

# Flex & Bison 개발 환경 설정

- OS : Ubuntu 18.04 LTS, Mac OS
- Tool : bash shell, Vim ( or Nano, Emacs etc.. )

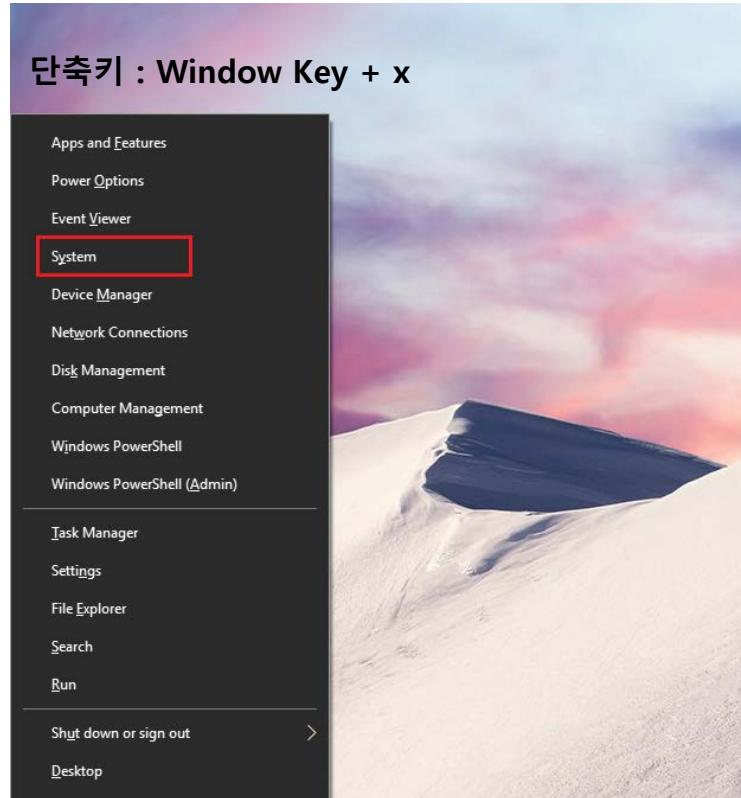
# 개요

1. WSL(Linux용 Windows 하위 시스템 설치) 방법
  2. Ubuntu 초기 환경 설정
  3. Flex & Bison 설치
  4. Word Count 프로그램 예제
- 부록. Makefile 만들기
- 부록. VS Code에서 실행

# 1. WSL (Linux용 Windows 하위 시스템) 설치 방법

※ 현재 사용하는 OS가 리눅스 배포판의 한 종류이거나 MacOS이면 건너뛰시면 됩니다.

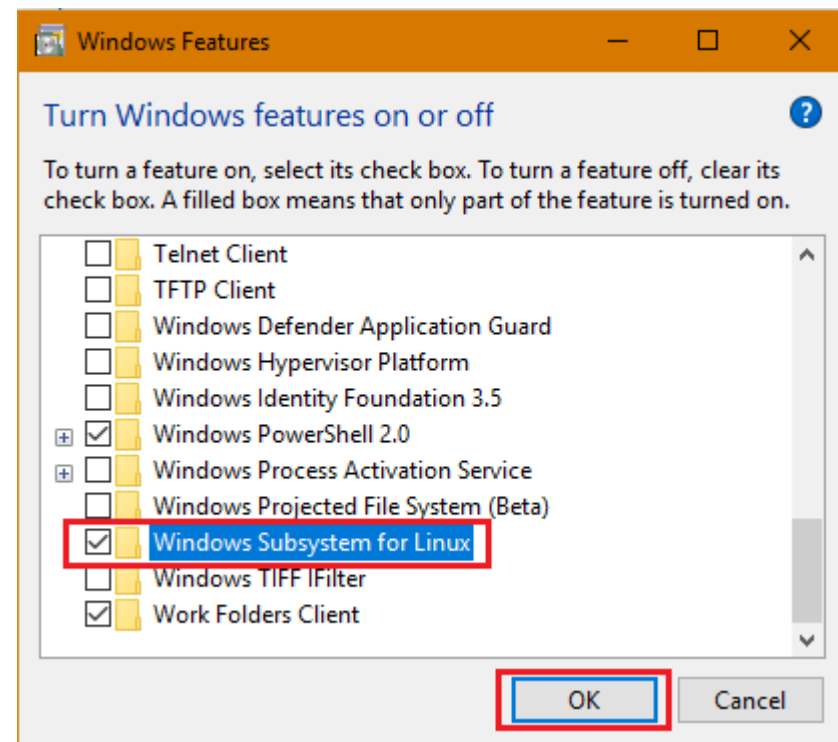
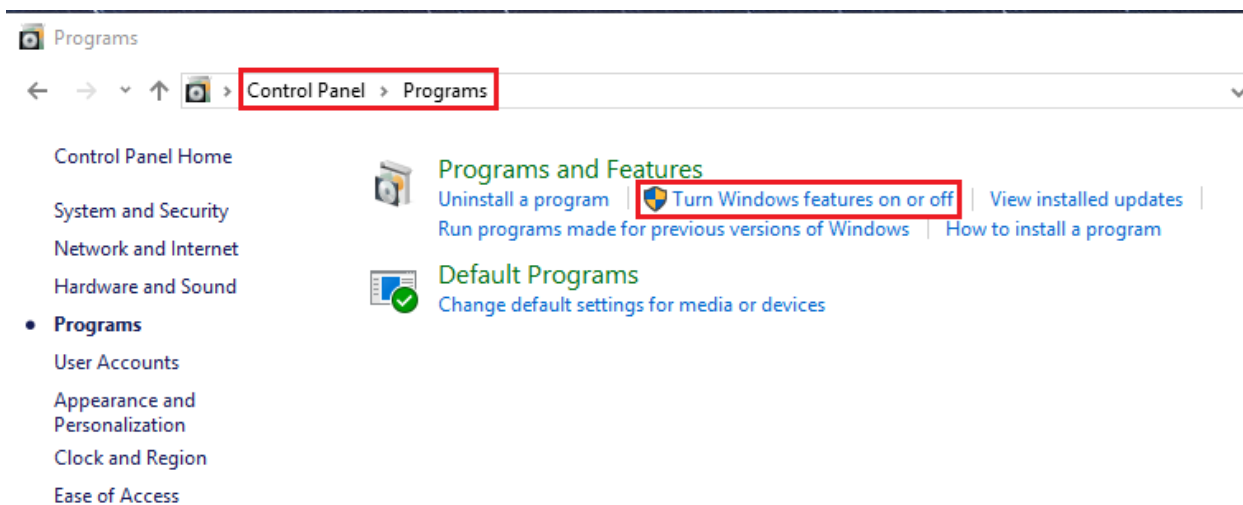
STEP 1. Windows10 버전 확인 – 꾸준히 업데이트를 진행했으면 문제 없습니다.  
( Windows10 OS 버전이 **1607** 이상이어야 합니다. )



