Advanced Practical Embedded Software

Homework 1 (50 Pts) - Due Wednesday 1/24 (midnight)

Name: Nikhil Divekar

Problem Set

[Problem 1 - 15 pts] Complete the Surveys & Agreements

- Lab Policy Agreement: https://goo.gl/forms/8rTSW3hfbPUb4RFh1
    - We will not be providing lab facilities this summer due to construction
      DONE
- Office Hours Survey: https://goo.gl/forms/jNWrsLzXaJ9PxvuP2
    - We will use this to create our office hour times based on your schedule!
      DONE
- Homework 1 Survey: https://goo.gl/forms/t0R6HiuVU6MPshCv2
    - Development kits need to be shipped to distance students.
      DONE

[Problem 2] Go To D2L and get familiar with the site. Do the following:

- Look at the Schedule. Note the midterm and final exam time in the syllabus.
- Read the C-coding Guidelines for the ESE Program (**ESE-C-CodingGuidelines.pdf**)

DONE

[Problem 3 - 5 pts] Set up your Slack account

We will be using this as an open discussion forum throughout the semester. Please be professional and do NOT share answers on this site. Use it to help one another and seek help from the instructor, Student Assistants (SA), and your classmates. I have already sent out a request for you to join. Please check your inbox (colroado.edu email). Add a picture, and use your actual name. https://ecee.slack.com.**Post to the channel your name, and the type of sensors you think you want to use for the final project.**

DONE

[Problem 4] Find a partner for the projects and acquire the development Kits.

Once you have a partner, you will need to meet with the TAs to get the development kits. They should announce when they are ready to hand out kits. Indicate your needs for these in the survey above. Please read the checkout policy carefully when receiving a kit from the department. 1 kit per group. Wait for contact our TAs Sameer.Vaze@colorado.eduor Richard.Noronha@colorado.eduon the status of your kit. Or work alone. Kits will go to groups first.

DONE

[Problem 5] Setup your environment

- Download Virtualbox
- Download the Ubuntu 16.04 LTS
- Install packages related to git, arm linux compiler(s) and any editors of your choosing.

DONE

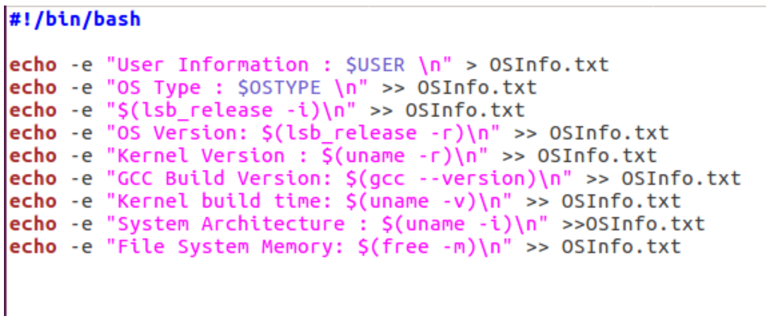
[Problem 6 - 10 Pts] Create a System Info Script

Using bash, write a simple script that can collect some of the important parts about your operating system platform and write that to a file. Add this file to your own git repository in a folder called utils or bin. **Include the script in your .pdf on dropbox**.The items that should be included in the output file are:

- User Information
- Operating System Type/Brand
- OS Distribution
- OS Version
- OS Version
- Kernel Version
- Kernel gcc version build
- Kernel build time
- System Architecture information

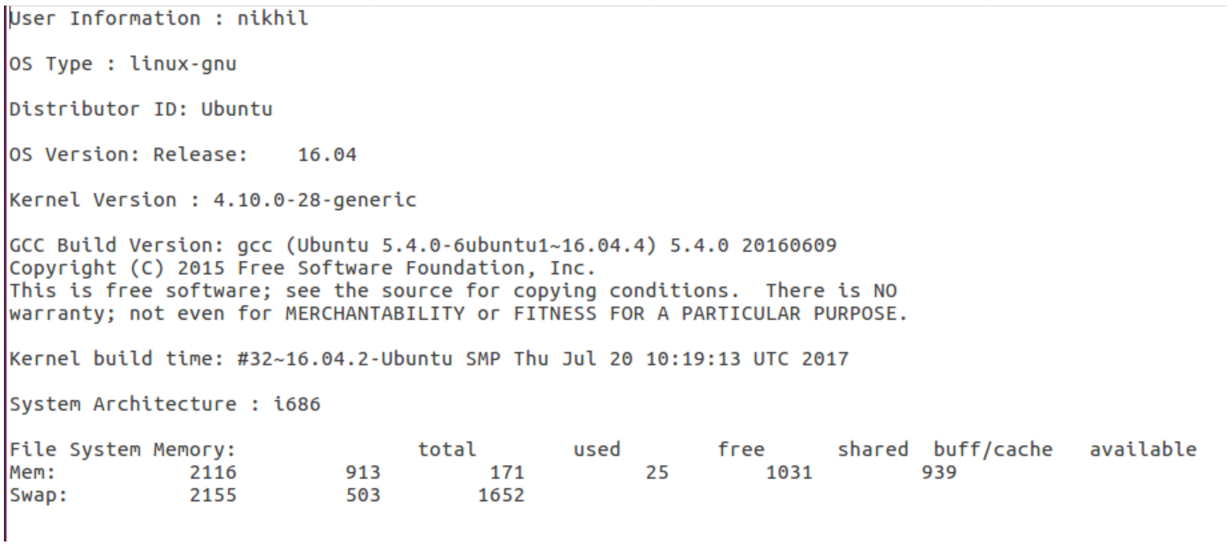
Information on File System Memory

The screenshots for the script that outputs some OS info to the text is given below



```
#!/bin/bash

echo -e "User Information : $USER \n" > OSInfo.txt
echo -e "OS Type : $OSTYPE \n" >> OSInfo.txt
echo -e "$(lsb_release -i)\n" >> OSInfo.txt
echo -e "OS Version: $(lsb_release -r)\n" >> OSInfo.txt
echo -e "Kernel Version : $(uname -r)\n" >> OSInfo.txt
echo -e "GCC Build Version: $(gcc --version)\n" >> OSInfo.txt
echo -e "Kernel build time: $(uname -v)\n" >> OSInfo.txt
echo -e "System Architecture : $(uname -i)\n" >>OSInfo.txt
echo -e "File System Memory: $(free -m)\n" >> OSInfo.txt
```

The screenshot of the output file is also given below



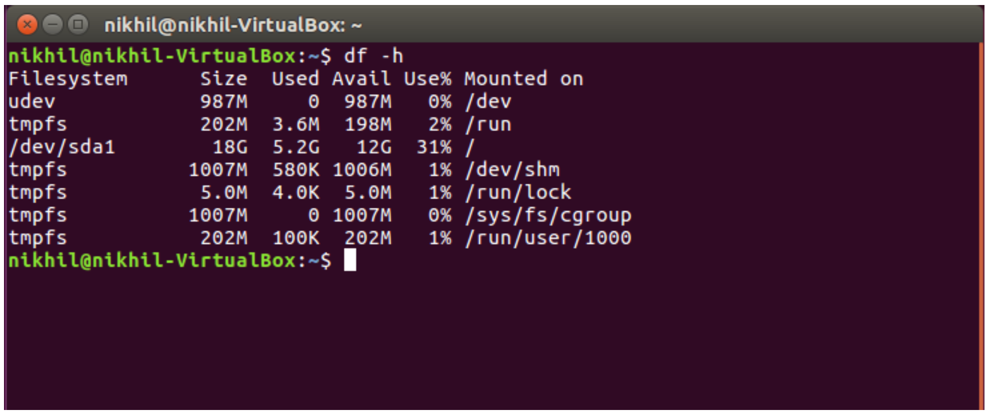
```
User Information : nikhil

OS Type : linux-gnu

Distributor ID: Ubuntu

OS Version: Release:    16.04

Kernel Version : 4.10.0-28-generic

GCC Build Version: gcc (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Kernel build time: #32~16.04.2-Ubuntu SMP Thu Jul 20 10:19:13 UTC 2017

System Architecture : i686

File System Memory:              total        used        free      shared  buff/cache   available
Mem:           2116         913         171          25        1031         939
Swap:          2155         503        1652
```

[Problem 7 - 10pts] Clone and build the Linux Kernel(s) to your VM

Before running any clone, run the command below to see how much space you have on your system. You should have at least 12GB of free memory on your system before building

$ df -h

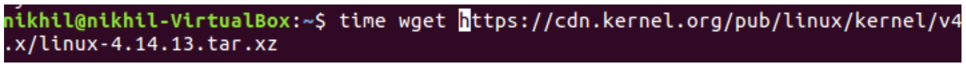
I see this when I run this command. It shows the available space.



