DATABASE LAB Computer Sci.& Eng. Hanyang University

Vi Editor

한양대학교 컴퓨터공학과

Dept. of Computer Science & Engineering

Hanyang University

vi 시작하기

- vi를 시작하려면, vi 명령어 입력
  - $ vi [file…]
- file은 사용자가 편집하고자 하는 파일의 이름
- file명을 지정하지 않고 시작할 수 도 있음

명령모드와 입력모드

- vi로 작업을 하면 모든 데이터는 **편집버퍼(Editing Buffer)**에 유지됨

- **입력모드(Input mode)**
  - 입력하는 모든 것이 편집버퍼에 입력됨

- **명령모드(Command mode)**
  - 입력하는 모든 것이 명령어로 해석됨
  - 입력모드에서 명령모드로 전환하는 방법 : **ESC키**

읽기 전용으로 **vi**사용하기: **-R** 옵션, **view**

- 중요한 파일을 수정하지는 않으면서 내용만을 보고 싶을 때 사용하는 두 가지 방법
  - **-R**(Read-only, 읽기전용) 옵션
  - **view** 명령어

vi 마치기

- 종료 명령을 입력할 수 있는 명령모드에 있어야 함
- **:wq** : 작업 내용을 저장하고 종료
- **:q!** : 작업 내용을 저장하지 않고 종료
- 종료시 데이터를 저장했는지 점검

vi 명령어를 배우는 전략

- 편집버퍼에 데이터를 넣을 때 반드시 다음 단계를 따라야 함
  1. 데이터를 쓰고 싶은 곳으로 커서를 옮겨야 한다
  2. 입력모드로 바꾸기 위한 명령을 사용한다
  3. 데이터를 입력한다
  4. 명령모드로 바꾸기 위해 **ESC**를 누른다
- 편집버퍼에 데이터가 있으면 어떤 일을 하기 위하여 준비되어야 할 다양한 명령들
  1. 커서를 움직이는 명령
  2. 입력모드로 전환하는 명령
  3. 변화를 주기 위한 명령

커서 이동하기

h	커서를 한 칸 왼쪽으로 이동
j	커서를 한 칸 아래쪽으로 이동
k	커서를 한 칸 위쪽으로 이동
l	커서를 한 칸 오른쪽으로 이동

LEFT	커서를 왼쪽으로 한 칸 이동
DOWN	커서를 아래쪽으로 한 칸 이동
UP	커서를 위쪽으로 한 칸 이동
RIGHT	커서를 오른쪽으로 한 칸 이동
BACKSPACE	커서를 왼쪽으로 한 칸 이동
SPACE	커서를 오른쪽으로 한 칸 이동

커서 이동하기 (cont'd)

-	커서를 이전 줄의 처음으로 이동
+	커서를 다음 줄의 처음으로 이동
RETURN	커서를 다음 줄의 처음으로 이동
0	커서를 현재 줄의 맨 앞으로 이동
$	커서를 현재 줄의 끝으로 이동
^	커서를 현재 줄의 첫 글자로 이동(탭이나 공백이 아닌)
w	커서를 다음단어의 첫 글자로 이동
e	커서를 다음단어의 끝 글자로 이동
b	커서를 이전단어의 첫 글자로 이동

커서 이동하기 (cont'd)

W	w와 동일, 문장 부호 무시
E	e와 동일, 문장 부호 무시
B	b와 동일, 문장 부호 무시
)	다음문장의 처음으로 이동
(	이전문장의 처음으로 이동
}	다음문단의 처음으로 이동
{	이전문단의 처음으로 이동
H	커서를 화면 맨 위로 이동
M	커서를 중간으로 이동
L	커서를 화면 맨 아래로 이동

편집버퍼에서 이동하기

- vi는 화면에 알맞을 만큼만 편집버퍼의 내용을 보여줌
- 화면에 나타나지 않은 편집버퍼의 다른 부분을 보기 위해서 사용할 수 있는 명령 :

^F	한 화면 아래로 이동
^B	한 화면 위로 이동
^D	반 화면 아래로 이동
^U	반 화면 위로 이동

각행에 번호 부여

- vi는 편집 버퍼에 있는 각 줄에 번호를 부여하여 관리함
- **:set number** 명령 사용

-***n*G**	줄 번호 *n*으로 이동하기
1G	편집버퍼의 첫 줄로 이동하기
G	편집버퍼의 마지막 줄로 이동하기

데이터를 편접 버퍼에 입력

- 새로운 데이터 입력에 사용되는 명령 :

i	입력모드로 전환, 커서 위치 앞에서 삽입
a	입력모드로 전환, 커서 위치 뒤에서 삽입
I	입력모드로 전환, 현재 줄의 앞에서 삽입
A	입력모드로 전환, 현재 줄의 뒤에서 삽입
o	입력모드로 전환, 현재 줄의 아래에 전개
O	입력모드로 전환, 현재 줄의 위에 전개

편집버퍼에서 내용 수정하기

- vi의 고치기 명령 :

r	단지 한 글자만 변경(입력모드로 바뀌지 않음)
R	입력하는 대로 겹쳐 써서 변경
s	삽입에 의한 한 단어의 변경
C	커서의 위치로부터 줄 끝까지 삽입에 의한 변경
cc	전체 줄을 삽입에 의해 변경
S	전체 줄을 삽입에 의해 변경
c*move*	커서부터 *move*까지 삽입에 의해 변경

