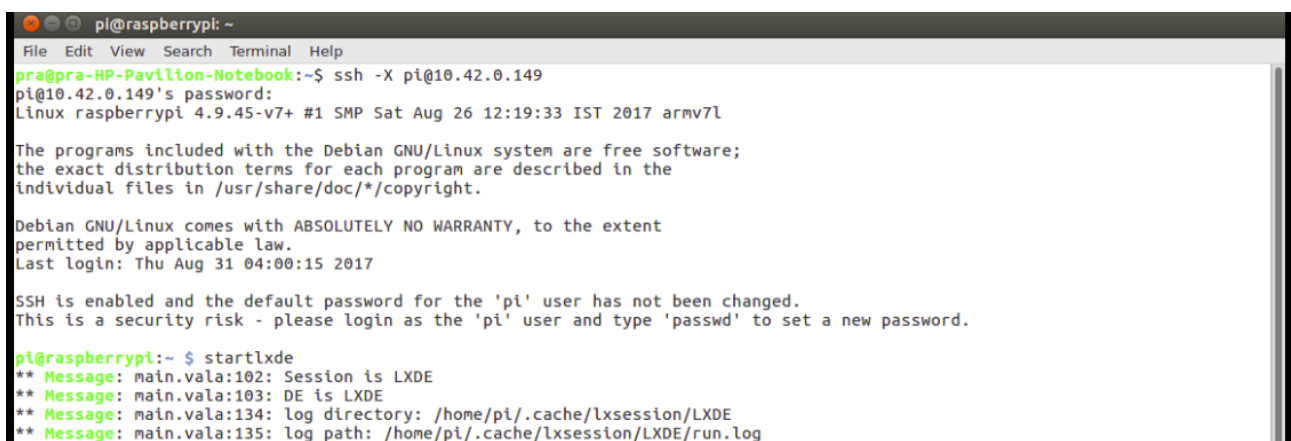


Kernel Modules on Raspbian Stretch

Submitted by:Prashanthi S.K
MT2016520

Procedure followed

I booted my Raspberry Pi with the Raspbian stretch distribution. Thereafter, I tried to set up networking on the pi so that I could connect to it using my laptop instead of using a monitor, keyboard and mouse. I learnt that the network interfaces had undergone changes in Stretch. I first noticed the enx..(mac address) naming instead of the usual eth0/eth1. Also, the normal configuration of /etc/network/interfaces was now split up into two: /etc/network/interfaces and /etc/dhcpd.conf. I initially tried using a combination of vnc server and viewer on Windows. I was able to log in to the pi, but internet sharing (ICS) on Windows was very troublesome. It wouldn't work half of the time. I also had to frequently reboot. I struggled with setting up networking on Windows for hours before I decided to move to Linux (Ubuntu) and use ssh (with x11 forwarding) with lxde. This setup was very easy and worked flawlessly. The only glitch was that ssh needed to be enabled before using this method, which is kind of circular, because you need a monitor to enable ssh when you're trying to avoid using a monitor in the first place. I pondered over this and decided that there must be an alternate method of enabling ssh. Somewhere on the net, I read that this is possible using a few hacks. I didn't try those because I had easy access to a monitor in the lab and directly used raspi-config to enable it. Now, I could connect to my pi using my laptop.



```
pi@raspberrypi: ~  
File Edit View Search Terminal Help  
pra@pra-HP-Pavilion-Notebook:~$ ssh -X pi@10.42.0.149  
pi@10.42.0.149's password:  
Linux raspberrypi 4.9.45-v7+ #1 SMP Sat Aug 26 12:19:33 IST 2017 armv7l  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Thu Aug 31 04:00:15 2017  
  
SSH is enabled and the default password for the 'pi' user has not been changed.  
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.  
  
pi@raspberrypi:~$ startlxde  
** Message: main.vala:102: Session is LXDE  
** Message: main.vala:103: DE is LXDE  
** Message: main.vala:134: log directory: /home/pi/.cache/lxsession/LXDE  
** Message: main.vala:135: log path: /home/pi/.cache/lxsession/LXDE/run.log
```