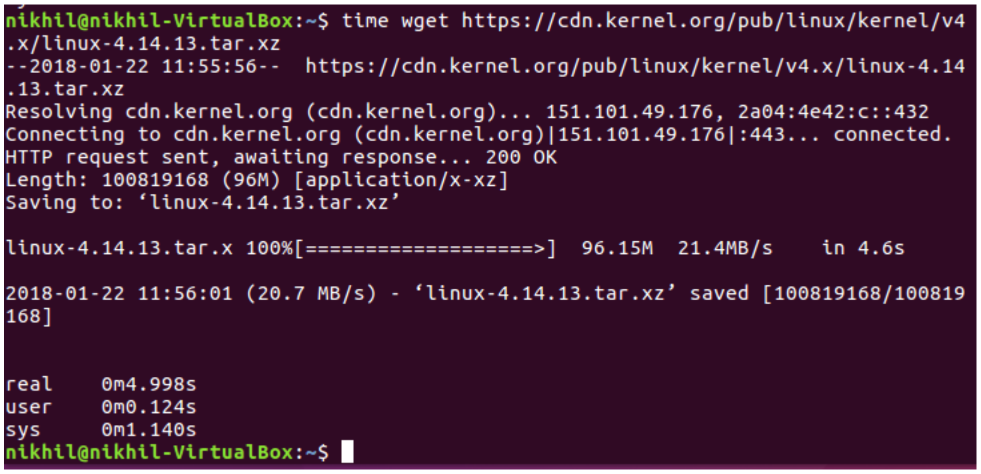
```
nikhil@nikhil-VirtualBox:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            987M     0  987M   0% /dev
tmpfs           202M  3.6M  198M   2% /run
/dev/sda1        18G  5.2G   12G  31% /
tmpfs          1007M  580K 1006M   1% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs          1007M     0 1007M   0% /sys/fs/cgroup
tmpfs           202M  100K  202M   1% /run/user/1000
nikhil@nikhil-VirtualBox:~$
```

Download the kernel source code, the 4.14.13 stable release. I suggest pulling this from torvalds' repository on github so you can easily update your repo (https://github.com/torvalds/linux). Or you can pull kernel from the https://www.kernel.org/site. You can pull files over http using wget. So you can run this:

$ wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.14.13.tar.xz

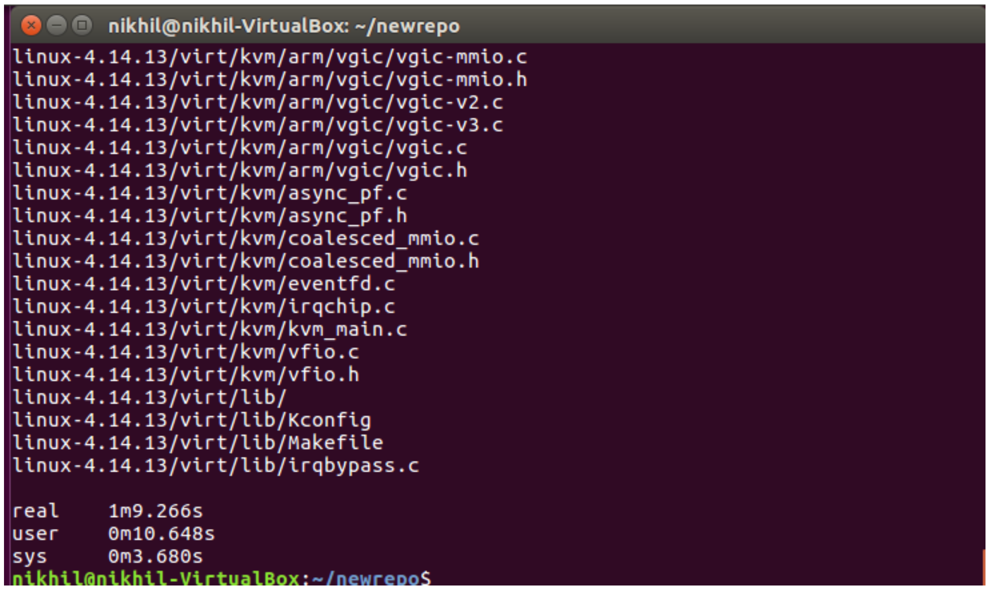


```
nikhil@nikhil-VirtualBox:~$ time wget https://cdn.kernel.org/pub/linux/kernel/v4
.x/linux-4.14.13.tar.xz
```



```
nikhil@nikhil-VirtualBox:~$ time wget https://cdn.kernel.org/pub/linux/kernel/v4
.x/linux-4.14.13.tar.xz
--2018-01-22 11:55:56--  https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.14
.13.tar.xz
Resolving cdn.kernel.org (cdn.kernel.org)... 151.101.49.176, 2a04:4e42:c::432
Connecting to cdn.kernel.org (cdn.kernel.org)|151.101.49.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 100819168 (96M) [application/x-xz]
Saving to: ‘linux-4.14.13.tar.xz’

linux-4.14.13.tar.x 100%[===================>]  96.15M  21.4MB/s    in 4.6s

2018-01-22 11:56:01 (20.7 MB/s) - ‘linux-4.14.13.tar.xz’ saved [100819168/100819
168]


real    0m4.998s
user    0m0.124s
sys     0m1.140s
nikhil@nikhil-VirtualBox:~$
```

Time keyword is used to get the time to download the package. It took 5 seconds to download the package



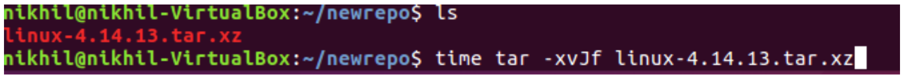
nikhil@nikhil-VirtualBox: ~/newrepo

```
linux-4.14.13/virt/kvm/arm/vgic/vgic-mmio.c
linux-4.14.13/virt/kvm/arm/vgic/vgic-mmio.h
linux-4.14.13/virt/kvm/arm/vgic/vgic-v2.c
linux-4.14.13/virt/kvm/arm/vgic/vgic-v3.c
linux-4.14.13/virt/kvm/arm/vgic/vgic.c
linux-4.14.13/virt/kvm/arm/vgic/vgic.h
linux-4.14.13/virt/kvm/async_pf.c
linux-4.14.13/virt/kvm/async_pf.h
linux-4.14.13/virt/kvm/coalesced_mmio.c
linux-4.14.13/virt/kvm/coalesced_mmio.h
linux-4.14.13/virt/kvm/eventfd.c
linux-4.14.13/virt/kvm/irqchip.c
linux-4.14.13/virt/kvm/kvm_main.c
linux-4.14.13/virt/kvm/vfio.c
linux-4.14.13/virt/kvm/vfio.h
linux-4.14.13/virt/lib/
linux-4.14.13/virt/lib/Kconfig
linux-4.14.13/virt/lib/Makefile
linux-4.14.13/virt/lib/irqbypass.c

real    1m9.266s
user    0m10.648s
sys     0m3.680s
nikhil@nikhil-VirtualBox:~/newrepo$
```

