# vim + ctags + cscope

## 1. vim

- 표준모드

  키 입력을 통해 vi 에 명령을 내리는 모드

  커서이동, 복사, 붙이기 등의 작업을 수행할 수 있다.

- 입력모드

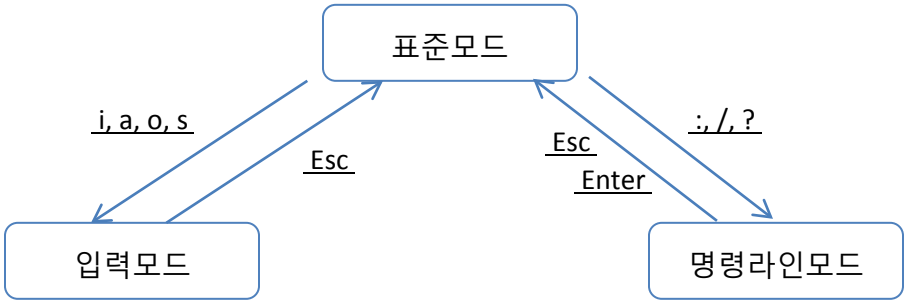
  실제로 문서를 편집하기 위한 모드

  키보드 타이핑 내용이 그대로 화면에 출력된다.

- 명령라인 모드

  명령 입력으로 동작을 수행

  파일을 읽어오거나 저장, 문자열 치환, 외부명령 실행, vi 종료 등을 수행할 수 있다.



## 1.1. vim 환경설정

vi 에서는 에디터 환경을 명령라인모드에 입력 혹은 .vimrc 파일에 저장하여 설정할 수 있다.
[.vimrc] 파일은 각 계정의 홈 디렉토리 아래에 저장하면 된다. (ex> /home/root/.vimrc)

```
set tabstop=2       "탭 간격을 2 칸으로 지정
set visualbell      "비프음 대신 화면 깜빡임
set cindent         "C 스타일 들여쓰기
set autoindent      "자동 들여쓰기
set smartindent     "지능적인 들여쓰기?
set incsearch       "점진적 검색
set ruler           "행,열번호
set hlsearch        "검색어 강조
set number          "왼쪽 줄번호

syntax on           "구문강조
filetype on         "파일 종류에 따른 구문 강조

colorscheme evening "컬러 테마 설정

"ctags
set tags=./tags
set tags+=/usr/src/linux-3.0.4/tags

"cscope
cs add /usr/src/linux-3.0.4/cscope.out
```

## 2. ctags

- 소스코드의 함수, 구조체 등의 태그 리스트를 구성한다.

### 2.1. 설치 방법

- Ubuntu, Debian 기준

```
# apt-get install ctags
```

### 2.2. 태그 생성 방법

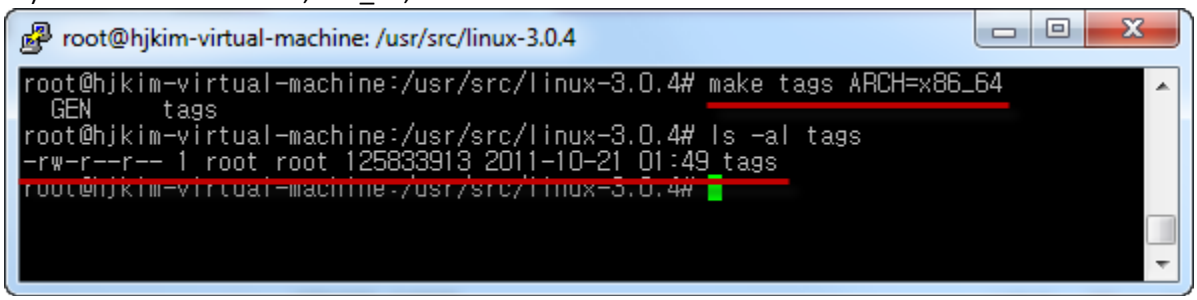
- 태그는 ctags [options] [file(s)] 명령을 이용하여 생성한다. 리눅스 커널과 같이 특정 디렉토리 아래에 있는 소스파일들에 대해 태그를 생성할 때는 -R 옵션을 사용한다.

```
# ctags -R
```

