

임베디드시스템설계실험 2주차 과제

1. Markdown

1.1 마크다운이란 ?

- 마크다운은 일반 텍스트 기반의 경량 마크업언어 이다.
일반 텍스트로 서식이 있는 문서를 작성하는 데 사용되며, 일반 마크업 언어에 비해 문법이 쉽고 간단한 것이 특징이다. 특히 HTML과 리치 텍스트(RTF) 등 서식 문서로 쉽게 변환 되기 때문에 응용 소프트웨어와 함께 배포되는 README 파일이나 온라인 게시물에 많이 사용된다.
확장 자는 .md 파일로 생성된다.

1.2 마크다운을 쓰는 이유 ?

- 쉽기 때문이다.
 - HTML을 모르는 사람이라더라도, 누구나 문법을 5분안에 배우고 쓸 수 있다.
- 글만 쓰면 알아서 깔끔하게 예쁘게 표현가능하다.
- 확장성이 좋다.
 - 문법이 간단하고 HTML로 변환이 가능해서 마크다운을 지원하는 에디터에 복사하면 바로 수정이 가능.
- 소스 코드 입력에 좋다.
 - 코딩하고 나서 소스를 그대로 입력 할 수 있기 때문에 복사해서 쓰기좋다.
- 프로그래머는 마크다운을 써야한다.
 - 논문, 저널, Github 등 프로그래머라면 마크다운을 필수 이다.

1.3 마크다운 문법 정리

- 헤더
제목을 만들려면 # 다음에 제목을 작성 합니다. 사용하는 숫자 기호의 수는 제목 수준과 일치 해야합니다 HTML의 <h1> 부터 <h6> 까지 제목을 표현할 수 있습니다.
 - h1: # 부(parts)에 사용됨.
 - h2: ## 장(chapters)에 사용됨.
 - h3: ###, 페이지 섹션에 사용함.
 - h4: ####, 하위 섹션에 사용됨.
 - h5: #####, 하위 섹션 아래의 하위 섹션에 사용됨.
 - h6: #####, 문단에
- 줄 바꿈
문장의 줄바꿈은 거의 작성된 문서에 형식을 따라 갑니다.(즉, "enter"로 줄바꿈 구분함) 적용이 안될 경우 아래를 참조 합니다.

동해물과 백두산이 마르고 닳도록
하느님이 보우하사 우리나라 만세

무궁화 삼천리 화려 강산

대한 사람 대한으로 길이 보전하세

결과

동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라 만세 무궁화 삼천리 화려 강산
대한 사람 대한으로 길이 보전하세

- 이미지

이미지를 연결 하려면 다음과 같이하십시오.
[Imgur](https://i.imgur.com/4EjEpQX.png)

이미지를 임베드하려면 다음과 같이하십시오.
![Imgur](https://i.imgur.com/Esgh1hd.png)

내부용 이미지를 연결하려면 다음과 같이하십시오.
Format: [Alt Text](Media/images/figure-reference.png)

내부용 이미지를 임베드하려면 다음과 같이하십시오.
Format: ![Alt Text](Media/images/figure-reference.png)

이때 내부용 이미지를 표현할때 상대주소로 해주는것이 적절하다.

그래야 폴더를 열었을때 사용자가 쉽게 이미지를 할수 있다.

절대경로로 하였을때 C드라이브 - 사용자 이름 폴더 - 등등 으로 참조가 되었을때
다른 사용자와 PC환경이 달라지기 때문에 절대경로 참조로 이미지 확인이 불가능해진다.

- 코드 블록

코드블록을 문서 중간에 넣으려면 띄어 쓰기로 코드 블록을 삽입 할 수 있다.

또는 `을 세번 입력하여 생성 가능하다.

``` 을 넣어서 이렇게 블록처리가 가능.

## 2. gcc

### 2.1 gcc란?

- **GNU 컴파일러 모음**(GNU Compiler Collection, 줄여서 **GCC**)는 [GNU 프로젝트](#)의 일환으로 개발되어 널리 쓰이고 있는 [컴파일러](#)이다.

GCC는 원래 C만을 지원했던 컴파일러로 이름도 GNU C 컴파일러 였다. 그러나 나중에 C++, 자바, 포트란, 에이다 등 여러 언어를 컴파일 할 수 있게 되면서, 현재의 이름으로 바뀌게 되었다.

- GCC는 [리처드 스톨만](#)이 1987년 GNU 프로젝트의 컴파일러로 작성했다.

GNU 시스템의 공식 컴파일러이므로 GCC는 많은 컴파일러와 운영 체제를 만드는 데 사용되었다. 한편, 시스템 네이티브 컴파일러를 사용했을 때 비해서 GCC를 사용하면 같은 파서로 코드를 처리하므로 이식성을 향상시킬 수 있다. GCC는 상용 컴파일러에 비해서 느린 코드를 생성했지만 최근 많이 개선되었다.

## 2.2 gcc 설치

- 리눅스에서의 gcc 설치 터미널 명령어