If you pull it from the kernel release page, untar this using the **tar** command. Look up the appropriate options. Start by typing in "man tar". If you clone the git repository **record the time it takes you to clone the repo and the size of the source repo (use the following).**

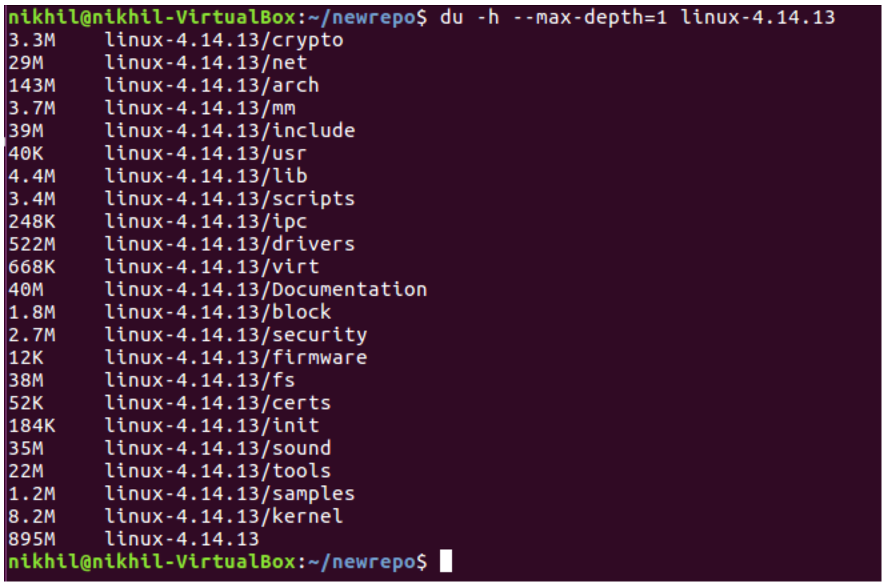
$ du -h --max-depth=1 *<folder>*

I used the wget command to pull the linux from the kernel page and used the following command to untar the downloaded package tar -xvJf linux-4.14.13.tar.xz

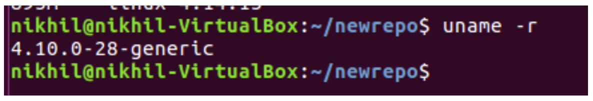


```
nikhil@nikhil-VirtualBox:~/newrepo$ ls
linux-4.14.13.tar.xz
nikhil@nikhil-VirtualBox:~/newrepo$ time tar -xvJf linux-4.14.13.tar.xz
```

Du command shows the files system details in the linux folder



```
nikhil@nikhil-VirtualBox:~/newrepo$ du -h --max-depth=1 linux-4.14.13
3.3M    linux-4.14.13/crypto
29M     linux-4.14.13/net
143M    linux-4.14.13/arch
3.7M    linux-4.14.13/mm
39M     linux-4.14.13/include
40K     linux-4.14.13/usr
4.4M    linux-4.14.13/lib
3.4M    linux-4.14.13/scripts
248K    linux-4.14.13/ipc
522M    linux-4.14.13/drivers
668K    linux-4.14.13/virt
40M     linux-4.14.13/Documentation
1.8M    linux-4.14.13/block
2.7M    linux-4.14.13/security
12K     linux-4.14.13/firmware
38M     linux-4.14.13/fs
52K     linux-4.14.13/certs
184K    linux-4.14.13/init
35M     linux-4.14.13/sound
22M     linux-4.14.13/tools
1.2M    linux-4.14.13/samples
8.2M    linux-4.14.13/kernel
895M    linux-4.14.13
nikhil@nikhil-VirtualBox:~/newrepo$
```



```
nikhil@nikhil-VirtualBox:~/newrepo$ uname -r
4.10.0-28-generic
nikhil@nikhil-VirtualBox:~/newrepo$
```

This shows the original linux version

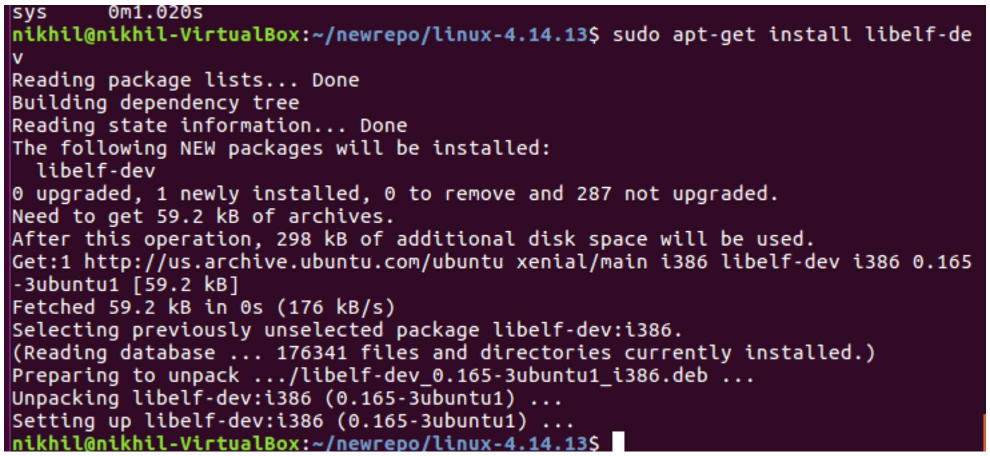
Before you build, find something to do more fun, it may take a few hours depending on how much RAM you gave your machine. My favorite is looking at the subreddit /r/corgi. You should build with the default. Be sure to use the "-j" option to specific more cores if you have them. These options will allow your build to finish more quickly and you likely do not need all of the standard features.

Build your kernel. Report the commands you used in your homework.

Before building we have to install some dependenices.

Commands for installing this packages are given below:

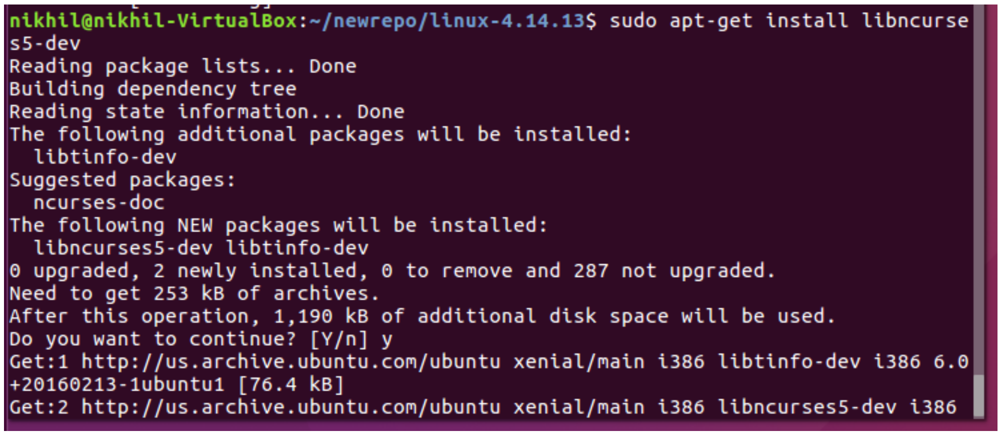
sudo apt-get install libelf-dev



```
sys     0m1.020s
nikhil@nikhil-VirtualBox:~/newrepo/linux-4.14.13$ sudo apt-get install libelf-de
v
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  libelf-dev
0 upgraded, 1 newly installed, 0 to remove and 287 not upgraded.
Need to get 59.2 kB of archives.
After this operation, 298 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/main i386 libelf-dev i386 0.165
-3ubuntu1 [59.2 kB]
Fetched 59.2 kB in 0s (176 kB/s)
Selecting previously unselected package libelf-dev:i386.
(Reading database ... 176341 files and directories currently installed.)
Preparing to unpack .../libelf-dev_0.165-3ubuntu1_i386.deb ...
Unpacking libelf-dev:i386 (0.165-3ubuntu1) ...
Setting up libelf-dev:i386 (0.165-3ubuntu1) ...
nikhil@nikhil-VirtualBox:~/newrepo/linux-4.14.13$
```

