일반

[CMake 튜토리얼] 3. CMakeLists.txt 기본 패턴

Posted **2017. 02. 26** *Updated* **2019. 02. 11** *Views* **11278** *Replies* **4**

ABOUT

ARTICLES
- 일반 (38)
- 건강 (9)
- 여행 (91)
- 독서 (5)
- 영화 (3)
- 박람회 (5)

ECE
- 일반 (60)
- PSpice (6)
- AVR (32)
- Android (8)
- Nginx (2)
- Apache (9)
- Linux (49)
- XE (11)
- Python (11)
- Security (3)

SHOWCASE

GUESTBOOK

▶ 이 글에서는 복붙으로 바로 활용할 수 있는 CMakeLists.txt 빌드 스크립트의 기본 패턴을 제시합니다. 빌드 결과물이 실행 바이너리 1개인 C 프로젝트를 관리하는 CMake 빌드 스크립트이며, **소스 파일 목록과 빌드 형상(Configuration)별 컴파일·링크 플래그, 링크 라이브러리**를 관리할 수 있습니다. 여기서 제시하는 패턴을 기본으로 이전 글을 참조하여 추가적으로 필요한 기능을 더해서 사용하시면 됩니다.

다음은 CMake 빌드 스크립트의 기본 패턴입니다. **<...>**로 표시한 부분만 적절히 수정하면 C 프로젝트 빌드 스크립트로 활용할 수 있습니다.

CMakeLists.txt

```
# 요구 CMake 최소 버전
CMAKE_MINIMUM_REQUIRED ( VERSION <버전> )

# 프로젝트 이름 및 버전
PROJECT ( "<프로젝트_이름>" )
SET ( PROJECT_VERSION_MAJOR <주_버전> )
SET ( PROJECT_VERSION_MINOR <부_버전> )


```

```
# 빌드 형상(Configuration) 및 주절주절 Makefile 생성 여부
SET ( CMAKE_BUILD_TYPE <Debug|Release> )
SET ( CMAKE_VERBOSE_MAKEFILE <true|false> )

# 빌드 대상 바이너리 파일명 및 소스파일 목록
SET ( OUTPUT_ELF
        "${CMAKE_PROJECT_NAME}-${PROJECT_VERSION_MAJOR}.${PROJECT_VERSION_MINOR}.out"
        )
SET ( SRC_FILES
        <소스_파일>
        <소스_파일>
        ...
        )

# 공통 컴파일러
SET ( CMAKE_C_COMPILER "<컴파일러>" )

# 공통 헤더 파일 Include 디렉토리 (-I)
INCLUDE_DIRECTORIES ( <디렉토리> <디렉토리> ... )

# 공통 컴파일 옵션, 링크 옵션
ADD_COMPILE_OPTIONS ( <컴파일_옵션> <컴파일_옵션> ... )
SET ( CMAKE_EXE_LINKER_FLAGS "<링크_옵션> <링크_옵션> ..." )

# 공통 링크 라이브러리 (-l)
LINK_LIBRARIES( <라이브러리> <라이브러리> ... )

# 공통 링크 라이브러리 디렉토리 (-L)
LINK_DIRECTORIES ( <디렉토리> <디렉토리> ... )

# "Debug" 형상 한정 컴파일 옵션, 링크 옵션
SET ( CMAKE_C_FLAGS_DEBUG "<컴파일_옵션> <컴파일_옵션> ..." )
SET ( CMAKE_EXE_LINKER_FLAGS_DEBUG "<링크_옵션> <링크_옵션> ..." )

# "Release" 형상 한정 컴파일 옵션, 링크 옵션
SET ( CMAKE_C_FLAGS_RELEASE "<컴파일_옵션> <컴파일_옵션> ..." )
SET ( CMAKE_EXE_LINKER_FLAGS_RELEASE "<링크_옵션> <링크_옵션> ..." )

# 출력 디렉토리
SET ( CMAKE_RUNTIME_OUTPUT_DIRECTORY ${CMAKE_BUILD_TYPE} )
SET ( CMAKE_LIBRARY_OUTPUT_DIRECTORY ${CMAKE_BUILD_TYPE}/lib )
SET ( CMAKE_ARCHIVE_OUTPUT_DIRECTORY ${CMAKE_BUILD_TYPE}/lib )

# 빌드 대상 바이너리 추가
ADD_EXECUTABLE( ${OUTPUT_ELF} ${SRC_FILES} )
```

예) 다음 빌드 스크립트는 main.c, foo.c, bar.c 세 개의 파일로 구성된 C 프로젝트를 빌드하기 위한 CMake 빌드 스크립트입니다.

