



## Navigation

• [LLVM Home](#) • | [Documentation](#) • » [Reference](#) • » [LLVM Command Guide](#) » [previous](#) • | [next](#) • | [index](#)

# llvm-ar – LLVM archiver ¶

## SYNOPSIS ¶

```
llvm-ar [-]{dmpqrstx}{abcDiLLNoOPsSTuUvV} [relpos] [count]
archive [files...]
```

## DESCRIPTION ¶

The llvm-ar command is similar to the common Unix utility, ar. It archives several files, such as objects and LLVM bitcode files into a single archive library that can be linked into a program. However, the archive can contain any kind of file. By default, llvm-ar generates a symbol table that makes linking faster because only the symbol table needs to be consulted, not each individual file member of the archive.

The llvm-ar command can be used to read archive files in SVR4, GNU, BSD and Darwin format, and write in the GNU, BSD, and Darwin style archive files. If an SVR4 format archive is used with the [r](#) (replace), [d](#) (delete), [m](#) (move) or [q](#) (quick update) operations, the archive will be reconstructed in the format defined by [--format](#).

Here's where llvm-ar departs from previous ar implementations:

The following option is not supported

[f] – truncate inserted filenames

The following options are ignored for compatibility

-plugin=<string> – load a plugin which adds support for other file formats

[l] – ignored in ar

### Symbol Table

Since llvm-ar supports bitcode files, the symbol table it creates includes both native and bitcode symbols.

### Deterministic Archives

By default, llvm-ar always uses zero for timestamps and UIDs/GIDs to write archives in a deterministic mode. This is equivalent to the [D](#) modifier being enabled by default. If you wish to maintain compatibility with other ar implementations, you can pass the [U](#) modifier to write actual timestamps and UIDs/GIDs.

### Documentation

- [Getting Started/Tutorials](#)
- [User Guides](#)
- [Reference](#)

### Getting Involved

- [Contributing to LLVM](#)
- [Submitting Bug Reports](#)
- [Mailing Lists](#)
- [IRC](#)
- [Meetups and Social Events](#)

### Additional Links

- [FAQ](#)
- [Glossary](#)
- [Publications](#)
- [Github Repository](#)

### This Page

- [Show Source](#)

### Quick search

## Windows Paths

When on Windows `llvm-ar` treats the names of archived files in the same case sensitive manner as the operating system. When on a non-Windows machine `llvm-ar` does not consider character case.

## OPTIONS ¶

`llvm-ar` operations are compatible with other `ar` implementations. However, there are a few modifiers ([L](#)) that are not found in other `ar` implementations. The options for `llvm-ar` specify a single basic Operation to perform on the archive, a variety of Modifiers for that Operation, the name of the archive file, and an optional list of file names. If the files option is not specified, it generally means either “none” or “all” members, depending on the operation. The Options, Operations and Modifiers are explained in the sections below.

The minimal set of options is at least one operator and the name of the archive.

## Operations ¶

### d [NT] ¶

Delete files from the archive. The [N](#) and [I](#) modifiers apply to this operation. The files options specify which members should be removed from the archive. It is not an error if a specified file does not appear in the archive. If no files are specified, the archive is not modified.

### m [abi] ¶

Move files from one location in the archive to another. The [a](#), [b](#), and [i](#) modifiers apply to this operation. The files will all be moved to the location given by the modifiers. If no modifiers are used, the files will be moved to the end of the archive. If no files are specified, the archive is not modified.

### p [v] ¶

Print files to the standard output stream. If no files are specified, the entire archive is printed. With the [v](#) modifier, `llvm-ar` also prints out the name of the file being output. Printing binary files is ill-advised as they might confuse your terminal settings. The [p](#) operation never modifies the archive.

### q [LT] ¶

Quickly append files to the end of the archive without removing duplicates. If no files are specified, the archive is not modified. The behavior when appending one archive to another depends upon whether the [L](#) and [I](#) modifiers are used:

- Appending a regular archive to a regular archive will append the archive file. If the [L](#) modifier is specified the members will be appended instead.
- Appending a regular archive to a thin archive requires the [T](#) modifier and will append the archive file. The [L](#) modifier is not supported.

- Appending a thin archive to a regular archive will append the archive file. If the **L** modifier is specified the members will be appended instead.
- Appending a thin archive to a thin archive will always quick append its members.

**r** [abTu] ¶

Replace existing files or insert them at the end of the archive if they do not exist. The **a**, **b**, **T** and **u** modifiers apply to this operation. If no files are specified, the archive is not modified.

**t**[v] .. option:: t [vO]

Print the table of contents. Without any modifiers, this operation just prints the names of the members to the standard output stream. With the **v** modifier, `llvm-ar` also prints out the file type (B=bitcode, S=symbol table, blank=regular file), the permission mode, the owner and group, are ignored when extracting files and set to placeholder values when adding size, and the date. With the **O** modifier, display member offsets. If any files are specified, the listing is only for those files. If no files are specified, the table of contents for the whole archive is printed.

**v** ¶

A synonym for the `--version` option.

