Name: **Tushar Nankani** Roll No: **1902112** Batch: **C23**

Operating System: Assignment 1

Problem Statement: Exploring internal commands of the Linux OS.

THEORY: What are internal commands ?

The UNIX system is command-based i.e things happen because of the commands that you key in. All UNIX commands are seldom more than four characters long. For a linux system, there are various internal commands with which it can be run more efficiently. Commands which are built into the shell are called internal commands.

COMMANDS:

1. **ls** -The ls command is used to list files or directories in Linux and other Unix-based operating systems. Just like you navigate in your File explorer or Finder with a GUI, the `ls` command allows you to list all files or directories in the current directory by default, and further interact with them via the command line. The syntax is :

   `ls [flags] [directory]`

```
$ ls
bin   etc         initrd.img.old  lib64       media  proc  sbin  sys  var
boot  home        lib             libx32      mnt    root  snap  tmp  vmlinuz
dev   initrd.img  lib32           lost+found  opt    run   srv   usr  vmlinuz.old
```

```
$ ls -l
total 96
drwxr-xr-x   2 root root  4096 Nov 10 12:21 bin
drwxr-xr-x   3 root root  4096 Nov 10 12:20 boot
drwxr-xr-x  16 root root  3840 Feb 24 16:30 dev
drwxr-xr-x  97 root root  4096 Nov 10 12:24 etc
drwxr-xr-x   3 root root  4096 Nov 10 12:18 home
lrwxrwxrwx   1 root root    33 Nov 10 12:20 initrd.img -> boot/initrd.img-4.4.0-193-generic
lrwxrwxrwx   1 root root    32 Nov 10 12:20 initrd.img.old -> boot/initrd.img-4.4.0-62-generic
drwxr-xr-x  19 root root  4096 Nov 10 12:21 lib
drwxr-xr-x   2 root root  4096 Nov 10 12:21 lib32
drwxr-xr-x   2 root root  4096 Nov 10 12:19 lib64
drwxr-xr-x   2 root root  4096 Nov 10 12:21 libx32
drwx------   2 root root 16384 Nov 10 12:16 lost+found
drwxr-xr-x   4 root root  4096 Nov 10 12:16 media
drwxr-xr-x   2 root root  4096 Feb 15  2017 mnt
drwxr-xr-x   6 root root  4096 Nov 10 12:23 opt
dr-xr-xr-x 109 root root     0 Feb 24 16:30 proc
drwx------   4 root root  4096 Feb 24 16:30 root
drwxr-xr-x  22 root root   820 Feb 24 16:50 run
drwxr-xr-x   2 root root 12288 Nov 10 12:23 sbin
drwxr-xr-x   2 root root  4096 Nov 10 12:21 snap
drwxr-xr-x   2 root root  4096 Feb 15  2017 srv
dr-xr-xr-x  13 root root     0 Feb 24 16:30 sys
drwxrwxrwt   7 root root  4096 Feb 24 16:31 tmp
drwxr-xr-x  13 root root  4096 Nov 10 12:23 usr
drwxr-xr-x  12 root root  4096 Nov 10 12:21 var
lrwxrwxrwx   1 root root    30 Nov 10 12:20 vmlinuz -> boot/vmlinuz-4.4.0-193-generic
lrwxrwxrwx   1 root root    29 Nov 10 12:20 vmlinuz.old -> boot/vmlinuz-4.4.0-62-generic
```

```
$ ls -a
.   bin   dev  home        initrd.img.old  lib32  libx32      media  opt   root  sbin  srv  tmp  var      vmlinuz.old
..  boot  etc  initrd.img  lib             lib64  lost+found  mnt    proc  run   snap  sys  usr  vmlinuz
$ ls
bin   dev  home        initrd.img.old  lib32  libx32      media  opt   root  sbin  srv  tmp  var      vmlinuz.old
boot  etc  initrd.img  lib             lib64  lost+found  mnt    proc  run   snap  sys  usr  vmlinuz
$ ls --author
bin   dev  home        initrd.img.old  lib32  libx32      media  opt   root  sbin  srv  tmp  var      vmlinuz.old
boot  etc  initrd.img  lib             lib64  lost+found  mnt    proc  run   snap  sys  usr  vmlinuz
```