The next step was to look up how to set up the pi so that it is ready to compile kernel modules. The first resource I found suggested using apt-get to install the kernel headers and that would be sufficient. To be honest, this looked too simple. I took another look at the assignment and in the references I found links to kernel building, firmware and rpi-source. After seeing this, I felt this was the path to follow--to manually compile the kernel, update to the latest and appropriate firmware and then download the source for that version. I was very puzzled because this seemed a very roundabout way of doing what kernel-headers offered in one line. I decided that this second method would teach me a lot more than the first (that being more like a black box) so I started following the steps to build the kernel.

During kernel building using cross-compilation, the first error I encountered was that I was using the vanilla gcc instead of arm gcc. This was quickly solved by an apt-get install. After that, make took very long despite using the -j flag. So I made the mistake of letting it run on its own and not looking at the screen output. After it was done, I took a look and everything seemed fine. I put the SD card back in and booted it up, only to find that both the keyboard and the mouse weren't working. I instantly understood that the kernel hadn't been compiled properly and felt miserable because I had ruined my sd card and I now had to repeat the whole ordeal again. I went back to looking at the steps I had followed. Since I had typed out the commands manually instead of copy-pasting them, I had missed out a space. Instead of arm-gnueabi-hf-INSTALL I had typed arm-gnueabi-hf-INSTALL. This had thrown a few errors initially which I missed out due to the deluge of statements being printed on the screen. I redid the process keeping my eye on the screen throughout. It was then that I realised that I now had a kernel7-backup.img which was a backup of the original vanilla kernel. I felt happy about this because I could always go back to this by adding a line in the config.txt. This gave me some reassurance and confidence and I was glad I followed this procedure. Another point that puzzled me was the existence of two kernel files: kernel and kernel7. I looked it up to find that Pi2 and 3 used 7 and the older ones use kernel.

This time around, my pi booted successfully. I saw that I had two kernels now--the original one and the one I had compiled. I tried compiling the module sachin using the makefile but it gave me an error saying no directory build. I navigated to the build directory of the currently used kernel to see an exclamation mark on both build and source--they were not present. I figured out that this is why rpi-source is required. I installed and ran rpi-source following the instructions on the github wiki page. I downgraded to a lower version of gcc so that it matched the gcc version that my kernel was compiled on. Thereafter, when I ran rpi-source, it gave me an error saying it could not obtain the gcc version. This made me go through the source code of rpi-source and I was astonished to find that it was pretty simple. I did a check of proc/version and gcc --version to find that they matched. So, I decided to use the skip-gcc flag and get the sources directly. I did this and the source files were downloaded--the exclamation marks over build and source vanished. I did a make and all 3 modules compiled successfully.

I happily did an insmod only to get a layout mismatch error. I called up modinfo and found that they were compiled for the wrong version of the kernel---the older one instead of the currently used one. I checked the build and source directories that rpi-source had populated to find the same thing--the source was for a different kernel version. This made me go through the source code of rpi-source again and I noticed rpi-update in one of the lines--that's when it struck me. I hadn't updated the firmware. I immediately did an rpi-update and then re-ran rpi source. This time, the modules were compiled for the right version of the kernel and I had no issue inserting or removing them. Though this whole process was extremely laborious and I had to repeat several of the steps, I learnt a lot from it--far more than if I had just used install headers. Therefore, I have no regrets about having taken the long route. It has taught me a lot.

