

File Specific commands

cat - concatenate files and print on screen

cat command displays the content of the file on standard output.

```
emertxe@DESKTOP-VKPN0S4:~/Test$ cat file_script.sh
#!/bin/bash

read -p "Enter 2 numbers" num1 num2
if [ $num1 -eq $num2 ]
then
    echo $num1 is equal to $num2
elif [ $num1 -lt $num2 ]
then
    echo $num2 is greater
else
    echo $num1 is greater
fi
```

Figure 1a Usage of cat command

```
emertxe@DESKTOP-VKPN0S4:~/Test$ cat
hello
hello
how are you
how are you
emertxe@DESKTOP-VKPN0S4:~/Test$
```

Figure 1b Usage of cat command

cat command without any argument reads input from stdin and when enter is pressed ,displays the content in stdout as shown in the above figure.

more - to view the content page wise

more command helps us to view the content page wise. If the file size is greater than the window size , more command help to view the content page wise.Use the enter key to scroll to the next page. Press q to quit.

more <file_name>

.

--More-- (23%)

Figure 2a Output of more command

More command can also be used to view more than one file at a time. Contents of files will be displayed with the file name as shown below.

```
emertxe@ubuntu:~/Dir1/demo/Test$
```

Figure 2b more command to view more than one file

less - to view the content with page wise

Less command is similar to more command with many features. Use enter to view page by page and use arrow keys to scroll up and down.press q to quit.

less <filename>

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
/etc/passwd
```

Figure 3a Output of less command

To view multiple files with less command

less <file1> <file2> ..

To switch between the files

:n - next file

:p - previous files

Ctrl + G -shows current file name along with line , byte and percentage statistics.

Wc command without any argument passed ,reads input from stdin and displays the counts as shown in the above figure.

head- outputs the first part

head helps us to output the first part of the file content.

```
emertxe@DESKTOP-VKPN0S4:~/Test$ head file_script.sh
#!/bin/bash

read -p "Enter 2 numbers" num1 num2
if [ $num1 -eq $num2 ]
then
    echo $num1 is equal to $num2
elif [ $num1 -lt $num2 ]
then
    echo $num2 is greater
else
emertxe@DESKTOP-VKPN0S4:~/Test$
```

Figure 5a output of head command

usage :

head <filename> - displays 1st 10 lines from the file by default

head -n <filename> - display 1st n lines from the file

head -cn <filename> -displays 1st n character from the file

```
emertxe@ubuntu:~$ head -c20 file_script.sh
#!/bin/bash

read -p emertxe@ubuntu:~$ head -c10 file_script.sh
#!/bin/basemertxe@ubuntu:~$ █
```

Figure 5b Output of head with -c option

tail -outputs the last part

tail helps us to output the last part of the file content.

```
emertxe@DESKTOP-VKPN0S4:~/Test$ tail file_script.sh
read -p "Enter 2 numbers" num1 num2
if [ $num1 -eq $num2 ]
then
    echo $num1 is equal to $num2
elif [ $num1 -lt $num2 ]
then
    echo $num2 is greater
else
    echo $num1 is greater
fi
emertxe@DESKTOP-VKPN0S4:~/Test$
```

Figure 6a output of tail command

usage:

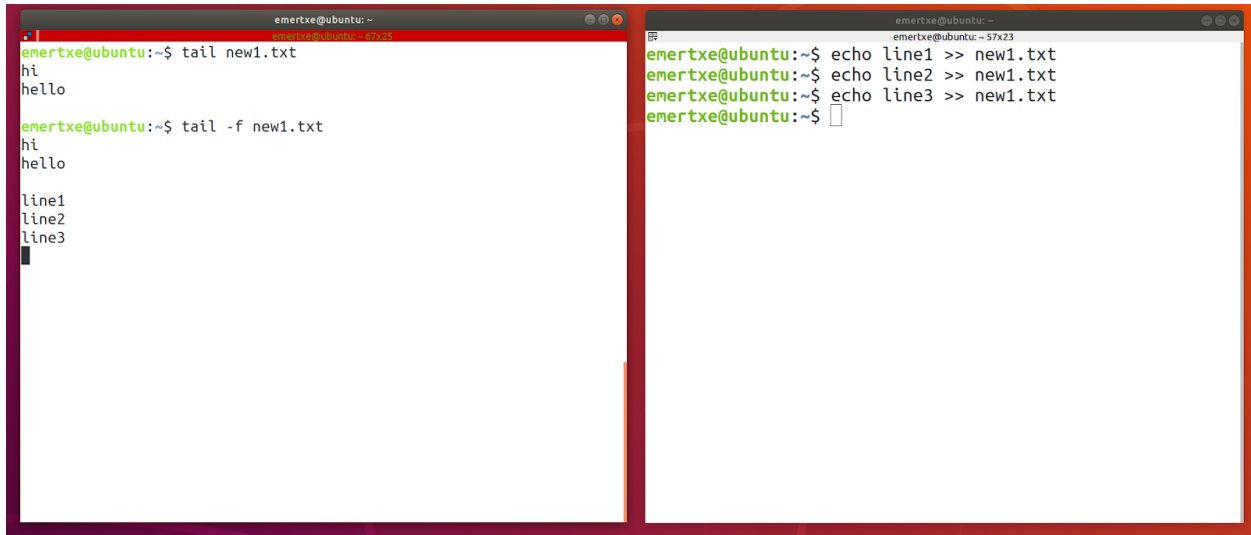
tail <filename> - displays last 10 lines from the file
tail -n <filename> - display last n lines from the file
tail -cn <filename> - displays last n character from the file

