exit / logout

Part -2

```
echo "hello world"
```

→ prints hello world

```
echo "hi" > file.txt
```

↳ standard output symbol

→ 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 input

```
echo "hi" >> file.txt
```

bye
hi

```
cat < file.txt > file1.txt
```

(we are creating file1.txt with file.txt contents)

```
ls -l
```

→ file.txt
→ file1.txt

pwd > new.txt

cat new.txt

→ /home/ubuntu

1 → standard output

2 → standard error message

echo "hi" > sample.txt

→ hi

echo "hii" 1> sample.txt

→ hii

ls /fake/dir → not existing directory

→ no such file error

ls /fake/dir > sample1.txt

→ no such file error

cat sample1.txt

→ file is empty

Solution

ls /fake/dir 2> sample1.txt

→ error message

cat sample1.txt

→ prints the error message

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

ls /fake/dir 2> /dev/null

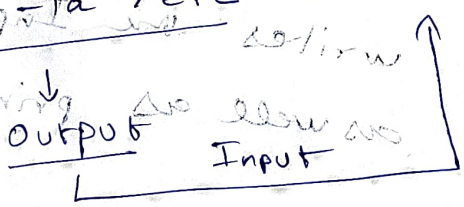
→ no output will be ~~seen~~ seen

/dev/null → null path in linux

—x—x—

etc file contains host specific config files

ls -la /etc | less



that is that function of pipe(|) symbol

—x—x—

tee command

(2 functions at a time)

tee → Given Input

↓

Display output

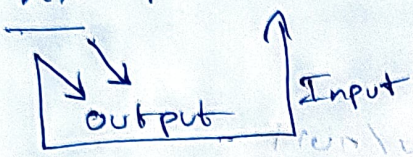
→ ls 1> out.txt

→ cat out.txt

→ list the files

→ pipeline

ls | tee new.txt



→ Display all files

new.txt

→ Displays all files

ls gives the output and by use of pipeline it sends Input to tee command and it writes the Input in new.txt file as well as prints the output.

—x—x—

printenv

→ Display environment variables and its values

export Name = "hema"

⇒ It will be added to environment variables

echo "hi $Name"

⇒ hi hema

unset Name

⇒ remove the entry from environments variable

export → set a variable

unset ⇒ removes it

printenv → prints all variables

—x—x—

Ubuntu —Sudo su —root→ root (superior user)

user

user ← exit

Regular Expressions

regex101

lorem ipsum

Small letters (a, b, z) → capital letters (A, B, C, D - Z)

a, b, c 0, 1 - 9, A, B, C, D - Z

↓

Integers (0-9)

a 3 Z ✓

A 3 a X

[abc][0-9][A-Z] ← [ ]

Ex

[^abc]

↳ can be any letter or Integer or symbol except a, b, z

[a - f]

[0123] → true if any one of the matches is 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

\

\Aword

\AThe → check whether the Input starts with "The"

~~\Z word~~

\work → check whether the Input ends with work.

word\Z

\b → Begining and end

when search for and we will get

→ and ✓
→ standard
→ andy
→ random

To avoid this we use

\band
and ✓
andy X

\band\b
and ✓

\B → Inbetween

\Band

→ Standard

→ random

→ Aldusand

→ handy

\B and \B

→ Standard

→ random

→ handy

\B and \b

→ Aldusand

\band\B

→ andy

\B[Aa]nd\b

→ Aldsand

→ AldusAnd

\d → digits [0-9]

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, _]

. → Any character (Except \n character)

↳ Enter/newline

[a-z]. [A-Z] (3 letters word)

↳ Anything can be 2nd Letter

^ → starts with

^The

↳ check wheather the start of the paragraph is The

$ → Ends with

apple$

↳ check the End of the para has

word apple

* → zero (or) more occurance

abc*

ab

abc

abcc

abccc...

abcabc ab bbac abc

c → can be occured 0 or n times

but should be followed by ab

+ → one (or) more occurance

abc+

abc

abcc

abcccc...

abcabc abbbbac abc

? → 0 or 1 occurance

abc?

