Major gcc Options | Coursera

The compiled code format will be **ELF** (Executable and Linkable Format), which makes using shared libraries easy; the older **a.out** format, while obsolete (although the name **a.out** survives, confusingly, as the default name for an output file), may still be used if the Linux kernel has been configured to support it.

Here is a list of some of the main options that can be given to **gcc**:

Compiler Path Options

Compiler Path Options

Option	Description
-I dir	Include **dir** in search for included files; cumulative
-L dir	Search **dir** for libraries; cumulative
-l	Link to **lib**; **-lfoo** links to **libfoo.so** if it exists, or to **libfoo.a** as a second choice

Compiler Preprocessor Options

Option	Description
-M	Do not compile; give dependencies for make
-H	Print out names of included files
-E	Preprocess only
-D def	Define **def**
-U def	Undefine **def**
-d	Print **#defines**

Compiler Warning Options

Option	Description
-v	Verbose mode, gives version number

Option	Description
-pedantic	Warn very verbosely
-w	Suppress warnings
-W	More verbose warnings
-Wall	Enable a bunch of important warnings

Compiler Debugging and Profiling Options

Option	Description
-g	Include debugging information
-pg	Provide profile information for **gprof**

Compiler Input and Output Options

Option	Description
-c	Stop after creating object files, do not link
-o file	Output is file ; default is **a.out**
-x lang	Expect input to be in **lang**, which can be **c**, **objective-c**, **c++** (and some others); otherwise, guess by input file extension

Compiler Control Options

Option	Description
-ansi	Enforce full ANSI compliance
-pipe	Use pipes between stages
-static	Suppress linking with shared libraries
-O[lev]	Optimization level; 0, 1, 2, 3; default is 0
-Os	Optimize for size; use all **-O2** options except those that increase the size

A good set of options to use is:

```
-O2 -Wall -pedantic
```

Make sure you understand any warnings; if you take the effort to obliterate them, you might save yourself a lot of debugging. However, do not use **-pedantic** when compiling code for the Linux kernel, which uses many **gcc** extensions.