2. **mkdir** - mkdir command in Linux allows the user to create directories (also referred to as folders in some operating systems). This command can create multiple directories at once as well as set the permissions for the directories. It is important to note that the user executing this command must have enough permissions to create a directory in the parent directory, or he/she may receive a 'permission denied' error.
   Syntax:

   ```
   mkdir [options...] [directories ...]
   ```

   ```
   $ cd root/
   $ mkdir OS
   $ ls
   OS
   $ mkdir -v DBMS
   mkdir: created directory 'DBMS'
   $ ls
   DBMS  OS
   ```

3. **chdir** - chdir changes the current working directory of the calling process to the directory specified in path.
   Syntax:

   ```
   chdir [ directory ]
   ```

   ```
   $ cd root/
   $ ls
   OS
   $ cd OS
   $ pwd
   /root/OS
   $
   ```

4. rmdir - rmdir command is used remove empty directories from the filesystem in Linux. The rmdir command removes each and every directory specified in the command line only if these directories are empty. So, if the specified directory has some directories or files in it then this cannot be removed by rmdir command.

```
$ cd ..
$ ls
OS
$ rmdir OS
$ ls
$ ls -v
$
```

5. cat-The cat (short for "concatenate") command is one of the most frequently used command in Linux/Unix like operating systems. cat command allows us to create single or multiple files, view contain of file, concatenate files and redirect output in terminal or files.
   Syntax:
   `cat [OPTIONS] [FILE_NAMES]`

```
$ mkdir OS
$ ls
OS
$ cd OS
$ cat >> a.txt
This is a file.
Line no. 2
XYZ
^Z
[1]+  Stopped                 cat >> a.txt
$ cat a.txt
This is a file.
Line no. 2
XYZ
```

```
$ cat -A a.txt
This is a file.$
Line no. 2$
XYZ$
$ cat -b a.txt
     1  This is a file.
     2  Line no. 2
     3  XYZ
$
```

6. rm - rm is a command-line utility for removing files and directories. It is one of the essential commands that every Linux user should be familiar with.
   Syntax:
   `rm [files or directory]`

```
$ ls
a.txt
$ rm -v a.txt
removed 'a.txt'
$ ls
$
```

7. mv -mv stands for move. mv is used to move one or more files or directories from one place to another in file system like UNIX. It has two distinct functions:
   (i) It rename a file or folder.
   (ii) It moves group of files to different directory.
   No additional space is consumed on a disk during renaming. This command normally works silently means no prompt for confirmation.
   Syntax:

   `mv file_to_be_moved destination_directory`

```
$ ls
$ cat >> a.txt
1
2
3
^Z
[2]+  Stopped                 cat >> a.txt
$ cat >> b.txt
A
B
C
^Z
[3]+  Stopped                 cat >> b.txt
$ ls
a.txt  b.txt
$ mv a.txt ./..
$ ls
b.txt
$ cd ..
$ ls
a.txt  OS
$
```

8. cp - cp stands for copy. This command is used to copy files or group of files or directory. It creates an exact image of a file on a disk with different file name. cp command require at least two filenames in its arguments.

```
$ ls
a.txt  OS
$ cp -a a.txt d.txt
$ cat d.txt
1
2
3
$
```

```
$ ls
a.txt  d.txt  OS
$ cp --backup a.txt e.txt
$ cat e.txt
1
2
3
$ ls
a.txt  d.txt  e.txt  OS
$
```

9. head - The head command, as the name implies, print the top N number of data of the given input. By default, it prints the first 10 lines of the specified files. If more than one file name is provided then data from each file is preceded by its file name.

   `head –no_of_lines file_name`

```
$ cat a.txt | head -1
1
$ head -2 a.txt
1
2
$
```

```
$ head -n -1 a.txt
1
2
$ head -c a.txt
head: invalid number of bytes: 'a.txt'
$ head -c -1  a.txt
1
2
3$
```

10. tail - It is the complementary of head command.The tail command, as the name implies, print the last N number of data of the given input. By default it prints the last 10 lines of the specified files. If more than one file name is provided then data from each file is precedes by its file name.

