01. GCC Compiler

국민대학교 소프트웨어학부

GCC Compiler (1)

- GNU Compiler: gcc
  - gcc는 GNU 에서 만든 컴파일러
  - 가장 대중적인 범용 컴파일러
    - 많은 옵션 및 기능을 제공하며, 다양한 CPU 아키텍처를 제공
    - 제공하는 CPU 아키텍처
      - ARM, DEC, AVR, i386, PPC, SPARC, M68XX 등등
      - Cross-Compile 을 지원함
    - 지원하는 언어
      - C, C++, Fortran, ada, Objective_C
  - 참조:
    - https://en.Wikipedia.org/wiki/GNU_Compiler_Collection
    - https://gcc.gnu.org/

- g++
  - 일반적으로 C++ 컴파일러로 지칭하나,
  - 사실상 “cc1plus” 가 실질적인 컴파일러

GCC Compiler (2)

- gcc 는 C 만을 위한 컴파일러가 아님
- gcc 로 C++ 컴파일이 가능하며
  - gcc 와 g++의 구분은
    - collect_gcc 에 의한 명시적 구분과
    - 컴파일을 위해 사용하는 라이브러리(C 혹은 C++)에 의해 구분됨
  - 마지막에 차이점에 대하여 간략히 소개함
- **본 과정은 g++ 위주로 설명함**

GCC Compiler

GCC Compiler 설치

- Ubuntu 16.04 기준
  - apt 저장소 업데이트 및 업그레이드 (옵션)
    - 저장소 변경 [link]

    $ sudo apt update

    $ sudo apt upgrade
  - 개발 도구 설치

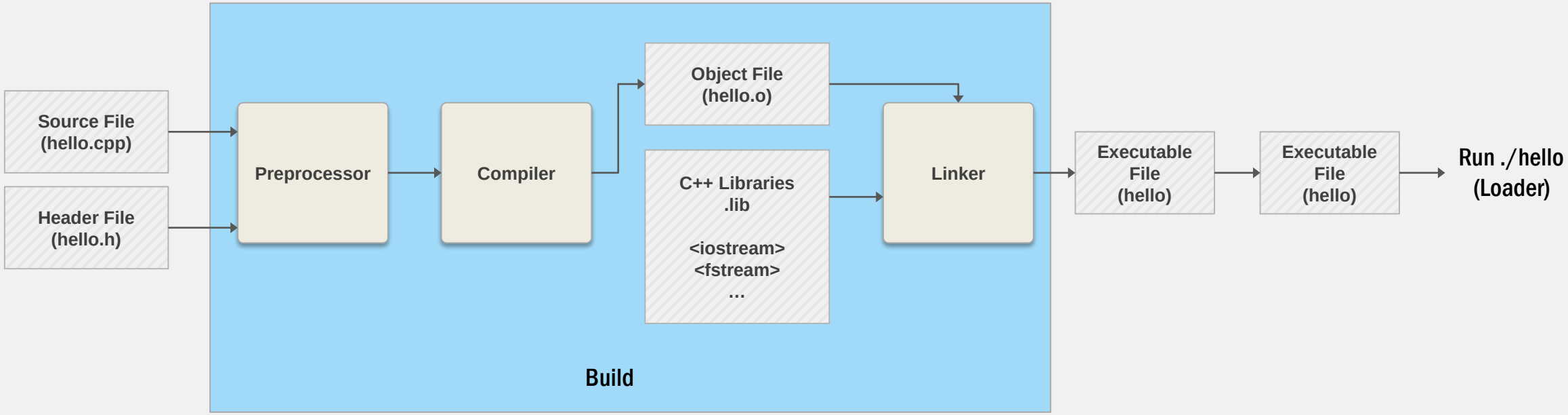
    $ sudo apt install build-essential

GCC Compiler 버전 확인

```
kmucs@localhost:~/cpp$ g++ --version
g++ (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609
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.
```

The C++ Compilation Process

- C++ 컴파일 과정
  - Overview



- The C++ compilation process -

The C++ Compilation Process (1)

- C++ 컴파일 과정 (1)
  - 예제: hello.cpp

```
#include <iostream>

int main(int argc, char *argv[])
{
     std::cout << "Hello World" << std::endl;

     return 0;
}
```

  - 표현식

```
g++ <filename.cpp>
```

  - 예제

```
kmucs@localhost:~/cpp$ g++ hello.cpp
kmucs@localhost:~/cpp$ ls
a.out  hello.cpp
```

  - a.out 생성됨

The C++ Compilation Process (2)

- C++ 컴파일 과정 (2)
  - 실행

```
kmucs@localhost:~/cpp$ ./a.out
Hello World
```

- 상기 컴파일 과정의 문제점
  - 단일 파일인 경우를 제외한 다수의 파일인 경우 모든 실행 파일의 이름이 동일하게 됨

- 해결책
  - 파일이름과 동일한 오브젝트 파일의 이름으로 컴파일을 수행한다.

```
g++ -c <filename.cpp>
```

  - 예제

```
kmucs@localhost:~/cpp$ g++ -c hello.cpp
kmucs@localhost:~/cpp$ ls
hello.cpp  hello.o
```