Observations

- Run make in demo1 directories to build the modules successfully

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~/a1 $ make -f Makefile0  
make -C /lib/modules/4.9.45-v7+/build M=/home/pi/a1 modules  
make[1]: Entering directory '/home/pi/linux-3ce72830a8c8bba33c37ebe4bee71ac3177451b0'  
Building modules, stage 2.  
MODPOST 3 modules  
CC /home/pi/a1/bhalu.mod.o  
LD [M] /home/pi/a1/bhalu.ko  
CC /home/pi/a1/deepa.mod.o  
LD [M] /home/pi/a1/deepa.ko  
CC /home/pi/a1/sachin.mod.o  
LD [M] /home/pi/a1/sachin.ko  
make[1]: Leaving directory '/home/pi/linux-3ce72830a8c8bba33c37ebe4bee71ac3177451b0'  
rm *.mod.[co]  
pi@raspberrypi:~/a1 $ ls  
bhalu.c  deepa.c  Kbuild          Module.symvers  sachin.o  
bhalu.ko  deepa.ko  Makefile0       sachin.c        x86  
bhalu.o  deepa.o  modules.order   sachin.ko  
pi@raspberrypi:~/a1 $  
pi@raspberrypi:~ $ uname -r  
4.9.45-v7+  
pi@raspberrypi:~ $ uname -m  
Linux 4.9.45-v7+ armv7l  
pi@raspberrypi:~ $
```

- Check the proper binary (x86 or ARM) format

By printing modinfo for all the modules, we can see whether it is compiled for ARM or x86

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~/a1 $ modinfo ./sachin.ko  
filename: /home/pi/a1/./sachin.ko  
license: GPL  
srcversion: 30155EC8B9613852C8FCA00  
depends:  
vermagic: 4.9.45-v7+ SMP mod_unload modversions ARMv7 p2v8  
pi@raspberrypi:~/a1 $ modinfo ./deepa.ko  
filename: /home/pi/a1/./deepa.ko  
license: GPL  
srcversion: FE5EFB81A775F50F5ED685B  
depends:  
vermagic: 4.9.45-v7+ SMP mod_unload modversions ARMv7 p2v8  
parm: flag:Simple flag value (int)  
pi@raspberrypi:~/a1 $ modinfo ./bhalu.ko  
filename: /home/pi/a1/./bhalu.ko  
license: GPL  
description: bhalu the happy and colorful bear  
srcversion: 5124E28D83E5C1B300D5E98  
depends:  
vermagic: 4.9.45-v7+ SMP mod_unload modversions ARMv7 p2v8  
parm: debug:Debug level, 0=silent (int)  
pi@raspberrypi:~/a1 $
```

Another method of checking is using the file command.

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~/a1 $ file sachin.ko  
sachin.ko: ELF 32-bit LSB relocatable, ARM, EABI5 version 1 (SYSV), BuildID[sha1]=41b30e8ea410a391afe0f1d300a0bbf7311157c3, not stripped  
pi@raspberrypi:~/a1 $ file deepa.ko  
deepa.ko: ELF 32-bit LSB relocatable, ARM, EABI5 version 1 (SYSV), BuildID[sha1]=d904a6b214792abcc2dc9c338f19a986b89ed1b4, not stripped  
pi@raspberrypi:~/a1 $ file bhalu.ko  
bhalu.ko: ELF 32-bit LSB relocatable, ARM, EABI5 version 1 (SYSV), BuildID[sha1]=080cffffa7d79f80ea61696cf415fd0204f68c75e, not stripped  
pi@raspberrypi:~/a1 $ sudo insmod ./sachin.ko  
pi@raspberrypi:~/a1 $ sudo insmod ./deepa.ko  
pi@raspberrypi:~/a1 $ sudo insmod ./bhalu.ko  
insmod: ERROR: could not load module ./bhalu.ko: No such file or directory  
pi@raspberrypi:~/a1 $ sudo insmod ./bhalu.ko  
pi@raspberrypi:~/a1 $
```

If the same command is run on an x86 kernel object, the result is:

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~/a1/x86 $ file sachin.ko  
sachin.ko: ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), BuildID[sha1]=d0aa4dcef7225ad3b5ab874b35af736d6cca14ac, not stripped
```

- Print module info for both modules and point out the parameter details

A parameter is a runtime option defined for a module whereas an attribute is defined for a device. Therefore, only parameters are displayed in modinfo.

