

제3장 C 프로그래밍 환경

학습 목표

- 문서 편집 : vi, geidt
- C 컴파일러 사용: gcc
- 컴파일 자동화: make
- 디버깅: gdb
- 통합개발환경: Eclipse
- 라이브러리 관리: ar
- 소스 관리: ctags
- 형상 관리: CVS, SVN, git





3.0 문서 편집



학습목표

- 유닉스에서 사용하는 편집기의 종류를 이해한다.
- 유닉스의 대표적 화면 편집기인 vi의 사용 방법을 익힌다.
- vi의 환경 설정 방법을 익힌다.

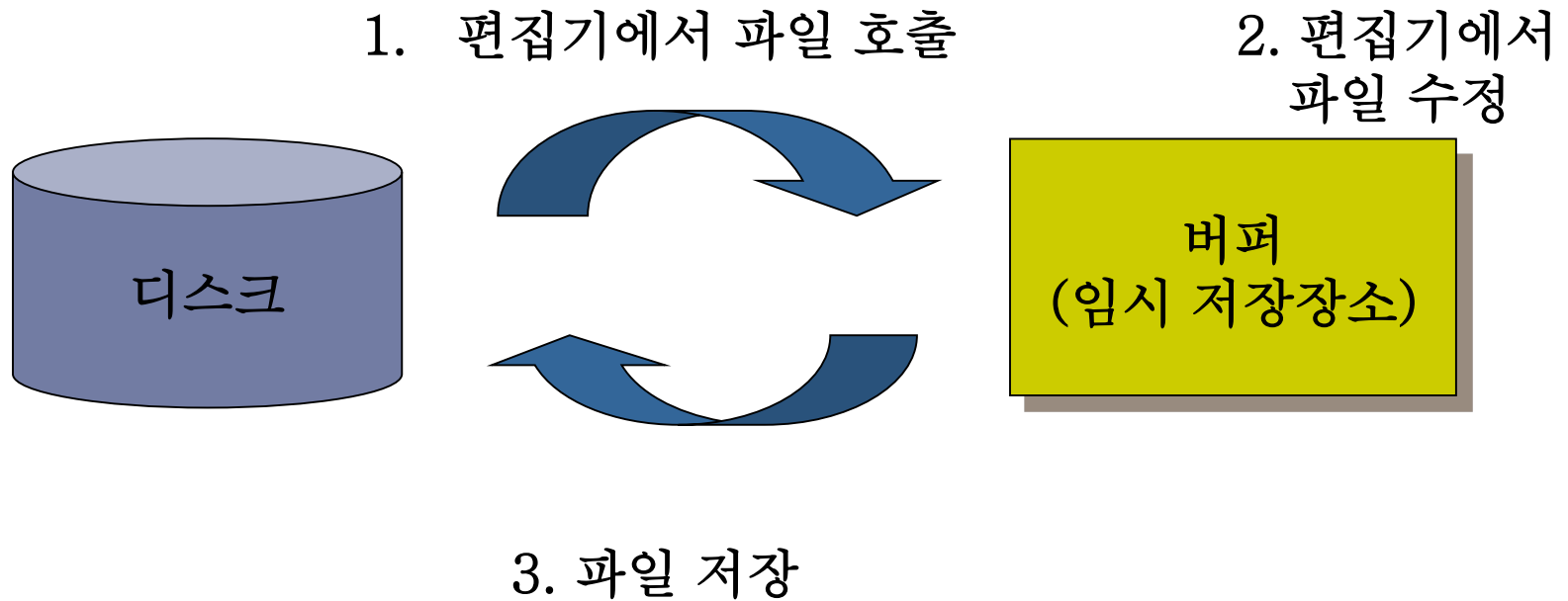


유닉스의 편집기

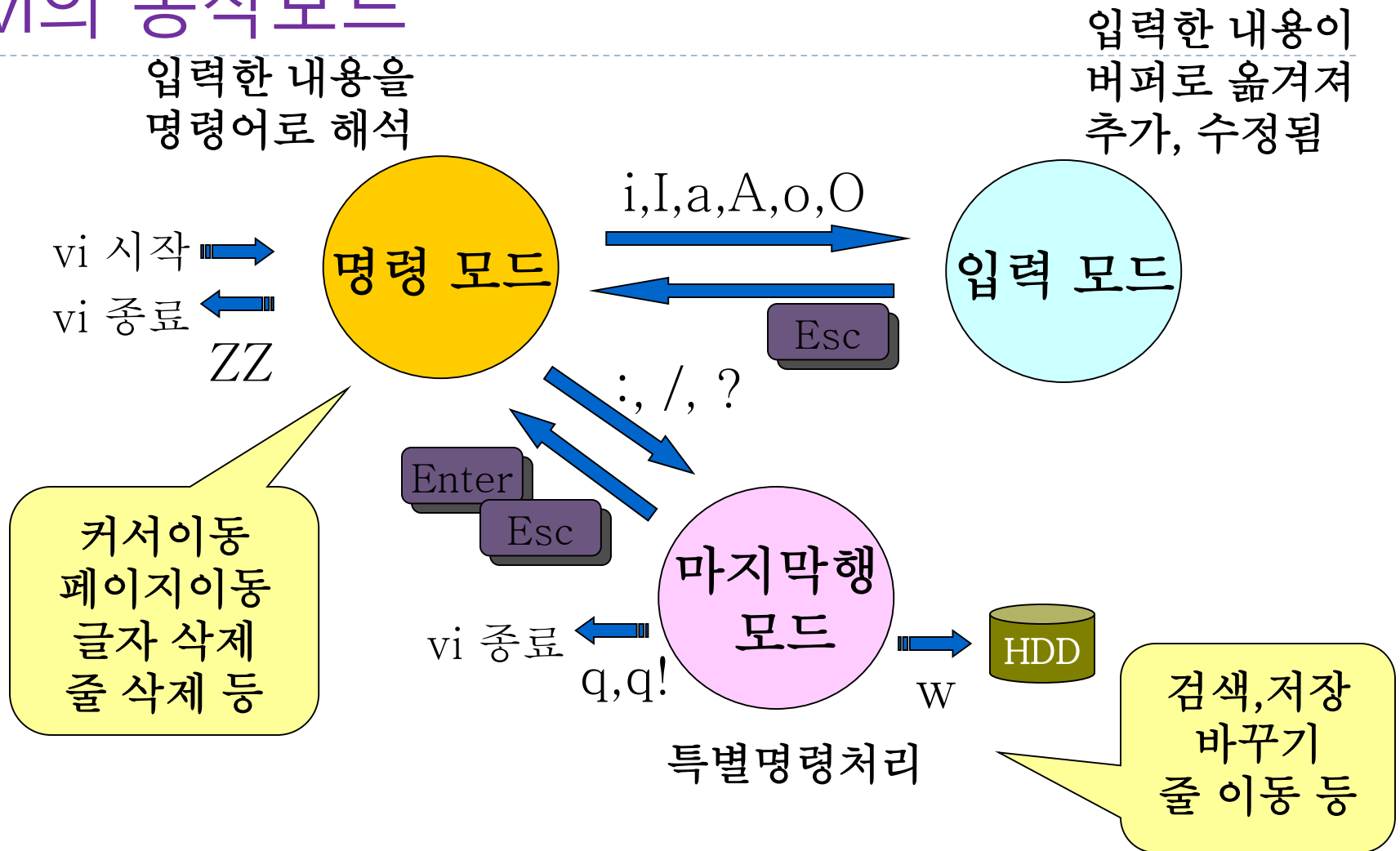
- 행 단위 편집기 (라인 편집기)
 - ed
 - ex
 - cat : 문서보기, 문서결합, 문서편집
- 화면 편집기
 - vi
 - 모든 UNIX에 있음.
 - emacs
 - 막강한 기능 제공.
 - 별도의 유틸리티, 설치해서 사용해야 함.



vi의 동작구조



vi의 동작모드



- vi에서는 대소문자를 별도의 명령으로 해석한다.