```
emertxe@ubuntu:~$ tail -c10 file_script.sh
$num2
fi
emertxe@ubuntu:~$ tail -c20 file_script.sh
number is $num2
fi
emertxe@ubuntu:~$ tail -c5 file_script.sh
2
fi
emertxe@ubuntu:~$ █
```

Figure 6b Output of tail with -c option

tail -f <filename> - prints the last 10 lines from the file and also prints new content whenever it is been appended to the file. This is called as following a file. tail command with -f command follows the file.

Ctrl + c to stop following a file.

The image shows two terminal windows side-by-side. The left window shows the command 'tail new1.txt' being executed, displaying the contents of the file: 'hi', 'hello', 'line1', 'line2', and 'line3'. The right window shows the command 'tail -f new1.txt' being executed, which follows the file as it grows. It shows the same initial output, but then it shows 'line1', 'line2', and 'line3' being added to the file, demonstrating the follow functionality.

```
emertxe@ubuntu:~$ tail new1.txt
hi
hello

emertxe@ubuntu:~$ tail -f new1.txt
hi
hello

line1
line2
line3
```

```
emertxe@ubuntu:~$ echo line1 >> new1.txt
emertxe@ubuntu:~$ echo line2 >> new1.txt
emertxe@ubuntu:~$ echo line3 >> new1.txt
emertxe@ubuntu:~$
```

Figure 6c tail command with follow option

Redirection

Redirection is an important feature in Linux that allows to redirect data to/from a file.

There are 2 types of redirection

1. Output redirection (> denotes redirection)
2. Input redirection (In-direction) (< denotes indirection)

1. Output redirection

Redirects the output of a command to a file instead of displaying the output on standard out.

We can also redirect the error to file. That's called error redirection.

We can control the data stream which is redirected to file by using file descriptors.

Operator used for redirection

1. 1> or > redirects from stdout to a file (data overwritten in file)
2. 2> redirects from stderr to a file (data overwritten in file)
3. 1>> or >> redirects from stdout to a file (data appended in file)
4. 2>> redirects from stderr to a file (data appended in file)
5. &> redirects from stdout and stderr to a file (data overwritten in file)
6. &>> redirects from stdout and stderr to a file (data appended in file)

7. `1>&2` redirects output to the file where error has been redirected
8. `2>&1` redirects error to the file where output has been redirected

```
emertxe@DESKTOP-VKPN0S4:~/Test$ echo hello > out.txt
emertxe@DESKTOP-VKPN0S4:~/Test$ cat out.txt
hello
emertxe@DESKTOP-VKPN0S4:~/Test$ ls > out.txt
emertxe@DESKTOP-VKPN0S4:~/Test$ cat out.txt
Dir2
file.txt
file2
file_script.sh
out.txt
recursion.sh
emertxe@DESKTOP-VKPN0S4:~/Test$ echo hello >> out.txt
emertxe@DESKTOP-VKPN0S4:~/Test$ cat out.txt
Dir2
file.txt
file2
file_script.sh
out.txt
recursion.sh
hello
emertxe@DESKTOP-VKPN0S4:~/Test$
```

Figure 7 Output redirection

