SECTION 21

**Introduction to AWK :**

The awk command is programming language, which requires no compiling, and allows,the user to use variables, numeric functions, string functions, and logical operators.

The awk command in unix is just like a scripting language which is used for text processing.

Awk is used like

1. A command
2. A scripting language

**awk command**

awk command in unix is most important command used to find and replace the text.

How awk command works?

* Awk reads data from a file or from its standard input, and output to its standard output.
* Awk views a text file as records and fields.
* Each line is a record and columns in lines/record are called fields.
* **By default fields are separated based on space**. (Note: we can also change the field separator with -F option)
* Awk command works on each line individually
* Awk has its own predefined variables like $0,$1,$2……$n and NR,NF …….

**Syntax:**

Awk options ‘pattern {action}’ filename

[or]

Command | Awk options ‘pattern {action}’

[or]

awk ‘Begin {start\_action} pattern/condition {action to perform on each line} END {stop\_action}’ filename

[or]

command | awk ‘Begin {start\_action} pattern/condition {action to perform on each line} END {stop\_action}’

[or]

awk -f awk\_script.awk filename ====>calling the awk file(in which the above command is saved)

Where the pattern indicates the pattern or the condition on which the action is to be executed for every line matching the pattern.

In case of a pattern not being present , the action will be executed for every line of the file.

**Awk can take the following options:**

-F fs To specify a field separator

-v var=value To declare a variable

-f file To specify a file that contains awk script

**awk command with action and basic variables :**

Execute awk command without options, pattern and action.

Action is a logic

Simple logic is print “something’

**EX:**

awk ‘{ }’ /etc/passwd ====> you won’t get anything.

awk ‘{ print “ok” }’ /etc/passwd ====>it will print ok n no.of times based on file lines/records

awk ‘{ print “ok”,”awk scripting” }’ /etc/passwd ====>it will print ok n no.of times based on file lines/records

**Basic variables:**

$0,$1,$2……

NR ---> no of records. It will display the record numbers/line numbers

NF ----> no of fields. It will display the count of fields in each and every line

FILENAME

**NOTE:**

For $0 you will get entire data in that file .

Don’t give quotations for the variables.

With $NF we have one advantage.it will prints the last field of each & every record/line

**EX:**

demo.txt

This is the first line

This is the second line

This is the third line

awk ‘{ print $1 }’ demo.txt

awk ‘{ print $1,$3 }’ demo.txt

awk ‘{ print $3,$1 }’ demo.txt

awk ‘{ print $3,$1 “using awk scripting” }’ demo.txt ====>clubbing string with variables.

awk ‘{ print NR,$1,$3 }’ demo.txt

awk ‘{ print NR,NF }’ demo.txt

[OR]

awk ‘{ print “The line no is: ”,NR, “The no of fields are: “NF }’ demo.txt

awk ‘{ print $NF }’ demo.txt ===> it will prints the last field of each & every record/line

awk ‘{ print FILENAME }’ demo.txt ===> it will print filename n no.of times based on file lines/records

**Introduction to awk scripting :**

awk ‘/root/ { print $0 } ’ /etc/passwd ==>it prints the matching records from the file

awk ‘NR>=3 { print $0 } ’ /etc/passwd

awk ‘NR>=3 { print NR,$0 } ’ /etc/passwd

**Syntax :**

awk ‘Begin {start\_action} pattern/condition {action to perform on each line} END {stop\_action}’ filename

awk -f awk\_script.awk filename ====>calling the awk file(in which the above command is saved)

Here are the actions

BEGIN block is performed before reading the file. (executes only once)

END block is performed after processing the file. (executes only once)

Rest of the actions are performed while processing the file. (it will executes repeatedly if needs)

**Note:**

Need at least one action to run awk script

No need of i/p for BEGIN actions.

Input is required only for middle actions and END actions

Ex 1:

awk ‘ ‘ /etc/passwd ======> if no i/p then there is no action i.e you won’t get o/p

Ex 2:

awk ‘BEGIN { print “ok” }’ ======> it prints string .

Ex 3:

awk 'BEGIN { print "==========working on /etc/passwd file=====" } /root/ { print $0 }END { print "=======completed work on /etcpasswd=====" }' /etc/passwd

OUTPUT :

==========working on /etc/passwd file=====

root:x:0:0:root:/root:/bin/bash

nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin

=======completed work on /etcpasswd=====

Ex 4:

[or]

awk -f myawk.awk /etc/passwd

OUTPUT :

==========working on /etc/passwd file=====

root:x:0:0:root:/root:/bin/bash

nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin

=======completed work on /etcpasswd=====

**Awk command with option, action and basic variables.**

About -F

awk -F : ‘{ print $1 }’ /etc/passwd =====>it will displays the first field

echo “/home/ec2-user” | awk -F / ‘{ print $2 }’ =====>it will displays the first field

echo “/home/ec2-user” | awk -F ec2 ‘{ print $2 }’ =====>it will displays the first field

About -v

Here -v is used to define some variable

echo “/home/ec2-user” | awk -F ec2 -v x=5 ‘{ print $2, x }’ =====>it will displays the first field

echo “/home/ec2-user” | awk -F ec2 -v x=5 -v y=10 ‘{ print $2, x ,y }’ =====>it will displays the first field

