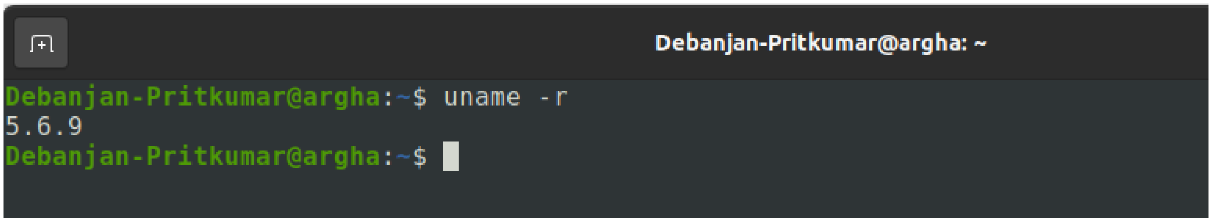
Advances in Operating Systems Design

Assignment 1 : (A) Configuring Linux Kernel

Debanjan Saha *19CS30014*
Pritkumar Godhani *19CS10048*

PART A

All the below experiments are done on the kernel [`Linux x86 v5.6.9`], with the `Ubuntu v20.04 (Focal)` Linux distribution.

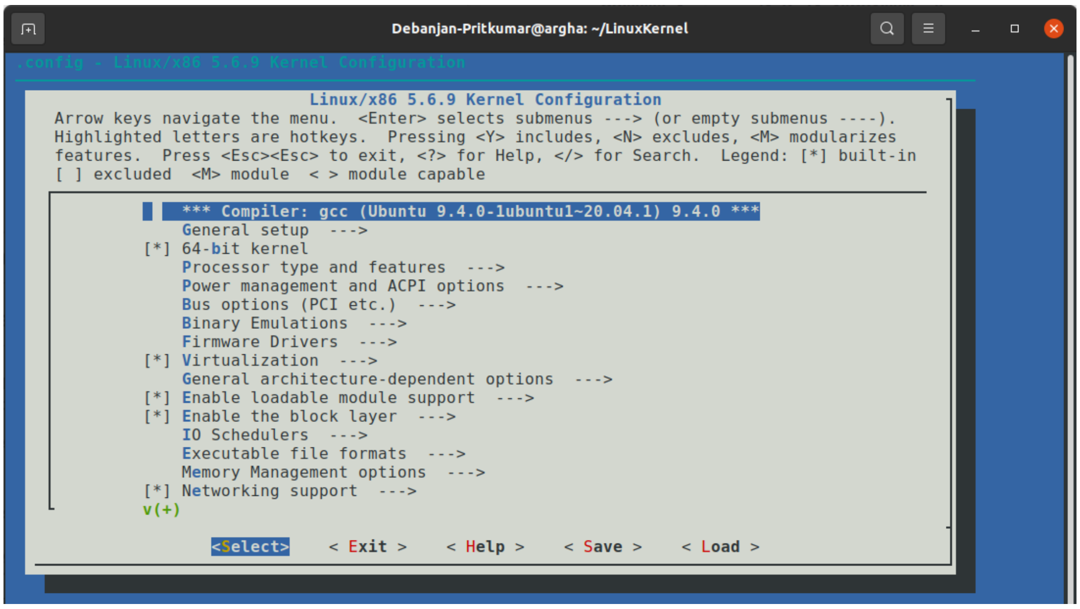


```
Debanjan-Pritkumar@argha:~$ uname -r
5.6.9
Debanjan-Pritkumar@argha:~$ 
```

Figure 1.0

The `Remmina` Remote Desktop Client (application on Ubuntu v20.04) was used for remote desktop logins.

In order to configure the kernel options, after navigating into the kernel root folder, `$ make menuconfig` command was executed that opened a TUI(Terminal User Interface) shown in *Figure 1.1* below.