CMakeLists.txt

```
# 요구 CMake 최소 버전
CMAKE_MINIMUM_REQUIRED ( VERSION 2.8 )

# 프로젝트 이름 및 버전
PROJECT ( "andromeda" )
SET ( PROJECT_VERSION_MAJOR 0 )
SET ( PROJECT_VERSION_MINOR 1 )

# 빌드 형상(Configuration) 및 주절주절 Makefile 생성 여부
SET ( CMAKE_BUILD_TYPE Debug )
SET ( CMAKE_VERBOSE_MAKEFILE true )

# 빌드 대상 바이너리 파일명 및 소스 파일 목록
SET ( OUTPUT_ELF
        "${CMAKE_PROJECT_NAME}-${PROJECT_VERSION_MAJOR}.${PROJECT_VERSION_MINOR}.out"
        )
SET ( SRC_FILES
        bar.c
        foo.c
        main.c
        )

# 공통 컴파일러
SET ( CMAKE_C_COMPILER "gcc" )

# 공통 헤더 파일 Include 디렉토리 (-I)
INCLUDE_DIRECTORIES ( include driver/include )

# 공통 컴파일 옵션, 링크 옵션
ADD_COMPILE_OPTIONS ( -g -Wall )
SET ( CMAKE_EXE_LINKER_FLAGS "-static -Wl,--gc-sections" )

# 공통 링크 라이브러리 (-l)
LINK_LIBRARIES( uart andromeda )

# 공통 링크 라이브러리 디렉토리 (-L)
LINK_DIRECTORIES ( /usr/lib )

# "Debug" 형상 한정 컴파일 옵션, 링크 옵션
SET ( CMAKE_C_FLAGS_DEBUG "-DDEBUG -DC_FLAGS" )
SET ( CMAKE_EXE_LINKER_FLAGS_DEBUG "-DDEBUG -DLINKER_FLAGS" )

# "Release" 형상 한정 컴파일 옵션, 링크 옵션
SET ( CMAKE_C_FLAGS_RELEASE "-DRELEASE -DC_FLAGS" )
SET ( CMAKE_EXE_LINKER_FLAGS_RELEASE "-DRELEASE -DLINKER_FLAGS" )

# 출력 디렉토리
```

```
53  SET ( CMAKE_RUNTIME_OUTPUT_DIRECTORY ${CMAKE_BUILD_TYP
    E} )
    SET ( CMAKE_LIBRARY_OUTPUT_DIRECTORY ${CMAKE_BUILD_TYP
    E}/lib )
    SET ( CMAKE_ARCHIVE_OUTPUT_DIRECTORY ${CMAKE_BUILD_TYP
    E}/lib )

    # 빌드 대상 바이너리 추가
    ADD_EXECUTABLE( ${OUTPUT_ELF} ${SRC_FILES} )
```

※ 소스 파일 목록(SRC_FILES) 관리에 대하여

이 글에서 제시한 패턴에서는 빌드 대상 소스 파일을 **SRC_FILES**변수에 모두 일일이 나열하도록 작성되어 있습니다. 프로젝트를 진행하면서 소스 파일을 추가하거나 삭제할 일이 있을 때마다 이 목록을 수정해 나가면 됩니다. 뭔가 좀 깔끔하지 못한(?)것 같은 기분이 들긴 하지만, ~~그냥 기분탓이니까~~ 다소 불편하더라도 소스 파일 목록은 이렇게 수동으로 관리하는 것이 옳습니다.

