OS Week 1 LAB ASSIGNMENT

1) Process related commands:

a. ps

- 'process status'
- used to list the currently running processes and their pid's along with some information depending on options .
- Syntax - ps [options]

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/160121733092$ ps
    PID TTY          TIME CMD
   2633 pts/0    00:00:00 bash
   2661 pts/0    00:00:00 ps
```

Options:

- ps -T : View all processes associated with this terminal.

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/160121733092$ ps -T
    PID    SPID TTY          TIME CMD
   2633    2633 pts/0    00:00:00 bash
   2828    2828 pts/0    00:00:00 ps
```

- ps -r : View all the running processes.

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/160121733092$ ps -r
    PID TTY      STAT   TIME COMMAND
   2848 pts/0    R+     0:00 ps -r
```

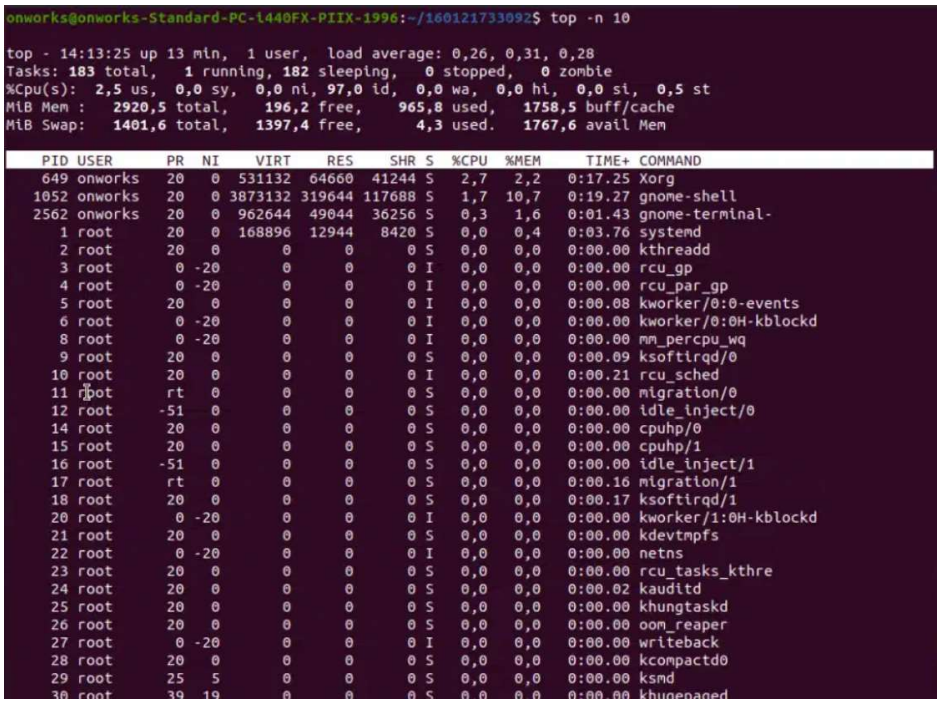
b. top

- It shows all the active linux processes.

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/160121733092$ top
top - 14:10:15 up 10 min,  1 user,  load average: 0,05, 0,43, 0,32
Tasks: 191 total,   2 running, 189 sleeping,   0 stopped,   0 zombie
%Cpu(s):  9,3 us,  1,0 sy,  0,0 ni, 89,5 id,  0,2 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   2920,5 total,    134,8 free,   1026,9 used,   1758,8 buff/cache
MiB Swap:   1401,6 total,   1397,4 free,      4,3 used.   1706,3 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
     27 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 writeback
     28 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kcompactd0
     29 root      25   5       0      0      0 S   0,0   0,0   0:00.00 ksmd
     30 root      39  19       0      0      0 S   0,0   0,0   0:00.00 khugepaged
     77 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kintegrityd
     78 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kblockd
     79 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 blkcg_punt_bio
     80 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 tpm_dev_wq
     81 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 ata_sff
     82 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 md
     83 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 edac-poller
     84 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 devfreq_wq
     85 root      rt   0       0      0      0 S   0,0   0,0   0:00.00 watchdogd
     86 root      20   0       0      0      0 I   0,0   0,0   0:00.05 kworker/u4:1-events_unbound
     87 root      20   0       0      0      0 S   0,0   0,0   0:00.26 kswapd0
     88 root      20   0       0      0      0 S   0,0   0,0   0:00.00 ecryptfs-kthrea
     90 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kthrotld
     91 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 acpi_thermal_pm
     92 root      20   0       0      0      0 S   0,0   0,0   0:00.00 scsi_eh_0
     93 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 scsi_tmf_0
     94 root      20   0       0      0      0 S   0,0   0,0   0:00.00 scsi_eh_1
     95 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 scsi_tmf_1
     96 root      20   0       0      0      0 I   0,0   0,0   0:00.31 kworker/u4:2-events_power_effic+
     97 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 vfio-irqfd-clea
     98 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 ipv6_addrconf
    108 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kstrp
    111 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kworker/u5:0
    124 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 charger_manager
    125 root       0 -20       0      0      0 I   0,0   0,0   0:01.08 kworker/1:1H-kblockd
    168 root      20   0       0      0      0 I   0,0   0,0   0:00.06 kworker/0:2-events
    171 root      20   0       0      0      0 I   0,0   0,0   0:00.06 kworker/0:3-events
```

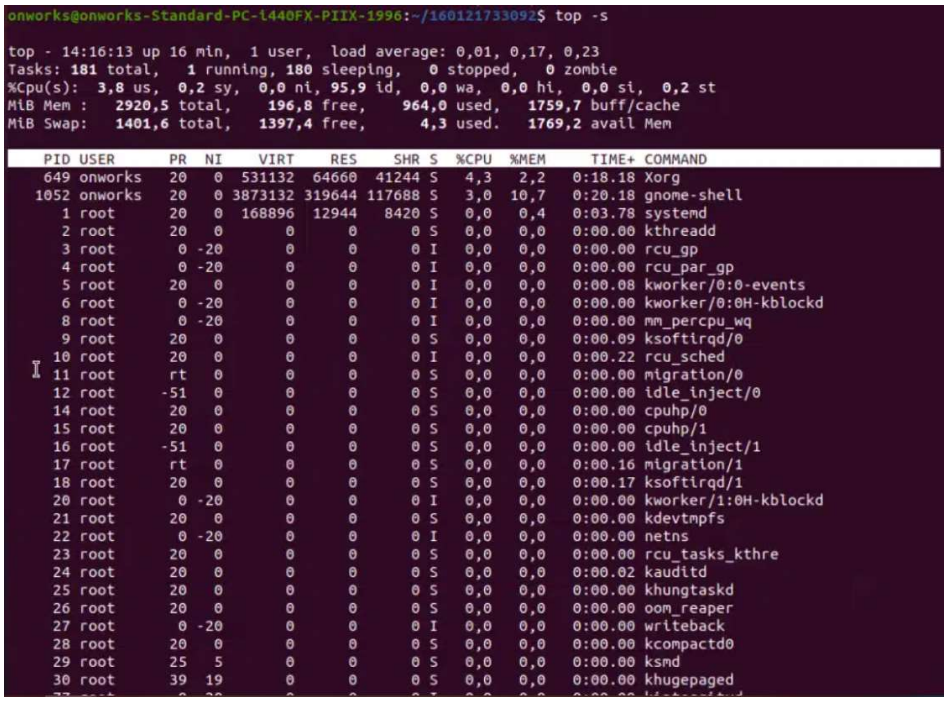
Options:

