

Open Source Software

Vim 편집기



About..

컴퓨터소프트웨어공학과

김 원 일

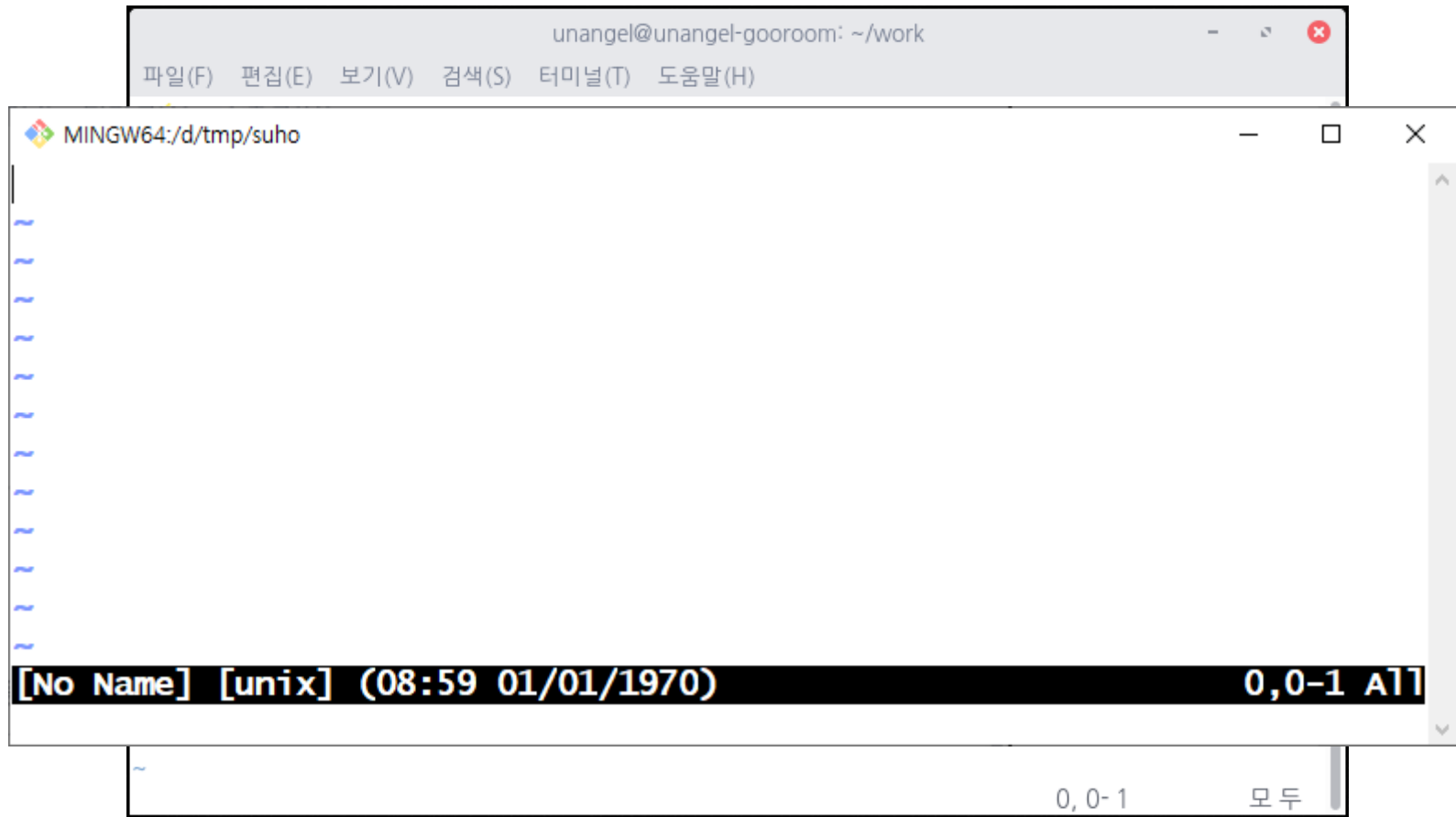


vi Editor – 1



- vi 편집기

- Linux의 대표적인 문서 편집기
- 최근 Vi 기능이 향상된 VIM(Vi IMproved)이 보편적으로 사용





vi Editor – 2



- **vi 편집기**

- Linux의 대표적인 **문서 편집기**로 설치 시 기본적으로 포함
- 미국 캘리포니아 대학의 대학원생 Billy Joy가 개발
 - 라인 편집기인 **ed**를 새롭게 개선
- **화면 단위로 프로그램이나 파일을 편집**
 - 80문자의 20개 라인 정도
 - 사용자는 커서를 이동시키면서 파일의 내용을 수정
 - 키보드 외의 다른 입력 장치가 없이도 모든 작업이 가능