## Windows specifications

Edition	Windows 10 Enterprise
<b>Version</b>	<b>1803</b>
Installed on	2/9/2019
OS build	17134.590

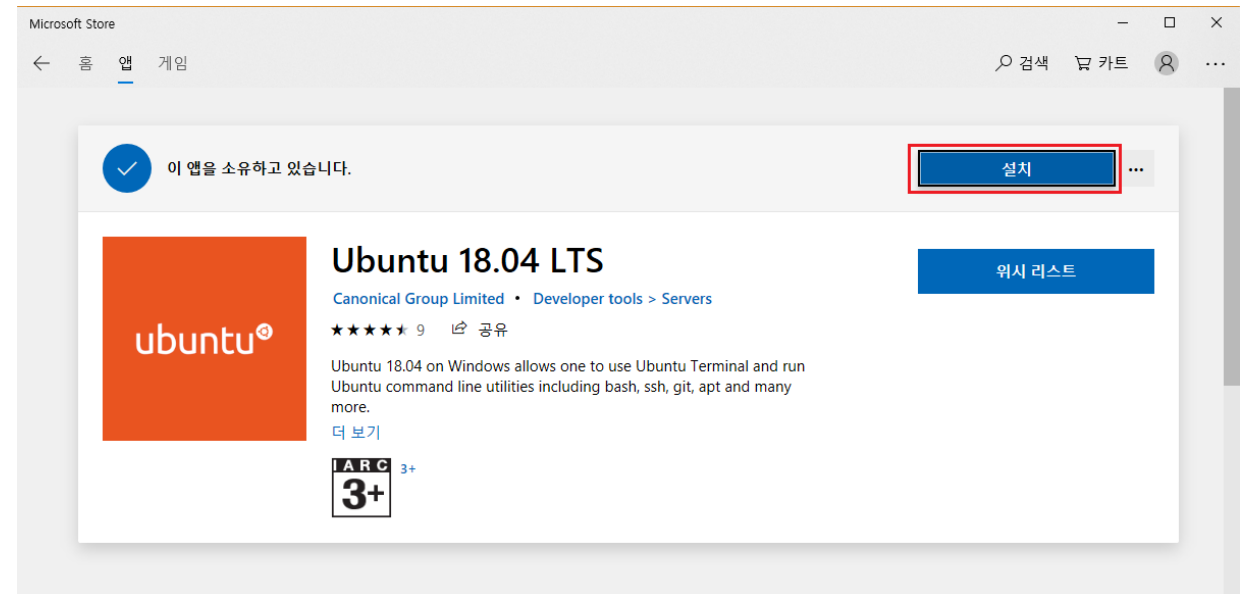
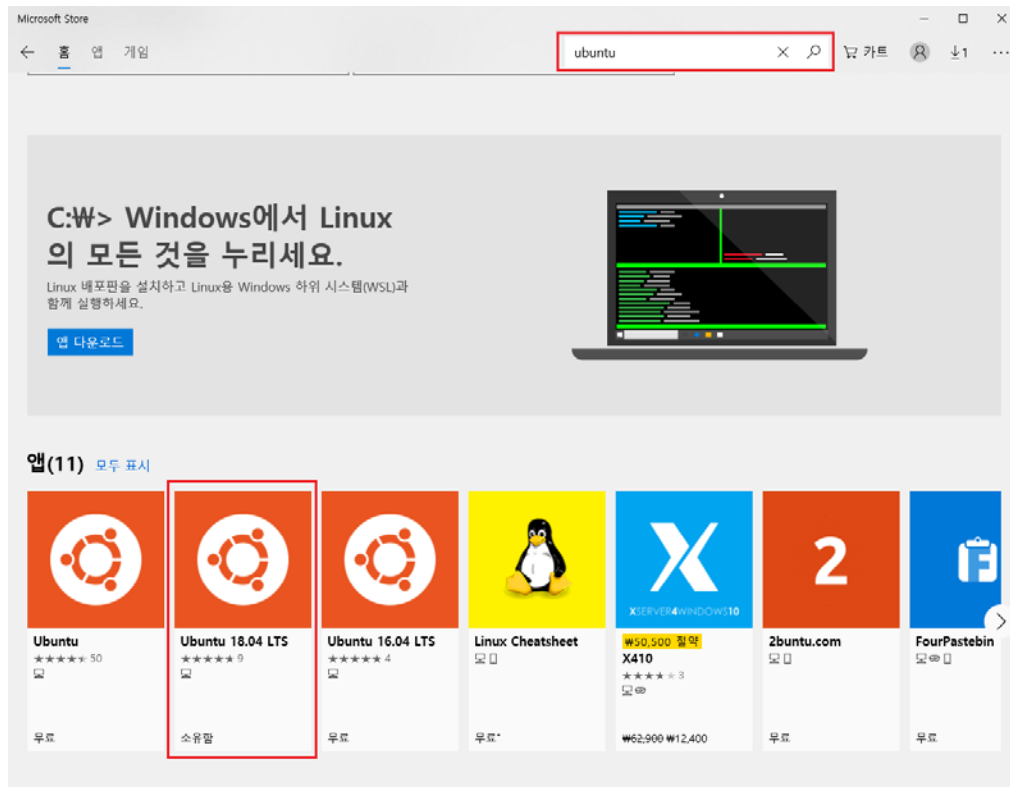
STEP 2. Linux용 Windows 하위시스템 활성화 후, **재부팅**  
( 제어판 -> 프로그램 -> Windows 기능 켜기/끄기 )