vi 시작하기

- vi

- 새로운 파일 시작
- 파일 저장할 때 이름 지정

- vi 파일이름

- 지정한 이름이 없으면 새로운 파일 생성
- 지정한 이름이 있으면 기존 파일 열기

- 기타

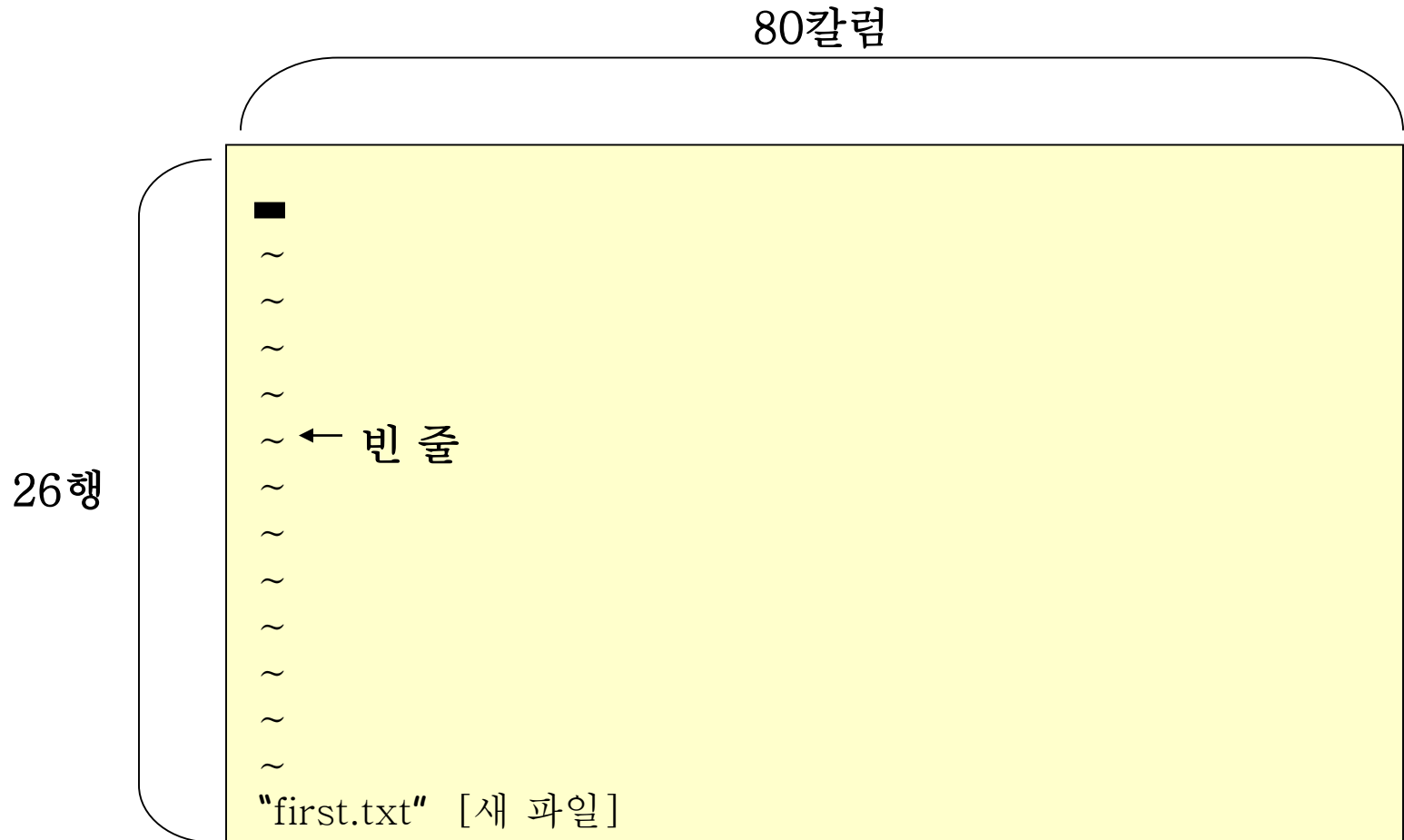
- vi -R filename 읽기 전용
- vi +line_no filename 파일을 열 때, 특정 라인으로 이동
- vi +/keyword filename 특정 키워드로 이동

```
telnet hanbit.co.kr
```

```
$ vi
```



vi 초기화면









- 화면크기에 따라 행과 칼럼수는 달라진다.

아기 입력 명령

모두 사용해보기

- 입력 명령(명령모드->입력모드)

명령키	수행 작업
	커서 앞에 삽입
	커서 뒤에 삽입
	현재 줄 첫 칸 앞에 텍스트 입력
	현재 줄 끝에 텍스트 입력
	현재 줄 다음에 삽입
	현재 줄 앞에 삽입



입력이 끝나면 “입력모드”에서 “명령모드”로 돌아와야 한다.



~~저장 및 종료 명령~~

○ 저장 명령

- 저장하거나 종료하려면 “명령모드”로 돌아와야 한다.

명령키	수행 작업
␣	현재의 파일명으로 파일 저장
␣	지정한 파일명으로 파일 저장

○ 종료 명령(저장후 종료 또는 그냥 종료)

명령키	수행 작업
␣	작업 내용을 저장하였으면 vi 종료 (파일을 수정한 상태에서 이 명령어를 사용하면 에러 출력)
␣	작업내용을 저장하지 않고 vi 종료
␣	작업 내용을 저장한 후 vi 종료
␣	작업 내용을 지정한 파일명으로 저장한 후 vi 종료
␣(shift-zz)␣	작업 내용을 저장한 후 vi 종료

입력 및 저장 실습 [1]

- 실습하기
 - 실습 디렉토리 구성

```
telnet hanbit.co.kr
```

```
$ cd  
$ mkdir Unix/ch4  
$ cd Unix/ch4  
$ vi hello.c
```

- 파일 생성 : hello.c

```
#include <stdio.h>  
main()  
{  
    printf("Hello World!\n");  
}
```

입력 및 저장 실습 [2]

- 실습하기 : vi hello.c

```
telnet hanbit.co.kr

#include <stdio.h>
main()
{
    printf("Hello World!\n");
}
```

```
telnet hanbit.co.kr

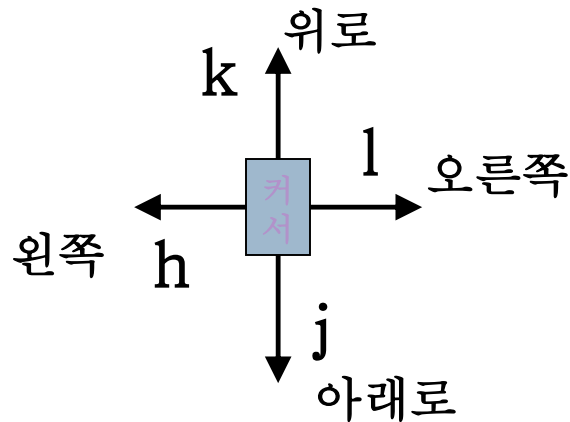
Hello#include <stdio.h>
main()
{
    printf("Hello World!\n");
}Add 한글
New line
```

hello2.c

- 1) i
- 2) Hello
- 3) Esc키
- 4) G (마지막 라인으로 이동)
- 5) o
- 6) New line
- 7) Esc키
- 8) :5 (5번째 라인으로 이동)
- 9) a
- 10) Add 한글
- 11) Esc키
- 12):w hello2.c

