CS 630 -Lab

1. Group members:

Soumyadeep Basu(sb2356@njit.edu)

Charanpreet Kaur(ckd22@njit.edu)

Anjo Sam Thomas(at672@njit.edu)

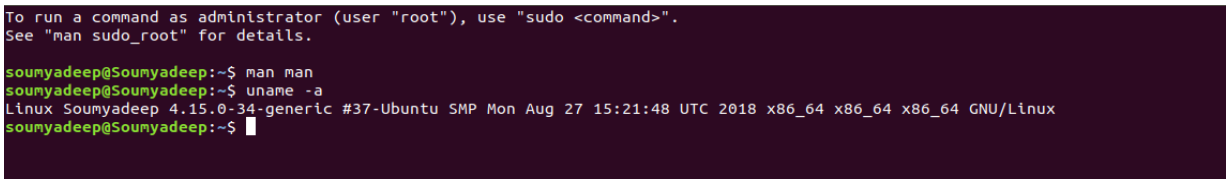
2. **man** Command

It helps to give the information of the command (manual pages of the command) passed as argument. ‘Man’ itself is a command and includes Name, synopsis, description, overview, defaults, options, and examples etc. when executed.

```
MAN(1)                                                    Manual pager utils                                                    MAN(1)

NAME
       man - an interface to the on-line reference manuals

SYNOPSIS
       man  [-C  file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-m system[,...]] [-M path] [-S list] [-e extension] [-i|-I] [--regex|--wildcard] [--names-only] [-a] [-u] [--no-subpages]
       [-P pager] [-r prompt] [-7] [-E encoding] [--no-hyphenation] [--no-justification] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] [[section] page[.section] ...] ...
       man -k [apropos options] regexp ...
       man -K [-w|-W] [-S list] [-i|-I] [--regex] [section] term ...
       man -f [whatis options] page ...
       man -l [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-P pager] [-r prompt] [-7] [-E encoding] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] file ...
       man -w|-W [-C file] [-d] [-D] page ...
       man -c [-C file] [-d] [-D] page ...
       man [-?V]

DESCRIPTION
       man is the system's manual pager.  Each page argument given to man is normally the name of a program, utility or function.  The manual page associated with each of these arguments is then found  and
       displayed.   A section, if provided, will direct man to look only in that section of the manual.  The default action is to search in all of the available sections following a pre-defined order ("1 n
       l 8 3 2 3posix 3pm 3perl 3am 5 4 9 6 7" by default, unless overridden by the SECTION directive in /etc/manpath.config), and to show only the first page found, even if page  exists  in  several  sec-
       tions.

       The table below shows the section numbers of the manual followed by the types of pages they contain.

       1   Executable programs or shell commands
       2   System calls (functions provided by the kernel)
       3   Library calls (functions within program librarles)
       4   Special files (usually found in /dev)
       5   File formats and conventions eg /etc/passwd
       6   Games
       7   Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
       8   System administration commands (usually only for root)
       9   Kernel routines [Non standard]

       A manual page consists of several sections.

       Conventional section names include NAME, SYNOPSIS, CONFIGURATION, DESCRIPTION, OPTIONS, EXIT STATUS, RETURN VALUE, ERRORS, ENVIRONMENT, FILES, VERSIONS, CONFORMING TO, NOTES, BUGS, EXAMPLE, AUTHORS,
       and SEE ALSO.

       The following conventions apply to the SYNOPSIS section and can be used as a guide in other sections.

       bold text          type exactly as shown.
       italic text        replace with appropriate argument.
       [-abc]             any or all arguments within [ ] are optional.
 Manual page man(1) line 1 (press h for help or q to quit)
```