Sachin does not have any parameters therefore, this is empty. Deepa has a parameter called flag which is an integer and has the description simple flag value.

Bhalu has a parameter called debug also an integer bearing the description debug level, 0=silent.

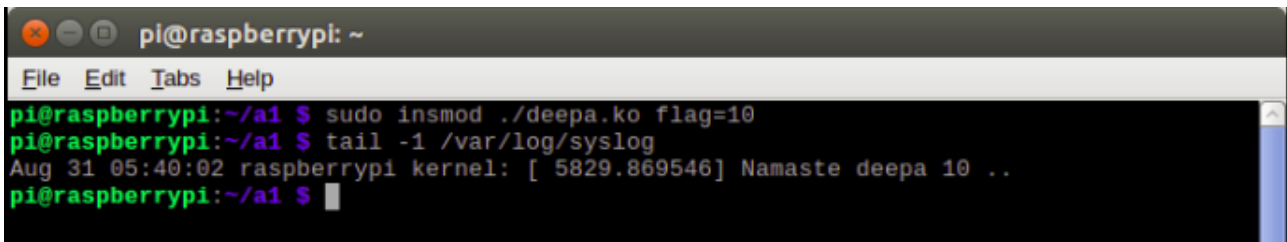
```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~/a1 $ modinfo ./sachin.ko  
filename: /home/pi/a1/./sachin.ko  
license: GPL  
srcversion: 30155EC8B9613852C8FCA00  
depends:  
vermagic: 4.9.45-v7+ SMP mod_unload modversions ARMv7 p2v8  
pi@raspberrypi:~/a1 $ modinfo ./deepa.ko  
filename: /home/pi/a1/./deepa.ko  
license: GPL  
srcversion: FE5EFB81A775F50F5ED685B  
depends:  
vermagic: 4.9.45-v7+ SMP mod_unload modversions ARMv7 p2v8  
parm: flag:Simple flag value (int)  
pi@raspberrypi:~/a1 $ modinfo ./bhalu.ko  
filename: /home/pi/a1/./bhalu.ko  
license: GPL  
description: bhalu the happy and colorful bear  
srcversion: 5124E28D83E5C1B300D5E98  
depends:  
vermagic: 4.9.45-v7+ SMP mod_unload modversions ARMv7 p2v8  
parm: debug:Debug level, 0=silent (int)  
pi@raspberrypi:~/a1 $
```

- Load sachin module and show the registration of the module in sysfs

The registration can be seen by doing an `ls -l` on `sys/module/sachin`. After the module is loaded, an entire directory whose name is same as the module name is created which represents the hierarchy for that module. After removing the module, it is no longer present under `sys/module`.

