

Opkg包管理器

像大多数Linux发行版（或Android或iOS这样的移动设备操作系统）一样，通过从软件包存储库（本地或Internet）下载和安装预制软件包，系统的功能可以显着升级。

该 `opkg` 实用程序是用于此作业的轻量级包管理器。`Opkg`是 `ipkg` NSLU2的Optware (<http://www.nslu2-linux.org/wiki/Optware/>)中使用的包管理器的一个分支，该软件旨在向嵌入式设备的固件添加软件。

`Opkg`是根文件系统的完整包管理器，包括内核模块和驱动程序，而`ipkg`只是将软件添加到单独的目录（例如 `/opt`）。

`Opkg`有时被称为`Entware`，因为它也在Entware存储库中 (<http://entware.wl500g.info>)用于嵌入式设备（OpenWRT社区包存储库的分支）。

软件包管理器 `opkg` 尝试使用存储库中的软件包来解析依赖关系 - 如果这种情况失败，它将报告错误并中止该软件包的安装。

缺少与第三方软件包的依赖关系大概可从软件包的源头获得。

要忽略依赖性错误，请传递该 `--force-depends` 标志。

❗如果您使用的是快照/中继/出血边缘版本，则如果存储库中的软件包的内核版本比内核版本更新，则安装软件包可能会失败。

在这种情况下，您将收到错误消息“无法满足...的以下依赖关系”。

对于LEDE固件的使用，我们热烈推荐使用Image Builder制作包含所需软件包的可闪烁映像。

❗在中继/快照时，内核和kmod包将被标记为保持。该 `opkg upgrade` 命令不会尝试更新它们。

调用

`opkg`必须有一个子命令参数：

用法: `opkg [options ...] sub-command [arguments ...]`

其中`sub-command`是以下之一：

您可以使用glob模式 ([https://en.wikipedia.org/wiki/Glob_\(programming\)](https://en.wikipedia.org/wiki/Glob_(programming)))。

包装操作

update

可用软件包的更新列表

只需检索如下所示的文件：例如 (https://downloads.lede-project.org/snapshots/packages/aarch64_armv8-a/base/Packages)，用于安装，并将其存储在RAM分区 (<https://en.wikipedia.org/wiki/tmpfs>)下 `/tmp/opkg-lists`。从LEDE 17.01起，在`opkg`升级后，该文件夹占用大约450KbB的空间。`OPKG`需要此文件夹的内容才能安装或升级软件包或打印有关它们的信息。您可以随时安全地删除此文件夹的内容，以释放一些RAM（重新启动时内容也会丢失），请勿 `opkg update` 在安装新软件包之前再次运行。

upgrade <pkgs>	<p>升级软件包</p> <p>要升级一组软件包，请运行。可以使用命令获取可升级包的列表。 ---- 由于 LEDE固件将基本系统存储在压缩的只读分区中，对基本系统软件包的任何更新都将写入读写分区，因此使用的空间大于覆盖旧版本的空间在压缩的基本系统分区中。建议检查内部闪存中的可用空间以及更新基本系统软件包的空间要求。升级您安装的软件包不应该有这个问题，因为它们已经在读写分区中，所以新的软件包将覆盖旧的分区，尽管在升级之前检查不会受到伤害。 作为一般的经验法则，更新基本软件包时，具有8 MiB或更高总闪存大小和没有用户安装软件包的设备不应存在空间问题，当然，为Extroot设置的设备将不会有任何空间问题。要检查可用空间，请从SSH 写入或在Luci网络界面（系统子菜单 ->软件）中的软件页面中查看内部存储空间剩余的空间。通过写检查要更新包的大小在SSH或检查在软件页面在表中列出的封装尺寸，或者你可以检查包的表在维基这里。虽然opkg中的“size”是压缩归档中程序包的大小，但jffs2或ubifs读写分区将对已安装的文件使用相同的压缩算法，因此在安装时应具有相似的大小。开发快照中的软件包库由构建机器人经常更新为新版本，因此由于与内核或内核相关软件包的依赖关系不足，很有可能您无法升级某些软件包。在这种情况下，建议您使用图像处理器，并使用所需的所有软件包制作新的固件映像，而不是通过opkg进行升级。 opkg upgrade packagename1 packagename2</p> <pre>opkg list-upgradable</pre> <p></p> <pre>df -h / opkg info package-name</pre> <p></p>
install <pkgs FQDN>	<p>安装软件包</p> <p>示例：</p> <div> <pre>opkg安装hiawatha</pre> </div> <div> <pre>opkg安装http://downloads.openwrt.org/snapshots/trunk/ar71xx/packages/hiawatha_7.7-2_ar71xx.ipk</pre> </div> <div> <pre>opkg install /tmp/hiawatha_7.7-2_ar71xx.ipk</pre> </div>
configure <pkgs>	<p>配置未包装的包</p>
remove <pkgs globp>	<p>删除包装</p>
flag <flag> <pkgs>	<p>标记一个或多个包裹。每次调用只允许一个标志。可用标志： 保持•noprune•用户•确定•安装•解包</p>

