DKMS for Developers

John A. Hull
Software Engineer
Dell, Inc.

Novell

What is DKMS?

- Dynamic Kernel Module Support
- Allows individual kernel modules to be upgraded without changing the entire kernel
  - → Add, build, install, remove, and track kernel modules
- Standardized framework for
  - Collecting driver source code
  - Building compiled-module binary files
  - installing/uninstalling modules into Linux kernel
  - Packaging driver source code and binary modules

DKMS Is/Is Not

IS	IS NOT
Distro-agnostic	Tied to specific distro
Architecture agnostic	x86-only
A backport helper	A way to maintain your drivers outside of kernel.org forever
Makes use of Kbuild	Kbuild replacement
2.4 and 2.6 kernels	<= 2.2 kernels

Ideal Use Cases for DKMS Packages

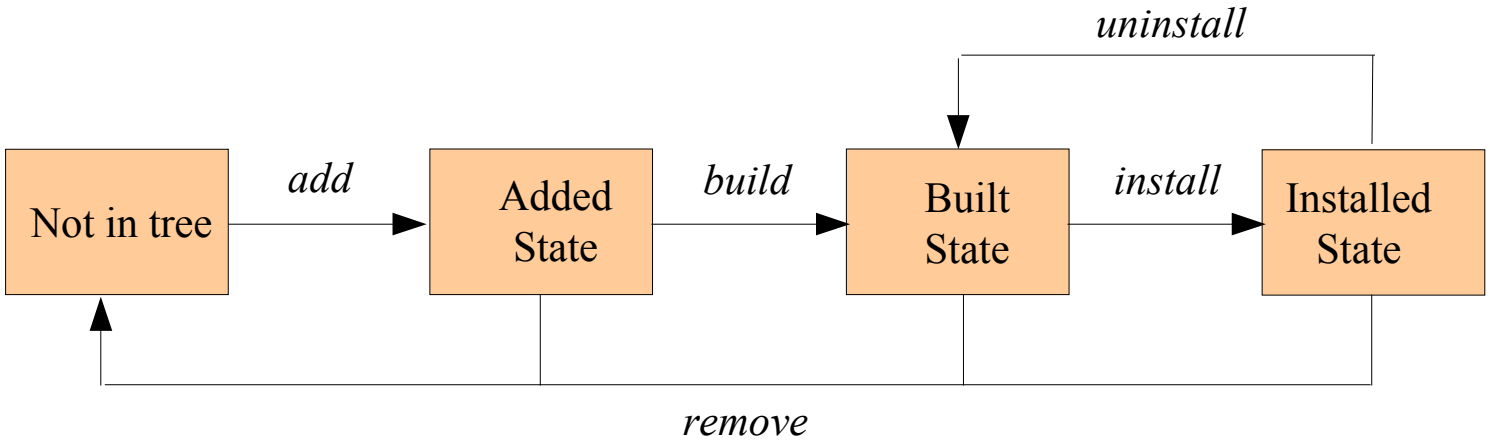
- Driver releases inteded for testers
  - Example: testing new megaraid driver on SLES 9 before submitting to SUSE/kernel.org for inclusion
- Backports for your kernel.org-merged driver intended for users
  - Example: Distributing latest megaraid driver for SLES 9 until accepted into next service pack
- Creating driver disks

DKMS Uses For Developers

- Framework to use on build system
  - Configurable build and source tree locations
  - Uses kernel-provided build mechanism
  - Supports multi-arch builds
- Simplifies testing of updated device drivers
  - Build binary modules from source code outside the kernel source tree
  - No need to recompile kernel
  - Driver update and rollback mechanism - version control
- Simplifies creation of driver RPMs and driver disks

