银河麒麟桌面操作系统 DKMS 包
制作流程及说明

编写人：　易腊梅	编写日期：　　2024-10-15
审核人：	审核日期：
批准人：	批准日期：

麒麟软件有限公司

生态与技术服务中心

2024 年 10 月 15 日

版本说明

版本号	版本说明	作者	日期	变更内容
V1.0	首次发布	易腊梅	2024-10-15	首次创建
V1.1	修改	易腊梅	2025-1-9	添加 3.2.2

目录

1 DKMS 功能

DKMS 全称是 DynamicKernel ModuleSupport，是用来生成 Linux 内核模块的一个框架，它可以帮我们维护内核外的驱动程序，这种模块的源代码一般不在 Linux 内核源代码树中。当新的内核安装时，DKMS 支持的内核模块会自动重建。

2 DKMS 包名规范

dkms 软件包的命名规范需遵循《银河麒麟桌面操作系统 V10-DEB 包打包规范》，在此基础上 DKMS 包的命名方式是需要在源码包名加"-dkms"后缀，即 packagename-dkms。

全名格式：packagename-dkms_version_platform.deb

长度要求：包全名长度最大为 255 个字符。

packagename 应包含驱动模块的设备类型、厂商、驱动及型号信息，命名规范为：{设备类型}-driver-厂商-{型号或系列}，示例如 net-driver-rtk-811B。如果该驱动包适用于此厂商该设备类型的所有型号，则命名规范建议为:{设备类型}-driver-厂商-{common}，示例如 net-driver-rtk-common。

2.1 依赖

依赖的包应该是原来软件包的基础上，加上 dkms 包及目标版本内核头文件 linux-headers。

2.2 源代码构建位置

DKMS 要求我们的代码目录必须以-的格式命名，构建模块所需源代码需要放在 /usr/src/MODULE_NAME-MODULE_VERSION 目录下，此目录是 DKMS 构建模块时使用的默认目录。

2.3 DKMS 命令

dkms 使用命令格式

```
dkms [action] [options] [module/module-version] [/path/to/source-tree] [/path/to/tarball.tar] [/path/to/driver.rpm]
```

2.3.1 ACTIONS

- 查看 DKMS 模块状态

```
dkms status [module/module-version] [-k kernel/arch]
```

查看 DKMS 模块状态，如若未指定 module 等信息，则默认显示所有 dkms 模块状态，模块状态有 added,built 及 installed 三种状态。

- 添加 DKMS 模块

```
dkms add [module/module-version] [/path/to/source-tree] [/path/to/tarball.tar]
```

- 移除 DKMS 模块

```
dkms remove [module/module-version] [-k kernel/arch] [--all]
```

--all 参数是将指定模块从所有内核中移除，-k <kernel-version>参数移除与指定内核版本匹配的模块。

- 编译 DKMS 模块

```
dkms build [module/module-version] [-k kernel/arch]
```

为指定内核编译 module/version，如若未指定-k 选项，则默认为当前运行的的内核和架构。

- 安装 DKMS 模块

```
dkms install [module/module-version] [-k kernel/arch] [/path/to/driver.rpm]
```

安装指定模块，如果未指定内核和架构，则默认安装到当前正在运行的内核中。如果当前模块未添加或被编译，使用 install 命令，dkms 则会去尝试添加和编译，如果添加和编译成功，dkms 则会安装此模块。

- 重新构建内核模块

```
dkms autoinstall
```

为当前内核重新构建所有的 dkms 模块。

- 卸载 DKMS 模块

```
dkms uninstall [module/module-version] [-k kernel/arch]
```

从指定内核中卸载已安装的模块，如果未指定-k 参数，则默认卸载在当前运行内核上卸载此模块。卸载的模块仍然处于 build 状态，如果需要完全删除此驱动，则需使用 remove 命令删除。

- 使用 DKMS 制作模块 deb 包

```
dkms mkdeb [module/module-version] [-k kernel/arch]
```