3. **uname -a**

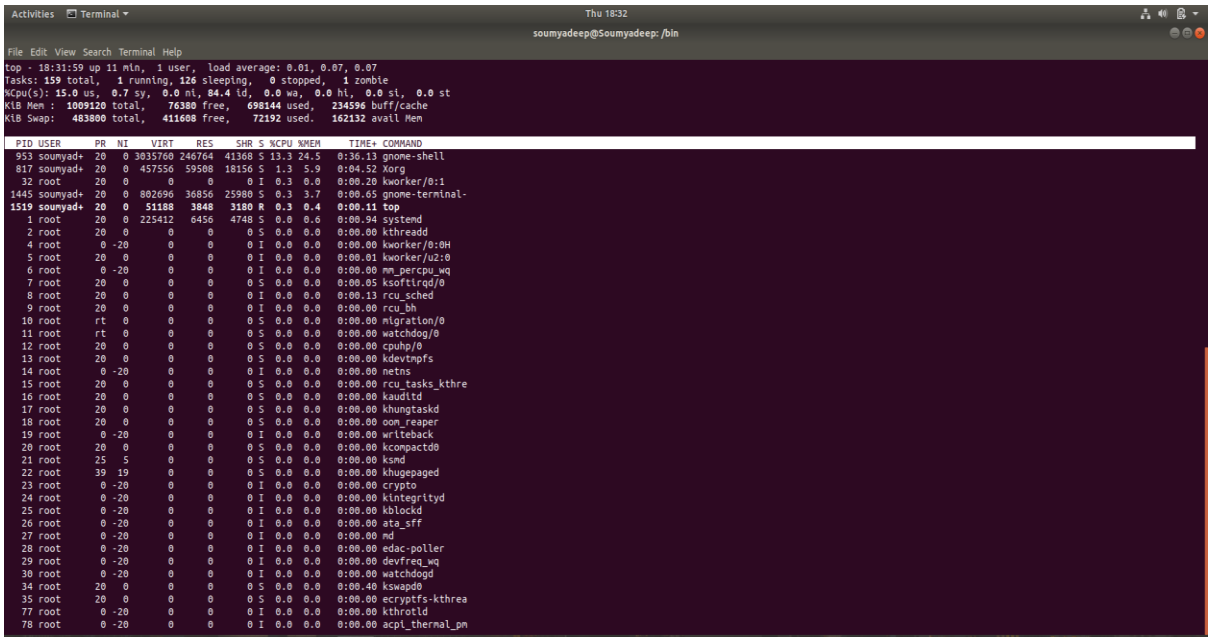
It shows the details about the Linux operating system. Kernel release, version and operating system are the details it mentions in the output.



```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

soumyadeep@Soumyadeep:~$ man man
soumyadeep@Soumyadeep:~$ uname -a
Linux Soumyadeep 4.15.0-34-generic #37-Ubuntu SMP Mon Aug 27 15:21:48 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux
soumyadeep@Soumyadeep:~$
```

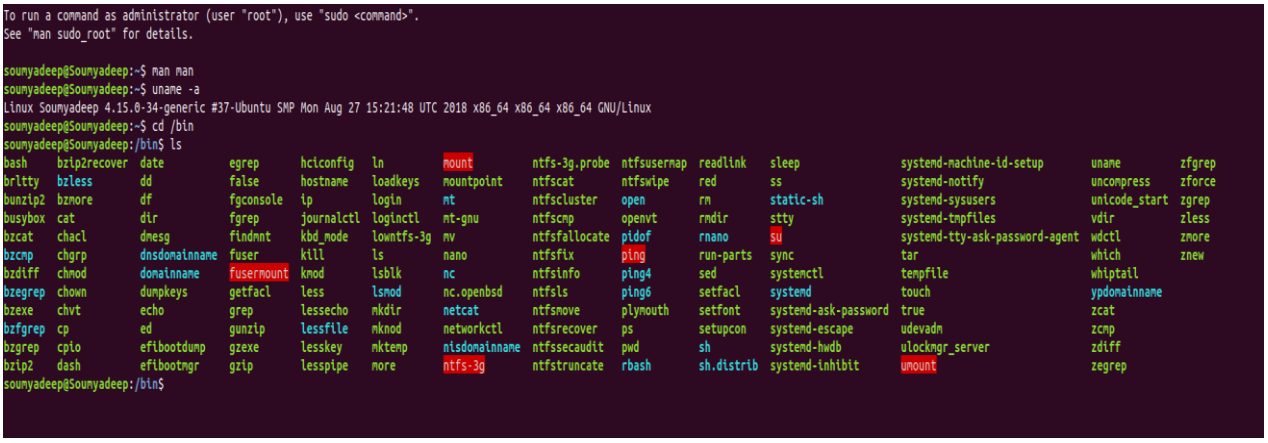
4. **top**

top provides the description about the system summary. It provides the list of processes running (Name and its Process Id(PID)), its CPU and memory usage. It also provides the overall system usage and load information helping in giving an overall state of the system.



5. **cd**

cd command allows us to change the current working directory. cd /bin allowed us to move to the bin directory under root.