삭제 및 복사

x 커서위치의 1문자 삭제
X 커서위치의 왼쪽 1문자 삭제
dd 커서가 있는 행을 삭제
ndd 커서가 있는 곳부터 n 행 삭제
d$ 커서의 위치에서 행 끝까지 삭제
d^ 맨 앞에서 커서위치의 왼쪽까지 삭제
yy 커서가 있는 행을 복사
nyy 커서가 있는 행부터 n 행을 복사
p 명령어로 삭제/복사된 텍스트를 현재 문자(행)의 뒤에 붙여넣는다
Y 현재 문자(행)의 앞에 붙여 넣는다

vi 환경설정

:set number	행 번호 보이게
:set nonumber	행 번호 안보이게
:set autoindent	들여쓰기 설정
:set noautoindent	들여쓰기 제거
:set list	문단,조판부호 보기
:set nolist	문단,조판부호 안보이게
:set window=30	한 화면당 행의 갯 수 30개로 지정
:set ignorecase	검색 시 대소문자 구별 제거
:set noignorecase	검색 시 대소문자 구별
:set all	현재 설정된 vi 모든 설정 값 보기

C 예제프로그램 작성 및 실행

$vi test.c

```
#include <stdio.h>

int main(int argc, char *argv[])
{
        if(argc != 2)
        {
                printf("error\n");
                exit(1);
        }
        printf("%s\n", argv[1]);

        return 0;
}
```

```
$ gcc –o test test.c
$ ./test system
```

여러 파일 컴파일 하기 (1/2)

```
$ cp test.c foo.c

$ vi bar.c
        #include<stdio.h>
        int bar(){
                printf("%d\n", 1234);
        }
```

여러 파일 컴파일 하기 (2/2)

```
$ gcc –c foo.c
$ gcc –c bar.c
$ gcc –o baz foo.o bar.o
```

위의 세명령을 하나로 합치면 :

```
$ gcc –o baz foo.c bar.c
```

Makefile 만들기

Makefile을 만들면 적은 단계를 거쳐 파일을 생성가능하게 한다
- Makefile 혹은 makefile로 파일을 작성
- 실행은 make 명령으로 실행

```
$ vi Makefile

run : foo.o bar.o
        gcc –o run foo.o bar.o

foo.o : foo.c
        gcc –c foo.c

bar.o : bar.c
        gcc –c bar.c
```

Makefile 만들기 : 확장된 Makefile (1/3)

```
$ vi main.c
        #include<stdio.h>
        int main(void){
                printf("==== main === \n");
                edit();
                return 0;
        }

$ vi edit.c
        #include<stdio.h>
        int edit(){
                printf("==== edit ==== \n");
        }
```

Makefile 만들기 : 확장된 Makefile (2/3)

```
$ vi main2.c
        #include<stdio.h>
        int main(void){
                printf("==== main2 === \n");
                read();
                return 0;
        }

$ vi read.c
        #include<stdio.h>
        int read(){
                printf("==== read ==== \n");
        }
```

Makefile 만들기 : 확장된 Makefile (3/3)

```
$ vi Makefile

install : all
        mv edimh /home/자신학번/expanded_makefile/test
        mv readimh /home/ 자신학번/expanded_makefile/test
all : edimh readimh
readimh : main2.o read.o
        gcc -o readimh main2.o read.o
edimh : main.o edit.o
        gcc -o edimh main.o edit.o
main.o : main.c
        gcc -c main.c
main2.o : main2.c
        gcc -c main2.c
edit.o : edit.c
        gcc -c edit.c
read.o : read.c
        gcc -c read.c
```

Makefile 만들기 : 매크로 사용

```
OBJECTS = main.o read.o write.o

test : $(OBJECTS)
        gcc -o test main.o read.o write.o

main.o : io.h main.c
        gcc -c main.c

read.o : io.h read.c
        gcc -c read.c

write.o : io.h write.c
        gcc -c write.c
```

Makefile 만들기 : 레이블 사용

```
OBJECTS = main.o read.o write.o
test : $(OBJECTS)
        gcc -o test main.o read.o write.o
main.o : io.h main.c
        gcc -c main.c
read.o : io.h read.c
        gcc -c read.c
write.o : io.h write.c
        gcc -c write.c
clean :
        rm $(OBJECTS)
```

```
% make clean
rm main.o read.o write.o
```

실습

- Makefile을 작성하여 컴파일 하시요
- write.c를 갱신한 후 컴파일 하시요
- io.h를 갱신한후 컴파일 하시오

io.h

```
#define a “io.h”
```

main.c

```
#include <stdio.h>
#include "io.h"
int main(void)
{
        printf("---main.c+%s---₩n", a);
        readfile();
        writefile();
        return 0;
}
```

read.c

```
#include <stdio.h>
#include "io.h“
void readfile(void)
{
        printf("---read.c+%s---₩n", a);
}
```

write.c

```
#include <stdio.h>
#include "io.h“
void writefile(void)
{
        printf("---write.c+%s---₩n", a);
}
```

Q & A

- Thank you :)