# 1.

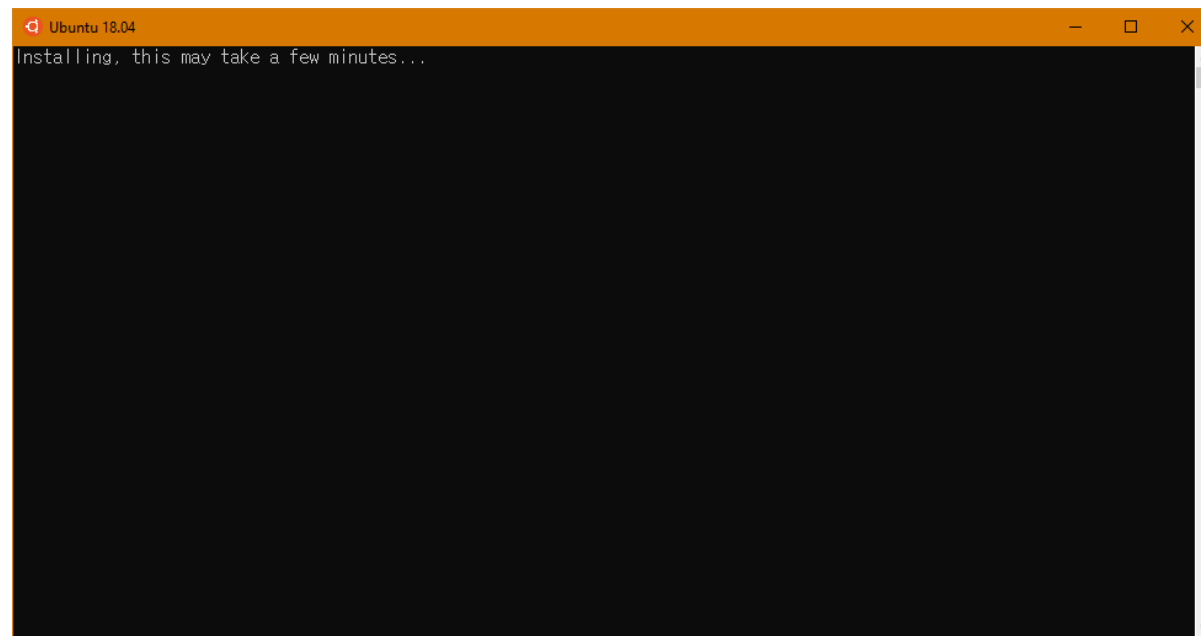
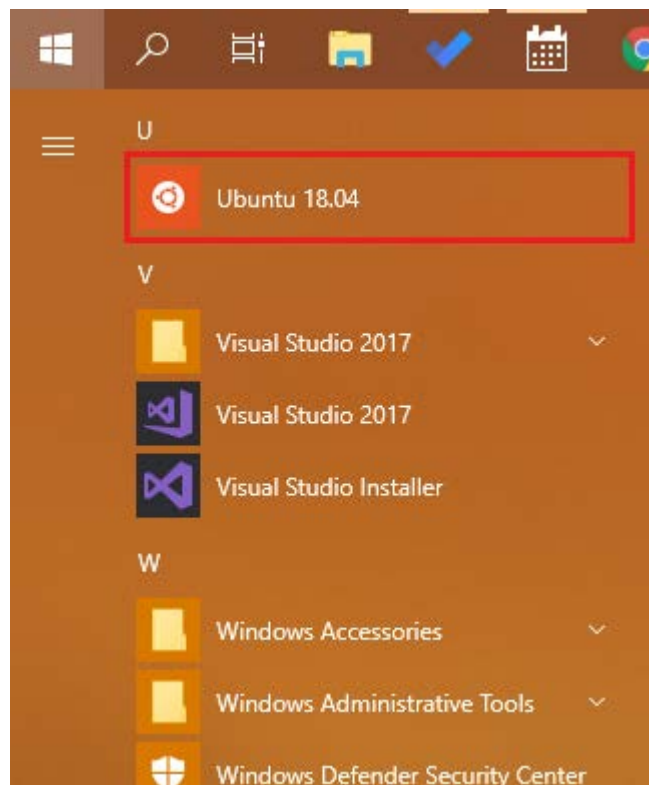
# WSL 설치 방법

**STEP 3. Microsoft Store에서 Ubuntu 18.04 LTS 설치**  
( 선호하는 다른 버전이나 다른 배포판을 설치해도 됩니다. )



## STEP 3. Ubuntu 18.04 실행

( 최초 실행 시, 몇 분간 설치를 진행 합니다. )



## STEP 4. 사용자 계정 생성

( 패스워드 설정 시, 화면에 출력되지 않으므로 주의해서 입력하세요. )

```
ing5uny@DESKTOP-79V69B3: ~  
Installing, this may take a few minutes...  
Please create a default UNIX user account. The username does not need to match your Windows username.  
For more information visit: https://aka.ms/wslusers  
Enter new UNIX username: ing5uny  
Enter new UNIX password:  
Retype new UNIX password:  
passwd: password updated successfully  
Installation successful!  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
ing5uny@DESKTOP-79V69B3: ~$
```



## 2. Ubuntu 초기 환경 설정

( MacOS도 실행 명령어만 다를 뿐 과정은 모두 같습니다.)