The C++ Compilation Process (3)

- C++ 컴파일 과정 (3)
  - “-c” 옵션

```
g++ -c <filename.cpp>
```

  - hello.cpp 를 컴파일 한다.
  - Link 는 수행하지 않는다

- C++ 링킹 과정
  - 표현식

```
g++ -o <Executable file name> <object file name.o>
```

  - 예제

```
kmucs@localhost:~/cpp$ g++ -o hello hello.o
kmucs@localhost:~/cpp$ ls
hello  hello.cpp  hello.o
```

The C++ Compilation Process (4)

- 실행

```
kmucs@localhost:~/cpp$ ./hello
Hello World
```

The C++ Build (1)

- C++ 빌드 과정 (1)
  - 출력

--save-temps: 컴파일 과정 중의 생산물을 저장

```
kmucs@localhost:~/cpp$ g++ -v --save-temps -o hello hello.cpp
```

-v: 컴파일 과정을 화면에 출력

    - 전처리 과정: cc1plus -E

```
…
gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.4)
COLLECT_GCC_OPTIONS='-v' '-save-temps' '-o' 'hello' '-shared-libgcc' '-
mtune=generic' '-march=x86-64'
 /usr/lib/gcc/x86_64-linux-gnu/5/cc1plus -E -quiet -v -imultiarch x86_64-
linux-gnu -D_GNU_SOURCE hello.cpp -mtune=generic -march=x86-64 -fpch-
preprocess -fstack-protector-strong -Wformat -Wformat-security -o hello.ii
…
```

    - 컴파일 과정: cc1plus -fpreprocessed

```
…
COLLECT_GCC_OPTIONS='-v' '-save-temps' '-o' 'hello' '-shared-libgcc' '-
mtune=generic' '-march=x86-64'
 /usr/lib/gcc/x86_64-linux-gnu/5/cc1plus -fpreprocessed hello.ii -quiet -
dumpbase hello.cpp -mtune=generic -march=x86-64 -auxbase hello -version -
fstack-protector-strong -Wformat -Wformat-security -o hello.s
…
```

The C++ Build (2)

- C++ 빌드 과정 (2)
  - 어셈블 과정: as

```
…
     compiled by GNU C version 5.4.0 20160609, GMP version 6.1.0, MPFR
version 3.1.4, MPC version 1.0.3
GGC heuristics: --param ggc-min-expand=100 --param ggc-min-heapsize=131072
Compiler executable checksum: c3fdb80f2154421ceaf9e22c85325a8d
COLLECT_GCC_OPTIONS='-v' '-save-temps' '-o' 'hello' '-shared-libgcc' '-
mtune=generic' '-march=x86-64'
 as -v --64 -o hello.o hello.s
…
```

  - 링킹 과정: collect2
    - 다음 페이지

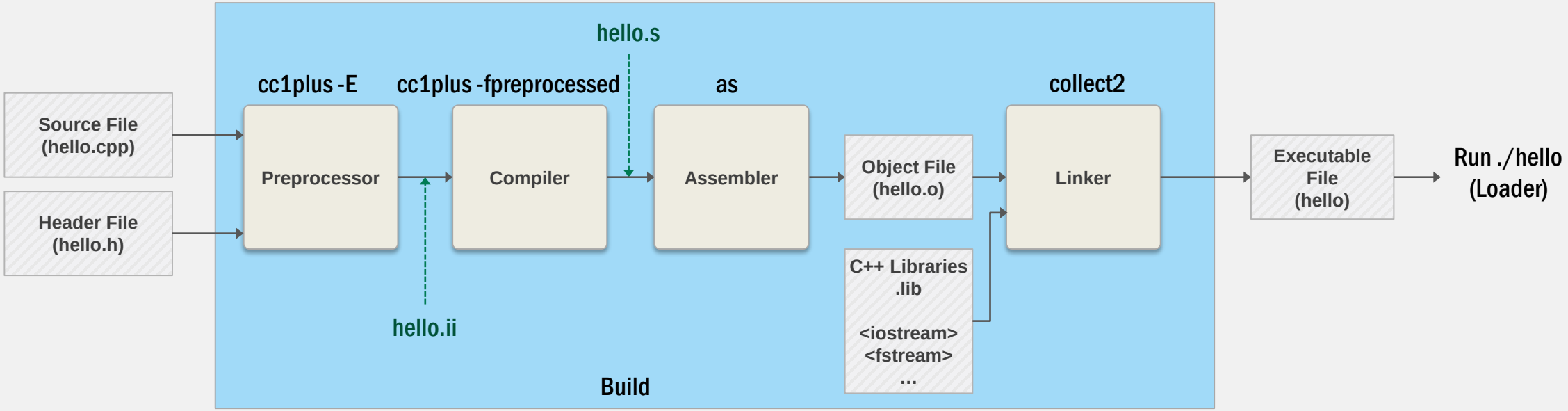
The C++ Build (3)