- top -n 10 : it will automatically exit after 10 number of repetitions.



```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/160121733092$ top -n 10

top - 14:13:25 up 13 min,  1 user,  load average: 0,26, 0,31, 0,28
Tasks: 183 total,   1 running, 182 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2,5 us,  0,0 sy,  0,0 ni, 97,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,5 st
MiB Mem :   2920,5 total,    196,2 free,    965,8 used,   1758,5 buff/cache
MiB Swap:   1401,6 total,   1397,4 free,      4,3 used.   1767,6 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    649 onworks   20   0  531132  64660  41244 S   2,7   2,2   0:17.25 Xorg
   1052 onworks   20   0 3873132 319644 117688 S   1,7  10,7   0:19.27 gnome-shell
   2562 onworks   20   0  962644  49044  36256 S   0,3   1,6   0:01.43 gnome-terminal-
      1 root      20   0  168896  12944   8420 S   0,0   0,4   0:03.76 systemd
      2 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kthreadd
      3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_par_gp
      5 root      20   0       0      0      0 I   0,0   0,0   0:00.08 kworker/0:0-events
      6 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kworker/0:0H-kblockd
      8 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 mm_percpu_wq
      9 root      20   0       0      0      0 S   0,0   0,0   0:00.09 ksoftirqd/0
     10 root      20   0       0      0      0 I   0,0   0,0   0:00.21 rcu_sched
     11 root      rt   0       0      0      0 S   0,0   0,0   0:00.00 migration/0
     12 root     -51   0       0      0      0 S   0,0   0,0   0:00.00 idle_inject/0
     14 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/0
     15 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/1
     16 root     -51   0       0      0      0 S   0,0   0,0   0:00.00 idle_inject/1
     17 root      rt   0       0      0      0 S   0,0   0,0   0:00.16 migration/1
     18 root      20   0       0      0      0 S   0,0   0,0   0:00.17 ksoftirqd/1
     20 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kworker/1:0H-kblockd
     21 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kdevtmpfs
     22 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 netns
     23 root      20   0       0      0      0 S   0,0   0,0   0:00.00 rcu_tasks_kthre
     24 root      20   0       0      0      0 S   0,0   0,0   0:00.02 kauditd
     25 root      20   0       0      0      0 S   0,0   0,0   0:00.00 khungtaskd
     26 root      20   0       0      0      0 S   0,0   0,0   0:00.00 oom_reaper
     27 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 writeback
     28 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kcompactd0
     29 root      25   5       0      0      0 S   0,0   0,0   0:00.00 ksmd
     30 root      39  19       0      0      0 S   0,0   0,0   0:00.00 khugepaged
```

- top -s : use top in secure mode.



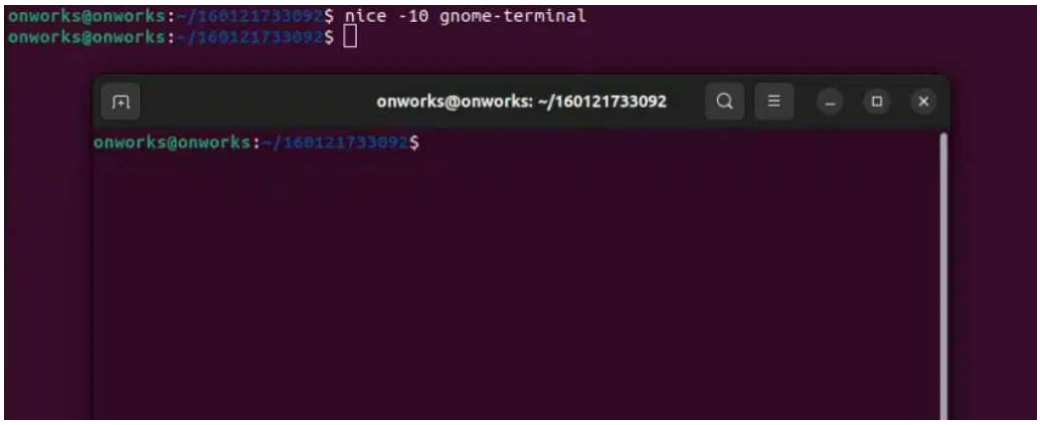
```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/160121733092$ top -s

top - 14:16:13 up 16 min,  1 user,  load average: 0,01, 0,17, 0,23
Tasks: 181 total,   1 running, 180 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3,8 us,  0,2 sy,  0,0 ni, 95,9 id,  0,0 wa,  0,0 hi,  0,0 si,  0,2 st
MiB Mem :   2920,5 total,    196,8 free,    964,0 used,   1759,7 buff/cache
MiB Swap:   1401,6 total,   1397,4 free,      4,3 used.   1769,2 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    649 onworks   20   0  531132  64660  41244 S   4,3   2,2   0:18.18 Xorg
   1052 onworks   20   0 3873132 319644 117688 S   3,0  10,7   0:20.18 gnome-shell
      1 root      20   0  168896  12944   8420 S   0,0   0,4   0:03.78 systemd
      2 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kthreadd
      3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_par_gp
      5 root      20   0       0      0      0 I   0,0   0,0   0:00.08 kworker/0:0-events
      6 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kworker/0:0H-kblockd
      8 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 mm_percpu_wq
      9 root      20   0       0      0      0 S   0,0   0,0   0:00.09 ksoftirqd/0
     10 root      20   0       0      0      0 I   0,0   0,0   0:00.22 rcu_sched
     11 root      rt   0       0      0      0 S   0,0   0,0   0:00.00 migration/0
     12 root     -51   0       0      0      0 S   0,0   0,0   0:00.00 idle_inject/0
     14 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/0
     15 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/1
     16 root     -51   0       0      0      0 S   0,0   0,0   0:00.00 idle_inject/1
     17 root      rt   0       0      0      0 S   0,0   0,0   0:00.16 migration/1
     18 root      20   0       0      0      0 S   0,0   0,0   0:00.17 ksoftirqd/1
     20 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kworker/1:0H-kblockd
     21 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kdevtmpfs
     22 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 netns
     23 root      20   0       0      0      0 S   0,0   0,0   0:00.00 rcu_tasks_kthre
     24 root      20   0       0      0      0 S   0,0   0,0   0:00.02 kauditd
     25 root      20   0       0      0      0 S   0,0   0,0   0:00.00 khungtaskd
     26 root      20   0       0      0      0 S   0,0   0,0   0:00.00 oom_reaper
     27 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 writeback
     28 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kcompactd0
     29 root      25   5       0      0      0 S   0,0   0,0   0:00.00 ksmd
     30 root      39  19       0      0      0 S   0,0   0,0   0:00.00 khugepaged
```

c. nice

- allows users to prioritise process execution before starting the process.

Options:

- nice -10 gnome-terminal : To set the priority of a process.



- nice --10 gnome-terminal : To set the negative priority of a process.

d. renice

- alters the nice value of one or more running process.

e. kill

- used to terminate the process manually. It sends signals to the process to terminate it.

Options:

- kill -l : to display all available signals.

- kill pid : specify the pid to kill it.

```
24867 pts/17   S+     0:00 ./hello
24870 ?        S      0:00 sleep 5
24871 pts/1    R+     0:00 ps -ax
onworks@onworks-Standard-PC-i440FX-PIIX-1996:/proc$ kill 24867
```

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/160121733092$ ./hello
Enter a number:
Terminated
```

2) Network related commands:

a. nslookup

- name server lookup
- command for getting information from DNS server.
- Syntax : nslookup example.com/[IP address]

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/160121733092$ nslookup www.google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   www.google.com
Address: 142.250.185.196
Name:   www.google.com
Address: 2a00:1450:4001:808::2004
```

b. traceroute:

- It prints the route that packet takes to the host.

```
onworks@onworks:~/160121733092$ traceroute www.google.com
traceroute to www.google.com (172.217.16.132), 30 hops max, 60 byte packets
 1  _gateway (10.0.2.2)  0.722 ms !N  0.623 ms !N  0.593 ms !N
```

Options:

- -4 : Use IP version 4 only IPv4.

```
onworks@onworks:~/160121733092$ traceroute -4  www.google.com
traceroute to www.google.com (142.250.186.132), 30 hops max, 60 byte packets
 1  _gateway (10.0.2.2)  0.739 ms !N  0.627 ms !N  0.597 ms !N
```

- -n : Stop the resolving of the IP addresses.

```
onworks@onworks:~/160121733092$ traceroute -n www.google.com
traceroute to www.google.com (172.217.18.100), 30 hops max, 60 byte packets
 1  10.0.2.2  0.621 ms !N  0.532 ms !N  0.506 ms !N
```

- -m : to set the maximum number of hops for the packet to reach the destination.

```
onworks@onworks:~/160121733092$ traceroute -m 5 www.google.com
traceroute to www.google.com (172.217.16.132), 5 hops max, 60 byte packets
 1  _gateway (10.0.2.2)  7.076 ms !N  7.011 ms !N  6.969 ms !N
```

c. **Ifconfig:**

- Interface Configurator
- It is used to configure the kernel-resident network interfaces. It is used at the boot time to set up the interfaces as necessary.

```
onworks@onworks:~/1601217330925 ifconfig
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::196f:c283:e126:fd3  prefixlen 64  scopeid 0x20<link>
        inet6 fec0::aea2:1610:487f:5959  prefixlen 64  scopeid 0x40<site>
        inet6 fec0::dca6:b233:1159:a95d  prefixlen 64  scopeid 0x40<site>
        ether 52:54:00:12:34:56  txqueuelen 1000  (Ethernet)
        RX packets 38322  bytes 53093756 (53.0 MB)
        RX errors 306  dropped 0  overruns 0  frame 306
        TX packets 8210  bytes 587672 (587.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 278  bytes 27073 (27.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 278  bytes 27073 (27.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Options :

- -s : it is used to display short list instead of details.