STEP 1. 설치되어 있는 패키지들, 최신 버전으로 업그레이드  
( 인터넷 환경에 따라 몇 분 정도 시간이 소요 됩니다. )

```
ing5uny@DESKTOP-79V69B3:~$ sudo apt-get update && sudo apt-get upgrade -y
```

```
Processing triggers for systemd (237-3ubuntu10.53) ...  
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...  
Processing triggers for dbus (1.12.2-1ubuntu1.3) ...  
Processing triggers for rsyslog (8.32.0-1ubuntu4.2) ...  
invoke-rc.d: could not determine current runlevel  
Processing triggers for ufw (0.36-0ubuntu0.18.04.2) ...  
Processing triggers for mime-support (3.60ubuntu1) ...  
Processing triggers for ca-certificates (20211016~18.04.1) ...  
Updating certificates in /etc/ssl/certs...  
0 added, 0 removed; done.  
Running hooks in /etc/ca-certificates/update.d...  
done.  
ing5uny@DESKTOP-79V69B3:~$
```

## STEP 2. Git 설치

( WSL은 STEP1을 했다면 이미 최신 버전이 설치되어 있습니다. )

```
ing5uny@DESKTOP-79V69B3:~$ sudo apt-get install git -y
```

```
ing5uny@DESKTOP-79V69B3:~$ git --version  
git version 2.17.1
```

제대로 설치되었는 지 확인하기 위해 꼭 버전 정보를 출력 해 보세요.

STEP 2. 텍스트 에디터 설치. 본인의 취향에 따라 Vi, Vim, nano, Emacs 등 골라 쓰세요.  
( WSL은 STEP1을 했다면 Emacs를 제외한 나머지는 이미 최신 버전이 설치되어 있습니다. )

```
ing5uny@DESKTOP-79V69B3: ~$ sudo apt-get install vim -y
```

```
ing5uny@DESKTOP-79V69B3: ~$ sudo apt-get install nano -y
```

※ nano가 윈도우의 메모장과 가장 유사합니다. Vim이나 Vi 등의 텍스트 에디터를 사용하는 데 적응하기 힘든 분은 nano를 쓰시면 됩니다.

# 3. Flex & Bison 설치 방법

( MacOS도 실행 명령어만 다를 뿐 과정은 모두 같습니다.)

# 3.

## Flex & Bison 설치 방법

### STEP 1. Flex 설치

( 인터넷 환경에 따라 몇 분 정도 시간이 소요 됩니다. )

```
ing5uny@DESKTOP-79V69B3: ~$ sudo apt-get install flex -y
```

```
ing5uny@DESKTOP-79V69B3: ~$ flex --version  
flex 2.6.4
```

# 3.

## Flex & Bison 설치 방법

### STEP 2. Bison 설치

( 인터넷 환경에 따라 몇 분 정도 시간이 소요 됩니다. )

```
ing5uny@DESKTOP-79V69B3:~$ sudo apt-get install bison -y
```

```
ing5uny@DESKTOP-79V69B3:~$ bison --version
bison (GNU Bison) 3.0.4
Written by Robert Corbett and Richard Stallman.

Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

## 4. Word Count 프로그램 예제



## STEP 1. 프로젝트 디렉토리 생성(WSL에서 실행)

```
kanteloper@DESKTOP-HV0OS4J:~$ mkdir compiler  
kanteloper@DESKTOP-HV0OS4J:~$ ls  
compiler
```

1. 작업 폴더 생성

2. 디렉토리 내용 확인

3. 작업 디렉토리가 생성 되었음을 확인

( WSL의 경우, 최초 설치 시 %home%(user\_name) 디렉토리에 아무것도 없습니다.)

**STEP 2. wc.y / wc.l 파일, 프로젝트 디렉토리로 이동**

( wc.y와 wc.l 파일은 아주 Bb 강의 노트 게시판에 첨부되어 있습니다. )

## ※ WSL 사용자의 경우

```
kanteloper@DESKTOP-HV00S4J:~$ cd compiler
kanteloper@DESKTOP-HV00S4J:~/compiler$
```

1. 프로젝트 디렉토리로 이동

```
kanteloper@DESKTOP-HV00S4J:~/compiler$ cp /mnt/c/Users/Ajou/Downloads/wc.y /mnt/c/Users/Ajou/Downloads/wc.l ./
kanteloper@DESKTOP-HV00S4J:~/compiler$ ls
Makefile wc.l wc.y
```

2. 프로젝트 디렉토리로  
wc.y / wc.l 파일 복사

복사할 파일1의 경로

복사할 파일2의 경로

복사된 파일이 저장될  
디렉토리 경로

( Windows 10에서 WSL의 디렉토리로 접근이 불가능 합니다. 따라서 WSL에서 Windows 10의 디렉토리로 접근해야 합니다.  
WSL에서 Windows의 C드라이브 경로는 "/mnt/c", D 드라이브 경로는 "/mnt/d" 입니다. )

## STEP 3. Bison 실행

