2009년 06월 16일

컴파일러 Option들

잠시 우리를 이롭게 해줄 컴파일러 option들에 대해서 몇 가지 정리하고 가겠습니다. 이제까지 들여다 봤던 option이랑, compiler에 관련하여 ads를 위주로 설명했지만, 대부분의 컴파일러는 다음의 option들을 지원하고 있으니, 알아두시면 편리할 꺼라 생각하고 있습니다.

tcc (armcc)가 어떤 option들을 제공하고 있는지를 알아보기 위해 command 창에 그냥 tcc를 쳐보면 다음과 같은 내용이 후루룩 올라가지요.

C:₩apps₩ADS12₩Bin>tcc

Thumb C Compiler, ADS1.2 [Build 805]

Usage: tcc [options] file1 file2 ... filen Main options:

-c Do not link the files being compiled -C Prevent the preprocessor from removing comments (Use with -E) -D<symbol> Define <symbol> on entry to the compiler -E Preprocess the C source code only -g<options> Generate tables for high-level debugging -I<directory> Include <directory> on the #include search path -J<directory> Replace the default #include path with <directory> -o<file> Name the file that holds the final output of the compilation -O0 Minimum optimization -O1 Restricted optimization for debugging -O2 Maximum optimization -S Output assembly code instead of object code -U<symbol> Undefine <symbol> on entry to the compiler -W<options> Disable all or selected warning messages

아이 지금까지 보던 게 몇 개 있네요. tcc, armcc, gcc 뭐 이런 거 전부다 비스므리한 option들을 가져요. 하나 하나 다시 한번 살펴 본다면요,

1) -o

tcc sphagetti.c 를 해주면, 자동으로 sphagetti.o를 만들어 줍니다만, -o 를 덧붙여 주면 output 이름을 정할 수 있습니다. 예를 들어 tcc -o ramen.o sphagetti.c 라고 하면 ramen.o로 output을 만들어 줍니다. 다른 option 어떤 것들과 섞어 써도 상관없으니 output 이름을 원할 때 마다 붙여주세요.

2) -E, -C

preprocessing 해 주는 option 또는 다른 말로 Preprocessing까지만 하고 그 뒤는 하지 마라. 결국 output은 흔히 말하는 .i 가 되겠죠. -C의 용법은, 사실은 Preprocessing할 때, /- *-이나 // 등의 remark 또는 comment들이 사라지는 게 빠지게 되는데 (기계어 만드는 데 전혀 도움이 안되니까) 이걸 없애지 마라는 option입니다. 사람이 preprocessing한 결과를 보고 싶을 때 사용합니다.

3) -S

Assembly로 만들어라 하는 option 또는 다른 말로 Assembly로만 만들고 object로는 만들지 마라. 그 뒤는 하지 마라. 결국 output은 Assembly 형식의 .s가 됩니다.

4) -c

linkable한 object만 만들어라 하는 option 또는 다른 말로 link는 하지 말고 object까지만 만들어라. 그 뒤는 하지 마라. 결국 output은 elf형식의 .o가 됩니다.

5) -I, -J

header file을 include할 때 -I(dir path) 에서 찾아서 header를 include하라는 option으로서 현재 compile 하고자 하는 c file과 다른 directory에 header가 존재할 경우에 -I로 그 path를 정할 수 있습니다. 주의할 점은 -Idir 로 directory이름과 I사이에 빈칸이 없어야 해요. 그리고 보통 compiler마다 default로 include path를 잡아 놓게 되는데, 이런 default path를 내 마음대로 바꾸고 싶을 때는 -J(dir path)로 선언해 주면, 기존 default include path가 없어지고 -J option으로 준 include path가 default path가 됩니다. 즉 #include <headernaem.h>로 "" 대신 <>로 include 했을 경우에 찾아가는 directory를 -J

option으로 설정하게 됩니다.