```
onworks@onworks:~/1601217330925 ifconfig -s
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
ens3      1500    38349    306      0 0          8237      0      0      0 BMRU
lo       65536      278      0      0 0           278      0      0      0 LRU
```

3) File related commands:

a. **mkdir**

- to create directories
- syntax - mkdir [options…] [directory_name..]

```
onworks@onworks:~/1601217330925 mkdir newdir
onworks@onworks:~/1601217330925 ls
newdir
onworks@onworks:~/1601217330925 ls -lrt
total 4
drwxrwxr-x 2 onworks onworks 4096 Sep 18 19:35 newdir
```

b. **cat**

- displays the content of a file onto the terminal
- syntax - cat file_name

```
onworks@onworks:~/1601217330925 cat mytxt.txt mytxt2.txt
This is a sample text file
A Jaya Adithya
```

c. **rm**
- used to remove files,directories etc
- syntax - rm file_name

```
onworks@onworks:~/1601217330925$ rm mytxt.txt
onworks@onworks:~/1601217330925$ cat mytxt.txt
cat: mytxt.txt: No such file or directory
```

d. **rmdir**
- to remove the directory
- syntax - rm file_name

```
onworks@onworks:~/1601217330925$ rmdir newdir
onworks@onworks:~/1601217330925$ ls -lrt
total 4
-rw-rw-r-- 1 onworks onworks 16 Sep 18 19:39 mytxt2.txt
```

e. **mv**
- command is used to rename file directories and move files from one location to another within a file system.
- For renaming - mv [source_file_name(s)] [Destination_file_name]

```
onworks@onworks:~/1601217330925$ ls
mytxt2.txt
onworks@onworks:~/1601217330925$ mv mytxt2.txt text1.txt
onworks@onworks:~/1601217330925$ ls
text1.txt
```

- For moving a file or directory - mv [source_file_name(s)] [Destinati on_path]

```
onworks@onworks:~/1601217330925$ rmdir newdir
onworks@onworks:~/1601217330925$ mkdir newdir
onworks@onworks:~/1601217330925$ mv text1.txt newdir
onworks@onworks:~/1601217330925$ cd newdir
onworks@onworks:~/1601217330925/newdir$ ls
text1.txt
```

f. **chmod** :
- It is used to change the access mode of a file.
- Syntax : chmod [options] [mode] [file_name]
- To set the file permissions for owner-read and execute, for group-read,for others-execute.

```
onworks@onworks:~/1601217330925$ ls -lrt
total 4
drwxrwxr-x 2 onworks onworks 4096 Sep 18 19:45 newdir
-rw-rw-r-- 1 onworks onworks    0 Sep 18 19:48 helloworld.c
onworks@onworks:~/1601217330925$ chmod 734 helloworld.c
onworks@onworks:~/1601217330925$ ls -lrt
total 4
drwxrwxr-x 2 onworks onworks 4096 Sep 18 19:45 newdir
-rwx-wxr-- 1 onworks onworks    0 Sep 18 19:48 helloworld.c
```

g. chown :

- The chown command changes the owner of the file or directory specified by the File or Directory parameter to the user specified by the Owner parameter.

Options:

- -c : reports when a file change is made

```
onworks@onworks:~/160121733092$ chown -c  onworks helloworld.c
```

- -v : shows the verbose information of every file processed.

```
onworks@onworks:~/160121733092$ chown -v onworks helloworld.c
ownership of 'helloworld.c' retained as onworks
```

h. link:

- It is used for creating links between files.

```
onworks@onworks:~/160121733092$ ln -s text1.txt new_text.txt
onworks@onworks:~/160121733092$ ls -lrt
total 12
drwxrwxr-x 2 onworks onworks 4096 Sep 18 19:45 newdir
-rwx-wxr-- 1 onworks onworks    0 Sep 18 19:48 helloworld.c
-rw-rw-r-- 1 onworks onworks    4 Sep 18 19:54 text1.txt
-rw-rw-r-- 1 onworks onworks    5 Sep 18 19:54 text2.txt
lrwxrwxrwx 1 onworks onworks    9 Sep 18 19:55 new_text.txt -> text1.txt
```

i. unlink:

- It is commonly used to remove the symbolic links between the files.

```
onworks@onworks:~/160121733092$ ls -lrt
total 16
drwxrwxr-x 2 onworks onworks 4096 Sep 18 19:45 newdir
-rwx-wxr-- 1 onworks onworks    0 Sep 18 19:48 helloworld.c
-rw-rw-r-- 2 onworks onworks    4 Sep 18 19:54 text1.txt
-rw-rw-r-- 2 onworks onworks    4 Sep 18 19:54 new_text2.txt
-rw-rw-r-- 1 onworks onworks    5 Sep 18 19:54 text2.txt
lrwxrwxrwx 1 onworks onworks    9 Sep 18 19:55 new_text.txt -> text1.txt
onworks@onworks:~/160121733092$ unlink new_text.txt
onworks@onworks:~/160121733092$ ls -lrt
total 16
drwxrwxr-x 2 onworks onworks 4096 Sep 18 19:45 newdir
-rwx-wxr-- 1 onworks onworks    0 Sep 18 19:48 helloworld.c
-rw-rw-r-- 2 onworks onworks    4 Sep 18 19:54 text1.txt
-rw-rw-r-- 2 onworks onworks    4 Sep 18 19:54 new_text2.txt
-rw-rw-r-- 1 onworks onworks    5 Sep 18 19:54 text2.txt
```

j. touch

- Used to create, change, modify the timestamps of a file.
- Syntax : touch file_name1, file_name2,file_name3,.....

```
onworks@onworks:~/160121733092$ touch text1.txt text2.txt new_text2.txt
onworks@onworks:~/160121733092$ ls -lrt
total 16
drwxrwxr-x 2 onworks onworks 4096 Sep 18 19:45 newdir
-rwx-wxr-- 1 onworks onworks    0 Sep 18 19:48 helloworld.c
-rw-rw-r-- 1 onworks onworks    5 Sep 18 20:04 text2.txt
-rw-rw-r-- 2 onworks onworks    4 Sep 18 20:04 text1.txt
-rw-rw-r-- 2 onworks onworks    4 Sep 18 20:04 new_text2.txt
```

Options :

- -a : To change or update the last access or modification times of a file.

```
onworks@onworks:~/160121733092$ touch -a helloworld.c
onworks@onworks:~/160121733092$ stat helloworld.c
  File: helloworld.c
  Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 803h/2051d      Inode: 662424      Links: 1
Access: (0734/-rwx-wxr--)  Uid: ( 1000/ onworks)   Gid: ( 1000/ onworks)
Access: 2023-09-18 20:07:43.849247159 +0200
Modify: 2023-09-18 19:48:16.935532849 +0200
Change: 2023-09-18 20:07:43.849247159 +0200
 Birth: 2023-09-18 19:48:16.935532849 +0200
```

4) Disk management:

a. df :

- also known as disk free.
- The df command displays the amount of disk space) available on the filesystem with each file name's argument.
- Syntax : df [options] [file_name]

```
onworks@onworks:~/160121733092$ df helloworld.c
Filesystem     1K-blocks     Used Available Use% Mounted on
/dev/sda3       30267332 13120820  15583684  46% /
```

Options:

- -a : used to display all the file system.

