

Open Source Software

# Vim 편집기 실습



About..

컴퓨터소프트웨어공학과

김 원 일



# vi 편집기 실습 - 1



- 파일 확인 후 다시 개방

- 저장된 파일 확인 후 편집기로 해당 파일을 다시 개방(open)
- 편집을 통해 vi 편집기 주요 명령어들을 간단하게 실습 진행

```
MINGW64:/f/suho
unangel@DESKTOP-I80QG85 MINGW64 /f/suho (master)
$ vi first.txt

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)
$ vi first.txt |
```



## vi 편집기 실습 - 2



### • 파일 확인 후 다시 개방

- 저장된 파일 확인 후 편집기로 해당 파일을 다시 개방(open)
- 편집을 통해 vi 편집기 주요 명령어들을 간단하게 실습 진행
- 기존에 작성한 파일을 이용하여 실습 진행
- 다른 파일을 이용해도 가능하나 파일에 맞게 조절이 필요

```
MINGW64:/f/suho
unangel@DESKTOP-I80QG85 MINGW64 /f/suho (master)
$ vi first.txt

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)
$ vi first.txt |
```



# vi 편집기 실습 - 3



## • 줄 단위 복사 - 1

- printf( )함수를 복사해서 2줄로 만들기
- 명령 모드로 복사하려는 줄(5라인)에서 yy → p 순으로 입력
- 명령 모드의 p는 커서가 있는 위치의 아래 줄에 복사
- 6라인에서 3 + p 로 복사하면 현재 복사된 내용이 3번 붙여 넣기 됨
- 명령 모드 명령어들 가운데 몇몇은 숫자와 조합하여 사용이 가능

```
MINGW64:/f/suho
1 #include <stdio.h>
2
3 int main()
4 {
5     printf( "Hello world\n" );
6     printf( "Hello world\n" );
7     return 0;
8 }
9
~
~
~
first.txt[+] [unix] (19:32 01/04/2025) 6,2-5 All
```



# vi 편집기 실습 - 4



## • 줄 단위 복사 - 2

- 여러 줄을 복사하려면 숫자 조합을 통해 복사할 수 있음
- 9라인에서 2 + yy 를 입력하면 9, 10라인의 내용이 복사됨
- 10라인에서 p를 입력하면 복사한 9, 10라인이 아래에 붙여 넣어짐
- 줄 단위 복사(yy) 및 붙여 넣기(p)는 모두 숫자와 조합 가능
- 이러한 명령어들은 숫자를 먼저 입력하고 명령어를 입력하는 순서

```
MINGW64:/f/suho
3 int main()
4 {
5     printf( "Hello world\n" );
6     printf( "Hello world\n" );
7     printf( "Hello world\n" );
8     printf( "Hello world\n" );
9     printf( "Hello world\n" );
10    return 0;
11    printf( "Hello world\n" );
12    return 0;
13 }
14
```

first.txt[+] [unix] (21:58 01/04/2025) 11,2-5 Bot



# vi 편집기 실습 - 5



## • 줄 단위 삭제

- 줄 삭제는 dd 명령어를 이용하며, 숫자와 조합이 가능
- 현재 편집 상태의 11라인에서 dd를 입력하면 printf( ) 라인 삭제
- 5라인으로 이동하여 6 + dd 입력 시 6줄이 삭제
- 삭제된 상태 그대로 p를 입력하면 삭제된 6줄이 아래 줄에 붙여 넣어짐
- dd 명령은 줄 단위 삭제이면서 동시에 복사인 잘라내기 명령어와 동일

```
MINGW64:/f/suho
1 #include <stdio.h>
2
3 int main()
4 {
5     return 0;
6     printf( "Hello world\n" );
7     printf( "Hello world\n" );
8     printf( "Hello world\n" );
9     printf( "Hello world\n" );
10    printf( "Hello world\n" );
11    return 0;
12 }
first.txt[+] [unix] (21:58 01/04/2025) 6,2-5 Top
6 more lines
```