- **vi의 모드**

- **입력** 모드 : 일반 텍스트 편집기의 입력 상태와 동일한 상태
- **명령어** 모드 : 커서의 이동을 통해 변경, 교체 및 검색하는 편집
- **라인** 모드 : ESC 키 입력 상태. 명령어 입력을 통한 기능 수행

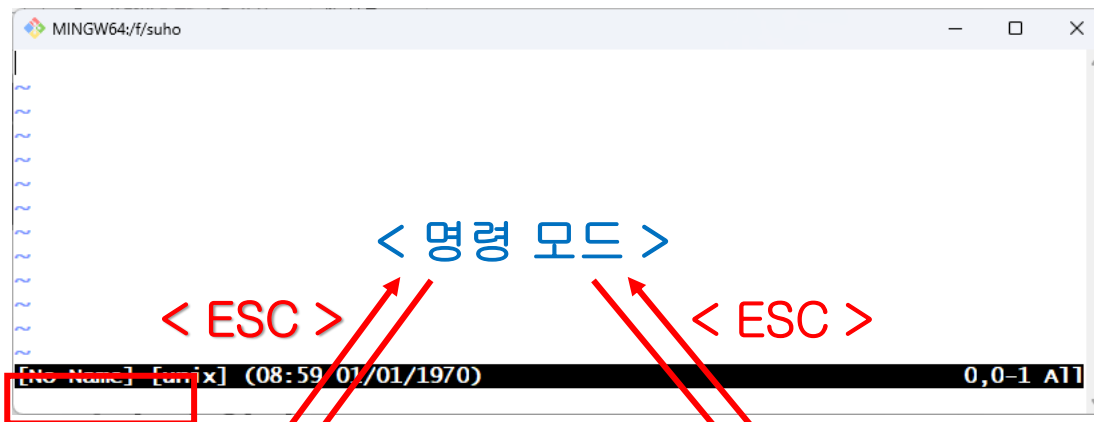


vi Editor – 3



• vi 편집기의 모드

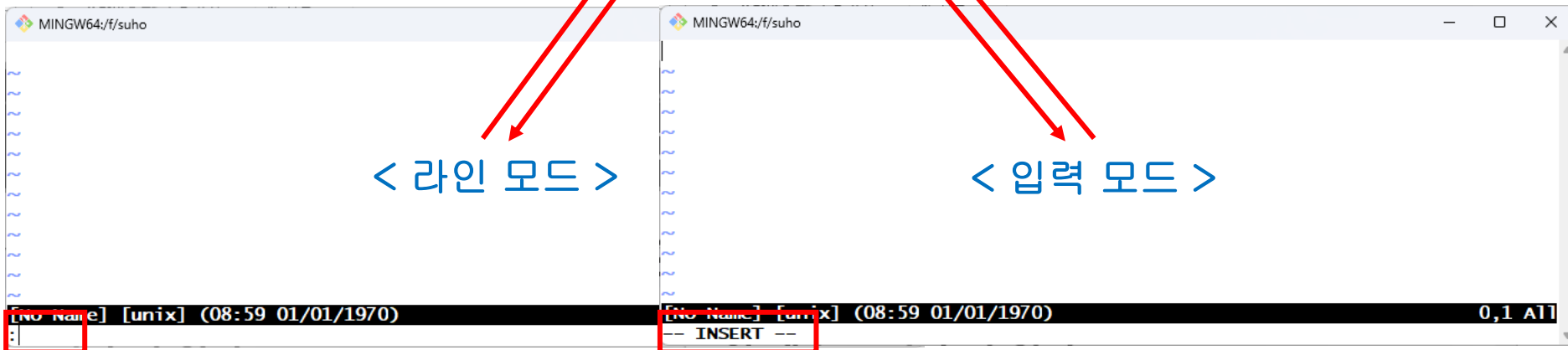
- 입력 모드 : 일반적인 텍스트의 입력이 가능한 모드
- 명령 모드 : vi의 텍스트 편집 및 읽기 모드
- 라인 모드 : vi가 지원하는 명령어 수행 모드