- 리눅스 커널의 태그를 생성할 때는 [ctags –R] 대신에 [make tags]를 이용하여 태그를 생성할 수 있다. 추가로, 특정 아키텍쳐에 대해 태그를 생성할때는 아래와 같이 ARCH 인자를 이용하면 된다.

```
# make tags ARCH=<ARCHITECTURE>
```

Ex) ARCHITECTURE = arm, x86_64, i386 …




### 2.3. 태그 탐색을 위한 vi 설정

- 어느 위치에서든 태그 검색을 가능하도록 하기 위해서 vi 의 명령 모드에 아래 예제를 이용하여 tags 파일의 경로를 등록한다.

```
set tags+=/usr/src/linux-3.0.4/tags
```

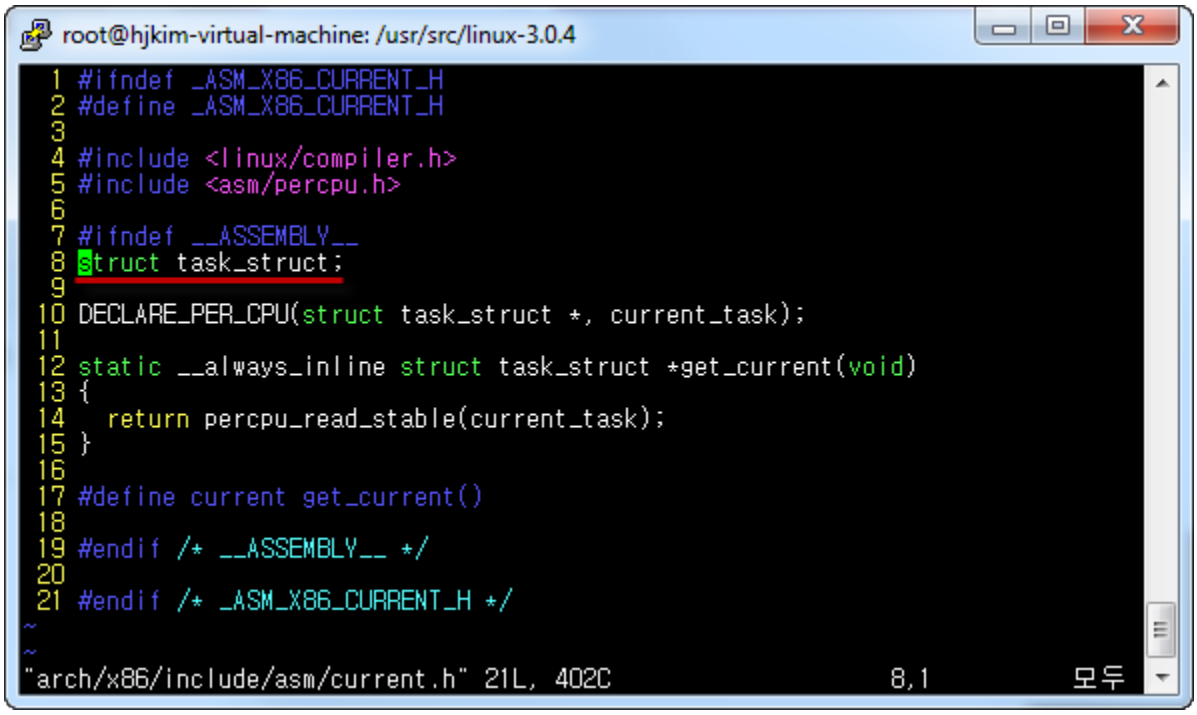
- 혹은 [.vimrc]파일에 위 명령을 등록하면 매번 vi 실행 때마다 명령을 입력할 필요 없이 이용할 수 있다.

