The Programming Environment

[CS 120] High-level Programming I: The C Programming Language

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 C89 (also known as C90). For all the assignments we will be using two compilers to build and link our C files:

- GNU Compiler Collection with a gcc compiler (7.4.0 or newer)
- Microsoft Visual C++ compiler (Visual Studio 2017 or newer)

Use of a *clang* compiler is not enforced, but still highly encouraged if you have a machine where it works properly. Most of the configuration flags for *clang* match *gcc*, as it has been designed as an in-place replacement for *gcc*. Because of this, their flags are presented together.

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 (7.4.0), use the following commands:

```
gcc -ansi -pedantic -Wall -Wextra -Wconverison –Werror
-o output.exe source.c
```

```
clang -ansi -pedantic -Wall –Wextra -Wconversion –Werror
-o output.exe source.c
```

Explanation of the options used:

- `-ansi` – Ensure C90 ANSI standards mode; equivalent of `-std=c90`.
- `-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 -ansi -pedantic -Wall -Wextra -Wconverison –Werror
-o source.o –c source.c
```

```
clang -ansi -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:

```
man gcc
```

```
man clang
```

2.2. Microsoft Visual C++

2.2.1. Compilation with linking

To compile and link the C programming language code with Microsoft Visual C++ (Visual Studio 2017), use the following command:

```
cl /W4 /WX /nologo /TC /Za /Feoutput.exe source.c
```

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.
- `/TC` – Assume the input file uses the C programming language, not C++.
- `/Za` – Disable optional language extensions; this flag may conflict with Windows API headers.
- `/Fe`*`filename`* – Produce an output executable file with a given `filename`.

2.2.2. Compilation without linking

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

```
cl /W4 /WX /nologo /TC /Za /Fooutput.obj /c source.c
```

Explanation of the additional options used:

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

2.2.3. Linking

To link compiled object code, use the following commands:

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

2.2.4. 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 webpage.