银河麒麟高级服务器操作系统 kmod 驱动

软件包制作流程及说明

麒麟软件有限公司

服务器研发中心

2025 年 05 月 19 日

版本说明

版本号	版本说明	作者	日期	变更内容
V1.0	编写文档	秦凡东	2025-05-19	首次创建

1　约定

文档默认阅读者已经掌握了基本的 RPM 软件包打包技术，例如打包流程和操作、RPM SPEC 文件的编写等。在此基础上，进行 kmod 软件包打包的指导说明。阅读者可以根据指导，将一个已经包含 Makefile 在内的、可成功编译的内核模块打包为 kmod 软件包。

2　Kmod 软件包工作原理说明

kmod 软件包通过调用 weak-modules 来实现一个内核版本编译的模块可以在多个内核版本下使用。weak-modules 的原理是通过在目标内核版本 weak-updates/目录内创建软连接，来实现内核模块在无需重新编译的情况下被多个内核版本使用。

例如，kmod 软件包 kmod-hello，提供内核模块：

/lib/modules/4.19.90-89.21.v2401.ky10.x86_64/extra/hello/hello.ko

通过 weak-modules --add-modules 处理后，根据不同的目标内核版本，会有若干个 weak-updates/目录的软链接被创建并指向真实的内核模块地址。以下为一个示例环境：

/lib/modules/4.19.90-89.11.v2401.ky10.x86_64/weak-updates/hello/hello.ko: symbolic link to /lib/modules/4.19.90-89.21.v2401.ky10.x86_64/extra/hello/hello.ko

/lib/modules/4.19.90-89.20.v2401.ky10.x86_64/weak-updates/hello/hello.ko: symbolic link to /lib/modules/4.19.90-89.21.v2401.ky10.x86_64/extra/hello/hello.ko

对应地，weak-modules --remove-modules 则用来删除这些软链接。

3 RPM SPEC 文件编写指导

3.1 事务处理

因为 kmod 软件包通过调用 weak-modules 工作,所以软件包的事务处理至关重要。将内核模块打包为 kmod 软件包，需要在事务处理中做以下动作：

1. 在软件包安装后，调用 weak-modules 在目标内核创建软链接

2. 在软件包卸载后，调用 weak-modules 删除对应的软链接

通常在%post 段（安装后执行）完成动作 1.，以下为 kmod-hello 软件包中的内容，可作为基本示例：

```
modules=( $(find
/lib/modules/%{kmod_kernel_version}.$(uname
-m)/extra/%{kmod_name} -name '*.ko') )
printf '%s\n' "${modules[@]}" | %{_sbindir}/weak-modules
--add-modules
```

因为 kmod-hello 将内核模块安装至内核的 extra/目录，所以首先找出 extra/目录下的内核模块。之后将这些内核模块通过管道传递给 weak-modules -add-modules 命令的标准输入，每行一个。

通常使用%preun 段（卸载前执行）和%postun 段（卸载后执行）完成动作 2.，以下为 kmod-hello 软件包中的内容，可作为基本示例：

```
%preun
mkdir -p $(dirname "%{dup_module_list}")
rpm -ql %{name}-%{version}-%{release}.%{_arch} | grep -P
'[.]ko$' >"%{dup_module_list}"
```

```
%postun
if [[ -r "%{dup_module_list}" ]]
then
  modules=( $(cat "%{dup_module_list}") )
  rm -f "%{dup_module_list}"
  printf '%s\n' "${modules[@]}" | %{_sbindir}/weak-modules
--remove-modules
fi
```

首先在%preun 段中，读取软件包的内核模块列表，存放到临时文件。卸载软件包后，通过临时文件读取软件包的内核模块列表，将这些内核模块通过管道传递给 weak-modules -remove-modules 命令的标准输入，每行一个。