此命令是为指定版本的模块创建 deb 二进制包。它使用/etc/dkms/template-dkms-mkdeb 中的模板 debian 目录作为包的基础。如果 DKMS 找到一个名为/usr/src/<module>-<module-version>/<module>-dkms-mkdeb 的文件，它将使用该文件夹。

- 创建 DKMS 模块源代码压缩包

```
dkms mktarball [module/module-version] [-k kernel/arch] [--archive /path/to/tarball.tar] [--source-only] [--binaries-only]
```

此命令将会创建指定内核模块的压缩包，包括源码及编译后的模块。可以指定一个内核来进行压缩，也可以针对多个内核来对模块进行压缩，如-k kernel1/arch1 -k kernel2/arch2。可以使用--archive 参数来指定要打包到压缩包的文件。如果想生成只含二进制文件的压缩包，则使用--binaries-only。如果想生成只包含源代码的压缩文件，则使用--source-only。【注意】若只压缩二进制包则需在 build 之后才能使用此命令创建压缩包。

- 使用 DKMS 加载 tarball

```
dkms ldtarball [/path/to/tarball.tar] [--force]
```

此命令可以加载 mktarball 命令打包的压缩包到 DKMS 树，加载后可以使用 dkms tatus 命令查看到对应的模块，此模块处于 built 状态。如果加载的模块文件目录已存在，则会发出警告并终止加载。如需覆盖，可使用--force。

2.3.2 OPTIONS

常用 options：

参数	说明
-m <module>/<module-version>	指定模块名字及版本
-v <module-version>	指定模块版本，只能跟-m 搭配使用（且-m 未指定模块版本），如-m hello -v 0.1
-k <kernel-version>/<arch>	指定内核版本及架构，可使用多个-k 参数来指定多内核及架构；

-a，--arch	指定系统架构，与-k 使用，如未指定则默认使用当前运行系统的架构。可使用多个-a 参数指定多个架构，但每个-a 参数均需与-k 配套使用，即-个-a 参数对应一个-k 参数，系统自动将第一个-a 参数与第一个-k 参数匹配在一起，以此类推；
-V, --version	打印当前已安装 dkms 版本并退出

2.4 dkms.conf 配置文件

在软件源码目录下，要包含一个 dkms.conf 配置文件，告诉 DKMS 如何编译。dkms.conf 配置文件常用参数见下：

- PACKAGE_NAME ：此选项指定模块名称，此名称与 dkms 添加，编译等命令时使用的-m 后的名称保持一致。此选项为 dkms.conf 的必填项；
- PACKAGE_VERSION ：指定模块的软件版本，此选项为 dkms.conf 的必填项；
- CLEAN：用于指定用于在构建模块前后进行清理的命令，如未设置，则默认为”make clean”命令；
- PATCH[#]：使用 PATCH 数组来指定补丁，每个 PATCH 都需要指定要应用的补丁文件名。且所有的补丁包都需要放置在源目录的补丁子目录 /usr/src/<module>-<module-version>/patches/ 。如果想使用指定的 patch，则需要用 PATCH_MATCH[#]配套使用，PATCH_MATCH[#]中指定相应的正则表达式，如果正则表达式与当前构建模块的内核匹配，则提醒 dkms 仅使用该 PATCH[#]。如若没有任何补丁被成功应用，则停止编译，编译失败信息会记录到 /var/lib/dkms/<module>/<module-version>/build/中；
- PATCH_MATCH[#]：参照 PATCH[#]指令说明。如只想在某些情况下应用补丁，则需要在编译之前使用 PATCH_MATCH[#]来指定正则表达式，该正则表达式指定与此 patch 匹配的内核，如当前内核与此匹配，则应用这个补丁；
- AUTOINSTALL：当设置为“yes”时，则/etc/rc.d/init.d/dkms_autoinstaller 服务会自动尝试在启动的内核上安装此模块。dkms_autoinstaller 服务是在系统引导时检查当前内核版本和已安装的 DKMS 模块，如果发现有模块需要更新，则会触发重新编译和安装操作，确保系统在内核更新后仍然能够正常使用这些模块。需要了解更多，可