```
emertxe@DESKTOP-VKPN0S4:~/Test$ ls Test
ls: cannot access 'Test': No such file or directory
emertxe@DESKTOP-VKPN0S4:~/Test$
emertxe@DESKTOP-VKPN0S4:~/Test$ ls Test 2> err.txt
emertxe@DESKTOP-VKPN0S4:~/Test$
emertxe@DESKTOP-VKPN0S4:~/Test$ cat err.txt
ls: cannot access 'Test': No such file or directory
emertxe@DESKTOP-VKPN0S4:~/Test$
emertxe@DESKTOP-VKPN0S4:~/Test$
```

Figure 8 Error redirection

In the below figure , both output and error is redirected in a single command


```

emertxe@DESKTOP-VKPN0S4:~/Test$ ls Test Dir2
ls: cannot access 'Test': No such file or directory
Dir2:
New file1 file2
emertxe@DESKTOP-VKPN0S4:~/Test$ ls Test Dir2 > out.txt 2> err.txt
emertxe@DESKTOP-VKPN0S4:~/Test$
emertxe@DESKTOP-VKPN0S4:~/Test$ cat out.txt
Dir2:
New
file1
file2
emertxe@DESKTOP-VKPN0S4:~/Test$ cat err.txt
ls: cannot access 'Test': No such file or directory
emertxe@DESKTOP-VKPN0S4:~/Test$
emertxe@DESKTOP-VKPN0S4:~/Test$

```

Figure 9 Output and error redirection to different files

To redirect both output and error to the same file use `&>`, as shown in below figure.

```

emertxe@DESKTOP-VKPN0S4:~/Test$ ls Test Dir2
ls: cannot access 'Test': No such file or directory
Dir2:
New file1 file2
emertxe@DESKTOP-VKPN0S4:~/Test$ ls Test Dir2 &> redirect.txt
emertxe@DESKTOP-VKPN0S4:~/Test$
emertxe@DESKTOP-VKPN0S4:~/Test$ cat redirect.txt
ls: cannot access 'Test': No such file or directory
Dir2:
New
file1
file2
emertxe@DESKTOP-VKPN0S4:~/Test$

```

Figure 10 Redirect both output and error to same file

To redirect the error message to the same file where output is been redirected

```

emertxe@ubuntu:~/Dir1/demo/Test$ ls New abc
ls: cannot access 'abc': No such file or directory
New:
file1 file2
emertxe@ubuntu:~/Dir1/demo/Test$ ls New abc > out.txt 2>&1
emertxe@ubuntu:~/Dir1/demo/Test$ cat out.txt
ls: cannot access 'abc': No such file or directory
New:
file1
file2
emertxe@ubuntu:~/Dir1/demo/Test$ █

```

Figure 10a Redirect error message to the file where output is been redirected

To redirect the output to the file where the error is being redirected.

```
emertxe@ubuntu:~/Dir1/demo/Test$ ls New abc 2> err.txt 1>&2
emertxe@ubuntu:~/Dir1/demo/Test$
emertxe@ubuntu:~/Dir1/demo/Test$ cat err.txt
ls: cannot access 'abc': No such file or directory
New:
file1
file2
emertxe@ubuntu:~/Dir1/demo/Test$ █
```

Figure 10b Redirect output to the file where error has been redirected.

2.Input redirection

Input redirection is to redirect the data from file to command .

1. < from a file to a command

```
emertxe@DESKTOP-VKPN0S4:~/Test$ wc < file_script.sh
12  38 191
```

Figure 11 Input redirection

Piping

Pipe is a form of redirection that is used in the linux operating system to send the output of one command or executable program to another command for further processing. Pipe is denoted by vertical bar character |

Operator used for piping

Command 1 | command 2 | command 3 |

Command1 output is given as input to command 2 and command 2 output is given as an input to command 3 and so on.

Note:

The command on the left hand side of the pipe operator should give output and the command on the right hand side should take input.