```
sudo apt-get install gcc
```

```
sudo apt-get install g++
```

gcc와 g++을 같이 설치하도록 하자.

## 2.2 gcc 컴파일 방법

- gcc 버전확인

```
gcc -v
```

- gcc 로 컴파일 할 파일명

```
gcc 파일명.c
```

a.out이라는 실행 파일이 생성된다.

- 실행파일의 이름을 지정하여 컴파일

```
gcc -o test test.c
```

test라는 이름으로 실행파일이 생성

- 실행파일 이름을 지정하지 않고 컴파일만하고 링크하지 않기

```
gcc -c test.c
```

test.o이라는 오브젝트 파일만 생성되었다. 이때 C파일 이름을 따라가게 된다.

## 2.3 gcc를 이용하여 컴파일 해보기

- Hello World 출력 프로그램 컴파일

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

```
gcc -o test Hello.c
```

```
kim@kim-VirtualBox:~/Coding$ ls
Hello.c HW1 HW2 LinkedRead LinkedRead.c LinkedRead.o Makefile
kim@kim-VirtualBox:~/Coding$ gcc -o test Hello.c
kim@kim-VirtualBox:~/Coding$ ls
Hello.c HW1 HW2 LinkedRead LinkedRead.c LinkedRead.o Makefile test
kim@kim-VirtualBox:~/Coding$./test
Hello world
kim@kim-VirtualBox:~/Coding$
```

Hello world가 정상 출력 된 모습.

## 3. vi 에디터

### 3.1 vi 란?

- vi는 유닉스 환경에서 가장 많이 쓰이는 문서 편집기이다.  
vi라는 이름은 한 줄씩 편집하는 줄단위 편집기가 아니라 한 화면을 편집하는 비주얼 에디터 라는 뜻에서 유래했다. 간결하면서도, 강력한 기능으로 열광적인 사용자가 많다.
- vi는 명령모드(command mode)와 편집모드(insert mode)가 있으며 일단 프로그램을 시작하면 일반적으로 명령모드로 시작하게 된다.  
이때 i 키를 누르면 편집모드로 들어갈 수 있다.
- esc키를 눌러서 편집모드로 나시 빠져 나올 수 있다.

### 3.2 vi에디터 설치

- vi 에디터를 리눅스 환경에서 설치해보자.