# vi 편집기 실습 - 6



## • 명령어 실행 취소

- 명령 모드에서 u 입력은 명령 모드에서 직전 입력한 명령을 취소
- 대부분의 명령 모드에서의 명령을 취소할 수 있음
- 한번만 취소하는 것이 아니라 여러 번에 걸쳐 취소가 가능
  - vi 실행 후의 모든 명령을 취소할 수 있어 수정 전 상태까지 취소 가능
  - 파일의 편집을 종료하고 vi를 종료하면, 명령 실행 정보가 삭제되므로 주의
- 취소한 명령을 다시 실행하려면 Ctrl + r 로 재실행

```
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] (22:51 01/04/2025) 8,0-1 All
```



# vi 편집기 실습 - 7



- Visual - 1

- 윈도우와 동일하게 블록으로 텍스트를 지정할 수 있음
- 명령 모드에서 v키를 입력하는 위치부터 블록으로 지정
- 블록으로 지정할 위치까지 이동 후 복사, 잘라내기 등의 명령 수행
  - 복사와 잘라내는 키를 두 번씩 입력해야 하지만, 블록에서는 한번만 입력해도 동작함
- 블록을 취소하려면 ESC 키 입력

```
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] (22:51 01/04/2025) 5,22-25 All
-- VISUAL -- 12
```





## vi 편집기 실습 - 8



- Visual - 2

- v 입력 후 e 키를 누르면 단어 단위로 이동하면서 블록을 잡음
- 명령 모드에서 Ctrl + v 입력 시 열 단위 블록
  - 블록 모드에서 열 단위로 입력이나 복사 또는 삭제(잘라 내기) 명령어 수행이 가능

```
MINGW64:/f/suho
1 #include <stdio.h>
2
3 int main()
4 {
5     printf( "Hello world\n" );
6     printf( "Hello world\n" );
7     printf( "Hello world\n" );
8     printf( "Hello world\n" );
9     return 0;
10 }
11
~
~
first.txt[+] [unix] (22:51 01/04/2025) 8,15-18 All
-- VISUAL BLOCK -- 4x5
```



# vi 편집기 실습 - 9



## • 한 글자 변경과 삭제

- 명령 모드에서 오타 수정을 위해서는 입력 모드로 전환이 필요
- 한 글자만 변경할 때는 모드 변경 없이 r 입력 후 변경 키 입력으로 수정
- 한 글자 또는 커서 위치에서 한 글자 삭제는 x로 삭제
- 글자를 지운 후, 다시 입력하는 것으로 위치 이동 없이 삭제 가능
- 누른 상태를 유지하면 x를 뿔 때까지 계속 한 글자씩 삭제가 이루어짐

```
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] (22:51 01/04/2025) 5,11-14 All
1 change; before #2 1 second ago
```



# vi 편집기 실습 - 10



## • 문자열 찾기

- 파일 내에서 특정 문자열을 찾고자 하는 경우
  - 검색을 위해 5번 줄을 복사해서 몇 줄 붙여 넣기로 여러 줄을 만듦
- 명령 모드에서 / 입력 후 찾으려는 문자열 입력 후 엔터키 입력
  - 찾으려는 문자열을 입력하면 가장 먼저 일치하는 문자열을 블록으로 표시
- n은 일치하는 다음 문자열로, N은 일치하는 이전 문자열로 커서 이동

```
MINGW64:/f/suho
1 #include <stdio.h>
2
3 int main()
4 {
5     printf( "Hello world\n" );
6     printf( "Hello world\n" );
7     printf( "Hello world\n" );
8     printf( "Hello world\n" );
9     return 0;
10 }
11
~
~
first.txt[+] [unix] (22:51 01/04/2025) 5,11-14 All
/Hello
```



# vi 편집기 실습 - 11



- 커서 위치 이동

- 명령 모드에서 gg 입력 시 파일의 최상 단 시작 위치로 이동
- G 입력 시 파일의 가장 끝으로 커서 위치가 이동

- 파일을 다른 이름으로 저장

- 명령 모드에서 "sav 다른 이름으로 저장할 파일이름"으로 저장
- 먼저 편집 중이던 파일이 아니라 새로 저장한 파일 수정으로 변경

```
MINGW64:/f/suho
1 #include <stdio.h>
2
3 int main
4 {
5     printf( "Hello world\n" );
6     printf( "Hello world\n" );
7     printf( "Hello world\n" );
8     printf( "Hello world\n" );
9     return 0;
10 }
11
first.txt [unix] (21:36 02/04/2025) 3,10 All
:sav second.txt
```

