

# 使用图像生成器（图像生成器）

这是一个预编译的环境，适合创建自定义图像，而无需从源代码编译它们。它下载预编译的软件包，并将它们集成到一个可闪存的映像中。这样做是有用的：

- 您希望以更小的闪光灯大小适合更多的包
- 你想遵循开发快照
- 您的设备具有**32MB**或更少的**RAM**，并且**opkg**无法正常工作
- 您想要大量闪存几十种设备，您需要一个特定的固件设置

## 先决条件

⚠️ 图像生成器仅在**64位Linux**中运行。然而，即使从**32位**窗口，您也可以**在VM（即虚拟机）中运行64位Linux**。

⚠️ 图像生成器具有构建系统构建系统 - 安装的一些相同的先决条件。最常见的发行版中的示例依赖关系：

于Debian / Ubuntu

```
apt-get install subversion build-essential libncurses5-dev zlib1g-dev gawk git ccache  
gettext libssl-dev xsltproc wget unzip python
```

CentOS的/ RHEL

```
yum install subversion git gawk gettext ncurses-devel zlib-devel openssl-devel libxslt  
wget  
yum group安装“开发工具”
```

## 获取图像生成器

您可以下载包含图像生成器的存档，它通常位于同一个下载页面，您可以在其中找到设备的固件映像。

例如，这是您可以下载**ar71xx /通用设备的所有固件映像**的页面

<https://downloads.lede-project.org/snapshots/targets/ar71xx/generic/> (<https://downloads.lede-project.org/snapshots/targets/ar71xx/generic/>)。

您将在其中找到一个带有图像构建器的**lede-imagebuilder-ar71xx-generic.Linux-x86\_64.tar.xz**存档。

此外，它始终由构建系统创建，因为需要创建映像文件。如果启用了“构建LEDE图像生成器”选项，则图像生成器将在您找到固件映像（`source/bin/targets/xxx`）的同一文件夹中生成，您可以使用该文件从编译期间获取的包中创建更多图像。

## 配置软件包存储库

将图像生成器从LEDE网页下载已经配置为从官方源下载任何非默认套餐。

软件包源在提取的目录中的 `repositories.conf` 文件中配置。源以 `opkg` 本机配置格式指定。这可以是官方软件包存储库或自定义生成的存储库。

的内容的一个例子 `repositories.conf` 从莱德-**imagebuilder-ar71xx-generic.Linux-x86\_64.tar.xz**:

```
##将您的自定义存储库放在这里，它们必须与架构和版本相匹配。
#src / gz reboot http://downloads.lede-project.org/snapshots
#src自定义文件: /// usr / src / lede / bin / ramips / packages

##远程包仓库
src / gz reboot_core http://downloads.lede-project.org/snapshots/targets/ramips/mt
7620/packages
src / gz reboot_base http://downloads.lede-project.org/snapshots/packages/mipsel_2
4kc/base
src / gz reboot_telephony http://downloads.lede-project.org/snapshots/packages/mip
sel_24kc/telephony
src / gz reboot_packages http://downloads.lede-project.org/snapshots/packages/mips
el_24kc/packages
src / gz reboot_routing http://downloads.lede-project.org/snapshots/packages/mipse
l_24kc/routing
src / gz reboot_luci http://downloads.lede-project.org/snapshots/packages/mipsel_2
4kc/luci

##这是本地软件包库，不要删除！
src imagebuilder文件: 包
```

将 `repositories.conf` 在 `imagebuilder` 你从源代码编译会缺乏“远程包库”链接。

如果要添加自定义本地存储库，请复制

```
src自定义文件: /// usr / src / lede / bin / ramips / packages
```

行并修改它以指向您的包和包列表的本地文件夹。

如果您有在线自定义存储库，请复制并修改

```
src / gz reboot http://downloads.lede-project.org/snapshots
```

行代替。

# 用法

**make image** 命令将为默认（必需）包的默认设备创建默认映像。在大多数情况下，这不是你想要的。

要更改这种不太有用的默认行为，您可以使用传递为参数的三个变量：

- *PROFILE* - 指定要构建的目标图像
- *PACKAGES* - 嵌入到图像中的包的列表
- *FILES* - 包含自定义文件的目录

示例语法：

```
$ make image PROFILE = XXX PACKAGES =“pkg1 pkg2 pkg3 -pkg4 -pkg5 -pkg6”FILES = files /
```

请参阅以下部分进行更深入的解释。**make** 命令完成后，生成的图像将被存储在 `bin / device-architecture` 目录中，就像编译它们一样。

这里输出的帮助：

可用命令：

帮助：这个帮助文本

**info**：显示可用目标配置文件的列表

**clean**：删除图像和临时构建文件

图像：构建图像（有关详细信息，请参阅下文）。

建筑图像：

默认情况下，“**make image**”将创建一个默认图像目标配置文件和软件包集。可以使用以下参数

改变：

```
make image PROFILE =“<profilename>”# 覆盖默认目标配置文件
```

```
make image PACKAGES =“<pkg1> [<pkg2> [<pkg3> ...]”# include extra packages
```

```
make image FILES =“<path>”# 包含来自<path>的额外文件
```

```
使图像BIN_DIR =“<path>”# 替代输出目录的图像
```

```
使图像EXTRA_IMAGE_NAME =“<string>”# 将此添加到输出图像文件名（消毒）
```

