GCC C/C++ compiler

CONCEPTS

gcc drives four stages: preprocess, compile, assemble, link. -E, -S and -c stop after the first three.

x.c source
↓ -E
x.i preprocessed
↓ -S
x.s assembly
↓ -c
x.o object
↓ link
a.out executable

Each stage has a flag that stops after it

Put libraries (-l) after the files that use them: the linker resolves symbols left to right.

COMPILE AND LINK

gcc *<file.c>* **-o** *<out>*
Build an executable

gcc -c *<file.c>*
Compile to an object file

gcc -E *<file.c>*
Preprocess only, print result

gcc -S *<file.c>*
Compile to assembly (.s)

gcc *<a.o>* *<b.o>* **-o** *<out>*
Link objects together

gcc *<a.c>* *<b.c>* **-o** *<out>*
Compile several files at once

gcc -v *<file.c>*
Show every sub-command run

LANGUAGE STANDARD

gcc -std=c17 *<file.c>*
Pick the C standard

g++ -std=c++20 *<file.cpp>*
Pick the C++ standard

WARNINGS

gcc -Wall *<file.c>*
Common warnings

gcc -Wextra *<file.c>*
Extra warnings on top of -Wall

gcc -Werror *<file.c>*
Treat warnings as errors

gcc -Wno-*<name>* *<file.c>*
Silence one warning

OPTIMIZATION

gcc -O0 *<file.c>*
No optimization (default)

gcc -O2 *<file.c>*
Standard release optimization

gcc -march=native *<file.c>*
Tune for this CPU

DEBUG AND PROFILE

gcc -g *<file.c>*
Add debug info for gdb

SANITIZERS

gcc -fsanitize=address -g *<file.c>*
AddressSanitizer: memory errors

gcc -fsanitize=undefined -g *<file.c>*
UBSan: undefined behavior

gcc -fsanitize=address,undefined -g *<file.c>*
Combine ASan and UBSan

gcc -fsanitize=thread -g *<file.c>*
ThreadSanitizer: data races

INCLUDE AND LINK PATHS

gcc -I*<dir>* *<file.c>*
Add a header search dir

gcc -L*<dir>* *<file.c>* **-l***<name>*
Add a library search dir

gcc *<file.c>* **-l***<name>*
Link libname.so or libname.a

gcc *<file.c>* **-lm**
Link the math library

gcc -pthread *<file.c>*
Enable POSIX threads

gcc -static *<file.c>*
Link statically

gcc -Wl,*<opt>* *<file.c>*
Pass an option to the linker

gcc -Wl,-rpath,*<dir>* *<file.c>*
Embed runtime library path

gcc $(pkg-config --cflags --libs *<pkg>***)** *<file.c>*
Flags from pkg-config

MACROS

gcc -D*<name>* *<file.c>*
Define a macro as 1

gcc -D*<name>*=*<value>* *<file.c>*
Define a macro with a value

gcc -dM -E -x c /dev/null
List predefined macros

DEPENDENCIES AND LIBRARIES

gcc -MMD -MP -c *<file.c>*
Write .d dependency file with build

gcc -fPIC -c *<file.c>*
Position-independent object

gcc -shared -o lib*<name>***.so** *<a.o>*
Build a shared library

ar rcs lib*<name>***.a** *<a.o>* *<b.o>*
Build a static library

LD_LIBRARY_PATH=*<dir>* **./** *<app>*
Find shared libs at run time

INSPECT THE COMPILER

gcc --version
Show the GCC version

RECIPES

Strict debug build with sanitizers

```
gcc -std=c17 -Wall -Wextra -Werror \
    -g -O0 -fsanitize=address,undefined \
    src/*.c -o app
```

Release build with header dependencies

```
# compile
gcc -O2 -MMD -MP -c src/main.c -o build/main.o
# link
gcc build/main.o -o app -lm
# make: -include build/*.d
```

Shared library and a program using it

```
gcc -fPIC -shared -o libfoo.so foo.c
gcc main.c -L. -lfoo -Wl,-rpath,'$ORIGIN' -o app
./app
```