ab

abc

abcabcabbabc

abc{3} → exactly 3 times

abccc

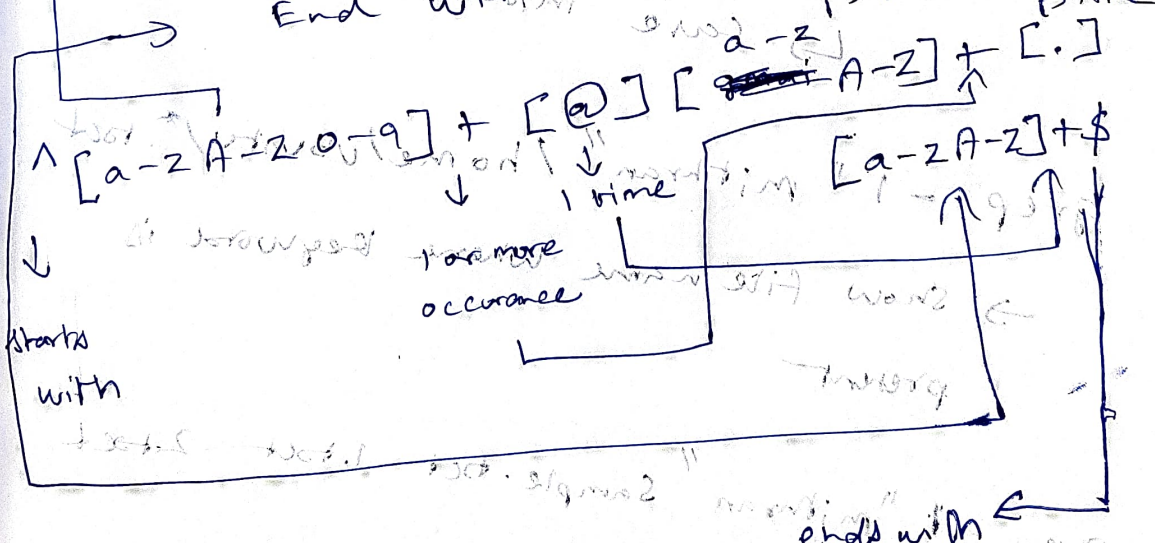
Regular Expression for Email:

→ should have @ (1 time) / letter

→ should have (1 letter)

→ start with " Alphabet (or) Integer

→ End with: Alphabet



→ hemachanderen2001@gmail.com

→ mith_rango.co ✗ → #hema.com ✗

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

[a-zA-Z]

↳ Regex for email

— X — X — X —

Grep, sed, awk

cat > sample.txt

now give input which will be written in sample.txt

grep "mithran" sample.txt

→ finds mithran

grep -i "mithran" sample.txt

↳ case insensitivity

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

→ show file name where keyword is present

-l "mithran" sample.txt 1.txt 2.txt

now file name

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

[a-zA-Z]+$

↳ Regex for email

— x — x — x —

Grep, sed, awk

cat > sample.txt

now give input which will be written in sample.txt

grep "mithran" sample.txt

→ finds mithran

grep -i "mithran" sample.txt

↳ case insensitivity

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

→ show file name where keyword is present

grep -l "mithran" sample.txt 1.txt 2.txt

→ show file name

grep -i -w "mithran" sample.txt

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

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

→ highlights captial words

finding a word in file → grep

sed → find & Replace
find → global occurance

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

↳ Replace

↳ Substitue

awk → fetch Info

awk '{print}' employee.txt → all data

| col1 | col2 | col3 |

awk '{print $1, $3}' employee.txt

| col1 | col3 |

awk '/sales/ {print $1, $3}' employee.txt

print sales account and col1_col3,

— x — x —

cat /var/log/syslog

↳ big log file

tail /var/log/syslog

→ 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 routine file

head -n 45 /var/log/syslog

output | tail -n 15 /var/log/syslog

Pipeline ↵ ↑ Input

awk '/sales/ {print $1, $3}' employee.txt

print sales account and col1, col3

— x — x —

cat /var/log/syslog

↳ big log file

tail /var/log/syslog

→ 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 1000 line file

head -n 45 /var/log/syslog

output | tail -n 15 /var/log/syslog

pipeline ↵ ↑ Input

Sort

Vim → linux text editor

vim employee.txt ; → Insert mode

↳ modify data in Insertion mode

→ Esc to exit the mode (insert)

→ dd to delete a line

:wq → save and quit

In command mode

:2d → delete 2 lines.

:%d → delete all lines

:q → quit

:q! → don't save

Sort new.txt

→ Sort based on alphabets

Sort -r new.txt

→ Sort in desending order (reversed)

ls -l | Sort

→ show files in sorted order

ls -l | sort -r

→ reverse order

uniq

new.txt

car
car
bike
bike
van
van
plane

uniq new.txt

plane
car
bike
van

uniq -c new.txt

1 plane
2 car
2 bike
2 van

→ duplicate (1->)

uniq -d new.txt

car
bike
van

new1.txt

car
car
bike
car
car

uniq -c new1.txt

car - 2
bike - 1
car - 2

2 car entries are see as it calculates only adjacent values

sort new1.txt

uniq -c new1.txt

car - 4
bike - 1

wc → word count

wc new.txt

14 14 71 new.txt

↓ ↓ ↳ byte (storage)

lines words

wc -l new.txt 14 (lines)

wc -c new.txt 71 (storage)

wc -w new.txt 14 (words)