```
emertxe@DESKTOP-VKPN0S4:~/Test$ ls | head -5
Dir2
err.txt
file.txt
file2
file_script.sh
emertxe@DESKTOP-VKPN0S4:~/Test$ ls | head -5 | tail -1
file_script.sh
emertxe@DESKTOP-VKPN0S4:~/Test$ ls | wc
      8      8     77
emertxe@DESKTOP-VKPN0S4:~/Test$
```

Figure 12 Usage of pipe

In the above figure, output of ls command is piped as a input to head and wc command

Command substitution methods

command substitution is to be used wherever commands are not on the start of a command line but output of the command needs to be used.

We can also store the output of a command to a variable by using below methods.

- 1.using backtick
- 2.using \$()

Usage:

variable-name=`command`

Or

variable-name=\$(command)

Example:

var=`ls`

Or

var=\$(ls)

```

emertxe@ubuntu:~$ var=`date`
emertxe@ubuntu:~$
emertxe@ubuntu:~$ echo $var
Thu Mar 24 13:20:03 IST 2022
emertxe@ubuntu:~$
emertxe@ubuntu:~$ echo "Today's date is $var"
Today's date is Thu Mar 24 13:20:03 IST 2022
emertxe@ubuntu:~$

```

Figure 13 Command Substitution

df - displays filesystem space usage information

df command is used for displaying the information of the file system present in the system. Output of the df varies from system to system.

```

emertxe@DESKTOP-VKPN0S4:~/Test$ df

```

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
rootfs	108521468	37111068	71410400	35%	/
none	108521468	37111068	71410400	35%	/dev
none	108521468	37111068	71410400	35%	/run
none	108521468	37111068	71410400	35%	/run/lock
none	108521468	37111068	71410400	35%	/run/shm
none	108521468	37111068	71410400	35%	/run/user
tmpfs	108521468	37111068	71410400	35%	/sys/fs/cgroup
C:\	108521468	37111068	71410400	35%	/mnt/c
E:\	101134332	125404	101008928	1%	/mnt/e
F:\	102399996	98184	102301812	1%	/mnt/f

Figure 14 Output of df command

du - disk usage

du command displays the disk space usage information. Usually to check the size of larger files or directory.

du -h - h option displays the output in human readable format

du -sh - displays the summarized size in human readable format

```
emertxe@DESKTOP-VKPN0S4:~$ du -h
0      ./Dir1
0      ./Test/Dir2/New
0      ./Test/Dir2
0      ./Test
28K    .
emertxe@DESKTOP-VKPN0S4:~$
```

Figure 15. Output of du command

cmp - compares files byte by byte

cmp command compares files byte by byte.

```
emertxe@DESKTOP-VKPN0S4:~$ cat file1.txt
apple
orange
emertxe@DESKTOP-VKPN0S4:~$ cat file2.txt
apple
mango
orang
helo
hoe
```

Figure 16 Contents of file1.txt and file2.txt

```
emertxe@DESKTOP-VKPN0S4:~$ cmp file1.txt file2.txt
file1.txt file2.txt differ: byte 7, line 2
```

Figure 17 Output of cmp command

diff -compares files line by line

Compare two files line by line and print the difference as output.

From the below example, 2nd line of file1.txt is different from 2-5 lines of file2.txt

```
emertxe@DESKTOP-VKPN0S4:~$ diff file1.txt file2.txt
2c2,5
< orange
---
> mango
> orang
> helo
> hoe
```

Figure 18 Output of diff command

sort - sort lines of text

```
emertxe@ubuntu:~/Dir1/demo$ sort fruits.txt
apple
apple
bacle
blueberry
lemon
lemon
Lemon
orange
orange
Orange
pinebacle
stawberry
watermelon
```

Figure 19 Sort the lines of file in alphabetical order

```
emertxe@ubuntu:~/Dir1/demo$ sort -r fruits.txt
watermelon
stawberry
pinebacle
Orange
orange
orange
Lemon
lemon
lemon
blueberry
bacle
apple
apple
```

Figure 20 Sort the lines of the file in reverse order