< ESC >

< 라인 모드 >

< 입력 모드 >





• 명령 모드

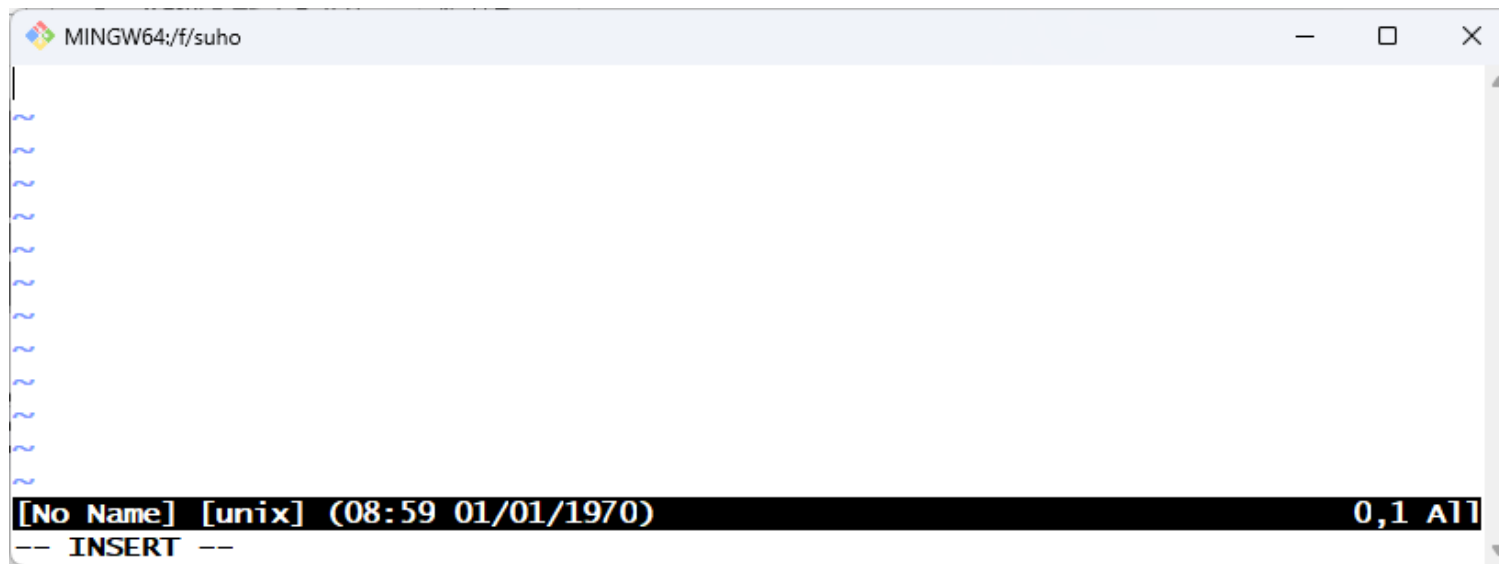
- 문서 편집을 수행할 수 없으며 보기만 할 수 있는 모드
- 키 입력을 통해 명령을 실행해야 하므로 읽기 전용 상태처럼 문서 확인
- 키 입력을 통해 다른 모드로 변경이 가능
 - 입력 모드로 변경 : 명령 모드에서 i, a, o 키를 입력하면 변경
 - 라인 모드로 변경 : 명령 모드에서 ":"을 입력하면 변경
- 좌측 하단에 아무런 내용이 출력되지 않는 상태





• 입력 모드

- 문서 편집을 수행할 수 있는 모드로 일반 편집기와 동일한 상태
- 키보드 입력은 모두 일반적인 입력으로 처리되어 문서 편집이 가능
- ESC 입력으로 명령 모드로 되돌아갈 수 있음
- 입력 모드에서는 직접적으로 라인 모드로 변경이 불가능
- 좌측 하단에 "-- INSERT --" 문자열이 표시되는 것으로 모드 확인





