

exit | logout

Part - 2

echo "hello world"

→ prints hello world
→ standard output symbol

echo "hi" > file.txt

→ file.txt will have hi text in it
(hi will be written in file.txt)

cat file.txt

→ hi

echo "bye" > file.txt

→ bye

(hi will be overwritten by bye)
→ append will put all

echo "hi" >> file.txt

bye

hi

cat <file.txt> file1.txt

ls -l

→ file.txt

→ file1.txt

c we are
creating
file1.txt with
file.txt
contents

pwd > new.txt

↳ /home/ubuntu

cat new.txt

→ /home/ubuntu

1 → standard output

↳ standard output stating "new file"

2 → standard error

↳ standard error message "No such file or directory"

echo "hi" > sample.txt

→ hi

echo "hi" > Sample.txt

→ hi

↳ not existing directory

ls /fake/directory

→ no such file error

echo "hi" > sample.txt

ls /fake/directory

→ no such file error

cat sample1.txt

→ file is empty

solution

ls /fake/directory 2> sample1.txt

→ error message

cat sample.txt

→ prints the error message

How not to get the error message
for incorrect directory and show
emptiness

ls /fake/dir 2> /dev/null

→ no output will be ~~seen~~ seen

/dev/null → null path in direct

-x-x-

etc file contains host specific config

files

ls -l /etc | less

Output

Input

that is that function of pipe(1) symbol

tee command

(2 functions at a time)

tee → Gives Input



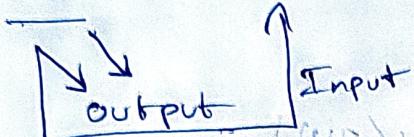
Display output

→ ls | > out.txt

→ cat out.txt

→ Pipeline

ls | tee



→ Display all files

new.txt

→ Displays all files

ls gives me output and by use of

pipeline it sends ~~as~~ Input to

command and it writes ~~as~~ Input
in new.txt file as well as prints

the output to next part of command

— x —

print env

→ Displays environment variables and its

values

export Name = "Hema"

→ It will be added to environment

(User defined variables)

Process Model

↳ echo \$Name
↳ hi Hema

→ hi hemar

unset Name

→ remove the entry from environment

variable

export → set a variable

unset → removes it

printenv → prints all variables

Ubuntu → sudo su -root

root (superior user)

user → []

User who has to run command
through in root

→ []

→ The user who has to run command

→ []

Logout user who has to run command

→ []

Logout user who has to run command

→ []

Logout user who has to run command

→ []

Regular Expression

regular

lorem ipsum

Small letters (a,b,c)

↑

a,b,c $0,1 - \alpha, \frac{A,B,C,D}{-2}$

→ Capital letter only

→ A,B letter word

Integers (0-9)

[abc][0-9][A-Z]

Box

[^abc]

↳ can be any small letter or Integer

except a,b,c

(Q&U required) true

[a-f]

true if abc

[023] → true if any one of them matches in present

[a-zA-Z]

↳ true if letters is from a-z

or A-Z

[0-5][5-9]

→ true if two integers are from given condition.

[.] → specific symbol with & by /

← X ← X

↓ word /

\ word

\ A The → check whether the Input
starts with The ↓ word /

\ word

\ work → check whether the Input
ends with work ↓ word /

word 12

↑
↓ word /

↓ word /

↓ word /

\ b → Begining and end ↓ word /

when search for and we will get

→ and ✓ ↓ word /

→ standard ↓ word /

→ andy

→ random

To avoid this we use ↓ word /

\ b and

and ✓

andy X

↓ word /

↓ word /

\ b and \ b

and ✓

$\backslash B \rightarrow$ In between $\backslash A$ and $\backslash C$

$\backslash B$ and

\rightarrow Standard

\rightarrow random

\rightarrow Aloud

\rightarrow handy

$\backslash B$ and $\backslash B$

\rightarrow Standard

\rightarrow random

\rightarrow handy

$\backslash B$ and $\backslash b$

\rightarrow Aloud

$\backslash b$ and $\backslash B$

\rightarrow handy

$\backslash B [Aa]nd \backslash b$

\rightarrow Aloud

\rightarrow Aloud And

\d → digits [0-9] Alphanumeric

True if input contains Integers

\D → Integers [Alphabets, Symbols]

\s → white space

(Shows white space in Input)