```
onworks@onworks:~/160121733092$ df -a
df: /run/user/1000/doc: Operation not permitted
Filesystem     1K-blocks     Used Available Use% Mounted on
sysfs                  0        0         0    - /sys
proc                   0        0         0    - /proc
udev             1439360        0   1439360   0% /dev
devpts                 0        0         0    - /dev/pts
tmpfs             296172     1604    294568   1% /run
/dev/sda3       30267332 13120820  15583684  46% /
securityfs             0        0         0    - /sys/kernel/security
tmpfs            1480848        0   1480848   0% /dev/shm
tmpfs               5120        4      5116   1% /run/lock
cgroup2                0        0         0    - /sys/fs/cgroup
pstore                 0        0         0    - /sys/fs/pstore
bpf                    0        0         0    - /sys/fs/bpf
systemd-1              -        -         -    - /proc/sys/fs/binfmt_misc
hugetlbfs              0        0         0    - /dev/hugepages
mqueue                 0        0         0    - /dev/mqueue
debugfs                0        0         0    - /sys/kernel/debug
tracefs                0        0         0    - /sys/kernel/tracing
fusectl                0        0         0    - /sys/fs/fuse/connections
configfs               0        0         0    - /sys/kernel/config
ramfs                  0        0         0    - /run/credentials/systemd-sysusers.service
/dev/loop0           128      128         0 100% /snap/bare/5
/dev/loop1         65024    65024         0 100% /snap/core20/1974
/dev/loop2         75648    75648         0 100% /snap/core22/858
/dev/loop3        242944   242944         0 100% /snap/firefox/2987
/dev/loop4        358144   358144         0 100% /snap/gnome-3-38-2004/143
/dev/loop6         93952    93952         0 100% /snap/gtk-common-themes/1535
/dev/loop5        497280   497280         0 100% /snap/gnome-42-2204/120
/dev/loop7         12672    12672         0 100% /snap/snap-store/959
/dev/loop8         54656    54656         0 100% /snap/snapd/19457
/dev/loop9           512      512         0 100% /snap/snapd-desktop-integration/83
/dev/sda2         524252     6216    518036   2% /boot/efi
binfmt_misc            0        0         0    - /proc/sys/fs/binfmt_misc
tmpfs             296168      180    295988   1% /run/user/1000
gvfsd-fuse             0        0         0    - /run/user/1000/gvfs
```

- -h : used to display the size in power of 1024.

```
onworks@onworks:~/160121733092$ df -h helloworld.c
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3        29G   13G   15G  46% /
```

- -T : to display the file type.

```
onworks@onworks:~/160121733092$ df -T helloworld.c
Filesystem     Type 1K-blocks     Used Available Use% Mounted on
/dev/sda3      ext4  30267332 13120820  15583684  46% /
```

b. **mount**:

- The mount command allows users to mount, i.e., attach additional child file systems to a particular mount point on the currently accessible file system. The command passes the mount instructions to the kernel, which completes the operation.
- Syntax : mount -t [type] [device] [dir]

```
onworks@onworks:~/160121733092$ mount -l -t ext4
/dev/sda3 on / type ext4 (rw,relatime,errors=remount-ro)
/dev/sda3 on /var/snap/firefox/common/host-hunspell type ext4 (ro,noexec,noatime,errors=remount-ro)
```

c. **unmount** :

- The umount command detaches the file system(s) mentioned from the file hierarchy.

```
onworks@onworks:~/160121733092$ umount /dev/sda3
umount: /var/snap/firefox/common/host-hunspell: must be superuser to unmount.
```

5) Write a C program to check whether the given number is even or odd. Run the program using *strace command* and note down all the system call used by your program.

onworks@onworks: ~/160121733092

```
#include<stdio.h>
int main()
{
int n;
printf("Enter a number: ");
scanf("%d", &n);
if(n%2==0)
printf("%d is even number", n);
else
printf("%d is odd number", n);
}
```

```
onworks@onworks:~/160121733092$ vi evenodd.c
onworks@onworks:~/160121733092$ gcc evenodd.c -o evenodd
onworks@onworks:~/160121733092$ strace ./evenodd
execve("./evenodd", ["./evenodd"], 0x7ffeaaa76e00 /* 54 vars */) = 0
brk(NULL)                               = 0x56393d469000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd2be7dd50) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f1201d87000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=58071, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 58071, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f1201d78000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\"\233}\305\t\5?\344\337^)\350b\231\21\360"..., 68, 896) = 68
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2216304, ...}, AT_EMPTY_PATH) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f1201a00000
mmap(0x7f1201a28000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x
7f1201a28000
mmap(0x7f1201bbd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7f1201bbd0
00
mmap(0x7f1201c15000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000) = 0x
7f1201c15000
mmap(0x7f1201c1b000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f1201
c1b000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f1201d75000
arch_prctl(ARCH_SET_FS, 0x7f1201d75740) = 0
set_tid_address(0x7f1201d75a10)         = 11573
set_robust_list(0x7f1201d75a20, 24)     = 0
rseq(0x7f1201d760e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7f1201c15000, 16384, PROT_READ) = 0
mprotect(0x56393d3a4000, 4096, PROT_READ) = 0
mprotect(0x7f1201dc1000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7f1201d78000, 58071)           = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x2), ...}, AT_EMPTY_PATH) = 0
getrandom("\x21\x57\x78\x2e\x0f\xf3\x66\x37", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x56393d469000
brk(0x56393d48a000)                     = 0x56393d48a000
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x2), ...}, AT_EMPTY_PATH) = 0
write(1, "Enter a number: ", 16Enter a number: )        = 16
read(0, 0x56393d4696b0, 1024)           = ? ERESTARTSYS (To be restarted if SA_RESTART is set)
--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---
read(0, 0x56393d4696b0, 1024)           = ? ERESTARTSYS (To be restarted if SA_RESTART is set)
--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---
read(0,
```