## ❖ vi 편집기 실습 - 12



### • 윈도우 개발자들의 실수 - 1

- 익숙하지 않은 상태에서 Ctrl + z로 취소를 입력하는 경우
  - vi가 일시 중단된 상태에서 잠시 쉘로 빠져 나오는 상황이 발생
  - 편집 중이던 vi는 실행이 잠시 멈춘 상태일 뿐, 종료된 상태는 아님
- 보통 vi 실행 중에 쉘에서 작업을 잠시 수행해야 할 경우에 사용
- 쉘 수행이 완료되면 fg(background) 명령으로 vi 복귀 가능
  - vi 종료가 아니므로 명령 모드 실행 명령어들이 그대로 존재
  - u(명령 실행 취소)나 ctrl + r(명령 재실행)을 바로 사용할 수 있음

```
MINGW64:/f/suho

unangel@DESKTOP-I80QG85 MINGW64 /f/suho (master)
$ ls
first.txt  test/

unangel@DESKTOP-I80QG85 MINGW64 /f/suho (master)
$ vi first.txt

[1]+  Stopped                  vi first.txt

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



# vi 편집기 실습 - 13

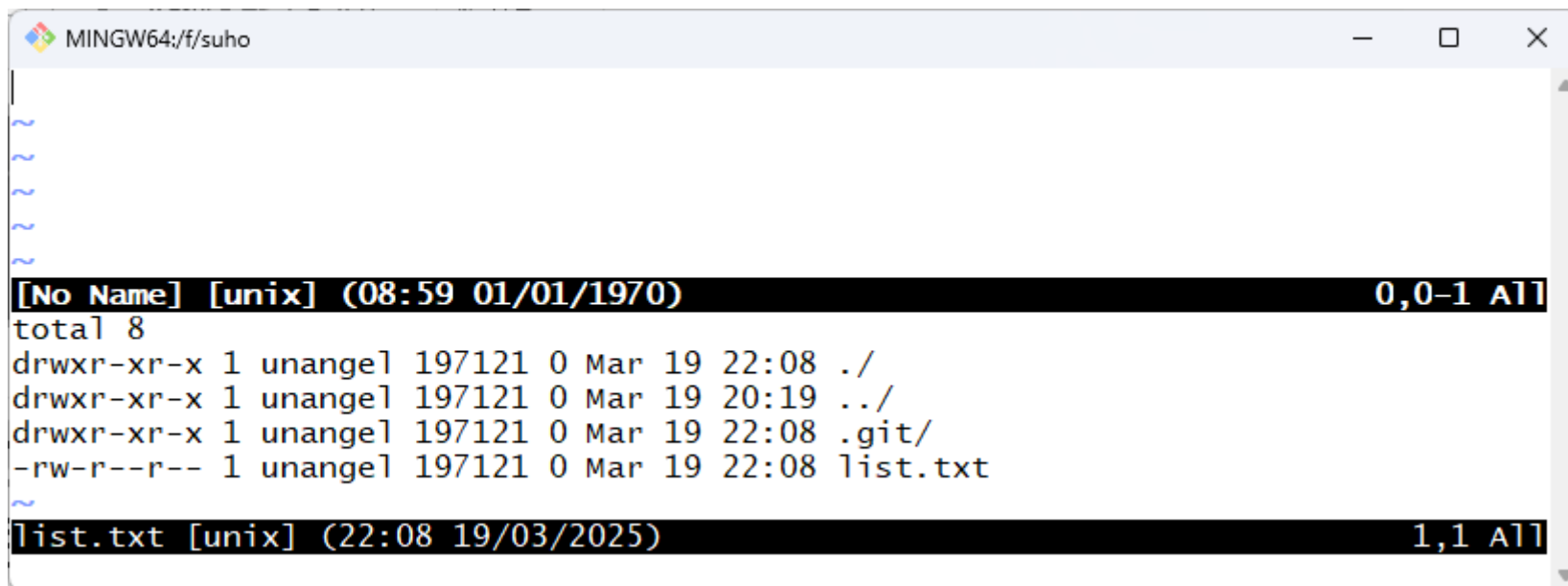


## • 윈도우 개발자들의 실수 - 2

- 문서 저장을 수시로 사용하는 아주 좋은 습관인 Ctrl + s 입력
- vi 자체가 멈춘 상태가 되고, 각종 모드도 사용이 불가능한 상태가 됨
- 스크롤 락(Scroll Lock)이 걸린 상태로 화면 잠금 상태
- Ctrl + q 입력으로 스크롤 락 상태를 해제할 수 있음
- git bash에서는 윈도우 개발자들을 위해 스크롤 락 키가 동작하지 않음
- 윈도우 프로그램 중 putty와 같은 프로그램 등에서도 발생할 수 있음
- 해제하는 방법을 알아두면 나중에 당황하지 않고 해제 가능

## ❖ vi 활용 - 1

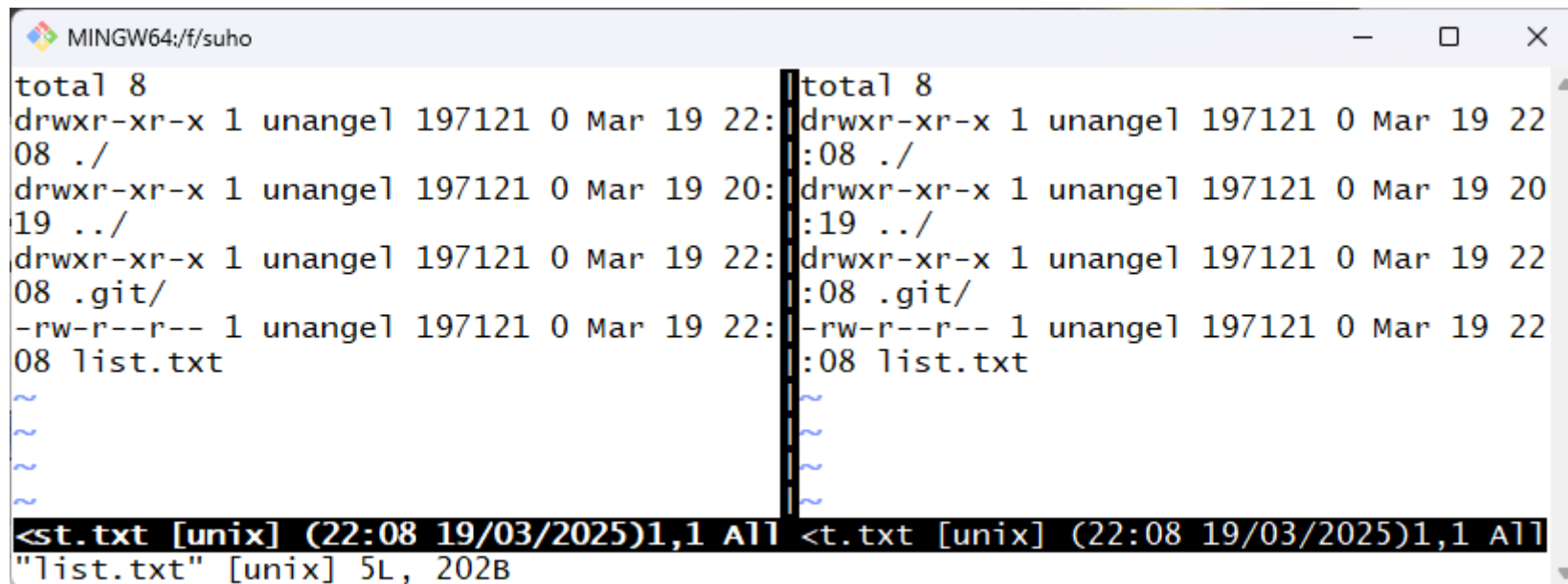
- 다수 개의 파일을 한꺼번에 열고 편집
  - vi 창을 가로 또는 세로로 분할하여 작업 가능
  - 분할된 창에서 여러 개의 파일을 동시 편집 가능
  - 각 창의 하단부의 검은색 블록에 파일에 대한 정보가 표시됨
  - 가로 분할 시에는 분리된 창에 파일이 없는 상태로 생성
- 창을 가로로 분할하기 : "**ctrl + w + n**"



