

The GCC Compiler

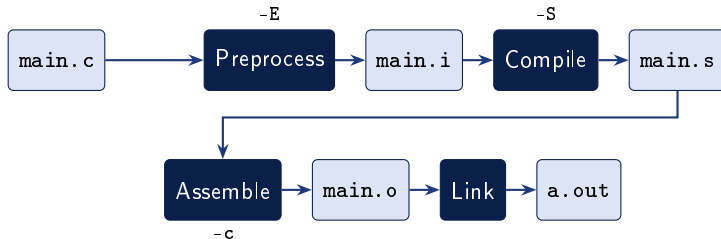
Compiler Construction, Lecture 3

K.R. Chowdhary, Professor of CS

July 14, 2019 | Fall 2019

GCC at a Glance

gcc is the GNU C and C++ compiler. One invocation runs preprocessing, compilation, assembly and linking.



General form

```
gcc [-c | -S | -E] [-std=standard] [-g] [-pg] [-O level]  
    [-foption...] [-o outfile] infile
```

Details: `man gcc`. Compile C++ programs by invoking `g++`.

Common Options

Option	Effect
-E	Stop after preprocessing; output goes to standard output
-S	Stop after compilation; produces assembly (.s) per input file
-c	Compile and assemble, but do not link; produces .o files
-o file	Place output in file (executable, object, assembly or preprocessed code)
-v	Print the commands run for each stage, and the compiler version
-O level	Optimize the generated code for time and space
-g, -pg	Debugging information; profiling information
-Wall	Enable all warning messages (-w disables all warnings)
-std=	Select the language standard
-save-temps	Keep all intermediate files (.i, .s, .o)

Options are not grouped: `-dv` is very different from `-d -v`. Order matters only when options of the same kind repeat, e.g. several `-L`.

File Suffixes Decide the Compilation

Suffix	Meaning
file.c	C source, not preprocessed
file.i	C source, preprocessed
file.ii	C++ source, preprocessed
file.m	Objective-C source (must be linked)
file.mm, .M	Objective-C++ source
file.h	C/C++ header file
file.s	Assembler code

Default output names

- -c: suffix .c, .i, .s replaced by .o
- -S: suffix replaced by .s
- no -o: executable is a.out

Hands-on: Output of Each Stage

main.c

```
#include<stdio.h>
int main()
{
    printf("\n Hello
    World\n");
    return 0;
}
```

Commands

```
$ gcc main.c                # executable a.out
$ gcc main.c -o main        # executable main
$ gcc -E main.c > main.i    # preprocessor output
$ gcc -S main.c             # assembly in main.s
$ gcc -c main.c             # object file main.o
$ gcc main.o                # link into a.out
$ gcc -save-temps main.c    # keep all stages
$ ls
a.out main.c main.i main.o main.s
$ gcc -Wall -v main.c -o main # show each stage
```

General

- C and C++ language options
- Warning and debugging options
- Optimization options
- Preprocessor, assembler and linker options
- Code generation options

Machine dependent (e.g. -mcpu=name)

- x86, x86 Windows, IA-64
- SPARC, Solaris-2, PowerPC, RISC-V
- DEC Alpha, VAX, PDP-11, M680x0, RS/6000

Takeaway

The front end is shared across targets, while the machine-dependent options select the back end. This is why one GCC can serve many architectures.

Review: `main.c` with a for-loop

1. gcc command for preprocessor output?
2. gcc command for assembly output?
3. gcc command to get all the outputs?
4. Command to count lines, characters and words?

Exercises

1. What is debugging, and which GCC options support it?
2. What does `-O` do, given a single IR optimization stage?
3. What is linking?

Laboratory

1. Compile `hello.c` keeping all outputs; compare and justify file sizes
2. Cross-compile Hello World for Alpha, M680x0, PDP-11, VAX and x86
3. With `main.c` and `largest.c`, produce:
 - ▶ `.o` files without linking
 - ▶ `.s` files
 - ▶ `a.out` by linking the `.o` files
 - ▶ `a.out` in one command