```
kanteloper@DESKTOP-HV00S4J:~/compiler$ bison -d wc.y
kanteloper@DESKTOP-HV00S4J:~/compiler$ ls
Makefile  wc.l  wc.tab.c  wc.tab.h  wc.y
```

Yacc 파일(.y)을 입력으로 bison를 실행시키면 결과물로 ~.tab.c / ~.tab.h 파일이 생성됩니다.

※ bison을 실행 시킬 때, d 옵션을 주어야 ~.tab.h 파일이 생성 됩니다.

## STEP 4. Flex 실행

```
kanteloper@DESKTOP-HV00S4J:~/compiler$ flex wc.l  
kanteloper@DESKTOP-HV00S4J:~/compiler$ ls  
Makefile lex.yy.c wc.l wc.tab.c wc.tab.h wc.y
```

Lex 파일(.l)을 입력으로 flex를 실행시키면 결과물로 lex.yy.c 파일이 생성됩니다.

## STEP 5. Lex 파일에 ~.tab.h 파일 include

```
1 %{
2
3 #include "wc.tab.h"
4
5 %}
6
7 %x
8
9 [a-zA-Z]+ { yylval.nchars = strlen(yytext); return WORD; }
10 \n      { return NEWLINE; }
11 <<EOF>> { return END; }
12 .      { return ETC; }
13
14 %x
15
```

## STEP 6. gcc로 컴파일

Yacc 라이브러리가 flex 라이브러리보다 먼저 로드되어야 하기 때문에 이 순서로 옵션을 줍니다.

```
kanteloper@DESKTOP-HV00S4J:~/compiler$ gcc -o wc wc.tab.c lex.yy.c -ly -lfl
kanteloper@DESKTOP-HV00S4J:~/compiler$ ls
Makefile  lex.yy.c  wc  wc.l  wc.tab.c  wc.tab.h  wc.y
kanteloper@DESKTOP-HV00S4J:~/compiler$
```

~.tab.c / lex.yy.c 2가지의 파일을 입력으로 컴파일하면 실행 파일이 생성됩니다.

※ gcc로 컴파일 할 때, o 옵션을 주면 실행 파일 이름을 지정할 수 있습니다.

※ gcc로 컴파일 할 때, 꼭 ly, fl 옵션을 주어야 합니다.

-ly : yacc 라이브러리 파일을 링크하기 위해 해당 옵션을 사용합니다.

-lfl : flex 라이브러리 파일을 링크하기 위해 해당 옵션을 사용합니다.

## STEP 6. Word Count 프로그램 실행

```
kanteloper@DESKTOP-HV00S4J:~/compiler$ ./wc
Hello compiler
compiler is so easy.
lines: 2, words: 6, chars: 36
```

아무 문장이나 입력한 뒤, Ctrl + D를 입력하여 프로그램을 종료합니다.  
위와 같이 출력되면 정상 동작한 것입니다.

※ 참고 : 부록에서 Makefile을 이용하여 이와 같은 프로그램 실행 과정을 간단히 해보겠습니다.

# 부록. Makefile 만들기



## STEP 1. 프로그램 빌드 도구인 make 설치

```
ing5uny@DESKTOP-79V69B3:~$ sudo apt-get install make
```

```
ing5uny@DESKTOP-79V69B3:~$ make --version
GNU Make 4.1
Built for x86_64-pc-linux-gnu
Copyright (C) 1988-2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

STEP 2. 본인이 사용하는 텍스트 에디터로 Makefile 만들기  
( 프로젝트 디렉토리 안에 생성하세요. )

```
ing5uny@DESKTOP-79V69B3:~$ cd compiler/  
ing5uny@DESKTOP-79V69B3:~/compiler$ vim Makefile
```

```
1 wc :    wc.l wc.y _  
2      bison -d wc.y  
3      flex wc.l  
4      gcc -o $@ wc.tab.c lex.yy.c -ly -lfl  
5  
6 .PHONY : clean  
7 clean :  
8      rm -rf *.tab.c *.tab.h *.yy.c wc
```

## STEP 2. 프로그램 빌드

```
ing5uny@DESKTOP-79V69B3:~/compiler$ make
bison -d wc.y
flex wc.l
gcc -o wc wc.tab.c lex.yy.c -ly -lfl
ing5uny@DESKTOP-79V69B3:~/compiler$ ls
Makefile lex.yy.c wc wc.l wc.tab.c wc.tab.h wc.y
```

Makefile을 만들면 lex와 yacc의 빌드 과정이 make 명령어 하나로 가능하다.

## STEP 3. 빌드 결과물 삭제

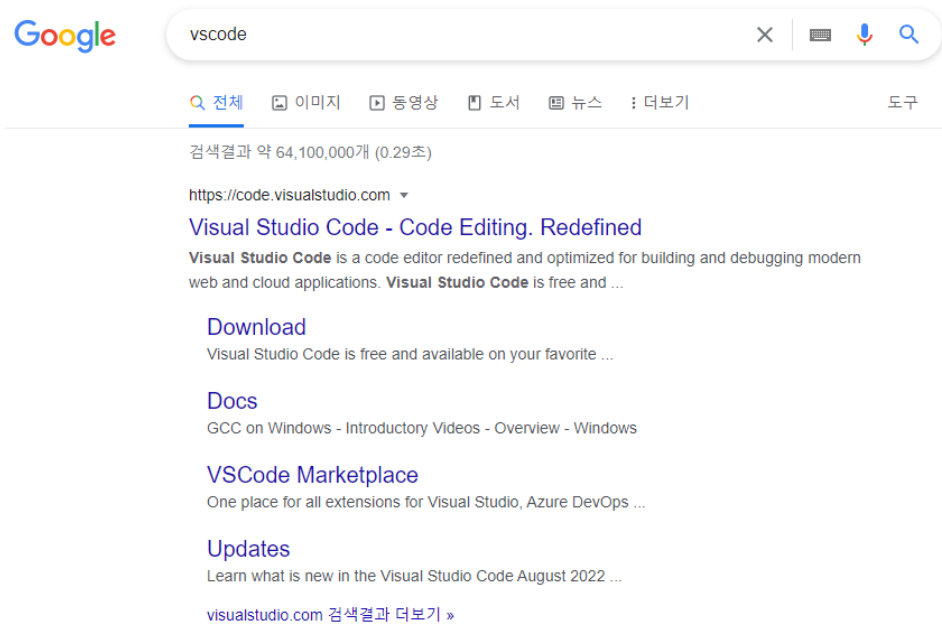
```
ing5uny@DESKTOP-79V69B3:~/compiler$ make clean
rm -rf *.tab.c *.tab.h *.yy.c wc
ing5uny@DESKTOP-79V69B3:~/compiler$ ls
Makefile wc.l wc.y
```