• 라인 모드

- 명령 또는 입력 모드가 아니므로 읽기 전용 처럼 문서를 확인하는 모드
- 가장 아래쪽에 :, / 등으로 현재 문서와 관련된 명령어 실행
- : 입력
 - 편집 중인 파일의 저장과 종료와 같은 명령어 실행
 - 편집기의 환경 설정 등을 수행하여 표시 또는 입력 방법 변경
- / 입력 : 현재 문서에서 특정 문자열을 검색하는 용도의 명령어 실행





• vi 편집기의 모드 변환

- 입력 모드 → 명령 모드 : ESC 키를 입력하여 변환
- 명령 모드 → 입력 모드 : 아래 표의 명령어를 입력하여 변환
- 라인 모드 : ESC 키를 누르고 콜론(:), /, ?을 입력
- 어떤 모드 인지 모를 경우 ESC키 2-3회 입력 → 명령모드

입력모드	내 용
a	현재 위치의 다음부터 입력 시작
A	현재 줄의 끝에서부터 입력 시작
i	현재 위치의 앞에서부터 입력 시작
I	현재 줄의 처음에서 입력 시작
o	현재 줄과 다음 줄 사이에 입력 시작
O	현재 줄과 앞 줄 사이에 입력 시작



vi 명령어 - 1



- vi 편집기 이동 명령

- 기본적인 커서의 이동

- 커서의 이동은 **H, J, K, L** 키를 이용 좌,하,상,우로 커서 이동
 - 모든 편집 명령 앞에 숫자를 붙이면 숫자만큼 의미가 포함됨
 - Ex) 3h : 좌로 3칸 이동

- 단어 단위 커서의 이동

- 한번에 한 문자씩 이동하는 단어 단위 커서를 지원

명령어	내 용
e or E	커서를 다음 단어의 끝 글자로 이동
b or B	커서를 이전 단어의 첫 글자로 이동



- vi 편집기 이동 명령
 - 라인 단위 커서 이동 방법

명령어	내 용
-	커서를 이전 줄의 처음으로 이동
+	커서를 다음 줄의 처음으로 이동
Enter	커서를 다음 줄의 처음으로 이동
0	커서를 현재 줄의 맨 앞으로 이동
\$	커서를 현재 줄의 맨 끝으로 이동
^	커서를 현재 줄의 첫 글자(공백이나 탭이 아닌)로 이동



vi 명령어 - 3



• vi 편집기 이동 명령

- 문단 단위 커서 이동 방법

- 문장의 시작과 끝으로 이동하기 위해서 (또는) 를 이용한다.
- 문단의 시작과 끝에는 { 또는 } 를 이용한다.

입력 키	내 용
(문장의 시작으로 이동
)	문장 끝으로 이동하여 다음 단어의 시작 첫 칸으로 커서 이동
{	문단의 시작으로 이동
}	문단 끝으로 이동
G	문서의 마지막 줄로 이동
nG	n번째 줄로 이동
`	이전의 커서 위치로 이동
M	화면 중간 줄에 커서 이동

줄 번호를 보이게 하려면 :set nu 줄 번호를 보이지 않게 하기 위해서는 :set nonu



- vi 편집기 삭제명령

명령어	내 용
x	커서가 있는 문자 삭제
X	커서의 왼쪽 문자 삭제
D	커서부터 줄의 끝까지 삭제
dd	현재 줄의 전체 삭제 및 복사 (즉, ctrl + x)
dw / db	커서 앞에 있는 한 글자를 삭제/ 커서 앞에 있는 한 단어를 삭제
dG	커서부터 편집 버퍼의 끝까지 삭제
d1G	커서부터 편집 버퍼의 맨 앞까지 삭제
d) / d}	문장의 나머지 삭제 / 문단의 나머지 삭제
dH	화면의 시작까지 삭제



vi 명령어 - 5



• vi 편집기 치환 명령

- 문자 및 문자열 단위의 치환

입력 키	내 용
r	커서가 있는 문자 대치
R	입력 모드로 한 문자씩 덮어씀
s	커서가 있는 문자 삭제 후 입력모드로 전환
S	커서가 있는 줄을 삭제한 후 입력모드로 전환
cb	커서가 있는 앞 문자 삭제 후 입력모드
u	변경된 내용의 취소