```
sudo apt-get install vi
```

- vim도 설치 해보자

```
sudo apt-get install vim
```

- 그래픽을 제공하는 gvim도 설치해보자

```
sudo apt-get install gvim
```

### 3.3 vi 명령어

- vi에는 명령모드, 편집모드 두가지가 존재한다.  
명령어를 잘 사용하면 빠르게 코딩이 가능하다.  
따라서 명령어를 숙지하고 있는것이 매우 중요하다.
- 명령모드 에서의 명령어 모음

| 모드           | 명령키                                        | 설명                          |
|--------------|--------------------------------------------|-----------------------------|
| 마지막<br>행 모드  | :q                                         | vi에서 작업한것이 없을때<br>vi 종료합니다. |
| :q!          | 작업한 내용을 저장하지 않고 종료합니다.                     |                             |
| :w[파일<br>명]  | 작업한 내용을 저장만 한다. 파일명을 지정하<br>면 새 파일로 저장합니다. |                             |
| :wq.<br>:wq! | 작업한 내용을 저장하고 vi를 종료합니다.                    |                             |
| 명령 모<br>드    | ZZ (대문자)                                   | 작업한 내용을 저장하고<br>vi를 종료합니다.  |

- 입력모드로 전환하기

| 명령키      | 설명                              |
|----------|---------------------------------|
| i        | 현재 커서 앞에 입력합니다.                 |
| a        | 현재 커서 뒤에 입력합니다.                 |
| o        | 커서가 위치한 행의 다음 행에 입력합니다.         |
| I(대문자 i) | 커서가 위치한 행의 첫 컬럼으로 이동하여 입력합니다.   |
| A        | 커서가 위치한 행의 마지막 컬럼으로 이동하여 입력합니다. |
| O        | 커서가 위치한 행의 이전 행에 입력합니다.         |

- 커서 이동하기

| 명령키       | 설명                                |
|-----------|-----------------------------------|
| k         | 커서를 위로 이동합니다.                     |
| j         | 커서를 아래로 이동합니다.                    |
| h         | 커서를 왼쪽으로 이동합니다.                   |
| l         | 커서를 오른쪽으로 이동합니다.                  |
| ^ / O     | 커서를 현재 행의 처음으로 이동합니다.             |
| \$        | 커서를 현재 행의 마지막으로 이동합니다.            |
| -         | 커서를 한줄 위 처음으로 이동합니다.              |
| + / Enter | 커서를 다음 행의 처음으로 이동합니다.             |
| H         | 커서를 화면의 맨 윗행으로 이동합니다.             |
| M         | 커서를 화면의 중간 행으로 이동합니다.             |
| L         | 커서를 화면의 맨 아랫행으로 이동합니다.            |
| w         | 커서를 다음 단어의 첫 글자로 이동합니다.           |
| b         | 커서를 앞 단어의 첫 글자로 이동합니다.            |
| e         | 커서를 다음 단어의 마지막 글자로 이동합니다.         |
| G         | 파일의 마지막 행으로 커서를 이동합니다.            |
| 행번호G      | 지정한 행 번호로 커서를 이동합니다.              |
| :행번호      | 지정한 행 번호로 커서를 이동합니다. (마지막 행 모드)   |
| :\$       | 파일의 마지막 행으로 커서를 이동합니다. (마지막 행 모드) |

- 화면 이동하기

| 명령 키                 | 설명                   |
|----------------------|----------------------|
| Ctrl + u             | 화면의 절반 만큼 위로 이동합니다.  |
| Ctrl + o             | 화면의 절반 만큼 아래로 이동합니다. |
| Ctrl + b / Page Up   | 한화면 위로 이동합니다.        |
| Ctrl + f / Page Down | 한화면 아래로 이동합니다.       |
| Ctrl + y             | 화면을 한 행 위로 이동합니다.    |
| Ctrl + e             | 화면을 한 행 아래로 이동합니다.   |

- 내용 수정하기

| 명령 키             | 설명                                                             |
|------------------|----------------------------------------------------------------|
| r                | 커서가 위치한 글자를 다른 글자로 수정합니다.                                      |
| cw, [수정할 단어 수]cw | 커서위치에서부터 현재 단어의 끝까지 수정합니다. 숫자cw는 커서 위치로부터 지정한 숫자의 단어 만큼 수정합니다. |
| s, [수정할 글자 수]s   | 커서 위치로부터 ESC키를 입력할때까지 수정합니다. 숫자s는 커서 위치로부터 지정한 숫자만의 글자를 수정합니다. |
| cc               | 커서가 위치한 행의 내용을 모두 수정합니다.                                       |
| C                | 커서 위치로부터 행의 끝까지 수정합니다.                                         |

- 내용 삭제하기

| 명령 키             | 설명                                            |
|------------------|-----------------------------------------------|
| x, [삭제할 글자 수]x   | 커서가 위치한 글자를 삭제합니다. x앞에 삭제할 글자수를 지정할 수도 있습니다.  |
| dw, [삭제할 단어 수]dw | 커서가 위치한 단어를 삭제합니다. dw앞에 삭제할 단어수를 지정할 수도 있습니다. |
| dd, [삭제할 행 수]dd  | 커서가 위치한 행을 삭제합니다. dd앞에 삭제할 행의수를 지정할 수도 있습니다.  |
| D                | 커서 위치로부터 행의 끝까지 삭제합니다.                        |

- 명령 취소하기

| 명령 키 | 설명                            |
|------|-------------------------------|
| u    | 명령을 취소합니다.                    |
| U    | 해당 행에서 한 모든 명령을 취소합니다.        |
| :e!  | 마지막으로 저장한 내용 이후의 것을 모두 취소합니다. |

- 범위 지정하기

| 명령 키     | 설명                        |
|----------|---------------------------|
| 1, \$, % | 1행부터 마지막 행까지 범위를 지정합니다.   |
| 1,.      | 1행부터 커서가 있는 행까지 지정합니다.    |
| ., \$    | 커서가 있는 행부터 마지막 행까지 지정합니다. |
| ., +2    | 현재 행과 두번째 아랫행까지 지정합니다.    |
| 10, 20   | 10행부터 20행까지 지정합니다.        |

- 내용 복사하기 / 잘라내기 / 붙이기

| 명령 키                  | 설명                                                               |
|-----------------------|------------------------------------------------------------------|
| yy, [복사할<br>행 수]yy    | 커서가 위치한 행을 복사합니다. yy앞에 복사할 행의 수를 지정할 수도 있습니다.                    |
| dd, [잘라내기<br>할 행 수]dd | 커서가 위치한 행을 잘라내기합니다. 삭제와 같은 명령어입니다. dd앞에 잘라내기할 행 숫자를 입력할 수도 있습니다. |
| p                     | 커서가 위치한 행의 아래쪽에 붙여넣기를 합니다.                                       |
| P                     | 커서가 위치한 행의 위쪽에 붙여넣기를 합니다.                                        |

- 내용 검색하기

| 명령 키 | 설명                     |
|------|------------------------|
| /문자열 | 문자열을 커서 아래 방향으로 검색합니다. |
| ?문자열 | 문자열을 커서 위 방향으로 검색합니다.  |
| n    | 원래 방향으로 다음 문자열을 검색합니다. |
| N    | 반대 방향으로 다음 문자열을 검색합니다. |

- 내용 치환하기

| 명령 키                        | 설명                                                |
|-----------------------------|---------------------------------------------------|
| :s/[대상문자열]/[바꿀문<br>자열]      | 커서가 위치한 행에서 첫번째로 나오는 대상문자열을 바꿀<br>문자열로 바꿉니다.      |
| :%s[대상문자열]/[바꿀문<br>자열]      | 파일 전체에서 모든 대상문자열을 바꿀문자열로 바꿉니다.                    |
| :[범위]s[대상문자열]/[바<br>꿀문자열]   | 범위 내 모든 각 행에서 첫번째로 나오는 대상문자열을 바꿀<br>문자열로 바꿉니다.    |
| :[범위]s[대상문자열]/[바<br>꿀문자열]g  | 범위 내 모든 행에서 대상문자열을 바꿀문자열로 바꿉니다.                   |
| :[범위]s[대상문자열]/[바<br>꿀문자열]gc | 범위 내 모든 행에서 대상문자열을 바꿀문자열로 바꾸되 수<br>정할 지 여부를 묻습니다. |

### 3.4 vi 환경설정

- vi 에디터에서 들여쓰기, 폰트 등 개발환경 설정을 해보자
- 홈 디렉토리에 .vimrc 파일을 생성해야한다.
- 에디터가 실행할때 이 파일을 실행하고나서 에디터가 실행된다.

- 화면 설정