Clean을 활용하면 손쉽게 사용하지 않는 파일들을 간단하게 삭제할 수 있다.

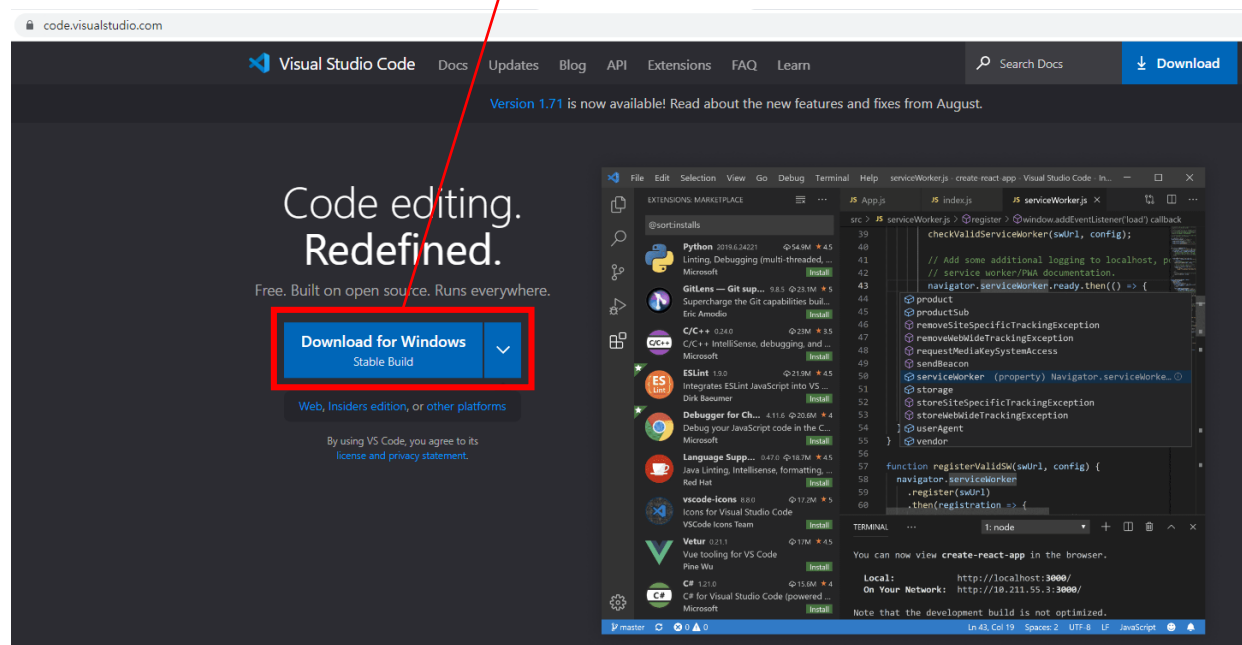
# 부록. VS Code에서 실행

## STEP 0. VS Code 설치

## 1. vs code 검색

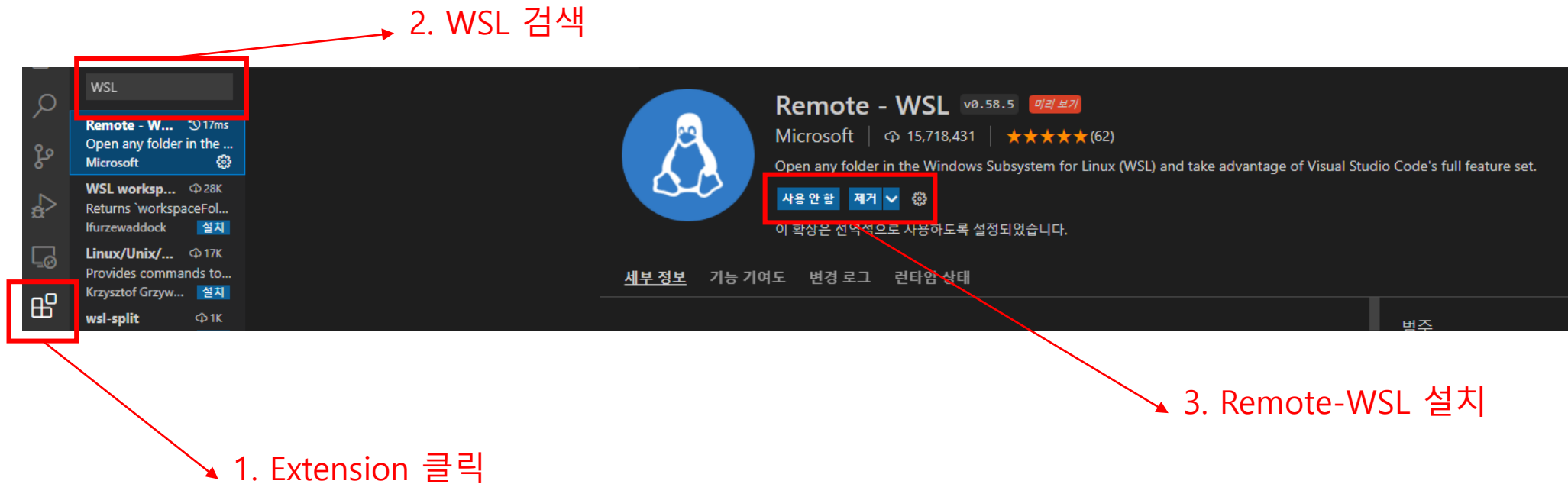


## 2. 본인의 환경에 맞게 설치



## STEP 1. VS Code의 Extension 에서 "Remote - WSL설치"

2. WSL 검색



1. Extension 클릭

3. Remote-WSL 설치

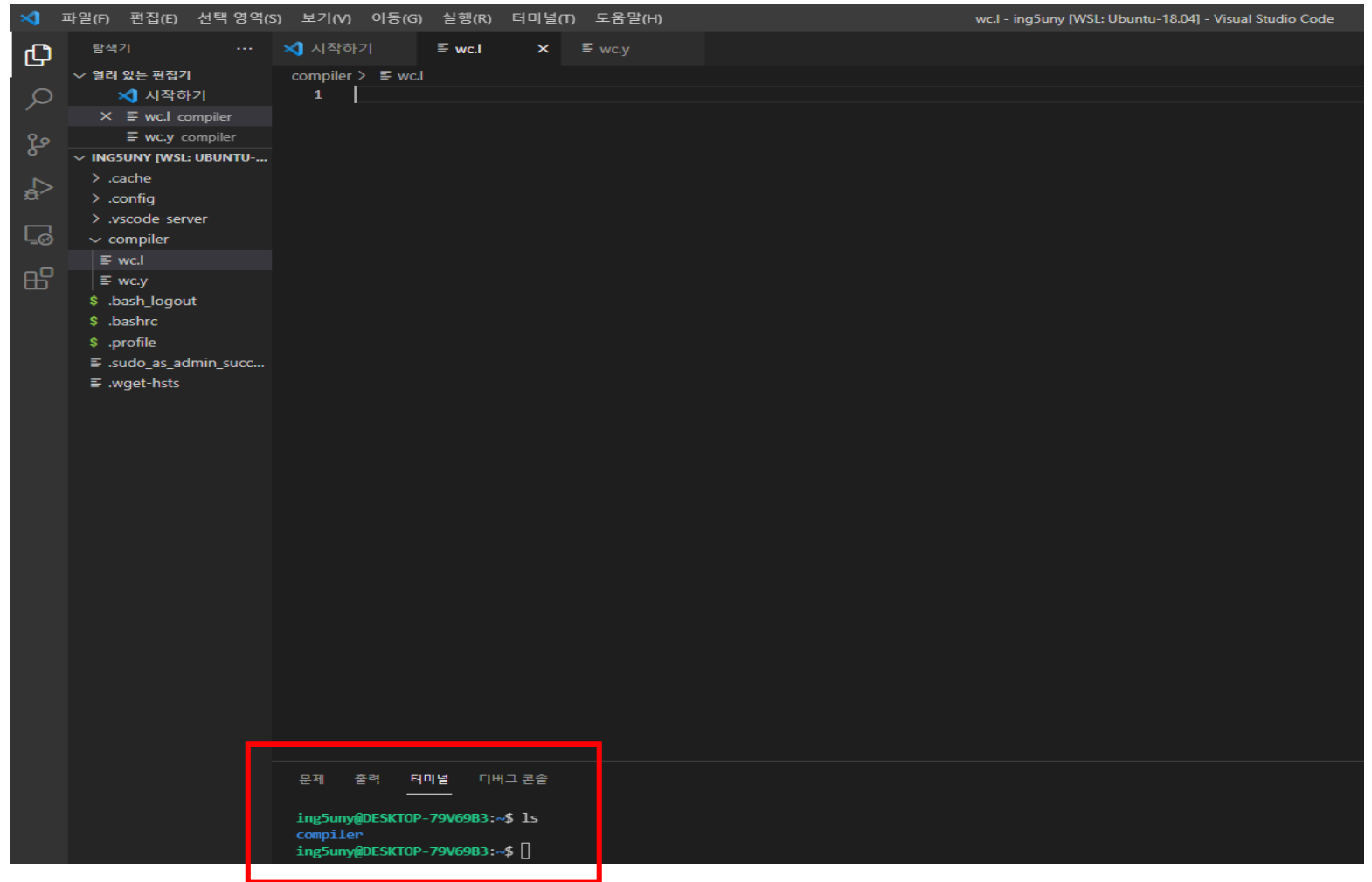
## STEP 2. WSL(ubuntu)에서 "code ."입력

```
ing5uny@DESKTOP-79V69B3: ~$ code .  
Installing VS Code Server for x64 (c13f1abb110fc756f9b3a6f16670df9cd9d4cf63)  
Downloading: 100%  
Unpacking: 100%
```

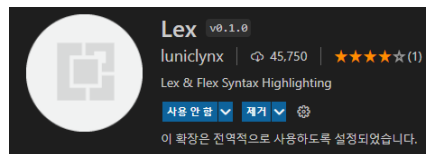


## STEP 3. VS Code 자동 실행

'터미널 - 새 터미널'을 통해  
코드 수정과 동시에  
기존 WSL에서 사용하던  
터미널처럼 사용 가능합니다.



