NAME

nm – list symbols from object files

SYNOPSIS

nm [**–A**|**–o**|**––print–file–name**] [**–a**|**––debug–syms**] [**–B**|**––format=bsd**] [**–C**|**––demangle**[=*style*]] [**–D**|**––dynamic**] [**–f***format*|**––format=***format*] [**–g**|**––extern–only**] [**–h**|**––help**] [**–l**|**––line–numbers**] [**––inlines**] [**–n**|**–v**|**––numeric–sort**] [**–P**|**––portability**] [**–p**|**––no–sort**] [**–r**|**––reverse–sort**] [**–S**|**––print–size**] [**–s**|**––print–armap**] [**–t** *radix*|**––radix=***radix*] [**–u**|**––undefined–only**] [**–V**|**––version**] [**–X 32_64**] [**––defined–only**] [**––no–demangle**] [**––plugin** *name*] [**––size–sort**] [**––special–syms**] [**––synthetic**] [**––with–symbol–versions**] [**––target=***bfdname*] [*objfile*...]

DESCRIPTION

GNU **nm** lists the symbols from object files *objfile*.... If no object files are listed as arguments, **nm** assumes the file *a.out*.

For each symbol, **nm** shows:

- The symbol value, in the radix selected by options (see below), or hexadecimal by default.
- The symbol type. At least the following types are used; others are, as well, depending on the object file format. If lowercase, the symbol is usually local; if uppercase, the symbol is global (external). There are however a few lowercase symbols that are shown for special global symbols (`u`, `v` and `w`).

  `A` The symbol's value is absolute, and will not be changed by further linking.

  `B`
  `b` The symbol is in the BSS data section. This section typically contains zero-initialized or uninitialized data, although the exact behavior is system dependent.

  `C` The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references.

  `D`
  `d` The symbol is in the initialized data section.

  `G`
  `g` The symbol is in an initialized data section for small objects. Some object file formats permit more efficient access to small data objects, such as a global int variable as opposed to a large global array.

  `i` For PE format files this indicates that the symbol is in a section specific to the implementation of DLLs. For ELF format files this indicates that the symbol is an indirect function. This is a GNU extension to the standard set of ELF symbol types. It indicates a symbol which if referenced by a relocation does not evaluate to its address, but instead must be invoked at runtime. The runtime execution will then return the value to be used in the relocation.

  `I` The symbol is an indirect reference to another symbol.

  `N` The symbol is a debugging symbol.

  `p` The symbols is in a stack unwind section.

  `R`
  `r` The symbol is in a read only data section.

  `S`
  `s` The symbol is in an uninitialized or zero-initialized data section for small objects.

  `T`
  `t` The symbol is in the text (code) section.

  `U` The symbol is undefined.

`u` The symbol is a unique global symbol. This is a GNU extension to the standard set of ELF symbol bindings. For such a symbol the dynamic linker will make sure that in the entire process there is just one symbol with this name and type in use.

`V`

`v` The symbol is a weak object. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. On some systems, uppercase indicates that a default value has been specified.

`W`

`w` The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the symbol is determined in a system-specific manner without error. On some systems, uppercase indicates that a default value has been specified.

`-` The symbol is a stabs symbol in an a.out object file. In this case, the next values printed are the stabs other field, the stabs desc field, and the stab type. Stabs symbols are used to hold debugging information.

`?` The symbol type is unknown, or object file format specific.

- The symbol name.

OPTIONS

The long and short forms of options, shown here as alternatives, are equivalent.

–A
–o
––print–file–name
Precede each symbol by the name of the input file (or archive member) in which it was found, rather than identifying the input file once only, before all of its symbols.

–a
––debug–syms
Display all symbols, even debugger-only symbols; normally these are not listed.

–B The same as **––format=bsd** (for compatibility with the MIPS **nm**).

–C
––demangle[=*style***]**
Decode (*demangle*) low-level symbol names into user-level names. Besides removing any initial underscore prepended by the system, this makes C++ function names readable. Different compilers have different mangling styles. The optional demangling style argument can be used to choose an appropriate demangling style for your compiler.

––no–demangle
Do not demangle low-level symbol names. This is the default.

–D
––dynamic
Display the dynamic symbols rather than the normal symbols. This is only meaningful for dynamic objects, such as certain types of shared libraries.

–f *format*
––format=*format*
Use the output format *format*, which can be `bsd`, `sysv`, or `posix`. The default is `bsd`. Only the first character of *format* is significant; it can be either upper or lower case.