-단어 단위의 치환

입력 키	내 용
cw	커서 위치의 문자열 치환
cW / cB	공백으로 구분된 뒤 단어를 삭제 후에 입력모드 공백으로 구분된 앞 단어 삭제 후 입력모드로 전환



- vi 편집기 치환 명령

- 라인 및 문단 단위의 치환 방법

입력키	내 용
s	커서가 있는 문자 삭제 후 입력모드로 전환
S	커서가 있는 줄을 삭제한 후 입력모드로 전환
C	커서가 있는 라인의 나머지를 삭제하고 입력모드로 전환
cc	커서가 있는 라인을 삭제하고 입력모드
cO	커서에서부터 라인의 시작까지 텍스트 바꾸기
c)	문장의 나머지 바꾸기
c}	문단의 나머지 바꾸기
cG	파일의 나머지 바꾸기
cH	화면의 시작까지 바꾸기
cL	화면의 나머지 바꾸기



vi 명령어 - 7



- vi 편집기 텍스트 이동 및 복사 명령
 - 텍스트 이동 명령

입력 키	내 용
p	삭제나 복사된 텍스트를 커서가 있는 문자나 라인 뒤에 삽입
P	삭제나 복사된 텍스트를 커서가 있는 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 커서 앞으로 삽입
d) P	문장의 나머지로 이동
d} P	문단의 나머지로 이동
dG P	파일의 나머지로 이동
dH P	화면 시작 부분으로 이동
dL P	화면의 나머지로 이동



- vi 편집기 텍스트 검색 명령

- 텍스트 검색 명령

- 슬러시 키는 패턴 검색을 위한 명령
 - 슬러시 다음에 원하는 검색 단어를 입력
 - 커서가 있는 곳부터 시작하여 검색 패턴을 검색

입력키	내 용
/pattern	텍스트에서 앞으로 패턴 검색
n	앞 또는 이전 검색 반복
N	반대방향으로 이전 검색 반복



vi 편집기 기초 실습



- 편집기 사용을 위한 기초 실습

- 실습은 아래와 같이 2가지 방식으로 진행

- 1) 파일명 입력 없이 문서를 편집하고 저장하는 실습

- 2) 파일명 입력 후 문서를 편집하고 저장하는 실습

- 각 실습은 한 단계씩 천천히 진행하여 익숙해지는 것이 목표

- 실습 진행 시 유의 사항

- 진행이 빠르면 바로 이야기할 것!

- 따라오지 못했다면 바로 손들 것!

- 수업 진행 중에 잠시 시간이 필요하다면 손들 것!



vi 편집기 기초 실습(1) - 1



• 새 파일 생성

- 실습할 디렉토리를 생성하고 이동, "git init"으로 저장소로 초기화
- 터미널에서 vi + <ENTER>로 vi 실행
- 파일명을 지정하지 않은 상태이기 때문에 편집 후 반드시 저장해야 함
- "i", "a" 또는 "o" 키를 입력하여 입력 모드로 변경
- 아래 코드를 그대로 입력. 입력 모드에서는 일반 메모장과 동일

```
MINGW64:/f/suho
#include <stdio.h>

void main()
{
    printf("Hello World\n" );
}

~
~
~
~
~

[No Name] [+] [unix] (08:59 01/01/1970) 7,1 All
-- INSERT --
```



vi 편집기 기초 실습(1) - 2



• 파일 저장

- 모두 입력 한 다음 ESC 키 2회 입력으로 명령 모드로 변경
 - ESC를 입력할 때 화면 깜빡임이 발생하면 명령 모드로 변경된 것
- 라인 모드에서 파일 저장을 위해 ":"을 입력
- 명령어를 입력할 수 있는 상태가 되면 "w file.txt"를 입력

```
MINGW64:/f/suho
#include <stdio.h>

void main()
{
    printf("Hello World\n" );
}
~
~
~
~
~
~
[No Name][+] [unix] (08:59 01/01/1970) 7,0-1 All
:w file.txt
~
file.txt [unix] (22:34 23/03/2025) 7,0-1 All
"file.txt" [New] [unix] 7L, 64B written
```