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~/a1 $ lsmod | grep sachin  
sachin                1004 0  
pi@raspberrypi:~/a1 $ tail -1 /var/log/syslog  
Aug 31 05:19:59 raspberrypi kernel: [ 4627.216932] Namaste sachin ..  
pi@raspberrypi:~/a1 $ ls -l /sys/module/sachin  
total 0  
-r--r--r-- 1 root root 4096 Aug 31 05:20 coresize  
drwxr-xr-x 2 root root  0 Aug 31 05:20 holders  
-r--r--r-- 1 root root 4096 Aug 31 05:20 initsize  
-r--r--r-- 1 root root 4096 Aug 31 05:20 initstate  
drwxr-xr-x 2 root root  0 Aug 31 05:20 notes  
-r--r--r-- 1 root root 4096 Aug 31 05:20 refcnt  
drwxr-xr-x 2 root root  0 Aug 31 05:20 sections  
-r--r--r-- 1 root root 4096 Aug 31 05:20 srcversion  
-r--r--r-- 1 root root 4096 Aug 31 05:20 taint  
--w----- 1 root root 4096 Aug 31 05:20 uevent  
pi@raspberrypi:~/a1 $ sudo rmmod sachin.ko  
pi@raspberrypi:~/a1 $ lsmod | grep sachin  
pi@raspberrypi:~/a1 $ ls -l /sys/module/sachin  
ls: cannot access '/sys/module/sachin': No such file or directory  
pi@raspberrypi:~/a1 $ tail -1 /var/log/syslog  
Aug 31 05:28:30 raspberrypi kernel: [ 5138.378917] Shubham sachin ...  
pi@raspberrypi:~/a1 $ sudo insmod ./sachin.ko  
pi@raspberrypi:~/a1 $
```

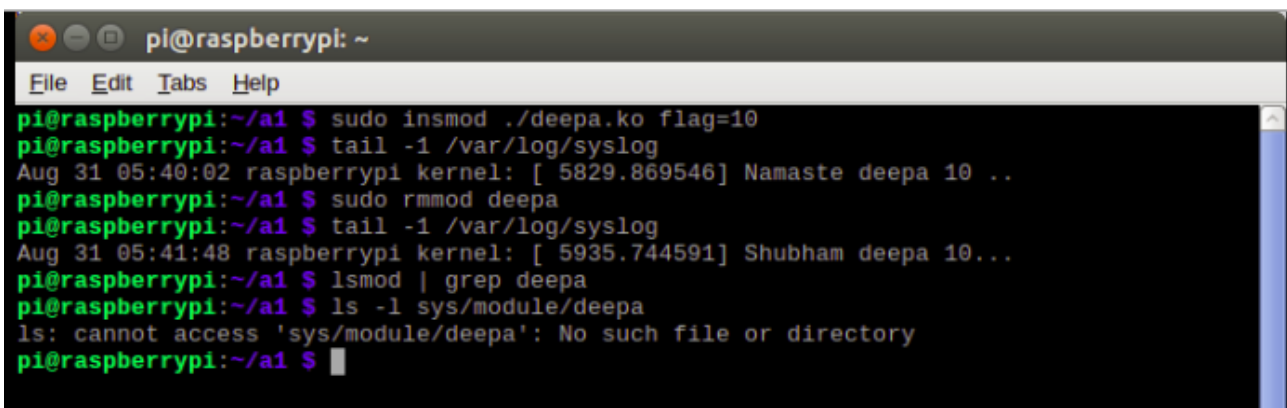
- Load deepa module and show its flag values in syslog



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~/a1 $ sudo insmod ./deepa.ko flag=10  
pi@raspberrypi:~/a1 $ tail -1 /var/log/syslog  
Aug 31 05:40:02 raspberrypi kernel: [ 5829.869546] Namaste deepa 10 ..  
pi@raspberrypi:~/a1 $
```

Deepa is loaded with a flag value of 10 which is reflected in syslog

- Unload deepa module and show that it is no longer loaded

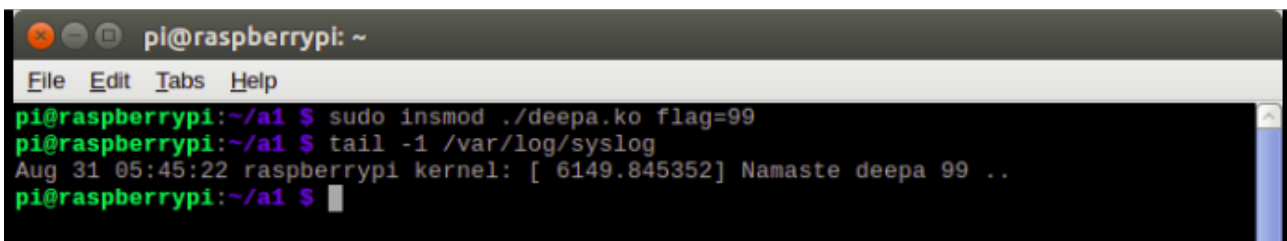


```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~/a1 $ sudo insmod ./deepa.ko flag=10  
pi@raspberrypi:~/a1 $ tail -1 /var/log/syslog  
Aug 31 05:40:02 raspberrypi kernel: [ 5829.869546] Namaste deepa 10 ..  
pi@raspberrypi:~/a1 $ sudo rmmod deepa  
pi@raspberrypi:~/a1 $ tail -1 /var/log/syslog  
Aug 31 05:41:48 raspberrypi kernel: [ 5935.744591] Shubham deepa 10...  
pi@raspberrypi:~/a1 $ lsmod | grep deepa  
pi@raspberrypi:~/a1 $ ls -l sys/module/deepa  
ls: cannot access 'sys/module/deepa': No such file or directory  
pi@raspberrypi:~/a1 $
```