–g

−−extern−only
Display only external symbols.

−h
−−help
Show a summary of the options to **nm** and exit.

−l
−−line−numbers
For each symbol, use debugging information to try to find a filename and line number. For a defined symbol, look for the line number of the address of the symbol. For an undefined symbol, look for the line number of a relocation entry which refers to the symbol. If line number information can be found, print it after the other symbol information.

−−inlines
When option **−l** is active, if the address belongs to a function that was inlined, then this option causes the source information for all enclosing scopes back to the first non-inlined function to be printed as well. For example, if `main` inlines `callee1` which inlines `callee2`, and address is from `callee2`, the source information for `callee1` and `main` will also be printed.

−n
−v
−−numeric−sort
Sort symbols numerically by their addresses, rather than alphabetically by their names.

−p
−−no−sort
Do not bother to sort the symbols in any order; print them in the order encountered.

−P
−−portability
Use the POSIX.2 standard output format instead of the default format. Equivalent to **−f posix**.

−r
−−reverse−sort
Reverse the order of the sort (whether numeric or alphabetic); let the last come first.

−S
−−print−size
Print both value and size of defined symbols for the `bsd` output style. This option has no effect for object formats that do not record symbol sizes, unless **−−size−sort** is also used in which case a calculated size is displayed.

−s
−−print−armap
When listing symbols from archive members, include the index: a mapping (stored in the archive by **ar** or **ranlib**) of which modules contain definitions for which names.

−t *radix*
−−radix=*radix*
Use *radix* as the radix for printing the symbol values. It must be **d** for decimal, **o** for octal, or **x** for hexadecimal.

−u
−−undefined−only
Display only undefined symbols (those external to each object file).

−V
−−version
Show the version number of **nm** and exit.

−X This option is ignored for compatibility with the AIX version of **nm**. It takes one parameter which must be the string **32_64**. The default mode of AIX **nm** corresponds to **−X 32**, which is not supported by GNU **nm**.

−−defined−only
Display only defined symbols for each object file.

−−plugin *name*
Load the plugin called *name* to add support for extra target types. This option is only available if the toolchain has been built with plugin support enabled.

If **−−plugin** is not provided, but plugin support has been enabled then **nm** iterates over the files in *${libdir}/bfd−plugins* in alphabetic order and the first plugin that claims the object in question is used.

Please note that this plugin search directory is *not* the one used by **ld**'s **−plugin** option. In order to make **nm** use the linker plugin it must be copied into the *${libdir}/bfd−plugins* directory. For GCC based compilations the linker plugin is called *liblto_plugin.so.0.0.0*. For Clang based compilations it is called *LLVMgold.so*. The GCC plugin is always backwards compatible with earlier versions, so it is sufficient to just copy the newest one.

−−size−sort
Sort symbols by size. For ELF objects symbol sizes are read from the ELF, for other object types the symbol sizes are computed as the difference between the value of the symbol and the value of the symbol with the next higher value. If the `bsd` output format is used the size of the symbol is printed, rather than the value, and **−S** must be used in order both size and value to be printed.

−−special−syms
Display symbols which have a target-specific special meaning. These symbols are usually used by the target for some special processing and are not normally helpful when included in the normal symbol lists. For example for ARM targets this option would skip the mapping symbols used to mark transitions between ARM code, THUMB code and data.

−−synthetic
Include synthetic symbols in the output. These are special symbols created by the linker for various purposes. They are not shown by default since they are not part of the binary's original source code.

−−with−symbol−versions
Enables the display of symbol version information if any exists. The version string is displayed as a suffix to the symbol name, preceeded by an @ character. For example **foo@VER_1**. If the version is the default version to be used when resolving unversioned references to the symbol then it is displayed as a suffix preceeded by two @ characters. For example **foo@@VER_2**.

−−target=*bfdname*
Specify an object code format other than your system's default format.

@*file*
Read command-line options from *file*. The options read are inserted in place of the original @*file* option. If *file* does not exist, or cannot be read, then the option will be treated literally, and not removed.

Options in *file* are separated by whitespace. A whitespace character may be included in an option by surrounding the entire option in either single or double quotes. Any character (including a backslash) may be included by prefixing the character to be included with a backslash. The *file* may itself contain additional @*file* options; any such options will be processed recursively.

SEE ALSO

ar(1), *objdump(1)*, *ranlib(1)*, and the Info entries for *binutils*.

COPYRIGHT