vi 편집기 기초 실습(1) - 3



• 파일 확인

- 라인 모드에서 명령어가 한번 실행되면 명령 모드로 돌아감
- 명령 모드에서 다시 ":" 입력으로 라인 모드로 변경
- "q" 명령으로 vi 편집기를 종료하고, "cat file.txt"로 내용 확인

```
MINGW64:/f/suho
#include <stdio.h>

void main()
{
    pri
}

~
~
~
~
~
file.txt [u
:q

MINGW64:/f/suho
unange1@DESKTOP-I8OQG85 MINGW64 /f/suho (master)
$ vi

unange1@DESKTOP-I8OQG85 MINGW64 /f/suho (master)
$ cat file.txt
#include <stdio.h>

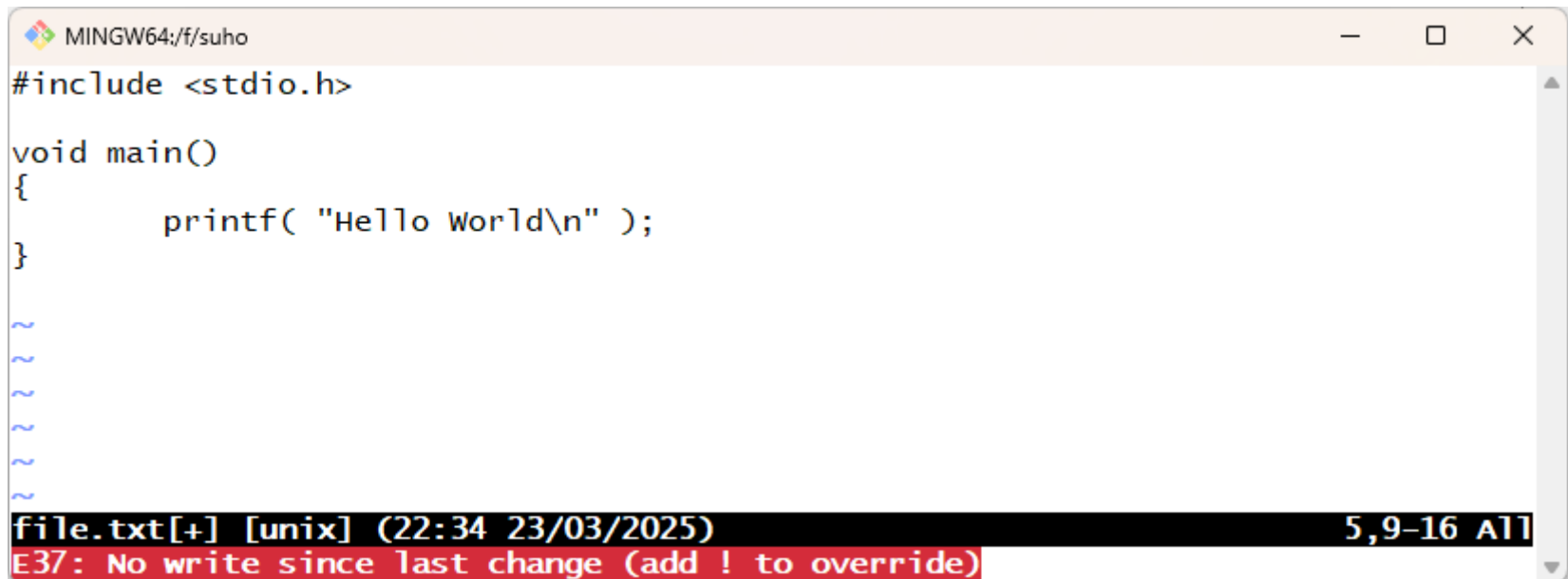
void main()
{
    printf("Hello world\n" );
}

unange1@DESKTOP-I8OQG85 MINGW64 /f/suho (master)
$
```

❖ vi 편집기 기초 실습(1) - 4

• 오류 메시지

- "w file.txt" 명령 이후 아래와 같이 붉은색 오류가 발생하는 경우
- 파일 기록 후, 수정이 발생한 상태에서 "q"로 종료하려는 경우 발생
- 파일 내용을 저장하지 않고, vi를 종료하려 했기 때문에 발생
- 해결 1 : 저장하려면 라인 모드에서 "w"를 다시 입력하고 "q"로 종료
- 해결 2 : 저장하지 않고 vi 종료를 위해 "q!" 입력
- 해결 3 : 편집을 계속하려면 ESC 키 입력 후 수정 계속