## 2.4. 태그 검색하기

- 태그 탐색으로 vi 시작하기

  [vi -t <keyword>]를 이용하면 현재 vi 설정에 등록되어있는 태그파일에서 <keyword>와 일치하는 첫번째 위치로 이동하여 vi 가 실행된다.

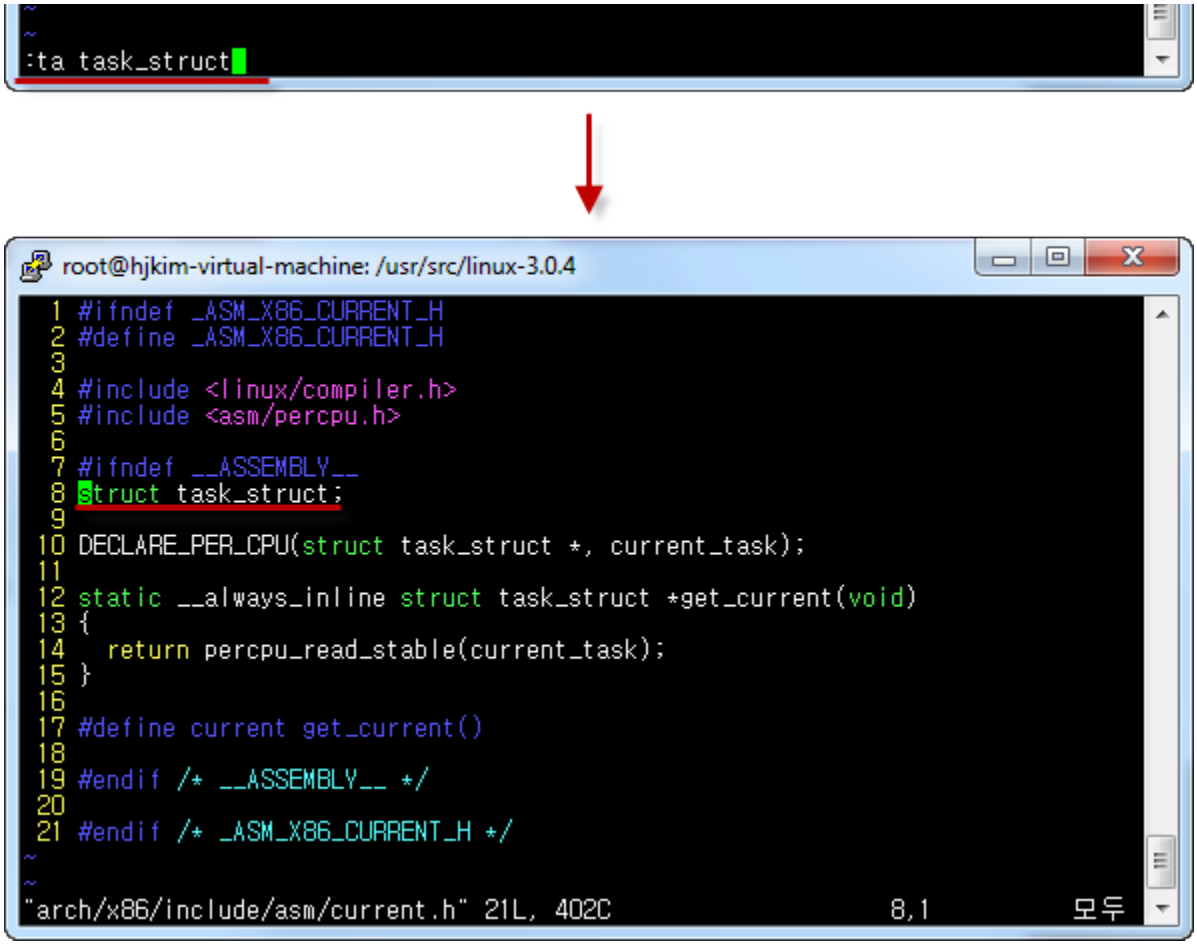
  예를들어, [vi -t task_struct] 로 실행하면 아래와 같은 결과를 볼 수 있다.