    `tail –no_of_lines file_name`

```
3$ tail -1 a.txt
3
$ tail a.txt
1
2
3
$
```

11. sort -SORT command is used to sort a file, arranging the records in a particular order. By default, the sort command sorts file assuming the contents are ASCII. Using options in sort command, it can also be used to sort numerically.

- SORT command sorts the contents of a text file, line by line.
- sort is a standard command line program that prints the lines of its input or concatenation of all files listed in its argument list in sorted order.
- The sort command is a command line utility for sorting lines of text files. It supports sorting alphabetically, in reverse order, by number, by month and can also remove duplicates.
- The sort command can also sort by items not at the beginning of the line, ignore case sensitivity and return whether a file is sorted or not. Sorting is done based on one or more sort keys extracted from each line of input.
- By default, the entire input is taken as sort key. Blank space is the default field separator.

```
$ cat >> file.cpp
#include<iostream>
using namespace std;
class Car{
    int groundclearence;
    public:
    int length;
    Car(){
        groundclearence=180;            // Line 1
        length=3500;
    }
  void setValue(int w, int l){
    groundclearence=w;
    length=l;
  }
};
int main(){
  Car Safari;
  Safari.groundclearence = 200;         // Line 2
  Safari.setValue(200,4660);           // Line 3
  return 0;
}

^Z
[4]+  Stopped                 cat >> file.cpp
```

```
$ sort file.cpp | head -5

    }
  }
}
};
$ sort file.cpp | head -10

    }
  }
}
};
    Car(){
  Car Safari;
class Car{
        groundclearence=180;            // Line 1
    groundclearence=w;
$
```

12. wc - wc Command in Linux (Count Number of Lines, Words, and Characters) On Linux and Unix-like operating systems, the wc command allows you to count the number of lines, words, characters, and bytes of each given file or standard input and print the result.

`wc [options] filenames`

wc -l : Prints the number of lines in a file.

wc -w : prints the number of words in a file.

wc -c : Displays the count of bytes in a file.

wc -m : prints the count of characters from a file.

wc -L : prints only the length of the longest line in a file.

```
$ wc file.cpp
 22  44 406 file.cpp
$ wc -l file.cpp
22 file.cpp
$ wc -w file.cpp
44 file.cpp
$ ec -c file.cpp
ec: command not found
$ wc -c file.cpp
406 file.cpp
$ wc -L file.cpp
49 file.cpp
```

13. chown - The chown command allows you to change the user and/or group ownership of a given file, directory, or symbolic link.

`chown [OPTIONS] USER[:GROUP] FILE(s)`

Ownership and Permissions: To protect and secure files and directory in Linux we use permissions to control what a user can do with a file or directory. Linux uses three types of permissions:

Read: This permission allows the user to read files and in directories, it lets the user read directories and subdirectories stores in it.

Write: This permission allows a user to modify and delete a file. Also it allows a user to modify its contents (create, delete and rename files in it) for the directories. Unless the execute permission is not given to directories changes does do affect them.

Execute: The write permission on a file allows it to get executed. For example, if we have a file named php.sh so unless we don't give it execute permission it won't run.

Types of file Permissions:

- User: These type of file permission affect the owner of the file.
- Group: These type of file permission affect the group which owns the file.

Instead of the group permissions, the user permissions will apply if the owner user is in this group.

 Other: These type of file permission affect all other users on the system.

```
NAME
       chown - change file owner and group

SYNOPSIS
       chown [OPTION]... [OWNER][:[GROUP]] FILE...
       chown [OPTION]... --reference=RFILE FILE...

DESCRIPTION
       This  manual  page  documents  the  GNU version of chown.  chown changes the user and/or group ownership of each
       given file.  If only an owner (a user name or numeric user ID) is given, that user is made  the  owner  of  each
       given  file,  and  the  files'  group  is not changed.  If the owner is followed by a colon and a group name (or
       numeric group ID), with no spaces between them, the group ownership of the files is changed as well.  If a colon
       but  no group name follows the user name, that user is made the owner of the files and the group of the files is
       changed to that user's login group.  If the colon and group are given, but the owner is omitted, only the  group
       of  the files is changed; in this case, chown performs the same function as chgrp.  If only a colon is given, or
       if the entire operand is empty, neither the owner nor the group is changed.

OPTIONS
       Change the owner and/or group of each FILE to OWNER and/or GROUP.  With --reference, change the owner and  group
       of each FILE to those of RFILE.

       -c, --changes
              like verbose but report only when a change is made

       -f, --silent, --quiet
              suppress most error messages

       -v, --verbose
              output a diagnostic for every file processed
```

14. **chmod** -In Unix-like operating systems, the chmod command is used to change the access mode of a file. The name is an abbreviation of change mode.

