



CS5354

UNIX TOOL PROGRAMMING

Awk : An Advanced Filters

- A. Sawarkar

Name.

Made in 1977

Authors: **A**ho, **W**einberger and **K**ernighan (AWK).

Like sed, it combine features of several filters.

Unlike othres filters, it operates at ***filed*** level and can easily access , transform and format individual fileds in line.

It also support extended regular expression(ERE)

In awk we use ? And + for RE.

g+: g,gg,ggg.....

Knowing awk will help you understand perl, which uses most of the awk constructs.

Simple awk filtering

Syntax:

`awk options 'selection_criteria {action}' file(s)`

selection_criteria in **awk** have wider scope than in **sed**

e.g `$awk '/director/ {print}' emp.lst`

```
adhoc@adhoc:~/Desktop$ awk '/director/ {print}' emp.lst
9876 | jai sharma          | director | production | 12/03/50 | 7000
2365 | barun sengupta         | director | personnel  | 11/05/47 | 7800
1006 | chanchal singhvi       | director | sales      | 03/09/38 | 6700
6521 | lalit chowduryg*       | director | marketing  | 26/09/45 | 8200
adhoc@adhoc:~/Desktop$ awk '/director/ {p}' emp.lst
adhoc@adhoc:~/Desktop$
```

Note: **Printing** is default action of **awk**.

```
adhoc@adhoc:~/Desktop$ awk '/director/ ' emp.lst
9876 | jai sharma          | director | production | 12/03/50 | 7000
2365 | barun sengupta         | director | personnel  | 11/05/47 | 7800
1006 | chanchal singhvi       | director | sales      | 03/09/38 | 6700
6521 | lalit chowduryg*       | director | marketing  | 26/09/45 | 8200
adhoc@adhoc:~/Desktop$
```

Simple awk filtering


For **pattern** matching **awk** uses regular expression in **sed** style

e.g

```
$awk -F"|" ' /[Aa]gg*[ar][ar]wal/' emp.lst
```

-F"|": what filed seprator to use.

Note: Default awk work on line.



The screenshot shows a terminal window with a dark background. The prompt is `adhoc@adhoc:~/Desktop$`. The first command is `awk -F"|" ' /[Aa]gg*[ar][ar]wal/' emp.lst`, which filters the `emp.lst` file for lines containing 'agg' followed by zero or more 'ar's, then 'wal'. The output shows three lines of employee data. The second command is `awk ' /[Aa]gg*[ar][ar]wal/' emp.lst`, which uses the default field separator (space) and filters for the same pattern. The output is identical to the first command.

```
File Edit View Search Terminal Help
adhoc@adhoc:~/Desktop$ awk -F"|" ' /[Aa]gg*[ar][ar]wal/' emp.lst
2476 | anil aggarwal      | manager | sales      | 01/05/59 | 5000
3564 | sudhir Agarwal      | executive| personnel  | 06/07/47 | 7500
0110 | v.k.agrawal         | g.m.    | marketing  | 31/12/40 | 9000
adhoc@adhoc:~/Desktop$ awk ' /[Aa]gg*[ar][ar]wal/' emp.lst
2476 | anil aggarwal      | manager | sales      | 01/05/59 | 5000
3564 | sudhir Agarwal      | executive| personnel  | 06/07/47 | 7500
0110 | v.k.agrawal         | g.m.    | marketing  | 31/12/40 | 9000
adhoc@adhoc:~/Desktop$
```

Splitting a line into fields

Awk uses **\$0**, to indicate the entire line.

\$1 onwards is specific field

`$awk -F'|' '/sales/ {print $2,$3,$4,$6}' emp.lst`

```
3212 | shyam saksena | d.g.m. | accounts | 12/12/55 | 6000
3564 | sudhir Agarwal | executive | personnel | 06/07/47 | 7500
2345 | j. b. sexena | g.m. | marketing | 12/03/45 | 8000
0110 | v.k.agrawal | g.m. | marketing | 31/12/40 | 9000
adhoc@adhoc:~/Desktop$ awk -F'|' '/sales/ {print $2,$3,$4,$6}' emp.lst
a.k.shukla | g.m. | sales | 6000
chanchal singhvi | director | sales | 6700
s.n. dasgupta | manager | sales | 5600
anil aggarwal | manager | sales | 5000
adhoc@adhoc:~/Desktop$
```

For selecting lines use built-in variable **NR**(line number)