**x** [oP] ¶

Extract archive members back to files. The **o** modifier applies to this operation. This operation retrieves the indicated files from the archive and writes them back to the operating system's file system. If no files are specified, the entire archive is extracted.

## Modifiers (operation specific) ¶

The modifiers below are specific to certain operations. See the Operations section to determine which modifiers are applicable to which operations.

**a** ¶

When inserting or moving member files, this option specifies the destination of the new files as being after the relpos member. If relpos is not found, the files are placed at the end of the archive. relpos cannot be consumed without either **a**, **b** or **i**.

**b** ¶

When inserting or moving member files, this option specifies the destination of the new files as being before the relpos member. If relpos is not found, the files are placed at the end of the archive. relpos cannot be consumed without either **a**, **b** or **i**. This modifier is identical to the **i** modifier.

**i** ¶

A synonym for the **b** option.

**L** ¶

When quick appending an archive, instead quick append its members. This is a feature for `llvm-ar` that is not found in `gnu-ar`.

**N** ¶

When extracting or deleting a member that shares its name with another member, the count parameter allows you to supply a positive whole number that selects the instance of the given name, with “1” indicating the first instance. If **N** is not specified the first member of that name will be selected. If count is not supplied, the operation fails. `*count*` cannot be

**o** ¶

When extracting files, use the modification times of any files as they appear in the archive. By default files extracted from the archive use the time of extraction.

**O** ¶

Display member offsets inside the archive.

**T** ¶

When creating or modifying an archive, this option specifies that the archive will be thin. By default, archives are not created as thin archives and when modifying a thin archive, it will be converted to a regular archive.

**v** ¶

When printing files or the archive table of contents, this modifier instructs `llvm-ar` to include additional information in the output.

## Modifiers (generic) ¶

The modifiers below may be applied to any operation.

**c** ¶

For the **r** (replace) and **q** (quick update) operations, `llvm-ar` will always create the archive if it doesn’t exist. Normally, `llvm-ar` will print a warning message indicating that the archive is being created. Using this modifier turns off that warning.

**D** ¶

Use zero for timestamps and UIDs/GIDs. This is set by default.

**P** ¶

Use full paths when matching member names rather than just the file name. This can be useful when manipulating an archive generated by another archiver, as some allow paths as member names. This is the default behavior for thin archives.

s ¶

This modifier requests that an archive index (or symbol table) be added to the archive, as if using `ranlib`. The symbol table will contain all the externally visible functions and global variables defined by all the bitcode files in the archive. By default `llvm-ar` generates symbol tables in archives. This can also be used as an operation.

S ¶

This modifier is the opposite of the `s` modifier. It instructs `llvm-ar` to not build the symbol table. If both `s` and `S` are used, the last modifier to occur in the options will prevail.

u ¶

Only update archive members with files that have more recent timestamps.

U ¶

Use actual timestamps and UIDs/GIDs.

## Other ¶

`--format=<type>` ¶

This option allows for default, gnu, darwin or bsd `<type>` to be selected. When creating an archive, `<type>` will default to that of the host machine.

`-h, --help` ¶

Print a summary of command-line options and their meanings.

`-M` ¶

This option allows for MRI scripts to be read through the standard input stream. No other options are compatible with this option.

`--version` ¶

Display the version of the `llvm-ar` executable.

`@<FILE>` ¶

Read command-line options and commands from response file `<FILE>`.

## MRI SCRIPTS ¶

`llvm-ar` understands a subset of the MRI scripting interface commonly supported by archivers following in the ar tradition. An MRI script contains a sequence of commands to be executed by the archiver. The `-M` option allows for an MRI script to be passed to `llvm-ar` through the standard input stream.

Note that `llvm-ar` has known limitations regarding the use of MRI scripts:

- Each script can only create one archive.

- Existing archives can not be modified.

## MRI Script Commands ¶

Each command begins with the command's name and must appear on its own line. Some commands have arguments, which must be separated from the name by whitespace. An MRI script should begin with either a [CREATE](#) or [CREATETHIN](#) command and will typically end with a [SAVE](#) command. Any text after either '\*' or ';' is treated as a comment.

`CREATE archive ¶`

Begin creation of a regular archive with the specified name. Subsequent commands act upon this archive.

`CREATETHIN archive ¶`

Begin creation of a thin archive with the specified name. Subsequent commands act upon this archive.

`ADDLIB archive ¶`

Append the contents of archive to the current archive.

`ADDMOD <file> ¶`

Append <file> to the current archive.

`DELETE <file> ¶`

Delete the member of the current archive whose file name, excluding directory components, matches <file>.

`SAVE ¶`

Write the current archive to the path specified in the previous [CREATE](#)/[CREATETHIN](#) command.

`END ¶`

Ends the MRI script (optional).

## EXIT STATUS ¶

If `llvm-ar` succeeds, it will exit with 0. Otherwise, if an error occurs, it will exit with a non-zero value.

### Navigation

• [LLVM Home](#) • | [Documentation](#) • » [Reference](#) • » [LLVM Command Guide](#) » [previous](#) • | [next](#) • | [index](#)