银河麒麟桌面操作系统非 DKMS 驱动
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

目录

1 非 dkms 驱动范围

非 dkms 驱动包括核内及核外驱动包。核内驱动跟随内核整编，命名、打包等规范与内核规范保持一致。本文档仅规范非 dkms 核外驱动打包成 deb 流程。

2 非 dkms 核外驱动包名规范

非 dkms 软件包需遵循《银河麒麟桌面操作系统 V10-DEB 包打包规范》。

全名格式：packagename_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。

3 非 dkms 核外模块编译成 deb 的流程

非 dkms 驱动包有编入内核及核外编译两种编译方式。

核内编译主要是将驱动源码添加到内核后整编内核，涉及内核维护和更新，此处不做过多介绍。接下来重点介绍核外编译驱动模块。

3.1 源码编译成 ko 模块

核外编译打包成 deb 包的流程包含两个步骤。首先需要将驱动编译成核外驱动，之后再将编译好的.ko 打包成 deb 包。

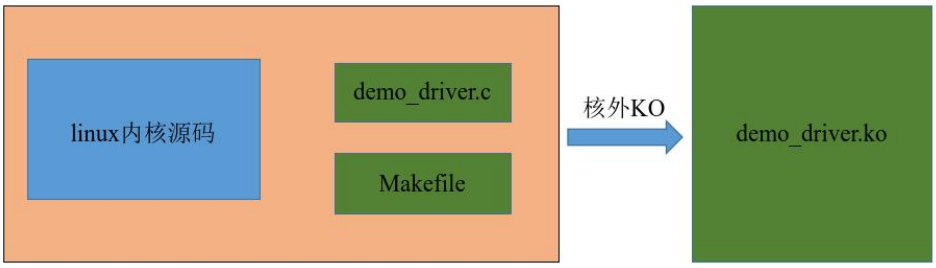

图 1 核外驱动编译模式

如果要将驱动作为内核的外部模块编译，主要通过 Makefile 中的 obj-[x]和 make modules 指令来处理。

3.1.1 obj-[x]说明

obj-[x]用于描述需要编译的目标的属性。

- obj-y：表示将指定对象编译为内核的一部分（即静态链接），这意味着该对象将包含在内核映像中，无法卸载或修改。通常情况下，obj-y 用于编译内核的核心组件，例如进程调度器、文件系统、网络协议栈等。
- obj-m：表示将指定对象编译为可加载模块（即动态链接），这意味着该对象将作为单独的模块编译，并在需要时加载到内核中。通常情况下，obj-m 用于编译内核的外围组件，例如设备驱动程序、文件系统模块、网络协议栈模块等。
- obj-$(CONFIG_YOUR_DEF): 同时也可是一个变量，例如放在 Kconfig 中共用户配置，本质还是 obj-y 和 obj-m

3.1.2 make 命令说明

Makefile 中编译内核模块指令为 make modules ，该指令的功能是编译内核中所有配置为模块的程序得到模块 ko 文件，make modules 命令只能在内核源码顶层目录下执行。

make modules 是编译所有的内核模块，如何单独编译一个指定的模块呢？可增加 M 参数，如：make -C $(KDIR) M=`pwd` modules

-C：$KDIR 内核源代码所在的目录。“make”在执行时实际上会切换到指定的目录，并在完成时切回。

-M: 选项用于指定模块源代码目录，让命令在指定的目录下查找模块源代码(pwd 获取当前目录路径)，编译生成 ko 文件

modules: 参数用于指定要编译的目标是模块文件。在 Linux 内核编译中，make 命令可以使用 modules 参数来指定只编译模块文件，而不编译整个内核。

3.2 将 ko 模块打包成 deb

需要制作 package 包的 deb 包，则需要创建 package 目录，在此目录下有两部分主要目录：DEBIAN 和软件具体安装目录（如 etc，usr，opt，lib，tmp 等）。

3.2.1 DEBIAN 目录说明

DEBIAN 目录下有主要文件和其他文件。

- 主要文件:
  - control——软件包的元数据(依赖包,之类)
  - rules——规定了如何构建软件包
  - copyright——软件包的版权信息
  - changelog——Debian 软件包的更新历史记录
- 其他文件:
  - 兼容文件
  - 监测文件
  - dh_install* 的保存目录（*.dirs, *.docs, *.manpages, ...）
  - 维护脚本（*.postinst, *.prerm, ...）
  - 源码/格式
  - 补丁（如您需要修改上游源代码）