커서이동명령 [1]

- 화살표 키 이용
- h, j, k, l 키 이용



이동	명령어
한 줄 위	
한 줄 아래	
한 문자 오른쪽	
한 문자 왼쪽	
줄의 시작	또는
줄의 마지막	
이전 줄의 처음	
다음 줄의 처음	또는

커서이동명령 [2]

- 현재 화면에서 커서 이동

이동	명령키
키 화면 맨 위로	H
키 화면 중간으로	M
키 화면 맨 아래로	L
다음 단어의 첫문자로	w
이전 단어의 첫문자로	b
다음 단어의 끝 글자로	e

특수문자와 영어 달라지는
기준으로 한글자로 침

이전 문단의 시작으로 이동
다음 문단의 시작으로 이동

- 지정한 곳으로 이동

이동	명령키
줄 번호 n 위치로	:n 또는 nG
파일의 끝 줄로 이동	:\$ 또는 G
n줄 만큼 앞으로 이동	n+
n줄 만큼 뒤로 위로 이동	n-
현재 문장의 처음으로	(
다음 문장의 처음으로)
현재 문단의 처음으로	{
다음 문단의 처음으로	}
다음 함수의 처음으로]]
현재 함수의 처음으로	[[

숫자 입력후 +

아래로 이동

위로 이동



커서이동명령 [3]

- 커서이동 예제

```
1 #include <stdio.h>
2
3 main() {
4     char c;
5
6     printf("Hello, World\n");
7     printf("=====\n");
8     printf("select menu item\n");
9     printf("1. unix\n");
10    printf("2. linux\n");
11    printf("=====\n");
12 }
```

Diagram illustrating cursor movement commands and their effects on the text:

- H** (Home): Moves the cursor to the beginning of the line (line 1).
- M** (Move): Moves the cursor to the beginning of the line (line 6).
- ^** (Caret): Moves the cursor to the beginning of the line (line 8).
- L** (Left): Moves the cursor to the left (line 12).

Red circles highlight specific characters in the code, and arrows indicate the cursor's path:

- Line 1: **#** is circled, with an arrow from **H** pointing to it.
- Line 6: **p** is circled, with an arrow from **M** pointing to it.
- Line 7: **=** is circled, with an arrow from **k** pointing to it.
- Line 8: **s** is circled, with an arrow from **^** pointing to it.
- Line 8: **e** is circled, with an arrow from **W** pointing to it.
- Line 8: **n** is circled, with an arrow from **\$** pointing to it.
- Line 9: **1** is circled, with an arrow from **e** pointing to it.
- Line 10: **2** is circled, with an arrow from **j** pointing to it.
- Line 11: **=** is circled, with an arrow from **b** pointing to it.
- Line 12: **}** is circled, with an arrow from **L** pointing to it.

커서 이동 실습

- 실습하기
 - vi hello.c

```
telnet hanbit.co.kr
```

```
#include <stdio.h>
main()
{
    printf("Hello World!\n");
}
```

- 1) 커서를 1행으로 이동 : 1G 또는 :1
- 2) 1행의 두번째 단어로 이동 : w
- 3) 2행으로 이동 : j
- 4) 커서를 좌로 이동 : h
- 5) 마지막행으로 이동 : G 또는 :\$

화면이동

- 화면에 나타나지 않은 부분으로 화면 이동(scroll)

컨트롤

```
#include <stdio.h>
main()
{
    printf("Hello 1Wn");
    printf("Hello 2Wn");
```

telnet hanbit.co.kr

```
printf("Hello 3Wn");
printf("Hello 4Wn");
printf("Hello 5Wn");
printf("Hello 6Wn");
printf("Hello 7Wn");
```

```
printf ("Hello 8Wn");
}
```

이 동	명령키
반 화면 위로	^u
반 화면 아래로	^d
한 화면 위로	^b
한 화면 아래로	^f
한 줄만 위로 화면이 이동	^y
한 줄만 아래로	^e
파일내용 다시 출력	^

화면 이동 실습

- 실습하기
 - vi /etc/profile
 - 순서에 따라 화면 이동을 실습






```
telnet hanbit.co.kr
```

```
# ident
# The profile
trap "" 2 3
export LOGNAME PATH
if [ "$TERM = "" ]
.....
```

```
1) :set nu      줄 번호
2) ^u
3) ^d
4) ^f
5) ^b
6) ^y
7) ^e
8) ^|
9) :set nonu
```