```
read(0, 10
"10\n", 1024)                   = 3
write(1, "10 is even number", 1710 is even number)       = 17
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Illegal seek)
exit_group(0)                           = ?
+++ exited with 0 +++
```

OS Week 2 LAB ASSIGNMENT

1) What is Shell? Explain about types of Shells

A shell is a special user program that provides an interface for the user to use operating system services. Shell accepts human-readable commands from users and converts them into something which the kernel can understand. It is a command language interpreter that executes commands read from input devices such as keyboards or from files. The shell gets started when the user logs in or starts the terminal.

Types of Shells:

1) The C Shell (csh):

   It was created by Bill Joy at the University of California at Berkeley. It incorporated features such as aliases and command history. It includes helpful programming features like built-in arithmetic and C-like expression syntax.

2) The Bourne Shell (sh):

   It was written by Steve Bourne at AT&T Bell Labs. It is the original UNIX shell. It is faster and more preferred. It lacks features for interactive use like the ability to recall previous commands. It also lacks built-in arithmetic and logical expression handling.

3) The Korn Shell (ksh):

   It was written by David Korn at AT&T Bell Labs. It is a superset of the Bourne shell. So it supports everything in the Bourne shell.It has interactive features. It includes features like built-in arithmetic and C-like arrays, functions, and string-manipulation facilities. It is faster than C shell. It is compatible with script written for C shell.

4) GNU Bourne-Again Shell (bash):

   It is compatible to the Bourne shell. It includes features from Korn and Bourne shell

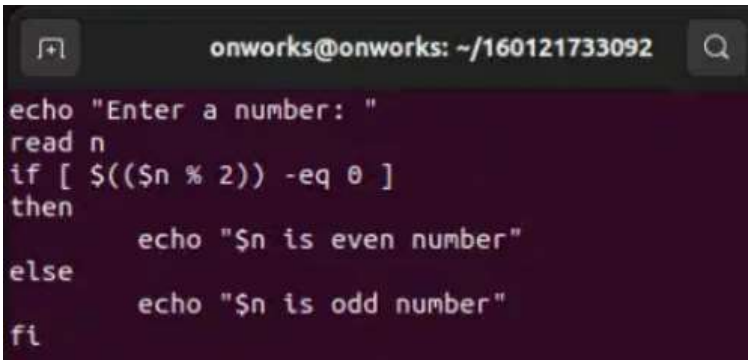
5) T-Shell (tsh):

   It was originally developed for the Plan 9 operating system, but has since been ported to other systems, including Linux, FreeBSD, and macOS.

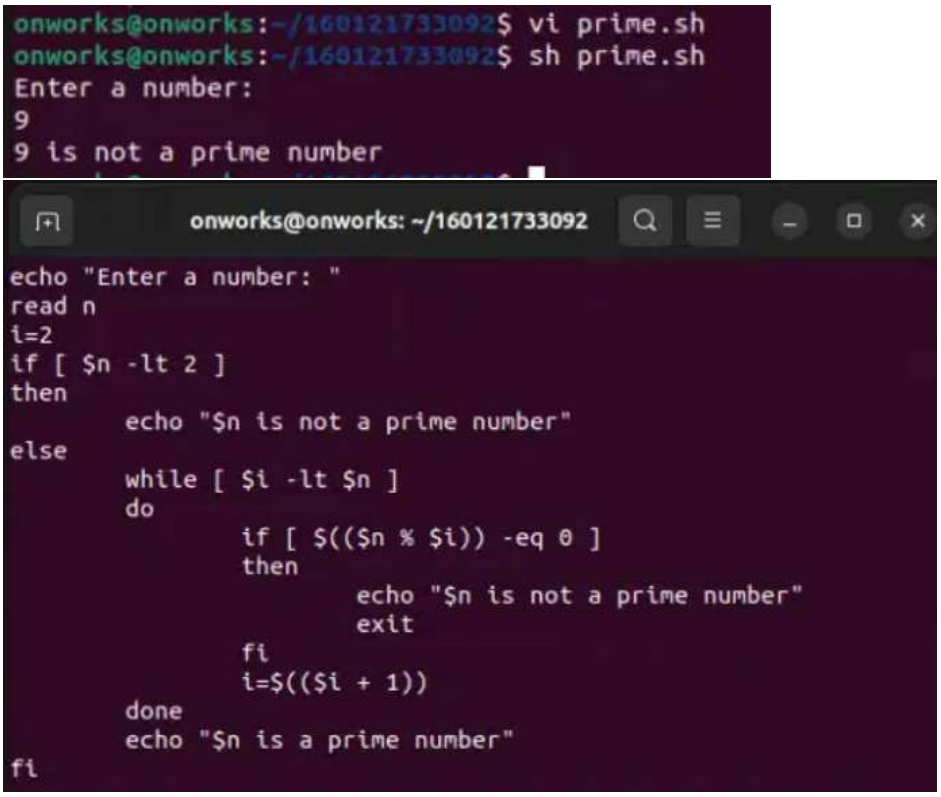
2) Write a Shell program to

i) To check given number is even or odd

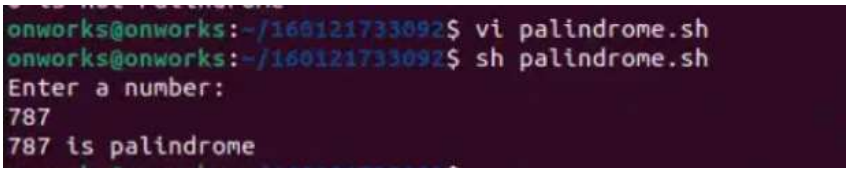
```
onworks@onworks:~/160121733092$ vi evenodd.sh
onworks@onworks:~/160121733092$ sh evenodd.sh
Enter a number:
4
4 is even number
```

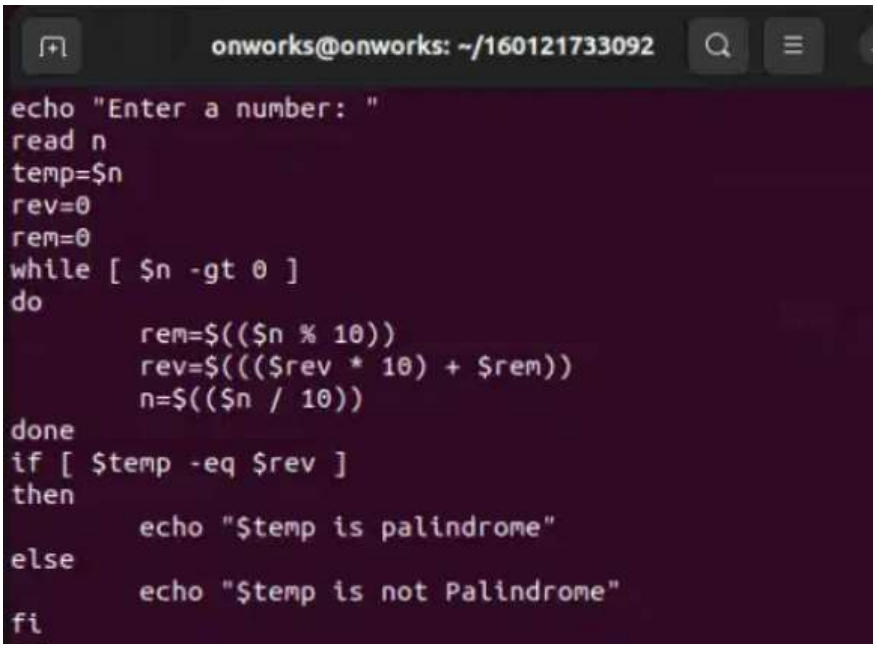


ii) **To check given number is prime or not**

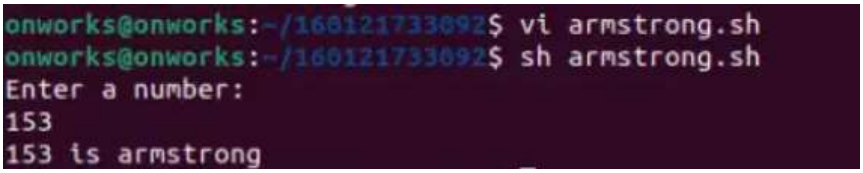


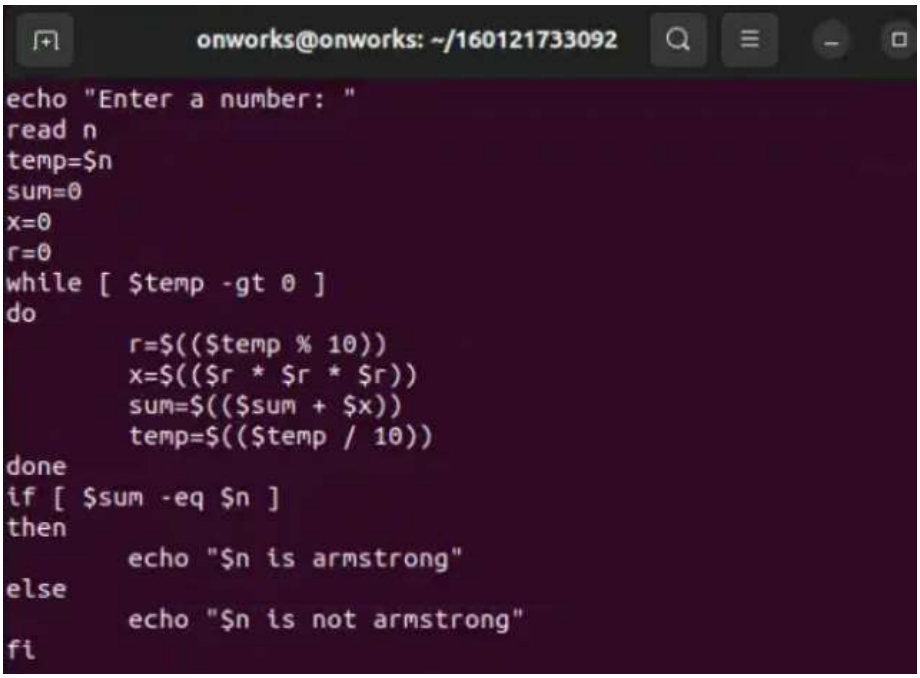
iii) **To check given number is palindrome number or not**





iv) To check given number is Armstrong number or not





3) Write about the following

a) shell

The shell can be defined as a command interpreter within an operating system like Linux/GNU or Unix. It is a program that runs other programs. The shell facilitates every user of the computer as an interface to the Unix/GNU Linux system. Hence, the user can execute different tools/utilities or commands with a few input data. The shell sends the result to the user over the screen when it has completed running a program which is the common output device. That's why it is known as "command interpreter". The shell is a programming language with complete constructs of a programming language such as functions, variables, loops, conditional execution, and many others.

b) Kernel

The kernel is the core component of an operating system. This provides a platform for programs and various services to run on top of it. The Linux kernel is modifiable according to the user's needs. The kernel virtualizes the computer's common hardware resources to provide each process with its own virtual resources. This makes the process seem as if it is the sole process running on the machine. The kernel is also responsible for preventing and mitigating conflicts between different processes.

c) Terminal

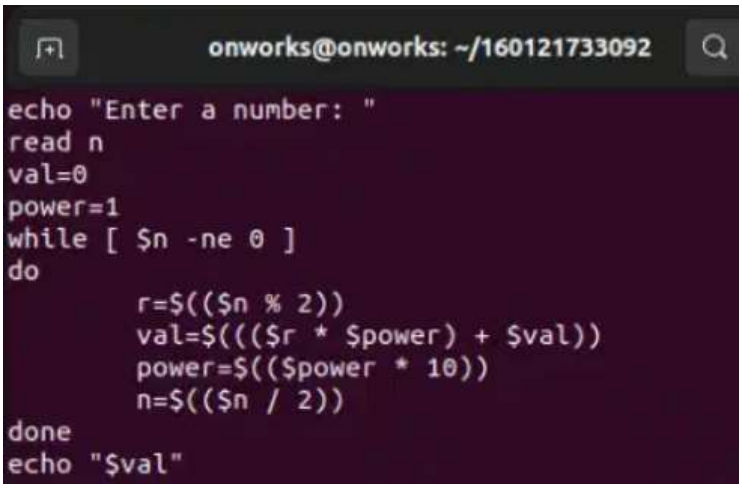
A Terminal is a program which is responsible for providing an interface to a user so that he/she can access the shell. It basically allows users to enter commands and see the output of those commands in a text-based interface. Large scripts that are written to automate and perform complex tasks are executed in the terminal.

A terminal typically consists of a text window or console where users can type commands. When a command is entered and executed, the terminal communicates with the operating system's shell (e.g., Bash, Zsh) to process the command and display the results. Terminals can be accessed locally on the computer itself or remotely over a network, making them a valuable tool for system administrators and developers

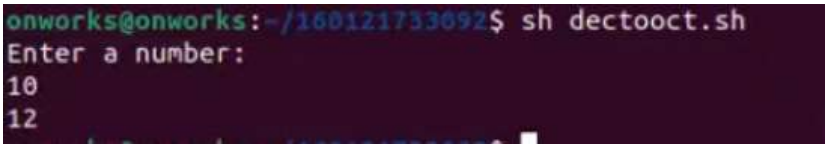
4) Write a Shell Script program to convert the

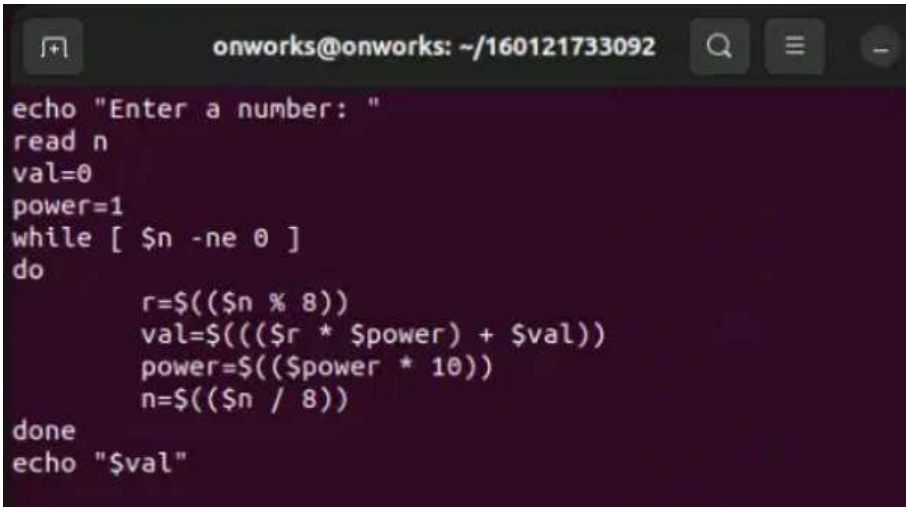
i) Base 10 number to binary