`$awk -F'|' 'NR == 3, NR == 6 {print NR, $2,$3,$6}' emp.lst`

```
3212 | shyam saksena | d.g.m. | accounts | 12/12/55 | 6000
3564 | sudhir Agarwal | executive | personnel | 06/07/47 | 7500
2345 | j. b. sexena | g.m. | marketing | 12/03/45 | 8000
0110 | v.k.agrawal | g.m. | marketing | 31/12/40 | 9000
adhoc@adhoc:~/Desktop$ awk -F'|' 'NR == 3, NR == 6 {print NR, $2,$3,$6}' emp.lst
3 sumit chakrobarty | d.g.m. | 6000
4 barun sengupta | director | 7800
5 n.k.gupta | chairman | 5400
6 chanchal singhvi | director | 6700
adhoc@adhoc:~/Desktop$
```

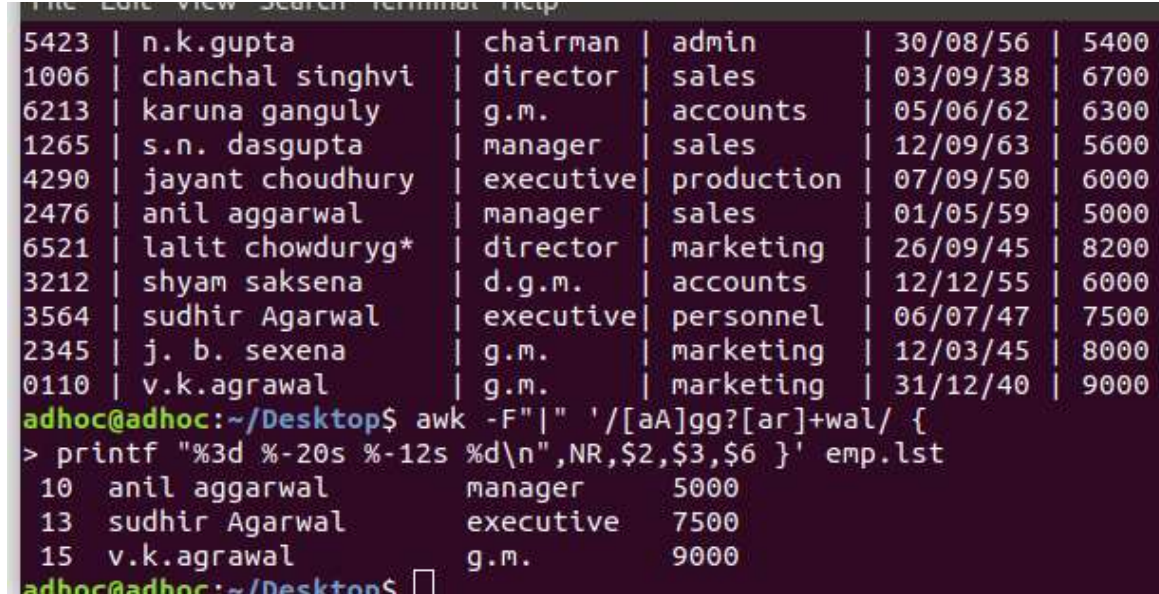

printf: Formatting output

Like a C printf statement , we can use in awk also

%s: format used for string data

%d: format used for numeric data

```
$awk -F'|' ' /[aA]gg?[ar]+wal/ {printf "%3d %-20s %-12s %d\n",  
NR,$2,$3,$6 }' emp.lst
```



```
5423 | n.k.gupta | chairman | admin | 30/08/56 | 5400  
1006 | chanchal singhvi | director | sales | 03/09/38 | 6700  
6213 | karuna ganguly | g.m. | accounts | 05/06/62 | 6300  
1265 | s.n. dasgupta | manager | sales | 12/09/63 | 5600  
4290 | jayant choudhury | executive | production | 07/09/50 | 6000  
2476 | anil aggarwal | manager | sales | 01/05/59 | 5000  
6521 | lalit chowduryg* | director | marketing | 26/09/45 | 8200  
3212 | shyam saksena | d.g.m. | accounts | 12/12/55 | 6000  
3564 | sudhir Agarwal | executive | personnel | 06/07/47 | 7500  
2345 | j. b. sexena | g.m. | marketing | 12/03/45 | 8000  
0110 | v.k.agrawal | g.m. | marketing | 31/12/40 | 9000  
adhoc@adhoc:~/Desktop$ awk -F'|' ' /[aA]gg?[ar]+wal/ {  
> printf "%3d %-20s %-12s %d\n",NR,$2,$3,$6 }' emp.lst  
10 anil aggarwal manager 5000  
13 sudhir Agarwal executive 7500  
15 v.k.agrawal g.m. 9000  
adhoc@adhoc:~/Desktop$
```

Note: (% -20s)- > total space 20 alloacte and -ve means left.

Variables and Expressions

Expressions comprise strings, numners, varaibles and operators.

Expr: $(a+b)*c+12$

Unlike in programming langauages, awk **doesn't** have char ,int, long ,double primitive data types.

Every expression can be interpreted either as string or numbers.

Awk also support user define varaibles which are case sensitive.