## PROFILE 变量

句法：

```
$ make image PROFILE = NAME_OF_PROFILE
```

### 预定义的配置文件

运行 `make info` 以获取定义的配置文件的列表。示例输出 `make info` 如下所示。

**ar71xx**通用配置文件

可用的配置文件：

默认：

默认配置文件

套件：kmod-usb-core kmod-usb2 kmod-usb-ohci kmod-usb-ledtrig-usbport

AI-BR100：

Aigale Ai-BR100

套餐：kmod-usb2 kmod-usb-ohci

RP-N53：

华硕RP-N53

包：

RT-n14u：

华硕RT-N14u

包：

WHR-1166d：

布法罗WHR-1166D

包：

WHR-300hp2：

布法罗WHR-300HP2

包：

- 还有很多更多 -

找到适当的配置文件后，将其传递给 `make image` 命令：

例如，如果要为Asus RT-N14u（从上面）生成默认映像。

```
$ make image PROFILE = rt-n14u
```

## 包变量

该 `PACKAGES` 变量指定使用Image Generator构建映像时要包含和/或排除的包的列表。

句法：

```
$ make image PACKAGES =“pkg1 pkg2 pkg3 -pkg4 -pkg5 -pkg6”
```

上述示例将包括pkg1, pkg2, pkg3, 并排除pkg4, pkg5, pkg6, 在每个排除的包之前注意“-”。

您不需要在此列表中列出所需的所有包的所有依赖关系，Image Generator将 `opkg` 自动解析包依赖关系并安装其他所需的包。

提示：可以使用以下命令获取设备上当前安装的软件包列表：

```
echo $(opkg list_installed | awk '{print $ 1}')
```

## 文件变量

该 `FILES` 变量允许将自定义配置文件包含在使用Image Generator构建的映像中。如果您需要在闪烁之前更改默认网络配置，或者正在准备用于大量闪存许多设备的映像，这将非常有用。

句法：

```
$ make image FILES = files /
```

</WRAP>

注：该 `files/` 文件夹必须在您发出`make`命令相同的文件夹。

## 例子

以下示例显示：

1. 创建配置文件的目录
2. 使用 `scp` 将 `uci` 配置文件从WL500GP路由器传输到 `files/etc/config` 目录
3. 使用自定义包和 `uci` 配置文件为WL500GP生成图像

```
mkdir -p文件/ etc / config  
scp root@192.168.1.1: / etc / config / network files / etc / config /  
scp root@192.168.1.1: / etc / config / wireless文件/ etc / config /  
scp root@192.168.1.1: / etc / config / firewall files / etc / config /  
make image PROFILE = wl500gp PACKAGES =“nano openvpn -ppp -ppp-mod-pppoe”FILES = f  
iles /
```

## 清理

要清理临时构建文件和生成的映像，请使用`make clean`命令。

## 从固件中删除无用的文件

- 1.使用完整的文件名创建文件“`files_remove`”：

```
/lib/modules/3.10.49/ts_bm.ko  
/lib/modules/3.10.49/nf_nat_ftp.ko  
/lib/modules/3.10.49/nf_nat_irc.ko  
/lib/modules/3.10.49/nf_nat_tftp.ko
```

- 2.修补Makefile

```

ifneq ( $ ( USER_FILES ) , )
    $ ( MAKE ) copy_files
    万一
+
+ ifneq ( $ ( FILES_REMOVE ) , )
+     @ echo
+     @ echo删除无用的文件
+
+     同时 读取文件名; 做
+         rm -rfv " $ ( TARGET_DIR ) $ $ filename " ; \
+     done < $ ( FILES_REMOVE ) ;
+ ENDIF
+
    $ ( MAKE ) package_postinst
    $ ( MAKE ) build_image

```

### 3.重建固件

```

#make image \
    PROFILE = tlwr841 \
    PACKAGES ="igmpproxy ip iptraf kmod-ipt-nathelper-extra openvpn-polarssl tcpdump-mini -firewall -ip6tables -kmod-ip6tables -kmod-ipv6 -odhcp6c -ppp -ppp-mod-pppoe"\
    FILES_REMOVE = "files_remove"

```

## 使用内部的所有包构建图像生成器

可以构建图像生成器并集成所有包，以便能够生成图像而无需下载包：

在图形配置中，选择“构建**LEDE**图像生成器”以构建图像生成器（no duh!），然后选择全局构建设置 → 默认选择所有包，保存并退出。然后构建映像，`IGNORE_ERRORS=1` 包括可能是未编译的未包含的程序包。

```
使IGNORE_ERRORS = 1
```

注意：不要在路径中调用 `make defconfig` 或保留旧 `.config` 文件，因为 `Select all packages by default` 只将包选择设置为尚未配置的包 [m]（`make defconfig` 将设置大多数包 [n]，即不建立）。

## 添加/修改配置文件

图像生成与配置文件名称相关联。如果您添加一个新的配置文件，而不必为图像生成**Makefile**添加适当的宏，则在使用自定义配置文件时，不会生成合适的固件文件。🚨确保删除 `/tmp` 目录以从配置文件中获取修改后的包选择。


`brcm47xx-for-Linux-i686` 的预编译包的配置文件的位置是 `target / linux / brcm47xx / profiles /`

值得注意的是，添加新配置文件需要完成的所有操作是将新文件添加到配置文件目录中。

以下是个人资料/ *100-Broadcom-b43.mk*个人资料档案：

```
定义Profile / Broadcom-b43
NAME: = Broadcom BCM43xx WiFi (默认)
包装: = kmod-b43 kmod-b43legacy
endif

定义Profile / Broadcom-b43 / Description
      软件包与使用Broadcom BCM43xx卡的硬件兼容
endif
$(eval $(call Profile, Broadcom-b43))
```

 最后修改：2017/05/08 12:24 由zigg

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/>)