\S → white space (non white space)

\w → a-z, A-Z, 0-9, _ Underscore

\W → ["a-z, A-Z, 0-9, Underscore]

→ Starts with a word

→ Any character Except \n character

[a-z], [A-Z] (3 letters) → Enter/newline

word 2nd letter

↳ Anything can be

\^ → starts with

\The

↳ check whether the start of the paragraph is The

\$ → Ends with [P-O] strings or

apple\$ animal sign to stop

[Ending up to check me End of the para by

word apple

(→ zero (or) more occurred)

(~~abc~~ abcc... and) → abc strings

-ab

→ P-O S-A strings

abc

abc... → P-O S-A strings

abcabc → abbacabc

can be occurred 0 or n times

→ can be followed by abc

followed by ab

[S-A]

[S-A] . [S-A]

+ → one (or) more occurred

abc +

abc

abcc

abcccc... → P-O S-A strings

abcabc → abbacabc

Q → 0 or n occurrence of a character in abc?

ab is not accepted

abc

abcabcabbbabc

Two abbb groups

abc{3} → exactly 3 times

abccc

→ exactly 3 groups of 2 letters

Regular Expression for Email Address

→ should have exactly one @ (1 time) / letter

→ should have mention of 1 letter part

→ start with Alphabets (or) Integer
End with Alphabets

^ [a-zA-Z0-9] + [@] [a-zA-Z] + [.]

↓ It shows first occurrence

Starts with

→ starts with dot(.)

ends with

→ hemachanderen2001@gmail.com ✓

→ mithunangoo.coX → #hema.comX

$\wedge [a-zA-Z0-9]+[@][a-zA-Z]+\.[.]\wedge$

$[a-zA-Z]$

↳ Regexe for email

$-x \rightarrow x \rightarrow x \rightarrow \dots$

Grep, Sed, awk

Linux

cat > Sample.txt

{eg}

now give input which will be

written in Sample.txt

root@mithran:~\$ cat Sample.txt

grep "mithran" Sample.txt

→ finds mithran word

grep - if "mithran" Sample.txt

→ shows mithran

↳ case insensitivity

grep -i "mithran" /home/ubuntu/*.txt

→ show file name where keyword is

present

→ "mithran" Sample.txt 1.txt 2.txt

now file name

→ now file name

$$\wedge [a-zA-Z0-9] + [\@] [a-zA-Z] + [.]$$

$$[a-zA-Z]+\$\n(3d)$$

↳ Regular expression for email

$$-x \rightarrow \overline{x} \rightarrow \overline{x} \rightarrow \overline{\text{odd}}$$

Grep, Sed, awk

Don H. S.

cat: > Sample batch ← {Exp 3rd}
 Is next which will be

now
written in Sample-set for Final

grep "mithran" Sample.txt

~~Paper~~ Birds mithran sand 10/10/12
21

~~Regd. Birds~~ misnomer I saw Jan 2007
dead stuff " Sample & at
" Misnomer

target - if "Mitran" - pump
subject - if "dangha" - blow - 720/2
adjective - if "danghi" - sensitivity

→ gave instead, to

Thomomys (Vandyke) [†] fact
mitratus [†] *Thomomys* (Vandyke) [†] fact
sp. [†]

→ show file name where keyword is present

group A1 "mimran" Sample set 1. Oct 2. Oct

→ show file name

grep -i -w "mithran" sample.txt

ISL 3.3 follows [E.g. If finds "mithran" sample.txt]

grep -i -w -n "mithran" sample.txt

ISL 3.3 follows [E.g. If finds "mithran" sample.txt]

grep "[A-Z]*" sample.txt

→ highlights capital words

finding a word in file → grep

sed → find & Replace

Find Global occurrence

s /is/IS/g sample.txt

sed | s /is/IS/g sample.txt

Substitute

awk → fetch Info

awk 'print' employee.txt

col1 | col2 | col3 | print

awk 'print \$1, \$2, \$3' employee.txt

awk 'print \$1, \$2, \$3' employee.txt

col1 | period3 | print

awk 'print \$1, \$2, \$3' employee.txt

col1 | period3 | print

awk 'print \$1, \$2, \$3' employee.txt

awk '{print \$1, \$3}' employee.txt
print sales account and col 1 - col 3

~~cat /var/log/syslog~~

cat /var/log/syslog

↳ big log file

tail /var/log/syslog | tail -10
→ last 10 lines

tail -n 3 /var/log/syslog

→ last 3 lines

head /var/log/syslog

→ first 10 lines

head -n 3 /var/log/syslog

→ first 3 lines

Printing (30 - 45) lines in 100line file

head -n 45 /var/log/syslog

↓
[output] | tail -n 15 /var/log/syslog
Pipeline ↪ ↑ Input

→ test command "grep" with
awk. i/sales/ & print \$1, \$3] employee.txt
print sales account and col 1, col 3 & sep.