```
emertxe@ubuntu:~/Dir1/demo$ sort -n numbers.txt
1
2
3
23
23
45
345
45768
```

Figure 21 Sort with -n arranges numeric contents of file in ascending order

```
emertxe@ubuntu:~/Dir1/demo$ sort -nr numbers.txt
45768
345
45
23
23
3
2
1
```

Figure 22 Sort with -nr arranges numeric contents of file in descending order

uniq-to omit or report repeated lines

The uniq command is to omit adjacent duplicate lines in a file. If the repeated lines are not adjacent those lines will not be omitted. To get all the repeated lines an adjacent sort command can be used. The output of sort command can be piped to uniq command to omit all repeated lines as shown in below figure

```
emertxe@ubuntu:~/Dir1/demo$ sort fruits.txt | uniq
apple
bacle
blueberry
lemon
Lemon
orange
Orange
pinebacle
stawberry
watermelon
```

Figure 23 To omit all repeated lines

```
emertxe@ubuntu:~/Dir1/demo$ sort fruits.txt | uniq -i
apple
bacle
blueberry
lemon
orange
pinebacle
stawberry
watermelon
```

Figure 24 omit repeated lines (-i option to consider case insensitive duplicate lines)

`uniq -D filename` - print all duplicate lines

`uniq -c filename` - print the number of occurrence of the line

Note: Both `sort` and `uniq` command don't modify the original file content. The modified content is only displayed in stdout.

find - to search for files in a directory hierarchy

Find command search for files and directories with given criteria.

Usage:

`find <where> <criteria> <what to do>`

`<where>` - the path where we need to search for

`<criteria>` - at what criteria we need to search

`<what to do>` - what needs to be done after finding the files which is matching the given criteria

(example, print or delete or copy) By default the results of the `find` command will be printed.

`find . -empty` -finds all empty directories and files in current working directory

`find . ! -empty` -finds all non-empty directories and files in current working directory

`find . -empty -type f` -finds all empty files in current working directory

`find . -empty -type d` -finds all empty directory in current working directory

`find . -name "file*"` -finds all file and directory whose names starting with the pattern "file"


```
emertxe@ubuntu:~/Dir1/demo/test$ ls
Dir1 file file1.txt file2.txt new.txt
emertxe@ubuntu:~/Dir1/demo/test$
emertxe@ubuntu:~/Dir1/demo/test$ find . -name "file*"
./file1.txt
./file
./file2.txt
```

Figure 25 find command to search for files and directories with given pattern

find . -type f -name "file*" -finds all file whose names starting with the pattern "file"

find . -type d -name "file*" -finds all directories whose names starting with the pattern "file"

```
emertxe@ubuntu:~/Dir1/demo/test$ find . -type d -name "file*"
./file
emertxe@ubuntu:~/Dir1/demo/test$ find . -type f -name "file*"
./file1.txt
./file2.txt
```

Figure 26 find command with -type option to search for particular file type

find . -type f -iname "file*" -finds all file whose names starting with the pattern "file" (case insensitive pattern)

grep - to print lines matching a pattern

Grep search for particular pattern in a given file and prints the entire line

```
emertxe@ubuntu:~/Dir1/demo$ grep apple fruits.txt
apple
abc apple
pineapple
apple
```

Figure 27 grep command to search for a pattern "apple" in a file

```
emertxe@ubuntu:~/Dir1/demo$ grep -i apple fruits.txt
apple
Apple
abc apple
pineapple
apple
```

Figure 28 grep command with -i option for case -insensitive search

```
emertxe@ubuntu:~/Dir1/demo$ grep -e apple -e hi fruits.txt
apple
apple
hello hi
hi hi
hi
hi Apple
orange hi hello
apple
pineapple
emertxe@ubuntu:~/Dir1/demo$
```

Figure 29 grep command to perform multiple pattern search in a file

```
emertxe@ubuntu:~/Dir1/demo$ grep -x Apple fruits.txt
Apple
```

Figure 30 grep command with -x option for extract pattern search

ls | grep "file" - search for files in current working directory with given pattern

grep "pattern" . - search for given pattern in all the files in current working directory

grep -r "pattern" . - search for a given pattern in all the files in the current working directory and its sub directories recursively.

grep -e <pattern1> -e <pattern2> - search for pattern1 and pattern2 in a file

tr -translate or delete characters

tr command is used to translate or delete the characters from given input.