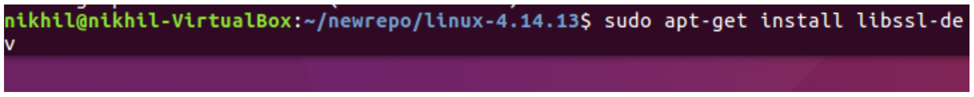
Dependencies can be installed using:

sudo apt-get install libncurses-dev



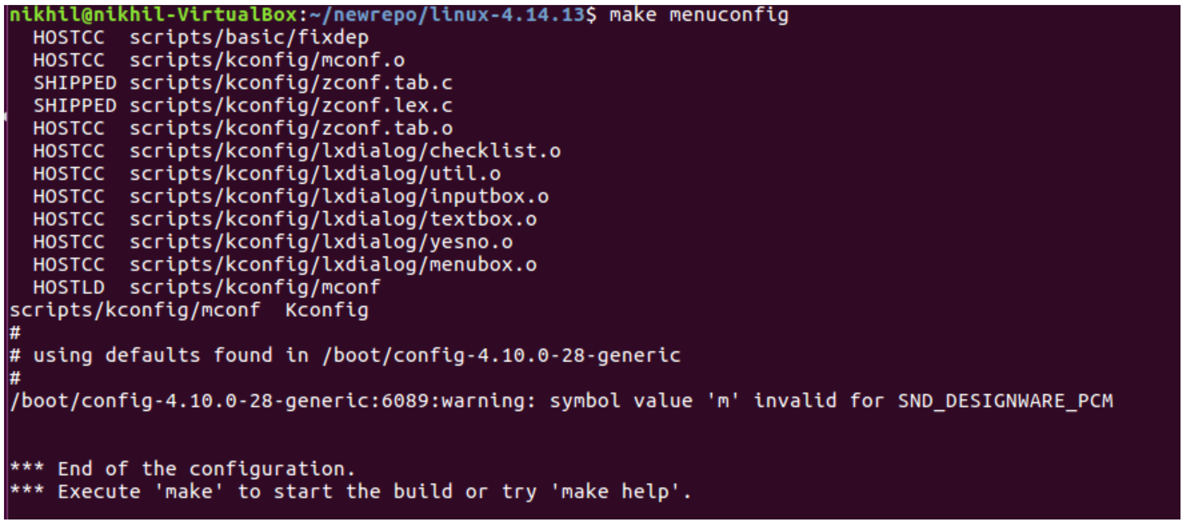
```
nikhil@nikhil-VirtualBox:~/newrepo/linux-4.14.13$ sudo apt-get install libncurse
s5-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libtinfo-dev
Suggested packages:
  ncurses-doc
The following NEW packages will be installed:
  libncurses5-dev libtinfo-dev
0 upgraded, 2 newly installed, 0 to remove and 287 not upgraded.
Need to get 253 kB of archives.
After this operation, 1,190 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/main i386 libtinfo-dev i386 6.0
+20160213-1ubuntu1 [76.4 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu xenial/main i386 libncurses5-dev i386
```

sudo apt-get install libssl-dev

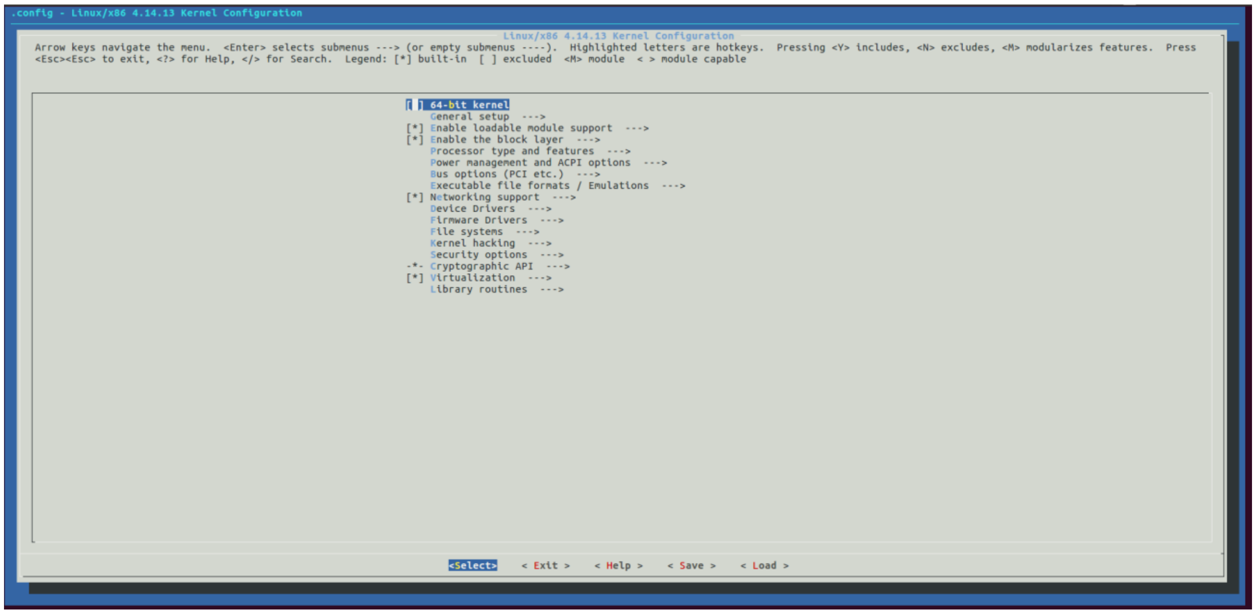


```
nikhil@nikhil-VirtualBox:~/newrepo/linux-4.14.13$ sudo apt-get install libssl-de
v
```

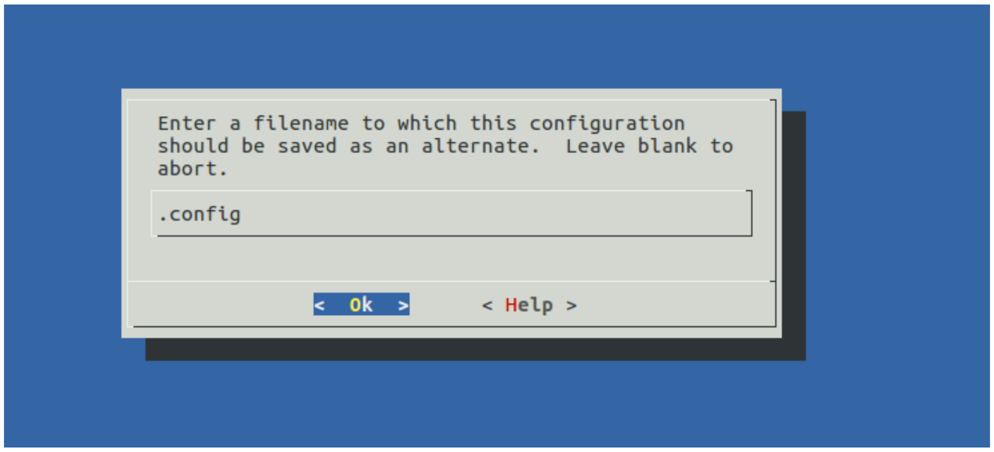
Then we do the menuconfig and save the current config in .config file