Deepa is no longer present in sys/module or proc/modules since it has been removed.

- Reload deepa with a different parameter and show its log entries

This time flag is set to 99.



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~/a1 $ sudo insmod ./deepa.ko flag=99  
pi@raspberrypi:~/a1 $ tail -1 /var/log/syslog  
Aug 31 05:45:22 raspberrypi kernel: [ 6149.845352] Namaste deepa 99 ..  
pi@raspberrypi:~/a1 $
```

- Load bhalu module and show read/write ops on its attributes in sysfs

Color can be written to whereas name is readonly and this can be seen while trying to write to name

```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~/a1 $ sudo insmod ./bhalu.ko
pi@raspberrypi:~/a1 $ cat /sys/devices/bhalu1/name
bhalu1
pi@raspberrypi:~/a1 $ cat /sys/devices/bhalu1/color
off
pi@raspberrypi:~/a1 $ echo blue | sudo tee /sys/devices/bhalu1/color
blue
pi@raspberrypi:~/a1 $ cat /sys/devices/bhalu1/color
blue
pi@raspberrypi:~/a1 $ echo newbhalu | sudo tee /sys/devices/bhalu1/name
tee: /sys/devices/bhalu1/name: Permission denied
newbhalu
pi@raspberrypi:~/a1 $

```

- What happens in /sys/module/ and /sys/devices/ directories when you load and unload bhalu module.

A directory called bhalu is created in sys/module and a directory called bhalu1(device name) is created in sys/devices. These directories are deleted when bhalu is unloaded. The device attributes are present under bhalu1 whereas the module details under bhalu.

```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~/a1 $ ls -l /sys/module/bhalu
total 0
-r--r--r-- 1 root root 4096 Aug 31 05:53 coresize
drwxr-xr-x 2 root root    0 Aug 31 05:53 holders
-r--r--r-- 1 root root 4096 Aug 31 05:53 initsize
-r--r--r-- 1 root root 4096 Aug 31 05:53 initstate
drwxr-xr-x 2 root root    0 Aug 31 05:53 notes
-r--r--r-- 1 root root 4096 Aug 31 05:53 refcnt
drwxr-xr-x 2 root root    0 Aug 31 05:53 sections
-r--r--r-- 1 root root 4096 Aug 31 05:53 srcversion
-r--r--r-- 1 root root 4096 Aug 31 05:53 taint
--w----- 1 root root 4096 Aug 31 05:48 uevent
pi@raspberrypi:~/a1 $ ls -l /sys/devices/bhalu1
total 0
-rw-r--r-- 1 root root 4096 Aug 31 05:49 color
-r--r--r-- 1 root root 4096 Aug 31 05:49 name
drwxr-xr-x 2 root root    0 Aug 31 05:52 power
-rw-r--r-- 1 root root 4096 Aug 31 05:52 uevent
pi@raspberrypi:~/a1 $ sudo rmmod bhalu
pi@raspberrypi:~/a1 $ ls -l /sys/module/bhalu
ls: cannot access '/sys/module/bhalu': No such file or directory
pi@raspberrypi:~/a1 $ ls -l /sys/devices/bhalu1
ls: cannot access '/sys/devices/bhalu1': No such file or directory
pi@raspberrypi:~/a1 $