```
syntax on " 형식별 구문 강조 표시
colorscheme [scheme명] " 테마 적용.
set number " 라인 넘버 표시. (= nu)
```



```

set showcmd " 사용자가 입력한 명령어 표시
set showmatch " 현재 선택된 괄호의 쌍을 표시
set relativenumber " 커서를 기준으로 라인 넘버 표시. 커서 위치에 따라 바뀜. (=
rnu)
set cursorline " 커서가 있는 라인을 강조 표시. (= cul)
set ruler " 커서 위치 표시. (= ru)
set laststatus=2 " 상태바 표시. (= ls) [0: 상태바 미표시 / 1: 2개 이상의 윈도우에서 표시 / 2: 항상 표시]
" 상태바 커스터마이징 %<item>으로 사용하며, \는 구분자로 공백을 넣을 경우는 구분자를 넣어줘야함.
set statusline=%F\ %y%m%r\ %=Line:\ %l/%L\ [%p%\]\ Col:%c\ Buf:%n
hi statusline ctermfg=white ctermbg=4 cterm=none "활성화된 상태바 배경색 및
폰트색 설정
hi statuslineNC ctermfg=white ctermbg=8 cterm=none " 윈도우가 2개 이상인 경우 비활성화된 윈도우의 배경색 및 폰트색 설정
set mouse=a " 마우스로 스크롤 및 리사이즈 가능. [n : Normal mode / v : Visual mode / i : Insert mode / a : All modes]

```

- 검색 설정

```

set hlsearch " 검색된 결과 강조 표시. (= hls)
set ignorecase " 검색시 대소문자를 구분하지 않음. (= ic)
set incsearch " 검색어를 입력할 때마다 일치하는 문자열을 강조해서 표시. (= is)
set smartcase " ignore 옵션이 켜져있더라도 검색어에 대문자가 있다면 정확히 일치하는 문자열을 찾음. (= scs)

```

- 들여쓰기 설정

