Slackware ARM project web site | Forum | Slackware ARM development documentation | Slackware ARM installation guides

Building Slackware AArch64 Linux Kernel packages from the Raspberry Pi Kernel fork

Document	Describe how to self-build Slackware Kernel packages from the Raspberry Pi Kernel fork
Version	1.00, June 2023
Author	Stuart Winter <mozes@slackware>
See Also	* Customising the Slackware Kernel packages * Transitioning to RPi Kernel fork packages

The Slackware ARM/AArch64 build system

The Slackware ARM build system was built to abstract the common functionality of the build scripts into a library and configuration file, enabling global changes to be made in a single location.

The Slackware ARM source tree works as an 'overlay' to the Slackware x86/64 upstream source tree. This way all assets (sources, patches, etc.) are *referenced* from the Slackware ARM build system, but are never duplicated. The Slackware ARM build scripts (called `<packagename>.SlackBuild`) are however customised to operate within the Slackware ARM build system, but produce identical packages to the x86 upstream versions. This approach enables architecture customisations to be easily made, whilst minimising effort required to synchronize source assets.

Assumptions

This document assumes that you're building on a full Slackware system, because the Slackware Linux Kernel package build process requires a number of development tools to be available.

This document provides some basic instructions using the `slackpkg` tool to ensure that your system has the latest versions of these packages available.

You'll find how to setup slackpkg here

If you don't have `slackpkg` set up or would prefer to do this manually, you may use the `upgradepkg –install-new` tool instead.

This document covers working on Slackware `-current` (the development branch), but you can use the same steps for a stable release by switching the name of the tree - e.g. `-current` → `-15.0`.

Preparing the environment

Download the Slackware AArch64 source tree

Within your home directory, create a directory to contain the Slackware AArch64 source tree:

```
$ cd # return to the root of your home directory (not the 'root' user):
$ mkdir slackware ; cd slackware
```

Determine where you are in the file system, as you will need this shortly:

```
$ pwd
/home/mozes/slackware
```

```
$ rsync \
   --delete -Pavv \
   ftp.arm.slackware.com::slackwarearm/slackwareaarch64-current .
```

Note the full stop/period on the end of the command - this denotes the present working directory

The full 'slackware' tree is required because the Kernel build script requires some of the package files to be available during build time, as certain tools and libraries are extracted from them to build the OS InitRD (Operating System Initial RAM Disk).

Install the prerequisite packages

Before the Kernel can be built, you need to ensure that you have certain packages installed. This list isn't fully inclusive because you should build the Kernel on a full system, but particularly for Slackware -current, you need to ensure that you have the latest version of the Kernel packages installed so that it's aligned with the Slackware AArch64 source tree.

If the packages aren't available for installation, use the `upgrade` option (`slackpkg upgrade <pkgname>`) to slackpkg instead to ensure that they're up to date.

Update slackpkg's cache

As **root**:

```
# slackpkg update
```

Install the Slackware ARM development kit

The Slackware ARM build system uses a package named `slackkit` that contains the library functions to build the packages. This package is installed by default, but if you don't have it :

As **root**, install/upgrade the package:

```
# slackpkg install slackkit
```

Install other dependencies

This list isn't exhaustive but should cover it (if more dependencies are required, please drop mozes@slackware.com a mail with them and I'll update this doc!)

```
# slackpkg install eudev mdadm lvm2 device-tree-compiler rsync
```

Configure the Slackware ARM build system

In this example the directory into which the Slackware AArch64 source tree was downloaded is `/home/mozes/slackware` - change this to what you noted down earlier

As **root**, you need to set up some symlinks to the location of your Slackware AArch64 source tree.

```
# cd # return to root's home directory
# ln -vfs /home/mozes/slackware/slackwareaarch64-current/slackware tgzstash
# ln -vfs /home/mozes/slackware/slackwareaarch64-current
```

Alternate Kernel source package building tool

There is a script within the Slackware AArch64 Kernel source directory (`slackwareaarch64-<version>/source/k`) named `build_altsrc_kernel.sh` which will help build Slackware packages from alternate Kernel source repositories.

It uses a 'plugin' style system to store settings for specific repositories.

Elevate privileges to `root`:

```
$ su -
```

Enter the directory in which the Slackware AArch64 source is stored. In this document, it's here:

```
# cd ~mozes/slackware/slackwareaarch64-current/source/k # the ~ expands to
the user's home directory (/home in this case)
```

General settings

The default settings should be good for most users, but note that by default it will need about 3G bytes of space in `/tmp`. If you don't have enough space, you should run it with the relevant command line operator `–tmpdir` to change the location.

The script supports the `–help` command line parameter, which explains the available options.

```
# ./build_altsrc_kernel.sh --help
```

Raspberry Pi Kernel fork git repository settings

By default the tool is configured for the Raspberry Pi Kernel fork online git repository, as this is what this author uses to build the packages provided to bridge the gap between the feature and bug fix disparity between the official Linux Kernel and Raspberry Pi's.

The Raspberry Pi settings can be found in `build_altsrc_kernel.conf/rpi`.

For most users you'll want to use the default settings.

Check which version of the Kernel will be built

First, check what version of the Linux Kernel is available at the repository:

```
# ./build_altsrc_kernel.sh -D
```

```
# ./build_altsrc_kernel.sh -D

----------------------------
Temporary directory.........: /tmp/altkernelpkgs/
Kernel source directory.....: /tmp/altkernelpkgs//src
Package store directory.....: /tmp/altkernelpkgs//pkg

GIT repository..............: https://github.com/raspberrypi/linux.git
Kernel branch...............: rpi-6.1.y
Slackware package build tag : 1_rpi
Save Kernel headers package : No

Hardware Model helper script:
/root/ac/source/k/build_altsrc_kernel.conf/default
----------------------------

Kernel version in web repository
https://raw.githubusercontent.com/raspberrypi/linux/rpi-6.1.y: 6.1.31
```

Build the Kernel

Continuing as root, run the build script:

```
# ./build_altsrc_kernel.sh
```

Transitioning to the RPi Kernel fork Slackware packages

After a few hours your Kernel will have been built, with the packages (by default) placed within `/tmp/altkernelpkgs/pkg`.

You should read the this guide before transitioning to them, as there are a few caveats due to the configuration disparity between the default RPi Kernel config and the generic Slackware Kernel configuration.