对应地，需要在软件包 SPEC 中定义事务处理过程中使用的宏：

```
%define
kernel_source %{_usrsrc}/kernels/%{kmod_kernel_version}.%{_arc
h}
%define
dup_state_dir %{_localstatedir}/lib/rpm-state/kmod-dups
%define
dup_module_list %{dup_state_dir}/rpm-kmod-%{kmod_name}-mod
ules
```

3.2 软件包命名

软件包应以”kmod-名称”命名，如 hello 内核模块，对应的 kmod 软件包名应为”kmod-hello”。在 SPEC 文件中可以通过以下宏定义指定软件包名称：

```
Name: kmod-hello
```

SPEC 文件建议保持和 Name 字段同名，如 kmod-hello.spec。

3.3 重新实现 post install 宏

在 kmod 文件开始处定义以下宏,以规避一个 brp-strip 会删除内核模块签名的问题，每个 kmod 的 RPM SPEC 都需要有以下的宏定义：

```
%define __spec_install_post /usr/lib/rpm/check-buildroot \
                                    /usr/lib/rpm/kylin/brp-ldconfig \
                                    /usr/lib/rpm/brp-compress \

/usr/lib/rpm/brp-strip-comment-note /usr/bin/strip /usr/bin/objdump \
                                    /usr/lib/rpm/brp-strip-static-archive
/usr/bin/strip \

/usr/lib/rpm/brp-python-bytecompile "" 1 \
                                    /usr/lib/rpm/brp-python-hardlink
```

3.4 不生成 debuginfo

在 RPM SPEC 开头定义以下宏以避免生成 debuginfo 软件包：

```
%define debug_package %{nil}
```

3.5 定义编译所需的内核版本

使用以下宏定义编译所需的内核版本：

```
%{!?kmod_kernel_version: %define kmod_kernel_version
4.19.90-89.21.v2401.ky10}
```

3.6 定义编译依赖和安装依赖

以下是必需的编译依赖和安装依赖，注意这里指定了只能在大于或等于 3.5

节定义的内核版本系统的环境中安装：

```
BuildRequires: elfutils-libelf-devel
BuildRequires: kernel-devel = %{kmod_kernel_version}
BuildRequires: kernel-rpm-macros
Requires(post): %{_sbindir}/weak-modules
Requires(postun): %{_sbindir}/weak-modules
Requires: kernel >= %{kmod_kernel_version}
```

3.7 定义必需的别名

```
Provides:
kernel-modules >= %{kmod_kernel_version}.%{_arch}
Provides: kmod-%{kmod_name}
= %{?epoch:%{epoch}:}%{version}-%{release}
```

3.7 在%install 中签名软件包

```
# Sign the modules(s)
%if %{?_with_modsign:1}%{!?_with_modsign:0}
# If the module signing keys are not defined, define them here.
%{!?privkey: %define
privkey %{_sysconfdir}/pki/SECURE-BOOT-KEY.priv}
%{!?pubkey: %define
pubkey %{_sysconfdir}/pki/SECURE-BOOT-KEY.der}
for module in $(find %{buildroot} -type f -name \*.ko)
do
  %{_usrsrc}/kernels/%{kmod_kernel_version}.%{_arch}/script
s/sign-file sha256 %{privkey} %{pubkey} $module;
done
  %endif
```

4 RPM SPEC 文件示例

以下为软件包和软件包中的 SPEC 文件示例。软件包如下:

