FreeSandal

樹莓派, 樹莓派之學習, 樹莓派之教育

OPENWRT 的世界：樹莓派 3B 【路由器】移星轉斗《五》編譯 三

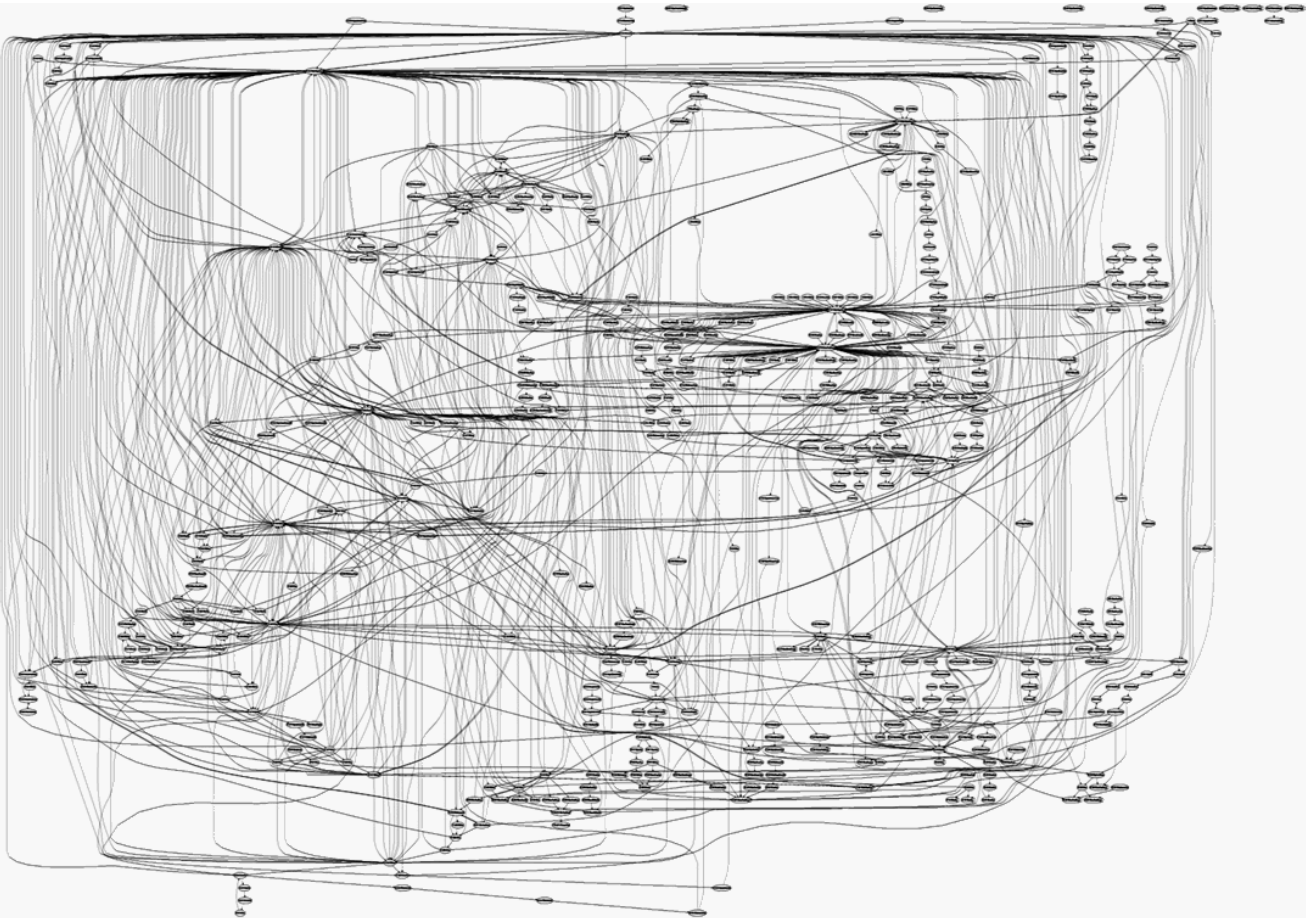
2018-11-30 | 懸鉤子 | 發表迴響

如果曾經有過『經驗』

How to build and install the latest Linux kernel from source

Sreehari

Computer Science Undergrad | BITS Pilani-Goa | Quizzer | Speaker Research-TEDxBITSGoa
Aug 30, 2016



A map of the Linux kernel

I just finished my first assignment for a course on Advanced Operating Systems. And I decided to document my approach for building the Linux kernel from source and implementing my own system call.