Comparison operators

Select name, designation and salary only for director or chairman

```
$awk -F"|" '$3 == "director" || $3 == "chairman" {  
    Printf "%-20s %-12s %d\n", $2,$3,$6 }' emp.lst
```

Select name, designation and salary of other than director and chairman

```
$awk -F"|" '$3 != "director" && $3 != "chairman" {  
    Printf "%-20s %-12s %d\n", $2,$3,$6 }' emp.lst
```


~ and !~operator

To locate only **g.m.**'s

```
$awk -F"|" ' $3 ~ g.m. { printf "....}" emp.lst
```

```
adhoc@adhoc:~/Desktop$ awk -F"|" ' $3 ~ /g.m./ ' emp.lst
2233 | a.k.shukla | g.m. | sales | 12/12/52 | 6000
5678 | sumit chakrobarty | d.g.m. | marketing | 19/04/43 | 6000
6213 | karuna ganguly | g.m. | accounts | 05/06/62 | 6300
3212 | shyam saksena | d.g.m. | accounts | 12/12/55 | 6000
2345 | j. b. sexena | g.m. | marketing | 12/03/45 | 8000
0110 | v.k.agrawal | g.m. | marketing | 31/12/40 | 9000
adhoc@adhoc:~/Desktop$
```

To locate only and only **g.m.**

```
$awk -F"|" ' $3 ~ /^g.m./ {print "....}" emp.lst
```

```
adhoc@adhoc:~/Desktop$ awk -F"|" ' $3 ~ /^g.m./ ' emp.lst
2233 | a.k.shukla | g.m. | sales | 12/12/52 | 6000
6213 | karuna ganguly | g.m. | accounts | 05/06/62 | 6300
2345 | j. b. sexena | g.m. | marketing | 12/03/45 | 8000
0110 | v.k.agrawal | g.m. | marketing | 31/12/40 | 9000
adhoc@adhoc:~/Desktop$
```

Number Comparsion

Operator	Significance
<	Less than
<=	Less than or equal to
==	Equal to
!=	Not equal to
>=	Greater than or equal to
>	Greater than
~	Matches a regular expression
!~	Doesn't match a regular expression

\$awk -F'|' '\$6 > 7500 {

Printf"%-20s %-12s %d\n",\$2,\$3,\$6 }' emp.lst

Select , either born in 1945 or salary greater than 8000

```
adhoc@adhoc:~$ cd Desktop
adhoc@adhoc:~/Desktop$ awk -F"|" ' $6 > 8000 || $5 ~ /45$/ ' emp.lst
6521 | lalit chowduryg* | director | marketing | 26/09/45 | 8200
0110 | v.k.agrawal | g.m. | marketing | 31/12/40 | 9000
adhoc@adhoc:~/Desktop$
```

Show only name , designation and salary whose salary greater than 7500.

$$\$awk -F'|' '{if ($6 > 7500 {$$

```
Printf “%-20s %-12s %d\n”, $2,$3,$6} emp.lst
```

```
adhoc@adhoc:~/Desktop$ awk -F"| " '$6 >7500 {  
> printf "%-20s %-12s %d\n", $2,$3,$6 }' emp.lst  
barun sengupta          director      7800  
lalit chowduryg*        director      8200  
j. b. sexena            g.m.         8000  
v.k.agrawal             g.m.         9000  
adhoc@adhoc:~/Desktop$ awk -F"| " '$6 >7500 {  
printf "%-20s %-12s %d", $2,$3,$6 }' emp.lst  
barun sengupta          director      7800 lalit chowduryg*    direc  
tor      8200 j. b. seadadhoc@adhoadhoc@adhoadadadadhoc@adadadadadadadadadhad  
adhoc@adhoc:~/Desktop$
```

Number Processing

Calculate HRA and DA as per condition

HRA= 40% of basic if basic >6500

DA =15% of basic if basic >6500

```
$awk -F"|" ' $6 > 6500 {
```

```
Printf "%-20s %-12s %d %d %d\n", $2,$3,$6,$6*0.4, $6*0.15}' emp.lst
```

```
adhoc@adhoc:~/Desktop$ awk -F"|" ' $6 > 6500 {
printf "%-20s %-12s %d %d %d\n", $2,$3,$6,$6*0.4, $6*0.15 }' emp.lst
jai sharma          director      7000 2800 1050
barun sengupta      director      7800 3120 1170
chanchal singhvi    director      6700 2680 1005
lalit chowduryg*    director      8200 3280 1230
sudhir Agarwal      executive     7500 3000 1125
j. b. sexena        g.m.         8000 3200 1200
v.k.agrawal         g.m.         9000 3600 1350
adhoc@adhoc:~/Desktop$
```

