Assignment : 1

List of Topics: Shell Scripting, Introduction to Awk

Examples:

1) Shell script that takes a path, and and gives name of the folder contains ls –R

path

2) How to check the mentioned filename/path is file or directory? If [-f

path] : For files

If [-d dirname] : For directory

If [-z file]: for empty file

3) Environment Variables: The PATH variable is **an environment variable that contains an ordered list of paths** that Linux will search for executables when running a command. Using these paths means that we do not have to specify an absolute path when running a command.

To check the current value of the PATH variable run the following command. echo

$PATH

To add new path into your system you can use

export PATH="$HOME/bin:$PATH"

4) grep "REGEX" filename

- ^ string starts with
- $ string ends with

```
kishan@Kishan:~/Desktop$ cat abc.txt
aa
b
ab
a b
a b c
c
kishan@Kishan:~/Desktop$ grep "^a" abc.txt
aa
ab
a b
a b c
kishan@Kishan:~/Desktop$ grep "[a-b]" abc.txt
aa
b
ab
a b
a b c
```

EX: Now find the string which ends with the either a or b and attach your execution on report.

Exercise:

1) Write a shell script sum.sh that takes an unspecified number of command line arguments (up to 9) of ints and finds their sum.
2) Modify the task 1 code to add a number to the sum only if the number is greater than 10.
3) Write a program which will display file/directory name with the status that it's a file or directory.
4) Display the name of folders which contain more than 2 files or directories(upto level 1 only).
5) Write a shell script which can delete folders names starting with "a" and contains exactly 1 file or directory.(upto level 1 only)

Submission:

1. Submit assignment with the report consists of an input file and output file with proper explanation of each output of all the exercises in pdf format. 2. Add all the outputs and a brief description of the commands used in the given demo scripts in the report.

3. Submitted code in a report or in a .sh file should be well commented. 4. Submit a zip file with

your student id which will consist of the folder for each script & respective outputs. one common report in a parent directory.

Eg:

- IDNOzip
  - report.pdf
  - PR1
    - Script
    - Its respective output
  - PR2

5. Submission Deadline:
6. Late submission will not be evaluated.

Demo Script Examples (Study these, copy and execute, and consider changes to perform similar functions)

Example-1 (hello)

```
#!/bin/bash


function f1 {
   echo Hello from $FUNCNAME!
   VAR="123"
}
f2() {
   p1=$1
   p2=$2
   sum=$((${p1} + ${p2}))
   echo "${sum}"
}
f1
echo ${VAR}
```

```
mySum="$(f2 1 2)"
echo mySum = $mySum

mySum="$(f2 10 -2)"
echo mySum = $mySum
```


Example-2 (sort)

```
#!/bin/bash

# test that at least one argument was passed
if [[ $# -le 0 ]]
then
   printf "Not enough arguments!\n"
   exit
fi

count=1

for arg in "$@"
do
   if [[ $arg =~ ^-?[0-9]+([0-9]+)?$ ]]
   then
      n[$count]=${arg}
```

```
        let "count += 1"
    else
        echo "$arg is not a valid integer!"
    fi
done

sort -n <(printf "%s\n" "${n[@]}")
```

Example-3 (regexp)

```
#!/bin/bash

if [ -z "$1" ]; then
    echo "Usage: $0 filename."
    exit 1
fi

filename=$1

while read -n 1 c
do
    echo "$c"
done < "$filename" | grep '[[:alpha:]]' | sort | uniq -c | sort -nr
```

Example-4 (date and time)

```
#!/bin/bash

# Print default output
echo `date`

# Print current date without the time
echo `date +"%m-%d-%y"`

# Use 4 digits for year
echo `date +"%m-%d-%Y"`

# Display time only
echo `date +"%T"`

# Display 12 hour time
echo `date +"%r"`

# Time without seconds
echo `date +"%H:%M"`

# Print full date
echo `date +"%A %d %b %Y %H:%M:%S"`

# Nanoseconds
```

```
echo Nanoseconds: `date +"%s-%N"`


# Different timezone by name
echo Timezone: `TZ=":US/Eastern" date +"%T"`
echo Timezone: `TZ=":Europe/UK" date +"%T"`

# Print epoch time - convenient for filenames
echo `date +"%s"`
# Print week number
echo Week number: `date +"%V"`

# Create unique filename
f=`date +"%s"`
touch $f
ls -l $f
rm $f

# Add epoch time to existing file
f="/tmp/test"
touch $f
mv $f $f.`date +"%s"`
ls -l "$f".*
rm "$f".*
```

Introduction to AWK

The awk is one of the most prominent text-processing utilities on Linux. AWK is an interpreted programming language. It is very powerful and specially designed for text processing. The input data is divided into records and fields. Awk operates on one record at a time until the end of the input is reached. Records are separated by a character called the record separator. The default record separator is the newline character, which means that each line in the text data is a record.

Awk has 3 components:

1) BEGIN
2) Body
3) END

How to use:

```
awk '{action}' <filename>
```

Examples:

1) Print square of first five integers.

```
BEGIN {
        sum = 0
}
{
        sum = sum + $4
}
END {
        print "Sum of all marks is: " sum
}
```

Save this file with .awk extension and run it using following command: awk -f awk_file data_file

Create this awk program and attach a screenshot on the report.

Try: awk 'BEGIN {i = 1; while (i < 6) { print i; ++i } }'

2) Extract the list of users who all logged in today.

who | awk '{print $1}'

3) Extract all the destination IP addresses present in the routing table. netstat

-nr | awk '**NR**>2 {print $1}'

4) Find all the processes in the Zombie state.

Script:

```
#! /bin/sh
echo $(ps -eo pid,stat | grep 'Z' | awk '{print $1}')
```

Extract the process whose state is I and attach a screenshot on the report. 5)

Following command print all pids whose ppid is 1.

Script:

```
ps -elf | awk '$5==1 {print $4}'
```

6) Find patterns.

Print line if field 1 matches regex in file.

```
awk '$1 ~ /regex/ {action}'
Ex: ls -l | awk '$9 ~/s$/ {print $9}'
```

For more information please refer:

https://www.shortcutfoo.com/app/dojos/awk/cheatsheet

Exercise:

1) Use the given text file(data) and create an awk file to display the name of the students who got less than 70 marks in Maths.
2) Use the given text file(data) along with the assignment and extract following results. (using awk command)
    a) Display all unique subjects.
    b) Display the name of the students who got more than 80 marks in any subject.
    c) Display names of the students who have chemistry as subject.
    d) Find the number of students who have History as subject.
3) Write a shell script to display the name of the directories which contain more than 2 files using awk command.(upto level 1 only).
4) Store some numbers in a file. Create a shell script which takes the numbers from the file and adds all numbers, also appends the result in the same file. 5) Take a input from the user and make a center pyramid of that much rows with * symbol.

Submission:

1. Submit assignment with the report consists of an input file and output file with proper explanation of each output of all the exercises in pdf format. 2. Add all the outputs and a brief description of the commands used in the given demo scripts in the report.
3. Submitted code in a report or in a .sh file should be well commented. 4. Submit a zip file with your student id which will consist of the folder for each script & respective outputs. one common report in a parent directory.

Eg:

- IDNO.zip
    - report.pdf
    - PR1
        - Script
        - Its respective output
    - PR2

5. Submission Deadline:
6. Late submission will not be evaluated.