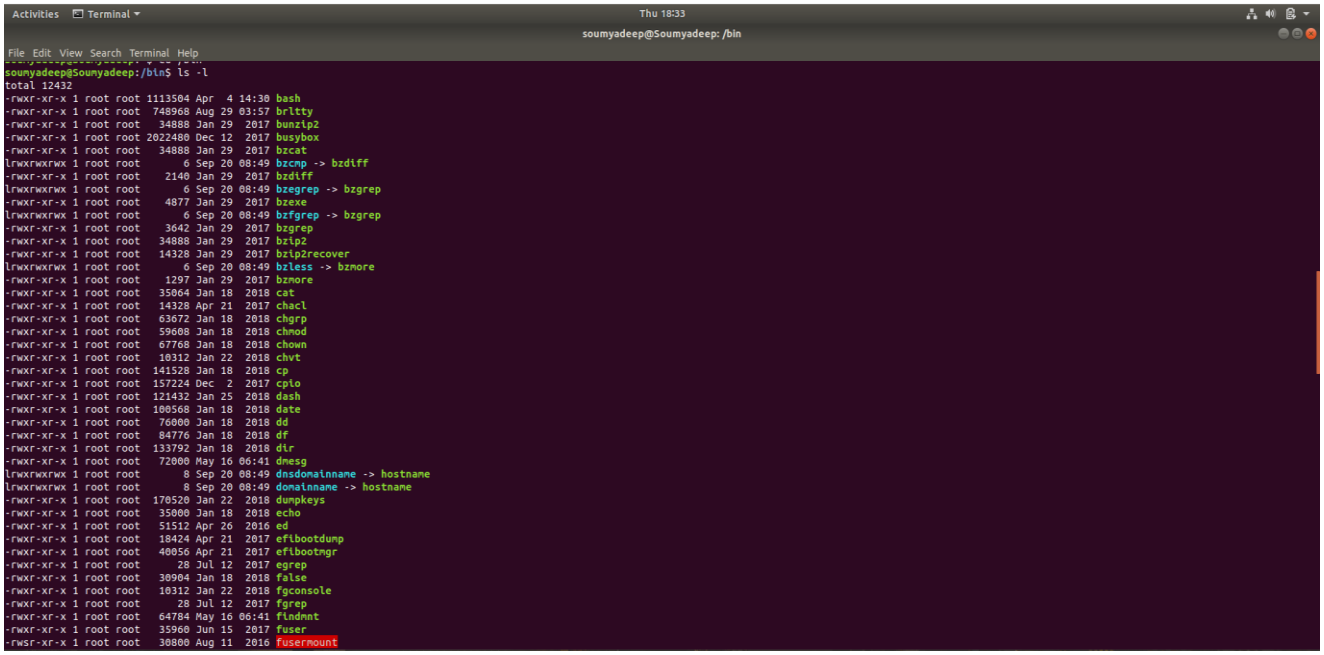


6. **ls -l**

ls lists the files and directories in the current or the specified directory. The -l option lists the files and directories in long format. It specifies '–' for files and 'd' for directories, 'l' for links, followed by its access permissions, the number of hard links to the file, the owner, group, block size, modification time and the name of the element.

Example:

"-rwxr-xr-x 3 root root 34480 Jan 29 2017 bunzip2"



7.

'|' - Pipe Operator – It passes the output of the first command on the left as input to the second command written on the right side of the operator.

Head command displays the first 10 lines of the input passed to it and head -n will display the first n number of lines of the input.

In the given example ls | head -1, the command lists the files on the directories and it is passed as the input to the head -1 which displays the first line from the list.

"ls > temp; head -1 < temp" also gives the same output. The output of ls is redirected to the 'temp' file. Then temp file is passed as input to head -1 command.

'>' and '<' are redirection operators. 'command > file' redirects the output of command to the file.

'**command < file**' passes the file as input to the command.

8. The command '**cd**' changes the current working directory based on the parameter passed on it.

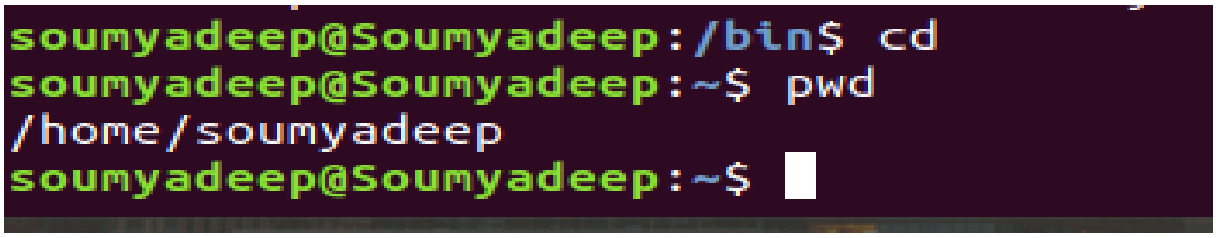
The command **'cd'** alone changes the current directory to the parent directory.

