Programming Environment

[CS 225] Advanced C/C++

IMPORTANT NOTE: This is the document that you consult for the compiler flags for ALL assignments.

1. Overview

In this course we are focusing on the C++ language in the standard version C++17, with some features originating from the C programming language.

From the trimester Fall 2020 onwards, all assignments in the course are graded automatically using Linux-based GNU g++ compilation environment supported by the new version of Moodle that is available for the course. All assignments in this course are platform-agnostic and – unlike in previous runs of the course – do not require Microsoft Windows 10 to compile.

Compatibility of code with Visual Studio VC++ or clang++ compilers will not be included as a part of the submission evaluation process, but it is important for students to learn how to use GNU g++, Microsoft Visual Studio C++, and clang++ compilers, and test created code before submission.

Most of basic configuration flags for clang++ match g++, as it has been designed as an in-place replacement for g++. Because of this, their flags are presented together.

After completing the [CS170] "*High-Level Programming II: The C++ Programming Language*" course, it is assumed that you are already familiar with *makefile* scripts and *Doxygen* documentation. If you do not feel comfortable with these tools, you may want to refresh your knowledge in these areas.

2. Compilers

2.1. GNU gcc and clang

2.1.1. Compilation with linking

To compile and link the C programming language code with gcc (9.3.0), use the following commands:

```
gcc -std=c99 -pedantic -Wall -Wextra -Wconverison –Werror
-o output.exe source.c
```

```
clang -std=c99 -pedantic -Wall –Wextra -Wconversion –Werror
-o output.exe source.c
```

Explanation of the options used:

- `-std=c99` – Ensure ISO C99 standards mode.
- `-pedantic` – Issue all the warnings demanded by strict ISO C; reject all programs that use forbidden extensions, and some other programs that do not follow ISO C.
- `-Wall` – Display all warnings for many common errors; this flag combines a large number of other, more specific warning options, which can be selected individually. Always use this flag.
- `-Wextra` – Display some extra warnings not enabled by `-Wall`. Always use this flag.
- `-Wconversion` – Display some extra warnings related to data type conversions that are not enabled by neither `-Wall` nor `-Wextra`. Always use this flag.
- `-Werror` – Treat all warnings as errors; compilation with warnings *will* fail. Always use this flag.
- `-o filename` – Produce an output executable file with a given `filename`.

2.1.2. Compilation without linking

To compile the code without linking, use the following commands:

```
gcc -std=c99 -pedantic -Wall -Wextra -Wconverison –Werror
-o source.o –c source.c
```

```
clang -std=c99 -pedantic -Wall -Wextra -Wconverison –Werror
-o source.o –c source.c
```

Explanation of the additional options used:

- `-c` – Compile only; do not link.

2.1.3. Linking

To link compiled object code, use the following commands:

```
gcc –o output.exe source.o
```

```
clang –o output.exe source.o
```

2.1.4. Getting help

To see the available flags and help options for gcc and clang compilers, type the following commands into the command prompt of an environment supporting the `man` manual pages[1] [2]:

```
man gcc
```

```
man clang
```

[1] https://man7.org/linux/man-pages/man1/gcc.1.html
[2] https://clang.llvm.org/docs/CommandGuide/clang.html

2.2. GNU g++ and clang++

2.2.1. Compilation with linking

To compile and link the C++ programming language code with g++ (7.4.0) or clang++ (8.0.0), use the following commands:

```
g++ -std=c++17 -pedantic -Wall -Wextra –Wconversion –Werror
-o output.exe source.cpp
```

```
clang++ -std=c++17 -pedantic -Wall -Wextra –Wconversion –Werror
-o output.exe source.cpp
```

Explanation of the options used:

- `-std=c++17` – Ensure ISO C++17 standards mode.
- `-pedantic` – Issue all the warnings demanded by strict ISO C and ISO C++; reject all programs that use forbidden extensions, and some other programs that do not follow ISO C and ISO C++.
- `-Wall` – Display all warnings for many common errors; this flag combines a large number of other, more specific warning options, which can be selected individually. Always use this flag.
- `-Wextra` – Display some extra warnings not enabled by `-Wall`. Always use this flag.
- `-Wconversion` – Display some extra warnings related to data type conversions that are not enabled by neither `-Wall` nor `-Wextra`. Always use this flag.
- `-Werror` – Treat all warnings as errors; compilation with warnings *will* fail. Always use this flag.
- `-o` *`filename`* – Produce an output executable file with a given `filename`.

2.2.2. Compilation without linking

To compile the code without linking, use the following commands:

```
g++ -std=c++17 -pedantic -Wall -Wextra –Wconversion –Werror
-o source.o –c source.cpp
```

```
clang++ -std=c++17 -pedantic -Wall -Wextra –Wconversion –Werror
-o source.o –c src.cpp
```

Explanation of the additional options used:

- `-c` – Compile only; do not link.

2.2.3. Linking

To link compiled object code, use the following commands:

```
g++ –o output.exe source.o
```

```
clang++ –o output.exe source.o
```

2.2.4. Getting help

To see the available flags and help options for g++ and clang++ compilers, type the following commands into the command prompt of an environment supporting the `man` manual pages:

```
man g++
```

```
man clang++
```

2.3. Microsoft Visual C++

2.3.1. Compilation with linking

To compile and link the code with Visual C++ (Visual Studio 2017), use the following commands:

```
cl /W4 /WX /nologo /EHa /std:c++17 /Za /Feoutput.exe source.cpp
```

Explanation of the options used:

- `/W4` – Set the warning level 4, showing all level 3 warnings plus additional warnings.
- `/WX` – Treat all warnings as errors; compilation with warnings *will* fail. Always use this flag.
- `/nologo` – Do not display Microsoft-specific banner when the compiler starts up.
- `/EHa` – Use an asynchronous (structured) the exception handling model.
- `/std:c++17` – Ensure ISO C++17 standards mode; replace with `/Tc` when compiling C code.
- `/Za` – Disable optional language extensions; this flag may conflict with Windows API headers.
- `/Fe`*`filename`* – Produce an output executable file with a given `filename`.

2.3.2. Compilation without linking

To compile the code without linking, use the following commands:

```
cl /W4 /WX /nologo /EHa /std:c++17 /Za /Fosource.obj /c source.cpp
```

Explanation of the additional options used:

- `/c` – Compile only; do not link.
- `/Fo`*`filename`* – Produce an object file with a given `filename`.

2.3.3. Linking

To link compiled object code, use the following commands:

```
link source.obj /OUT:filename.exe
```

2.3.4. Getting help

To see the available flags and help options for Microsoft Visual C++ compiler and linker, type the following commands into the command prompt:

```
cl /?
```

```
link /?
```

You can also visit Microsoft C/C++ Building Reference[3] webpage.

[3] https://docs.microsoft.com/en-us/cpp/build/reference/c-cpp-building-reference