~~내용~~ 삭제 및 취소

- 명령모드에서 동작

명령어	삭제 대상	수행 작업
	문자	커서 위치의 문자 삭제(예:3x)
	문자	커서 바로 앞 글자를 삭제
	줄의 일부	커서 위치부터 줄 끝까지 삭제
		방금 수행한 명령 취소
		해당 줄의 모든 편집 취소

한번 더 누르면 다시 편집 돌아옴

○ # : 숫자

- 모든 명령어에는 숫자를 붙일 수 있으며, 해당 숫자만큼 명령어 수행 반복

내용 삭제 및 취소

- 명령모드에서 동작

명령어	삭제 대상	수행 작업
	줄의 일부	커서가 있는 위치에서 행의 맨앞까지 (맨뒤까지) 삭제
	단어	커서 위치의 단어 삭제
	줄	커서 위치의 줄 삭제

커서가 단어 가운데에 있으면
앞글자는 안사라지고 커서부터
삭제가 됨



삭제 및 취소 실습

- 실습하기
 - vi hello2.c

telnet hanbit.co.kr

```
Hello#include <stdio.h>
main()
{
    printf("Hello World!\n");
}Add 한글
New line
```

telnet hanbit.co.kr







```
#include <stdio.h>
main()
{
    printf("Hello World!\n");
}
```



- 1) 1G
- 2) 5x
- 3) jjjj
- 4) l
- 5) D
- 6) j
- 7) dd
- 8) u
- 9) dd
- 10):w

내용 수정

- 명령모드에서 동작
 - 변경 후(즉, 입력 모드) 다시 명령모드로 자동으로 돌아옴
 - 수정 모드와 비슷한 역할

키	수정 대상	수행 작업
	문자	현재 커서위치의 한 문자 변경
	문자열	현재 커서부터 ESC 입력까지 변경
	단어	커서 위치부터 현재 단어의 끝까지 내용 변경
	줄	커서가 위치한 줄의 내용 변경
	문자열	현재 커서부터 내용 변경(예:5s)
	줄 일부	커서 위치에서 줄 끝까지 내용 변경

수정 실습

- 실습하기
 - vi hello2.c

telnet hanbit.co.kr

```
#include <stdio.h>
main()
{
    printf("Hello World!\n");
}
```

telnet hanbit.co.kr


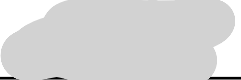



```
#include <stdio.h>
abcd[ ]
{
    write("Hello World!\n");
}
```



- 1) 1G
- 2) j
- 3) cw 단어를 지우고 입력모드로 들어감
- 4) abcd
- 5) Esc키
- 6) l (소문자 L)
- 7) r [1
- 8) r]
- 9) j j b
- 10) 6s
- 11) write
- 12) Esc키
- 13) :w

편집기능 - 복사, 잘라내기, 붙이기

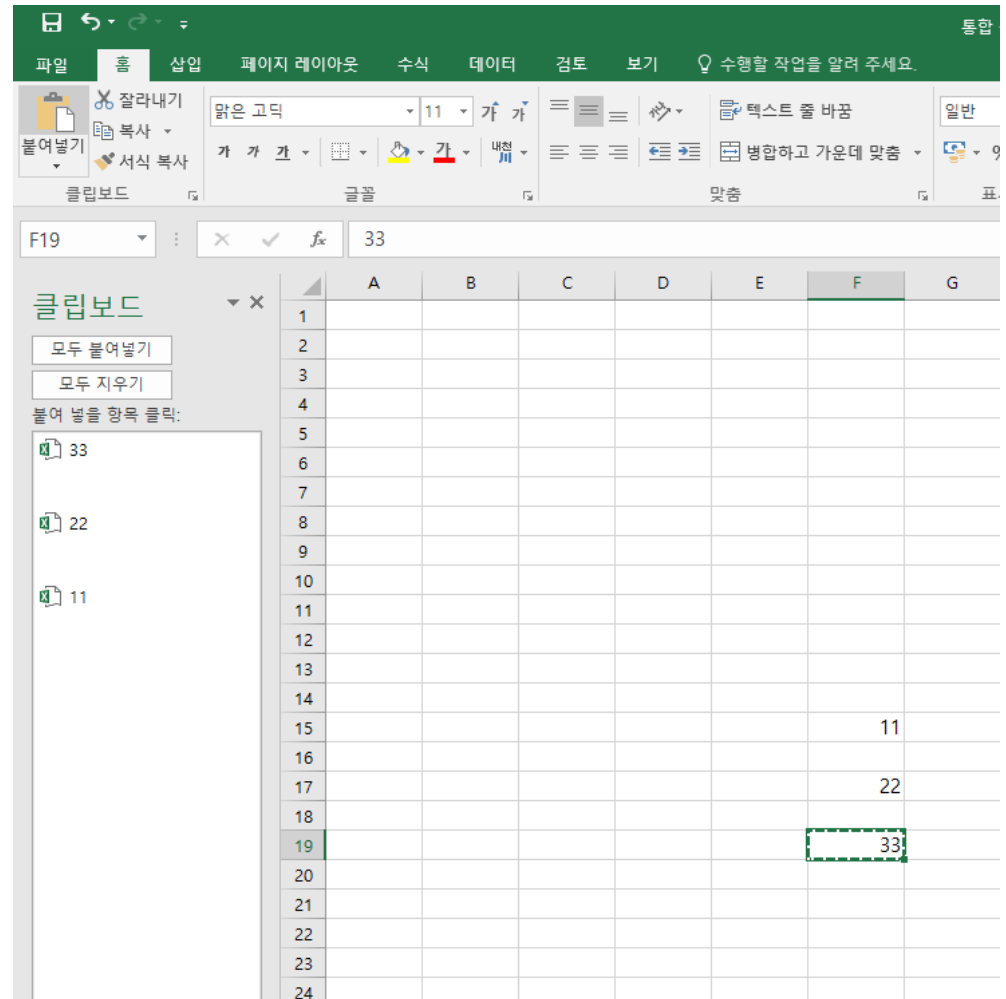
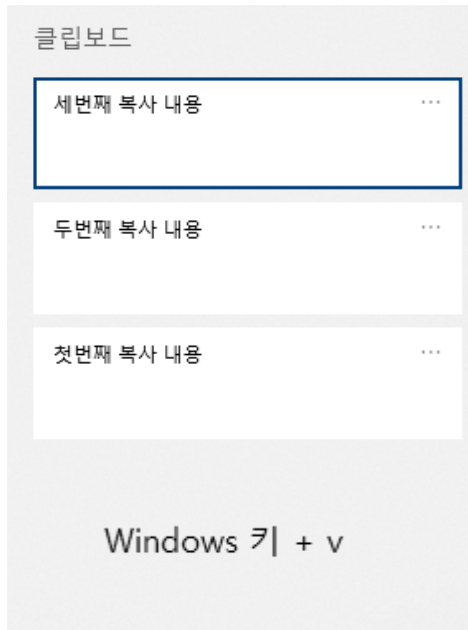
- 명령모드에서 동작

