

# MC214 LAB-2 REPORT

Nisarg Suthar

202003030

[9 Sept, 2021]

---

**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.

## SCRIPT:

```
# start of awk script
```

```
BEGIN{  
print "\n Name of the students who got less than 70 marks in Maths  
:\n"  
}
```

```
{  
if((NR>1 && $3=="Maths" && $4<70))          # checking all  
lines after first for
```

```
subject maths and marks less than 70  
{print "", $2}                                # printing  
the line if found  
}
```

```
END{  
print "\n"  
}
```

```
#end of awk script
```

**INPUT (As command Line Argument):**

data.txt

## OUTPUT:

Name of the students who got less than 70 marks in Maths :

Tom

Bob

## EXPLANATION:

First, we write a prompt in the **BEGIN** block of the **awk** file.

Then in the **BODY**, we check for all lines after the first line (**NR>1**) if the subject (**\$3**) is "Maths" and the marks obtained (**\$4**) are less than 70.

If such a record exists then we print the name of the person (**\$2**) on the terminal.

Lastly, we print an empty line in the **END** block.

**EXERCISE 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.

**SCRIPT:**

```
#!/bin/bash
```

```
# Script to find the unique subjects, students getting more than 80 marks
```

```
# students having Chemistry as their subject and
```

```
# number of students having History as their subject
```

```
echo -e "\n\nThe Report Card:\n"          # printing the report once  
cat data.txt
```

```
# (A) Displaying the unique subjects
```

```
echo -e "\n\n(A) The unique subjects are :\n"
```

```
cat data.txt | awk 'NR>1 {print " " $3}' | sort | uniq      # here  
uniq removes repeated subjects
```

```
# (B) Finding students with >80 marks
```

```
echo -e "\n\n(B) Students who got more than 80 marks :\n"
```

```
cat data.txt | awk 'NR>1 && $4>80 {print "",$2}' | sort      #  
marks are in column 4
```

```
# (C) Students studying Chemistry
```

```
echo -e "\n\n(C) The students who have Chemistry as subject :\n"
```

```
cat data.txt | awk 'NR>1 && $3=="Chemistry" {print "",$2}' | sort  
# subjects are in column 3
```

```
# (D) Number of History students
```

```
echo -e -n "\n\n(D) Number of students who have History as their  
subject : "
```

```
echo $(cat data.txt | awk 'NR>1 && $3=="History" {print $2} ' | wc -  
l)      # wc -l counts the students
```

```
echo
```

```
# END of Script
```

## INPUT:

NIL

## OUTPUT:

The Report Card:

```
id name subject marks grade
1 John History 78 C
2 Mark Chemistry 65 D
3 Julie History 85 B
4 Emily Biology 90 A
5 Sam Chemistry 89 B
6 Tom Maths 68 D
7 Bob Maths 55 E
```

(A) The unique subjects are :

```
Biology
Chemistry
History
Maths
```

(B) Students who got more than 80 marks :

```
Emily
Julie
Sam
```

(C) The students who have Chemistry as subject :

```
Mark
Sam
```

(D) Number of students who have History as their subject : 2

## EXPLANATION:

(A) At first, I have displayed the data file just for reference.

To display the unique subjects the **data file** is piped to **awk 'NR>1 {print " "\$3}'** command which filters out the subject's column after the first headings row and this is in turn piped to the **sort** command and which is in turn piped to the **unique** command to finally obtain all the unique subjects.

(B) To display the students getting more than 80 marks the data file is piped to **awk 'NR>1 && \$4>80 {print "", \$2}'** command which filters out the mark's column with more than 80 marks after the first headings row and prints the names of the corresponding students which is in turn piped to the sort command to finally obtain all the students with more than 80 marks.

(C) To display the students having Chemistry subject the data file is piped to **awk 'NR>1 && \$3=="Chemistry" {print "", \$2}'** command which filters out the subject's column with Chemistry after the first headings row and prints the names of the corresponding students which is in turn piped to the sort command to finally obtain all the students studying Chemistry.

