $1 print"
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./myshell.sh
hello, Taba
hello, print
[ec2-user@ip-172-31-8-194 ~]$ ./myshell.sh test
hello, Taba
hello, test print
```

Shell Script

❖ Shell Script 기초 문법

- Local 키워드를 이용한 전역변수 & 지역변수

```
#!/bin/bash
```

```
str="str_hello"
```

```
function Taba() {
```

```
    local str2="str2"
```

```
    echo $str1
```

```
    echo $str2
```

```
}
```

```
Taba
```

```
echo "global $str"
```

```
echo "local $str2"
```

```
[ec2-user@ip-172-31-8-194 ~]$ cat myshell.sh
#!/bin/bash

str="str_hello"
function Taba() {
    local str2="str2"
    echo $str1
    echo $str2
}
Taba
echo "global $str"
echo "local $str2"
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./myshell.sh
str_hello
str2
global str_hello
local
```

Shell Script

❖ Shell Script 기초 문법

- 예약 변수 & 환경 변수
 - 시스템에서 미리 정해둔 변수들이 존재

`#!/bin/bash`

`echo "Home : $HOME"`(사용자 홈 디렉토리)

`echo "Path : $PATH"`(실행파일 찾을 디렉토리 경로)

`echo "pwd : $PWD"`(현재 작업중인 디렉토리 경로)

`Echo "user : $USER"`(사용자 이름)

`Echo "OS type : $OSTYPE"`(운영체제 종류)

```
[ec2-user@ip-172-31-8-194 ~]$ cat system_v.sh
#!/bin/bash

echo "Home : $HOME"
echo "Path : $PATH"
echo "pwd : $PWD"
echo "user : $USER"
echo "OS type : $OSTYPE"
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./system v.sh
Home : /home/ec2-user
Path : /home/ec2-user/.local/bin:/home/ec2-us
bin:/usr/local/sbin:/usr/sbin
pwd : /home/ec2-user
user : ec2-user
OS type : linux-gnu
```


Shell Script

❖ Shell Script 기초 문법

- 매개변수
 - \$1, \$2, \$3 ... \${10}
- 셸 스크립트는 특수변수라고 불리는 매개변수가 존재
- 사용자가 전달하는 변수나 함수의 인자를 전달하는데 사용

```
[ec2-user@ip-172-31-8-194 ~]$ $0./test.sh $11 $22 3 4 ... $nn
```

Shell Script

❖ Shell Script 기초 문법

- 매개변수를 이용한 예제
 - \$1, \$2, \$3 ... \${10}

- ./myshell.sh \$1 \$2 \$3 \$4 입력 시

Hello world(\$1+\$2) 와 더하기 값 출력

```
./myshell.sh $1 $2 $3 $4
```

```
first var : hello  
second var : world  
hello world  
result=7
```

Shell Script

❖ Shell Script 기초 문법

- \${변수:=문자열}
- 변수가 설정되어있지 않거나 NULL 이라면 문자열로 치환

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh  
#!/bin/bash
```

```
var1="Tabal"  
var2=""
```

```
echo "print var1 : $var1"  
echo "print var2 : $var2"
```

```
echo ""  
echo "print var1 : ${var1:=Tabal test}"  
echo "print var2 : ${var2:=hello test}"
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
```

```
print var1 : Tabal  
print var2 :
```

```
print var1 : Tabal test  
print var2 : hello test
```

Shell Script

❖ Shell Script 기초 문법

- \${변수:=문자열}
- 기본값 설정을 통해 출력
 - Username : 입력 값이 없을 경우 “guest”를 출력
 - Greeting : 입력 값이 없을 경우 “welcome”을 출력

```
[ec2-user@ip-172-31-8-194 ~]$ ./myshell.sh  
username:Guest  
greeting: Welcome
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./myshell.sh  
username:seokhyun  
greeting: Taba education
```

Shell Script

❖ Shell Script 기초 문법

- \${변수:?에러 메시지}
- 변수가 설정되어있지 않거나 NULL이라면 에러 메시지 출력 후 종료
- 에러메시지를 설정하지 않을경우
-> "Parameter null or not set" 출력