`chmod [OPTIONS] MODE FILE...`

```
$ ls
a.txt  d.txt  e.txt  file.cpp  os
$ ls -l
total 20
-rw-r--r-- 1 root root    6 Feb 24 17:09 a.txt
-rw-r--r-- 1 root root    6 Feb 24 17:09 d.txt
-rw-r--r-- 1 root root    6 Feb 24 17:16 e.txt
-rw-r--r-- 1 root root  406 Feb 24 17:25 file.cpp
drwxr-xr-x 2 root root 4096 Feb 24 17:10 os
$ chmod 777 a.txt
$ ls -l
total 20
-rwxrwxrwx 1 root root    6 Feb 24 17:09 a.txt
-rw-r--r-- 1 root root    6 Feb 24 17:09 d.txt
-rw-r--r-- 1 root root    6 Feb 24 17:16 e.txt
-rw-r--r-- 1 root root  406 Feb 24 17:25 file.cpp
drwxr-xr-x 2 root root 4096 Feb 24 17:10 os
$
```

15. **chgrp** - chgrp command in Linux is used to change the group ownership of a file or directory. All files in Linux belong to an owner and a group. You can set the owner by using “chown” command, and the group by the “chgrp” command.

```
chgrp [OPTION]… GROUP FILE…
chgrp [OPTION]… –reference=RFILE FILE…
```

Example 1: To change the group ownership of a file.

```
sudo chgrp geeksforgeeks abc.txt
```

Here the group name of the file abc.txt was changed from kcVirtual to geeksforgeeks. Note that when files are created the groupname of the file
is same as the owner under which the file was created.

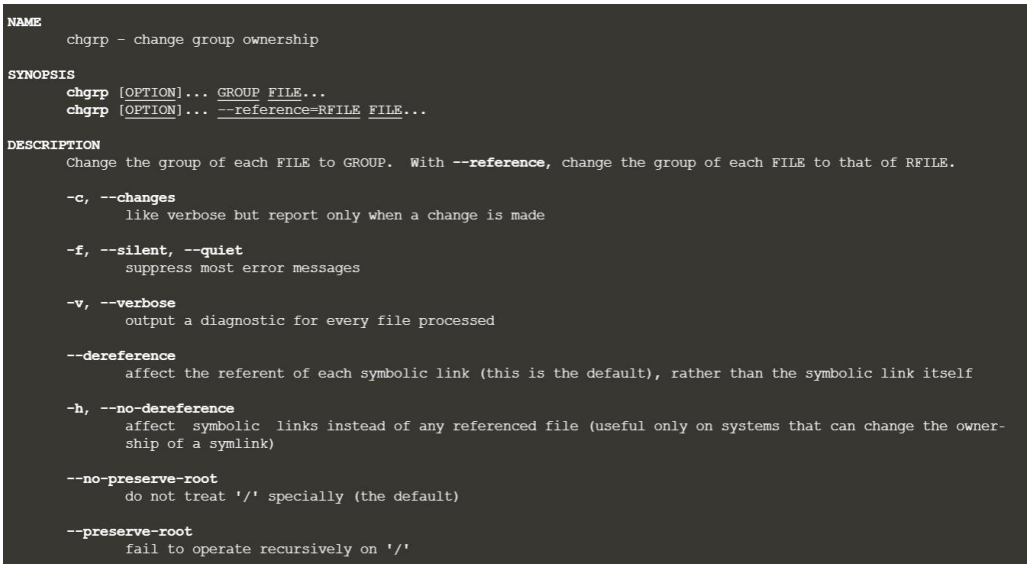
Example 2: To change the group ownership of a folder.

```
sudo chgrp geeksforgeeks GFG
```