```
emertxe@ubuntu:~/Dir1/demo$ echo hello | tr [a-z] [A-Z]
HELLO
emertxe@ubuntu:~/Dir1/demo$ echo HELlo | tr [:lower:] [:upper:]
HELLO
```

Figure 31 tr command to translate characters from lowercase to uppercase

```
emertxe@ubuntu:~/Dir1/demo$ echo HELlo 123 | tr " " "\n"
HELlo
123
```

Figure 32 tr command to translate space to newline

```
emertxe@ubuntu:~/Dir1/demo$ echo HELlo 123 | tr -d [:digit:]
HELlo
emertxe@ubuntu:~/Dir1/demo$
emertxe@ubuntu:~/Dir1/demo$ echo HELlo 123 | tr -cd [:digit:]
123emertxe@ubuntu:~/Dir1/demo$
emertxe@ubuntu:~/Dir1/demo$
emertxe@ubuntu:~/Dir1/demo$ echo HELlo 123 | tr -d H
Ello 123
```

Figure 33 tr command with -d and -cd

tr -d command to delete a character or set of characters as shown in above figure.

```
emertxe@ubuntu:~/Dir1/demo$ echo hellllo | tr -s l
helo
```

Figure 34 tr command with -s to squeeze repeated characters and replace with single occurrence

cat file_name | tr [:upper:] [:lower:] - to translate all the characters in file from upper to lower

sed - stream editor for filtering and transforming text

Stream editor which allows us to edit the files from the command line. sed command takes input from file or input from pipe.

```
emertxe@ubuntu:~/Dir1/demo$ cat fruits.txt
apple
orange
pineapple
stawberry
Lemon
watermelon
```

Figure 35 content of fruits.txt file

```
emertxe@ubuntu:~/Dir1/demo$ sed 's/e/E/' fruits.txt
appleE
orangE
pinEapple
stawbErry
LEmon
watErmelon
emertxe@ubuntu:~/Dir1/demo$ sed 's/e/E/g' fruits.txt
appleE
orangE
pinEappleE
stawbErry
LEmon
watErMElon
```

Figure 36 sed command to substitute 'e' with 'E' in all lines of file

- sed 's/e/E/' fruits.txt - Substitutes 1st occurrence of character e with E in all lines
- sed 's/e/E/g' fruits.txt - Substitute all occurrence of character e with E in all lines
- sed '3s/e/E/g' fruits.txt - substitute all occurrence of character e with E in only 3rd line
- sed -i 's/apple/xyz/g' fruits.txt - substitute all occurrences of string apple with xyz in all lines permanently.

Note: sed command with -i option modifies the original file and the output will not be displayed on screen. sed command without -i option doesn't modify the file content rather applies the changes and prints the output on screen.

```
emertxe@ubuntu:~/Dir1/demo$ sed -n '4p' fruits.txt
stawberry
emertxe@ubuntu:~/Dir1/demo$ sed -n '2,4p' fruits.txt
orange
pineapple
stawberry
```

Figure 37. Sed command to print lines

sed '3p' <filename> - print 3rd line of the file along with other lines

sed -n '3p' <filename> -print 3rd line omitting all other lines

```
emertxe@ubuntu:~/Dir1/demo$ sed '2d' fruits.txt
apple
pineapple
stawberry
Lemon
watermelon
```

Figure 38 sed command to delete a line

sed 'nd' <filename> - delete nth line from the file and print other lines

sed '/apple/d' <filename> - delete all the lines which is having pattern "apple"

sed '2,4d' <filename> -delete 2 to 4 lines from a given file

cut - used to remove sections from each line of files

```
emertxe@ubuntu:~/Dir1/demo$ cut -c1-2 fruits.txt
ap
or
pi
st
Le
wa
emertxe@ubuntu:~/Dir1/demo$ cut -c1,4 fruits.txt
al
on
pe
sw
Lo
we
```

Figure 39 Usage of cut command with -c option

cut -c1 file -cuts 1st character from each line of the file

cut -c1-3 file -cuts 1st to 3rd character from each line of the file

cut -c1,4 file -cuts 1st and 4th character from each line of the file