```
MINGW64:/f/suho
~
~
~
~
~
[No Name] [unix] (08:59 01/01/1970) 0,0-1 All
total 8
drwxr-xr-x 1 unangel 197121 0 Mar 19 22:08 ./
drwxr-xr-x 1 unangel 197121 0 Mar 19 20:19 ../
drwxr-xr-x 1 unangel 197121 0 Mar 19 22:08 .git/
-rw-r--r-- 1 unangel 197121 0 Mar 19 22:08 list.txt
~
list.txt [unix] (22:08 19/03/2025) 1,1 All
```

## ❖ vi 활용 - 2

- 창을 세로로 분할하기 : "**ctrl + w + v**"
  - 세로 분할 시에는 현재 파일이 동시에 열림
  - 동일 파일은 어느 쪽에서 편집해도 동시에 변경됨
- 분할된 창 간 이동 : "**ctrl + w + w**"
  - 분할된 창이 2개일 때, 하나의 창을 닫으면 남은 창은 그대로 존재
  - 창을 닫을 때는 vi 종료 명령과 동일한 명령을 사용



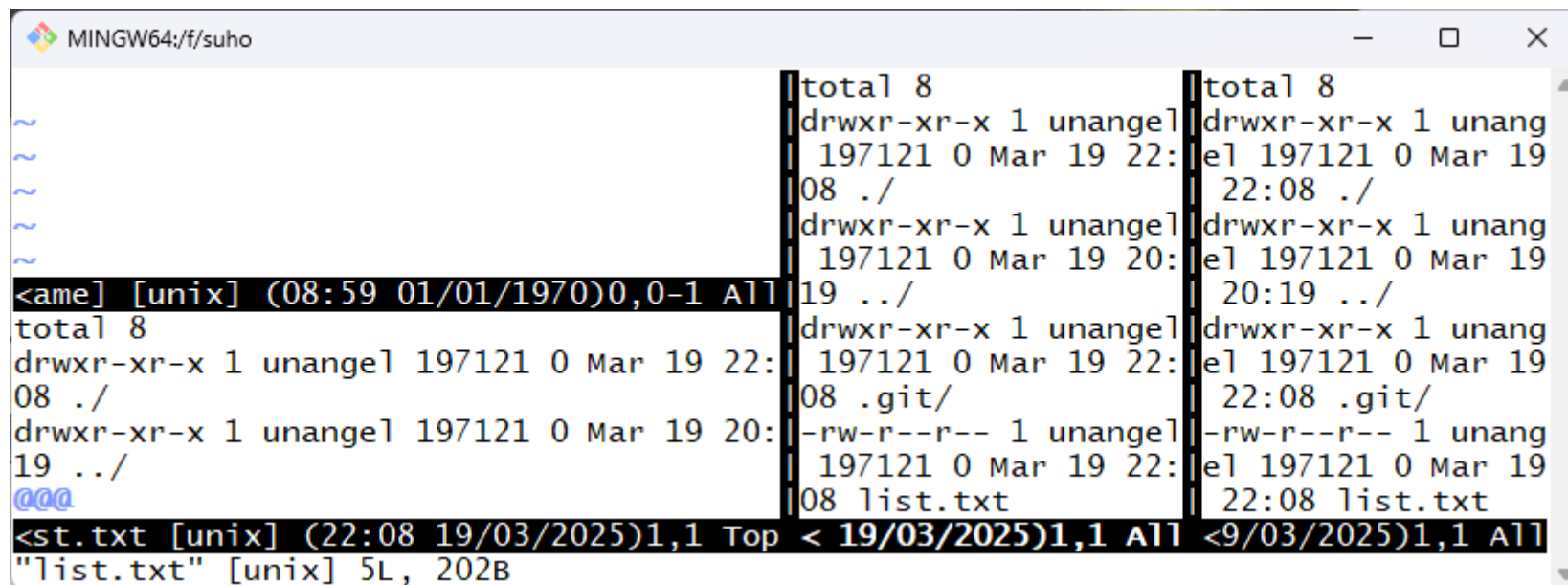
```
MINGW64:/f/suho
total 8
drwxr-xr-x 1 unangel 197121 0 Mar 19 22:08 ./
drwxr-xr-x 1 unangel 197121 0 Mar 19 20:19 ../
drwxr-xr-x 1 unangel 197121 0 Mar 19 22:08 .git/
-rw-r--r-- 1 unangel 197121 0 Mar 19 22:08 list.txt
~
~
~
~
<st.txt [unix] (22:08 19/03/2025)1,1 All
"list.txt" [unix] 5L, 202B