(D) To display the number of students having History subject the data file is piped to **awk 'NR>1 && \$3=="History" {print \$2}'** command which filters out the subject's column with History after the first headings row and prints the names of the corresponding students which is in turn piped to the word count (**wc -l**) command to finally obtain the number of students studying History.

**EXERCISE 3:** Write a shell script to display the name of the directories which contain more than 2 files using awk command. (up to level 1 only).

**SCRIPT:**

```
#!/bin/bash

# Ex3 : Shell script to display the names of folders containing more
than two files.

# creating a sample directory for testing

mkdir -p "sample"

cd "sample"                # making test files and directories in
the sample directory

mkdir -p "a"
mkdir -p "aa"
mkdir -p "ab"
mkdir -p "b"
mkdir -p "bb"
mkdir -p "ba"

touch "a/f1"; touch "a/f2"
touch "aa/f1"
touch "ab/f1"; touch "ab/f2"; touch "ab/f3"
touch "b/f1"
touch "bb/f1"; touch "bb/f2"
touch "ba/f1"; touch "ba/f2"; mkdir -p "ba/f3"

cd ..                      # back to parent directory

echo -e "\nThe created sample directories and files:\n"

ls -R sample              # printing all the directories and contained
files

# finding the directories having more than 2 files

echo -e "\nThe directories having more than 2 files/directories are
:"
```

```

folders=$(ls sample)          # storing all the directories in folders
variable

for i in $folders              # iterating over all the
directories
do

    # awk counts number of files

    if [[ $(ls -1 "sample/$i" | awk 'BEGIN {n=0} {++n} END{
if((n>2)) print }') ]]
    then
        echo $i
    fi        # end of if

done                # end of loop

echo                # printing an empty line for aesthetic purpose

# end of script

```

## INPUT:

Nil

## OUTPUT:

The created sample directories and files:

```

sample:
a  aa  ab  b  ba  bb

```

```

sample/a:
f1  f2

```

```

sample/aa:
f1

```

```

sample/ab:
f1  f2      f3

```

```

sample/b:
f1

```

```

sample/ba:

```

```
f1  f2      f3
```

```
sample/ba/f3:
```

```
sample/bb:
```

```
f1  f2
```

The directories having more than 2 files/directories are :

```
ab
```

```
ba
```

## EXPLANATION:

Here we have first created a **sample** directory containing some test directories with a few files. We display them on terminal using the **ls -R sample** command. Then we have to find all the directories containing more than 2 files or directories.

First, we store all the directories in the sample folder in the **folders** variable. Then we iterate over them and decide whether to print the directory name using command:

```
if [[ $(ls -1 "sample/$i" | awk 'BEGIN {n=0} {++n} END{ if((n>2))  
print "1"}') ]]
```

Here, **ls "sample/\$i"** gives the list of files and directories in the **\$i** directory. Then we send this output to **awk 'BEGIN {n=0} {++n} END{ if((n>2)) print }'** command using pipe (|) which if the directory contains more than two files prints a non-empty string. Finally, the outer if prints the name of the directory (**\$i**) on the terminal if it receives a non-empty string.



**EXERCISE 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.

### **SCRIPT:**

#### **BASH SCRIPT:**

```
#!/bin/bash

# Script to add all the numbers in a file

touch sample.txt      # creating a sample file
echo -en "3\n5\n8\n11" > sample.txt

# displaying the created file
echo -e "\nThe file created :"
cat sample.txt

# Adding the numbers using awk
awk -f ex4.awk sample.txt >> sample.txt

# displaying the file after appending the sum
echo -e "\n\nFile after appending the sum:"
cat sample.txt

echo

# end of script
```

#### **AWK SCRIPT:**

```
# awk script to sum all numbers in a file

BEGIN{
sum=0      # initialising sum variable
}

{
sum=sum+$1      # adding successive numbers
}

END{
print "\nThe sum of the numbers is : " sum # printing final sum
}

# end of awk script
```

**INPUT**(As a file sample.txt):

3  
5  
8  
11

**OUTPUT:**

The file created :

3  
5  
8  
11

File after appending the sum:

3  
5  
8  
11

The sum of the numbers is : 27

**EXPLANATION:**

Here we create a **sample** file and then insert a few numbers in it.