명령어	수행 작업
	현재 커서가 위치한 단어를 복사
	현재 행을 버퍼로 복사 (예:4yy)
	현재 행 다음에 버퍼 내용 삽입
	현재 행 위쪽에 버퍼 내용을 삽입
	현재 행을 잘라내기

- ❑ 행 삭제와 잘라내기는 동일한 동작이다.
 - ❑ Copy & Paste
 - ❑ Cut & Paste



버퍼의 사용



버퍼의 사용



- 버퍼

- vi는 작업 내용을 버퍼에 저장 – 실행 취소 가능
- 복사하기, 잘라내기 사용

- 버퍼 종류

- Unnamed buffer (이름 없는 버퍼)
- Named buffers (이름이 있는 버퍼) “a, “b, … “z
- Numbered buffers(번호가 있는 버퍼) “1, “2, …,“9

- 사용 예

-  -> 현재 행부터 아래로 3줄을 a버퍼에 저장
-  -> a버퍼의 내용을 붙이기



마지막행 모드에서 복사와 잘라내기

- 마지막 행 모드에서 사용
 - 범위 지정 방법 : 다음 슬라이드 참조

명령어	수행 작업
:#y	#으로 지정한 행을 복사(:10y -> 10행을 복사)
:<범위>y	범위로 지정한 행을 복사(예, :10,20y -> 10행~20행까지 복사)
:#d	#으로 지정한 행을 삭제(:10d -> 10행을 삭제)
:<범위>d	범위로 지정한 행을 삭제(예, :10,20d -> 10행~20행을 삭제)
:pu	현재 행 다음에 버퍼내용 붙이기 6번째 행에 붙여넣기 됨
:#pu	#으로 지정한 행 다음에 버퍼내용 붙이기(예, :5pu)

- 범위 지정 없이 명령어를 사용하면 보통 현재 줄 또는 커서가 있는 위치에서 명령이 실행됩니다.
- 범위를 지정하면 지정한 줄들 사이에서 명령어가 실행되어 더 넓은 범위에 작업을 적용할 수 있습니다.

범위지정 방법

차이 알아보기

- 편집하는 범위를 지정하는 방법
- 마지막행 모드에서 사용
- : 범위 편집명령 형태로 결합되어 사용

범위	의 미
1,\$	첫 줄에서 마지막 줄까지(파일내의 모든 줄)
%	첫 줄에서 마지막 줄까지(파일내의 모든 줄)
1,.	첫 줄에서 현재 줄까지
.,\$	현재 줄에서 마지막 줄까지
.-2	현재 줄에서 앞쪽으로 2번째 줄 (주의: , 없음)
10,20	10번째 줄에서 20번째 줄까지

버퍼를 이용한 복사 실습

- 실습하기

- vi hello2.c -> hello.c로 복사

hello2.c

telnet hanbit.co.kr

```
#include <stdio.h>
abcd[ ]
{
    write("Hello World!\n");
}
```



telnet hanbit.co.kr

```
#include <stdio.h>
abcd[ ]
{
    write("Hello World!\n");
    write("Hello World!\n");
}
```

hello.c

telnet hanbit.co.kr





```
#include <stdio.h>
abcd[ ]
main()
{
    printf("Hello World!\n");
}
```