- vi [명령모드]에서 태그 탐색하기

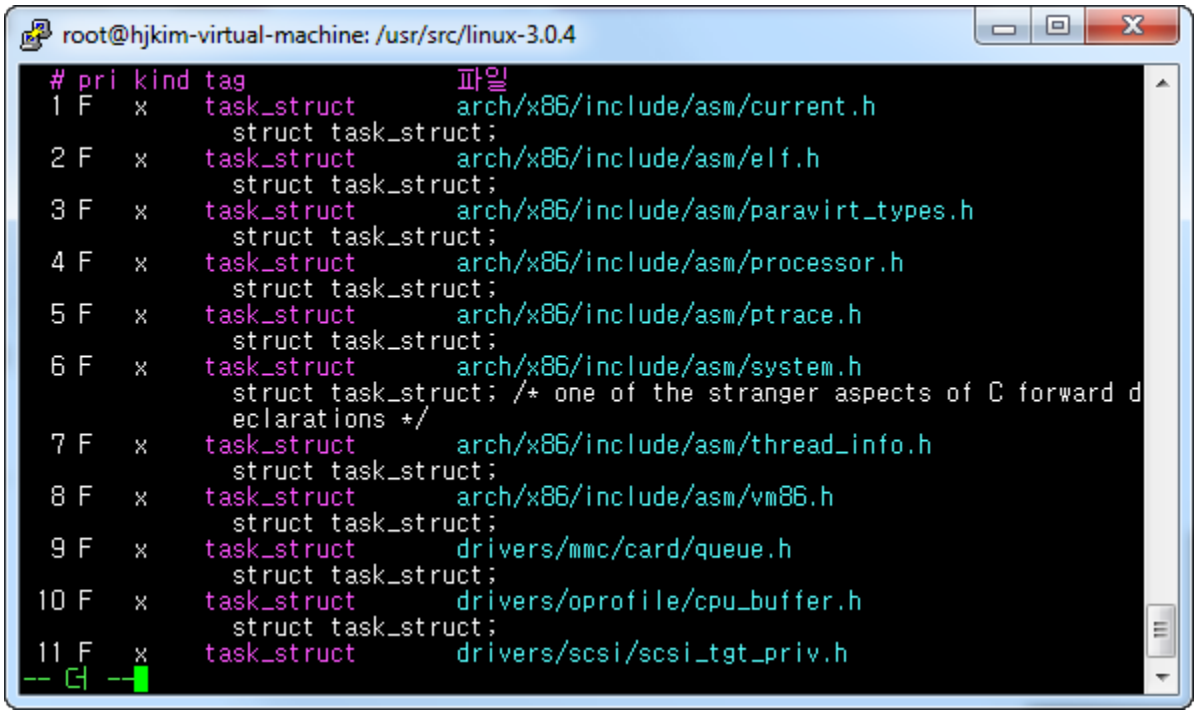
  vi 실행 중 특정 태그를 검색하고자 할 때는 명령모드에서 ta, tj 등의 명령어를 이용하여 검색할 수 있다.

  **ta 명령어**를 사용할 경우, vi 실행 중 명령모드에서 [:ta task_struct]를 입력하면 위쪽과 같은 결과를 볼 수 있다.




  다음 태그로 이동할 때는 [:tn], 이전 태그로 이동할때는 [:tp]를 이용한다.

**tj 명령어**를 이용하면 키워드와 일치하는 태그가 한 개일 경우 바로 이동, 두 개 이상일 경우 목록을 출력하고 그 중 선택하여 이동할 수 있다. 예를 들어, [:tj ext4]를 입력하면 아래와 같은 목록이 나오고, 그 중 번호를 선택하여 해당 위치로 이동할 수 있다.