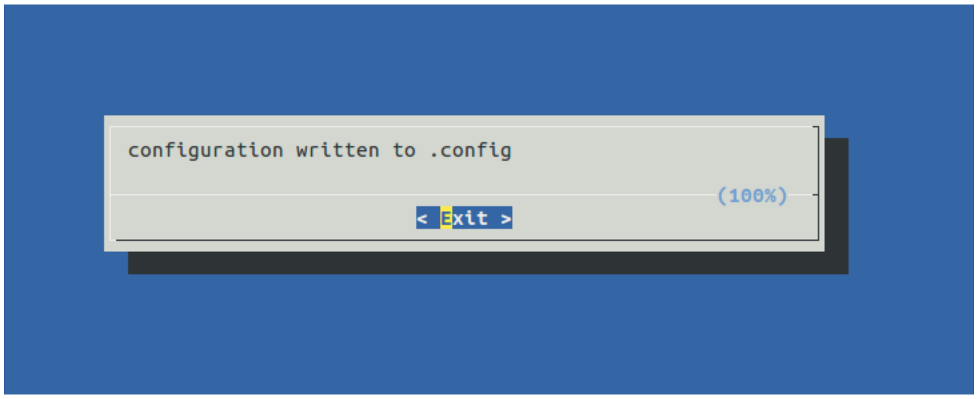
```
nikhil@nikhil-VirtualBox:~/newrepo/linux-4.14.13$ make menuconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/mconf.o
  SHIPPED scripts/kconfig/zconf.tab.c
  SHIPPED scripts/kconfig/zconf.lex.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
  HOSTCC  scripts/kconfig/lxdialog/util.o
  HOSTCC  scripts/kconfig/lxdialog/inputbox.o
  HOSTCC  scripts/kconfig/lxdialog/textbox.o
  HOSTCC  scripts/kconfig/lxdialog/yesno.o
  HOSTCC  scripts/kconfig/lxdialog/menubox.o
  HOSTLD  scripts/kconfig/mconf
scripts/kconfig/mconf  Kconfig
#
# using defaults found in /boot/config-4.10.0-28-generic
#
/boot/config-4.10.0-28-generic:6089:warning: symbol value 'm' invalid for SND_DESIGNWARE_PCM


*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.
```



```
.config - Linux/x86 4.14.13 Kernel Configuration
                         Linux/x86 4.14.13 Kernel Configuration
 Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press
 <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                    [ ] 64-bit kernel
                        General setup  --->
                    [*] Enable loadable module support  --->
                    [*] Enable the block layer  --->
                        Processor type and features  --->
                        Power management and ACPI options  --->
                        Bus options (PCI etc.)  --->
                        Executable file formats / Emulations  --->
                    [*] Networking support  --->
                        Device Drivers  --->
                        Firmware Drivers  --->
                        File systems  --->
                        Kernel hacking  --->
                        Security options  --->
                    -*- Cryptographic API  --->
                    [*] Virtualization  --->
                        Library routines  --->

                 <Select>    < Exit >    < Help >    < Save >    < Load >
```

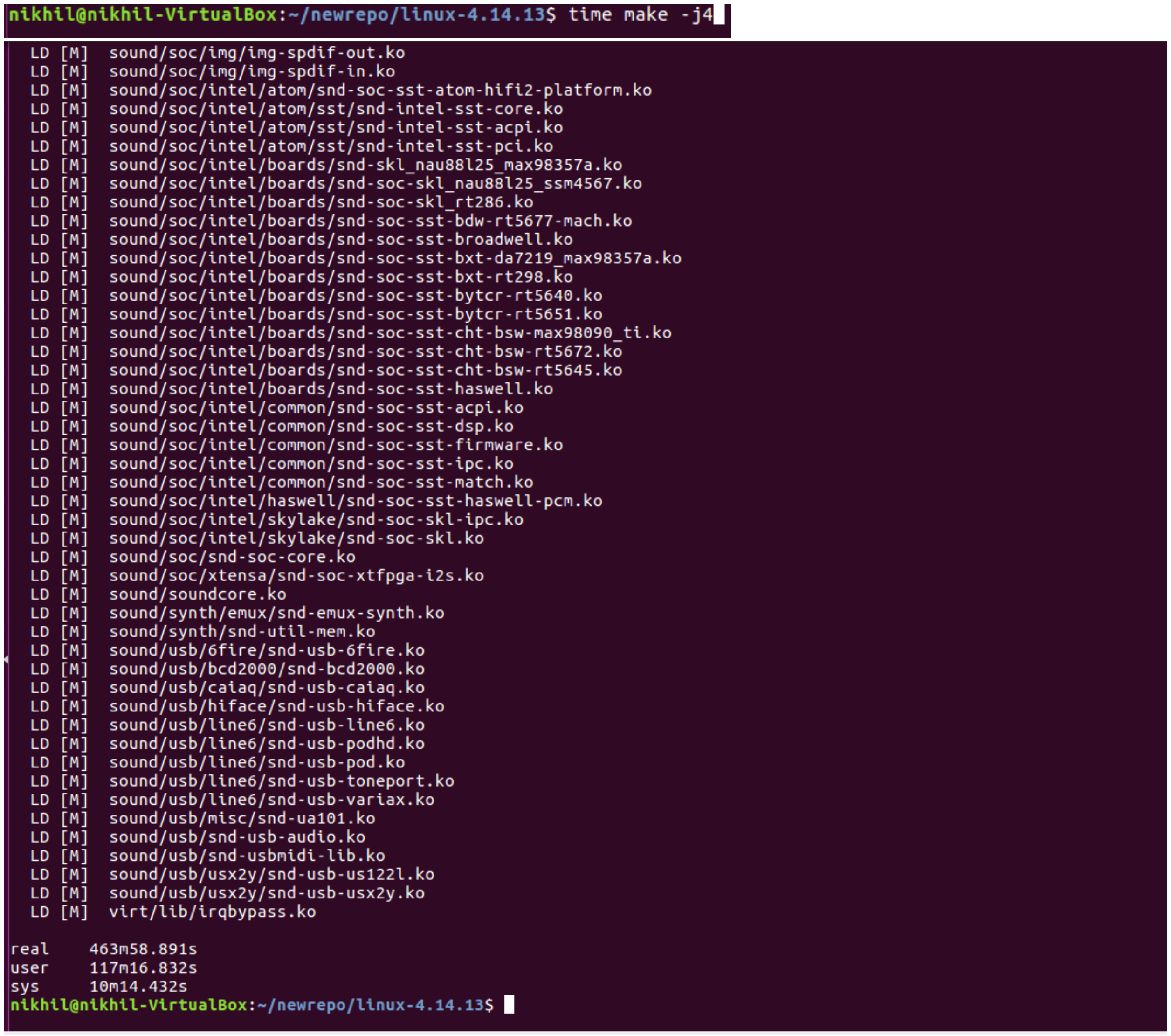


```
 Enter a filename to which this configuration
 should be saved as an alternate.  Leave blank to
 abort.

 .config

          <  Ok  >        < Help >
```



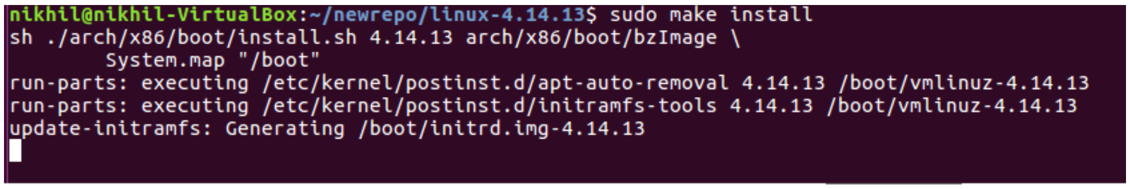
```
configuration written to .config
```