About -f

***demo.txt***

This is the first line

This is the second line

This is the third line

***action.awk***

{ print $2 }

awk -f action.awk demo.txt

**Simple HELLO WORLD awk script :**

**Syntax :**

awk ‘Begin {start\_action} pattern/condition {action to perform on each line} END {stop\_action}’ filename

awk -f awk\_script.awk filename ====>calling the awk file(in which the above command is saved)

awk ‘BEGIN { print “Hello world !!” }’

***first we need to findout the location of that awk i.e which awk***

***helloworld.awk***

#!/bin/awk -f

BEGIN { print “Hello world !!” }

**Running as a script**

./helloworld.awk

[or]

**Running as a command**

awk -f helloworld.awk ==>if you use this command then no need to give shebang line in script

**How to define a variable , display a variable and execute multiple statements**

awk ‘BEGIN { a=5 }‘

awk ‘BEGIN { a=5;print a }‘

***simple.awk*** *(while executing multiple statements semicolon must req if you give all cmds in single line)*

#!/bin/awk -f

BEGIN { a=5;print a }

**Running as a script**

./simple.awk

**Running as a command**

awk -f simple.awk ==>if you use this command then no need to give shebang line in script

***simple.awk*** *(while executing multiple statements no need semicolon if you give each cmd in new line)*

#!/bin/awk -f

BEGIN {

a=5

print a

}

**Running as a script**

./simple.awk ===>need sheband line to run

**Running as a command**

awk -f simple.awk ==>if you use this command then no need to give shebang line in script

**Awk command or awk script : Reading variable values from command line ?**

echo “2 6” | awk ‘{ print “a=”$1, “b=”$2}’

[or]

***ab.txt***

2 6

awk ‘{ print a=$1; b=$2 print a,b }’ ab.txt

[or]

***ab.txt***

2 6

5 7

awk ‘{ print a=$1; b=$2 ; print “a=”a, “b=”b}’ ab.txt

***display.awk***

BEGIN {

}

{

a=$1

b=$2

print “a=”a, “b=”b

}

awk -f display.awk abc.txt

***display.awk***

#!/bin/awk -f

BEGIN {

}

{

a=$1

b=$2

print “a=”a, “b=”b

}

./display.awk abc.txt

***display.awk***

#!/bin/awk -f

BEGIN {

Print “Finding a and b values for ab.txt”

}

{

a=$1

b=$2

print “a=”a, “b=”b

}

END{

Print “completed reading a and b values”

}

./display.awk abc.txt

How to use awk command/script in shell scripts ?

awk ‘BEGIN { print “hi” }’

[or]

***hi.awk***

BEGIN { print “hi” }

**awk -f hi.awk**

***demo.sh***

#!/bin/bash

pwd

date

awk ‘BEGIN { print “hi” }’

**./demo.sh**

***demo.sh***

#!/bin/bash

pwd

date

awk ‘BEGIN { print “hi” }’

awk -f hi.awk

**./demo.sh**

***demo.sh***

#!/bin/bash

pwd

date

a=$(awk ‘BEGIN { print “hi” }’)

b=$(awk -f hi.awk)

echo “a value is : $a”

echo “b value is : $b”

**./demo.sh**

How to read variables for awk using shell script with read command ?

echo “3 6” |awk ‘{ a=$1;b=$2;print a,b }’

read\_values.sh

read -p “enter a value: ”a

read -p “enter a value: ”b

echo $a $b“ | awk ‘{ x=$1;y=$2; print “x=”x,”y=”y}’

read\_values.sh

read -p “enter a value: ”a

read -p “enter a value: ”b

echo $a $b“ | awk ‘{ x=$1;y=$2; print “sum=”x+y}’

read\_values.sh

read -p “enter a value: ”a

read -p “enter a value: ”b

file.txt

hhhhdhodipafioa

awk -v x=$a -v y=$b ‘{ print “sum=”x+y }’ file.txt xxxxxxxxxxxxxxx we are not taking i/p from file.txt

awk -v x=$a -v y=$b ‘BEGIN { print “sum=”x+y }’ ==>BEGIN block doesn’t required any input

read\_values.sh

read -p “enter a value: ”a

read -p “enter a value: ”b

awk -v a=$a -v b=$b ‘BEGIN { print a+b }’

Desing Arithmetic calculator with shell script using awk command

***arithcal.sh***

#!/usr/bin/env bash

clear

read -p "Enter num 1:" x

read -p "enter num 2:" y

echo "============MENU=============="

echo "1.Addition"

echo "2.sub"

echo "3.Mul"

echo "4.div"

echo "========================="

read -p "enter your option (1-4):" A

case $A in

1)

awk -v a=$x -v b=$y 'BEGIN { print "ADDITION=:"a+b }'

;;

2)

awk -v a=$x -v b=$y 'BEGIN { print "SUB=:"a-b }'

;;

3)

awk -v a=$x -v b=$y 'BEGIN { print "MUL=:"a\*b }'

;;

4)

awk -v a=$x -v b=$y 'BEGIN { print "Div=:"a/b }'

;;

\*)

echo "you entered invalid option"

;;

esac

Systemct status httpd > demo.txt

Systemctl status httpd | awk ‘NR==3 { print $2 }’

[or]

awk ‘NR==3 { print $2 }’ demo.txt

cat /etc/passwd | awk -F : ‘{print $1}’

cat -n /etc/passwd =====>it will shows data with line numbers

Ex:

awk ‘{ print $0 }’ /etc/passwd ====>like cat command it displays the entire data from file

awk ‘BEGIN { print “ok” }’ ====>like echo command it displays the string.