

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] [-l-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]] [-M[browser]] [-X[dpl]] [-Z] [[section] page[.section] ...] ...
  man -k [apropos options] regex ...
  man -K [-w-W] [-S list] [-l-I] [--regex] [section] term ...
  man -f [subatts 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]] [-M[browser]] [-X[dpl]] [-Z] file ...
  man -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 libraries)
  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 [abc] are optional.

Manual page man(1) line 1 (press h for help or q to quit)
```

3. uname -a

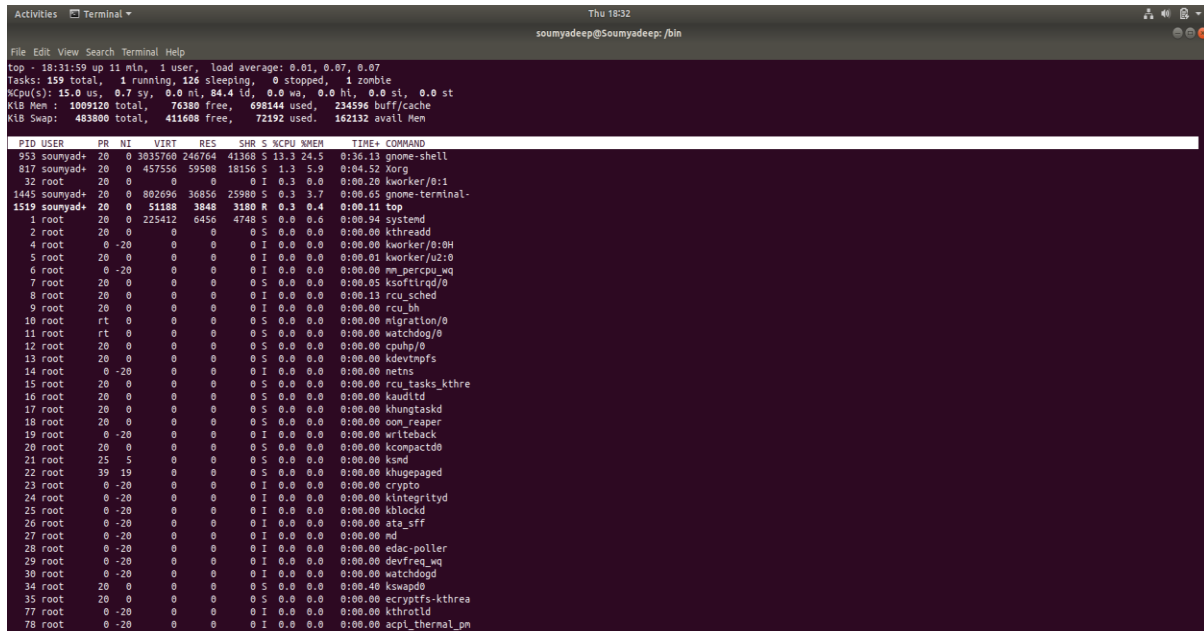
It shows the details about the Linux operating system. Kernel release, version and operating system are the details it mentions in the output.

