How to compile a kernel module for Raspberry pi?

Asked 5 years, 9 months ago Active 8 months ago Viewed 65k times

24

14

I'm having trouble compiling a kernel module for a raspberry pi. I want to compile a "hello world" kernel module using the raspberry pi itself.

I am using raspbian wheezy 3.6.11+.

I tried following the directions at http://elinux.org/RPi_Kernel_Compilation.

Here is the Makefile I am using:

```
obj-m += hello-1.o

all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

Here is the source code for hello-1.c:

```
/*
 *  hello-1.c - The simplest kernel module.
 */
#include <linux/module.h>   /* Needed by all modules */
#include <linux/kernel.h>   /* Needed for KERN_INFO */

int init_module(void)
{
    printk(KERN_INFO "Hello world 1.\n");

    /*
     * A non 0 return means init_module failed; module can't be loaded.
     */
    return 0;
}

void cleanup_module(void)
{
    printk(KERN_INFO "Goodbye world 1.\n");
}
```

Here's what I get when I try to make the project:

```
root@raspberrypi:/home/pi/hello-module# make
make -C /lib/modules/3.6.11+/build M=/home/pi/hello-module modules
make: *** /lib/modules/3.6.11+/build: No such file or directory.  Stop.
make: *** [all] Error 2
```

```
make -C /lib/modules/3.6.11+/build M=/home/pi/hello-module modules
make[1]: Entering directory `/lib/modules/3.6.11+/build'
make[1]: *** No rule to make target `modules'.  Stop.
make[1]: Leaving directory `/lib/modules/3.6.11+/build'
make: *** [all] Error 2
```

I have GNU Make 3.81 and gcc (Debian 4.6.3-14+rpi1) 4.6.3 installed. I also installed the linux source using

> sudo apt-get install linux-source

Any ideas on what I might do to get this to compile?

linux makefile raspberry-pi kernel-module raspbian

asked Nov 23 '13 at 20:23

user3025582
166 1 2 7

On the Raspberry there is no "build" subdirectory in the modules directory. I could confirm that on my RPI. Hence the question: What is the "correct" way to create it with the required data for module building? – Black Nov 29 '13 at 13:10

Since you've already got some specific answers, here's a short overview on the reasons for the problem. You're using the Raspberry Pi foundation's kernel (the `raspberrypi-firmware` package in Raspbian) which doesn't provide headers you need for making kernel modules. You can either install them using the rpi-source tool. Or switch to Raspian's kernel+headers instead (add a `firmware` entry after `wheezy main contrib ...` in `/etc/apt/sources.lst` , update packages, remove `raspberrypi-firmware` , install `raspberrypi-firmware-nokernel` . – nh2 Jan 11 '15 at 19:35

Great instructions in this question and the answer i compiled into a script – x29a May 21 '16 at 21:52

6 Answers

13

When compiling a module the `-C` parameter should point to the source tree where the kernel was built (don't clean it up!). If you built it on the pi its likely in a directory under your home directory.

The `build` directory under `/lib/modules/<version>` is a Debian-ism, where a cut-down version of the source tree is provided with just enough context to build modules against. The kernels from the Raspberry Pi Foundation kernels don't ship with a `build` directory.

They may be a bit out of date, but raspbian provides a kernel as a Debian-style package, which should include the `build` directory you could use to build kernel modules against.

```
sudo aptitude install linux-image-rpi-rpfv linux-headers-rpi-rpfv
```

13

Here are the steps I used to build the Hello World kernel module on Raspbian.

1. Perform `sudo rpi-update`

   See https://github.com/Hexxeh/rpi-update for details on `rpi-update`. You have to be on the latest firmware and associated kernel to be able to perform the next step.

2. Install and run `rpi-source` to install the source code that built the latest kernel that you are running. This will create the correct entry in `/lib/modules` for the kernel that you are running. Note: you don't need to be root to run this, however the script will perform certain tasks using `sudo` and the root password will be requested during the script execution.

   Instructions to install `rpi-source` can be found at https://github.com/notro/rpi-source/wiki

Once those steps are performed you should be able to `make` the Hello World kernel module.

