How to write programs in GCC compiler in C?

Let us write a program to see how it looks like.

Collapse | Copy Code

```
Example: test.c
```

Collapse | Copy Code

```
__asm__(".code16\n");
__asm__("jmpl $0x0000, $main\n");

void main() {
}
```

Collapse | Copy Code

```
File: test.ld
```

Collapse | Copy Code

```
ENTRY(main);
SECTIONS
{
    . = 0x7C00;
    .text : AT(0x7C00)
    {
        *(.text);
    }
    .sig : AT(0x7DFE)
    {
        SHORT(0xaa55);
    }
}
```

How to compile a C program? On the command prompt type the below:

- gcc -c -g -Os -march=i686 -ffreestanding -Wall -Werror test.c -o test.o
- ld -static -Ttest.ld -nostdlib --nmagic -o test.elf test.o
- objcopy -O binary test.elf test.bin

What does the above commands means to us anyway?

- gcc -c -g -Os -march=i686 -ffreestanding -Wall -Werror test.c -o test.o:

  This command converts the given C code into respective object code which is an intermediate code generated by the compiler before converting into machine code.

What does each flag mean?

- -c: It is used to compile the given source code without linking.
- -g: Generates debug information to be used by GDB debugger.
- -Os: optimization for code size
- -march: Generates code for the specific CPU architecture (in our case i686)
- -ffreestanding: A freestanding environment is one in which the standard library may not exist, and program startup may not necessarily be at ‘main’.
- -Wall: Enable all compiler's warning messages. This option should always be used, in order to generate better code.
- -Werror: Enable warnings being treated as errors
- test.c: input source file name
- -o: generate object code
- test.o: output object code file name.

With all the above combinations of flags to the compiler, we try to generate object code which helps us in identifying errors, warnings and also produce much efficient code for the type of CPU. If you do not specify march=i686 it generates code for the machine type you have or else it on order to port it always better to specify which type of CPU are you targeting for.

- ld -static -Ttest.ld -nostdlib --nmagic test.elf -o test.o:

This is the command to invoke linker from the command prompt and I have explained below what are we trying to do with the linker.

What does each flag mean?

- -static: Do not link against shared libraries.
- -Ttest.ld: This feature permits the linker to follow commands from a linker script.
- -nostdlib: This feature permits the linker to generate code by linking no standard C library startup functions.
- --nmagic:This feature permits the linker to generate code without _start_SECTION and _stop_SECTION codes.
- test.elf: input file name(platform dependent file format to store executables Windows: PE, Linux: ELF)
- -o: generate object code
- test.o: output object code file name.

What is a linker?

It is the final stage of compilation. The ld(linker) takes one or more object files or libraries as input and combines them to produce a single (usually executable) file. In doing so, it resolves references to external symbols, assigns final addresses to procedures/functions and variables, and revises code and data to reflect new addresses (a process called relocation).

Also remember that we have no standard libraries and all fancy functions to use in our code.

- objcopy -O binary test.elf test.bin

This command is used to generate platform independent code. Note that Linux stores executables in a different way than windows. Each have their own way storing files but we are just developing a small code to boot which does not depend on any operating system at the moment. So we are dependent on neither of those as we don't require an Operating system to run our code during boot time.

Why use assembly statements inside a C program?

In Real Mode, the BIOS functions can be easily accessed through software interrupts, using Assembly language instructions. This has lead to the usage of inline assembly in our C code.

How to copy the executable code to a bootable device and then test it?

To create a floppy disk image of 1.4mb size, type the following on the command prompt.

- dd if=/dev/zero of=floppy.img bs=512 count=2880

To copy the code to the boot sector of the floppy disk image file, type the following on the command prompt.

- dd if=test.bin of=floppy.img

To test the program type the following on the command prompt

- bochs