```

set autoindent " 새로운 라인이 추가될 때, 이전 라인의 들여쓰기에 자동으로 맞춤. (= ai)
set expandtab " Tab을 Space로 변경. (= et)
set tabstop=4 " 탭으로 들여쓰기시 사용할 스페이스바 개수. (= ts)
set shiftwidth=4 " <<, >> 으로 들여쓰기시 사용할 스페이스바 개수. (= sw)
set softtabstop=4 " 스페이스바 n개를 하나의 탭으로 처리. (= sts)
" ex) 스페이스바 4개가 연속으로 있다면 백스페이스로 스페이스바를 지우면 스페이스바 4개를 하나의 탭으로 인식해 삭제.
filetype indent on " indent.vim 파일에 설정된 파일 형식별 들여쓰기 적용.

```

- 입력 설정

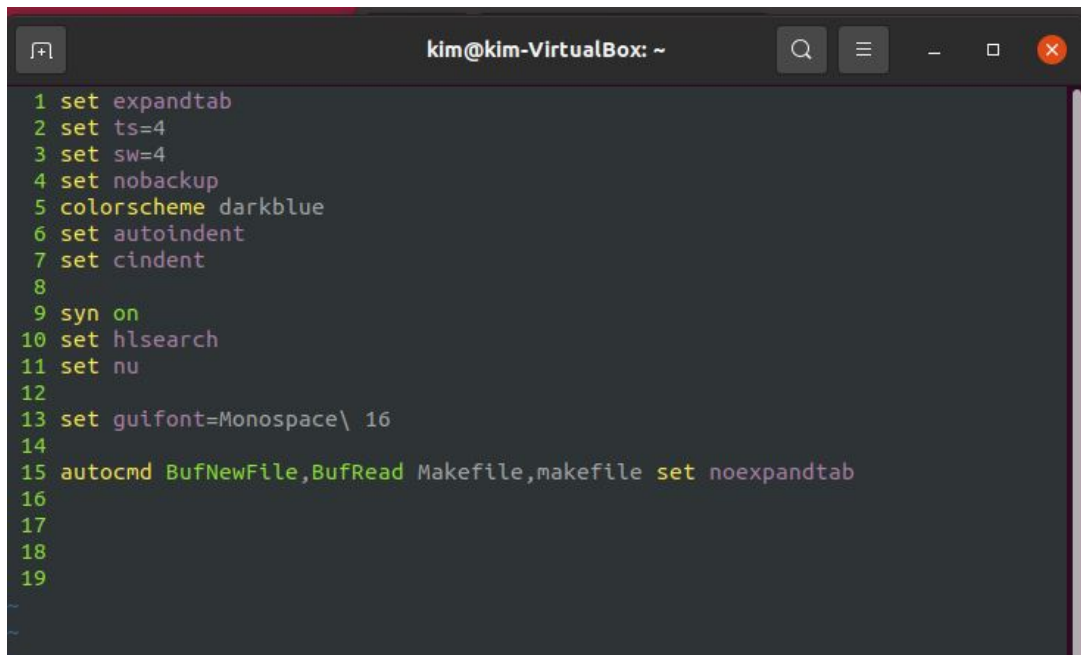
```

set clipboard=unnamed " vim에서 복사한 내용이 클립보드에 저장
set backspace=eo1,start,indent " 라인의 시작과 끝의 들여쓰기를 백스페이스로 지움.
set history=1000 " 편집한 내용 저장 개수 (되돌리기 제한 설정)
set paste " 다른 곳에서 복사한 내용을 붙여넣을 때, 자동 들여쓰기가 적용되는 것을 막아 복사한 내용을 들여쓰기없이 복사.
set pastetoggle=<F2> " paste 옵션이 적용되면 들여쓰기가 옵션이 제대로 작동하지 않기 때문에 toggle식으로 옵션을 키고 끌 수 있음.