Then we invoke the awk script and pass the sample file as an argument.

In the awk script we initialize a **sum** variable in the **BEGIN** section.

Then in the **BODY** we successively add the numbers in the file to the sum variable.

In the **END** section we print the **sum** which is appended to the sample file by redirection operator.

```
awk -f ex4.awk sample.txt >> sample.txt
```

At last, we display the new file with the sum of numbers appended at end of the file.

**EXERCISE 5:** Take a input from the user and make a center pyramid of that much rows with \* symbol.

**SCRIPT:**

```
#!/bin/bash

# Script to print a centre pyramid
echo

# taking user input for rows
read -p "Enter a number : " x

echo

# using a loop to print the rows
for i in $(seq $x)
do
    # y variable contains the number of spaces to be printed on
    # each row before asterisk
    y=`expr $x - $i`

    for j in $(seq $y)          # printing the spaces on each row
using loop
    do
        echo -n " "
    done

    for j in $(seq $i)          # printing asterisk on each row using a
loop
    do
        echo -n ' *'
    done

    echo          # printing newline after each row

done          # end of loop to print the pyramid

echo

# end of script
```

**INPUT:**

5

## OUTPUT:

```
  *
 * *
* * *
* * * *
* * * * *
```

## EXPLANATION:

Here we first take a number from user in variable x.

Then we iterate a loop for x number of times where variable I takes values from 1 to x.

Each time we count the number of spaces to be printed on each row by taking difference x-i. And then we print the asterisks \* for i times.

While printing the asterisk we add a space before it to make the resulting output look like a center pyramid.

## EXAMPLE SCRIPTS

### SCRIPT 1: SQUARES OF NUMBERS

# awk script to print the square of first five positive integers

```
BEGIN{
print "\n The Squares of first five positive integers:"
i=1;
while(i<6)      # while loop
{print "",i*i;i++}
print "\n"
}
```

# end of script

A screenshot of a terminal window with a dark background and a stylized dragon logo. The terminal title bar shows the user 'adastra' on a VM named 'adAstra-VM' at the directory '~/Desktop/MC214\_Labs/Lab-2/examples'. The prompt is 'adastra@adAstra-VM:~/Desktop/MC214\_Labs/Lab-2/examples\$'. The user has entered the command 'awk -f ex1.awk'. The output of the script is displayed: 'The Squares of first five positive integers:' followed by the numbers '1', '4', '9', '16', and '25' on separate lines. The prompt is now 'adastra@adAstra-VM:~/Desktop/MC214\_Labs/Lab-2/examples\$ |' with a cursor.

```
adastra@adAstra-VM: ~/Desktop/MC214_Labs/Lab-2/examples
adastra@adAstra-VM:~/Desktop/MC214_Labs/Lab-2/examples$ awk -f ex1.awk

The Squares of first five positive integers:
1
4
9
16
25

adastra@adAstra-VM:~/Desktop/MC214_Labs/Lab-2/examples$ |
```

### EXPLANATION:

Here we use only the **BEGIN** block of awk file. We iterate over a loop starting with 1 till 5 and print the square of them one by one.

