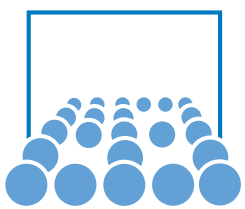
Technische Universität München

Build Process

Tutorial for Advanced Programming

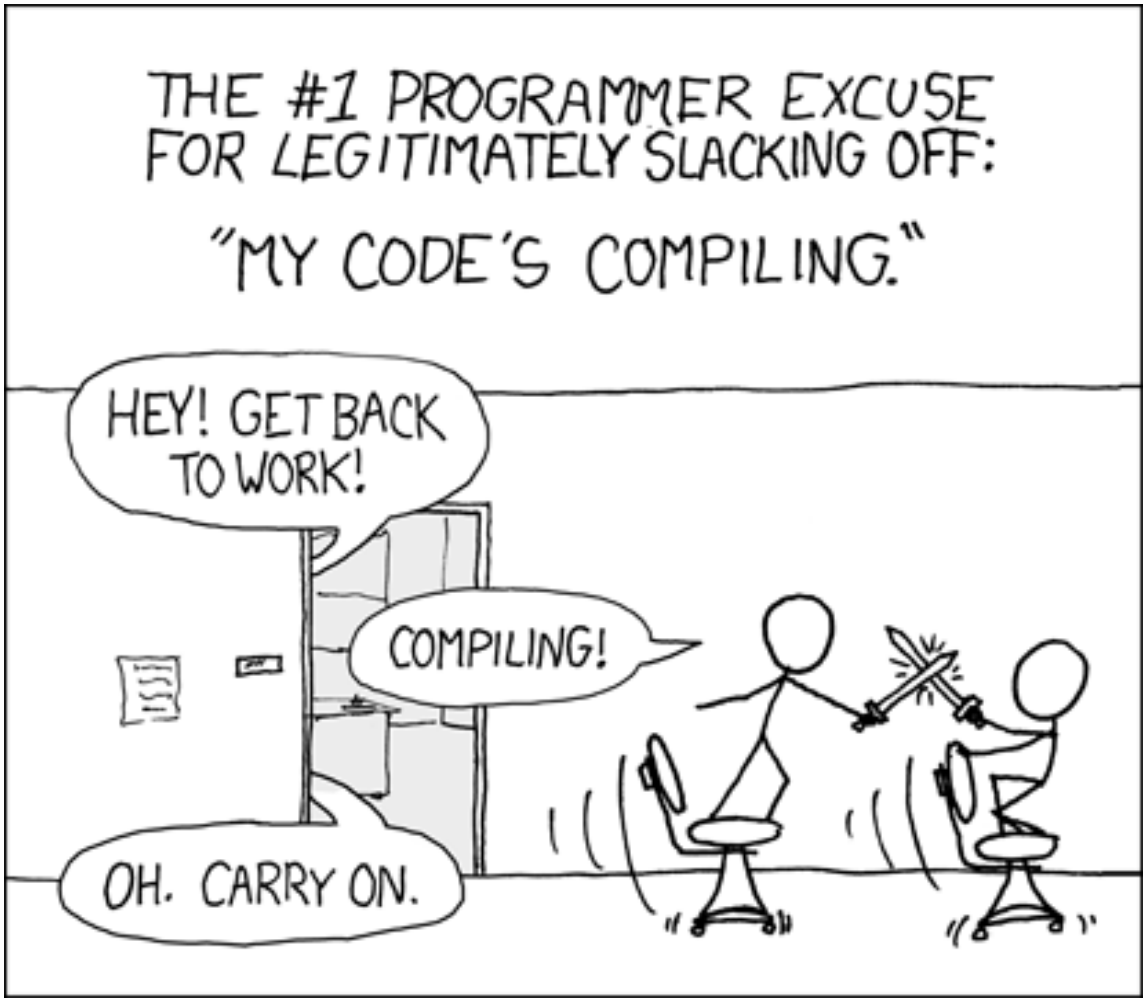
Friedrich Menhorn

December 1, 2015



Contents

COMPILER????


THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."
HEY! GET BACK TO WORK!
COMPILING!
OH. CARRY ON.

Contents

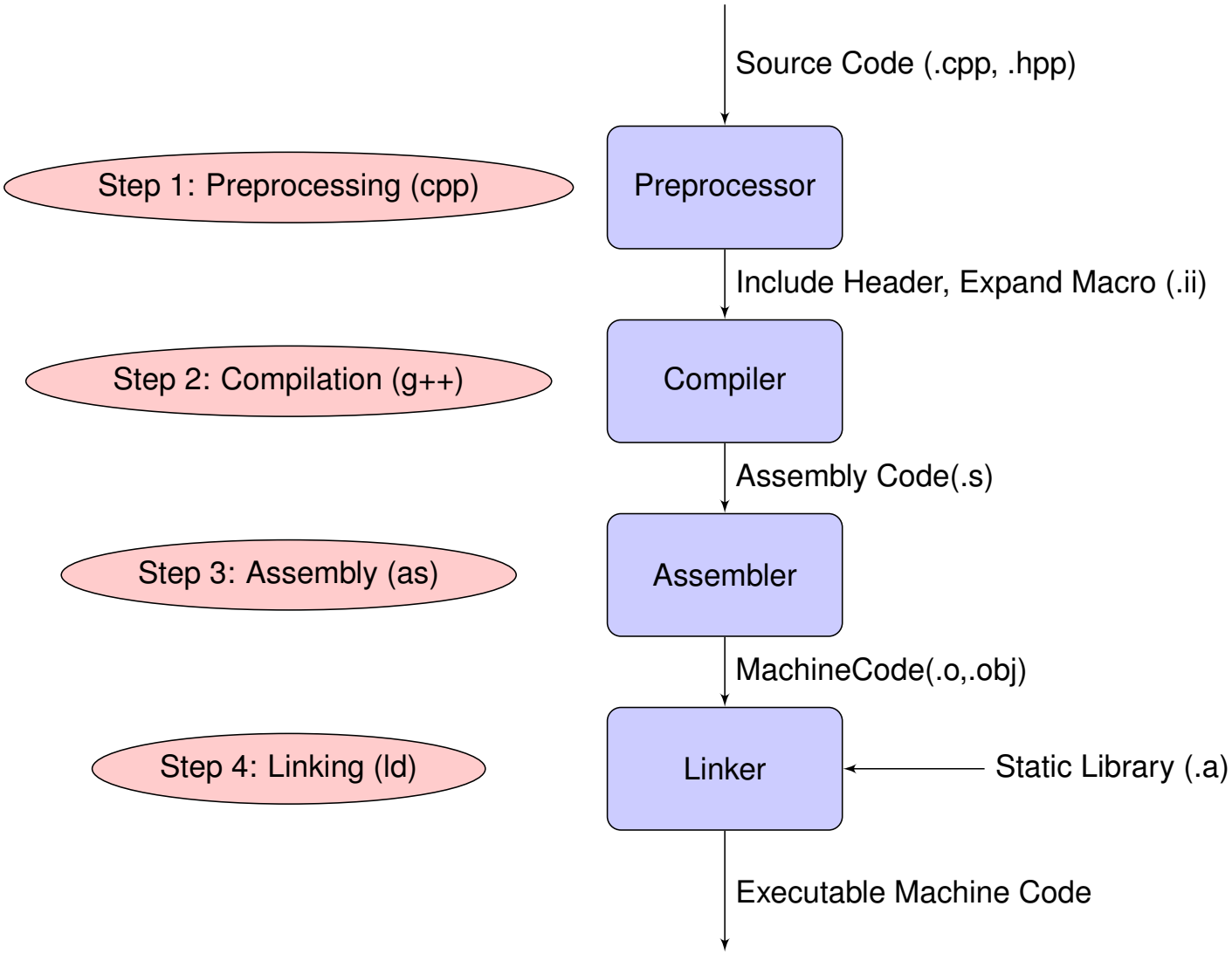
Some history and introduction

- Original *GNU C Compiler (GCC)* developed by Richard Stallman in 1984
- Part of the *GNU Toolchain*:
  - GNU Make
  - GNU Debugger
  - ...