## 2.5. ctags 명령어 목록

- vi 의 표준모드 또는 명령라인 모드에서 자주 사용하는 ctags 명령어는 아래와 같다.

| 명령 | 설명 |
|---|---|
| :ta keyword | keyword 와 일치하는 태그 위치로 이동 |
| :ta /keyword | keyword 가 포함된 태그 검색 |
| :tj keyword | keyword 와 일치하는 태그 목록을 출력하고 선택하여 이동<br>(일치하는 태그가 한개일 경우 바로 이동) |
| Ctrl + ] | 커서가 위치한 keyword 의 정의 부분으로 이동 |
| Ctrl + t | 이전 위치로 이동 |
| :tn | 다음 태그로 이동 |
| :tp | 이전 태그로 이동 |

# 3. cscope

- ctags 는 변수, 함수, 매크로, 구조체등은 검색되는 반면 함수를 호출한 부분, 함수에 의해 호출되는 부분들은 검색이 되지 않는다.
- ctags 의 부족한 부분들을 채우기 위해 cscope 을 사용한다.
- 심볼(변수, 함수, 매크로, 구조체 등), 전역 선언, 특정 함수에 의해 호출되는 함수, 특정 함수를 호출하는 함수, 문자열 등을 검색할 수 있다.

## 3.1. 설치 방법

- Ubuntu, Debian 기준

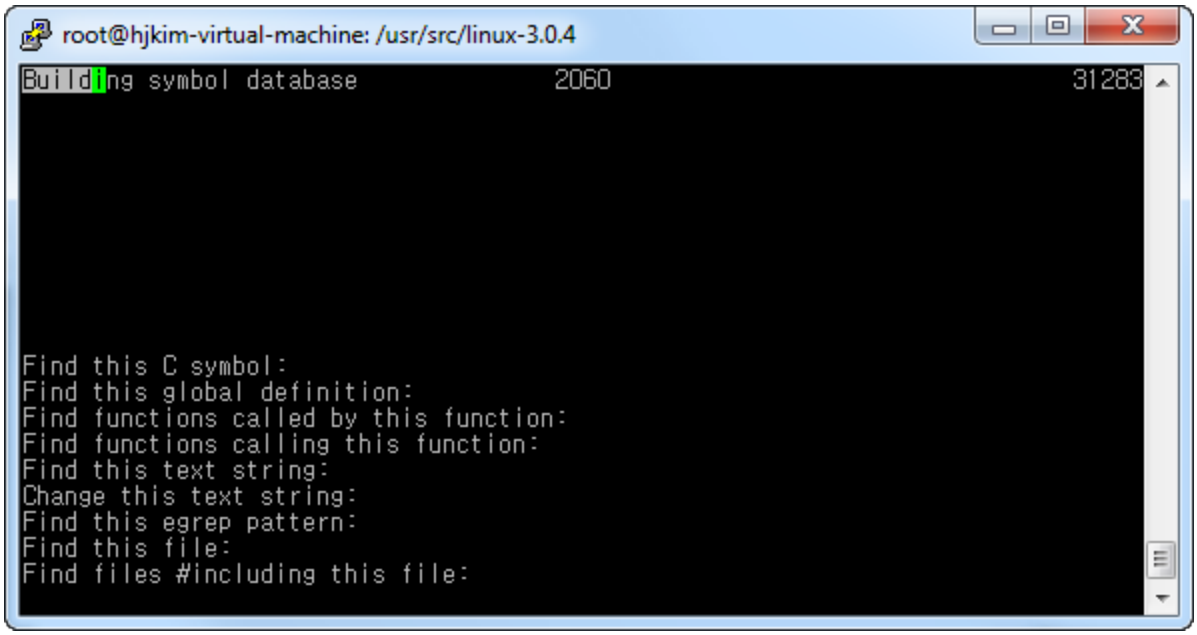
```
# apt-get install cscope
```

## 3.2. Cscope Database 생성 방법

- Cscope Database 를 만들기 위해서 우선 파일 목록을 만들어야 한다. Find 명령어를 이용하여 파일 목록을 생성한다.

```
# find . \( -name '*.c' -o -name '*.h' -o -name '*.s' -o -name '*.S' \) -print > cscope.files
```