## STEP 4. Flex를 위한 extension 설치



```
1  %{
2
3  #include "wc.tab.h"
4
5  %}
6
7  %%
8
9  [a-zA-Z]+ { yylval.nchars = strlen(yytext); return WORD; }
10 \n      { return NEWLINE; }
11 <<EOF>> { return END; }
12 .      { return ETC; }
13
14 %%
```



```
1  %{
2
3  #include "wc.tab.h"
4
5  %}
6
7  %%
8
9  [a-zA-Z]+ { yylval.nchars = strlen(yytext); return WORD; }
10 \n      { return NEWLINE; }
11 <<EOF>> { return END; }
12 .      { return ETC; }
13
14 %%
```

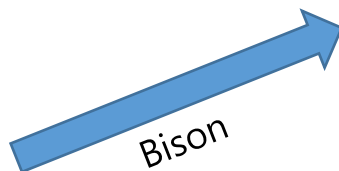
'Lex' extension을 통해 조금더 시각적인 효과를 줄 수 있습니다.

## STEP 5. Bison을 위한 extension 설치

```

1  %{
2
3  #include <stdio.h>
4
5  int yylex();
6  int yyerror(char const *);
7
8  int chars = 0;
9  int words = 0;
10 int lines = 0;
11 %}
12
13 %token END
14 %token <nchars> WORD
15 %token NEWLINE ETC
16
17 %union { int nchars; }
18
19 %%
20
21 wordlist: /* nothing */
22 | wordlist expr END { printf("lines: %d, words: %d, chars: %d\n", lines, words, chars); YYACCEPT; }
23 ;
24
25 expr: term { /* nothing */ }
26 | expr term { /* nothing */ }
27
28 term : WORD { words++; chars += $1; }
29 | NEWLINE { chars++; lines++; }
30 | ETC { chars++; }
31 ;

```



Bison

```

1  %{
2
3  #include <stdio.h>
4
5  int yylex();
6  int yyerror(char const *);
7
8  int chars = 0;
9  int words = 0;
10 int lines = 0;
11 %}
12
13 %token END
14 %token <nchars> WORD
15 %token NEWLINE ETC
16
17 %union { int nchars; }
18
19 %%
20
21 wordlist: /* nothing */
22 | wordlist expr END { printf("lines: %d, words: %d, chars: %d\n", lines, words, chars); YYACCEPT; }
23 ;
24
25 expr: term { /* nothing */ }
26 | expr term { /* nothing */ }
27
28 term : WORD { words++; chars += $1; }
29 | NEWLINE { chars++; lines++; }
30 | ETC { chars++; }
31 ;

```



VSCode-YACC

```

1  %{
2
3  #include <stdio.h>
4
5  int yylex();
6  int yyerror(char const *);
7
8  int chars = 0;
9  int words = 0;
10 int lines = 0;
11 %}
12
13 %token END
14 %token <nchars> WORD
15 %token NEWLINE ETC
16
17 %union { int nchars; }
18
19 %%
20
21 wordlist: /* nothing */
22 | wordlist expr END { printf("lines: %d, words: %d, chars: %d\n", lines, words, chars); YYACCEPT; }
23 ;
24
25 expr: term { /* nothing */ }
26 | expr term { /* nothing */ }
27
28 term : WORD { words++; chars += $1; }
29 | NEWLINE { chars++; lines++; }
30 | ETC { chars++; }
31 ;

```

## STEP N. 연동 확인

```
1  %{
2
3  #include "wc.tab.h"
4
5  %}
6
7  %%
8
9  [a-zA-Z]+ { yyval.nchars = strlen(yytext); return WORD; }
10 \n      { return NEWLINE; }
11 <<EOF>> { return END; }
12 .      { return ETC; }
13
14  %%
```

```
ing5uny@DESKTOP-79V69B3:~/compiler$ bison -d wc.y
ing5uny@DESKTOP-79V69B3:~/compiler$ ls
Makefile wc.l wc.tab.c wc.tab.h wc.y
ing5uny@DESKTOP-79V69B3:~/compiler$ flex wc.l
ing5uny@DESKTOP-79V69B3:~/compiler$ ls
Makefile lex.yy.c wc.l wc.tab.c wc.tab.h wc.y
ing5uny@DESKTOP-79V69B3:~/compiler$ gcc -o wc wc.tab.c lex.yy.c -ly -lfl
ing5uny@DESKTOP-79V69B3:~/compiler$ ls
Makefile lex.yy.c wc wc.l wc.tab.c wc.tab.h wc.y
ing5uny@DESKTOP-79V69B3:~/compiler$ ./wc
Hi compiler
Compiler is easy.
^C
ing5uny@DESKTOP-79V69B3:~/compiler$ ./wc
Hi compiler
Compiler is easy.
lines: 2, words: 5, chars: 30
```

기존 방법에서 알려준 command를 입력하여 동작을 확인합니다.

## STEP N+1. 연동 확인

```
wc0.l
wc0.l
1 /*
2  * [ONLY USING FLEX]
3  * https://www.oreilly.com/library/view/flex-bison/9780596805418/ch01.html 참고
4  * 아래 커멘드로 실행가능
5  * $ flex fb1-1.1
6  * $ cc lex.yy.c -lfl
7  * $ ./a.out
8  *
9  */
10
11
12 %{
13 int chars = 0;
14 int words = 0;
15 int lines = 0;
16 %}
17
18 %%
19
20 [a-zA-Z]+ { words++; chars += strlen(yytext); }
21 \n        { chars++; lines++; }
22 .         { chars++; }
23
24 %%
25
26 int main(int argc, char **argv)
27 {
28     yylex();
29     printf("%8d%8d%8d\n", lines, words, chars);
30 }
31
```

문제 출력 디버그 콘솔 터미널

```
ajou@DESKTOP-BQVEH82:~$ flex wc0.l
ajou@DESKTOP-BQVEH82:~$ cc lex.yy.c -lfl
ajou@DESKTOP-BQVEH82:~$ ./a.out
Hello Compiler!!
EEEEAAASSSYYYY.
      2      3      33
```

wc0.l 만을 사용하여 .l파일을 실행하기

end