Variables

Select name , designation and salary whose basic is >7500 . Also show there counter.

```
adhoc@adhoc:~/Desktop$ awk -F"|" ' $6 > 7500 {  
count = count + 1  
printf "%3d %-20s %-12s $%d\n", count,$2,$3,$6}' emp.lst  
 1 barun sengupta          director      $7800  
 2 lalit chowduryg*        director      $8200  
 3 j. b. sexena            g.m.         $8000  
 4 v.k.agrawal             g.m.         $9000  
adhoc@adhoc:~/Desktop$
```

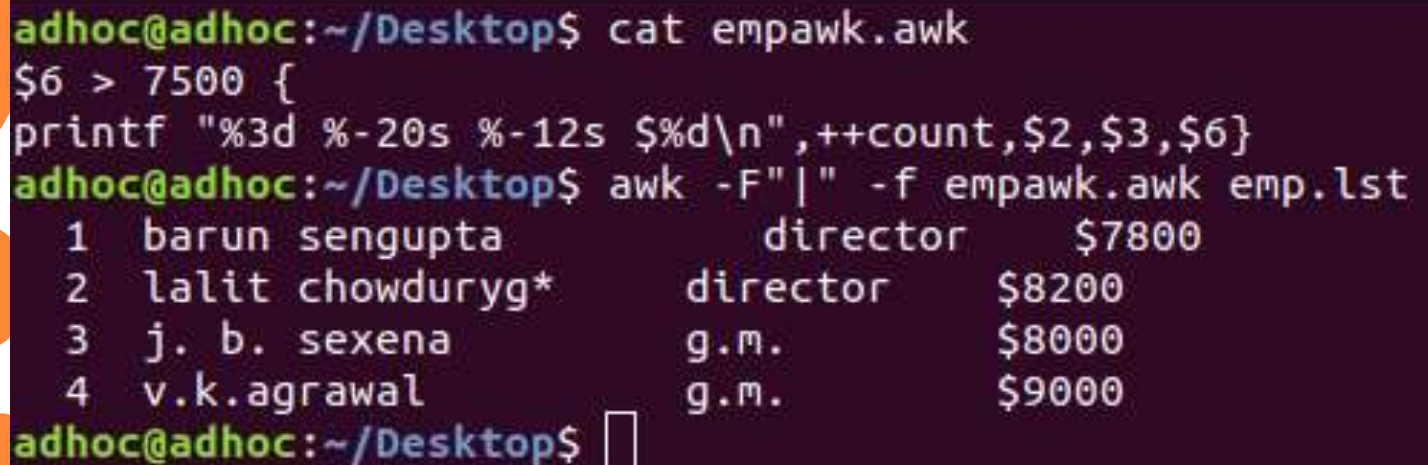
Note: Intial value of variable is 0 (by default)

-f : storing awk program in a file

By using -f we can directly take “selection_criteria and action” from file.
Dont use ‘ (single quotes) in file for criteria and action



```
empawk.awk
~/Desktop
Open [icon] Save [icon] [icon] [icon] [icon]
1 $6 > 7500 {
2 printf "%3d %-20s %-12s $%d\n", ++count, $2, $3, $6}
awk Tab Width: 8 Ln 2, Col 34 INS
```



```
adhoc@adhoc:~/Desktop$ cat empawk.awk
$6 > 7500 {
printf "%3d %-20s %-12s $%d\n", ++count, $2, $3, $6}
adhoc@adhoc:~/Desktop$ awk -F"|" -f empawk.awk emp.lst
 1 barun sengupta          director      $7800
 2 lalit chowduryg*        director      $8200
 3 j. b. sexena            g.m.         $8000
 4 v.k.agrawal             g.m.         $9000
adhoc@adhoc:~/Desktop$
```

BEGIN and END section

BEGIN AND END section are option .