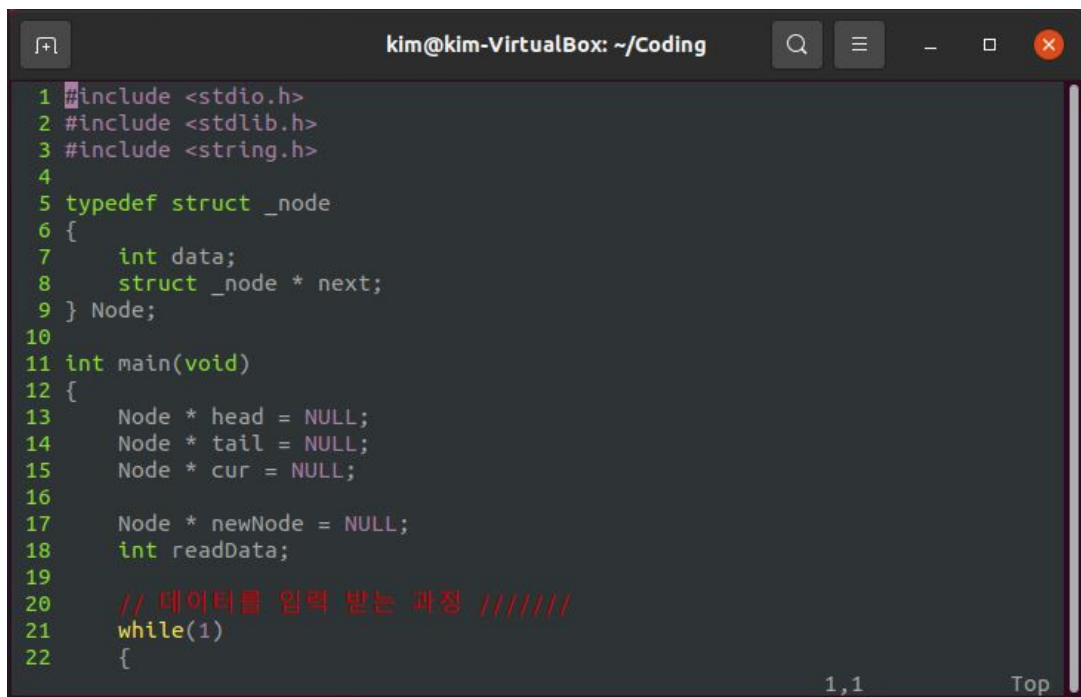
```

### 3.5 vi 에디터 환경설정 적용

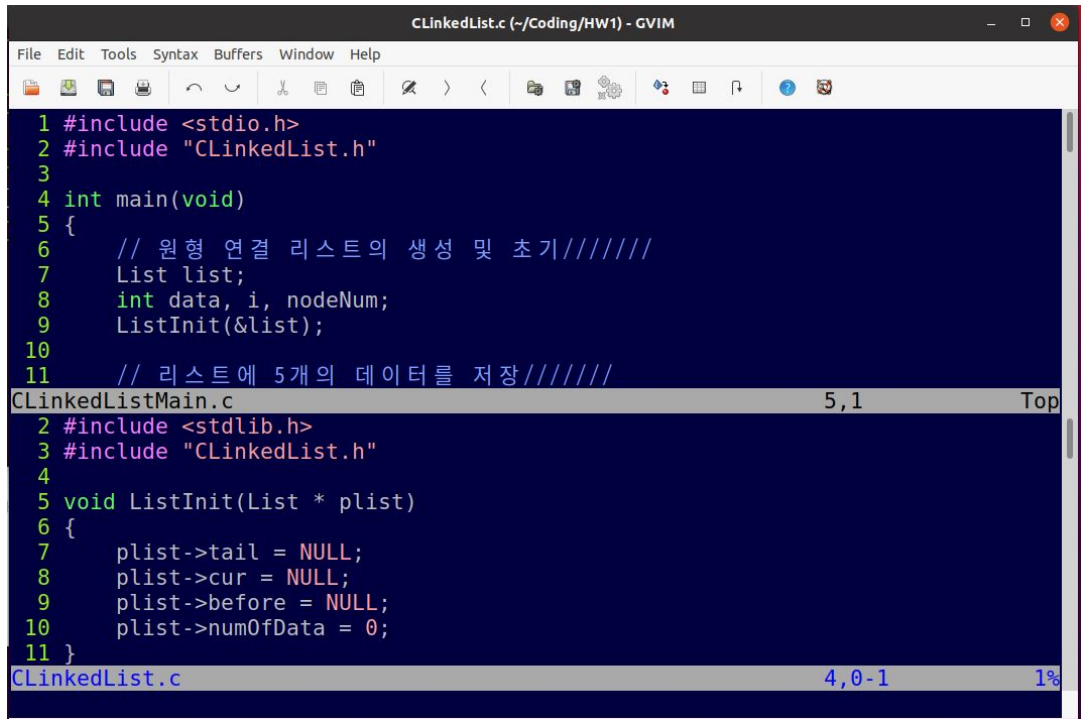
- vi 에디터 환경설정 명령어가 많은데 실제로 적용해보자.