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh
#!/bin/bash

str1="str1"
str2=""
error_msg="sorry"

echo ${str1:?$error_msg}
echo ${str2:?$error_msg}

[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
str1
./str v.sh: line 8: str2: sorry
```

Shell Script

❖ Shell Script 기초 문법

- 그 외에 변환과 설명

변환	설명
\${변수-문자열}	변수가 설정되지 않은 경우, 문자열을 변수로 치환
\${변수:-문자열}	변수가 설정되지 않았거나 NULL인 경우 문자열을 변수로 치환
\${변수=문자열}	변수가 설정되지 않은 경우, 변수에 문자열을 저장하고 치환
\${변수:=문자열}	변수가 설정되지 않았거나 NULL인 경우 문자열을 변수에 저장하고 치환
\${변수:?에러 메세지}	변수가 설정되지 않았거나 NULL인 경우, 에러메세지 출력하고 종료
\${변수:시작위치}	변수값이 문자열인 경우, 시작 위치부터 문자열 길이 끝까지 출력
\${변수:시작위치:길이}	변수값이 문자열인 경우, 시작 위치부터 길이까지 출력

Shell Script

❖ Shell Script 기초 문법

- 문자열 패턴 및 변경

패턴, 변경	설명
\${변수#패턴}	변수 앞 에서부터 처음 찾은 패턴과 일치하는 패턴 앞의 모든 문자열 제거
\${변수##패턴}	변수 앞 에서부터 마지막 찾은 패턴과 일치하는 패턴 앞의 모든 문자열 제거
\${변수%패턴}	변수 뒤 에서부터 처음 찾은 패턴과 일치하는 패턴 뒤의 모든 문자열 제거
\${변수%%패턴}	변수 뒤 에서부터 마지막 찾은 패턴과 일치하는 패턴 뒤의 모든 문자열 제거
\${#변수}	변수의 길이
\${변수/찾는문자열/바꿀문자열}	변수에서 찾는 문자열을 바꿀 문자열로 변경, 없으면 삭제
\${변수/#찾을문자열/바꿀문자열}	변수에서 문자열에서 찾을 문자열 시작과 맞으면 문자열 변경
\${변수/%찾을문자열/바꿀문자열}	변수에서 문자열에서 찾을 문자열 마지막과 맞으면 문자열 변경

Shell Script

❖ Shell Script 기초 문법

- \${변수-문자열}

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh
#!/bin/bash
name=${USER-"unknown user"}
echo "hello, $name!"
[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
hello, ec2-user!
```


Shell Script

❖ Shell Script 기초 문법

- \${변수:시작위치}

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh
#!/bin/bash
str="hello my name is seokhyun"
echo "${str:6}"
[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
my name is seokhyun
```

Shell Script

❖ Shell Script 기초 문법

- \${변수:시작위치:길이}

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh
#!/bin/bash
str="hello 6my 12name is seokhyun"
echo "${str:6:7}"
[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
my name
```

Shell Script

❖ Shell Script 기초 문법

- \${변수:시작위치:길이}

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh
#!/bin/bash
str="hello 6my 12name is seokhyun"
echo "${str:6:7}"
[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
my name
```

Shell Script

❖ Shell Script 기초 문법

- \${변수#패턴}

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh
#!/bin/bash

filename="test.txt"

echo "${filename#*.}"

[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
txt
```

Shell Script

❖ Shell Script 기초 문법

- \${변수%패턴}

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh
#!/bin/bash

filename="test.txt"

echo "${filename%.txt}"

[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
test
```

Shell Script

❖ Shell Script 기초 문법

- 그 외의 패턴

- \${변수/찾는문자열/바꿀문자열}
- \${변수//찾는문자열/바꿀문자열}
- \${변수/찾는문자열}
- \${변수//찾는문자열}
- \${변수/#찾을문자열/바꿀문자열}
- \${변수/%찾을문자열/바꿀문자열}

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh  
#!/bin/bash
```

```
test_str="hello ann seok hyun hello ann"
```

```
echo ${test_str/hello/hi}
```

```
echo ${test_str//hello/hi}
```

```
echo ${test_str/hello}
```

```
echo ${test_str//hello}
```

```
echo ${test_str/#he/what?}
```

```
echo ${test_str/%lo/what??}
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
```

```
hi ann seok hyun hello ann
```

```
hi ann seok hyun hi ann
```

```
ann seok hyun hello ann
```

```
ann seok hyun ann
```

```
what?llo ann seok hyun hello ann
```

```
hello ann seok hyun hello ann
```

Shell Script

❖ Shell Script 기초 문법

- 조건문

if [첫 번째 조건식] then

수행문

Elif [두 번째 조건식] then

수행문

Else

수행문

fi

```
[ec2-user@ip-172-31-8-194 ~]$ cat if_case.sh
#!/bin/bash
v1=10
v2=10

if [ $v1 = $v2 ]
then
    echo True
else
    echo False
fi

[ec2-user@ip-172-31-8-194 ~]$ ./if_case.sh
True
```

Shell Script

❖ Shell Script 기초 문법

• 조건문

- 사용자가 입력한 2 개의 숫자를 비교하는 스크립트 작성

- 스크립트는 2개의 숫자를 매개변수로 받는다.