Example: cd in the screenshot changed the directory from bin to the home directory 'soumyadeep'.

pwd: This command displays the present working directory.

Example: In the screenshot, the pwd command displays the present directory 'soumyadeep'.

/home/soumyadeep

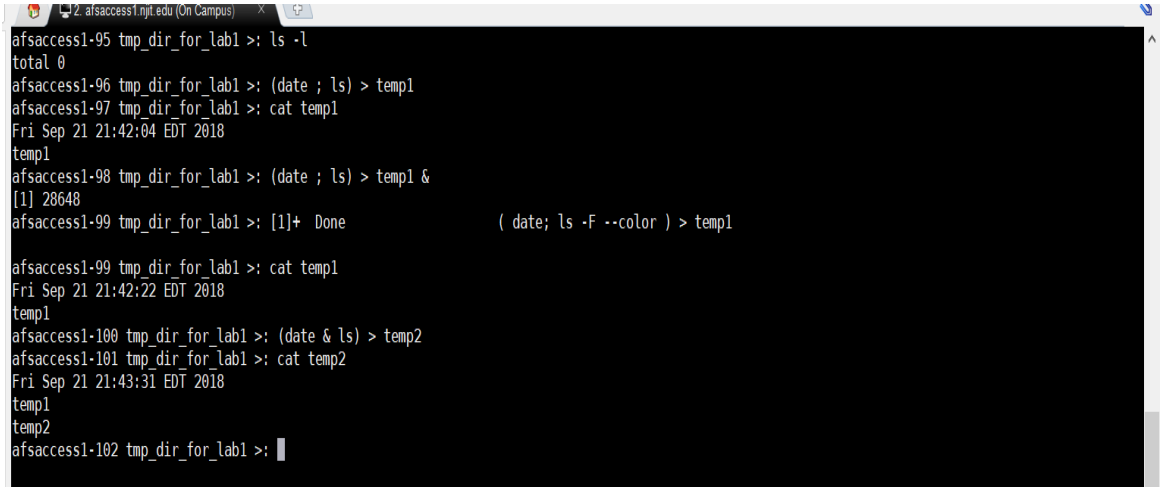


9. **mkdir:** This command is required to make a new directory in the present directory.

Example: **mkdir tmp_dir_for_lab1** makes a new directory tmp_dir_for_lab1 in ckd22 directory.

cd mkdir tmp_dir_for_lab1 changes the present directory from ckd22 to tmp_dir_for_lab1.

10.



(date ; ls) > temp1

The **';'** allows us to sequentially execute the commands one after the other.

'&' allows us to run the command in the background.

temp1 and temp2 are created differently but in both cases, the date output is concatenated with the directory listing. Thus, the date value is different based on what time it was run. Also, the listing is different in both based on the files created while ls was run.

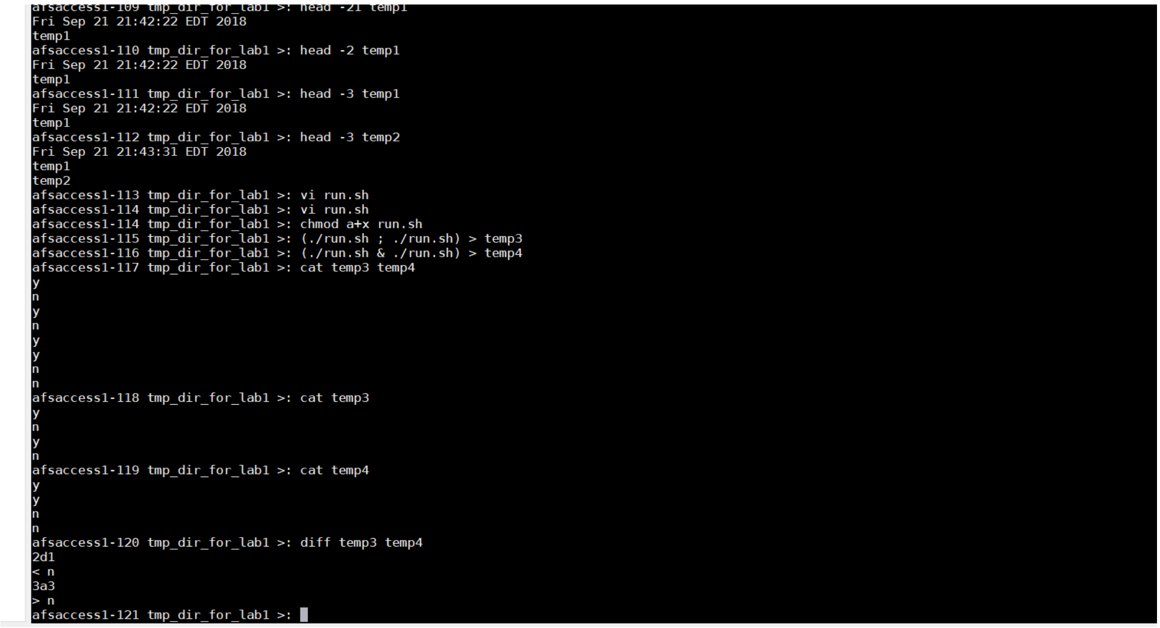
11.

```
#!/bin/sh

 echo y
 sleep 10
 echo n
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"run.sh" 5L, 39C
```