```
onworks@onworks:~/160121733092$ vi dectobin.sh
onworks@onworks:~/160121733092$ sh dectobin.sh
Enter a number:
10
1010
```

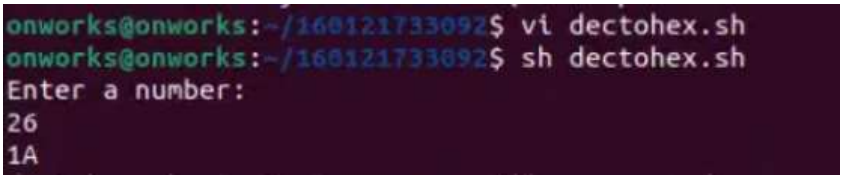


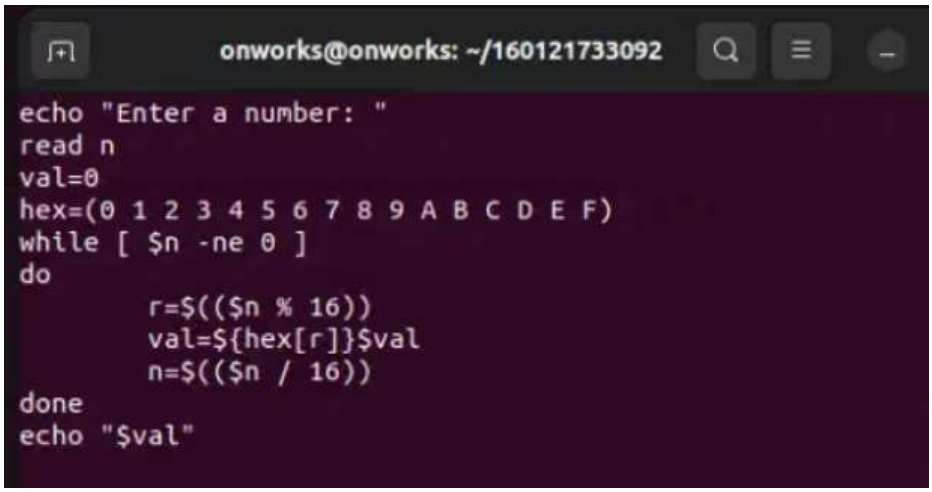
ii) Base 10 to octal



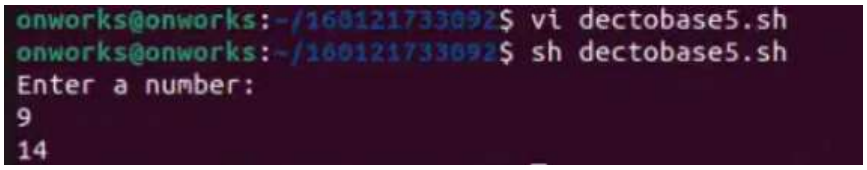


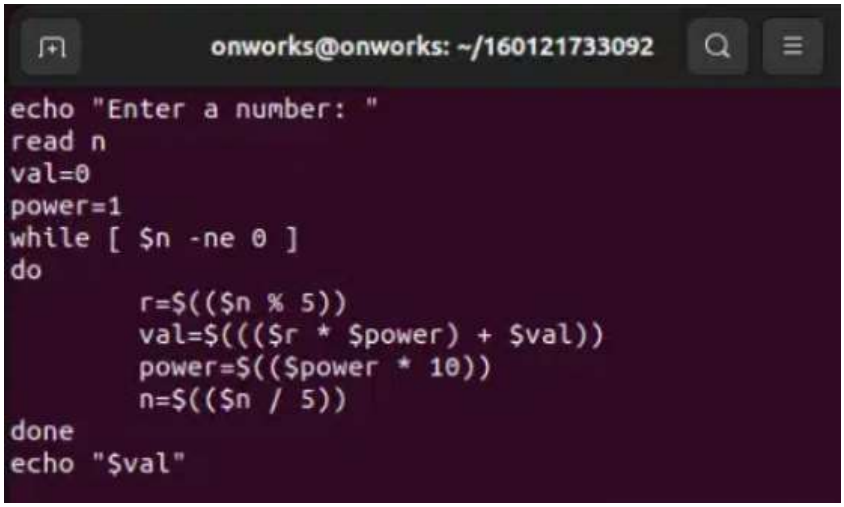
iii) Base 10 to Hexa decimal



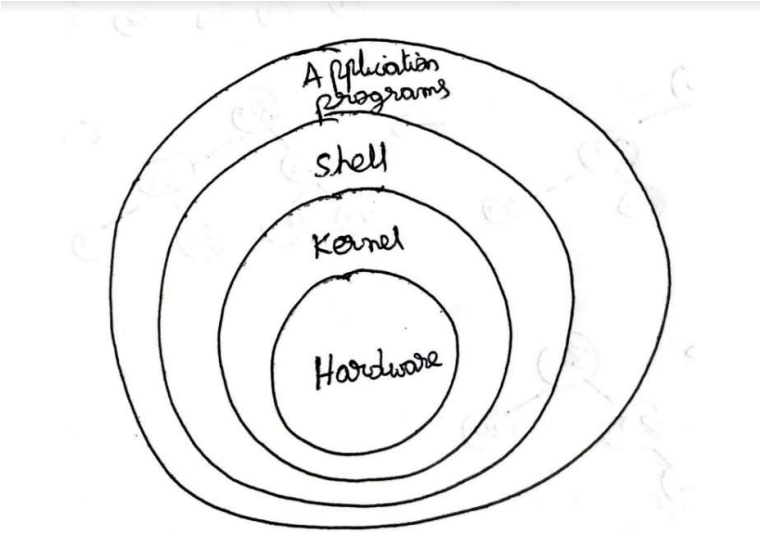


iv) Base 10 to Base 5





5) Draw the architecture diagram of Linux/Unix Operating System



1) **Write a C program to demonstrate the write and read system call [Hint: Use the Even or Odd Program]**