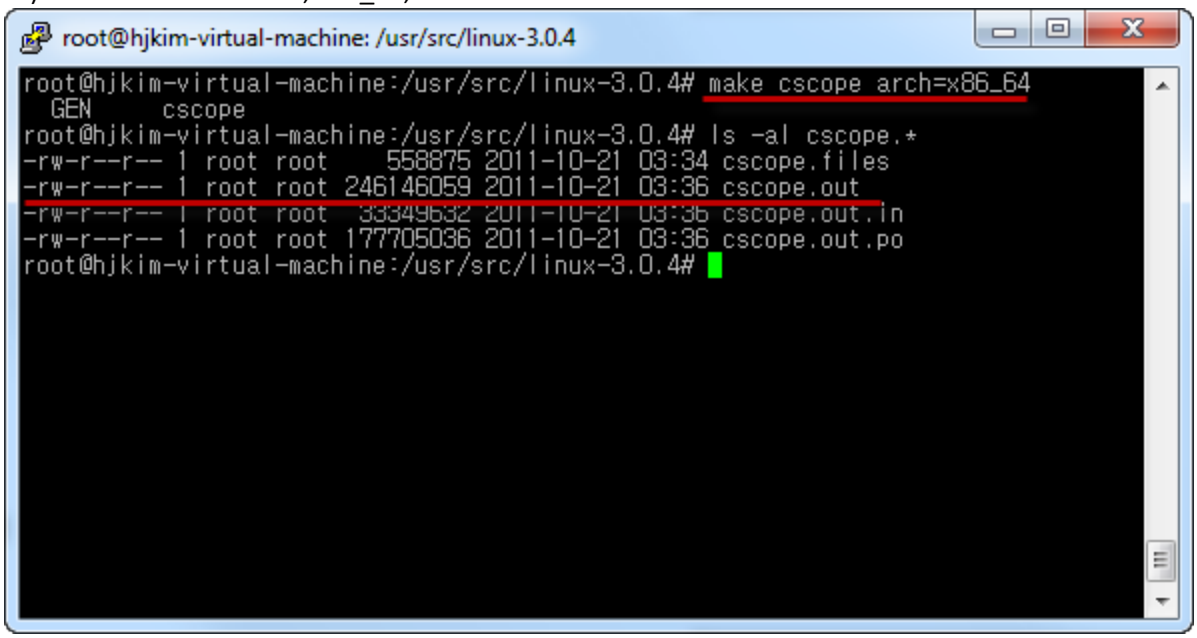
- 그리고나서, `cscope -i <filelist file>`명령을 이용하여 데이터베이스를 생성한다. 생성된 파일의 이름은 `cscope.out` 이다.



- 리눅스 커널의 cscope database 를 생성할때는 [make cscope]명령어를 이용하여 생성할 수도 있다. ctag 와 마찬가지로, arch 인자를 이용하여 특정 아키텍쳐에 대하여 cscope database 를 생성하는것도 가능하다.

```
# make cscope ARCH=<ARCHITECTURE>
```

Ex) ARCHITECTURE = arm, x86_64, i386 …




## 3.3. cscope databae 탐색을 위한 vi 설정

- vi 에서 명령모드에 아래 예제와 같이 입력하여 vi 가 cscope database 를 이용할 수 있게 만든다.

```
: cs add /usr/src/linux-3.0.4/cscope.out
```