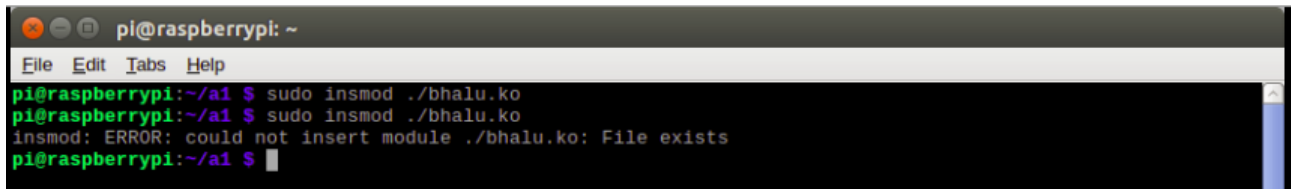
```

- Module bhalu registers a device but not a driver. Observe what happens in devices/ and drivers/ of /sys/bus/platform/ when you load and unload bhalu module.

There are no change in both /devices and /drivers. This is because although bhalu.c includes the linux header for platform_devices, it only declares itself to be a normal device and driver as opposed to a platform one. If the platform device register and unregister calls were used then there would be changes under these folders.

- What happens if you load bhalu a second time?

The loading fails because the module already exists in the kernel.



```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~/a1 $ sudo insmod ./bhalu.ko
pi@raspberrypi:~/a1 $ sudo insmod ./bhalu.ko
insmod: ERROR: could not insert module ./bhalu.ko: File exists
pi@raspberrypi:~/a1 $

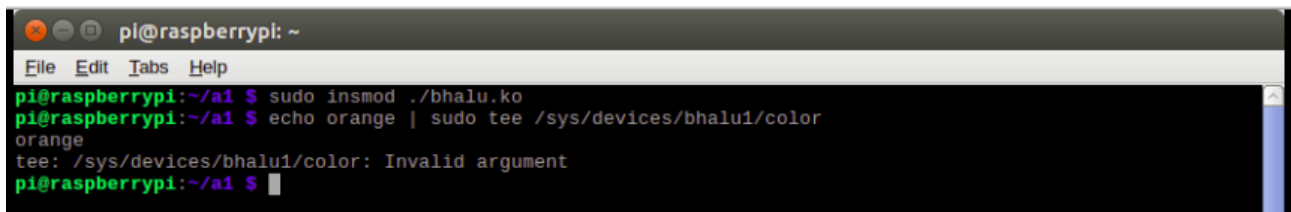
```

- What are the valid color values for bhalu?

The valid colors are off, red, green, blue, white which internally translate to 0, 1, 2, 3, 4.

- What happens if you use an invalid color value?

Tee throws an invalid argument error. This is because in the setter function for color in bhalu, the for loop iterates through all the valid colors comparing them with the entered color at each step. If no match is found, it returns EINVAL which is the errno for invalid argument.



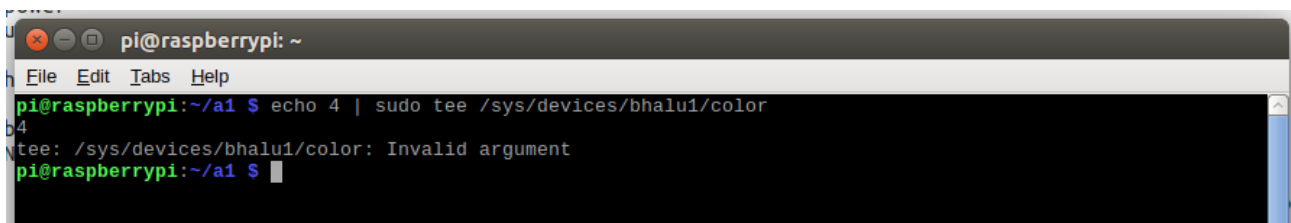
```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~/a1 $ sudo insmod ./bhalu.ko
pi@raspberrypi:~/a1 $ echo orange | sudo tee /sys/devices/bhalu1/color
orange
tee: /sys/devices/bhalu1/color: Invalid argument
pi@raspberrypi:~/a1 $

```

- What happens if you use a number to set color?

We still get the same invalid argument error. This is because although the strings are internally mapped to numbers using #define, the reverse mapping is not present i.e. numbers are not translated to the strings they represent. Therefore, the strcmp fails and it returns EINVAL.



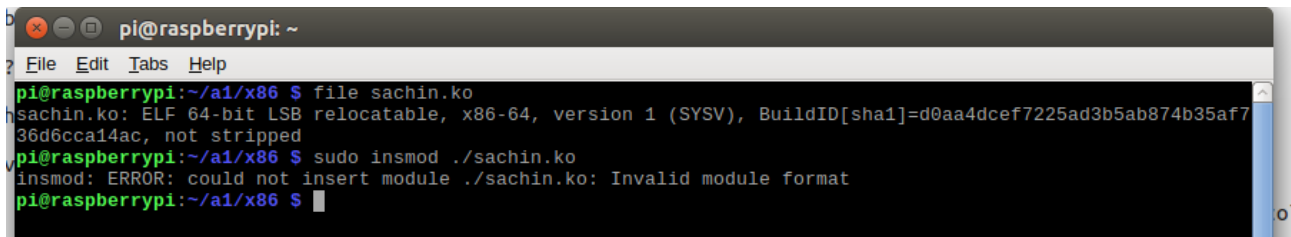
```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~/a1 $ echo 4 | sudo tee /sys/devices/bhalu1/color
4
tee: /sys/devices/bhalu1/color: Invalid argument
pi@raspberrypi:~/a1 $

```


- What happens if you try to load a module compiled for x86 kernel on Raspberry Pi?

The load operation fails with an invalid module format error. Insmod checks whether the kernel that the module was compiled for and the kernel that it is currently being loaded on are the same. If not, it throws this error. The kernel that the module has been compiled for can be verified using modinfo. Current kernel is obtained using uname -r.