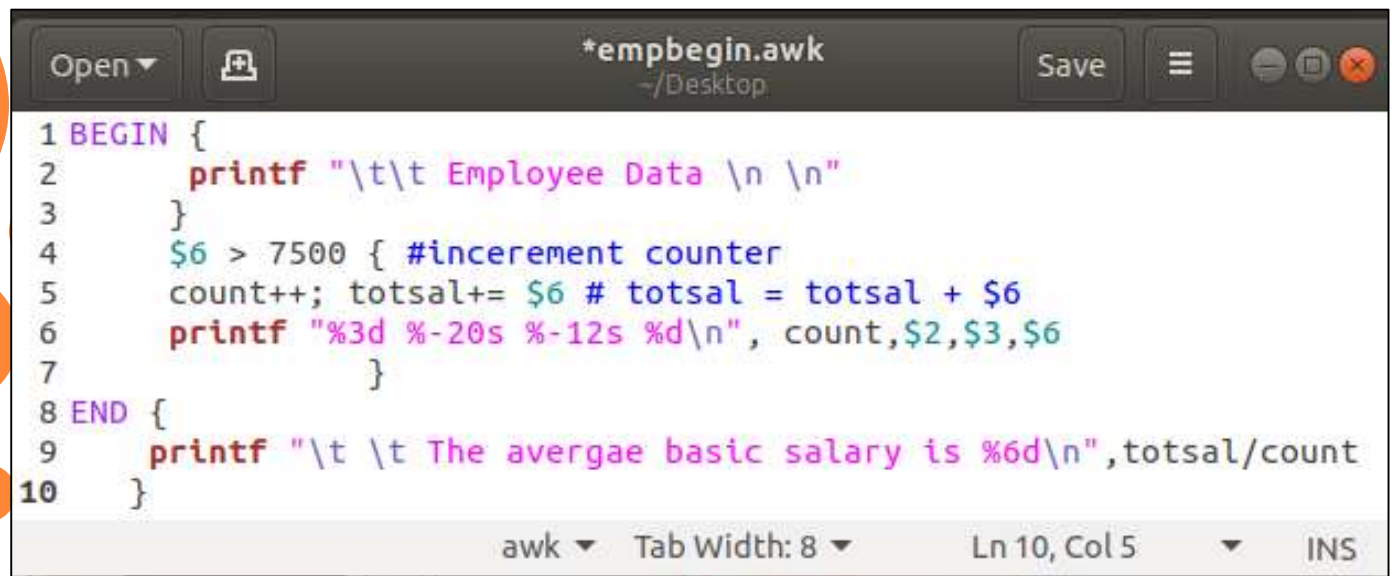
Syntax;

BEGIN { action }

END { action }

Want to show “Employee Data” at the begining of output and
“The average Basic salary is “ at the end of output.

```
$awk -F'|' -f empbegin.awk emp.lst
```



```
1 BEGIN {
2     printf "\t\t Employee Data \n \n"
3 }
4 $6 > 7500 { #incereament counter
5     count++; totalsal+= $6 # totalsal = totalsal + $6
6     printf "%3d %-20s %-12s %d\n", count,$2,$3,$6
7 }
8 END {
9     printf "\t \t The avergae basic salary is %6d\n",totalsal/count
10 }
```


BEGIN and END section

```
*empbegin.awk
~/Desktop

1 BEGIN {
2     printf "\t\t Employee Data \n \n"
3 }
4 $6 > 7500 { #incereement counter
5     count++; totals+= $6 # totals = totals + $6
6     printf "%3d %-20s %-12s %d\n", count,$2,$3,$6
7 }
8 END {
9     printf "\t \t The avergae basic salary is %6d\n",totals/count
10 }
```

awk Tab Width: 8 Ln 10, Col 5 INS

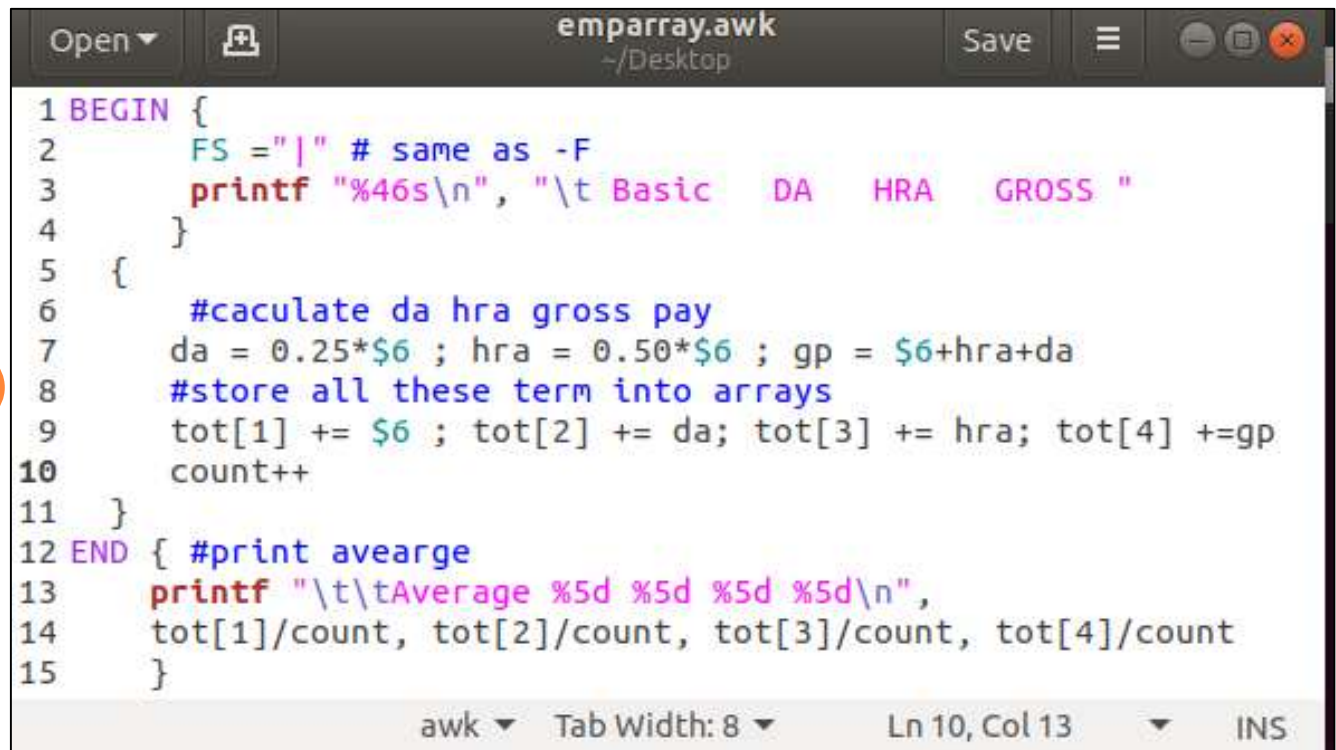
```
dhoc@adhoc:~/Desktop$ awk -F"|" -f empbegin.awk emp.lst
Employee Data

1 barun sengupta          director      7800
2 lalit chowduryg*       director      8200
3 j. b. sexena            g.m.         8000
4 v.k.agrawal             g.m.         9000
                        The avergae basic salary is      8250
dhoc@adhoc:~/Desktop$
```