阅读 dkms_autoinstaller 说明；

- MAKE[0]：用来设定编译的命令，默认的 make 命令放在 MAKE[0]中。MAKE[#]数组中的其他编译命令只有在与 MAKE_MATCH[#]匹配时才会使用。如果 MAKE_MATCH[#]中没有为任何 MAKE[#](#>0)设置正则表达式，则忽略该 MAKE[#]指令。如果多个 MAKE_MATCH 指令与正在构建的内核匹配，则最后一个匹配的 MAKE[#]将用于构建模块。如果没有指定 MAKE 指令，或者如果没有 MAKE_MATCH 与正在构建的内核匹配，DKMS 将尝试使用通用的 MAKE 命令来构建模块。一般情况下是不用设定的；
- MAKE_MATCH[#]：请参阅 MAKE[#]说明。此数组指定正则表达式，当与正在构建的内核匹配时，则使用执行对应 MAKE[#]指令来构建模块；
- BUILT_MODULE_NAME[#]：用来指定模块的名称。如果 DKMS 模块包中包含多个要安装的模块时，则必须对所有模块编写此选项。这个选项内容不需要包含.o 或者.ko 信息。需要注意的是选项 BUILT_MODULE_NAME，BUILT_MODULE_LOCATION，DEST_MODULE_NAME 以及 DEST_MODULE_LOCATION 的编号（也就是#）必须保持一致，且必须从 0 开始（如 BUILT_MODULE_NAME[0]="qla2200" BUILT_MODULE_NAME[1]="qla2300"）；
- BUILT_MODULE_LOCATION[#]=这个选项是告诉 DKMS 编译后的模块放在哪个路径。设置此参数时，需要根据驱动源码的目录结构来设置，应该设置为被编译驱动模块.o 文件所在的目录，此目录路径是相对于 dkms.conf 所在目录的相对路径（如 helloworld 目录下存在 dkms.conf，Makefile，src 目录。src 目录下为 helloworld.c 及 Makefile 文件，那么 BUILT_MODULE_NAME[0]=src）。如果未设置，DKMS 会在源文件的根目录下查找你的#号模块。请注意，对于 dkms 包中的每个模块，#的数值必须与 BUILT_MODULE_NAME 、BUILT_MODULE_LOCATION 、DEST_MODULE_NAME 和 DEST_MODULE_LOCATION 中的每一个相同，并且编号应从 0 开始（例如 BUILT_MODULE_LOCATION[0]= “ some/dir/ ” BUILT_MODULE_LOCATION[1]= “other/dir/”）;如若指定此选项，ko 文件存放到 /var/lib/dkms/MODULE-NAME/MODULE-VERSION/`uname -r`/ARCH/module/MODULE-NAME.ko；
- DEST_MODULE_NAME[#]：此指令可用于指定应安装的模块的名称。这将把模块从 BUILT_module_NAME[#]重命名为 DEST_module_NAME[#]。此指令不应明确包

含任何尾随的“.o”或“.ko”。如果未设置，则假定其值与 BUILT_MODULE_NAME[#]相同。请注意，对于 dkms 包内的每个模块，#的数值必须与 BUILT_module_NAME 中的每个模块相同， BUILT_MODULE_LOCATION、DEST_MODULE_NAME 和 DEST_MODULE_LOCATION，编号应从 0 开始（例如，DEST_MMODULE_NAME[0]=“qla2200_6x” DEST_MOULE_NAME[1]=“qla300_6x”）。