```
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”

```
Activities Terminal Thu 18:33 soumyadeep@Soumyadeep: /bin
File Edit View Search Terminal Help
soumyadeep@Soumyadeep: /bin$ ls -l
total 12432
-rwxr-xr-x 1 root root 1113584 Apr 4 14:30 bash
-rwxr-xr-x 1 root root 748968 Aug 29 03:57 brltty
-rwxr-xr-x 1 root root 34888 Jan 29 2017 bunzip2
-rwxr-xr-x 1 root root 2022488 Dec 12 2017 busybox
-rwxr-xr-x 1 root root 34888 Jan 29 2017 bzip
-rwxr-xr-x 1 root root 6 Sep 20 08:49 bzip -> bzipdiff
-rwxr-xr-x 1 root root 2140 Jan 29 2017 bzipdiff
-rwxr-xr-x 1 root root 6 Sep 20 08:49 bzipgrep -> bzgrep
-rwxr-xr-x 1 root root 4877 Jan 29 2017 bzipgrep
-rwxr-xr-x 1 root root 6 Sep 20 08:49 bzipgrep -> bzgrep
-rwxr-xr-x 1 root root 3642 Jan 29 2017 bzipgrep
-rwxr-xr-x 1 root root 34888 Jan 29 2017 bzip2
-rwxr-xr-x 1 root root 14328 Jan 29 2017 bzip2recover
-rwxr-xr-x 1 root root 6 Sep 20 08:49 bzless -> bzmore
-rwxr-xr-x 1 root root 1297 Jan 29 2017 bzmore
-rwxr-xr-x 1 root root 35064 Jan 18 2018 cat
-rwxr-xr-x 1 root root 14328 Apr 21 2017 chacl
-rwxr-xr-x 1 root root 63672 Jan 18 2018 chgrp
-rwxr-xr-x 1 root root 59608 Jan 18 2018 chmod
-rwxr-xr-x 1 root root 67768 Jan 18 2018 chown
-rwxr-xr-x 1 root root 10312 Jan 22 2018 chvt
-rwxr-xr-x 1 root root 141528 Jan 18 2018 cp
-rwxr-xr-x 1 root root 157224 Dec 2 2017 cpio
-rwxr-xr-x 1 root root 121432 Jan 25 2018 dash
-rwxr-xr-x 1 root root 190568 Jan 18 2018 date
-rwxr-xr-x 1 root root 76608 Jan 18 2018 dd
-rwxr-xr-x 1 root root 84776 Jan 18 2018 df
-rwxr-xr-x 1 root root 133792 Jan 18 2018 dir
-rwxr-xr-x 1 root root 72008 May 16 06:41 dmesg
-rwxr-xr-x 1 root root 8 Sep 20 08:49 dnsdomainname -> hostname
-rwxr-xr-x 1 root root 8 Sep 20 08:49 domainname -> hostname
-rwxr-xr-x 1 root root 170520 Jan 22 2018 dumpleys
-rwxr-xr-x 1 root root 35008 Jan 18 2018 echo
-rwxr-xr-x 1 root root 51512 Apr 26 2016 ed
-rwxr-xr-x 1 root root 18424 Apr 21 2017 efibootdump
-rwxr-xr-x 1 root root 40856 Apr 21 2017 efibootmgr
-rwxr-xr-x 1 root root 28 Jul 12 2017 egrep
-rwxr-xr-x 1 root root 30904 Jan 18 2018 false
-rwxr-xr-x 1 root root 10312 Jan 22 2018 fgconsole
-rwxr-xr-x 1 root root 28 Jul 12 2017 fgrep
-rwxr-xr-x 1 root root 64784 May 16 06:41 findmnt
-rwxr-xr-x 1 root root 95960 Jun 15 2017 fuser
-rwxr-xr-x 1 root root 30800 Aug 11 2016 fusermount
```

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.

```
afssaccess1-182 ~ >: ls | head -1
ARCS.README@
afssaccess1-183 ~ >: ls > temp; head -1 < temp
ARCS.README@
afssaccess1-184 ~ >: ls
ARCS.README@ BACKUP-AFS.ACCOUNT/ bnfo601/ f18.bnfo.615.001@ IHLp656/ unv13e/ temp unvp13e/ unvp13e.tar.gz zz
afssaccess1-185 ~ >:
```

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

```
soumyadeep@Soumyadeep:/bin$ cd
soumyadeep@Soumyadeep:~$ pwd
/home/soumyadeep
soumyadeep@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**.

```
afsaccess1-67 ~ >: mkdir tmp_dir_for_lab1
afsaccess1-68 ~ >: cd tmp_dir_for_lab1
afsaccess1-69 tmp_dir_for_lab1 >: pwd
/home/cad/u/c/k/ckd22/tmp_dir_for_lab1
afsaccess1-70 tmp_dir_for_lab1 >:
```

10.