list [pkg globp]	列出可用包 <div> 软件包名称 - 版本 - 说明 </div> <p>说明可以包含换行符，因此只使用grep是inapt，因为grep是基于行的。</p>
list-installed	列出已安装的包
list-upgradable	列出已安装和升级的软件包
list-changed-conffiles	列出用户修改的配置文件
files <pkg>	列出属于<pkg>的文件。该包必须已安装才能使其工作。例： <div> opkg文件asterisk18 软件包asterisk18（1.8.4.4-1）安装在根目录下，并具有以下文件： /usr/lib/asterisk/modules/res_rtp_multicast.so /usr/lib/asterisk/modules/codec_ulaw.so /etc/asterisk/features.conf /usr/lib/asterisk/modules/format_wav_gsm.so /usr/lib/asterisk/modules/app_macro.so /usr/lib/asterisk/modules/chan_sip.so /usr/lib/asterisk/modules/app_dial.so /usr/lib/asterisk/modules/app_playback.so /usr/lib/asterisk/modules/format_gsm.so /usr/lib/asterisk/modules/func_callerid.so /usr/lib/asterisk/modules/func_timeout.so /etc/asterisk/asterisk.conf /etc/asterisk/modules.conf /usr/lib/asterisk/modules/format_wav.so /etc/asterisk/extensions.conf /etc/init.d/asterisk /etc/asterisk/manager.conf /usr/lib/asterisk/modules/res_rtp_asterisk.so /etc/asterisk/logger.conf /etc/asterisk/rtp.conf /usr/lib/asterisk/modules/codec_gsm.so /etc/asterisk/indications.conf /usr/lib/asterisk/modules/func_strings.so /usr/lib/asterisk/modules/app_echo.so /usr/lib/asterisk/modules/format_pcm.so /etc/asterisk/sip_notify.conf /etc/asterisk/sip.conf 在/ etc /默认/星号 / usr / sbin目录/星号 /usr/lib/asterisk/modules/pbx_config.so /usr/lib/asterisk/modules/func_logic.so </div>
search <file globp>	列表包提供<file>
info [pkg globp]	显示<pkg>的所有信息

包装: horst
版本: 2.0-rc1-2
取决于: libncurses
规定:
状态: 安装用户安装
部分: 网
建筑: ar71xx
维护者: Bruno Randolf <br1@einfach.org>
MD5Sum: 378cea9894ec971c419876e822666a6a
尺寸: 19224
文件名: horst_2.0-rc1-2_ar71xx.ipk
来源: feeds / packages / net / horst
说明: [horst]是802.11无线网络的扫描和分析工具
特别是IBSS (ad-hoc) 模式和网状网络 (OLSR)。