- DEST_MODULE_LOCATION[#]：此指令指定模块编译后应安装到的目标位置，设置的路径是/lib/modules/$(uname -r)/kernel/目录的相对路径，且路径后面必须加上/，如设置 DEST_MODULE_LOCATION[0]=/drivers/scsi/，则表示 0 号模块安装到 /lib/modules/$(uname -r)/kernel/drivers/scsi/目录下。如果未设置，则默认为 /lib/modules/$(uname -r)/updates/目录下。请注意，对于 dkms 包内的每个模块，#的数值必须与 BUILT_MODULE_NAME 中的每个模块相同，
  BUILT_MODULE_LOCATION，DEST_MODULE_NAME 以及
  DEST_MODULE_LOCATION 的编号必须从 0 开始（比如，
  DEST_MODULE_LOCATION[0]="/kernel/drivers/something/"
  DEST_MODULE_LOCATION[1]="/kernel/drivers/other/")；

3 dkms 使用实例

3.1 安装 dkms 包

在使用 dkms 构建模块前，需保证系统内已安装 dkms 包及目标版本内核头文件 linux-headers 相关包。

```
sudo apt-get install dkms
sudo apt-get install linux-headers-VERSION linux-headers-VERSION-generic
```

3.2 准备源码

3.2.1 新研项目源码

本实例构建 helloworld 模块，源码目录为 helloworld-1.2，目录内有 helloworld.c 及 Makefile 文件，相关文件代码见下 helloworld.c：

```
//helloworld.c
```

```
#include <linux/kernel.h>
#include <linux/module.h>
//内核模块初始化函数
static int __init hello_init(void)
{
    printk(KERN_INFO "DKMS-Hello World enter\n");
    return 0;
}
//内核模块退出函数
static void __exit hello_exit(void)
{
     printk(KERN_INFO "DKMS-Hello World exit\n");
}
module_init(hello_init);
module_exit(hello_exit);

MODULE_LICENSE("GPL");
```

Makefile 文件

```
obj-m := helloworld.o
all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

接下来我们需要准备 dkms.conf 文件并放到 helloworld-1.2 目录下，dkms.conf 文件内容见下：

```
PACKAGE_NAME=“helloworld”
PACKAGE_VERSION="1.2"
CLEAN="makeclean"
MAKE[0]="makeall KVERSION=$kernelver"
```

```
BUILT_MODULE_NAME[0]="helloworld"
DEST_MODULE_LOCATION[0]="/updates"
AUTOINSTALL="yes"
```

准备好以上文件后，将 helloworld-1.2 目录拷贝至/usr/src/目录下。

3.2.2 已有项目源码

本示例在某网卡厂商（下称 X 网卡）已有驱动源码基础上，新增 dkms.conf，Makefile 文件。

以 X 网卡驱动 XXX_1.1.13 版本为例，原有源码目录结构见下（目录结构中涉及到网卡名称已用 XXX 替代，并隐去部分目录和文件）：

```
kylin@kylin-pc:~/XXX_1.1.13$ tree
.
├── common
│   ├── kcompat.c
│   ├── XXX_api.c
│   ├── XXX_common.c
│   ├── XXX_debugfs.c
│   ├── XXX_macsec.c
│   ├── XXX_param.c
│   ├── XXX_procfs.c
│   ├── XXX_ptp.c
│   ├── XXX_sriov.c
│   ├── XXX_sysfs.c
│   └── XXX_westlake.c
├── include
│   ├── kcompat_defs.h
│   ├── kcompat_gcc.h
│   ├── kcompat.h
│   ├── kcompat_impl.h
│   ├── kcompat_openeuler_defs.h
```

```
|   ├── kcompat_oracle_defs.h
|   ├── kcompat_overflow.h
|   ├── kcompat_rhel_defs.h
|   ├── kcompat_sles_defs.h
|   ├── kcompat_std_defs.h
|   ├── kcompat_ubuntu_defs.h
|   ├── XXX_api.h
|   ├── XXX_common.h
|   ├── XXX_debug.h
|   ├── XXX.h
|   ├── XXX_macsec.h
|   ├── XXX_macsec_struct.h
|   ├── XXX_osdep2.h
|   ├── XXX_osdep.h
|   ├── XXX_register.h
|   ├── XXX_sriov.h
|   ├── XXX_type.h
|   └── XXX_westlake.h
├── script
|   └── XXX_xelmem
|       ├── dump_phy.sh
|       ├── Makefile
|       ├── run.sh
|       └── XXX_xelmem.c
└── sdk
    ├── common.mk
    ├── Makefile
    ├── XXX_ethtool.c
    ├── XXX_lib.c
```