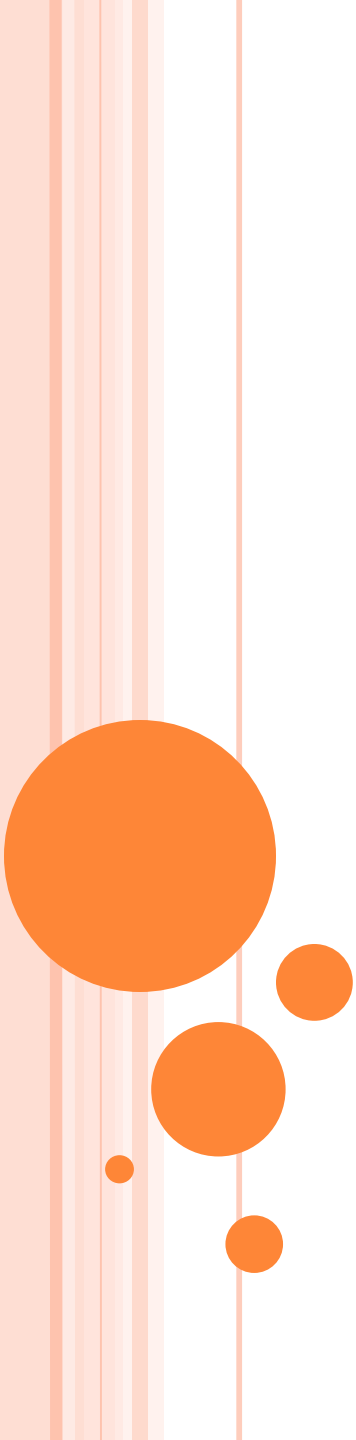
ARRAYS

Awk arrays are different from those regular programming array in many respects:

- .They are not formally defined, Declare a moment it is used
- .Array elements are initialized to zero or an empty string
- .Arrays expand automatically
- .The index can be virtually anything, it can even be a string



```
1 BEGIN {
2     FS = "|" # same as -F
3     printf "%46s\n", "\t Basic    DA    HRA    GROSS "
4 }
5 {
6     #caculate da hra gross pay
7     da = 0.25*$6 ; hra = 0.50*$6 ; gp = $6+hra+da
8     #store all these term into arrays
9     tot[1] += $6 ; tot[2] += da; tot[3] += hra; tot[4] +=gp
10    count++
11 }
12 END { #print avearge
13     printf "\t\tAverage %5d %5d %5d %5d\n",
14         tot[1]/count, tot[2]/count, tot[3]/count, tot[4]/count
15 }
```



```
emparray.awk
~/Desktop

1 BEGIN {
2     FS = "|" # same as -F
3     printf "%46s\n", "\t Basic    DA    HRA    GROSS "
4 }
5 {
6     #caculate da hra gross pay
7     da = 0.25*$6 ; hra = 0.50*$6 ; gp = $6+hra+da
8     #store all these term into arrays
9     tot[1] += $6 ; tot[2] += da; tot[3] += hra; tot[4] +=gp
10    count++
11 }
12 END { #print avearge
13     printf "\t\tAverage %5d %5d %5d %5d\n",
14     tot[1]/count, tot[2]/count, tot[3]/count, tot[4]/count
15 }
```

awk Tab Width: 8 Ln 10, Col 13 INS

```
adhoc@adhoc: ~/Desktop

File Edit View Search Terminal Help
adhoc@adhoc:~/Desktop$ awk -f emparray.awk emp.lst
                Basic    DA    HRA    GROSS
        Average  6700   1675   3350  11725
adhoc@adhoc:~/Desktop$
```