## SCRIPT 4: IDLE KERNEL THREADS

```
#!/bin/bash

# Script to print the pids of idle kernel threads

echo -e "\nThe idle kernel threads :"
```

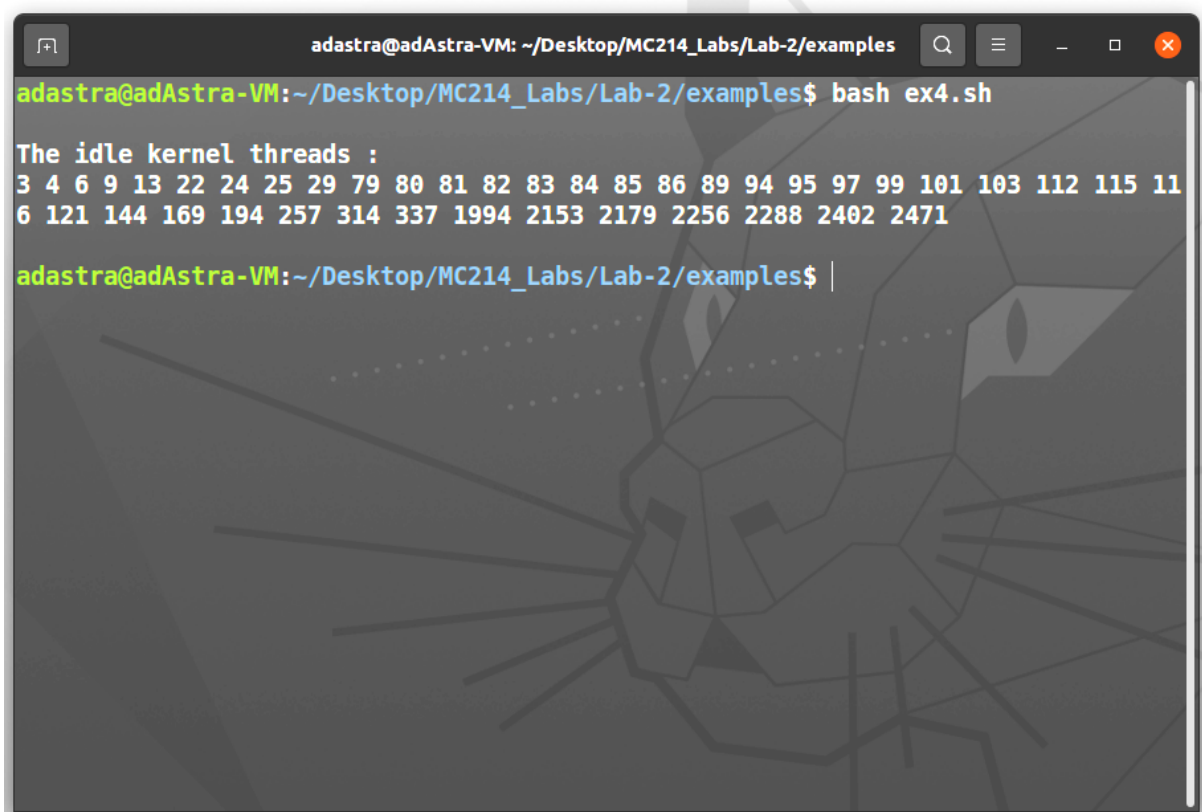
```
echo $(ps -eo pid,stat | grep 'I' | awk 'NR>1 {print "\n" $1}') #
option I filters idel kernel threads
```

```
echo
```

```
# end of script
```



```
adastra@adAstra-VM: ~/Desktop/MC214_Labs/Lab-2/examples
adastra@adAstra-VM:~/Desktop/MC214_Labs/Lab-2/examples$ bash ex4.sh

The idle kernel threads :
3 4 6 9 13 22 24 25 29 79 80 81 82 83 84 85 86 89 94 95 97 99 101 103 112 115 11
6 121 144 169 194 257 314 337 1994 2153 2179 2256 2288 2402 2471

adastra@adAstra-VM:~/Desktop/MC214_Labs/Lab-2/examples$ |
```

## EXPLANATION:

Here we are printing the PIDs of all the threads which are in idle.

First we get the list of all processes with their information using **ps -eo pid,stat**.

Then we filter out the information of the idle threads using **grep 'I'**. Finally, we print the PIDs of all those threads using **awk 'NR>1 {print "\n" \$1}'**. Here the PIDs are stored in the first column of the output of **ps** so we use **\$1** in **awk** command.