Slackware ARM - Frequently Asked Questions

Most of the questions are added here arise from questions posed on the Slackware ARM forum.

Hardware

Q. Will Slackware ARM run on <insert device name>?

A. This is the most frequently asked question.

The answer is “Probably”.

Slackware ARM user land (all of the packages) will run on any ARM cpu which is an armv5te or higher. Most CPUs meet these requirements - especially new ARM devices.

It is impossible for Slackware ARM to support each ARM device out of the box. Unlike in the x86 world where one Kernel can be used for all machines, in the ARM world we need one kernel per ARM architecture (hence why Slackware ARM has two kernel packages - one for the 'Kirkwood' System On Chip architecture and another for the 'ARM Versatile' board), so unless your device is currently supported by Slackware ARM, you'll need to compile your own kernel and determine how to adjust the boot loader used on the device (u-boot is the most common choice). Don't confuse the user land (the packages) support with the Kernel. There's no need to “port” Slackware ARM to another ARM architecture (unless you want to recompile some packages for better efficiency when executing on a higher grade CPU than ARMv5) - usually you will just need a new kernel and that kernel's modules, and maybe some additional driver modules for X.org. If you're new to Linux and ARM, this presents a huge learning opportunity for which you will need to set aside a large slice of time, tenacity, curiosity and interest!

One of the easiest ways to check whether your device has any semblance of Linux community support is to check whether Debian or Fedora runs upon it. If so, many of the Debian users tend to write excellent documentation describing how they installed Linux upon their device. Most of this information can be used to assist with shoehorning Slackware ARM on to the device.

Q. Why does Slackware ARM support device X but not Y?

A. This question is often asked in reference to the Raspberry Pi, which by far has the most wide spread mind share. Slackware ARM is developed by one person whose interest in ARM is very specific. ARM devices have to have Real Time Clocks, preferably SATA support and most importantly have support in the mainline Kernel. It's impossible to progress an Operating System if you're tied to a specific version of the Kernel (even if it's for a short period) because it slows down the development cycle. In addition, the devices targeted for “official” support (devices supported directly from the Slackware ARM tree) need to be open and not require propitiatory/licenced firmware and hacking in order to permit boot of an alternative OS. The Slackware ARM community fills in gaps where there is demand.

Installation specifics

Q. I get a CRC error trying to boot the Kernel

A. You may have used ASCII transfer mode if you downloaded via FTP. Re-download using BINARY transfer mode.

Q. I am trying to chroot into the miniroot filesystem from an existing ARM Linux installation, but receive segfaults.

A. This is usually because the kernel you're running is older than 2.6.31 (Slackware ARM's glibc is compiled to require a kernel version 2.6.31 or greater).

You need to upgrade the kernel on your existing ARM Linux installation.

Packages and user land

Q. Slackware ARM has a newer kernel than Slackware x86 - can we expect x86 to catch up?

A. Slackware ARM aims to follow as many kernel configuration options in x86 as possible in order to provide a consistent user experience, but the ARM port uses which ever kernel is the most appropriate at the time with regard to the ARM architecture (usually the latest kernel is preferred, or necessary to provide support for a newly supported architecture).

Compiling software on ARM

Building the Slackware ARM packages

Q. The SlackBuild scripts in Slackware ARM fail to run. How can I rebuild the Slackware ARM packages from the source directory?

A. This README file explains the process.

Q. Are the Slackware ARM packages cross compiled?

A. Not directly. The packages are all built natively on a running Slackware ARM OS, but use distcc to a bunch of x86/x86_64 machines that runs a basic “cross” toolchain that match the versions of the toolchain et al present in slackwarearm-current. The cross toolchain can be found here. It's usually been tested and builds on slackware64-current but is unsupported.

Q. Does Slackware ARM use Scratchbox?

A. Slackware ARM used ScratchBox back in 2002 to originally build the first batch of packages in order to bootstrap the port, and some effort was made in producing scripts to help setup a Slackware-like environment. However, this work was abandoned shortly after the port was able to be self supporting.

3rd party SlackBuild scripts

Q. As a maintainer of SlackBuild scripts, how can I differentiate between Slackware 14.2 (Software Floating Point) and Slackware 15.0 (Hardware Floating Point) and greater?

A. This code snippet checks the target quadlet from gcc and sets the default compiler flags.

Cross compiling software

Q. How can I cross compile software for use on Slackware ARM?

A. If you wish, you can cross compile software using a suitable cross compiler that outputs ARM binaries. There are a number of cross compilers available in binary form, or you can build your own (although that's much more work but you'll learn a lot!). The reason that Slackware ARM packages are built natively is because many packages aren't suitable for cross compilation without making many manual adjustments to Makefiles and configure scripts. Also, if your package is linking against system libraries, it can be tricky to get this working properly in a cross compiler environment. In my experience, it's far easier to get a fast ARM machine (a quad core Orange Pi) and use distcc. The amount of time you spend waiting for the build to finish is far off set by the amount of time hacking build systems to work properly in a “cross” environment.

General

Q. Why was the ARM port renamed from ARMedslack to Slackware ARM?

A. Originally the ARM port was named ARMedslack because it was not an official port, and as such could not hold Slackware name. In 2009, ARMedslack became the official port but continued with the ARMedslack name until 2012 when it was renamed Slackware ARM in time for the release of Slackware 14.00. The delay in changing the name was simply due to the amount of work required in doing so!

From:
https://docs.slackware.com/ - **SlackDocs**

Permanent link:
https://docs.slackware.com/slackwarearm:faq

Last update: **2017/11/15 09:16 (UTC)**