```
johnma@raspberrypi ~/HelloWorld $ make
make -C /lib/modules/3.12.19+/build M=/home/johnma/HelloWorld modules
make[1]: Entering directory `/home/johnma/linux-
c3db7205bcd8988cf7c185e50c8849542554b1f5'
  CC [M]  /home/johnma/HelloWorld/hello.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/johnma/HelloWorld/hello.mod.o
  LD [M]  /home/johnma/HelloWorld/hello.ko
make[1]: Leaving directory `/home/johnma/linux-
c3db7205bcd8988cf7c185e50c8849542554b1f5'

johnma@raspberrypi ~/HelloWorld $ sudo insmod hello.ko
johnma@raspberrypi ~/HelloWorld $ tail -1 /var/log/syslog
May 15 13:45:39 raspberrypi kernel: [59789.169461] Hello World :)

johnma@raspberrypi ~/HelloWorld $ sudo rmmod hello.ko
johnma@raspberrypi ~/HelloWorld $ tail -1 /var/log/syslog
May 15 13:46:10 raspberrypi kernel: [59819.503503] Goodbye World!
```

answered May 15 '14 at 17:51

HeatfanJohn
5,341 2 27 39

2 Thankyou, this worked beautifully for me. The only hiccup I had was running `rpi-source`. I encountered a Python error saying it couldn't find /proc/config.gz. Running `sudo modprobe configs` fixed it. – Andy J Aug 8 '15 at 9:30

9

You first need kernel headers (and the corresponding kernel binary) to build your module.
Like Greg said, the raspbian distribution provides the packages :

Then, tell raspbian to boot your newly installed kernel (3.10-3-rpi for me).
Append this at end of `/boot/config.txt` and reboot your Pi :

```
# Parameters to boot on raspbian kernel (linux-image-rpi-rpfv package)
kernel=vmlinuz-3.10-3-rpi
initramfs initrd.img-3.10-3-rpi followkernel
```

Then, modify your Makefile to point the freshly installed kernel headers :

```
KERNEL_HEADERS=/lib/modules/$(shell uname -r)/build

obj-m := hello-1.o

all:
    @$(MAKE) -C $(KERNEL_HEADERS) M=$(PWD) modules

clean:
    @$(MAKE) -C $(KERNEL_HEADERS) M=$(PWD) clean
```

edited Apr 6 '14 at 9:46

answered Apr 5 '14 at 15:39

JayDee
91 1 3

`Then, tell raspbian to boot your newly installed kernel (3.10-3-rpi for me).` How can you tell the version? – gromit190 May 3 '17 at 18:46

8

This was a pain. I had to compile and install a kernel mode driver.After long search, i got the headers for pi 2 (3.18.7-v7+) from here.

```
sudo apt-get install dkms build-essential
wget http://www.niksula.hut.fi/~mhiienka/Rpi/linux-headers-rpi/linux-headers-
3.18.7-v7%2b_3.18.7-v7%2b-2_armhf.deb
sudo dpkg -i linux-headers-3.18.7-v7+_3.18.7-v7+-2_armhf.deb
```

answered Mar 30 '15 at 19:26

Ryu_hayabusa
2,382 1 22 27

This is awesome! – Hengjie May 12 '15 at 12:10

When installing the linux headers I get the following error: "./arch/arm/include/asm/memory.h:24:25: fatal error: mach/memory.h: No such file or directory". Any thoughts? – gromit190 Jul 30 '15 at 20:21

I was working on the exact same sample on my RPI with the exact same kernel. I managed to compile the module on my RPI but when I issued insmod I received an error. I followed the

I had to change the Makefile to match the new environment.

```
obj-m += hello-1.o

all:
        make ARCH=arm CROSS_COMPILE=${CCPREFIX} -C /home/cstick/rpi/linux-rpi-
3.6.y M=$(PWD) modules
clean:
        make -C /home/cstick/rpi/linux-rpi-3.6.y M=$(PWD) clean
```

edited Nov 29 '13 at 2:33

answered Nov 28 '13 at 15:25

cstick
376 3 10

0

I meet with the same problem and just fix it by `sudo apt-get install raspberrypi-kernel-headers`

answered Dec 23 '18 at 13:39

Michael Hou
1 1
