NAME

module-assistant − manage kernel modules packages

SYNOPSIS

module-assistant [**-fihnqstv**] [**-k** *source/headers directory*] [**-l** *kernel versions*] { **update** | **search** | **prepare** | **auto-install** | **list** | **list-available** | **list-installed** | **auto-unpacked** | **get** | **build** | **install** | **clean** | **purge** | **fakesource** } [*pkg ...*]

m-a ...

DESCRIPTION

module-assistant is the command-line tool for handling module-source packages that have been prepared for the Debian distribution. It helps users to build and install module package(s) easily for one or more custom kernels.

Unless the **-h**, or **--help** option is given, one of the commands below should be present to invoke a function explicitly. If no (or no valid) command is specified and the dialog tool is available, a simple graphical user interface will pop up and guide you trough the available functionality.

NOTE: don't even think about using some random linux-source-x.y.z package contents (or linux-x.y.z tarball from the Internet) to specify the kernel source for your currently running kernel. **Don't!** Those source is not exactly what have been used to build the running kernel and its configuration most likely does not match yours. You need the configured kernel source directory or at least the derived linux-headers-... package containing the kernel configuration for the exact kernel version (complete version string). If you do not understand anything of the above, run "m-a prepare" and/or look at the description and contents of some linux-headers-... package. Please run the **module-assistant prepare** command once before you do anything else.

For some packages, linux-headers (reduced source) is not enough. You will have the choice to run a completely customized kernel, or to recreate the source that have been used to build the current one. The **fakesource** function may be useful, see below.

In order to configure a kernel source properly, you need to make sure that the file version.h is generated. To get it, configure the options as usual (**make menuconfig** etc.) and run **make dep** (for kernel 2.4.x) or **make prepare** (for newer ones).

COMMANDS

Most commands require a specification of the package names that they should be applied on. **pkg** can be a single word (package name) or multiple names. The word **all** will be expanded to the list of all available packages, the word **alli** to the list of currently installed (source) packages and the word **allu** will be expanded to the list of packages that seem to be installed and unpacked in the base source directory. If a source package with the given name is not available, **module-assistant** (abbreviated: **m-a**) will extend the package name with the popular suffixes like -kernel, -driver, -module, -source and combinations of them.

Multiple commands can be specified in one invocation, eg. "**m-a clean,get,build arla cdfs**" is the short way to write "module-assistant clean arla-modules-source ; module-assistant clean cdfs-src ; module-assistant get arla-modules-source cdfs-src ; module-assistant build arla-modules-source cdfs-src" (or similar).

If you do not like the dialog/whiptail GUI, feel free to use the **-t** switch to disable it.

update update is used to resynchronize the version index files from their sources. This happens with helper scripts provided by the packages. **module-assistant** has a default built-in list of the packages that it should look for but other packages can be registered by **module-assistant** if the maintainer adds the helper script.

prepare

Tries to determine the name of the required linux-headers package (either the one matching the currently running kernel or for the versions specified with **-l**), installs it if needed and creates the /usr/src/linux symlink if needed. Also installs the build-essential package to ensure that a sane

compiler environment is established.

fakesource
Experimental function which tries to determine the name of the required/compatible linux-source package, installs it, modifies the Makefile to look like the original source and runs configuration routines as needed. Warning: DO NOT RELY ON THE RESULTING SOURCE. It may be very different from the original version.

list | list-available | la
list-available (abbreviated with la) presents a list of details about specified packages, including installed version, installable versions and recently built binary packages. If the package names are omitted, shows all known packages. With **-v**, prints long package paths.

list-installed | li
Synonym to list alli. Acts like list-available but limits the list to the installed source packages.

search Synonym to list -s. Looks for locally compiled packages first and (if none found) searches for alternative installation candidates with apt-cache.

get get followed by the package list installs the package source, downloading source packages when needed.

build build is followed by one or more source packages that should be built. It chooses the kernel source appropriate for the currently running kernel unless different directories have been specified. If the build fails, look for the most recent log file in /var/cache/modass (or the user-specified location).

install install is followed by one or more packages desired for installation. The last built package for the current running kernel is chosen.

auto-install | a-i
auto-install is followed by one or more packages desired for installation. It will run prepare to configure your system to build packages, get the package source, try to build it for the current kernel and install it. You can use alli or allu shortcuts to select all installed modules source packages or only those that have been unpacked before (similar to the make-kpkg tool normally does, looking in $MODULE_LOC)

auto-build | a-b
like auto-install but does not install the package immediately

clean clean clears the build directories of the kernel packages.

purge purge clears the information cache of a source package and removes all binary packages locally built from it (that module-assistant knows about). USE WITH CARE!

OPTIONS

-t

--text-mode
Show pure build/install/update logs, no progress bars.