```
afsaccess1-95 tmp_dir_for_lab1 >: ls -l
total 0
afsaccess1-96 tmp_dir_for_lab1 >: (date ; ls) > temp1
afsaccess1-97 tmp_dir_for_lab1 >: cat temp1
Fri Sep 21 21:42:04 EDT 2018
temp1
afsaccess1-98 tmp_dir_for_lab1 >: (date ; ls) > temp1 &
[1] 28648
afsaccess1-99 tmp_dir_for_lab1 >: [1]+ Done          (date; ls -F --color) > temp1

afsaccess1-99 tmp_dir_for_lab1 >: cat temp1
Fri Sep 21 21:42:22 EDT 2018
temp1
afsaccess1-100 tmp_dir_for_lab1 >: (date & ls) > temp2
afsaccess1-101 tmp_dir_for_lab1 >: cat temp2
Fri Sep 21 21:43:31 EDT 2018
temp1
temp2
afsaccess1-102 tmp_dir_for_lab1 >:
```

(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
y
n
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.

```
afscad@afscad1-123 tmp_dir_for_lab1:~$ echo $PATH
/afs/cad.njit.edu/sw.common/matlab-2018a/bin:/afs/cad/linux/java10/bin:/afs/cad/linux/anaconda3.6/anaconda/bin:/bin:/afs/cad/sw.common/bin:/usr/site/bin:/usr/ucs/bin:/usr/lib64/qt-3.3/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/afs/cad/linux/compiler/bin:/sbin
afscad@afscad1-124 tmp_dir_for_lab1:~$
```

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

14.

```
#include <stdio.h>

int main( int argc, char* argv[] ){
    fprintf(stdout,"Hello, Linux!\n");
    return 0; }
```

```
afscad@afscad1-125 tmp_dir_for_lab1:~$ vi hello.c
afscad@afscad1-126 tmp_dir_for_lab1:~$ gcc hello.c -o hello
afscad@afscad1-127 tmp_dir_for_lab1:~$ ./hello
Hello, Linux!
afscad@afscad1-128 tmp_dir_for_lab1:~$ vi hello.c
afscad@afscad1-129 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.

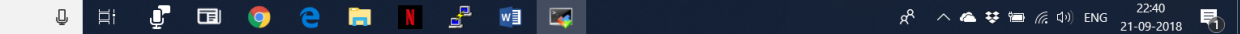
‘fprintf’ – is a library function to print the command line output to the specified file or device.

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

15.

```
afsaccess1-129 tmp_dir_for_lab1 >: cp temp2 temp5
afsaccess1-130 tmp_dir_for_lab1 >: cat temp5
Fri Sep 21 21:43:31 EDT 2018
temp1
temp2
afsaccess1-131 tmp_dir_for_lab1 >: cat temp 2
cat: temp: No such file or directory
cat: 2: No such file or directory
afsaccess1-132 tmp_dir_for_lab1 >: cat temp2
Fri Sep 21 21:43:31 EDT 2018
temp1
temp2
afsaccess1-133 tmp_dir_for_lab1 >: 
```

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



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.

```
2. afsaccess1.njit.edu (On Campus) x
afsaccess1-175 tmp_dir_for_lab1 >: ls
hello* hello.c run.sh* temp1 temp2 temp3 temp5
afsaccess1-176 tmp_dir_for_lab1 >: rm temp*
afsaccess1-177 tmp_dir_for_lab1 >: ls
hello* hello.c run.sh*
afsaccess1-178 tmp_dir_for_lab1 >: cd ..
afsaccess1-179 ~ >: rm -r tmp_dir_for_lab1
afsaccess1-180 ~ >: ls
ARCS.README@ BACKUP-AFS.ACCOUNT/ bnfo601/ f18.bnfo.615.001@ IHLP656/ public_html/ unpv13e/ unpv13e.tar.gz zz
afsaccess1-181 ~ >: 
```