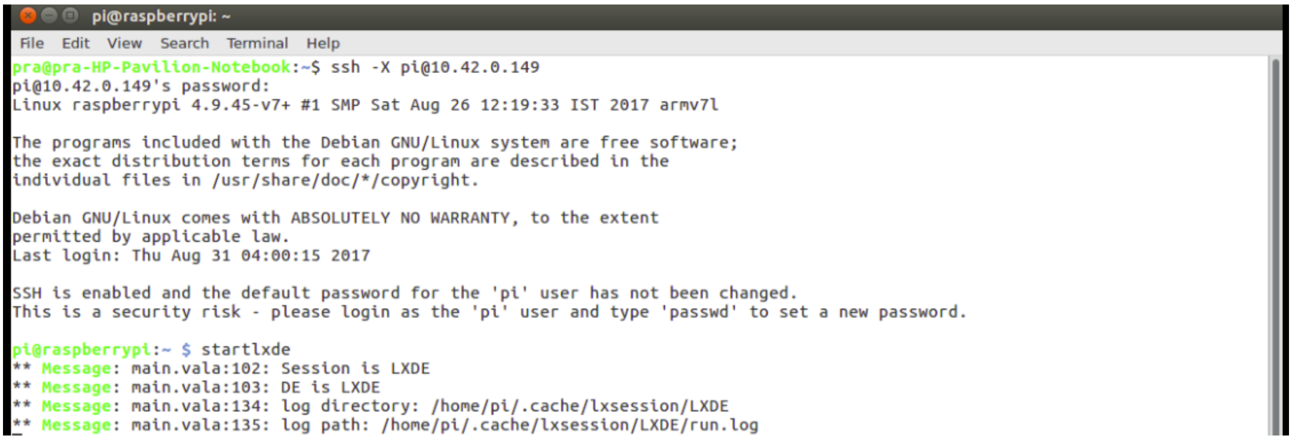
<u>Kernel Modules on Raspbian Stretch</u>

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.I learnt that the network interfaces had undergone changes in Stretch.I first noticed the enx..(mac address) naming instead of the ususal eth0.Also,the normal configuration of /etc/network/interfaces was now split up into two:/etc/network/interfaces and /etc/dhcpcd.conf.I inititally 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 netwroking on Windows for hours before I decided to move to Linux(Ubuntu) and use ssh 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 teh first palce.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.



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 ap-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.

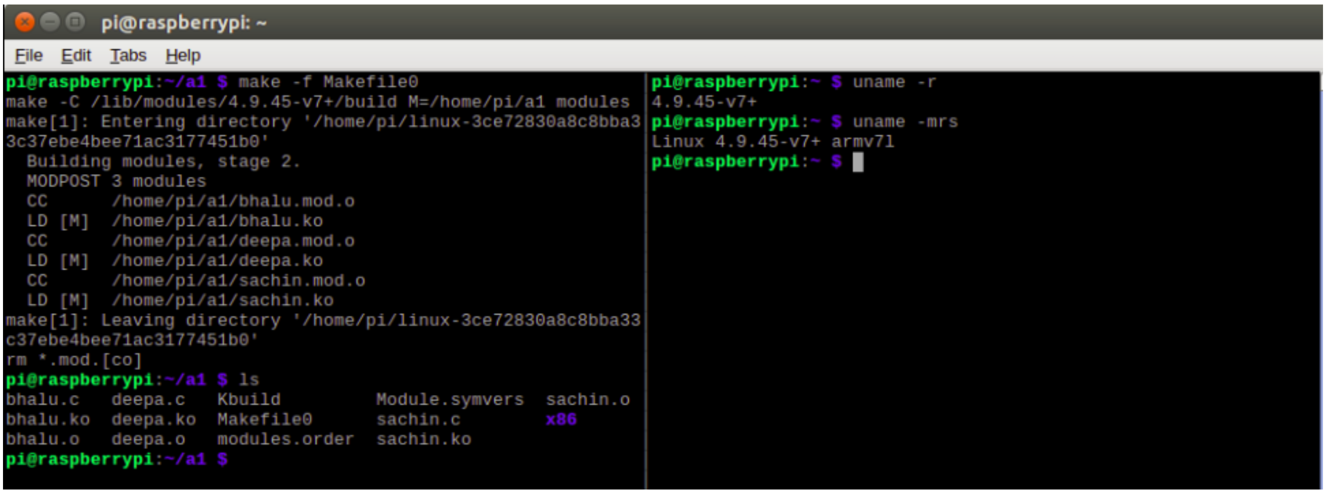
During kernel building using cross-compilation,the first error I encountered was that I was using the vanilla gcc instead of arm gcc.This was quickly solved by an ap-get isntall.After that,make took very long despite using the -jn flag.So I made the mistake of letting it run on its own and not looking 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 becuase I had runied 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 teh commands manually instead of copy-pasting them,I had missed out a space.Instead of arm-gnueabihf- INSTALL I had typed arm-gnueabihf-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 kernel-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 compiled.I tried compiling the module sachin using the makefile but it gave me an error saying no directory build.I naviagted to the build directory of the currently used kernel to build an exclamatory 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 wiki page.I downgraded to a lower version of gcc so that it matched the gcc version that my kernel was compiled on.Therafter,when I ran rpi-source,it gave me an error saying it could not obtain the gcc version.This made me go through the spource 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 taht they matched.So,I decided to use the skip-gcc flag and get the sources directly.I did this and teh source files were downloaded--the exclamatory 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.There ended my happiness.I called up modinfo and found that they were compiled for the wrong version of the kernel---the older one isntaed 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 trhough 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 isntall 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