- 1) :4 a라는 버퍼에 현재 줄 복사 됨
- 2) yy 현재 줄 복사
- 3) p 아래쪽에 붙여넣음
- 4) kkk
- 5) dd 삭제
- 6) p 붙여넣음
- 7) "ayy
- 8) :w
- 9) :e hello.c
- 10) "ap a버퍼에 바로 붙여 넣음

검색 기능

- 마지막행 모드에서 사용

명령어	수행 작업
	현재 위치부터 파일의 아래 방향으로 문자열 탐색
	현재 위치부터 파일의 위 방향으로 문자열 탐색
	다음 문자열 탐색 (“/문자열”의 경우 아래로 이동, “?문자열”의 경우 위로이동)
	역방향으로 문자열 탐색 (n과 반대 방향으로 이동)

문자열 탐색 실습

- 실습하기
 - vi hello2.c

```
telnet hanbit.co.kr
```

```
#include <stdio.h>
abcd[ ]
{
    write("Hello World!\n");
    write("Hello World!\n");
}
```

1) /abcd
2) n
3) ?write
4) n
5) N



바꾸기 기능 (find and replace 기능)

- 마지막행 모드에서 사용

명령어	수행 작업
: [문자열1]	커서가 위치한 줄에서만 문자열1을 문자열2로 바꿈
: [문자열1] [문자열2]	<범위>안의 모든 줄에 대해서 각 줄의 첫번째 문자열1을 찾아 문자열2로 바꿈
[문자열1] [문자열2]	<범위>안의 모든 줄에 대해서 모든 문자열1을 문자열2로 바꿈
[문자열1] [문자열2] [문자열3]	<범위>안의 모든 줄에 대해서 각 문자열1을 문자열2로 치환할 때 수정할지 안 할지를 묻는다

범위를 이용한 바꾸기 실습

- 실습하기

범위 설정하면 계속 유지 됨

- vi hello2.c

```
telnet hanbit.co.kr
```

```
#include <stdio.h>
abcd[ ]
{
    write("Hello World!\n");
    write("Hello World!\n");
}
```

첫번째, 두번째 줄 포함 t를 ct로 바꿈

```
telnet hanbit.co.kr
```

```
#include <ctbio.h>
main[ ]
{
    write("Hi Worlb!\n");
    write("Hi Worlb!\n");
}
```

%없으면 다 바뀜

- 1) j
- 2) :s/abcd/main/
- 3) :%s/d/b/g
- 4) :1,2 s/st/ct/g
- 5) :1,\$ s/Hello/Hi/
g
1부터 끝까지 Hello를 Hi로 바꿈
- 6) :w

기타 기능 [1]

- 파일 읽어오기 / 여러 파일 편집



명령어	수행
:파일명	안적으면 지금 열려있는 파일이 삽입 지정한 파일을 현재 커서 위치에 삽입
:파일명	파일명 안적으면 오류 발생 현재 파일 대신 지정한 파일을 읽음
:	vi 시작시 여러 파일을 지정하였을 경우 다음 파일로 이동

○ vi에서 셸 명령 실행

명령어	수행 작업
:명령	vi를 중단하고 지정한 명령 수행 (vi로 돌아올 때 :↵)
:명령	실행 결과를 편집 파일에 끼워넣기 한다
:	vi를 잠시 빠져나가서 셸을 수행 (vi로 돌아올때 :exit)

기타 기능 [1]

- 마크 기능

명령어	내용
	커서를 특정 행에 위치시킨 후에 a부터 z 중 하나 문자를 마크 식별자로 사용한다.
	'`'와 함께 식별자를 입력하면 해당 표시(Mark)가 있는 행으로 커서가 이동한다. (~ 키 아래에 위치한 키)



셸 명령 실행 실습

- 실습하기
 - vi hello2.c

```
telnet hanbit.co.kr
```

```
#include <ctbio.h>
main[ ]
{
    write("Hi Worlb!\Wn");
    write("Hi Worlb!\Wn");
}
```

```
1) :!ls -l
2) Enter ↵
3) :sh
4) ls -l
5) exit
```



기타 기능 [2]

- 알아두면 유용한 명령키들

명령어	수 행
● 파일명	파일 이름을 지정한 이름으로 변경
● %.old	현재 파일을 .old 이름으로 저장해 둘 때
^g	기본적인 파일정보 출력(파일명, 라인수 등)
J	현재 줄과 다음 줄 연결
.	바로 이전에 수행한 명령 재 실행
>>	Tab 크기만큼 오른쪽으로 이동 (들여쓰기 기능) 예) 3>> 커서가 포함된 라인을 포함하여 3라인을 탭만큼 오른쪽으로 이동
<<	Tab 크기만큼 왼쪽으로 이동
~	현재 커서 위치의 한 문자를 소문자 혹은 대문자로 전환

기타 명령어 실습

- 실습하기
 - vi hello.c

telnet hanbit.co.kr

```
#include <stdio.h>
abcd[ ]
main()
{
    printf("Hello World!\n");
}
```

telnet hanbit.co.kr

```
#include <stdio.h>
ABCD[ ]main()
{
    printf("Hello World!\n");
}
```



1) :2

2) J

3) ~ 소문자를 대문자로 바꿈

4) .