```
        ├── XXX_main.c
        ├── XXX_txrx_common.h
        ├── XXX_version.h
        └── XXX_xsk.c


    6 directories, 48 files
```

在 XXX_1.1.13 目录下新增 dkms.conf 文件，由于 XXX 编译目录在 sdk 目录，故 BUILT_MODULE_LOCATION[0]="sdk"，详细内容参考如下：

```
    PACKAGE_NAME="XXX"
    PACKAGE_VERSION="1.1.13"
    AUTOINSTALL="yes"
    BUILD_EXCLUSIVE_KERNEL="^5.4.*|^5.10.*"
    BUILT_MODULE_NAME[0]="XXX"
    BUILT_MODULE_LOCATION[0]="sdk"
    DEST_MODULE_LOCATION[0]="/updates/drivers/net/ethernet/XXX/XXX"
    # Find out how many CPU cores can be use if we pass appropriate -j option to make.
    # DKMS could use all cores on multicore systems to build the kernel module.
    num_cpu_cores()
    {
        if [ -x /usr/bin/nproc ]; then
            nproc
        else
            echo "1"
        fi
    }
    MAKE[0]="unset KERNELRELEASE;make -j$(num_cpu_cores)"
```

【注】上述 dkms.conf 文件中使用 XXX 替代了网卡名称。

继续在 XXX_1.1.13 目录下新增 Makefile 文件，此 Makefile 文件调用 sdk 目录下的 Makefile 文件并进行后续构建，参考内容见下：

```
SDK_DIR := sdk
all:
  @echo "Building in $(SDK_DIR) $(MAKE)..."
  $(MAKE) -C $(SDK_DIR)
clean:
  @echo "Cleaning in $(SDK_DIR)..."
  $(MAKE) -C $(SDK_DIR) clean
```

3.3 使用 dkms 管理 helloworld 模块

3.3.1 添加代码

将源代码添加到 dkms：

```
sudo dkms add helloworld/1.2
或者
sudo dkms add -m helloworld -v 1.2
```

以上两种命令模式执行效果相同，下同，后不再赘述。

```
kylin@kylin-pc:~$ sudo dkms add helloworld/1.2
Creating symlink /var/lib/dkms/helloworld/1.2/source ->
                        /usr/src/helloworld-1.2
DKMS: add completed.
```

3.3.2 查看状态

查看 helloworld 模块状态。

```
kylin@kylin-pc:~$ dkms status
helloworld, 1.2: added
```

3.3.3 编译

编译 helloworld 模块

```
kylin@kylin-pc:~$ sudo dkms build -m helloworld -v 1.2
Kernel preparation unnecessary for this kernel.  Skipping...
```

```
Building module:
cleaning build area...(bad exit status: 127)
make -j4 KERNELRELEASE=5.4.18-110-genericall KVERSION=5.4.18-110-generic...
Signing module:
 - /var/lib/dkms/helloworld/1.2/5.4.18-110-generic/x86_64/module/helloworld.ko
Secure Boot not enabled on this system.
cleaning build area...(bad exit status: 127)
DKMS: build completed.
```

3.3.4 安装

安装 helloworld 模块

```
kylin@kylin-pc:~$ sudo dkms install helloworld/1.2
helloworld.ko:
Running module version sanity check.
 - Original module
    - No original module exists within this kernel
 - Installation
    - Installing to /lib/modules/5.4.18-110-generic/updates/
depmod...
DKMS: install completed.
```

3.3.5 卸载

卸载 helloworld 模块