kmod-
hello-1.0-1.ky10.src.rp···

SPEC 文件如下：

```
# Fix for the SB-signing issue caused by a bug in
/usr/lib/rpm/brp-strip
# https://bugzilla.redhat.com/show_bug.cgi?id=1967291
%define __spec_install_post \
  /usr/lib/rpm/check-buildroot \
  /usr/lib/rpm/kylin/brp-ldconfig \
  /usr/lib/rpm/brp-compress \
  /usr/lib/rpm/brp-strip-comment-note /usr/bin/strip
/usr/bin/objdump \
  /usr/lib/rpm/brp-strip-static-archive /usr/bin/strip \
  /usr/lib/rpm/brp-python-bytecompile "" 1 \
  /usr/lib/rpm/brp-python-hardlink \

# If kmod_kernel_version isn't defined on the rpmbuild line,
define it here.
%{!?kmod_kernel_version: %define kmod_kernel_version
4.19.90-89.21.v2401.ky10}

%define kmod_name hello
%define debug_package %{nil}
%define
```

```
kernel_source %{_usrsrc}/kernels/%{kmod_kernel_version}.%{_arch}
    %define dup_state_dir %{_localstatedir}/lib/rpm-state/kmod-dups
    %define dup_module_list %{dup_state_dir}/rpm-kmod-%{kmod_name}-modules


    Name: kmod-%{kmod_name}
    Version: 1.0
    Release: 1
    Summary: %{kmod_name} kernel module(s)
    Group: System Environment/Kernel
    License: GPLv2
    URL: http://www.example.com/

    Source0: %{kmod_name}-%{version}.tar.xz

    ExclusiveArch: x86_64 aarch64 loongarch64

    BuildRequires: elfutils-libelf-devel
    BuildRequires: kernel-devel = %{kmod_kernel_version}
    BuildRequires: kernel-rpm-macros

    Provides: kernel-modules >= %{kmod_kernel_version}.%{_arch}
    Provides: kmod-%{kmod_name} = %{?epoch:%{epoch}:}%{version}-%{release}
```

```
Requires(post): %{_sbindir}/weak-modules
Requires(postun): %{_sbindir}/weak-modules
Requires: kernel >= %{kmod_kernel_version}

%description
%{kmod_name} kernel module(s).

%prep
%autosetup -n %{kmod_name}-%{version}

%build
%{__make} %{?_smp_mflags} all KDIR=%{kernel_source}

%install
%{__install}
-d %{buildroot}/lib/modules/%{kmod_kernel_version}.%{_arch}/ext
ra/%{kmod_name}/
%{__install}
*.ko %{buildroot}/lib/modules/%{kmod_kernel_version}.%{_arch}/e
xtra/%{kmod_name}/

# strip modules(s)
find %{buildroot} -type f -name '*.ko' -exec %{__strip}
--strip-debug \{\} \;

# Sign the modules(s)
```

```
	%if %{?_with_modsign:1}%{!?_with_modsign:0}
	# If the module signing keys are not defined, define them here.
	%{!?privkey: %define
privkey %{_sysconfdir}/pki/SECURE-BOOT-KEY.priv}
	%{!?pubkey: %define
pubkey %{_sysconfdir}/pki/SECURE-BOOT-KEY.der}
	for module in $(find %{buildroot} -type f -name \*.ko)
	do
	  %{_usrsrc}/kernels/%{kmod_kernel_version}.%{_arch}/script
s/sign-file sha256 %{privkey} %{pubkey} $module;
	done
	%endif


	%post
	modules=( $(find
/lib/modules/%{kmod_kernel_version}.$(uname
-m)/extra/%{kmod_name} -name '*.ko') )
	printf '%s\n' "${modules[@]}" | %{_sbindir}/weak-modules
--add-modules


	%preun
	mkdir -p $(dirname "%{dup_module_list}")
	rpm -ql %{name}-%{version}-%{release}.%{_arch} | grep -P
'[.]ko$' >"%{dup_module_list}"


	%postun
	if [[ -r "%{dup_module_list}" ]]
```

```
    then
        modules=( $(cat "%{dup_module_list}") )
        rm -f "%{dup_module_list}"
        printf '%s\n' "${modules[@]}" | %{_sbindir}/weak-modules
--remove-modules
    fi


    %files
    %defattr(644,root,root,755)
    /lib/modules/%{kmod_kernel_version}.%{_arch}/


    %changelog
    * Thu May 15 19:40:09 CST 2025 Qin Fandong
<qinfandong@kylinos.cn> - 1.0
    - Say hello.
```

以上演示了 kmod 软件包的原理和制作流程，实际过程中需要针对具体的编译构建来定制打包过程。