主要文件中主要介绍 control 文件。control 文件主要描述软件包的名称（Package），版本（Version），Installed-Size（大小），Maintainer（打包人和联系方式）以及描述（Description）等，是 deb 包必须具备的描述性文件，以便于软件的安装管理和索引。

表 1 control 字段说明

字段	用途	示例
Package	程序名称	中间不能有空格
Version	软件版本	1.0.0
Description	程序说明	-
Section	软件类别	utils, net, mail, text, x11
Priority	软件对应系统的重要性	required, standard, optional, extra 等;
Essential	是否是系统最基本的包	yes/no，若为 yes,则不允许卸载（除非强制性卸载）
Architecture	软件所支持的平台架构	i386, amd64, m68k, sparc, alpha, powerpc 等，也可写 any 表示任意

Source	软件包的源代码名称	-
Depends	软件所依赖的其他软件包和库文件	若依赖多个软件包和库文件，采用逗号隔开
Pre-Depends	软件安装前必须安装、配置依赖性的软件包和库文件	常用于必须的预运行脚本需求
Recommends	推荐安装的其他软件包和库文件	-
Suggests	建议安装的其他软件包和库文件	-

- preinst 文件：在 Deb 包文件解包之前（即软件安装前），将会运行该脚本。可以停止作用于待升级软件包的服务，直到软件包安装或升级完成。
- postinst 文件：负责完成安装包时的配置工作。如新安装或升级的软件重启服务。软件安装完后，执行该 Shell 脚本，一般用来配置软件执行环境，必须以“#!/bin/sh”为首行。
- prerm 文件：该脚本负责停止与软件包相关联的 daemon 服务。它在删除软件包关联文件之前执行。
- postrm 文件：负责修改软件包链接或文件关联，或删除由它创建的文件。软件卸载后，执行该 Shell 脚本，一般作为清理收尾工作，必须以“#!/bin/sh”为首行。

3.2.2 具体安装目录说明

在 package 目录下还有如下目录：

- usr/bin：存放可执行文件，安装完成后会在系统的/usr/bin 下生成可执行文件；
- usr/share/icons：图标文件存放目录，安装后会在系统/usr/share/icons 目录下生成图标文件；
- usr/share/applications：存放桌面 desktop 文件，安装后会在/usr/share/applications 目录下生成.desktop 文件；
- lib/modules/`uname -r`/extra：存放核外 ko 文件，安装后对应 deb 后，系统/lib/modules/`uname -r`/extra 目录下会生成 ko 文件；
- etc/modules-load.d/:存放开机自动加载 ko 模块的 conf 配置文件，安装后对应 deb 后，系统/etc/modules-load.d 目录下会生成对应 conf 文件；

其他更多打包信息请参照《银河麒麟桌面操作系统 V10-DEB 包打包规范》。

4 非 dkms 核外驱动源码打包成 deb 实例

4.1 非 dkms 核外驱动模块编译 ko

4.1.1 源码准备

本实例构建 hello 模块，以下为 hello.c 源码内容：

```
#include <linux/init.h>
#include <linux/module.h>
static int hello_init(void)
{
    printk(KERN_INFO " Hello World enter\n");
    return 0;
}

static void hello_exit(void)
{
    printk(KERN_INFO " Hello World exit\n ");
}
module_init(hello_init);
module_exit(hello_exit);

MODULE_AUTHOR("kylinsoft");
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("A simple Hello World Module");
MODULE_ALIAS("a test module");
```

准备 Makefile 文件

```
KDIR=/lib/modules/`uname -r`/build
obj-m := hello.o
.PHONY: default clean
default:
```

```
	make -C $(KDIR) M=`pwd` modules
clean:
	make -C $(KDIR) M=`pwd` modules clean
```

4.1.2 编译

开始编译，执行 make

```
kylin@kylin-pc:~/hello$ sudo make
make -C /lib/modules/`uname -r`/build M=`pwd` modules
make[1]: 进入目录“/usr/src/linux-headers-5.4.18-110-generic”
  CC [M]  /home/kylin/hello/hello.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/kylin/hello/hello.mod.o
  LD [M]  /home/kylin/hello/hello.ko
make[1]: 离开目录“/usr/src/linux-headers-5.4.18-110-generic”
```

查看编译后的文件信息

```
kylin@kylin-pc:~/hello$ ls
hello.c  hello.ko  hello.mod  hello.mod.c  hello.mod.o  hello.o
Makefile  modules.order  Module.symvers
```