```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~/a1/x86 $ file sachin.ko
sachin.ko: ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), BuildID[sha1]=d0aa4dcef7225ad3b5ab874b35af736d6cca14ac, not stripped
pi@raspberrypi:~/a1/x86 $ sudo insmod ./sachin.ko
insmod: ERROR: could not insert module ./sachin.ko: Invalid module format
pi@raspberrypi:~/a1/x86 $

```

Observations from Readme

- insmod uses file paths while lsmod and rmmod uses module name. Why?

This is because before the file is loaded the kernel is not aware of the module name--hence the path is to be specified. Whereas once the module is loaded, it is given a unique name by the kernel. To unload it, this name can be used.

- bhalu combines both driver code and device spec into a single module. Can you guess which lines deal with driver and which deal with device?

Driver and device have separate structures referring to them. The setter and getter functions for the attributes, the register and release device all are part of the device code. Whereas module init and exit, bhalu_init and bhalu_exit are part of the driver code.

- In /sys/devices/bhalu1/ you will notice attributes other than the ones you defined. Read Documentation/driver-model/overview.txt for clues.

The other attributes present apart from color are name, power and uevent. These are created by the kernel and are part of the global layer in the driver model. The name file contains the name assigned by the kernel to the module and power indicates the current power state.

Time taken: Around 22 hrs spread out over 6 days

References(in addition to those mentioned on LMS)

- 1)<https://www.kernel.org/doc/Documentation/driver-model/platform.txt>
- 2)<https://github.com/notro/rpi-source/wiki>
- 3)<https://github.com/notro/rpi-source/blob/master/rpi-source>
- 4)Basic tmux tutorial <http://www.hamvocke.com/blog/a-quick-and-easy-guide-to-tmux/>
- 5)<https://stackoverflow.com/questions/15610570/what-is-the-difference-between-a-linux-platform-driver-and-normal-device-driver>
- 6)<https://stackoverflow.com/questions/3220277/what-do-the-makefile-symbols-and-mean>
- 7)man pages for tee,insmod,modinfo,rmmod,dmesg