```
ylin@kylin-pc:~$ sudo dkms uninstall helloworld/1.2
-------- Uninstall Beginning --------
Module:  helloworld
Version: 1.2
Kernel:  5.4.18-110-generic (x86_64)
```

```
-------------------------------------
Status: Before uninstall, this module version was ACTIVE on this kernel.
helloworld.ko:
 - Uninstallation
   - Deleting from: /lib/modules/5.4.18-110-generic/updates/
 - Original module
   - No original module was found for this module on this kernel.
   - Use the dkms install command to reinstall any previous module version.
depmod...
DKMS: uninstall completed.
```

3.3.6 删除

删除 helloworld 模块

```
kylin@kylin-pc:~$ sudo dkms remove helloworld/1.2 --all
-------- Uninstall Beginning --------
Module:  helloworld
Version: 1.2
Kernel:  5.4.18-110-generic (x86_64)
-------------------------------------
Status: This module version was INACTIVE for this kernel.
depmod...
DKMS: uninstall completed.
------------------------------
Deleting module version: 1.2
completely from the DKMS tree.
------------------------------
Done.
```

3.4 使用 dkms 将源码编译成 deb 包

安装依赖包

```
sudo apt-get install dh-make libdigest-md5-file-perl
```

制作 deb 包

```
kylin@kylin-pc:~$ sudo dkms mkdeb helloworld/1.2
或
kylin@kylin-pc:~$ sudo dkms mkdeb helloworld/1.2
Using /etc/dkms/template-dkms-mkdeb
copying template...
.........
DKMS: mkdeb completed.
Moving built files to /var/lib/dkms/helloworld/1.2/deb...
Cleaning up temporary files...
```

查看打包成 deb 的 helloworld 模块包

```
kylin@kylin-pc:/var/lib/dkms/helloworld/1.2/deb$ ls
helloworld-dkms_1.2_amd64.deb
```

制作好的 deb 包在/var/lib/dkms/helloworld/1.2/deb/下。

【注意】使用 dkms mkdeb 命令打包需确保 helloworld 已添加到 dkms 树。

3.5 dkms 驱动包验证

3.5.1 安装验证

安装 dkms 驱动 deb 包:

```
kylin@kylin-pc:~$ sudo dpkg -i helloworld-dkms_1.2_amd64.deb
正在选中未选择的软件包 helloworld-dkms。
(正在读取数据库 ... 系统当前共安装有 207945 个文件和目录。)
准备解压 helloworld-dkms_1.2_amd64.deb ...
正在解压 helloworld-dkms (1.2) ...
正在设置 helloworld-dkms (1.2) ...
```

```
Loading new helloworld-1.2 DKMS files...
Building for 5.4.18-116-generic
Building for architecture x86_64
Building initial module for 5.4.18-116-generic
Secure Boot not enabled on this system.
.....
depmod....
DKMS: install completed.
```

【注】篇幅限制，省略部分打印信息。

查看安装后驱动状态

```
kylin@kylin-pc:~$ dkms status | grep helloworld
helloworld, 1.2, 5.4.18-116-generic, x86_64: installed
```

记载 helloworld 模块并查看内核打印信息

```
kylin@kylin-pc:~$ sudo modprobe helloworld
kylin@kylin-pc:~$ lsmod | grep helloworld
helloworld                16384  0
kylin@kylin-pc:~$ dmesg
[ 1870.569029] DKMS-Hello World enter
```

3.5.2 卸载验证

卸载 dkms 驱动

```
kylin@kylin-pc:~$ sudo dpkg -P helloworld-dkms
(正在读取数据库 ... 系统当前共安装有 207951 个文件和目录。)
正在卸载 helloworld-dkms (1.2) ...
-------- Uninstall Beginning --------
Module:  helloworld
Version: 1.2
Kernel:  5.4.18-116-generic (x86_64)
.............
depmod...
```

```
DKMS: uninstall completed.
------------------------------
Deleting module version: 1.2
completely from the DKMS tree.
------------------------------
Done.
```

【注】篇幅限制，省略部分打印信息。

查看卸载 deb 包后的 dkms 状态