DKMS Concepts

DKMS Life Cycle



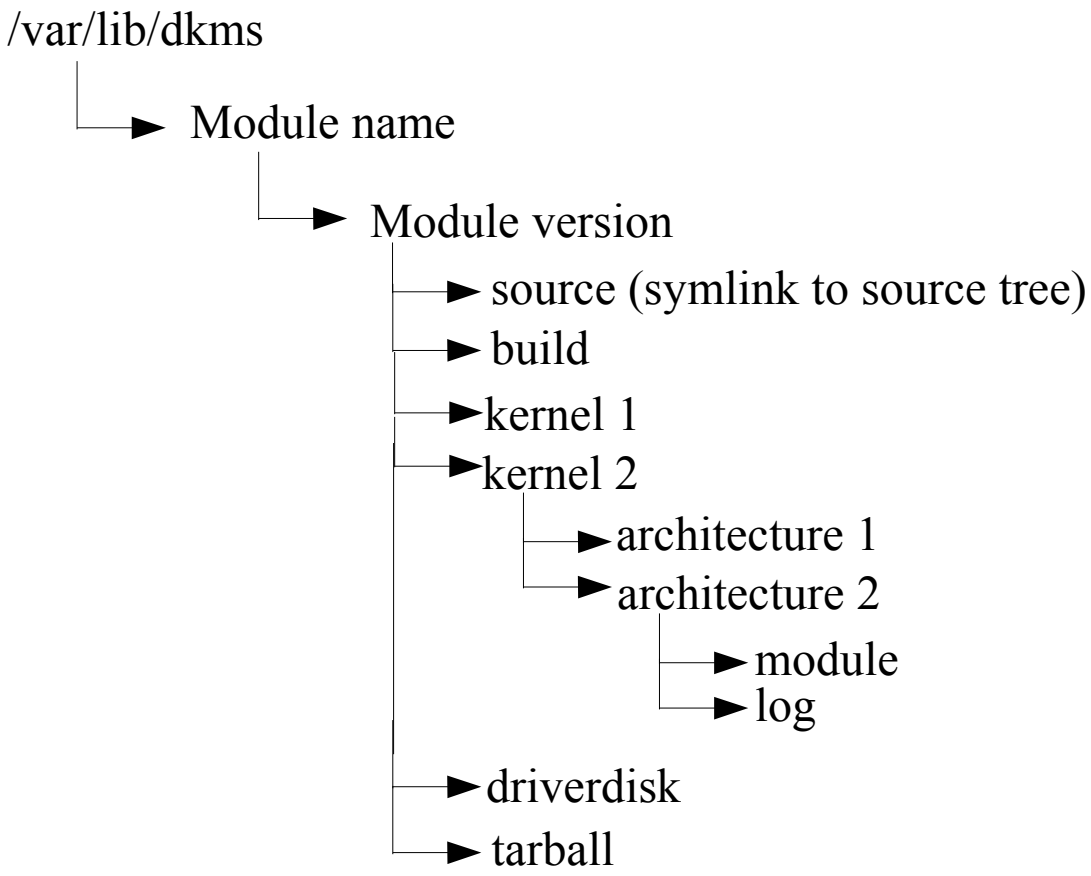
Commands: add, build, install, uninstall, remove, status, match, mktarball, ldtarball, mkdriverdisk mkrpm