Note1: 该尺寸是gzip压缩tar归档文件的大小。在安装包得到不可靠和解压缩，但是再次JFFS2使用压缩本身。
Note2: 由于JFFS2的压缩是透明的，像这样的命令 `ls` 将始终报告未压缩文件的大小。

status [pkg globp]	显示<pkg>的所有状态
download <pkg>	下载<pkg>到当前目录
compare-versions <v1> <op> <v2>	比较版本 v1 和 v2 使用的运营商 <= , < , > , >= , = , << 或者 >>
print-architecture	列出可安装的包体系结构
whatdepends [-A] [pkgname pat]+	这仅适用于已安装的软件包。所以如果你想知道，一个包和它所有的依赖关系需要多少存储空间，那么你现在就必须把这个信息和 info -option 一起分开。
whatdependsrec [-A] [pkgname pat]+	这仅适用于已安装的软件包。所以如果你想知道，一个包和它所有的依赖关系需要多少存储空间，那么你现在就必须把这个信息和 info -option 一起分开。
whatprovides [-A] [pkgname pat]+	
whatconflicts [-A] [pkgname pat]+	
whatreplaces [-A] [pkgname pat]+	

选项

选项	长	描述
-A		查询所有软件包，而不仅仅是安装的软件包
-V[<level>]	--verbosity[=<level>]	将详细级别设置为<level>。可用的详细级别： 0错误只有 1个普通消息（默认） 2个信息性消息 3调试 4调试级别2
-f<conf_file>	--conf<conf_file>	使用<conf_file>作为opkg配置文件。默认是 /etc/opkg.conf
	--cache<directory>	使用包缓存
-d<dest_name>	--dest<dest_name>	使用<dest_name>作为软件包安装，删除和升级的根目录。 <dest_name>应该是从配置文件定义的目标名称（但也可以是一个目录名称）。
-o <dir>	--offline-root<dir>	使用<dir>作为脱机安装软件包的根目录。
	--add-arch<arch>:<prio>	以优先级注册架构
	--add-dest<name>:<path>	使用给定路径注册目的地
强制选项		
	--force-depends	尽管依赖关系不能安装/删除
	--force-maintainer	覆盖预先存在的配置文件
	--force-reinstall	重新安装软件包
	--force-overwrite	从其他软件包覆盖文件
	--force-downgrade	允许opkg降级软件包
	--force-space	禁用可用空间检查
	--force-checksum	忽略校验和不匹配
	--force-postinstall	即使在离线模式下也运行postinstall脚本

	<code>--noaction</code>	无动作 - 仅测试
	<code>--download-only</code>	无动作 - 仅限下载
	<code>--nodeps</code>	不遵循依赖关系
	<code>--force-removal-of-dependent-packages</code>	删除软件包和所有依赖项
	<code>--autoremove</code>	删除自动安装以满足依赖关系的软件包
<code>-t</code>	<code>--tmp-dir</code>	指定tmp-dir。

例子

要安装软件包，请运行以下命令。可用包的列表在重新启动时丢失，因此请确保在尝试安装包之前更新列表

```
opkg更新
opkg install <package>
```

寻找

- `opkg list` 将只显示 `Package name – Version – Description`
- `opkg info` 将显示所有可用信息。

您可以直接使用glob模式 ([https://en.wikipedia.org/wiki/Glob_\(programming\)](https://en.wikipedia.org/wiki/Glob_(programming)))，还可以编写一个小的 脚本 ([https://en.wikipedia.org/wiki/Shell script](https://en.wikipedia.org/wiki/Shell_script))来使用正则表达式 ([https://en.wikipedia.org/wiki/Regular expression](https://en.wikipedia.org/wiki/Regular_expression))，否则进一步处理信息。使用pipe（|）和 `grep` / `awk` 或 `sed` 过滤该输出：

- `opkg list | grep pattern`
- `opkg list | awk '/pattern/ {print $0}'`
- `opkg info kmod-ipt-* | awk '/length/ {print $0}'`
- `opkg list-installed | awk '{print $1}' | sed ':M;N;$!bM;s#\n# #g'`
- `var="packagename1 packagename2 packagename2"; for i in $var; do opkg info $i; done;`
- `opkg depends dropbear` 也不行。

组态

主要的配置文件是 `/etc/opkg.conf` 。它可能看起来像这样：

```
破坏根/  
dest ram / tmp  
lists_dir ext / var / opkg-lists  
选项overlay_root / overlay
```