```
onworks@onworks:~/160121733092$ vi evenodd.c
onworks@onworks:~/160121733092$ gcc evenodd.c
onworks@onworks:~/160121733092$ ./a.out
Enter a number: 10
Even Number
```

onworks@onworks: ~/160121733092

```
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>
#include<string.h>
int main(int argc, char *argv[])
{
        int fd;
        char sample[10];
        write(1, "Enter a number: ", 16);
        read(0, sample, 2);
        if(atoi(sample)%2==0)
        {
                write(1,"Even Number", 11);
        }
        else
        {
                write(1, "Odd Number", 10);
        }
}
```

2) **Write a C program to demonstrate the use of following system calls [Each one example program]**
i)write()
ii)read()
iii)open()
iv)close()
v)lseek()

```
onworks@onworks:~/160121733092$ vi write.c
onworks@onworks:~/160121733092$ gcc write.c
onworks@onworks:~/160121733092$ ./a.out
Enter the File Name: sample.txt
Enter the content you would like to write: Hello
```

```c
#include<unistd.h>
#include<stdlib.h>
#include<fcntl.h>
#include<stdio.h>
#include<string.h>
int main(int argc, char *argv[]){
        char filename[20], buffer[100];
        write(1, "Enter the File Name: ", 21);
        //read(0, filename, 20);
        scanf("%s", filename);
        int fd=open(filename, O_WRONLY, 742);
        //printf("%d", fd);
        if (fd==-1){
                write(2, "File Not Found", 14);
        }
        else{
                write(1, "Enter the content you would like to write: ", 43);
                //read(0, buffer, 100);
                scanf("%s", buffer);
                write(fd, buffer, strlen(buffer));
                close(fd);
        }
}
~
```

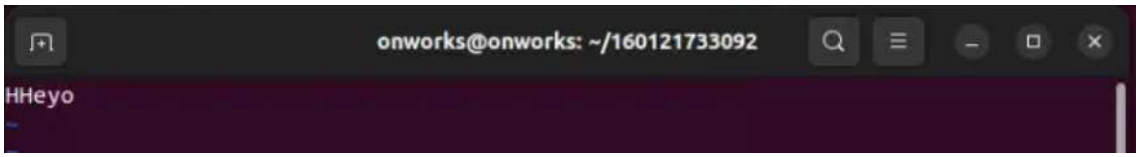
```c
#include<unistd.h>
#include<stdlib.h>
#include<fcntl.h>
#include<stdio.h>
#include<string.h>
int main(int argc, char *argv[]){
        char filename[20], buffer[100];
        write(1, "Enter the File Name: ", 21);
        //read(0, filename, 20);
        scanf("%s", filename);
        int fd=open(filename, O_RDONLY, 742);
        //printf("%d", fd);
        if (fd==-1){
                write(2, "File Not Found", 14);
        }
        else{
                read(fd, buffer, 100);
                write(1, buffer, strlen(buffer));
                close(fd);
        }
}
```

```
onworks@onworks:~/160121733092$ vi read.c
onworks@onworks:~/160121733092$ gcc read.c
onworks@onworks:~/160121733092$ ./a.out
Enter the File Name: sample.txt
Helloonworks@onworks:~/160121733092$
```

```
onworks@onworks:~/160121733092$ vi lseek.c
onworks@onworks:~/160121733092$ gcc lseek.c
onworks@onworks:~/160121733092$ ./a.out
File Name: sample.txt
```

onworks@onworks: ~/160121733092

```c
#include<stdio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<unistd.h>
int main(int argc, char *argv[])
{
        char file[20], sample[20];
        write(1, "File Name: ", 11);
        scanf("%s", file);
        int fd=open(file,O_RDWR,777);
        lseek(fd, 1, SEEK_CUR);
        write(fd, "Hey", 3);
        close(fd);
}
```



3) **What is a *File Descriptor*? Explain how it is assigned when a file is opened by a process?**

A file descriptor is a number that uniquely identifies an open file in a computer's operating system. It describes a data resource, and how that resource may be accessed.

File descriptors are assigned when a file is opened by a process as follows:

Standard File Descriptors:

- Standard Input (stdin, file descriptor 0): By default, when a process starts, file descriptor 0 points to the standard input, which is usually the keyboard. If you read from file descriptor 0, you read from the keyboard.
- Standard Output (stdout, file descriptor 1): File descriptor 1 points to the standard output, which is typically the terminal.
- Standard Error (stderr, file descriptor 2): File descriptor 2 points to the standard error, which is also the terminal by default. Error messages are often sent to file descriptor 2.

A file is opened using system call, open() in C. When a file is opened, the operating system returns a new file descriptor if the operation is successful. The file descriptor is an integer greater than or equal to 3.

Once a file is opened, the process can read from or write to the file using the read() and write() system calls.

It's essential for a process to close file descriptors when they are no longer needed. It is done by using the close() system call.

4) **Can a File descriptor is duplicated? If yes, Justify your answer with suitable example?**

Yes, a File descriptor can be duplicated.

By duplicating a file descriptor, multiple parts of a program can access the same file simultaneously

Consider the following example:

```
onworks@onworks:~/160121733092$ gcc fd.c
onworks@onworks:~/160121733092$ ./a.out
onworks@onworks:~/160121733092$ vi fd.c
onworks@onworks:~/160121733092$ vi duplicate.txt
```

onworks@onworks: ~/160121733092