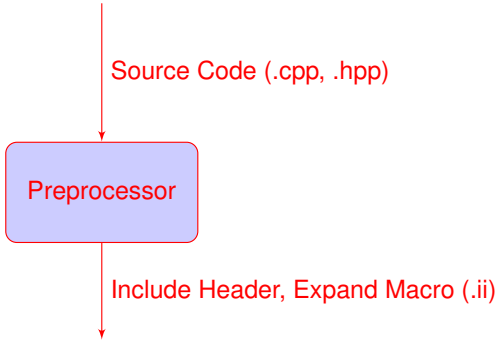
- Portable and available on many systems (Windows: MinGW/Cygwin)
- Since 2005: GCC version 4
- `$ gcc --version`
  `gcc (Ubuntu 4.9.2-10ubuntu13) 4.9.2`

Contents

The pipeline



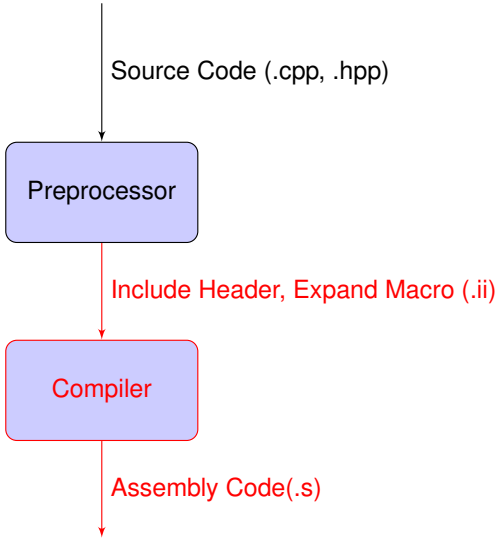
Preprocessing




```
$ cpp <CPPFlags> -I<dir> file.cpp > file.ii
```

- `CPPFlags` Sets preprocesser flags (e.g. -D)
- `-I<dir>` Sets include path (all `#include ...`) to `<dir>`
- Searched during compilation
- `$ cpp -v` returns include paths

Compiling



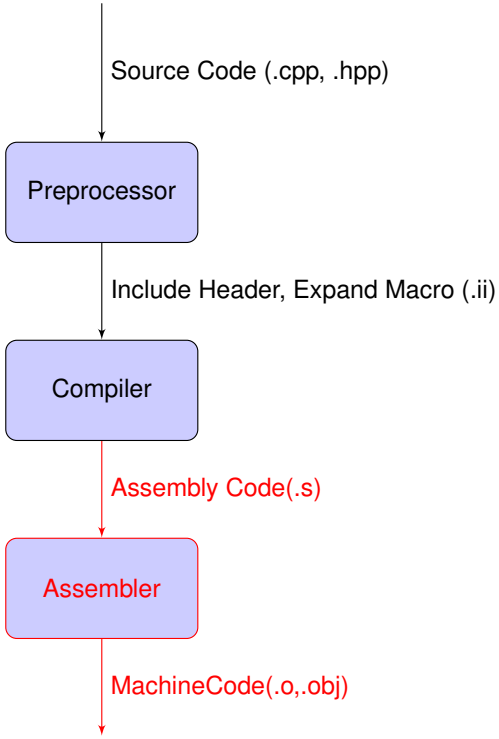

```
$ g++ <CCFlags> -S file.ii
```

- `CCFlags` Flags used during compilation e.g.
  - `-g` generates symbolic debugging information
  - `-S` generates assembler code
- More: `https://gcc.gnu.org/onlinedocs/gcc/Option-Summary.html`