```
emertxe@ubuntu:~/Dir1/demo$ echo red,green,yellow,orange | cut -d , -f 1
red
emertxe@ubuntu:~/Dir1/demo$ echo red,green,yellow,orange | cut -d , -f 2
green
emertxe@ubuntu:~/Dir1/demo$
```

Figure 40 Usage of cut command with -d option and -f

cut -d and -f option uses a character as a delimiter and splits the input and with -f option, we can print a particular field after splitting.

In the above figure ',' is used as a delimiter. Both -d and -f should be used together

Offset - To get range of character or elements from a variable or array

Usage:

`${<variable_name/array_name>:character position:Number of characters}`

var	=	1	2	3	a	b	c
Char position⇒		0	1	2	3	4	5

```
emertxe@ubuntu:~$ var=123abc
emertxe@ubuntu:~$
emertxe@ubuntu:~$ echo ${var:0:3}
123
emertxe@ubuntu:~$ echo ${var:2:3}
3ab
emertxe@ubuntu:~$ echo ${var: -1}
c
emertxe@ubuntu:~$ echo ${var:0:-1}
123ab
emertxe@ubuntu:~$ echo ${var:0:-2}
123a
emertxe@ubuntu:~$ █
```

Figure 41 To get range of characters from a variable

```

emertxe@ubuntu:~$ arr=(hello hi how are you)
emertxe@ubuntu:~$ echo ${arr[@]:0:4}
hello hi how are
emertxe@ubuntu:~$ echo ${arr[@]:1:2}
hi how
emertxe@ubuntu:~$ echo ${arr[@]:1:${#arr[@]}-1)}
hi how are you
emertxe@ubuntu:~$ echo ${arr[1]:0:1}
h
emertxe@ubuntu:~$ echo ${arr[0]:0:3}
hel
emertxe@ubuntu:~$ echo ${arr[0]:2:3}
llo
emertxe@ubuntu:~$

```

Figure 42 To get range of elements from an array

split - used to split a file into pieces

split command splits linux file into pieces.

```

emertxe@ubuntu:~/Dir1/demo$ mkdir Test
emertxe@ubuntu:~/Dir1/demo$
emertxe@ubuntu:~/Dir1/demo$ wc -l fruits.txt
6 fruits.txt
emertxe@ubuntu:~/Dir1/demo$ split -l 2 fruits.txt Test/
emertxe@ubuntu:~/Dir1/demo$
emertxe@ubuntu:~/Dir1/demo$ ls Test/
aa ab ac
emertxe@ubuntu:~/Dir1/demo$ █

```

Figure 43 Split command with -l option to split the file with specified lines

The above figure shows how to split the file into multiple files which have 2 lines in each splitted file.

split -b 2 file - split every 2 bytes from file into new file.

Note: By default new file names will be generated after splitting .

To combine all files back to a single file , redirection can be used as shown below