-k

--kernel-dir
The kernel source directories to be used for builds. You can specify multiple directories with multiple options or separated by commas or line separators (e.g using **-k "‘echo /usr/src/linux-headers-*‘"**). The kernel versions detected in this directories are automatically added to the list of target kernel versions (see **--kvers-list** for details).

-l

--kvers-list
List of kernel version strings (as in KVERS) to act on. If omitted, the version string of the currently running kernel is inserted. If **--kernel-dir** specifies additional source directories, the kernel versions that belong to them will be inserted too.

The locations of the kernel source (or headers) that belong to this kernel versions are either detected by a lookup in the "usual" locations on Linux systems, or they must be specified with the **--kernel-dir** option.

-v

--verbose
Shows a bit more information, like full paths of the binary packages.

-n

--no-rebuild
If a package that is to be generated does already exist in the target directory (maybe in on older version), **-n** prevents from building the package again.

The default behaviour is to skip when exactly the same file (with the same filename) is to be generated as the one that already exists, and the new filename could be detected before starting the build process (depends on the module package).

-f

--force Never look for target file (in another version) and force a build. For the get command, download a newer version of a package even if it is already installed.

-u

--userdir
All relevant environment variables with paths will be redirected to new directories under the one specified with this option.

-i

--non-inter
When the package build was not successful, just continue with other candidates. By default, module-assistant will suggest to examine the build log. This option may also modify the behaviour of dpkg and apt-get to reduce the need for human interaction and install build dependencies as needed.

-o

--unpack-once
Try to not unpack twice. The option needs to be also specified when the package is being unpacked for the first time. Experimental option, don't rely on it.

-O

--not-unpack
Never unpack the source tarball. Useful after manual manipulation of module source.

-q

--quiet Suppress some of the noisy messages during the processing.

-S

--sudo-cmd
A replacement command for superuser commands to be used instead of sudo.

-s

--apt-search
See search command for details.

-h

--help Prints the usage overview.

ENVIRONMENT VARIABLES

You can export the following environment variables to modify the behaviour of the build scripts. Some packages may ignore them or interpret them differently.

KPKG_DEST_DIR
KPKG_DEST_DIR specify the target directory where the resulting Debian package should be installed into. However, many packages ignore this variable and install the file into the directory above the kernel source directory or above the current directory.

KERNELDIRS
KERNELDIRS specifies or extends the list of kernel source/header directory which m-a should build modules for. See /-k/-Option for details.

SIGNCHANGES
If SIGNCHANGES is set, .changes files will be generated (calling kdist_image rule instead of kdist) and debsign (or gpg or pgp) will be executed to sign the changes.

KMAINT | DEBFULLNAME | DEBNAME
Specifies the realname of the person building the package (interesting for .changes file only)

KEMAIL | DEBEMAIL
Specifies the email address of the person building the package (interesting for .changes file only).

MODULE_LOC
A different location for the (already extracted) module source directories. Default is */usr/src/modules*.

MA_VARDIR
A different location for cached data, used by helper scripts from **module-assistant**. Default is */var/cache/modass*.

MA_APTCMD
Command to download install packages, to use instead of the **apt-get**.

MOD_SRCDIR
A different location for module source tarballs. Default is */usr/src*.

ROOT_CMD
Wrapper command to execute command as root. If you are not root, fakeroot is chosen automatically. This variable must be interpreted by individual packages so some of them may ignore it. However, you can still run module-assistant inside of the ROOT_CMD wrapper.

NON-ROOT USAGE

module-assistant can work without being root. However you won't be able to use apt-get or dpkg to install the packages, and you cannot write to */var/cache/modass* on a normal Debian system. So the commands are **get**, **install**, **auto-install** and **prepare** are taboo for regular users. However, if the sudo program is installed, it will be invoked for apt-get and dpkg operations. All remaining commands except of **list** require additional environment variables to move the target paths to locations writable for the user. They all can be trimmed to a certain location (a writable directory) using the **-u** switch.

FILES

*/usr/share/modass/packages/**
List of helper scripts shipped with the module-assistant package.

*/usr/share/modass/overrides/**
Helper scripts installed by other packages.

SEE ALSO

make-kpkg(1), */usr/share/doc/module-assistant/README*

BUGS

See the module-assistant bug page <URL:http://bugs.debian.org/src:module-assistant>. If you wish to report a bug in module-assistant, please use the **reportbug**(1) command.

RETURN CODES

0 Success

1..249 various errors during the build process

254 problem with permissions

255 fixable error after user intervention

TODO

Quicklist (fast output without details)

Integration into APT and/or into the init system

"Aggressive" debianisation using the templates set (to generate a package with guessed name from any source that looks like being compatible with kernel 2.6 build system)

Automatic transformation of kernel sources to generate .udeb packages

AUTHOR

Module-Assistant was written by Eduard Bloch <blade@debian.org> for the Debian distribution.