```
kim@kim-VirtualBox: ~
1 set expandtab
2 set ts=4
3 set sw=4
4 set nobackup
5 colorscheme darkblue
6 set autoindent
7 set cindent
8
9 syn on
10 set hlsearch
11 set nu
12
13 set guifont=Monospace\ 16
14
15 autocmd BufNewFile,BufRead Makefile,makefile set noexpandtab
16
17
18
19
```



```
kim@kim-VirtualBox: ~/Coding
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 typedef struct _node
6 {
7 int data;
8 struct _node * next;
9 } Node;
10
11 int main(void)
12 {
13 Node * head = NULL;
14 Node * tail = NULL;
15 Node * cur = NULL;
16
17 Node * newNode = NULL;
18 int readData;
19
20 // 데이터를 입력 받는 과정 //////////
21 while(1)
22 {
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
216
```



```
1 #include <stdio.h>
2 #include "CLinkedList.h"
3
4 int main(void)
5 {
6 // 원형 연결 리스트의 생성 및 초기////////
7 List list;
8 int data, i, nodeNum;
9 ListInit(&list);
10
11 // 리스트에 5개의 데이터를 저장////////
CLinkedListMain.c 5,1 Top
2 #include <stdlib.h>
3 #include "CLinkedList.h"
4
5 void ListInit(List * plist)
6 {
7 plist->tail = NULL;
8 plist->cur = NULL;
9 plist->before = NULL;
10 plist->numOfData = 0;
11 }
CLinkedList.c 4,0-1 1%
```

- 이중창 명령어

:sp 띄울파일명

- 명령창 이동

Ctrl + ww

해당 명령어로 다른창으로 이동이 가능하다.

## 4. Makefile

### 4.1 Makefile 이란?

- Makefile은 파일 관리 유틸리티이다.
- 우리는 앞서 gcc를 이용하요 소스파일을 컴파일 해보았다.  
그러면 항상 gcc를 입력해서 컴파일 해야되나?
- Makefile은 일련의 gcc명령어를 집합시켜서 make 커맨드 한번에 컴파일 할 수있도록 해주는 강력한 유틸리티이다.

### 4.2 Make를 쓰는 이유

- 각 파일에 대한 반복적 명령의 자동화로 인한 시간 절약
- 프로그램의 종속 구조를 빠르게 파악 할 수 있으며 관리가 용이
- 단순 반복 작업 및 재작성을 최소화

## 4.3 Makefile 만들기

- 수업시간에 만든 Makefile 파일 예시 이다.

```
help:
 echo "make all"
 echo "make clean"

all:
 gcc -c main.c foo.h
 gcc -c foo.c foo.h

 gcc -o main main.o foo.o foo.h

clean:
 rm *.o
 rm main
```

- make help  
터미널에서 make help를 입력하면 all 과 clean이 정의되어 있다고 알려준다.
- make all  
터미널에서 make all을 입력하면 all(타겟)에 레이블이 되어 있으므로 gcc -c main.c 부터 차례대로 명령을 실행한다.
- make clean  
터미널에서 make clean을 입력하면 clean(타겟)에 레이블이 되어 있으므로 rm \*.o  
rm main  
등 미리 정의된 삭제 리눅스 명령어 실행된다.
- 중요한점은 레이블을 생성하고 Tab이 꼭 들어가야 된다. !!!!
- Makefile은 꼭 C, 헤더 등 같은 디렉토리에서 M은 대문자로 만들어야한다!

## 4.4 Makefile로 컴파일 해보기