Assembling

```
$ as -o file.o file.s
```

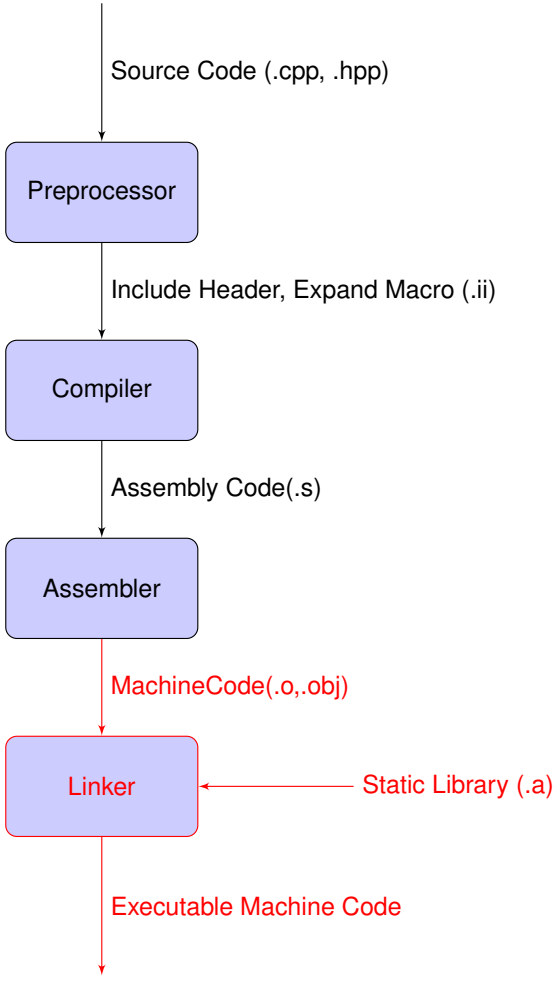
- Converts assembly code into machine code in object file `file.o`



Linking

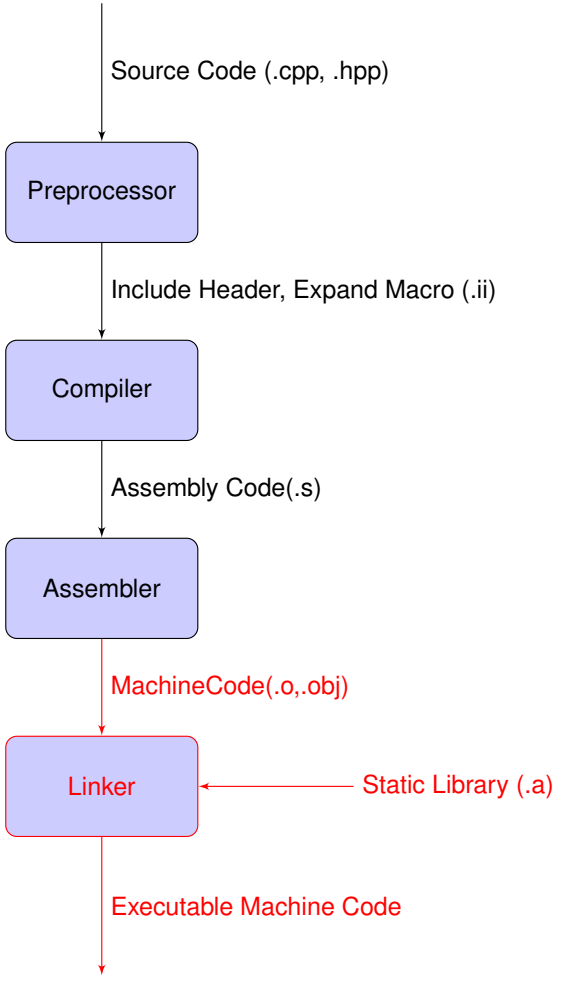
```
$ ld -o exe file1.o...fileN.o <libs>
```