CMake에서 제공하는 Generator Expression을 활용하면 단 한 줄로 특정 확장자의 파일들을 일괄적으로 찾아서 목록에 넣을 수 있습니다. 이렇게 빌드 스크립트를 작성하면 더욱 깔끔한 것 같다는 착각이 들 수 있지만, 실상은 그 반대입니다. 그 이유는 Visual Studio와 같은 IDE에서 왜 같은 프로젝트 디렉토리에 있는 소스 파일을 자동으로 탐지해서 프로젝트에 포함시키지 않는지에 대해 잘 생각해 보면 알 수 있습니다.

디버깅을 할 때 오류의 원인을 찾기 위해 특정 소스파일을 빌드 대상에서 제외하고 시험 빌드를 수행하는 경우가 있습니다. 혹은, 같은 프로젝트 디렉토리 내에 서로 다른 버전의 써드파티 소스코드를 위치시키고 필요에 따라 바꿔 가며 빌드를 시도하는 경우가 있습니다. 소스 파일 목록을 모두 나열해 놓은 경우 **빌드에서 제외할 소스 파일만 주석 처리하고 빌드를 시도하면 끝**입니다. 반면, 소스 파일 일괄 추가 방식으로 빌드 스크립트를 작성했다면... 머리에 쥐가 나기 시작할 것입니다. 최악의 경우 결국 소스 파일 목록을 작성해야 할 수도 있습니다.

그러니까, 이런 불상사를 막기 위해 프로젝트 시작 시점부터 소스 파일 목록은 수동으로 관리하도록 해야 합니다. 파일을 첨삭할 때 알파벳 순서로 정렬해 놓으면 나중에 찾을 때 보다 수월합니다. 프로젝트 규모가 점점 방대해져서 소스 파일 갯수가 관리하기 힘들 정도로 많아지면, 적절한 기준에 따라 소스 파일 목록을 나누고 각 목록별로 라이브러리를 작성하도록 한 뒤, 이들 라이브러리를 모아서 최종 실행 바이너리를 작성하도록 빌드 스크립트를 구성하면 됩니다. 튜토리얼에서는 다루지 않았지만, CMake 빌드 스크립트를 여러개로 쪼개서 계층화시켜 관리하는 기법도 생각해 볼 수 있습니다.

예전에 모 오픈소스 임베디드 펌웨어의 소스 코드를 받았는데, Makefile에다가 소위 "Recursion Magic"이라고 해서 모든 디렉토리를 재귀적으로 뒤져서 소스 파일을 자동으로 찾아 빌드하도록 해놓아서 기겁을 했던 적이 있습니다.
- 빌드 스크립트를 그렇게 깔끔하게(?) 만들어서 배포하면 열어 보는 사람이 "우와! 신기하다 @.@" 이럴 것 같죠? 절대 그렇지 않습니다. 빌드 절차가 어떻게 구성되는지 파악조차 하기 어렵고, 대체 어떤 소스 파일들이 빌드 대상인지도 알 수가 없으며, 프로젝트를 필요에 따라 수정하면서 디버깅하기 매우 난감하기 때문에 (들리지는 않겠지만) 욕만 바가지로 얻어먹을 것입니다.

결국 그걸 다 일일이 분석하고 CMake 빌드 스크립트로 다시 작성하면서 수도 없이 ... ~~이거 만든넘 Shi발 Shi벌^^~~ 했었다는 후문입니다. 오픈소스인지라 뭐 이런 불만을 표출할 수도 없고, 차라리 눈에 안띄었으면 나았는데 그것도 아니고 그랬으니 말이죠. (헌데 그때 한 삽질에서 쌓은 내공 덕문에 이 튜토리얼을 쓰고 있는지도 모르겠네요.ㅎㅎ)

Makefile과 마찬가지로 CMakeLists.txt도 처음 만들어놓고 팽개쳐놓는 게 아닌, 프로젝트를 진행하면서 점진적으로 관리해야 할 대상으로 여겨야 합니다. 그 관리 대상 중 대표적인 것이 바로 **소스 파일 목록**입니다.