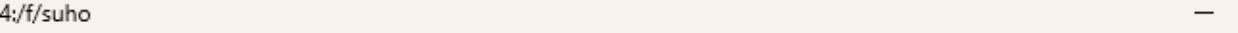
```
MINGW64:/f/suho
#include <stdio.h>

void main()
{
    printf( "Hello World\n" );
}

~
~
~
~
~
~
file.txt[+] [unix] (22:34 23/03/2025) 5,9-16 All
E37: No write since last change (add ! to override)
```



- **파일명 입력으로 파일 생성**

- 
- ```
MINGW64:/f/suho
unange1@DESKTOP-I8OQG85 MINGW64 /f/suho (master)
$ vi first.txt
```





## vi 편집기 기초 실습(2) - 2



### • 파일 내용 입력

- "i", "o", "a" 등의 키 입력으로 파일에 내용을 입력
- 아래의 C언어 코드를 입력
- 이전 실습과 달리 파일이 존재하기 때문에 저장 후 종료 명령이 변경
- 코드 입력 완료 후, ESC 입력하여 명령 모드 전환
- ":" 입력하여 라인 모드에서 "wq"로 저장하고 종료를 동시 수행

```
MINGW64:/f/suho
#include <stdio.h>

int main()
{
 printf("Hello world\n");
 return 0;
}
~
~
~
~
first.txt[+] [unix] (08:59 01/01/1970) 8,1 All
-- INSERT --
```



## vi 편집기 기초 실습(2) - 3



### • 입력한 내용 확인

- cat 명령어로 파일의 내용을 확인
- 입력한 상태 그대로 출력되었는지 확인
- vi 편집기의 사용이 쉽지 않으나 간단한 편집용으로 사용 가능
- 윈도우 메모장과 비교해서 크게 어려운 편은 아님
- 익숙하지 않을 뿐, 익숙해지면 나름 편집용으로 사용 가능

```
MINGW64:/f/suho

unangel@DESKTOP-I80QG85 MINGW64 /f/suho (master)
$ cat first.txt
#include <stdio.h>

int main()
{
 printf("Hello world\n");
 return 0;
}

unangel@DESKTOP-I80QG85 MINGW64 /f/suho (master)
$ |
```



Open Source Software

# Vim 편집기 환경설정



About..

컴퓨터소프트웨어공학과

김 원 일



# vi 편집기 환경설정 - 1



## • 편집기 환경 설정 - 1

- 명령 모드에서 다음의 명령어 차례로 실행
  - set nu : 줄 번호를 왼쪽에 출력
  - set ai : 자동 줄 맞춤 옵션 실행
  - set ts=4 : 탭 입력 시 공백이 4개가 입력되도록 설정
- 명령어 입력 시, 관련된 설정이 즉시 편집기에 반영

```
MINGW64:/f/suho
1 #include <stdio.h>
2 |
3 int main()
4 {
5 printf("Hello world\n");
6 return 0;
7 }
8
~
~
~
~
~
~
first.txt [unix] (18:45 25/03/2025) 2,0-1 All
:set nu
```



## vi 편집기 환경설정 - 2

### • 편집기 환경 설정 - 2

- 환경 설정한 상태에서 "q" 명령어로 편집기를 종료
- 셸에서 "vi first.txt" 명령으로 다시 편집기를 실행
- 설정했던 환경이 모두 초기화 된 것을 확인할 수 있음
- 편집기 실행 후 입력한 환경 설정은 종료 시 모두 해제가 기본 설정

```
MINGW64:/f/suho
#include <stdio.h>

int main()
{
 printf("Hello world\n");
 return 0;
}
~
~
~
~
~
~
first.txt [unix] (20:44 25/03/2025) 2,0-1 All
```