```
#include<stdio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<unistd.h>
int main(){
        int fd=open("duplicate.txt",O_WRONLY);
        if(fd==-1)
        {
                write(2, "File not found", 14);
        }
        else
        {
                int dup_fd = dup(fd);
                write(dup_fd, "Duplcate", 8);
                write(fd, "Original", 8);
        }
}
```

onworks@onworks: ~/160121733092

```
DuplcateOriginal
```

5) Explain about Linux File System

A Linux file system is a structured collection of files on a disk drive or a partition. A partition is a segment of memory and contains some specific data. In our machine, there can be various partitions of the memory. Generally, every partition contains a file system.

Linux file system has a hierarchal file structure as it contains a root directory and its subdirectories. All other directories can be accessed from the root directory. A partition usually has only one file system, but it may have more than one file system.

The general-purpose computer system needs to store data systematically so that we can easily access the files in less time. It stores the data on hard disks (HDD) or some equivalent storage type. The Linux file system contains the following sections:

- The root directory (/)
- A specific data storage format (EXT3, EXT4, BTRFS, XFS and so on)
- A partition or logical volume having a particular file system.

Some key features of Linux file system are as following:

Specifying paths: Linux does not use the backslash (\) to separate the components; it uses forward slash (/) as an alternative. For example, as in Windows, the data may be stored in C:\ My Documents\ Work, whereas, in Linux, it would be stored in /home/ My Document/ Work.

Partition, Directories, and Drives: Linux does not use drive letters to organize the drive as Windows does. In Linux, we cannot tell whether we are addressing a partition, a network device, or an "ordinary" directory and a Drive.

Case Sensitivity: Linux file system is case sensitive. It distinguishes between lowercase and uppercase file names. Such as, there is a difference between test.txt and Test.txt in Linux. This rule is also applied for directories and Linux commands.

File Extensions: In Linux, a file may have the extension '.txt,' but it is not necessary that a file should have a file extension. While working with Shell, it creates some problems for the beginners to differentiate between files and directories. If we use the graphical file manager, it symbolizes the files and folders.

Hidden files: Linux distinguishes between standard files and hidden files, mostly the configuration files are hidden in Linux OS. Usually, we don't need to access or read the hidden files. The hidden files in Linux are represented by a dot (.) before the file name (e.g., .ignore). To access the files, we need to change the view in the file manager or need to use a specific command in the shell.

6) Explain about Linux Directory Structure in detailed

The Linux directory structure is hierarchical and follows a standardized layout defined by the Filesystem Hierarchy Standard (FHS). The following are essential directories in the Linux directory structure:

1. Root Directory (/):

The top-level directory in the Linux file system hierarchy. All other directories and files are subdirectories or files contained within the root directory.

2. Standard Subdirectories:

Linux directories serve specific purposes and organize files and system resources logically.

/bin: Essential user command binaries (e.g., ls, cp, mv) are stored here. These binaries are required for system repair and recovery.

/boot: Contains the bootloader configuration and the Linux kernel. Boot-related files are stored here.

/dev: Contains device files for all hardware devices on the system. Interacting with these files allows programs to communicate with hardware components.

/etc: System-wide configuration files and shell scripts are stored here. Administrators configure system settings and software using files in this directory.

/home: Home directories for individual users are located here. Each user has a separate subdirectory (e.g., /home/username).

/lib and /lib64: Libraries essential for binaries in /bin and /sbin are stored here.

/media: Mount point for removable media devices such as USB drives and CD-ROMs.

/mnt: Temporarily mounted file systems.

/opt: Optional software packages can be installed here. It's often used for software that is not part of the default installation.

/proc: A virtual file system providing information about processes and system status.

/root: Home directory for the root user.

/run: Contains system runtime data, such as PID files and sockets, which are recreated on boot.

/sbin: System binaries essential for system administration tasks. Only accessible by the root user.

/srv: Data for services provided by the system (e.g., websites) can be placed here.

/sys: A virtual file system exposing information and configuration options for the kernel and devices.

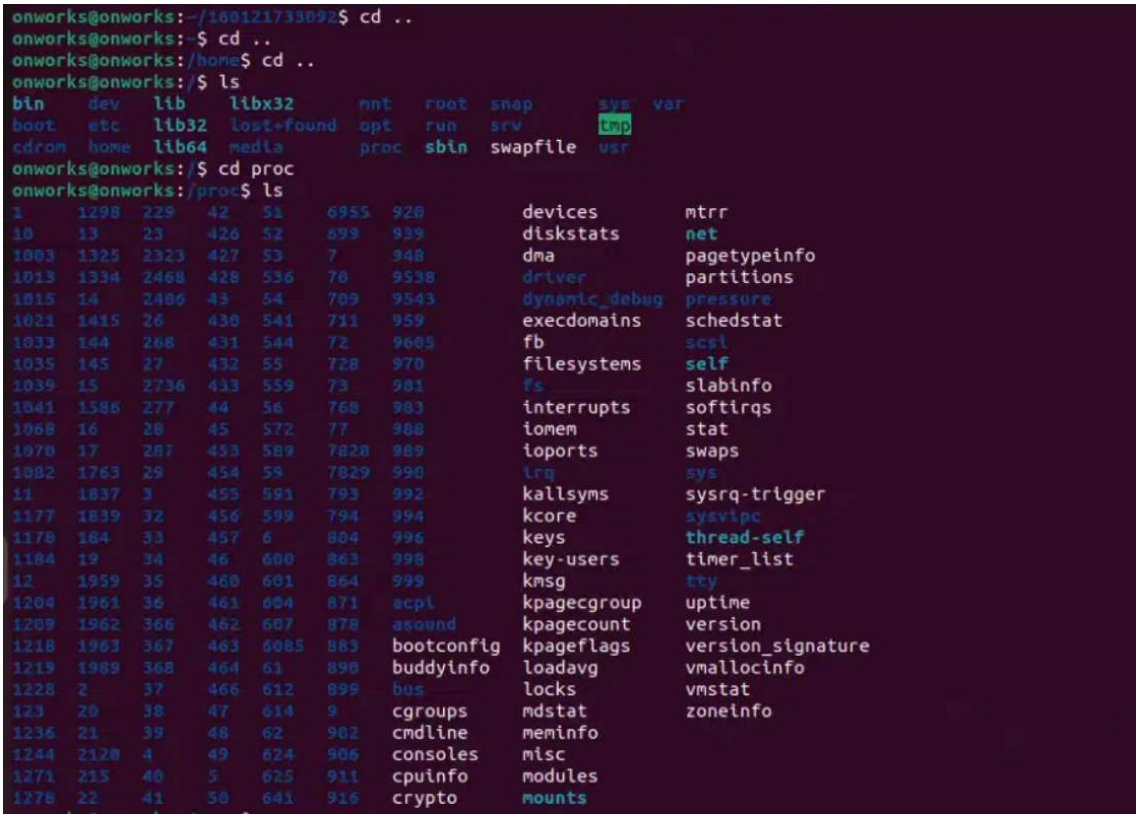
/tmp: Temporary files created by system and users. Files here are deleted upon reboot.

/usr: Contains user binaries, libraries, documentation, and source code. Has subdirectories like /usr/bin, /usr/lib, and /usr/share.

/var: Contains variable data files, such as logs, databases, mail, and temporary files. Files in /var are expected to grow over time.

7) What does a *./proc* folder contain. Explain with an example program

The /proc folder contains the following:



The /proc directory in a Linux system contains a virtual file system that exposes information about running processes and system parameters to users and system administrators. It provides a dynamic view of the kernel, processes, and various system-related information. The files and subdirectories in the /proc directory allow you to access and manipulate kernel and process-related information using a file-like interface. The proc file system also provides a communication medium between kernel space and user space.

ls -l /proc :This command is used to list all the files and directories under the /proc directory.

ls -ltr /proc/pid: If we want to check information about the process with pid, we can use this command.

ls -ltr /proc/pid/status: To View The status of the process with PID , we can use this command.

ls -ltr /proc/pid/statm: To View The memory usage of the process with PID, we can use this command