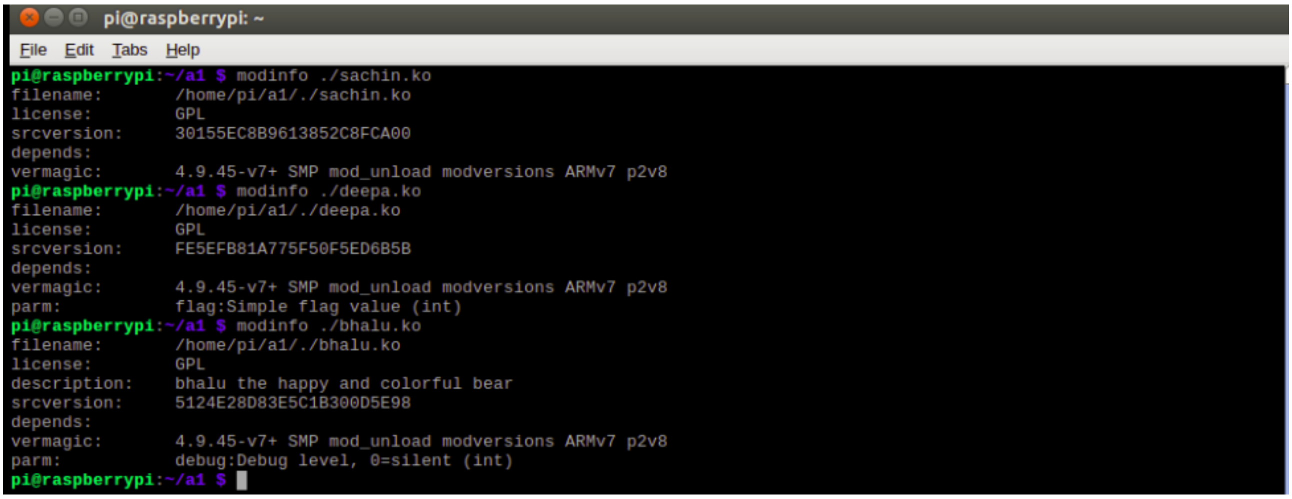


```
pi@raspberrypi:~/a1 $ make -f Makefile0
make -C /lib/modules/4.9.45-v7+/build M=/home/pi/a1 modules
make[1]: Entering directory '/home/pi/linux-3ce72830a8c8bba3
3c37ebe4bee71ac3177451b0'
  Building modules, stage 2.
  MODPOST 3 modules
  CC      /home/pi/a1/bhalu.mod.o
  LD [M]  /home/pi/a1/bhalu.ko
  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-3ce72830a8c8bba33
c37ebe4bee71ac3177451b0'
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 -mrs
Linux 4.9.45-v7+ armv7l
pi@raspberrypi:~ $
```

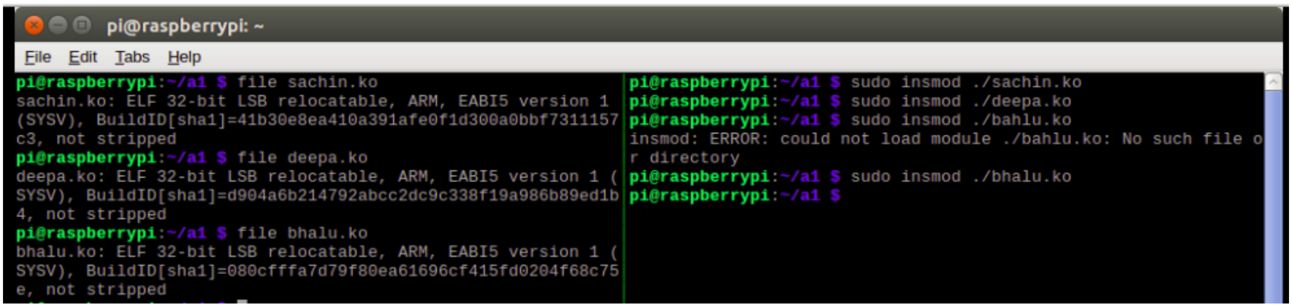
- Check the proper binary (x86 or ARM) format

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



```
pi@raspberrypi:~/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:     FE5EFB81A775F50F5ED6B5B
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:~/a1 $ file sachin.ko
sachin.ko: ELF 32-bit LSB relocatable, ARM, EABI5 version 1
(SYSV), BuildID[sha1]=41b30e8ea410a391afe0f1d300a0bbf7311157
c3, not stripped
pi@raspberrypi:~/a1 $ file deepa.ko
deepa.ko: ELF 32-bit LSB relocatable, ARM, EABI5 version 1 (
SYSV), BuildID[sha1]=d904a6b214792abcc2dc9c338f19a986b89ed1b
4, not stripped
pi@raspberrypi:~/a1 $ file bhalu.ko
bhalu.ko: ELF 32-bit LSB relocatable, ARM, EABI5 version 1 (
SYSV), BuildID[sha1]=080cfffa7d79f80ea61696cf415fd0204f68c75
e, not stripped
```

```
pi@raspberrypi:~/a1 $ sudo insmod ./sachin.ko
pi@raspberrypi:~/a1 $ sudo insmod ./deepa.ko
pi@raspberrypi:~/a1 $ sudo insmod ./bahlu.ko
insmod: ERROR: could not load module ./bahlu.ko: No such file o
r directory
pi@raspberrypi:~/a1 $ sudo insmod ./bhalu.ko
pi@raspberrypi:~/a1 $
```