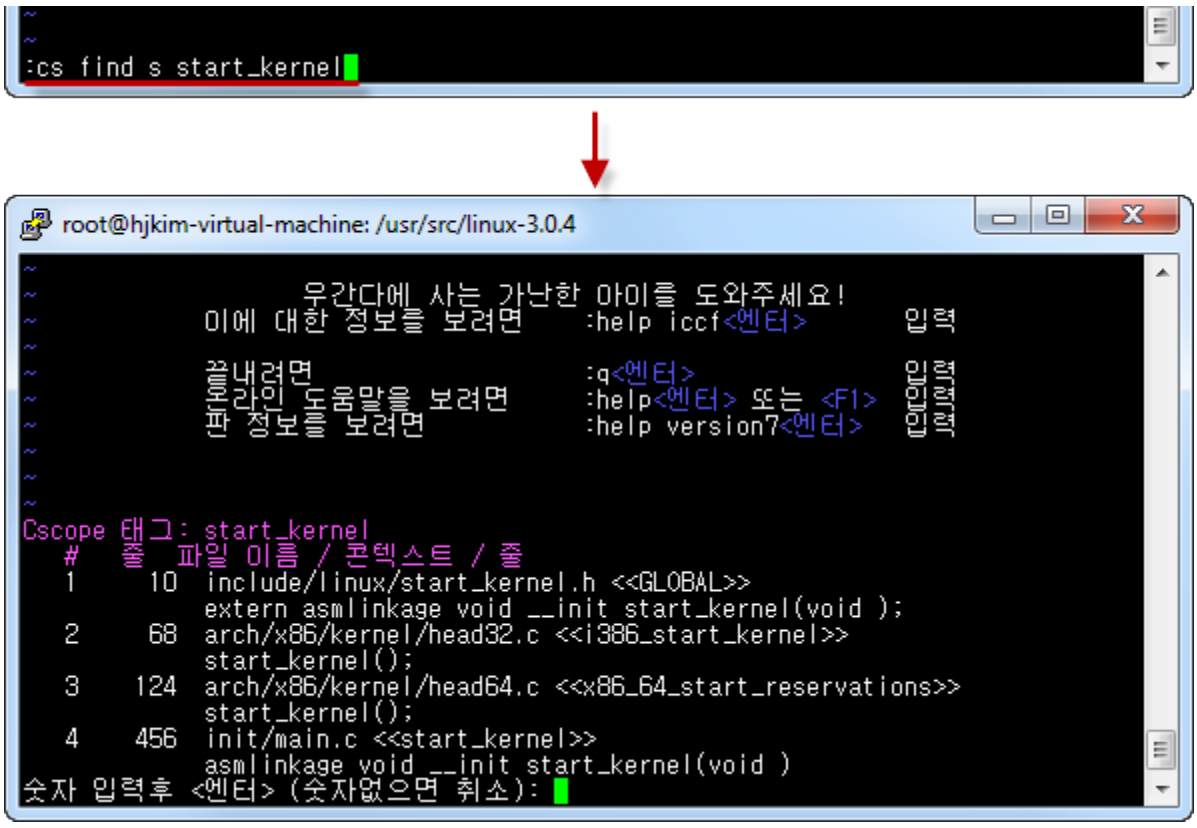
- 혹은 [.vimrc] 파일에 위 명령을 추가하여 설정을 할 수도 있다.

## 3.4. vi 에서 cscope 를 이용해 검색하기

- cscope 는 [:cs find (검색 유형) (검색어)] 형태로 검색을 한다. 이 때 사용되는 검색 유형은 아래와 같다.

| 검색 유형 | 설명 |
|---|---|
| s | C 심볼을 검색한다. |
| g | 전역 선언을 검색한다. |
| d | 함수에 의해 호출되는 함수들을 검색한다. |
| c | 함수를 호출하는 함수들을 검색한다. |
| t | 텍스트 문자열을 검색한다. |
| e | 확장 정규식을 사용하여 검색한다. |
| f | 파일 이름을 검색한다. |
| i | 이 파일을 인클루드 하는 파일을 검사한다. |

- 예를 들어, start_kernel 이라는 함수를 검색할 경우 아래와 같이 입력한다.




이 때 검색할 태그 번호를 선택하고 엔터를 누르면 해당 태그로 이동한다.

다음 검색어는 [:cn] 이전 검색어는 [:cp]로 이동할 수 있다.