There are a number of blogs that already tell you how to go about doing this, but some of them are outdated, and some seem unnecessarily complicated. My goal is to present a straightforward approach for doing this, which should hopefully help you save a lot of time.

Compiling the Linux Kernel from source can seem like a daunting task, even to someone who's pretty comfortable with computers in general. It can also get really irritating if you aren't following the right instructions.

So, here's a guide to help you through the process of building the kernel from source, and it's a guide that works! You will not have to worry about messing up your system or wasting your

time.

Why build the kernel from source?

If you plan to work on the internals of the Linux kernel or change its behavior, you'll need to recompile the kernel on your system.

Here are a few specific cases where you'll need to know how to work with the kernel's source code:

1. You want to write a really cool 'Hello world' program. (Each time you implement your own system call or modify kernel source code, you will need to recompile the kernel to implement the changes)
2. You want to enable experimental features on your kernel that are not enabled by default (or, disable default features that you don't want)
3. You want to debug kernel source code, enable support for a new piece of hardware, or make modifications to its existing configurations
4. You're doing a course on Advanced Operating Systems and have no choice but to do this!

In each of the above situations, learning how to build the kernel from source will come in handy.

.....

可能容易掌握 OpenWrt

Build system – Usage

(!) Do everything as normal user, don't use root user or sudo!
(!) Do not build in a directory that has spaces in its full path
write all command line commands for build system commands in a terminal window opened in the <`buildsystem root`> directory, e.g. `~/source/` if you wrote the **git clone** command in your home folder (default terminal location)

1. Update the sources.
2. Update and install package feeds.

3. Configure the firmware image you want to obtain.
4. Start the build. (automatically compile toolchain, cross-compile sources, package packages, and generate an image ready to be flashed).
5. Proceed with the firmware flashing instructions: Factory install or Sysupgrade –