Default build using make -j4

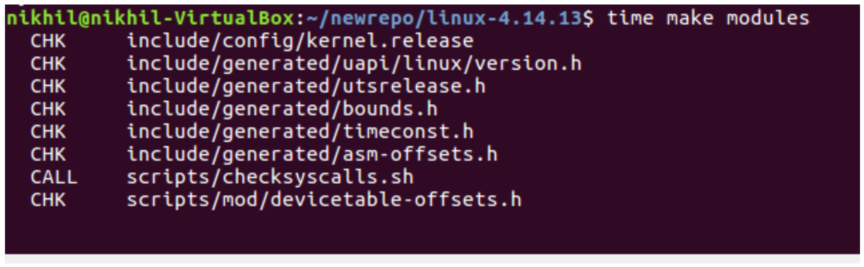


```
nikhil@nikhil-VirtualBox:~/newrepo/linux-4.14.13$ time make -j4
  LD [M]  sound/soc/img/img-spdif-out.ko
  LD [M]  sound/soc/img/img-spdif-in.ko
  LD [M]  sound/soc/intel/atom/snd-soc-sst-atom-hifi2-platform.ko
  LD [M]  sound/soc/intel/atom/sst/snd-intel-sst-core.ko
  LD [M]  sound/soc/intel/atom/sst/snd-intel-sst-acpi.ko
  LD [M]  sound/soc/intel/atom/sst/snd-intel-sst-pci.ko
  LD [M]  sound/soc/intel/boards/snd-skl_nau88l25_max98357a.ko
  LD [M]  sound/soc/intel/boards/snd-soc-skl_nau88l25_ssm4567.ko
  LD [M]  sound/soc/intel/boards/snd-soc-skl_rt286.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-bdw-rt5677-mach.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-broadwell.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-bxt-da7219_max98357a.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-bxt-rt298.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-bytcr-rt5640.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-bytcr-rt5651.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-cht-bsw-max98090_ti.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-cht-bsw-rt5672.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-cht-bsw-rt5645.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-haswell.ko
  LD [M]  sound/soc/intel/common/snd-soc-sst-acpi.ko
  LD [M]  sound/soc/intel/common/snd-soc-sst-dsp.ko
  LD [M]  sound/soc/intel/common/snd-soc-sst-firmware.ko
  LD [M]  sound/soc/intel/common/snd-soc-sst-ipc.ko
  LD [M]  sound/soc/intel/common/snd-soc-sst-match.ko
  LD [M]  sound/soc/intel/haswell/snd-soc-sst-haswell-pcm.ko
  LD [M]  sound/soc/intel/skylake/snd-soc-skl-ipc.ko
  LD [M]  sound/soc/intel/skylake/snd-soc-skl.ko
  LD [M]  sound/soc/snd-soc-core.ko
  LD [M]  sound/soc/xtensa/snd-soc-xtfpga-i2s.ko
  LD [M]  sound/soundcore.ko
  LD [M]  sound/synth/emux/snd-emux-synth.ko
  LD [M]  sound/synth/snd-util-mem.ko
  LD [M]  sound/usb/6fire/snd-usb-6fire.ko
  LD [M]  sound/usb/bcd2000/snd-bcd2000.ko
  LD [M]  sound/usb/caiaq/snd-usb-caiaq.ko
  LD [M]  sound/usb/hiface/snd-usb-hiface.ko
  LD [M]  sound/usb/line6/snd-usb-line6.ko
  LD [M]  sound/usb/line6/snd-usb-podhd.ko
  LD [M]  sound/usb/line6/snd-usb-pod.ko
  LD [M]  sound/usb/line6/snd-usb-toneport.ko
  LD [M]  sound/usb/line6/snd-usb-variax.ko
  LD [M]  sound/usb/misc/snd-ua101.ko
  LD [M]  sound/usb/snd-usb-audio.ko
  LD [M]  sound/usb/snd-usbmidi-lib.ko
  LD [M]  sound/usb/usx2y/snd-usb-us122l.ko
  LD [M]  sound/usb/usx2y/snd-usb-usx2y.ko
  LD [M]  virt/lib/irqbypass.ko

real    463m58.891s
user    117m16.832s
sys     10m14.432s
nikhil@nikhil-VirtualBox:~/newrepo/linux-4.14.13$
```

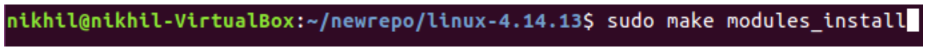
sudo make install


```
nikhil@nikhil-VirtualBox:~/newrepo/linux-4.14.13$ sudo make install
sh ./arch/x86/boot/install.sh 4.14.13 arch/x86/boot/bzImage \
        System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 4.14.13 /boot/vmlinuz-4.14.13
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 4.14.13 /boot/vmlinuz-4.14.13
update-initramfs: Generating /boot/initrd.img-4.14.13
```

sudo make modules

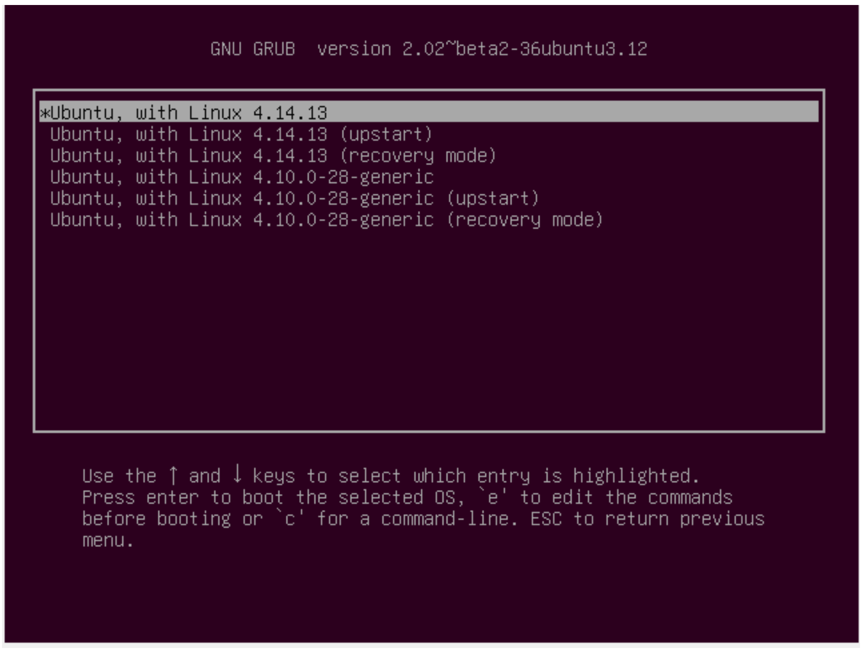

```
nikhil@nikhil-VirtualBox:~/newrepo/linux-4.14.13$ time make modules
  CHK     include/config/kernel.release
  CHK     include/generated/uapi/linux/version.h
  CHK     include/generated/utsrelease.h
  CHK     include/generated/bounds.h
  CHK     include/generated/timeconst.h
  CHK     include/generated/asm-offsets.h
  CALL    scripts/checksyscalls.sh
  CHK     scripts/mod/devicetable-offsets.h
```

sudo make modules_install


```
nikhil@nikhil-VirtualBox:~/newrepo/linux-4.14.13$ sudo make modules_install
```

Reboot your VM after you have compiled your kernel, and switch which kernel the GRUB menu allows you to start given your compiled kernel. **Obtain a screenshot showing your newly installed kernel using some of the commands from class to prove it booted correctly (see problem 6).**

This is the grub menu after installing and restarting the VM. This shows the original linux and newly installed linux