```

kim@kim-VirtualBox:~/uP/00_compilation$ ls
foo.c foo.h main.c Makefile
kim@kim-VirtualBox:~/uP/00_compilation$ make all
gcc -c main.c foo.h
main.c: In function 'main':
main.c:8:5: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
 8 | printf("return 성공\n");
 | ~~~~~
main.c:8:5: warning: incompatible implicit declaration of built-in function 'printf'
main.c:4:1: note: include '<stdio.h>' or provide a declaration of 'printf'
 3 | #include "foo.h"
 +++ |+#include <stdio.h>
 4 |
gcc -c foo.c foo.h
gcc -o main main.o foo.o foo.h
kim@kim-VirtualBox:~/uP/00_compilation$./main
return 성공
kim@kim-VirtualBox:~/uP/00_compilation$

```

- 간단한 파일로 헤더파일과 C파일로 실행파일을 만들어 보았다.

실행사진 파일을 보면 make all 커맨드로 gcc 명령어가 실행된 모습을 보일 수 있다.

## 5. Header

### 5.1 헤더 파일이란?

- 컴퓨터 프로그래밍에서, 특히 C와 C++ 프로그래밍 언어에서, 헤더 파일(header file) 또는 인클루드 파일

(include file)은 컴파일러에 의해 다른 소스 파일에 자동으로 포함된 소스 코드의 파일이다.

일반적으로 헤더 파일들은 다른 소스 파일 속의 첫 부분에 포함된다.

- 우리가 자주쓰는 stdio.h 과 stdlib.h 등등 컴파일러에 포함된 소스 파일이다.
- 프로그래밍 시작시 맨 처음에 선언하는 이유가 여기 소스파일부터 읽어서 프로그램을 시작하라는 의미이다.

### 5.2 헤더파일 작성 Tip

- 수업시간에 작성한 파일에는 #ifndef \_name 이라고 정의된 부분이 없다.
- 이말인 즉슨, 헤더파일이 프로그램에서 딱 한번만 필요하기때문에 사용한다.

```

#ifndef _FOO_H
#define _FOO_H
~~~
#endif

```

으로 헤더파일을 작성하는게 좋다.

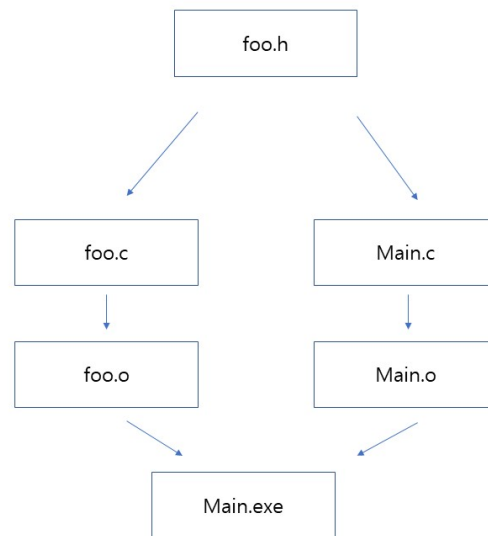
## 5.3 헤더파일을 사용하는 이유는?

- 일일이 프로그래밍 하는 프로그래머는 좋은 성과를 낼 수 없다.
- 아무리 머리가 좋아도, 더 머리가 뛰어난 사람이 제공하는 라이브러리가 있다.
- 굳이 일일이 코딩 할 필요 없이 라이브러리(헤더) 파일을 적극 사용하여 프로그램을 만들자.
- C언어 및 여러 컴파일러에서 제공해주는 헤더파일은 수백가지가 넘는다.
- 이미 기능은 만들어져 있고 우리는 잘 사용하기만 하면된다.
- 또한 사용자 정의 헤더파일을 만들때, 프로그래밍을 좀 더 간편화 하기위하여 사용한다 !!!

## 6. C 코드 정리

### 6.1 수업 중 만든 프로그램

- 수업중 만든 프로그램을 정리해보자



- 우리가 만든 프로그램의 조직을 나타내는 그림이다.
- foo.h가 먼저 선언되어 있고 foo.c 와 main.c 각각 참조한다.
- 오브젝트 파일이 생성되며, 실행파일을 만들수 있다.

### 6.2 코드 분석

- foo.h 파일

```
int foo(int i);
```

foo 라는 함수만 정의 되어 있다.

- foo.c 파일

```
#include "foo.h"

int foo (int i){
    return i+1;
}
```

foo 헤더를 참고하고 foo 함수에 대한 것이 정의 되어 있다. 호출된 값의 +1을 하여 반환한다.

- main.c 파일

```
#include "foo.h"
int main(){
    int a = foo(10);
    if(a==11){
        printf("return 성공\n");
    }
    else{ }

    return 0;
}
```

foo 함수를 호출하여 호출한 값의 +1이 들어오면 return 성공을 출력한다.

## 6.3 어떤 코드가 좋은 코드인가?

- 프로그램 작성시 좋은 코드를 만드려면 어떻게 하면 될까?
  - 프로그램을 단순화 하자.
  - 헤더파일을 만들자
  - 프로토타입을 정의하고 파일을 세분화하자
  - 누구나 알수 있게 코드를 짜자
  - 주석을 열심히 달자
  - Easy is Simple !!
- 나만의 라이브러리를 만들어서 적극 활용하자 !!!!
- 고수의 라이브러리 불러오는 습관도 활용하자 !!!!