**1. AWK Operations:**  
(a) Scans a file line by line  
(b) Splits each input line into fiel  
(c) Compares input line/fields to pattern  
(d) Performs action(s) on matched lines

**Example:**  
Consider the following text file as the input file for all cases below.

$cat > 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

1. **Default behavior of Awk :**By default Awk prints every line of data from the specified file.

$ awk '{print}' employee.txt

**Output:**

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

In the above example, no pattern is given. So the actions are applicable to all the lines. Action print without any argument prints the whole line by default, so it prints all the lines of the file without failure.

**2. Print the lines which matches with the given pattern.**

$ awk '/manager/ {print}' employee.txt

**Output:**

ajay manager account 45000

varun manager sales 50000

amit manager account 47000

In the above example, the awk command prints all the line which matches with the ‘manager’.

**3. Splitting a Line Into Fields :**For each record i.e line, the awk command splits the record delimited by whitespace character by default and stores it in the $n variables. If the line has 4 words, it will be stored in $1, $2, $3 and $4 respectively. Also, $0 represents the whole line.

$ awk '{print $1,$4}' employee.txt

**Output:**

ajay 45000

sunil 25000

varun 50000

amit 47000

tarun 15000

deepak 23000

sunil 13000

satvik 80000

In the above example, $1 and $4 represent Name and Salary fields respectively.

**Built In Variables In Awk**

Awk’s built-in variables include the field variables—$1, $2, $3, and so on ($0 is the entire line) — that break a line of text into individual words or pieces called fields.

**NR:** NR command keeps a current count of the number of input records. Remember that records are usually lines. Awk command performs the pattern/action statements once for each record in a file.

**NF:** NF command keeps a count of the number of fields within the current input record.

**FS:** FS command contains the field separator character which is used to divide fields on the input line. The default is “white space”, meaning space and tab characters. FS can be reassigned to another character (typically in BEGIN) to change the field separator.

**RS:** RS command stores the current record separator character. Since, by default, an input line is the input record, the default record separator character is a newline.

**OFS:** OFS command stores the output field separator, which separates the fields when Awk prints them. The default is a blank space. Whenever print has several parameters separated with commas, it will print the value of OFS in between each parameter.

**ORS:** ORS command stores the output record separator, which separates the output lines when Awk prints them. The default is a newline character. print automatically outputs the contents of ORS at the end of whatever it is given to print.

**Examples:**

**Use of NR built-in variables (Display Line Number)**

$ awk '{print NR,$0}' employee.txt

**Output:**

1 ajay manager account 45000

2 sunil clerk account 25000

3 varun manager sales 50000

4 amit manager account 47000

5 tarun peon sales 15000

6 deepak clerk sales 23000

7 sunil peon sales 13000

8 satvik director purchase 80000

In the above example, the awk command with NR prints all the lines along with the line number.

**Use of NF built-in variables (Display Last Field)**

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

**Output:**

ajay 45000

sunil 25000

varun 50000

amit 47000

tarun 15000

deepak 23000

sunil 13000

satvik 80000

In the above example $1 represents Name and $NF represents Salary. We can get the Salary using $NF , where $NF represents last field.

**Another use of NR built-in variables (Display Line From 3 to 6)**

$ awk 'NR==3, NR==6 {print NR,$0}' employee.txt

**Output:**

3 varun manager sales 50000

4 amit manager account 47000

5 tarun peon sales 15000

6 deepak clerk sales 23000

**More Examples**

**For the given text file:**

$cat > test.txt

A B C

Tarun A12 1

Man B6 2

Praveen M42 3

1. **To print the first item along with the row number(NR) separated with ” – “ from each line in test.txt:**

$ awk '{print NR "- " $1 }' test.txt

1 - Tarun

2 – Manav

3 - Praveen

1. **To return the second row/item from test.txt:**

$ awk '{print $2}' test.txt

A12

B6

M42

1. **To print any non empty line if present**

$ awk 'NF > 0' test.txt

0

**4) To find the length of the longest line present in the file:**

$ awk '{ if (length($0) > max) max = length($0) } END { print max }' test.txt

13

1. **To count the lines in a file:**

$ awk 'END { print NR }' test.txt

3

1. **Printing lines with more than 10 characters:**

$ awk 'length($0) > 10' test.txt

Tarun A12 1

Praveen M42 3

1. **To find/check for any string in any column:**

$ awk '{ if($3 == "B6") print $0;}' test.txt

**8) To print the squares of first numbers from 1 to n say 6:**

$ awk 'BEGIN { for(i=1;i<=6;i++) print "square of", i, "is",i\*i; }'

square of 1 is 1

square of 2 is 4

square of 3 is 9

square of 4 is 16

square of 5 is 25

square of 6 is 36

| **ID** | **Name** | **Age** | **Company** | **Skill** |
| --- | --- | --- | --- | --- |
| **101** | **John** | **28** | **Apple** | **iOS** |
| **102** | **Chris** | **33** | **NextDoor** | **Android** |
| **103** | **Smith** | **45** | **Facebook** | **Haskell** |
| **104** | **Jack** | **33** | **Apple** | **Java** |
| **105** | **May** | **27** | **Google** | **Python** |