# vi 편집기 환경설정 - 3



## • 홈 디렉터리

- 리눅스에는 로그인한 사용자만 사용할 수 있는 디렉터리가 존재
- 사용자의 개인 설정과 개인 파일들을 저장하고 관리
- "./"과 "../" 처럼 홈 디렉터리를 가리키는 특정 경로가 존재
- 어느 경로에서나 "~/"는 항상 자신의 디렉터리를 가리킴
- Bash의 기본 홈 디렉터리 경로와 정보 확인

```
MINGW64:/c/Users/unangel
unangel@DESKTOP-I80QG85 MINGW64 /f/suho (master)
$ cd ~

unangel@DESKTOP-I80QG85 MINGW64 ~
$ pwd
/c/Users/unangel

unangel@DESKTOP-I80QG85 MINGW64 ~
$ ls
'3D Objects' /
AppData /
'Application Data' @
Contacts /
Cookies @
Desktop /
```



## vi 편집기 환경설정 - 4



### • 윈도우에서 홈 디렉터리

- 윈도우가 설치된 드라이브에 "Users" 디렉터리의 사용자 아이디
- Bash가 윈도우에 설치되어 해당 경로를 그대로 사용하도록 구성
- 리눅스 환경을 시뮬레이션 했기 때문에 홈 디렉터리도 동일하게 처리
- 작업 디렉터리로 복귀하기 위해 "cd -"를 입력
- 현재 위치 바로 이전의 디렉터리로 이동하는 명령

```
MINGW64:/f/suho

unange1@DESKTOP-I8OQG85 MINGW64 ~
$ pwd
/c/Users/unange1

unange1@DESKTOP-I8OQG85 MINGW64 ~
$ cd -
/f/suho

unange1@DESKTOP-I8OQG85 MINGW64 /f/suho (master)
$ |
```



## • vi 편집기 환경 설정 파일

- 홈 디렉터리에 “.vimrc” 파일을 생성하고, 편집 후 저장
- “vi ~/.vimrc” 명령어로 현재 위치에서 홈 디렉터리의 파일 생성
- “i” 또는 “a” 입력 후 이전에 설정했던 편집기 설정 값을 입력하고 저장
- 명령 모드에서 “wq”로 저장 후 편집기를 종료

[illegible]



# vi 편집기 환경설정 - 6



## • 편집기 설정 적용 확인

- 다시 first.txt 파일을 열었을 때 적용되었는지 확인
- 개인 설정으로 저장되어 있기 때문에 편집기 실행할 때마다 적용
- 만약 적용되지 않았다면 "source ~/.vimrc" 명령으로 즉시 적용
- source 명령어는 현재 셸에 스크립트를 즉시 적용하라는 명령

```
MINGW64:/f/suho
1 #include <stdio.h>
2
3 int main()
4 {
5 printf("Hello world\n");
6 return 0;
7 }
8
~
~
~
~
first.txt [unix] (20:44 25/03/2025) 2,0-1 All
"first.txt" [unix] 8L, 75B
```



# vi 편집기 환경설정 - 7



## • set 명령을 이용한 vi 옵션 설정

| 옵 션        | 약 자    | 디폴트 값    | 기 능                                                    |
|------------|--------|----------|--------------------------------------------------------|
| autoindent | ai     | noai     | 자동적으로 들여쓰기를 한다.                                        |
| Ignorecase | Ic     | noic     | 검색과 치환에서 대문자와 소문자를 구분하지 않는다.                           |
| magic      | magic  | magic    | 검색에서 메타 문자들을 사용할 수 있게 한다.                              |
| Number     | Nu     | Nonu     | 편집기의 각 라인에 라인 번호가 표시된다.                                |
| redraw     | redraw | noredraw | 각 문자를 항상 자기 위치에 나타낸다.<br>느린 속도를 가진 단말기에 설정하는 것이 좋다.    |
| mesg       | mesg   | mesg     | 문서 편집 동안 메시지가 화면에 출력되는 것을 허용한다.                        |
| Showmode   | smd    | nosmd    | 현재 편집 모드의 상태를 화면에 표시한다.                                |
| tabstop    | ts     | ts = 8   | Tab 키에 대응되는 공백의 개수를 지정한다.                              |
| terse      | terse  | noterse  | 오류 메시지를 간략하게 표현한다.                                     |
| window     | wi     | wi = 23  | vi 화면의 라인수를 지정한다.                                      |
| wrapmargin | wm     | wm = 0   | 여백의 크기를 지정한다.<br>wm = 10 인 경우, 한 라인에 70개의 문자까지 쓸 수 있다. |