total 8
drwxr-xr-x 1 unangel 197121 0 Mar 19 22:08 ./
drwxr-xr-x 1 unangel 197121 0 Mar 19 20:19 ../
drwxr-xr-x 1 unangel 197121 0 Mar 19 22:08 .git/
-rw-r--r-- 1 unangel 197121 0 Mar 19 22:08 list.txt
~
~
~
~
<t.txt [unix] (22:08 19/03/2025)1,1 All
```



## ❖ vi 활용 - 3

- 창을 다중 분할하여 사용
  - 열린 창에서 새로운 파일을 열기 : **":e file-name"**
  - 터미널을 최대화하여 다수 개의 파일을 동시에 편집 가능
- 바로 이전 창으로 이동 : **"ctrl + w + p"**
  - 바로 이전 창이므로 현재, 이전 창으로만 이동이 이루어짐
  - 현재 편집 중인 창을 기준으로 이루어짐



The screenshot shows a MINGW64 terminal window with a split-screen view of two vi editors. The left editor is editing a file named 'list.txt' and the right editor is editing a file named 'list.txt'. Both editors show the same content: a directory listing with permissions, owner, group, size, date, and filename. The left editor's status line shows '<st.txt [unix] (22:08 19/03/2025)1,1 Top' and the right editor's status line shows '< 19/03/2025)1,1 All'. The left editor's command line shows '<st.txt [unix] (22:08 19/03/2025)1,1 Top' and the right editor's command line shows '< 19/03/2025)1,1 All'.

```
MINGW64:/f/suho
~
~
~
~
~
<ame] [unix] (08:59 01/01/1970)0,0-1 All
total 8
drwxr-xr-x 1 unangel 197121 0 Mar 19 22:
08 ./
drwxr-xr-x 1 unangel 197121 0 Mar 19 20:
19 ../
@@@
<st.txt [unix] (22:08 19/03/2025)1,1 Top
"list.txt" [unix] 5L, 202B

total 8
drwxr-xr-x 1 unangel 197121 0 Mar 19 22:
08 ./
drwxr-xr-x 1 unangel 197121 0 Mar 19 20:
19 ../
08 .git/
-rw-r--r-- 1 unangel 197121 0 Mar 19 22:
08 list.txt

total 8
drwxr-xr-x 1 unang
el 197121 0 Mar 19
22:08 ./
drwxr-xr-x 1 unang
el 197121 0 Mar 19
20:19 ../
drwxr-xr-x 1 unang
el 197121 0 Mar 19
22:08 .git/
-rw-r--r-- 1 unang
el 197121 0 Mar 19
22:08 list.txt
< 19/03/2025)1,1 All
<9/03/2025)1,1 All
```



## • 다중 창의 크기 조절

- 다수의 창을 열고 작업을 진행하는 경우 창 크기 조절이 가능
- 창 크기는 현재 커서가 있는 창을 기준으로 조절이 이루어짐
- 잘못된 키 조합이나 특정 시스템에서는 다른 동작이 발생할 수 있음
- 키 조합을 연속적으로 하지 않아도 순서대로 진행하면 동작함

명령 모드 제어	설명	단축키
-	다중 창의 크기를 동일하게 변경	Ctrl + w + =
:res 크기	입력된 크기 숫자로 세로 길이 설정	-
:res +크기	입력된 크기 숫자만큼 현재 창을 세로로 늘림	Ctrl + Shift + w, (Shift) +
:res -크기	입력된 크기 숫자만큼 현재 창을 세로로 줄임	Ctrl + Shift + w, (Shift) -
:vertical res +크기	입력된 크기 숫자만큼 현재 창을 가로로 늘림	Ctrl + Shift + w, (Shift) >
:vertical res -크기	입력된 크기 숫자만큼 현재 창을 가로로 줄임	Ctrl + Shift + w, (Shift) <