

NAME

depmod - Generate modules.dep and map files.

SYNOPSIS

depmod [-b *basedir*] [-m *moduledir*] [-o *outdir*] [-e] [-E *Module.symvers*]
[-F *System.map*] [-n] [-v] [-A] [-P *prefix*] [-w] [*version*]

depmod [-e] [-E *Module.symvers*] [-F *System.map*] [-n] [-v] [-P *prefix*]
[-w] [*version*] [*filename...*]

DESCRIPTION

Linux kernel modules can provide services (called "symbols") for other modules to use (using one of the EXPORT_SYMBOL variants in the code). If a second module uses this symbol, that second module clearly depends on the first module. These dependencies can get quite complex.

depmod creates a list of module dependencies by reading each module under <BASEDIR>/<MODULEDIR>/*version*. By default <MODULEDIR> is /lib/modules and <BASEDIR> is empty. See options below to override when needed. It determines what symbols each module exports and needs. This list is written to **modules.dep**, and a binary hashed version named modules.dep.bin, in the same directory. If filenames are given on the command line, only those modules are examined (which is rarely useful unless all modules are listed). **depmod** also creates a list of symbols provided by modules in the file named modules.symbols and its binary hashed version, modules.symbols.bin. Finally, **depmod** will output a file named modules.devname if modules supply special device names (devname) that should be populated in /dev on boot (by a utility such as systemd-tmpfiles).

If a *version* is provided, then that kernel version's module directory is used rather than the current kernel version (as returned by **uname -r**).

OPTIONS**-a, --all**

Probe all modules. This option is enabled by default if no file names are given in the command-line.

-A, --quick

This option scans to see if any modules are newer than the **modules.dep** file before any work is done: if not, it silently exits rather than regenerating the files.

-b basedir, --basedir=basedir

Override the base directory <BASEDIR> where modules are located. If your modules are not currently in the (normal) directory /lib/modules/*version*, but in a staging area, you can specify a *basedir* which is prepended to the directory name. This *basedir* is stripped from the resulting **modules.dep** file, so it is ready to be moved into the normal location. Use this option if you are a distribution vendor who needs to pre-generate the meta-data files rather than running **depmod** again later.

If a relative path is given, it's relative to the current working directory.

Example:

```
depmod -b /my/build/staging/dir/
```

This expects all input files under /my/build/staging/dir/lib/modules/\$(**uname -r**) and generates index files under that same directory.

-m moduledir, --moduledir=moduledir

Override the module directory <MODULEDIR>, which defaults to /lib/modules prefix set at build time. This is useful when building **modules.dep** file in *basedir* for a system that uses a different prefix, e.g. /usr/lib/modules vs /lib/modules.

Relative and absolute paths are accepted, but they are always relative to the *basedir*.

Examples:

```
depmod -b /tmp/build -m /kernel-modules
```

```
depmod -b /tmp/build -m kernel-modules
```

This expects all input files under /tmp/build/kernel-modules/\$(**uname -r**) and generates index files

under that same directory.

Without an accompanying **-b** argument, the `moduledir` is relative to `/`. Example:

```
depmod -m foo/bar
```

This expects all input files under `/foo/bar/$(uname -r)` and generates index files under the same directory. Unless `libkmod` is prepared to handle that arbitrary location, it won't work in runtime.

-o *outdir*, **--outdir**=*outdir*

Set the output directory where **depmod** will store any generated file. *outdir* serves as a root to that location, similar to how *basedir* is used. Also this setting takes precedence and if used together with *basedir* it will result in the input being that directory, but the output being the one set by *outdir*.

If a relative path is given, it's relative to the current working directory.

Example:

```
depmod -o /my/build/staging/dir/
```

This expects all input files under `/lib/modules/$(uname -r)` and generates index files under `/my/build/staging/dir/lib/modules/$(uname -r)`.

-C *file or directory*, **--config**=*file or directory*

This option overrides the default configuration files. See **depmod.d(5)**.

-e, **--errsyms**

When combined with the **-F** option, this reports any symbols which a module needs which are not supplied by other modules or the kernel. Normally, any symbols not provided by modules are assumed to be provided by the kernel (which should be true in a perfect world), but this assumption can break especially when additionally updated third party drivers are not correctly installed or were built incorrectly.

-E *Module.symvers*, **--symvers**=*Module.symvers*

When combined with the **-e** option, this reports any symbol versions supplied by modules that do not match with the symbol versions provided by the kernel in its *Module.symvers*. This option is mutually incompatible with **-F**.

-F *System.map*, **--filesyms**=*System.map*

Supplied with the *System.map* produced when the kernel was built, this allows the **-e** option to report unresolved symbols. This option is mutually incompatible with **-E**.

-h, **--help**

Print the help message and exit.

-n, **--show**, **--dry-run**

This sends the resulting **modules.dep** and the various map files to standard output rather than writing them into the module directory.

-P

Some architectures prefix symbols with an extraneous character. This specifies a prefix character (for example `'_'`) to ignore.

-v, **--verbose**

In verbose mode, **depmod** will print (to stdout) all the symbols each module depends on and the module's file name which provides that symbol.

-V, **--version**

Show version of program and exit. See below for caveats when run on older kernels.

-w

Warn on duplicate dependencies, aliases, symbol versions, etc.

COPYRIGHT

This manual page originally Copyright 2002, Rusty Russell, IBM Corporation. Portions Copyright Jon Masters, and others.

SEE ALSO

depmod.d(5), **modprobe(8)**, **modules.dep(5)**

BUGS

Please direct any bug reports to kmod's issue tracker at <https://github.com/kmod-project/kmod/issues/> alongside with version used, steps to reproduce the problem and the expected outcome.

AUTHORS

Numerous contributions have come from the linux-modules mailing list <linux-modules@vger.kernel.org> and Github. If you have a clone of kmod.git itself, the output of **git-shortlog(1)** and **git-blame(1)** can show you the authors for specific parts of the project.

Lucas De Marchi <lucas.de.marchi@gmail.com> is the current maintainer of the project.