dkms.conf

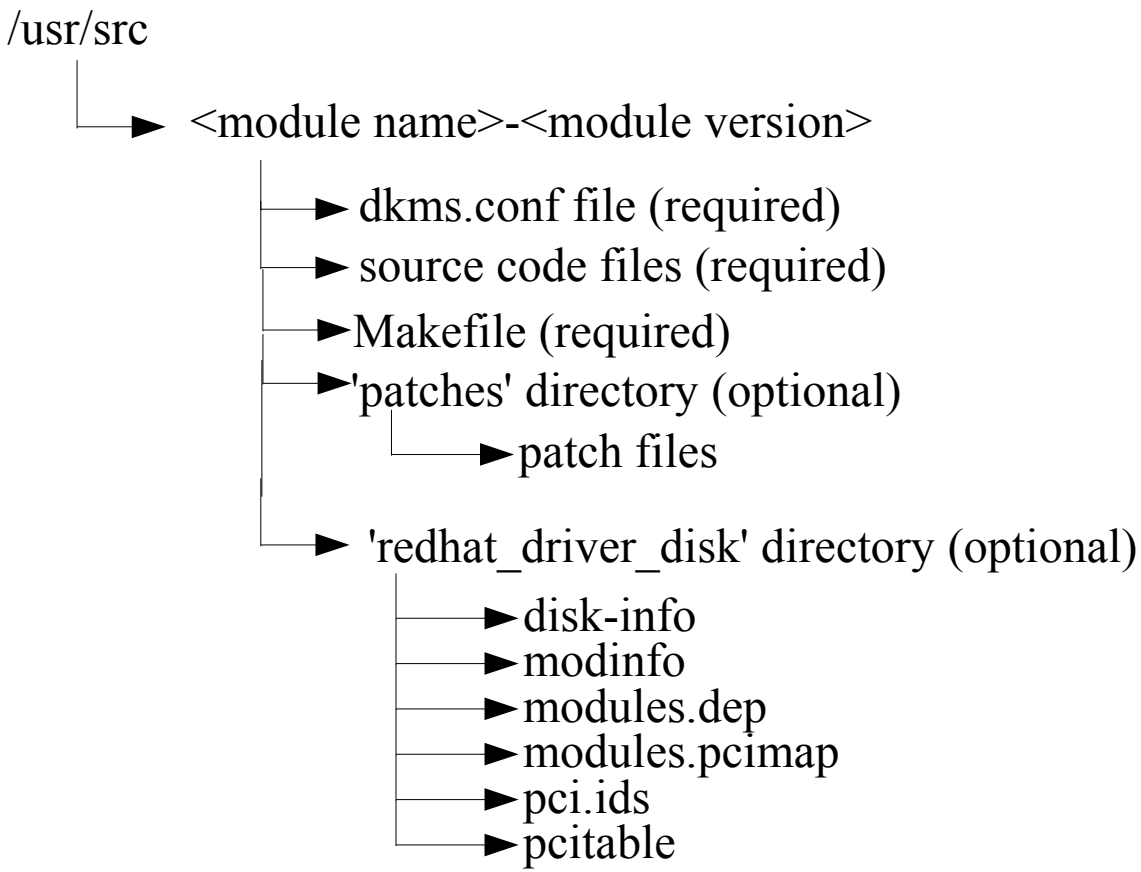
- Gives DKMS the necessary configuration information needed to build, install, and package binary modules
- Minimal file:

```
PACKAGE_NAME="e1000"
PACKAGE_VERSION="5.2.32"
BUILT_MODULE_NAME[0]="e1000"
DEST_MODULE_LOCATION[0]="/kernel/drivers/net"
```

DKMS Tree


/var/lib/dkms
Module name
Module version
source (symlink to source tree)
build
kernel 1
kernel 2
architecture 1
architecture 2
module
log
driverdisk
tarball

Source Tree


/usr/src
<module name>-<module version>
dkms.conf file (required)
source code files (required)
Makefile (required)
'patches' directory (optional)
patch files
'redhat_driver_disk' directory (optional)
disk-info
modinfo
modules.dep
modules.pcimap
pci.ids
pcitable

framework.conf

- Allows user or admin to set system-wide variables for DKMS
  - source tree location (default: /usr/src)
  - DKMS tree location (default: /var/lib/dkms)
  - install tree location (default: /lib/modules)
  - Verbosity level

Using DKMS: An Example

Process for Creating Packages

1. Create and configure source tree
2. Configure dkms.conf file
3. Add module and module version to DKMS tree
4. Build modules for kernel versions
5. Create DKMS tarball
6. Create DKMS RPM
7. Create driver disk

Configure Source Tree

- Create /usr/src/<module name>-<module version>
- Place source code and Makefile into this directory
- Put license file into directory
- If Makefile is from kernel tree:
  - Replace obj-$CONFIGVAR option with obj-m to ensure that module(s) will get built
- If Makefile is custom:
  - Modify to allow DKMS to pass kernel version and kernel source path as variable when performing a build
  - Must not have `uname -r` referenced in body

Source tree example

`ls /usr/src/megaraid-2.20.4.4`

dkms.conf

Kconfig.megaraid

mega_common.h

megaraid_mbox.h

Makefile

megaraid_ioctl.h

megaraid_mm.c

mbox_defs.h

megaraid_mbox.c

megaraid_mm.h

Configure dkms.conf File (1)