→ test command "grep" with
awk. i/sales/ & print \$1, \$3] employee.txt
→ X → ~~new log.txt~~ output print

cat /var/log/syslog | tail -n 10 | grep "syslog"

↳ big log file

↓ enormous log file

tail

↑

tail /var/log/syslog | tail -n 10

→ last 10 lines

tail -n 3 /var/log/syslog

→ last 3 lines

head /var/log/syslog

→ first 10 lines

head -n 3 /var/log/syslog

→ first 3 lines

Printing (30 - 45) lines in footer file

head -n 45 /var/log/syslog

↓ output | tail -n 15 /var/log/syslog

Pipeline ↪

↑ Input

Sort

→ sort | sort -2

Vim → Linux Text editor

vim employee.txt

; → Insert mode

↳ modify data in Insert mode

→ Esc → exit the mode (Insert)

→ dd → delete a line

:wq → save and quit

In command mode

: 2d → delete 2 lines.

: 1, d → delete all lines

: q → quit

: q! → don't save

sort new.txt

→ sort based on alphabets

sort → new.txt (↑) (creatively)

→ sort in descending order (↓) (creatively)

ls -l | sort

→ show files in sorted order

1s -1 | sort -r

→ reverse order
sort -r | count ← min

unique \rightarrow $\{ \}$

→ fast - $\log n$ min

new - $\text{dict} = \{ \}$ in sorted $\{ \}$ from car

car (insert) \rightarrow count art size of $\{ \}$ \leftarrow min

bike \rightarrow art $\{ \}$ art size of $\{ \}$ \leftarrow min

bike \rightarrow art $\{ \}$ art size of $\{ \}$ \leftarrow min

van \rightarrow art $\{ \}$ art size of $\{ \}$ \leftarrow min

van \rightarrow art $\{ \}$ art size of $\{ \}$ \leftarrow min

plane \rightarrow art $\{ \}$ art size of $\{ \}$ \leftarrow min

unique new - $\text{dict} = \{ \}$ art $\{ \}$ art size of $\{ \}$ \leftarrow min

plane \rightarrow art $\{ \}$ art size of $\{ \}$ \leftarrow min

car \rightarrow art $\{ \}$ art size of $\{ \}$ \leftarrow min

bike \rightarrow art $\{ \}$ art size of $\{ \}$ \leftarrow min

van \rightarrow art $\{ \}$ art size of $\{ \}$ \leftarrow min

unique - c new - $\text{dict} = \{ \}$

1 plane \rightarrow art $\{ \}$ art size of $\{ \}$ \leftarrow min

2 car \rightarrow art $\{ \}$ art size of $\{ \}$ \leftarrow min

2 bike \rightarrow art $\{ \}$ art size of $\{ \}$ \leftarrow min

van \rightarrow art $\{ \}$ art size of $\{ \}$ \leftarrow min

→ duplicate $\{ \}$ \rightarrow $\{ \}$ \leftarrow min

unique - d new - $\text{dict} = \{ \}$

car \rightarrow art $\{ \}$ art size of $\{ \}$ \leftarrow min

bike \rightarrow art $\{ \}$ art size of $\{ \}$ \leftarrow min

van \rightarrow art $\{ \}$ art size of $\{ \}$ \leftarrow min

new1.txt

car

car

bike

car

car

uniq -c new1.txt

car - 2

bike - 1

car - 2

2 car entries are
seen as it calculated
only adjacent
values

sort new1.txt

uniq -c new1.txt

car - 4

bike - 1

wc → word count

wc new1.txt

14 14 71 new1.txt

↓ ↓ ↳ byte (storage)

14 lines words

wc -l new1.txt 14 (lines)

wc -c new1.txt 71 (storage)

wc -w new1.txt 14 (words)