Example 3: To recursively change the group ownership of a folder and all of its contents.

```
sudo chgrp -R geeksforgeeks GFG
```

The group of the folder GFG and its contents F1, F2 was all kcvirtual initially and they were changed to geeksforgeeks with the single command.



```
NAME
       chgrp - change group ownership

SYNOPSIS
       chgrp [OPTION]... GROUP FILE...
       chgrp [OPTION]... --reference=RFILE FILE...

DESCRIPTION
       Change the group of each FILE to GROUP.  With --reference, change the group of each FILE to that of RFILE.

       -c, --changes
              like verbose but report only when a change is made

       -f, --silent, --quiet
              suppress most error messages

       -v, --verbose
              output a diagnostic for every file processed

       --dereference
              affect the referent of each symbolic link (this is the default), rather than the symbolic link itself

       -h, --no-dereference
              affect  symbolic  links instead of any referenced file (useful only on systems that can change the owner-
              ship of a symlink)

       --no-preserve-root
              do not treat '/' specially (the default)

       --preserve-root
              fail to operate recursively on '/'
```

16. umask - On Linux and Unix operating systems, all new files are created with a default set of permissions. The umask utility allows you to view or to set the file mode creation mask, which determines the permissions bits for newly created files or directories.

When using the term Umask, we are referring to one of the following two meanings:

- The user file creation mode mask that is used to configure the default permissions for newly created files and directories
- The command “umask” which is used to set the umask value

-

```
$ umask
0022
$ umask 002
$ umask
0002
$
```

17. ps - Linux provides us a utility called ps for viewing information related with the processes on a system which stands as abbreviation for “Process Status”. ps command is used to list the currently running processes and their PIDs along with some other information depends on different options.

```
$ ps
  PID TTY          TIME CMD
  826 pts/0    00:00:00 bash
 1758 pts/0    00:00:00 cat
 1844 pts/0    00:00:00 cat
 1845 pts/0    00:00:00 cat
 2166 pts/0    00:00:00 cat
 2501 pts/0    00:00:00 ps
$ ps -a
  PID TTY          TIME CMD
 1758 pts/0    00:00:00 cat
 1844 pts/0    00:00:00 cat
 1845 pts/0    00:00:00 cat
 2166 pts/0    00:00:00 cat
 2502 pts/0    00:00:00 ps
$
```

18. pipe - A pipe is a form of redirection (transfer of standard output to some other destination) that is used in Linux and other Unix-like operating systems to send the output of one command/program/process to another command/program/process for further processing. The Unix/Linux systems allow stdout of a command to be connected to stdin of another command. You can make it do so by using the pipe character '|'.

The syntax for the pipe or unnamed pipe command is the | character between any two commands:

`Command-1 | Command-2 | … | Command-N`

```
$ ps | head -5
  PID TTY          TIME CMD
  826 pts/0    00:00:00 bash
 1758 pts/0    00:00:00 cat
 1844 pts/0    00:00:00 cat
 1845 pts/0    00:00:00 cat
```

19. Redirection Operators-

A redirection operator is a special character that can be used with a command, like a Command Prompt command or DOS command, to either redirect the input to the command or the output from the command.

With redirection, the above standard input/output can be changed.

Output Redirection

The '>' symbol is used for output (STDOUT) redirection.

Note: Use the correct file name while redirecting command output to a file. If there is an existing file with the same name, the redirected command will delete the contents of that file and then it may be overwritten."

If you do not want a file to be overwritten but want to add more content to an existing file, then you should use '>>' operator.