- Useful commands: `ldd executable` and `gcc -v`
- LDFlags:
  - Flags used during linking
  - `-L<dir>` Sets library path to `<dir>`
  - `-lxxx` Specifies library with name `libxxx.a`
  - `<libs> = -L<dir> + -lxxx`
  - Linker needs all of these informations!!
- For convenience:
  `$ g++ -o exe file1.o...fileN.o <libs>`



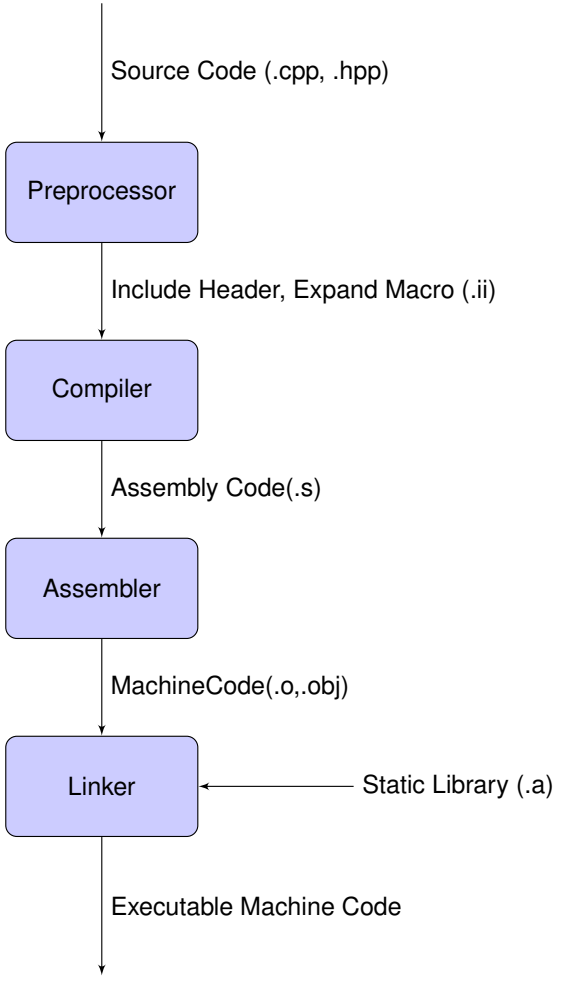
Linking

- Static libaries (.a):
  - Linked during compile time
  - Machine code of external functions is copied into executable
- Shared libraries (.so):
  - Linked during runtime
  - When linked, only small table is created
  - OS loads the needed machine code → *dynamic linking*
  - Makes executable smaller and saves disk space
  - Library can be shared between programs
  - Library automatically updated!!



GCC helps you

But each step individually??? That sounds exhausting!