프로젝트가 일단 정궤도에 오르면 Makefile과 달리 의존성을 일일이 나열할 필요 없이 파일 목록만 첨삭하면 되므로 훠얼~씬 간편하기도 하니 말이죠.ㅎㅎ

연관글

[CMake 튜토리얼] 2. CMakeLists.txt 주요 명령과 변수 정리 (29178) *1
[CMake 튜토리얼] 1. CMake 소개와 예제, 내부 동작 원리 (30065)
[Make 튜토리얼] Makefile 예제와 작성 방법 및 기본 패턴 (29680) *2

TAG • Make, Makefile, CMake, CMakeLists.txt

‹ **Prev** [Windows] 다중 NIC(LAN카드) 환경에서 Routing Table 설정 - rou...

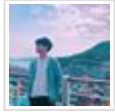
[CMake 튜토리얼] 2. CMakeLists.txt 주요 명령과 변수 정리 **Next** ›

Comment **'4'**

권오용 권오용 2019.02.11 05:00:54

SET (SRC_FILES ...> 문법으로 소스코드 추가시 소스코드가 100개가 넘어갈경우에 한번에 지정할수 있는 방식도 있나요?

댓글 **1**



Jinsoo Heo Jinsoo Heo 2019.02.10 17:46:39

정리가 너무 잘되어 있어서 보기 좋았습니다. 정말 큰 도움됐습니다. 감사합니다.

댓글 **1**

번호	분류	제목	글쓴이	최근 수정일	조회 수
191	일반	[WSL] Windows Subsystem for Linux - 디스플레이 서버 설정 및 GUI 사용하기	TUW	2019.01.15	2348
190	일반	[TeraTerm] 명령줄 인수와 공개키 인증으로 간편하게 SSH 접속하기	TUW	2018.11.06	770
189	일반	[WSL] Windows Subsystem for Linux - SSH 서버 자동 시작 설정하기	TUW	2018.11.06	1563
188	일반	[WSL] Windows Subsystem for Linux - SSH 서버 세팅하기	TUW	2018.11.09	2239
187	일반	[WSL] Windows Subsystem for Linux - Bash.exe를 Ubuntu와 유사하게 설정하기	TUW	2018.11.06	1369
186	일반	[WSL] Windows Subsystem for Linux - 초기 설치와 Ubuntu 배포판 설치	TUW	2018.11.06	2345
185	일반	[AutoHotkey] 단축키(Hotkey) 스크립트 작성과 자동 시작 등록	TUW	2018.11.08	1524
184	일반	[AutoHotkey] 소개와 설치 및 기본 설정 - GUI 예시, 기본 에디터 변경	TUW	2018.11.11	1355
183	일반	Windows에서 포트 포워딩(Port Forwarding) 설정하기 - Netsh	TUW	2018.02.03	8105
182	Security	[SSL/HTTPS] Let's Encrypt 무료 SSL 인증서 발급/설치/관리 - certbot 사용법	TUW	2019.02.25	11191
181	Security	[SSL/HTTPS] StartSSL/StartCom 사태와 Let's Encrypt로의 이전 **1**	TUW	2018.05.02	3414
180	Linux	[Ubuntu] Windows와 멀티부팅 환경에서 시간이 맞지 않는 현상 해결하기	TUW	2017.06.08	7016
179	일반	[Windows] 다중 NIC(LAN카드) 환경에서 Routing Table 설정 - route 명령	TUW	2018.06.13	15883
»	일반	[CMake 튜토리얼] 3. CMakeLists.txt 기본 패턴 **4**	TUW	2019.02.11	11278

번호	분류	제목	글쓴이	최근 수정일	조회 수
177	일반	[CMake 튜토리얼] 2. CMakeLists.txt 주요 명령과 변수 정리 1	TUW	2018.05.31	29178
176	일반	[CMake 튜토리얼] 1. CMake 소개와 예제, 내부 동작 원리	TUW	2018.06.13	30065

목록 검색

‹ Prev 1 2 3 4 5 6 7 ... 12 Next ›