- Plain text file that is “sourced in” by DKMS
- Required directives to set:
  - PACKAGE NAME
  - PACKAGE VERSION
  - BUILT_MODULE_NAME[0]
  - DEST_MODULE_LOCATION[0]
- Useful (but optional) directives to set:
  - REMAKE_INITRD=”yes” (for storage device drivers)
  - MODULES_CONF_ALIAS_TYPE[0]

Configure dkms.conf File (2)

- BUILT_MODULE_LOCATION[0]
- DEST_MODULE_NAME[0]
- PATCH
- AUTOINSTALL=”yes”
- MAKE[0]

- MAKE[0] does not have to be set
  - Will use default kernel make command if not set
  - Only should be used for custom Makefile
- Many directives are arrays
- All other directives can be found in DKMS man page

dkms.conf Example

```
PACKAGE_NAME=”megaraid”
PACKAGE_VERSION=”2.20.4.4”
BUILT_MODULE_NAME[0]=”megaraid_mbox”
DEST_MODULE_LOCATION[0]=”/kernel/drivers/scsi/”
BUILT_MODULE_NAME[1]=”megaraid_mm”
DEST_MODULE_LOCATION[1]=”/kernel/drivers/scsi/”
MODULES_CONF_ALIAS_TYPE[0]=”scsi_hostadapter”
REMAKE_INITRD=”yes”
```

Add and Build Modules

- Add the module/version to DKMS tree
  - dkms add -m <module> -v <version>
- Build modules for the necessary kernels
  - dkms build -m <module> -v <version> -k <kernel>
  - Build process output dumped to /var/lib/dkms/<module>/<version>/build/make.log
- `dkms status` shows the status of the modules

dkms status output

`dkms status`

megaraid, 2.20.4.4, 2.6.5-7.97-smp, x86_64: built
megaraid, 2.20.4.4, 2.6.5-7.97-default, x86_64: built
megaraid, 2.20.4.4, 2.6.5-7.97-smp, i586: built
megaraid, 2.20.4.4, 2.6.5-7.97-default, i586: built
megaraid, 2.20.4.4, 2.6.5-7.139-smp, x86_64: built
megaraid, 2.20.4.4, 2.6.5-7.139-default, x86_64: built
megaraid, 2.20.4.4, 2.6.5-7.139-smp, i586: built
megaraid, 2.20.4.4, 2.6.5-7.139-default, i586: built

Create dkms tarball

- Create a dkms tarball
  - dkms mktarball -m <module> -v <version> -k <kernel> -a <arch>
  - Can also specify --source-only or –binaries-only
  - Will grab all specified modules and the source tree
- To load tarball on another system
  - dkms ldtarball --archive=<tarball name>
- Easy way to back up and save work, and to move it between systems
- Can include modules for multiple kernels and architectures in tarball

Create Device Driver RPM

- Create the RPM
  - dkms mkrpm -m <module> -v <version> -k <kernel> -a <arch>
  - Will use **<module>-dkms-rpm.spec** from source tree if exists, otherwise **/etc/dkms/template-dkms-rpm.spec**
  - Creates a dkms tarball, includes in RPM
- When installed, RPM will load DKMS tarball
  - If kernels exist for built modules, installs modules
  - If no modules for running kernel, will build and install
- Can include modules for multiple kernels and architectures in RPM

Create Driver Disk

- Create distribution-specific driver disks (RH or SUSE)
  - dkms mkdriverdisk -d <distro> -r <release> -m <module> -v <version> -k <kernel> -a <arch>
- Recommendations
  - Make multi-arch driver disks
  - Create modules for kernels available on distribution meda (e.g. default, smp, bigsmp kernels)

Example: Creating a DUD and RPM for SLES 9 SP1

```
# dkms add -m megaraid -v 2.20.4.4
# dkms build -m megaraid -v 2.20.4.4 -k 2.6.5-1.139-default -a i586
# dkms build -m megaraid -v 2.20.4.4 -k 2.6.5-1.139-smp -a i586
# dkms build -m megaraid -v 2.20.4.4 -k 2.6.5-1.139-default -a x86_64
# dkms build -m megaraid -v 2.20.4.4 -k 2.6.5-1.139-smp -a x86_64
# dkms mkrpm -m megaraid -v 2.20.4.4 -k 2.6.5-1.139-smp -a x86_64 \
   -k 2.6.5-1.139-default -a x86_64 -k 2.6.5-1.139-smp -a i586 \
   -k 2.6.5-1.139-default -a i586
# dkms mkdriverdisk -m megaraid -v 2.20.4.4 -d suse -r sles9 \
    -k 2.6.5-1.139-smp -a x86_64 -k 2.6.5-1.139-default -a x86_64 \
    -k 2.6.5-1.139-smp -a i586 -k 2.6.5-1.139-default -a i586
```