Input redirection

The '<' symbol is used for input(STDIN) redirection

```
$ cat >> a.txt
tuyscukvbsdv
djvlhdbfbd
fbkfhvbzkbf
232342
2452
23423
12323547
4564
5645
423
43
4534
6524
54
73
453
^[[15~64
^Z
[1]+  Stopped                 cat >> a.txt
```

```
$ sort a.txt > sorted.txt
$ cat sorted.txt
12323547
64
232342
23423
2452
423
43
453
4534
4564
54
5645
6524
73
djvlhdbfbd
fbkfhvbzkbf
tuyscukvbsdv
$
```

Name: **Tushar Nankani** Roll No: **1902112** Batch: **C23**

20. Explore commands for following:

a. **Display top 10 processes in descending order.**

```
$ ps aux | sort -nk +4 | tail
```

```
$ ps aux | sort -nk +4 | tail
root       612  0.0  0.2  47620  3500 ?        Ss   15:28   0:00 /sbin/rpcbind -f -w
syslog     637  0.0  0.2 256388  3336 ?        Ssl  15:28   0:00 /usr/sbin/rsyslogd -n
ntp        799  0.0  0.3 110032  5132 ?        Ssl  15:28   0:00 /usr/sbin/ntpd -p /var/run/ntpd.pid -g -u 109:116
root         1  0.0  0.3  37628  5644 ?        Ss   15:28   0:01 /sbin/init nosplash nosplash
root      1081  0.0  0.3  22652  5208 pts/0    Ss   15:29   0:00 -bash
root      1069  0.0  0.4  92796  6856 ?        Ss   15:29   0:00 sshd: root@pts/0
root       651  0.0  0.4 275872  6448 ?        Ssl  15:28   0:00 /usr/lib/accountsservice/accounts-daemon
root       765  0.0  0.4  65508  6296 ?        Ss   15:28   0:00 /usr/sbin/sshd -D
root       767  0.0  2.9 518524 44996 ?        Ssl  15:28   0:00 /usr/bin/containerd
root       840  0.0  5.2 499264 79832 ?        Ssl  15:28   0:00 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

Explanation: **ps** returns all running processes which are then *sorted by the 4th field* in numerical order and the top 10 are sent to STDOUT.

b. **Display the process with highest memory usage.**

```
$ top
```

```
top - 16:27:22 up 58 min,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:  99 total,   1 running,  97 sleeping,   1 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni, 99.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.2 st
KiB Mem :  1531700 total,  1167484 free,    89624 used,   274592 buff/cache
KiB Swap:  4118524 total,  4118524 free,        0 used.  1291028 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
    1 root      20   0   37628   5644   3988 S   0.0  0.4   0:01.38 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.01 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S   0.0  0.0   0:00.08 rcu_sched
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 migration/0
   10 root      rt   0       0      0      0 S   0.0  0.0   0:00.01 watchdog/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:00.01 watchdog/1
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 migration/1
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.01 ksoftirqd/1
   15 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:0H
```

Explanation: One of the best commands for looking at memory usage is top. One extremely easy way to see what processes are using the most memory is to start top and then press shift+m to switch the order of the processes shown to rank them by the percentage of memory each is using.

c. Display current user logged in and logname

`$ w`

The **w** command shows information about the Linux users currently on the server, and their running processes.

`$ who`

Linux shows **who** is logged on.

`$ whoami`

Find out who you are currently logged in as on Linux.

`$ id`

Alternative to whoami.

```
$ w
 16:35:12 up 4 min,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0    172.17.0.22      16:30    0.00s  0.02s  0.00s w
$ who
root     pts/0        2021-02-24 16:30 (172.17.0.22)
$ whoami
root
$ id
uid=0(root) gid=0(root) groups=0(root)
$
```

d. Display current shell, home directory, operating system type, current path setting, current working directory.

- Current shell

  $ echo $SHELL

  There are multiple methods to do so, but this one is the simplest.

```
$ echo $SHELL
/bin/bash
```

- Home Directory

  $ pwd

Home directory information can be achieved by the following command.

```
$ pwd
/root
```

- OS Type

  $ sb_release -a

```
Distributor ID: Ubuntu
Description:    Ubuntu 16.04.7 LTS
Release:        16.04
Codename:       xenial
```

- Current Path Setting

  $ echo $PATH | TR “:” “\n” | nl

- Current Working Directory

  $ pwd

```
$ pwd
/root
```

e. Display OS version, release number, kernel version.

- OS Version, Release Number

```
$ sb_release -a
```

```
Distributor ID: Ubuntu
Description:    Ubuntu 16.04.7 LTS
Release:        16.04
Codename:       xenial
```

- Kernel Version

```
$ uname -r
```

```
$ uname -r
4.4.0-193-generic
$
```