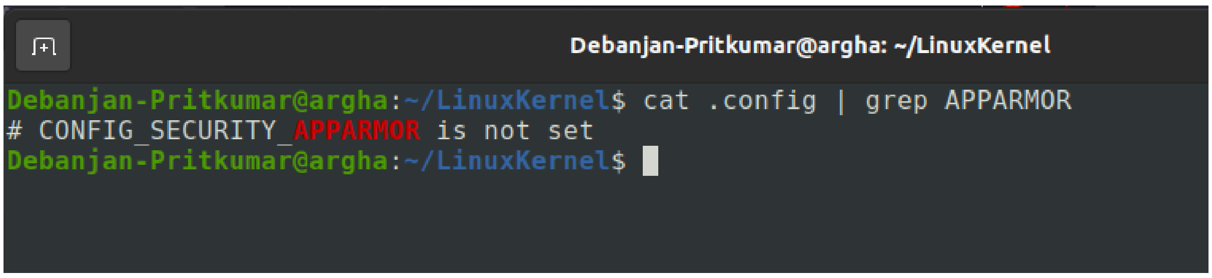


```
.config - Linux/x86 5.6.9 Kernel Configuration
                     Linux/x86 5.6.9 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
  features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in
  [ ] excluded  <M> module  < > module capable

             *** Compiler: gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0 ***
                 General setup  --->
             [*] 64-bit kernel
                 Processor type and features  --->
                 Power management and ACPI options  --->
                 Bus options (PCI etc.)  --->
                 Binary Emulations  --->
                 Firmware Drivers  --->
             [*] Virtualization  --->
                 General architecture-dependent options  --->
             [*] Enable loadable module support  --->
             [*] Enable the block layer  --->
                 IO Schedulers  --->
                 Executable file formats  --->
                 Memory Management options  --->
             [*] Networking support  --->
             v(+)

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

Figure 1.1

Each of the below subsections detail the exact location of the options that we enabled/disabled/re-configured and the checks performed on the system.
Note that all the below options were applied altogether (not individually). Once the `.config` file was saved, the kernel v5.6.9 was recompiled & reinstalled, and the system was rebooted so as to allow all the changes to be properly applied system-wide.

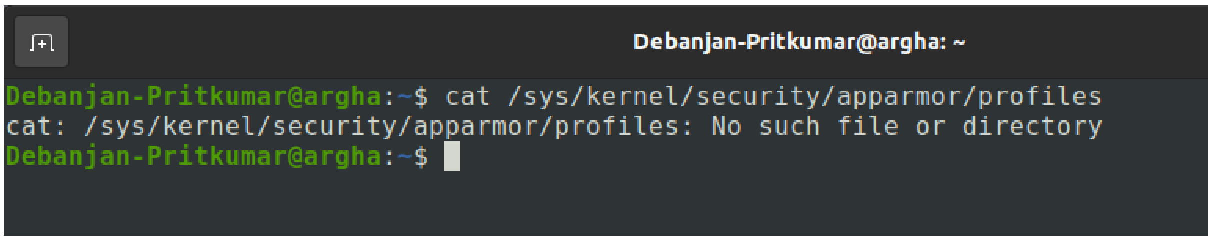
1. **Excluding AppArmor Support**
   The option to add/exclude AppArmor support was located in the below locale in the kernel configuration TUI :
   ```
   Security Options --->
             [ ] AppArmor support
   ```
   By default, AppArmor was enabled in the kernel configuration. The option was unselected.
   After the kernel rebuild, we `grep`'ed the `.config` file to check if the option was actually set or not. *Figure 1.2* shows the output of the check.



```
Debanjan-Pritkumar@argha: ~/LinuxKernel
Debanjan-Pritkumar@argha:~/LinuxKernel$ cat .config | grep APPARMOR
# CONFIG_SECURITY_APPARMOR is not set
Debanjan-Pritkumar@argha:~/LinuxKernel$
```

Figure 1.2

We also checked the AppArmor status by inspecting the profiles file as suggested in the Assignment description, which also returned "not found". See *Figure 1.3*.



```
Debanjan-Pritkumar@argha: ~
Debanjan-Pritkumar@argha:~$ cat /sys/kernel/security/apparmor/profiles
cat: /sys/kernel/security/apparmor/profiles: No such file or directory
Debanjan-Pritkumar@argha:~$
```

Figure 1.3

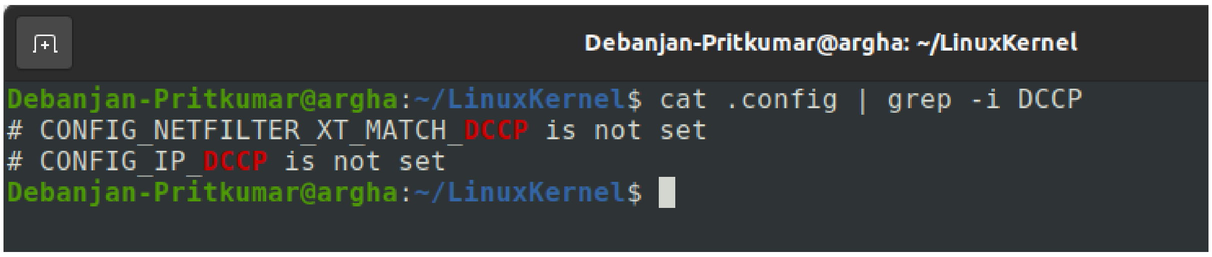
Both the above checks confirmed that AppArmor was excluded from our kernel build.

2. **Excluding DCCP Protocol**
   The option for enabling/disabling the DCCP Protocol was located in the below locale in the kernel configuration TUI :