Troubleshooting

- Double- and triple-check the dkms.conf file
- View module build errors in

  /var/lib/dkms/<module>/<version>/build/make.log
- Check that Makefile accepts kernel source and kernel version info from DKMS
- Check that all required source code is present

Conclusion

- Framework to use on build system
  - Configurable build and source tree locations
  - Uses kernel-provided build mechanism
  - Supports multi-arch builds
- Simplifies testing of updated device drivers
  - Build binary modules from source code outside the kernel source tree
  - No need to recompile kernel
  - Driver update and rollback mechanism - version control
- Simplifies creation of driver RPMs and driver disks

More Information/Status

- Available as a GPL open-source project from:http://linux.dell.com/dkms
- Mailing list: dkms-devel@lists.us.dell.com
- Extensively tested on Red Hat Enterprise Linux and Novell SUSE Linux Enterprise Server; included on Mandrake 10.1 CDs.
- Multiple architectures: Intel x86, AMD64/EM64T, Itanium
- Shipping on every Dell server with Linux.

Backup

Add Command

dkms add -m <module name> -v <module version>

Requirements

- Source code in source tree
  - /usr/src/<module name>-<module version>
- Properly-configured dkms.conf file in source tree

Novell.

Build Command

dkms build -m <module name> -v <module version>

Options:

- -k <kernel version>
- -a <arch>
- --kernelsourcedir <kernel source dir path>
- --config <kernel configfile location>

Requirements

- Module name/version must be in “added” status

install/uninstall Command

dkms install -m <module name> -v <module version>

dkms uninstall -m <module name> -v <module version>

Options:

- -k <kernel version>
- -a <arch>

Requirements

- Driver module must be in built state for kernel/architecture combo

mkdriverdisk Command (Red Hat)

dkms mkdriverdisk -d <distro> -m <module name> -v <module version> -k <kernel version> -a <arch>

Options:

- <distro> = 'redhat' allows 1 arch on disk
- <distro> = 'redhat2' allows multiples arches on disk

Requirements

- Driver modules must be in built state for kernel/architecture combo
- For Red Hat driver disks, redhat_driver_disk must exist and include disk-info, modinfo, modules.dep, modules.pcimap, pci.ids, and pcitable

mkdriverdisk Command (SUSE)

dkms mkdriverdisk -d suse -r <release> -m <module name> -v <module version> -k <kernel version> -a <arch>

Options:

- <release> is the version of SUSE, i.e. 'sles9' for enterprise, or '9.2' for professional releases

Requirements

- Driver modules must be in built state for kernel/architecture combo

mkrpm Command

dkms mkrpm -m <module name> -v <module version> -k <kernel version> -a <arch>

Options:

- Can specify more than one kernel/arch combo

Requirements

- Driver modules must be in built state for kernel/architecture combo

mktarball Command

dkms mktarball -m <module name> -v <module version>

Options:

- -k <kernel version>, -a <arch>
- --binaries-only
- --source-only

Requirements

- module/version must be in added state

AUTOINSTALL directive

- dkms_autoinstaller service provided by DKMS, starts on boot
- If AUTOINSTALL=”yes” set in dkms.conf, the service will check if module is built for current kernel
  - If module not built, service will build and install
- **WHEN TO USE**
  - Only for non-storage device drivers
  - Only for device driver that will not be accepted by distro in a newer kernel

Standard DKMS make commands

- 2.4 kernels:

make -C $kernel_source_dir SUBDIRS=$dkms_tree/$module/$module_version/build modules

-

- 2.6 kernels:

make -C $kernel_source_dir M= $dkms_tree/$module/$module_version/build

Novell