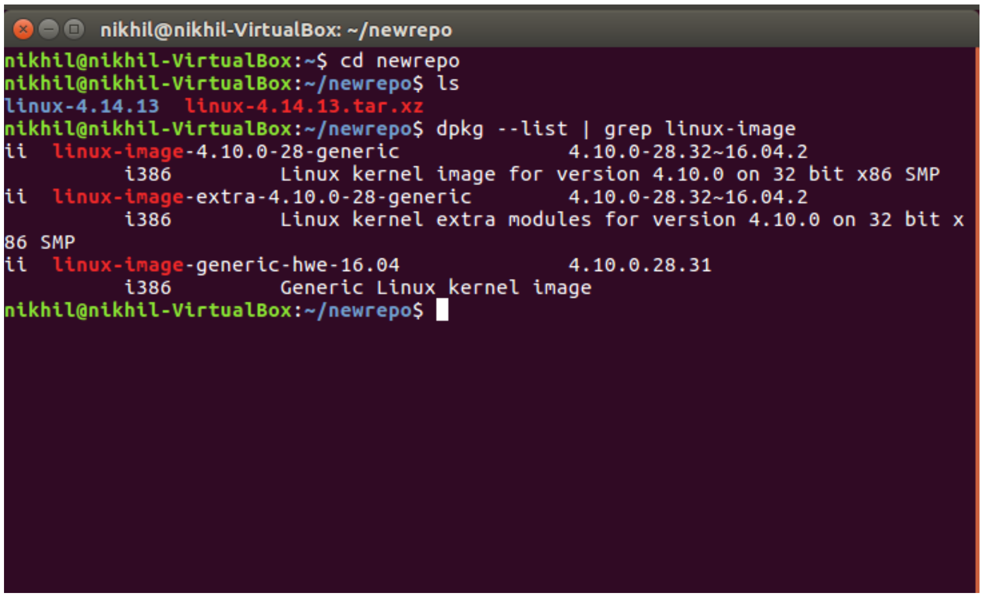

```
GNU GRUB  version 2.02~beta2-36ubuntu3.12

*Ubuntu, with Linux 4.14.13
 Ubuntu, with Linux 4.14.13 (upstart)
 Ubuntu, with Linux 4.14.13 (recovery mode)
 Ubuntu, with Linux 4.10.0-28-generic
 Ubuntu, with Linux 4.10.0-28-generic (upstart)
 Ubuntu, with Linux 4.10.0-28-generic (recovery mode)

   Use the ↑ and ↓ keys to select which entry is highlighted.
   Press enter to boot the selected OS, `e' to edit the commands
   before booting or `c' for a command-line. ESC to return previous
   menu.
```

View the currently installed kernels with this command. **Report this output in your assignment.**

$ dpkg --list | grep linux-image

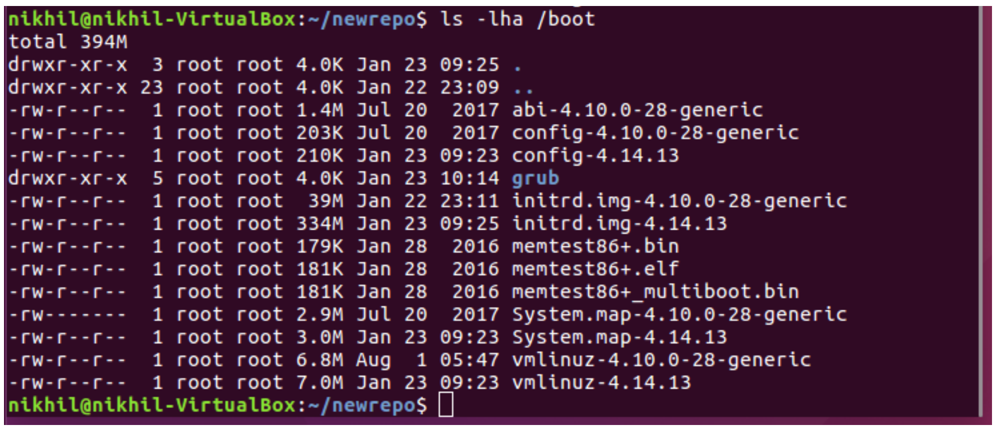
$ ls -lha /boot

This is the dpkg command which doesn't show the linux image which is installed by user



```
nikhil@nikhil-VirtualBox:~$ cd newrepo
nikhil@nikhil-VirtualBox:~/newrepo$ ls
linux-4.14.13  linux-4.14.13.tar.xz
nikhil@nikhil-VirtualBox:~/newrepo$ dpkg --list | grep linux-image
ii  linux-image-4.10.0-28-generic              4.10.0-28.32~16.04.2
         i386         Linux kernel image for version 4.10.0 on 32 bit x86 SMP
ii  linux-image-extra-4.10.0-28-generic        4.10.0-28.32~16.04.2
         i386         Linux kernel extra modules for version 4.10.0 on 32 bit x
86 SMP
ii  linux-image-generic-hwe-16.04              4.10.0.28.31
         i386         Generic Linux kernel image
nikhil@nikhil-VirtualBox:~/newrepo$ 
```

However ls -lha /boot shows all the linux images and files



```
nikhil@nikhil-VirtualBox:~/newrepo$ ls -lha /boot
total 394M
drwxr-xr-x  3 root root 4.0K Jan 23 09:25 .
drwxr-xr-x 23 root root 4.0K Jan 22 23:09 ..
-rw-r--r--  1 root root 1.4M Jul 20  2017 abi-4.10.0-28-generic
-rw-r--r--  1 root root 203K Jul 20  2017 config-4.10.0-28-generic
-rw-r--r--  1 root root 210K Jan 23 09:23 config-4.14.13
drwxr-xr-x  5 root root 4.0K Jan 23 10:14 grub
-rw-r--r--  1 root root  39M Jan 22 23:11 initrd.img-4.10.0-28-generic
-rw-r--r--  1 root root 334M Jan 23 09:25 initrd.img-4.14.13
-rw-r--r--  1 root root 179K Jan 28  2016 memtest86+.bin
-rw-r--r--  1 root root 181K Jan 28  2016 memtest86+.elf
-rw-r--r--  1 root root 181K Jan 28  2016 memtest86+_multiboot.bin
-rw-------  1 root root 2.9M Jul 20  2017 System.map-4.10.0-28-generic
-rw-r--r--  1 root root 3.0M Jan 23 09:23 System.map-4.14.13
-rw-r--r--  1 root root 6.8M Aug  1 05:47 vmlinuz-4.10.0-28-generic
-rw-r--r--  1 root root 7.0M Jan 23 09:23 vmlinuz-4.14.13
nikhil@nikhil-VirtualBox:~/newrepo$ 
```

Find the apt-get command that will allow you to purge old kernels. Do not delete any (yet). **Report the command you used in the homework**.

The command to purge old kernel is: sudo apt-get purge <old_kernel_name>

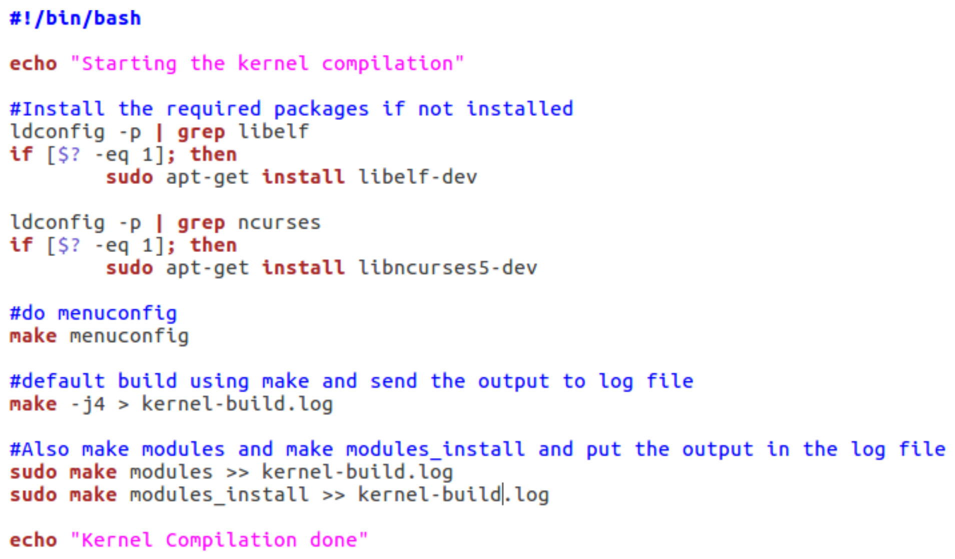
[Problem 8 - 10 Pts] Create a linux kernel build script

Using bash, write a simple script that can compile the kernel with a single command, **build_kernel** (or similar). Add this script to your git repository in a folder called utils or bin. **Include the script in your .pdf on dropbox.** Add your utils or bin folder to your *path* environment variable. Do this by modifying your

.bashrc or .bash_profile script to append a path to your repos binutils path.**Include your modification path command(s) in your writeup.** Grab an example out of this command while writing the build output to a file called **kernel-build.log**(or similar).