Function

Awk has several built-in functions

As `length()`, `index(s1,s2)`, `substr(str,m,n)`, `split(str, arr,ch)`, `system()`, etc

length(): determine the length of argument, if no argument then entire line assume to be argument

```
adhoc@adhoc:~/Desktop$ awk -F'|' 'length > 1024' emp.lst
adhoc@adhoc:~/Desktop$ awk -F'|' 'length > 100' emp.lst
adhoc@adhoc:~/Desktop$ awk -F'|' 'length > 70' emp.lst
adhoc@adhoc:~/Desktop$ awk -F'|' 'length > 66' emp.lst
adhoc@adhoc:~/Desktop$ awk -F'|' 'length > 65' emp.lst
2233 | a.k.shukla | g.m. | sales | 12/12/52 | 6000
9876 | jai sharma | director | production | 12/03/50 | 7000
5678 | sumit chakrobarty | d.g.m. | marketing | 19/04/43 | 6000
1006 | chanchal singhvi | director | sales | 03/09/38 | 6700
6213 | karuna ganguly | g.m. | accounts | 05/06/62 | 6300
1265 | s.n. dasgupta | manager | sales | 12/09/63 | 5600
4290 | jayant choudhury | executive | production | 07/09/50 | 6000
2476 | anil aggarwal | manager | sales | 01/05/59 | 5000
6521 | lalit chowduryg* | director | marketing | 26/09/45 | 8200
3212 | shyam saksena | d.g.m. | accounts | 12/12/55 | 6000
3564 | sudhir Agarwal | executive | personnel | 06/07/47 | 7500
2345 | j. b. sexena | g.m. | marketing | 12/03/45 | 8000
0110 | v.k.agrawal | g.m. | marketing | 31/12/40 | 9000
adhoc@adhoc:~/Desktop$
```

\$awk -F'|' 'length(\$2) < 11' emp.lst

```
0110 | v.k.agrawal | g.m. | marketing | 31/12/40 | 9000
adhoc@adhoc:~/Desktop$ awk -F'|' 'length($2) < 11' emp.lst
adhoc@adhoc:~/Desktop$ awk -F'|' 'length($2) < 15' emp.lst
5423 | n.k.gupta | chairman | admin | 30/08/56 | 5400
adhoc@adhoc:~/Desktop$
```


Function

index(s1,s2): determine position of a string s2 within larger string s1.

```
File Edit View Search Terminal Help
adhoc@adhoc:~/Desktop$ awk -F|" 'x= index($2,"gu") {printf "%s\n", x }' emp.lst
11
6
12
10
adhoc@adhoc:~/Desktop$
```

substr(str,m,n): extract substring from string **str**, **m** starting and **n** indicate number of character to be extracted.

```
3564 | sudhir Agarwal | executive| personnel | 06/07/47 | 7500
adhoc@adhoc:~/Desktop$ awk -F|" 'substr($5,8,2) > 45 && substr($5,8,2) < 52' emp.lst
9876 | jai sharma | director | production | 12/03/50 | 7000
2365 | barun sengupta | director | personnel | 11/05/47 | 7800
4290 | jayant choudhury | executive| production | 07/09/50 | 6000
3564 | sudhir Agarwal | executive| personnel | 06/07/47 | 7500
adhoc@adhoc:~/Desktop$
```

Function

split(str,arr,ch): breakup a string str on the delimiter ch and store the fields in an array arr[].

```
$ awk -F\| '{split($5,ar,"/"); printf "%19s"ar[3] " "ar[2]" " ar[1] " \n"}' emp.lst
```

```
adhoc@adhoc:~/Desktop$ cat emp.lst
2233 | a.k.shukla | g.m. | sales | 12/12/52 | 6000
9876 | jai sharma | director | production | 12/03/50 | 7000
5678 | sumit chakrobarty | d.g.m. | marketing | 19/04/43 | 6000
2365 | barun sengupta | director | personnel | 11/05/47 | 7800
5423 | n.k.gupta | chairman | admin | 30/08/56 | 5400
1006 | chanchal singhvi | director | sales | 03/09/38 | 6700
6213 | karuna ganguly | g.m. | accounts | 05/06/62 | 6300
1265 | s.n. dasgupta | manager | sales | 12/09/63 | 5600
4290 | jayant choudhury | executive | production | 07/09/50 | 6000
2476 | anil aggarwal | manager | sales | 01/05/59 | 5000
6521 | lalit chowduryg* | director | marketing | 26/09/45 | 8200
3212 | shyam saksena | d.g.m. | accounts | 12/12/55 | 6000
3564 | sudhir Agarwal | executive | personnel | 06/07/47 | 7500
2345 | j. b. sexena | g.m. | marketing | 12/03/45 | 8000
0110 | v.k.agrawal | g.m. | marketing | 31/12/40 | 9000
adhoc@adhoc:~/Desktop$ awk -F\| '{split($5,ar,"/"); printf "%19s"ar[3] " "ar[2]" " ar[1] " \n"}' emp.lst
1952 12 12
1950 03 12
1943 04 19
1947 05 11
1956 08 30
1938 09 03
1962 06 05
```