```
kylin@kylin-pc:~$ dkms status | grep helloworld
```

执行上述命令无打印信息。

查看卸载 deb 包后 dmesg 打印信息

```
kylin@kylin-pc:~$ dmesg
.....
[ 2124.812349] DKMS-Hello World exit
```

【注】篇幅限制，省略部分打印信息。

3.5.3 内核升级验证

本实例实验环境原有内核版本为 5.4.18-116，升级的内核版本为 5.4.18-122。安装 5.4.18-122 内核相关包：

```
kylin@kylin-pc:/media/kylin/kylin/内核$ sudo dpkg -i *.deb
输入密码
正在选中未选择的软件包 linux-headers-5.4.18-122。
(正在读取数据库 ... 系统当前共安装有 207951 个文件和目录。)
准备解压 linux-headers-5.4.18-122_5.4.18-122.111_all.deb ...
正在解压 linux-headers-5.4.18-122 (5.4.18-122.111) ...
正在选中未选择的软件包 linux-headers-5.4.18-122-generic。
.....
正在设置 linux-headers-5.4.18-122-generic (5.4.18-122.111) ...
/etc/kernel/header_postinst.d/dkms:
```

```
 * dkms: running auto installation service for kernel 5.4.18-122-generic

Kernel preparation unnecessary for this kernel.  Skipping...

Running the pre_build script:
checking for a BSD-compatible install... /bin/install -c
.....

Building module:
cleaning build area...(bad exit status: 127)
make -j4 KERNELRELEASE=5.4.18-122-genericall
KVERSION=5.4.18-122-generic...
Signing module:
 -
/var/lib/dkms/helloworld/1.2/5.4.18-122-generic/x86_64/module/helloworld.ko
Secure Boot not enabled on this system.
cleaning build area...(bad exit status: 127)

DKMS: build completed.

helloworld.ko:
Running module version sanity check.
 - Original module
   - No original module exists within this kernel
 - Installation
   - Installing to /lib/modules/5.4.18-122-generic/updates/

depmod...
```

```
DKMS: install completed.
.....
正在生成 grub 配置文件 ...
找到主题：/usr/share/grub/themes/UKUI/theme.txt
找到 Linux 镜像：/boot/vmlinuz-5.4.18-122-generic
找到 initrd 镜像：/boot/initrd.img-5.4.18-122-generic
找到 Linux 镜像：/boot/vmlinuz-5.4.18-116-generic
找到 initrd 镜像：/boot/initrd.img-5.4.18-116-generic
找到 initrd 镜像：/boot/initrd.img-5.4.18-122-generic
完成
```

【注】篇幅限制，省略部分安装打印信息。

重启系统后查看 dkms 驱动包情况

```
kylin@kylin-pc:~/桌面$ dkms status
amdgpu, 5.13.20.5.1-1395274, 5.4.18-116-generic, amd64: installed
amdgpu, 5.13.20.5.1-1395274, 5.4.18-116-generic, x86_64: installed (original_module exists)
.....
helloworld, 1.2, 5.4.18-116-generic, x86_64: installed
helloworld, 1.2, 5.4.18-122-generic, x86_64: installed
netflow_info, 1.0, 5.4.18-116-generic, x86_64: installed
netflow_info, 1.0, 5.4.18-122-generic, x86_64: installed
netflow_payload, 1.0, 5.4.18-116-generic, x86_64: installed
netflow_payload, 1.0, 5.4.18-122-generic, x86_64: installed
.....
```

重启系统，切换为 5.4.18-122 内核后加载 helloworld 模块并查看打印信息：

```
kylin@kylin-pc:~$ uname -r
5.4.18-122-generic
kylin@kylin-pc:~$ sudo modprobe helloworld
kylin@kylin-pc:~$ lsmod | grep helloworld
```

```
helloworld              16384  0
kylin@kylin-pc:~$ dmesg
[ 1870.569029] DKMS-Hello World enter
```