available only to teachers and students in classrooms or at home.

We created **run.sh script** that displays y, waits for 10 seconds and then display n.

12.



```
afsaccess1-109 tmp_dir_for_lab1 >: head -21 temp1
Fri Sep 21 21:42:22 EDT 2018
temp1
afsaccess1-110 tmp_dir_for_lab1 >: head -2 temp1
Fri Sep 21 21:42:22 EDT 2018
temp1
afsaccess1-111 tmp_dir_for_lab1 >: head -3 temp1
Fri Sep 21 21:42:22 EDT 2018
temp1
afsaccess1-112 tmp_dir_for_lab1 >: head -3 temp2
Fri Sep 21 21:43:31 EDT 2018
temp1
temp2
afsaccess1-113 tmp_dir_for_lab1 >: vi run.sh
afsaccess1-114 tmp_dir_for_lab1 >: vi run.sh
afsaccess1-114 tmp_dir_for_lab1 >: chmod a+x run.sh
afsaccess1-115 tmp_dir_for_lab1 >: (./run.sh ; ./run.sh) > temp3
afsaccess1-116 tmp_dir_for_lab1 >: (./run.sh & ./run.sh) > temp4
afsaccess1-117 tmp_dir_for_lab1 >: cat temp3 temp4
y
n
y
n
y
y
n
n
afsaccess1-118 tmp_dir_for_lab1 >: cat temp3
y
n
y
n
afsaccess1-119 tmp_dir_for_lab1 >: cat temp4
y
y
n
n
afsaccess1-120 tmp_dir_for_lab1 >: diff temp3 temp4
2d1
< n
3a3
> n
afsaccess1-121 tmp_dir_for_lab1 >: 
```

diff compares 2 files line by line and shows the difference between them. **diff file1 file2** gives us the changes required in file1 to make it similar to file2.

temp3 and temp4 are different as in the first case, the run.sh script is executed one after other since they are separated by semicolon. In the second case, both scripts are run simultaneously, the first script being run in the background and the second script in the foreground. Hence in the first case, we are getting output from the first script followed by the second script. In the second case, we are getting output from both scripts together.

13.

```
afsaccess1-123 tmp_dir_for_lab1 >: echo $PATH
/afs/cad.njit.edu/sw.common/matlab-2018a/bin:/afs/cad/linux/java10/bin:/afs/cad/linux/anaconda3.6/anaconda/bin:/bin:/afs/cad/sw.common/bin:/usr/site/
bin:/usr/ucs/bin:/usr/lib64/qt-3.3/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/afs/cad/linux/compilers/bin:/sbin
afsaccess1-124 tmp_dir_for_lab1 >:
```

$PATH is an environment variable that stores the paths of executable files. Each path is separated by a colon.

14.

```
2. afsaccess1.njit.edu (On Campus)
#include <stdio.h>

 int main( int argc, char* argv[] ){

         fprintf(stdout,"Hello, Linux!\n");

                 return 0; }
~
~
~
~
```

```
afsaccess1-125 tmp_dir_for_lab1 >: vi hello.c
afsaccess1-126 tmp_dir_for_lab1 >: gcc hello.c -o hello
1afsaccess1-127 tmp_dir_for_lab1 >: ./hello
Hello, Linux!
afsaccess1-128 tmp_dir_for_lab1 >: vi hello.c
afsaccess1-129 tmp_dir_for_lab1 >:
```

‘#include <stdio.h>’:

‘stdio.h’ is a library header file. It includes the standard input and output functions. Include is written to add the library in the c program so that the related functions can be used in it.

‘fprintf’ – is a library function to print the command line output to the specified file or device.

‘stdout’ – is the pointer to the file system which here is used in fprintf. It prints to the default output device.

15.

cp temp2 temp 5: cp command copies the content of file temp2 to new file temp5.

16. **rm temp*** is used to remove all the files starting with temp in the directory.