AWK Operations:

(a) Scans a file line by line

(b) Splits each input line into fields

(c) Compares input line/fields to pattern

(d) Performs action(s) on matched lines

Syntax:

```
awk options 'selection _criteria {action }' input-file > output-file
```

Create a file named employee.txt and add this data.

```
ajay manager account 45000
sunil clerk account 25000
varun manager sales 50000
amit manager account 47000
tarun peon sales 15000
deepak clerk sales 23000
sunil peon sales 13000
satvik director purchase 80000
```

```
devops@devops:~$ awk {print} employee.txt
ajay manager account 45000
sunil clerk account 25000
varun manager sales 50000
amit manager account 47000
tarun peon sales 15000
deepak clerk sales 23000
sunil peon sales 13000
satvik director purchase 80000
```

Even, you can use single quotes.

$ awk ‘{print}’ employee.txt

$ **awk '{print $0}' employee.txt**

Demo 1: Use of awk

1. Print manager data from employee.txt
   awk '/manager/ {print}' employee.txt

```
devops@devops:~$ awk '/manager/{print}' employee.txt
ajay manager account 45000
varun manager sales 50000
amit manager account 47000
```

2. Print name and salary of the employee
   awk '{print $1, $4}' employee.txt

```
devops@devops:~$ awk '{print $1, $4}' employee.txt
ajay 45000
sunil 25000
varun 50000
amit 47000
tarun 15000
deepak 23000
sunil 13000
satvik 80000
```

Practice 1: put a space between the fields in the output.

```
devops@devops:~$ awk '{print $1,"",$4}' employee.txt
ajay  45000
sunil  25000
varun  50000
amit  47000
tarun  15000
deepak  23000
sunil  13000
satvik  80000
devops@devops:~$ awk '{print $1,":",$4}' employee.txt
ajay : 45000
sunil : 25000
varun : 50000
amit : 47000
tarun : 15000
deepak : 23000
sunil : 13000
satvik : 80000
```

Built-in Variables in AWK

NR	Adds line number to each record in output
NF	Count of fields. Use $FF to print last field in the data
FS	Field separator. Default is space. To use another field separator use -F
RS	Current record separator
OFS	Stores the output field separator (,)
ORS	Stores the output record separator (newline)

3. Print line number with every record in the output.
   awk '{print NR, $1, $4}' employee.txt

```
devops@devops:~$ awk '{print NR,$1,$2}' employee.txt
1 ajay manager
2 sunil clerk
3 varun manager
4 amit manager
5 tarun peon
6 deepak clerk
7 sunil peon
8 satvik director
```

4. Print last field with NF.
   awk '{print NR, $1, $NF}' employee.txt

```
devops@devops:~$ awk '{print NR,$1,$NF}' employee.txt
1 ajay 45000
2 sunil 25000
3 varun 50000
4 amit 47000
5 tarun 15000
6 deepak 23000
7 sunil 13000
8 satvik 80000
```

5. Print record 2 to 5 for employees.

```
devops@devops:~$ awk 'NR==2,NR==5 {print NR,$1,$NF}' employee.txt
2 sunil 25000
3 varun 50000
4 amit 47000
5 tarun 15000
```

6. Print total records in the file.
   awk 'END {print NR}' employee.txt

```
devops@devops:~$ awk 'END {print NR}' employee.txt
8
```

7. Print number of characters in the record 1 of employee.txt file.

```
devops@devops:~$ awk 'NR==1 {print length}' employee.txt
26
```

Demo 2 – Check all users available on your system.

$ cat /etc/passwd

$ awk -F “:” ‘{print $1}’ /etc/passwd

$ sudo awk -F ":" '{print $1 "\t" $6}' /etc/passwd

Demo 3: check all shells in your system and unique.

$ sudo awk -F "/" '/^\// {print $NF}' /etc/shells

```
devops@devops:~$ sudo awk -F "/" '/^\// {print $NF}' /etc/shells
sh
sh
bash
bash
rbash
rbash
dash
screen
tmux
```

The df command displays the amount of disk space available on the filesystem with each file name's argument.

$ df

Demo 4: check df command and search “/dev/sdc” filesystem details. Display filesystem name and block+used columns total.

$ df | grep "/dev/sdc" | awk '{print $1,$2+$3}'

$ df | awk '/\/dev\/sdc/ {print $1,$2+$3}'

```
devops@devops:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           196M  1.4M  195M   1% /run
efivarfs        256K   45K  212K  18% /sys/firmware/efi/efivars
/dev/sda2       9.3G  7.0G  1.9G  80% /
tmpfs           978M     0  978M   0% /dev/shm
tmpfs           5.0M  8.0K  5.0M   1% /run/lock
/dev/sda1       537M  6.4M  531M   2% /boot/efi
tmpfs           196M  128K  196M   1% /run/user/1000
devops@devops:~$ df | grep /dev/sda | awk '{print $1,$2,$3}'
/dev/sda2 9674340 7244824
/dev/sda1 549816 6516
devops@devops:~$ df -h | grep /dev/sda | awk '{print $1,$2,$3}'
/dev/sda2 9.3G 7.0G
/dev/sda1 537M 6.4M
```

~~Few real-world scripts to practice:~~

Log Rotation Script: Write a script to rotate log files to prevent them from growing too large.

```
#!/bin/bash
log_file="/path/to/logfile.log"
max_size=1000000 # 1MB
if [ $(wc -c < "$log_file") -gt $max_size ]; then
    mv "$log_file" "$log_file.old"
    touch "$log_file"
fi
```

System Monitoring Script: Develop a script to monitor system resource usage and send alerts if thresholds are exceeded.

```
#! /bin/bash

# This script check the memory usage  and alerts if it is more than threshold limit.

mem_threshold=90

mem_usage=$(free | grep Mem | awk '{print $3/$2 * 100}')

echo $mem_usage
mem_usage_1=${mem_usage%.*}
echo $mem_usage_1

if [ $mem_usage_1 -gt $mem_threshold ]
then
echo "Memory usage high"
else
echo "Memory usage  low"
fi
```

Automated Backup Script: Create a script to automatically backup files or directories to a specified location.

```
#!/bin/bash
backup_dir="/path/to/backup"
source_dir="/path/to/source"
timestamp=$(date +"%Y%m%d%H%M%S")
tar -czf "$backup_dir/backup_$timestamp.tar.gz" "$source_dir"
```

File Transfer Script: Develop a script to transfer files securely between servers using SCP or SFTP.

```
#!/bin/bash
source_file="/path/to/source/file"
destination_server="user@hostname:/path/to/destination/"
scp "$source_file" "$destination_server"
```