6) -D, -U

Define option으로서, 매크로 선언인 #define과 같은 역할을 합니다. 굳이 source code안에 #define을 하지 않아도 컴파일러에게 #define을 직접 해줄 수 있는 거죠. 예를 들면 -DFEATURE_OPT 라고 하면 #define FEATURE_OPT 와 같고, -DTHREE=3 하면 #define THREE 3 이라고 한 것과 같습니다. U는 #undef과 같은 것으로 source code안에 define중에 무력화 하고 싶은 #define 즉 macro가 있는 경우에 사용할 수 있습니다. I option과 마찬가지로 D또는 U와 매크로 사이에는 빈칸이 없어야 해요. 보통은 선택적 컴파일을 위하여 이런 option을 쓰게 되는데 다음과 같은 예가 많이 쓰입니다.

```
#ifdef WINDOWS_XP
    #define KERNEL_VER 100
#elif defined (WINDOWS_NT)
    #define KERNEL_VER 201
#elif defined (LINUX)
    #define KENREL_VER 202
#endif
```

소스 코드에 #define WINDOWS_XP 또는 컴파일러 option에 -DWINDEOS_XP 라고 해주면 KERNEL_VER은 100이 되겠습니다. 여러 가지 OS를 지원하기 위한 프로그램들이 이런 식의 선택적 컴파일 소스를 제공하고 있습니다.

7) -g

Debugging 정보 즉 DWARF 정보를 ELF file에 넣어 두어라. 라는 뜻입니다. elf 형식은 실제 기계어 binary와 symbol 정보 그리고 마지막 -g option에 의한 debugging 정보까지 포함시켜, debugging을 할 수 있는 여건을 만들 수 있습니다.

8) -W

Warning Level을 결정할 수 있는 option으로서 가장 강력한 Levle은 물론 all입니다. -Wall을 option으로 주면 모든 Warning 을 compile 결과로 받아 볼 수 있으며, 이것이 좋은 코딩 습관이 됩니다.

9) -O1, -O2, ..

얼마나 컴파일 결과물을 Optimization 할 것이냐의 option입니다.

-O1은 C로 코딩 한 내용이 거의 그대로 기계어로 만들어 질 것이고, 그러므로 -g option으로 디버깅 정보를 넣는다면 거의 완벽한 디버깅을 할 수 있고 컴파일 시간이 단축된다는 장점이 있지만 다만 RO, 즉 program 영역이 memory를 더 차지 한다는 단점이 있고, -O2 역시 optimization을 -O1보다 많이 하게 하는 option이며, -O1보다는 훨씬 많이 메모리를 절약할 수 있다는 장점이 있습니다. 단점으로는 debugging 정보를 넣는다 치더라도 optimization이 많이 되어 코드와 assembly가 matching이 잘 되지 않는 단점이 있습니다. 머리만 좋으면 뭐가 문제겠습니까만! 이밖에 또 이용되는 option들이 좀 있는데 더 소개해 보겠습니다. 라이브러리 관련된 option인데요. linker한테 통하는 거겠죠.

10) -l

-l라이브러리이름을 주게 되면 특정한 library를 link할 때 참고해서 link하라는 의미 입니다.

11) -L

앞에서 Include Directory이름을 따로 줄 수 있듯이, Library도 기본적으로 searching할 Directory이름을 option으로 줄 수 있어요. -L뒤에 곧바로 붙여서 Directory이름을 주면

Library link를 할 때 -L뒤에 option으로 준 Directory에서 부터 라이브러리를 찾아 들어 갑니당

 컴파일러 최적화 option에 대한 단상.

Speed최적화를 할 것인지, Space최적화를 할 것인지는 -Ospace, -Ospeed 옵션을 사용한다고 했는데요, 자기 Software 최적화 한다고 머리 쓰면서 이상하게 짜지 말고 요즘 컴파일러는 최적화를 잘 수행하기 때문에, 이런 부분은 컴파일러에 맡기는 것이 현명하다는 것이 대세라니까요

컴파일러, compiler, Hardware, compile, Assembly, lib, arm

Linked at at 2009/06/16 20:11

... nbsp; ⓖ Library를 만들자 - 남한테 보여주기 싫어 ⓗ Lib을 까보자 ⓘ 컴파일러 option들 ⓙ 변수의 scope와 그 생애 (Memory Map) ⓚ Memory Ma ... more