- 입력된 2개의 숫자를 조건에 따라 결과 출력

- $A = B$: 두 숫자는 같습니다.
- $A > B$: 첫 번째 숫자가 더 큼니다.
- $A < B$: 두 번째 숫자가 더 큼니다.

- `-eq` : 2 개의 값이 같으면

- `-gt` : 값1이 값 2보다 크면 참

- `-lt` : 값 1이 값2보다 작으면 참

```
[ec2-user@ip-172-31-8-194 ~]$ ./if_case.sh 1 3
v1 < v2
[ec2-user@ip-172-31-8-194 ~]$ ./if_case.sh 1 1
v1 = v2
[ec2-user@ip-172-31-8-194 ~]$ ./if_case.sh 3 1
v1 > v2
```


Shell Script

❖ Shell Script 기초 문법

- 조건문

case \$변수 in

조건값 1)

수행문 ;;

조건값 2)

수행문 ;;

*) //조건 1, 조건 2외

수행문

esac

```
[ec2-user@ip-172-31-8-194 ~]$ cat if_case.sh
```

```
#!/bin/bash
```

```
case $1 in
    hi)
        echo hi
        ;;
    hello)
        echo hello
        ;;
    *)
esac
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./if_case.sh hi
```

```
hi
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./if_case.sh hello
```

```
hello
```

Shell Script

❖ Shell Script 기초 문법

- 반복문

for 변수 in [범위]

do

반복할 수행문

done

```
[ec2-user@ip-172-31-8-194 ~]$ cat iter.sh  
#!/bin/bash
```

```
for num in 1 2 3 4 5  
do  
    echo $num;  
done
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./iter.sh
```

```
1  
2  
3  
4  
5
```

Shell Script

❖ Shell Script 기초 문법

- 반복문

for 변수 in [범위]

do

반복할 수행문

done

```
[ec2-user@ip-172-31-8-194 ~]$ cat iter.sh
```

```
#!/bin/bash
```

```
for file in $HOME/*
```

```
do
```

```
    echo $file;
```

```
done
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./iter.sh
```

```
/home/ec2-user/cscope.files
```

```
/home/ec2-user/cscope.out
```

```
/home/ec2-user/day5
```

```
/home/ec2-user/if_case.sh
```

```
/home/ec2-user/iter.sh
```

```
/home/ec2-user/myshell.sh
```

```
/home/ec2-user/str_v.sh
```

Shell Script

❖ Shell Script 기초 문법

- 반복문

for 변수 in [범위]

do

반복할 수행문

done

- {시작..끝..증가값}

```
[ec2-user@ip-172-31-8-194 ~]$ cat iter.sh
```

```
#!/bin/bash
```

```
for num in {1..10..2}
```

```
do
```

```
    echo $num;
```

```
done
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./iter.sh
```

```
1
```

```
3
```

```
5
```

```
7
```

```
9
```

Shell Script

❖ Shell Script 기초 문법

- 반복문

for 변수 in [범위]

do

반복할 수행문

done

- 배열, 리스트 사용 둘 다 동일

```
[ec2-user@ip-172-31-8-194 ~]$ cat iter.sh
```

```
#!/bin/bash
```

```
arr=("a" "b" "c" "d" "e")
```

```
for str in ${arr[@]}
```

```
do
```

```
    echo $str;
```

```
done
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./iter.sh
```

```
a
```

```
b
```

```
c
```

```
d
```

```
e
```

Shell Script

❖ Shell Script 기초 문법

- 반복문

While [\$변수1 연산자 \$변수2]

do

수행문

done

```
[ec2-user@ip-172-31-8-194 ~]$ cat iter.sh
```

```
#!/bin/bash
```

```
n=1
```

```
while [ $n -lt 5 ]
```

```
do
```

```
    echo $n
```

```
    n=$((n+1))
```

```
done
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./iter.sh
```

```
1
```

```
2
```

```
3
```

```
4
```