```
emertxe@ubuntu:~/Dir1/demo/Test$ ls
aa  ab  ac
emertxe@ubuntu:~/Dir1/demo/Test$ cat * > fruits.txt
emertxe@ubuntu:~/Dir1/demo/Test$
emertxe@ubuntu:~/Dir1/demo/Test$ cat fruits.txt
apple
orange
pineapple
strawberry
Lemon
watermelon
emertxe@ubuntu:~/Dir1/demo/Test$ wc -l fruits.txt
6 fruits.txt
emertxe@ubuntu:~/Dir1/demo/Test$ █
```

Figure 44 Combining all files back to a single file

File compression

Compression is needed to conserve the disk space

- When there is a need to send large files as an attachment via the email, it is good practice to compress first
- Compression & Decompression utilities - gzip & gunzip(.gz)
- The degree of compression depends on
 - The type of the file
 - Its size
 - Compression program used
- Example
 - Html files compress more
 - GIF & JPEG image files compress very less, as they are already in compressed form

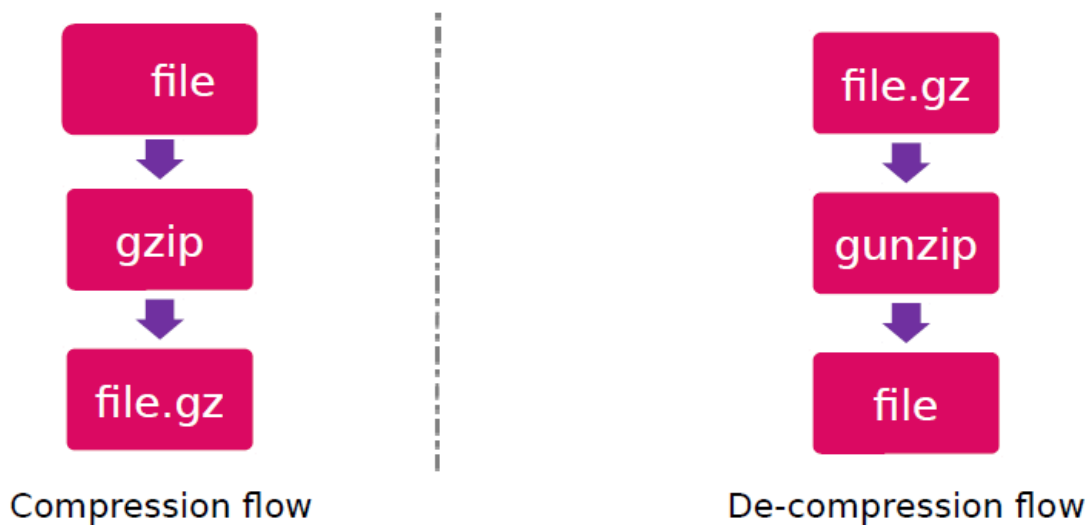


Figure 45 File compression and decompression flow

Command used to compress and decompress a file

`gzip filename`

`gunzip filename`

Recursive compression and decompression (-r option), for directory compression

- `gzip -r <directory>` : To compress files in whole directory
- `gunzip -r <directory>` : To de-compress files in whole directory

```
emertxe@ubuntu:~/Dir1/demo/Dir$ ls
new.txt
emertxe@ubuntu:~/Dir1/demo/Dir$ gzip new.txt
emertxe@ubuntu:~/Dir1/demo/Dir$ ls
new.txt.gz
emertxe@ubuntu:~/Dir1/demo/Dir$ gunzip new.txt.gz
emertxe@ubuntu:~/Dir1/demo/Dir$ ls
new.txt
emertxe@ubuntu:~/Dir1/demo/Dir$
```

Figure 46 Compress and decompress a file

File Archival

File Archival

- Used for creating disk archive that contains a group of files or an entire directory structure
- An archive file is a collection of files and directories that are stored in one file
- Archive file is not compressed, it uses the same amount of disk space as all the individual files and directories
- In case of compression, compressed file occupies less space
- Combination of archival & compression also can be done



Figure 47 File archival flow

Commands used for file archival

- `tar -cvf <archive name> <file-names>`
- `tar -xvf <path of archive name>`

```
emertxe@ubuntu:~/Dir1/demo/Dir$ ls
new.txt
emertxe@ubuntu:~/Dir1/demo/Dir$ gzip new.txt
emertxe@ubuntu:~/Dir1/demo/Dir$ ls
new.txt.gz
emertxe@ubuntu:~/Dir1/demo/Dir$ tar -cvf new.tar new.txt.gz
new.txt.gz
emertxe@ubuntu:~/Dir1/demo/Dir$ ls
new.tar  new.txt.gz
```

Figure 48 create a tar file

```
emertxe@ubuntu:~/Dir1/demo/Dir$ ls
new.tar  new.txt.gz
emertxe@ubuntu:~/Dir1/demo/Dir$ rm new.txt.gz
rm: remove regular file 'new.txt.gz'? y
emertxe@ubuntu:~/Dir1/demo/Dir$ ls
new.tar
emertxe@ubuntu:~/Dir1/demo/Dir$ tar -xvf new.tar
new.txt.gz
emertxe@ubuntu:~/Dir1/demo/Dir$ ls
new.tar  new.txt.gz
emertxe@ubuntu:~/Dir1/demo/Dir$ gunzip new.txt.gz
emertxe@ubuntu:~/Dir1/demo/Dir$ ls
new.tar  new.txt
emertxe@ubuntu:~/Dir1/demo/Dir$ █
```

Figure 49 untar a file