- C++ 빌드 과정 (3)
  - 링킹 과정: collect2

```
…
/usr/lib/gcc/x86_64-linux-gnu/5          -plugin /usr/lib/gcc/x86_64-
linux-gnu/5/liblto_plugin.so -plugin-opt=/usr/lib/gcc/x86_64-linux-
gnu/5/lto-wrapper -plugin-opt=-fresolution=hello.res -plugin-opt=-pass-
through=-lgcc_s -plugin-opt=-pass-through=-lgcc -plugin-opt=-pass-
through=-lc -plugin-opt=-pass-through=-lgcc_s -plugin-opt=-pass-through=-
lgcc --sysroot=/ --build-id --eh-frame-hdr -m elf_x86_64 --hash-style=gnu
--as-needed -dynamic-linker /lib64/ld-linux-x86-64.so.2 -z relro -o hello
/usr/lib/gcc/x86_64-linux-gnu/5/../../../x86_64-linux-gnu/crt1.o
/usr/lib/gcc/x86_64-linux-gnu/5/../../../x86_64-linux-gnu/crti.o
/usr/lib/gcc/x86_64-linux-gnu/5/crtbegin.o -L/usr/lib/gcc/x86_64-linux-
gnu/5 -L/usr/lib/gcc/x86_64-linux-gnu/5/../../../x86_64-linux-gnu -
L/usr/lib/gcc/x86_64-linux-gnu/5/../../../../lib -L/lib/x86_64-linux-gnu -
L/lib/../lib -L/usr/lib/x86_64-linux-gnu -L/usr/lib/../lib -
L/usr/lib/gcc/x86_64-linux-gnu/5/../../.. hello.o -lstdc++ -lm -lgcc_s -
lgcc -lc -lgcc_s -lgcc /usr/lib/gcc/x86_64-linux-gnu/5/crtend.o
/usr/lib/gcc/x86_64-linux-gnu/5/../../../x86_64-linux-gnu/crtn.o
…
```

The C++ Build (4)

- C++ 빌드 과정



- The C++ compilation process -

– 생산된 중간 파일들

```
kmucs@localhost:~/cpp$ ls
hello  hello.cpp  hello.ii  hello.o  hello.s
```

The C++ Compile using gcc

- gcc 를 이용한 컴파일

```
kmucs@localhost:~/cpp$ gcc -v --save-temps -o hello hello.cpp -lstdc++
```

  - 옵션 –l<libname> :
    - 링킹 시에 제공된 라이브러리<libname> 에서 명시된 이름들을 찾을 수 있도록 함.
    - -lstdc++
      - 표준 C++ 라이브러리를 사용함을 컴파일러에게 알려줌
  - 출력

```
…
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/5/lto-wrapper
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 5.4.0-
6ubuntu1~16.04.4' --with-bugurl=file:///usr/share/doc/gcc-5/README.Bugs --
enable-languages=c,ada,c++,java,go,d,fortran,objc,obj-c++ --prefix=/usr --
program-suffix=-5 --enable-shared --enable-linker-build-id --
libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --
libdir=/usr/lib --enable-nls --with-sysroot=/ --enable-clocale=gnu --
enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-
libstdcxx-abi=new --enable-gnu-unique-object
…
```

The C++ Compile using g++ (1)

- g++ 를 이용한 컴파일
  - 출력

```
kmucs@localhost:~/cpp$ g++ -v --save-temps -o hello hello.cpp
```

```
…
COLLECT_GCC=g++
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/5/lto-wrapper
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 5.4.0-
6ubuntu1~16.04.4' --with-bugurl=file:///usr/share/doc/gcc-5/README.Bugs --
enable-languages=c,ada,c++,java,go,d,fortran,objc,obj-c++ --prefix=/usr --
program-suffix=-5 --enable-shared --enable-linker-build-id --
libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --
libdir=/usr/lib --enable-nls --with-sysroot=/ --enable-clocale=gnu --
enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-
libstdcxx-abi=new
…
```

- 즉 gcc 로도 C++ 컴파일이 가능함
  - 단, 관련 라이브러리들이 자동으로 링크되지 않으므로 옵션을 사용하여 컴파일을 수행하여야 함
  - **g++ 사용을 권장**

The C++ Compile using g++ (2)

- 가장 일반적인 g++ 사용법

```
kmucs@localhost:~/cpp$ g++  -Wall -W -o hello hello.cpp
```

  - 옵션 "- Wall -W"
    - 컴파일 시에 경고메시지를 출력
  - 예

```
kmucs@localhost:~/cpp$ g++ -Wall -W -o hello hello.cpp
hello.cpp:3:14: warning: unused parameter 'argc' [-Wunused-parameter]
 int main(int argc, char *argv[])
              ^
hello.cpp:3:31: warning: unused parameter 'argv' [-Wunused-parameter]
 int main(int argc, char *argv[])
```

  - 경고메시지(unused parameter …)
    - 선언 이후 한번도 사용되지 않았다는 경고 메시지

- g++ 의 옵션은 매우 많으며, 자세한 내용은 필요 시 다루기로 함
  - 참조: man g++

- 디버깅 관련 방법은 추후에 다룸

```
kmucs@localhost:~/cpp$ man g++
```