您可以看到，它设置默认文件夹。

- 默认的根目录（默认/），
- 默认的ram磁盘（默认/ tmp） - 用于存储软件包列表的默认文件夹（默认/ var / opkg-lists，仍然是ram磁盘） - 什么是overlay目录（默认/覆盖）

这些选项中的大多数必须保留为默认值，否则没有任何真正的理由被更改。

您可能要更改 `lists_dir ext /var/opkg-lists` 到 `lists_dir ext /path/on/disk`，如果您的设备有32 MIB或更少的RAM和你在一个外部驱动器扩展固件的存储空间，让你可以使用opkg而不会造成内存不足的错误。

调整存储库

饲料设置 `/etc/opkg/distfeeds.conf`

```
src / gz reboot_core http://downloads.lede-project.org/snapshots/targets/ramips/mt7620/packages  
src / gz reboot_base http://downloads.lede-project.org/snapshots/packages/mipsel_24kc/base  
src / gz reboot_telephony http://downloads.lede-project.org/snapshots/packages/mipsel_24kc/telephony  
src / gz reboot_packages http://downloads.lede-project.org/snapshots/packages/mipsel_24kc/packages  
src / gz reboot_routing http://downloads.lede-project.org/snapshots/packages/mipsel_24kc/routing  
src / gz reboot_luci http://downloads.lede-project.org/snapshots/packages/mipsel_24kc/luci
```

还有另外一个用于自定义供稿的文件 `/etc/opkg/customfeeds.conf`

```
# 在这里添加您的自定义包Feed  
#  
# src / gz example_feed_name http://www.example.com/path/to/files
```

本地存储库

您可以配置opkg以在本地获取包：

```
src / gz本地文件: /// path / to / packagesDirectory
```

LEDE使用多个存储库，每个存储库都需要一个唯一的标识符。使用他们的原始名称是合乎逻辑的，例如：

```
...
src / gz base file: /// path / to / packages / directory / packages / base
src / gz luci file: /// path / to / packages / directory / packages / luci
src / gz包文件: /// path / to / packages / directory / packages / packages
src / gz oldpackages file: /// path / to / packages / directory / packages / oldpa
ckages
...等等
```

改变架构

LEDE正在使用封装架构，许多设备从同一个池中提取软件包。
以下段落需要校对，并且可能需要修改以适应当前的LEDE状态。

默认情况下，**opkg**仅允许具有架构 **a11** (=独立于体系结构) 的包和已安装目标的体系结构。
为了下载和安装外部目标架构的软件包，可以 `/etc/opkg.conf` 使用 **arch** 选项来覆盖允许的架构列表：

```
全部100
arch brcm4716 200
arch brcm47xx 300
```

该示例将允许在（特定的SoC）目标 **brcm47xx** 上安装软件包（编译为在 **brcm47xx SoC** /设备系列上运行 **brcm4716**）。

该数字指定优先级索引，该索引用于 **opkg** 确定哪些包在多个体系结构中可用时更愿意使用。

代理支持

要 **opkg** 通过代理使用，请将以下内容添加到 `/etc/opkg.conf`：

```
选项http_proxy http://proxy.example.org:8080/
选项ftp_proxy ftp://proxy.example.org:2121/
```

使用以下选项对代理服务器进行身份验证：

```
选项proxy_username xxxx
选项proxy_password xxxx
```

由于板载**wget**的限制，验证可能会失败（应该不是这样，因为当前**wget**由**uclient-fetch**提供）。在这种情况下，尝试传递用户名和密码作为url的一部分。

```
选项http_proxy http: // username: password@proxy.example.org: 8080 /
选项ftp_proxy http: // username: password@proxy.example.org: 2121 /
```

故障排除

空间不足

如果`opkg`空间不足，它通常无法清理干净地将悬挂的锁定文件放在适当位置，导致 *Could not obtain administrative lock* 错误。可以通过发出 `rm /usr/lib/opkg/lock` 命令来删除锁定文件。