加载 hello.ko 模块并查看打印信息

```
kylin@kylin-pc:~/hello$ sudo insmod hello.ko
kylin@kylin-pc:~/hello$ lsmod | grep hello
hello                  16384  0
kylin@kylin-pc:~/hello$ dmesg
[24599.690084]  Hello World enter
```

4.2 打包成 deb 包

4.2.1 打包准备

创建需要打包的模块目录，此处为 hello（此处的 hello 目录需与 4.1 章节的 hello 目录做区分，可使用其他名称，如 hello-1.1）。

```
kylin@kylin-pc:~$ mkdir hello
```

创建其他相关目录

```
kylin@kylin-pc:~$ mkdir hello/DEBIAN
kylin@kylin-pc:~$ mkdir -p hello/etc/modules-load.d
kylin@kylin-pc:~$ mkdir -p hello/lib/modules/`uname -r`/extra
```

在 hello/DEBIAN 目录下编写 control 文件，内容见下：

```
Package: hello
Version: 1.1
Depends: linux-image
Architecture: amd64
Maintainer: kylin <kylin@kylinos.cn>
Priority: optional
Description: make ko to deb test
```

在 hello/DEBIAN 目录下编写 postinst 文件，此文件主要用于 deb 安装之后自动加载驱动到内核，其中 depmod 命令用于更新/lib/modules/`uname -r`/modules.dep 文件，使 modprobe 命令能够找到驱动位置。modprobe 命令用于加载驱动。内容如下:

```
#!/bin/sh
depmod
modprobe hello
exit 0
```

在 hello/DEBIAN 目录下编写 postrm 文件，此文件主要用于 deb 卸载之后自动卸载驱动并更新/lib/modules/`uname -r`/modules.dep 文件，内容如下:

```
#!/bin/sh
modprobe -r hello
depmod
exit 0
```

在 hello/etc/modules-load.d 目录下编写 hello.conf 文件，添加开机启动之后自动加载的模块名称，内容如下:

```
hello
```

拷贝 hello.ko 至 hello/lib/modules/`uname -r`/extra/

```
kylin@kylin-pc:~$ cp hello.ko hello/lib/modules/`uname -r`/extra
```

hello 目录的文件情况

```
kylin@kylin-pc:~/hello$ tree
.
├── DEBIAN
│   ├── control
│   ├── postinst
│   └── postrm
├── etc
│   └── modules-load.d
│       └── hello.conf
└── lib
    └── modules
        └── 5.4.18-110-generic
            └── extra
                └── hello.ko
7 directories, 5 files
```

4.2.2 打包

将已编好的 ko 打包成 deb

```
kylin@kylin-pc:~$ sudo dpkg -b hello hello_1.1_amd64.deb
dpkg-deb: 正在 'hello_1.1_amd64.deb' 中构建软件包 'hello'。
```

【注意】编译 hello.ko 和打包成 deb 的环境需保持一致，否则在打包过程可能会出现各种问题。

4.3 安装与卸载 deb 包

4.3.1 安装与验证

安装此 deb 包

```
kylin@kylin-pc:~$ sudo dpkg -i hello_1.1_amd64.deb
(正在读取数据库 ... 系统当前共安装有 390548 个文件和目录。)
准备解压 hello_1.1_amd64.deb ...
正在解压 hello (1.1) 并覆盖 (1.1) ...
正在设置 hello (1.1) ...
正在处理用于 kysec-utils (3.3.6.1-0k6.50) 的触发器 ...
```

查看 hello 驱动是否已经加载

```
kylin@kylin-pc:~$ lsmod | grep hello
hello                  16384  0
```

重启查看驱动是否自动加载

```
kylin@kylin-pc:~$ sudo reboot
kylin@kylin-pc:~$ lsmod | grep hello
hello                  16384  0
```

4.3.2 卸载与验证

卸载 deb 包

```
kylin@kylin-pc:~$sudo dpkg --purge hello
(正在读取数据库 ... 系统当前共安装有 390549 个文件和目录。)
正在卸载 hello (2.10.2-1) ...
正在处理用于 man-db (2.10.2-1) 的触发器 ...
正在处理用于 install-info (6.8-4build1) 的触发器 ...
```

查看 hello 驱动是否卸载

```
kylin@kylin-pc:~$ lsmod | grep hello
```

【注意】如需在系统更新内核时自动编译并加载驱动，请参照 dkms 驱动方式，本文档不涉及自动编译并加载驱动。