```
[*] Networking Support --->
          Networking Options --->
               < > The DCCP Protocol
                    --- The DCCP Protocol
```

   However unlike AppArmor, the DCCP Protocol was by default unselected; so we did not make any changes.

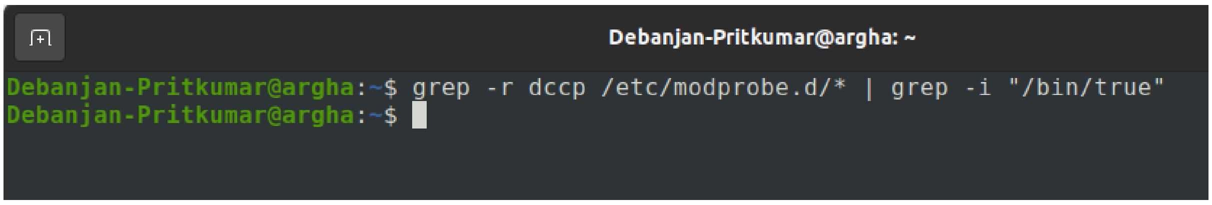
   Similar to above, we first check the `.config` file to confirm if the DCCP option is not set.



```
Debanjan-Pritkumar@argha: ~/LinuxKernel
Debanjan-Pritkumar@argha:~/LinuxKernel$ cat .config | grep -i DCCP
# CONFIG_NETFILTER_XT_MATCH_DCCP is not set
# CONFIG_IP_DCCP is not set
Debanjan-Pritkumar@argha:~/LinuxKernel$
```

Figure 1.4

   Then we use the check using `modprobe` that probes whether modules concerning DCCP are enabled or not.



```
Debanjan-Pritkumar@argha: ~
Debanjan-Pritkumar@argha:~$ grep -r dccp /etc/modprobe.d/* | grep -i "/bin/true"
Debanjan-Pritkumar@argha:~$
```

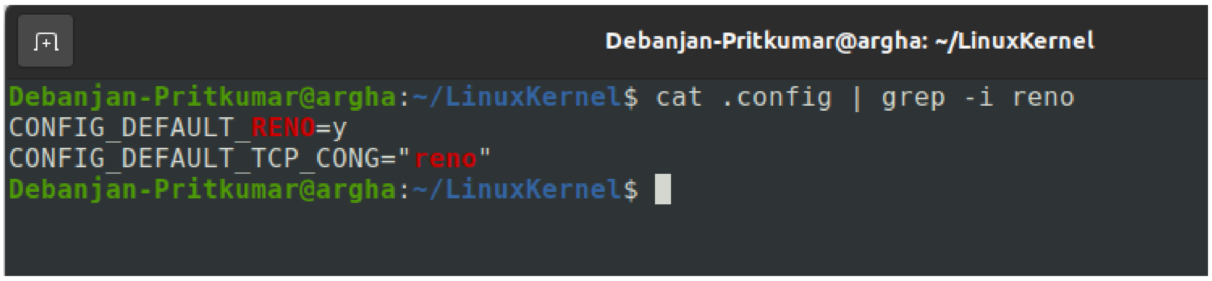
Figure 1.5

   This shows that the DCCP protocol is disbaled from the kernel built on the system concerned.

3. **Changing the Default TCP Congestion Control Algorithm to RENO**
   The option for changing the default TCP Congestion Control Algorithm was located in the below locale in the kernel configuration TUI :

```
[*] Networking Support --->
        Networking Options --->
           [*] TCP: advanced congestion control --->
                Default TCP congestion control (Reno)
```

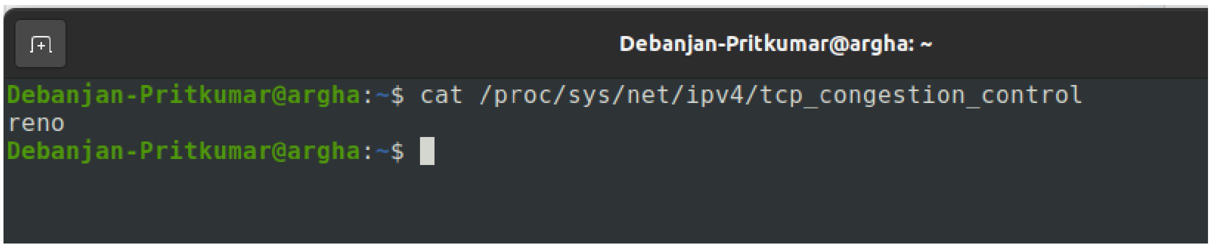
   The default congestion control algorithm was set to `Cubic` prior to reconfiguration. That was changed to `Reno`.



```
Debanjan-Pritkumar@argha:~/LinuxKernel$ cat .config | grep -i reno
CONFIG_DEFAULT_RENO=y
CONFIG_DEFAULT_TCP_CONG="reno"
Debanjan-Pritkumar@argha:~/LinuxKernel$
```

Figure 1.6

   The following options set in the `.config` file confirm that the default congestion control algorithm has been switched to `Reno`.



```
Debanjan-Pritkumar@argha:~$ cat /proc/sys/net/ipv4/tcp_congestion_control
reno
Debanjan-Pritkumar@argha:~$
```

Figure 1.7

   This confirms and verifies that the default TCP congestion control algorithm has been changed to Reno in the new build of the kernel.