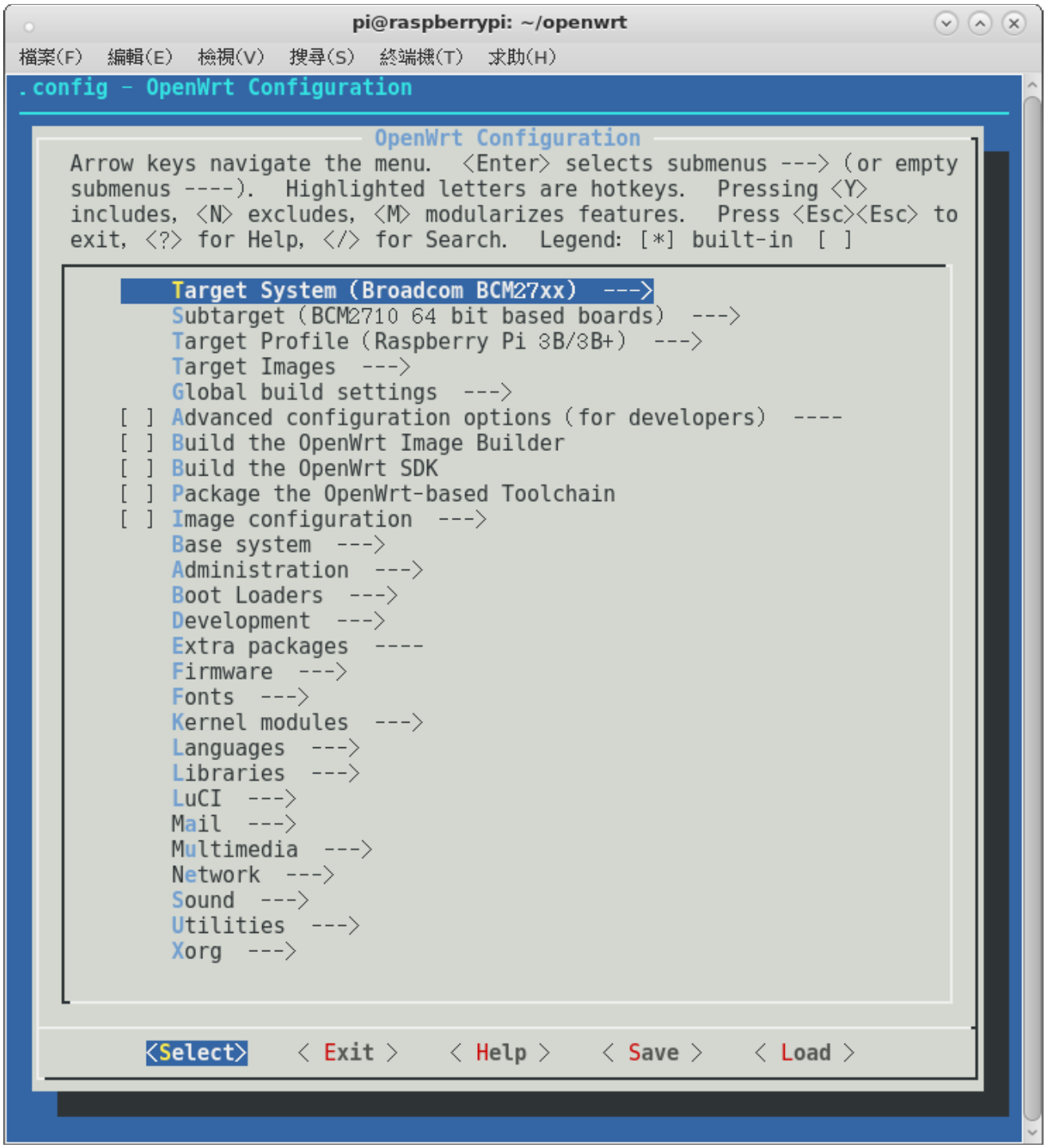
……

Image Configuration

Typical actions:

1. run `make menuconfig` and set target;
2. run `make defconfig` to set default config for build system and device;
3. run `make menuconfig` and modify set of package;
4. run `make download` (download all dependency source files before final make, enables multi-core compilation);
5. run `scripts/diffconfig.sh >mydiffconfig` (save your changes in the text file `mydiffconfig`);

假使祇是添加程式庫已有的軟件



認識

Make menuconfig

The **build system configuration interface** handles the selection of the target platform, packages to be compiled, packages to be included in the firmware file, some kernel options, etc. Start the build system configuration interface by writing the following command:

```
1 make menuconfig
```

This will update the dependencies of your existing configuration automatically, and you can now proceed to build your updated images.

You will see a list of options. This list is really the top of a tree. You can select a list item, and descend into its tree.

To search for the package or feature in the tree, you can type the "/" key, and search for a string. This will give you its locations within the tree.

For most packages and features, you have three options: y, m, n which are represented as follows:

- pressing **y** sets the **<*>** built-in label
  This package will be compiled and included in the firmware image file.
- pressing **m** sets the **<M>** package label
  This package will be compiled, but **not** included in the firmware image file. (E.g. to be installed with opkg after flashing the firmware image file to the device.)
- pressing **n** sets the **< >** excluded label
  The source code will not be processed.

When you save your configuration, the file **<buildroot dir>/.config** will be created according to your configuration.

When you open `menuconfig` you will need to set the build settings in this order (also shown in this order in `menuconfig`'s interface):

1. Target system (general category of similar devices)
2. Subtarget (subcategory of Target system, grouping similar devices)
3. Target profile (each specific device)
4. Package selection
5. Build system settings

6. Kernel modules

Select your device's **Target system** first, then select the right **Subtarget**, then you can find your device in the **Target profile**'s list of supported platforms.

.....

或許足夠矣◎

欲求新創者，最好先閱讀

Development Articles

- UCI defaults
- Adding a new device
- Creating Packages
- Multicast DNS Daemon
- The Procd System Manager
- RPC daemon
- uBus IPC/RPC System
- netifd (Network Interface Daemon) – Technical Reference
- Vagrant-based automatic setup for a LEDE build environment in a VM
- Including custom files/configurations in a firmware image
- Build for multiple targets from the same build tree (build environments aka config swapping)
- "Hello, world!" for LEDE – Developing your first application
- Building image for UEFI based system
- defining firmware partitions for all DTS targets

知其體系也☆