5) .



6) .

7) :w

vi 환경 설정

나머지는 알 필요 x

- vi의 환경을 설정하는 특수명령과 변수들

명령어	수행 작업
:set 	파일 내용의 각 줄에 줄 번호 표시 (보이기만 할 뿐 저장은 되지 않는다.)
:set nonu	줄 번호 취소
:set 	눈에 보이지 않는 특수문자표시(tab:^\I, eol:\$ 등)
:set nolist	특수문자보기 기능 취소
:set showmode	현재 모드 표시
:set noshowmode	현재 모드 표시기능 취소
:set	set으로 설정한 모든 vi변수 출력
:set all	모든 vi 변수와 현재 값 출력
:set ts=4	Tab 크기를 4로 조정

기타 명령

명령어	수행 작업
:g/pattern	파일에서 패턴이 포함된 가장 마지막 위치를 찾아 이동
:g/pattern/p	파일에서 패턴이 포함된 모든 라인을 출력한다.
:g/pattern/d	파일에서 패턴이 포함된 모든 라인을 삭제한다.



실습 최종 파일

- hello.c

```
telnet hanbit.co.kr
```

```
#include <stdio.h>
ABCD[ ]main()
{
    printf("Hello World!\n");
}
```

```
kk, H
: % s/s/c
:%s/d/b/g
:%s/printf/write/g
:2 1+ j
6x
$, III
r[ r]
yy
p P
```

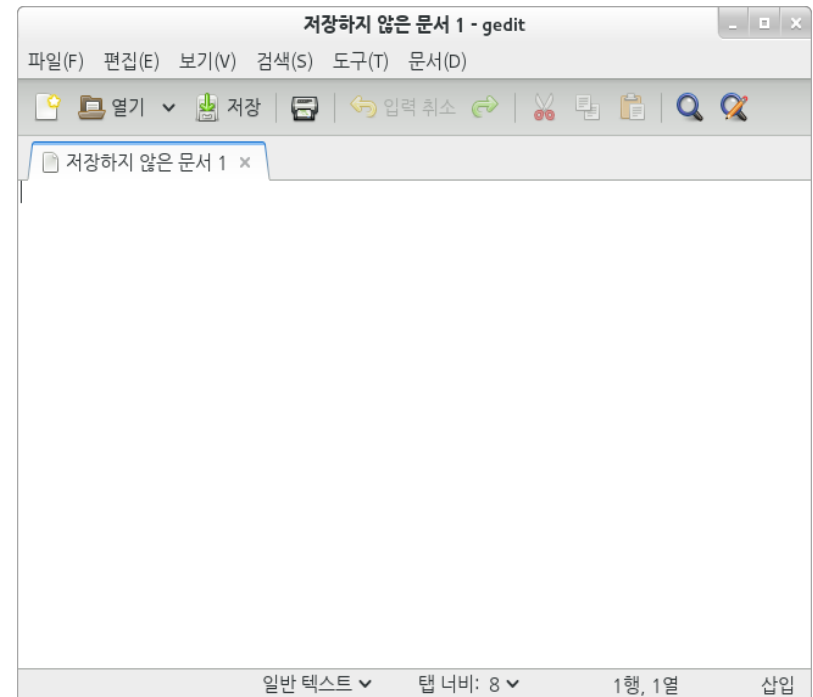
- hello2.c

```
telnet hanbit.co.kr
```

```
#inclube <ctbio.h>
main[ ]
{
    write("Hi Worlb!\n");
    write("Hi Worlb!\n");
}
```

gedit 문서편집기

- GNU의 대표적인 GUI 텍스트 편집기
- GNOME 환경의 기본 편집기
 - 텍스트, 프로그램 코드, 마크업 언어 편집에 적합
 - 깔끔하고 단순한 GUI
- gedit 실행 방법
 - 메인 메뉴
 - [프로그램] -> [보조 프로그램] -> [지에디트] 선택
 - 터미널
 - `$ gedit [파일이름] &`
 - 파일 관리자:
 - 텍스트 파일 클릭하면 자동실행



gedit 메뉴

- 파일
 - 새로 만들기, 열기, 저장, 되돌리기, 인쇄
- 편집
 - 입력 취소, 다시 실행, 잘라내기, 복사, 붙여넣기, 삭제
- 보기
 - 도구모음, 상태표시줄, 전체화면, **강조 모드**
- 검색
 - 찾기, 바꾸기, 줄로 이동
- 도구
 - 맞춤법 검사, 오타가 있는 단어 강조, 언어 설정, 문서 통계
- 문서
 - 모두 저장, 모두 닫기, 새 탭 그룹, 이전 문서