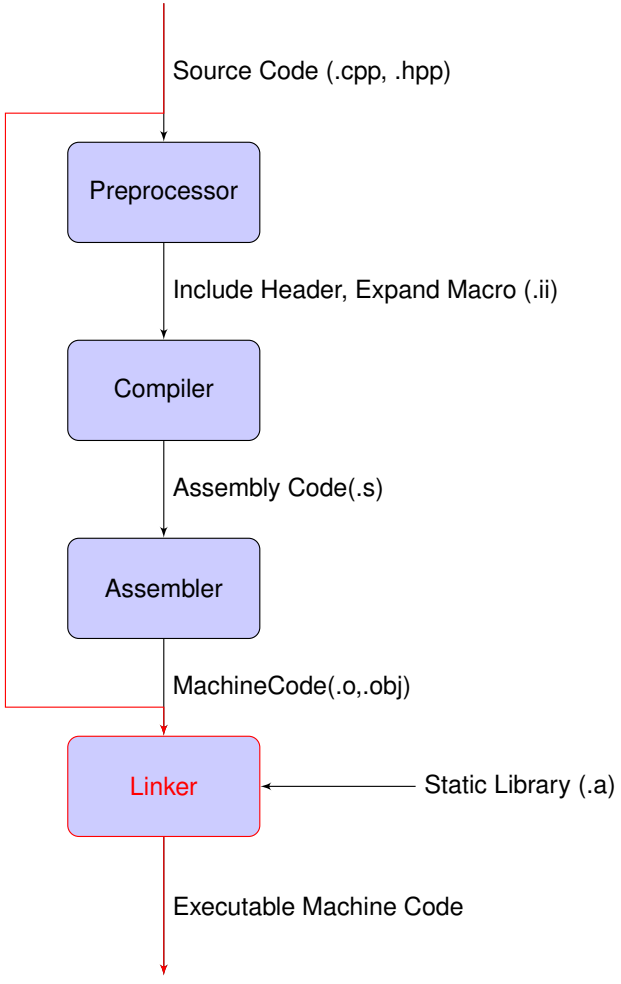


GCC helps you

Compiling in two steps:

Here: `CFLAGS = CPPFLAGS + CCFLAGS`

```
$ g++ $(CFLAGS) $(INCLUDES) -c file1.cpp
$ g++ $(CFLAGS) $(INCLUDES) -c file2.cpp
$ g++ $(CFLAGS) $(INCLUDES) -c file3.cpp
$ g++ $(CFLAGS) $(INCLUDES) -c ...
$ g++ $(CFLAGS) $(INCLUDES) -c fileN.cpp
$ g++ $(LDFLAGS) -o exec file1.o file2.o ...
fileN.o
```

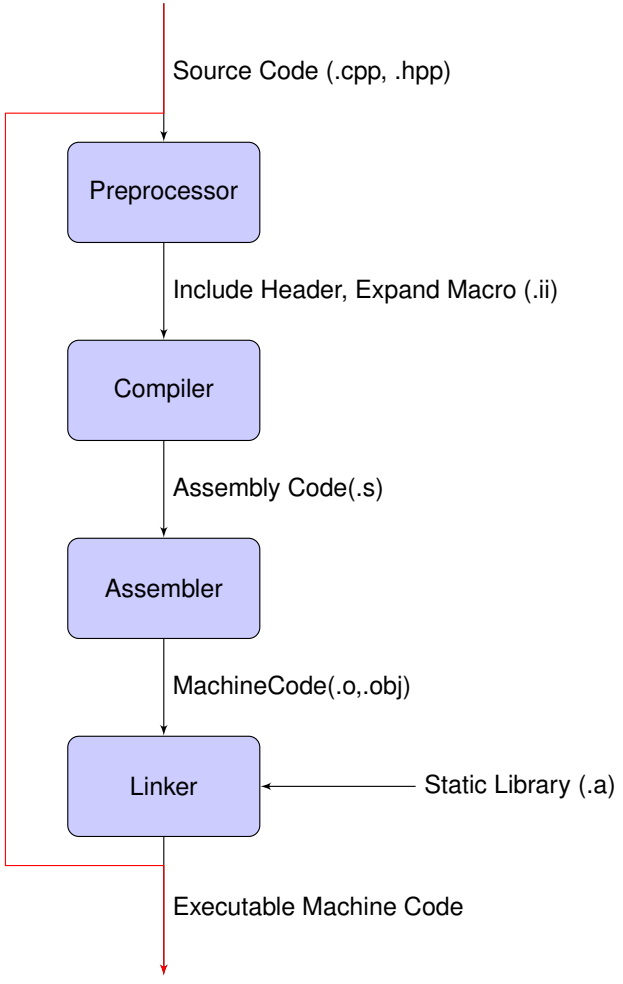


GCC helps you

Compiling in one step:

Here: `CFLAGS = CPPFLAGS + CCFLAGS`

```
$ g++ $(CFLAGS) $(INCLUDES) $(LDFLAGS) -o
exec file1.cpp file2.cpp ... fileN.cpp
```



Contents

What if I want to know more?

- More detailed information about gcc build process and makefiles:
  `www3.ntu.edu.sg/home/ehchua/programming/cpp/gcc_make.html`
- Funny example about what is actually in your binary
  `http://www.muppetlabs.com/~breadbox/software/tiny/teensy.html`