Answer:

So I wrote a simple script that will compile the kernel with one command. The script is as given below:



```
#!/bin/bash

echo "Starting the kernel compilation"

#Install the required packages if not installed
ldconfig -p | grep libelf
if [$? -eq 1]; then
        sudo apt-get install libelf-dev

ldconfig -p | grep ncurses
if [$? -eq 1]; then
        sudo apt-get install libncurses5-dev

#do menuconfig
make menuconfig

#default build using make and send the output to log file
make -j4 > kernel-build.log

#Also make modules and make modules_install and put the output in the log file
sudo make modules >> kernel-build.log
sudo make modules_install >> kernel-build.log

echo "Kernel Compilation done"
```

This script is named as build_kernel.sh

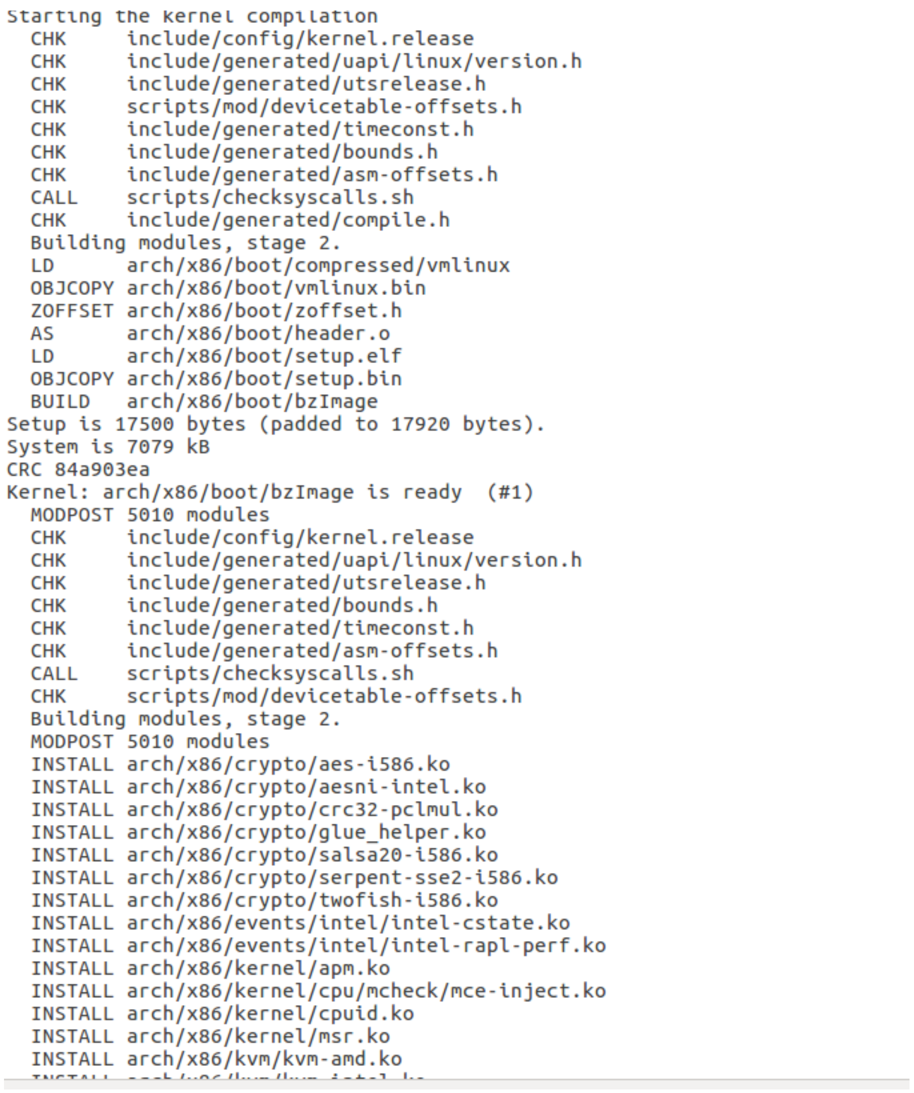
Then I updated the path variable with the address where the script is located. This allowed me to run the script with just the command.

I modified the path using command using the following command:

export PATH="$PATH:~/newrepo/linux-4.14.13"

So in the script I also outputted the data during the build to the file called kernel-build.log which I have attached in the D2L.

Attached is the screenshot of the log file



```
Starting the kernel compilation
  CHK     include/config/kernel.release
  CHK     include/generated/uapi/linux/version.h
  CHK     include/generated/utsrelease.h
  CHK     scripts/mod/devicetable-offsets.h
  CHK     include/generated/timeconst.h
  CHK     include/generated/bounds.h
  CHK     include/generated/asm-offsets.h
  CALL    scripts/checksyscalls.sh
  CHK     include/generated/compile.h
  Building modules, stage 2.
  LD      arch/x86/boot/compressed/vmlinux
  OBJCOPY arch/x86/boot/vmlinux.bin
  ZOFFSET arch/x86/boot/zoffset.h
  AS      arch/x86/boot/header.o
  LD      arch/x86/boot/setup.elf
  OBJCOPY arch/x86/boot/setup.bin
  BUILD   arch/x86/boot/bzImage
Setup is 17500 bytes (padded to 17920 bytes).
System is 7079 kB
CRC 84a903ea
Kernel: arch/x86/boot/bzImage is ready  (#1)
  MODPOST 5010 modules
  CHK     include/config/kernel.release
  CHK     include/generated/uapi/linux/version.h
  CHK     include/generated/utsrelease.h
  CHK     include/generated/bounds.h
  CHK     include/generated/timeconst.h
  CHK     include/generated/asm-offsets.h
  CALL    scripts/checksyscalls.sh
  CHK     scripts/mod/devicetable-offsets.h
  Building modules, stage 2.
  MODPOST 5010 modules
  INSTALL arch/x86/crypto/aes-i586.ko
  INSTALL arch/x86/crypto/aesni-intel.ko
  INSTALL arch/x86/crypto/crc32-pclmul.ko
  INSTALL arch/x86/crypto/glue_helper.ko
  INSTALL arch/x86/crypto/salsa20-i586.ko
  INSTALL arch/x86/crypto/serpent-sse2-i586.ko
  INSTALL arch/x86/crypto/twofish-i586.ko
  INSTALL arch/x86/events/intel/intel-cstate.ko
  INSTALL arch/x86/events/intel/intel-rapl-perf.ko
  INSTALL arch/x86/kernel/apm.ko
  INSTALL arch/x86/kernel/cpu/mcheck/mce-inject.ko
  INSTALL arch/x86/kernel/cpuid.ko
  INSTALL arch/x86/kernel/msr.ko
  INSTALL arch/x86/kvm/kvm-amd.ko
```

[Problem 9] Extra For Fun - Automate your environment and home directory

You will be working on many different environments. There are many ways you can do this. One is using vagrant (as mentioned in class) and the second is creating a repository to track your environment.

Option 1) Vagrant is a tool that helps you build and manage your virtual machines. Meaning you can you should be able to quickly recreate the environment of any VM you have created before very quickly and as many times as you want.

Option 2) Create a repo that will help automate your environment setup, install packages, create links to your .bashrc, .bash_profile, .vimrc, etc, so that you can easily re-set up any machine when you configure it.