Finally '`nolto-rel`' configures the compiler for incremental linking where code generation is forced, a final binary is produced, and the intermediate code for later link-time optimization is stripped. When multiple object files are linked together the resulting code is better optimized than with link-time optimizations disabled (for example, cross-module inlining happens), but most of benefits of whole program optimizations are lost.

During the incremental link (by '`-r`') the linker plugin defaults to '`rel`'. With current interfaces to GNU Binutils it is however not possible to incrementally link LTO objects and non-LTO objects into a single mixed object file. If any of object files in incremental link cannot be used for link-time optimization, the linker plugin issues a warning and uses '`nolto-rel`'. To maintain whole program optimization, it is recommended to link such objects into static library instead. Alternatively it is possible to use H.J. Lu's binutils with support for mixed objects.

`-fuse-ld=bfd`
: Use the `bfd` linker instead of the default linker.

`-fuse-ld=gold`
: Use the `gold` linker instead of the default linker.

`-fuse-ld=lld`
: Use the LLVM `lld` linker instead of the default linker.

`-l`*library*
`-l` *library*
: Search the library named *library* when linking. (The second alternative with the library as a separate argument is only for POSIX compliance and is not recommended.)

  The '`-l`' option is passed directly to the linker by GCC. Refer to your linker documentation for exact details. The general description below applies to the GNU linker.

  The linker searches a standard list of directories for the library. The directories searched include several standard system directories plus any that you specify with '`-L`'.

  Static libraries are archives of object files, and have file names like '`lib`*library*`.a`'. Some targets also support shared libraries, which typically have names like '`lib`*library*`.so`'. If both static and shared libraries are found, the linker gives preference to linking with the shared library unless the '`-static`' option is used.

  It makes a difference where in the command you write this option; the linker searches and processes libraries and object files in the order they are specified. Thus, '`foo.o -lz bar.o`' searches library '`z`' after file '`foo.o`' but before '`bar.o`'. If '`bar.o`' refers to functions in '`z`', those functions may not be loaded.

`-lobjc`
: You need this special case of the '`-l`' option in order to link an Objective-C or Objective-C++ program.