此外，`opkg`可能不会删除它正在安装的文件。

一种方法是获取它正在安装的文件列表，然后删除它们。

用适当的包装替换网址。

```
(cd /; \
wget -qO- http://download.link.to.package | \
tar -Oxz ./data.tar.gz | tar -tz | xargs rm)
```

但是，上述一行不会删除与负责的程序包一起安装的依赖项。此外，它留下空的目录。以下脚本旨在修复这些脚本。

```
#!/bin/sh
#takes一个参数/参数：未正确安装的软件包的名称，并应与其依赖关系一起删除
#example: ./opkgremovepartlyinstalledpackage.sh pulseaudio-daemon

#get将与包x一起安装的所有软件包的列表
PACKAGES=`opkg --force-space --noaction install $ 1 | grep"http: " | cut -f 2 -
d" " | sed的/\.$//`
opkg更新

因为我在$ PACKAGES
做
    LIST=`wget -qO- $ 1 | tar -Oxz ./data.tar.gz | tar -tz | 排序-r | sed的/
^./\// overlay \ / upper /`
    对于$ LIST中的f
        做
            如果[-f $ f]
                然后
                    echo“删除文件$ f”
                    rm -f $ f
            科幻
            如果[-d $ f]
                然后
                    echo“尝试删除目录$ f（仅适用于空目录）”
                    rmdir $ f
            科幻
        科幻
    DONE
DONE
回声“您可能需要重新启动以便可用空间变得可见”
```

将其保存为 `opkgremovepartlyinstalledpackage.sh` 某个位置，将其设置为可执行文件，`chmod +x ./opkgremovepartlyinstalledpackage.sh` 然后执行 `./opkgremovepartlyinstalledpackage.sh <package-name> .`

本地存储库

 将其转换为LEDE并解释其实际在做什么。

实例：

```
R = 44685
搜索= “http://downloads.openwrt.org/snapshots/trunk/ar71xx/generic”
替换= “文件： /// MNT / SD卡/共享/用户/网络/ R $ R”
sed -i -e“! $ search! $ replace! ” /etc/opkg.conf
```

分享第二台路由器：

```
ln -s / mnt / sdcard / shared / users / www / www / pendrive
```

在第二个路由器中：

```
R = 44685
搜索= “downloads.openwrt.org/snapshots/trunk/ar71xx/generic”
取代= “192.168.1.1/pendrive/r$r”
sed -i -e“! $ search! $ replace! ” /etc/opkg.conf
```

非标准安装目的地

由于它的历史记录（**ipkg**），**opkg**可以指定一个不同于根目录的目的地来进行软件包安装，但由于大多数软件包本身不支持这个目的，所以它比实际上更有好奇心。

有更多空间来安装软件包的推荐方法是**Extroot**

警告：在大多数情况下，此解决方案将无法开箱即用，**LEDE**软件包旨在安装在根文件系统中，并可能需要额外的符号链接或修改才能在所改变的路径下工作！

默认的**opkg.conf**实际上包含三个目的地：


```
破坏根/
dest ram / tmp
dest mnt / mnt
```

目标行的格式只是关键字**dest**，后面跟着这个目的地的名称（这可以是任何东西），后跟文件系统位置。可以在**opkg**命令行上指定任何如此配置的目标，如下所示：

```
opkg安装somepackage -d destination_name
```

该**DEST**参数必须是指定义的目的地之一 **/etc/opkg.conf**，例如，**-d ram** 以安装软件包 **/tmp/**。

如果要在任何其他目的地比root安装内核模块，您可能需要首先阅读：[https \(https://dev.openwrt.org/ticket/10739\) : //dev.openwrt.org/ticket/10739 \(https://dev.openwrt.org/ticket/10739\)](https://dev.openwrt.org/ticket/10739)

 最后一次修改：2017/03/26 10:23 通过bobafetthotmail

除非另有说明，本维基的内容将根据以下许可证获得许可：CC Attribution-Share Alike 4.0 International
(<http://creativecommons.org/licenses/by-sa/4.0/>)