Control Flow

if_else

```
If (condition is true) {  
Statement  
} else {  
Statement  
}
```

e.g: `$awk -F"|" '{if($6>7500) printf "
If (NR>=3 && NR<=6)
....`

Note: Conditional statement (ternary operator)

```
If($6<6000)  
da=0.25*$6  
else  
da=1000
```

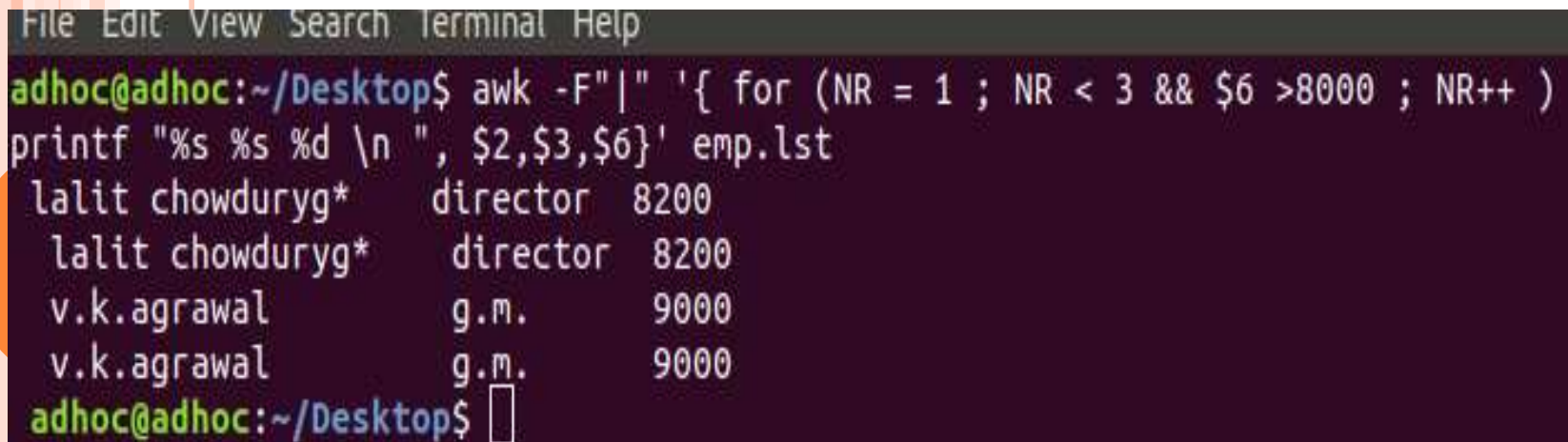
Or `$6<6000 ? da=0.25*$6 : da =1000`

Control Flow

LOOP: for, while

For (i=1 ; i<=5 ; i++)

```
$awk -F"|" '{ for (NR = 1 ; NR < 3 && $6 >8000 ; NR++ )  
printf "%s %s %d \n ", $2,$3,$6}' emp.lst
```



```
File Edit View Search Terminal Help  
adhoc@adhoc:~/Desktop$ awk -F"|" '{ for (NR = 1 ; NR < 3 && $6 >8000 ; NR++ )  
printf "%s %s %d \n ", $2,$3,$6}' emp.lst  
lalit chowduryg*    director    8200  
lalit chowduryg*    director    8200  
v.k.agrawal        g.m.        9000  
v.k.agrawal        g.m.        9000  
adhoc@adhoc:~/Desktop$
```

Control Flow

while

K=0

While(condition) {

Statement

Updation

}

`$awk -F"|" '{NR = 1`

`> while (NR < 3 && $6 > 8000)`

`> { printf "%s %s %d \n", $2,$3,$6 NR++ } }' emp.lst`

```
adhoc@adhoc:~/Desktop$ awk -F"|" '{NR = 1
> while ( NR < 3 && $6 > 8000)
> { printf "%s %s %d \n", $2,$3,$6 NR++ } }' emp.lst
lalit chowduryg*      director      82001
lalit chowduryg*      director      82002
v.k.agrawal           g.m.         90001
v.k.agrawal           g.m.         90002
adhoc@adhoc:~/Desktop$
```