CenturyLink™

AWK SCRIPTING

Subhasis Samantaray

What is awk?

Created at Bell Lab by: Alfred **A**ho, Peter **W**einberger, and Brian **K**ernighan

AWK is a scripting language used for manipulating data and generating reports.

Variants of awk:

awk, nawk, mawk, pgawk, …

GNU awk: gawk

AWK : Input File

	$1		$2		$3		$4		$5		$6	..... $NF
	FIELD/COLUMN		FIELD/COLUMN		FIELD/COLUMN		FIELD/COLUMN		FIELD/COLUMN		FIELD/COLUMN	
LINE / RECORD		FS		FS		FS		FS		FS		
LINE / RECORD		FS		FS		FS		FS		FS		
LINE / RECORD		FS		FS		FS		FS		FS		
LINE / RECORD		FS		FS		FS		FS		FS		
LINE / RECORD		FS		FS		FS		FS		FS		
LINE / RECORD		FS		FS		FS		FS		FS		
LINE / RECORD		FS		FS		FS		FS		FS		
LINE / RECORD		FS		FS		FS		FS		FS		
LINE / RECORD		FS		FS		FS		FS		FS		

Each record/line contains **'n'** number of fields separated by Filed separator (**FS**). Space/TAB is the default filed separator

AWK supports two types of buffers: record and field

FIELD BUFFER: one for each fields in the current record.
$1 – First Field
$2 – Second filed …….. **$NF** – Last Field

RECORD BUFFER: **$0** holds the entire record

CenturyLink™

Example : Input File separated by | (pipeline)

sxsama2@HMLINUX1:~/AWK

```
[sxsama2@HMLINUX1 AWK]$ cat EMP.DAT
7369|SMITH|CLERK|7902|17-DEC-80|800||20
7499|ALLEN|SALESMAN|7698|20-FEB-81|1600|300|30
7521|WARD|SALESMAN|7698|22-FEB-81|1250|500|30
7566|JONES|MANAGER|7839|02-APR-81|2975||20
7654|MARTIN|SALESMAN|7698|28-SEP-81|1250|1400|30
7698|BLAKE|MANAGER|7839|01-MAY-81|2850||30
7782|CLARK|MANAGER|7839|09-JUN-81|2450||10
7788|SCOTT|ANALYST|7566|19-APR-87|3000||20
7839|KING|PRESIDENT||17-NOV-81|5000||10
7844|TURNER|SALESMAN|7698|08-SEP-81|1500|0|30
7876|ADAMS|CLERK|7788|23-MAY-87|1100||20
7900|JAMES|CLERK|7698|03-DEC-81|950||30
7902|FORD|ANALYST|7566|03-DEC-81|3000||20
7934|MILLER|CLERK|7782|23-JAN-82|1300||10
[sxsama2@HMLINUX1 AWK]$
```

Example : Input File separated by , (comma)

sxsama2@HMLINUX1:~/AWK

```
[sxsama2@HMLINUX1 AWK]$ cat EMP.CSV
7369,SMITH,CLERK,7902,17-DEC-80,800,,20
7499,ALLEN,SALESMAN,7698,20-FEB-81,1600,300,30
7521,WARD,SALESMAN,7698,22-FEB-81,1250,500,30
7566,JONES,MANAGER,7839,02-APR-81,2975,,20
7654,MARTIN,SALESMAN,7698,28-SEP-81,1250,1400,30
7698,BLAKE,MANAGER,7839,01-MAY-81,2850,,30
7782,CLARK,MANAGER,7839,09-JUN-81,2450,,10
7788,SCOTT,ANALYST,7566,19-APR-87,3000,,20
7839,KING,PRESIDENT,,17-NOV-81,5000,,10
7844,TURNER,SALESMAN,7698,08-SEP-81,1500,0,30
7876,ADAMS,CLERK,7788,23-MAY-87,1100,,20
7900,JAMES,CLERK,7698,03-DEC-81,950,,30
7902,FORD,ANALYST,7566,03-DEC-81,3000,,20
7934,MILLER,CLERK,7782,23-JAN-82,1300,,10
[sxsama2@HMLINUX1 AWK]$
```

AWK SYNTAX

```
awk [options] 'SEARCH CRITERIA{ACTION}'

awk [options] awk_script_file INPUT_FILE(s)
```

Options:

-F	Is used to specify the input field separator
-f	Is used to specify the name of awk script file

How awk works ?

- Awk reads the input files one record /line at a time.
- For each record , it matches with given pattern .It performs the corresponding action for the matching record(s) else no action will be performed .
- Either search pattern or action are optional.
- If the search pattern is not specified, then Awk performs the defined actions for each record .
- Print is the default action
- If the action is not given, print all that lines that matches with the given pattern(s)
- Empty braces with out any action does nothing. It wont perform default printing operation.
- Each statement in Actions should be delimited by semicolon.

Basic awk Program

Syntax:

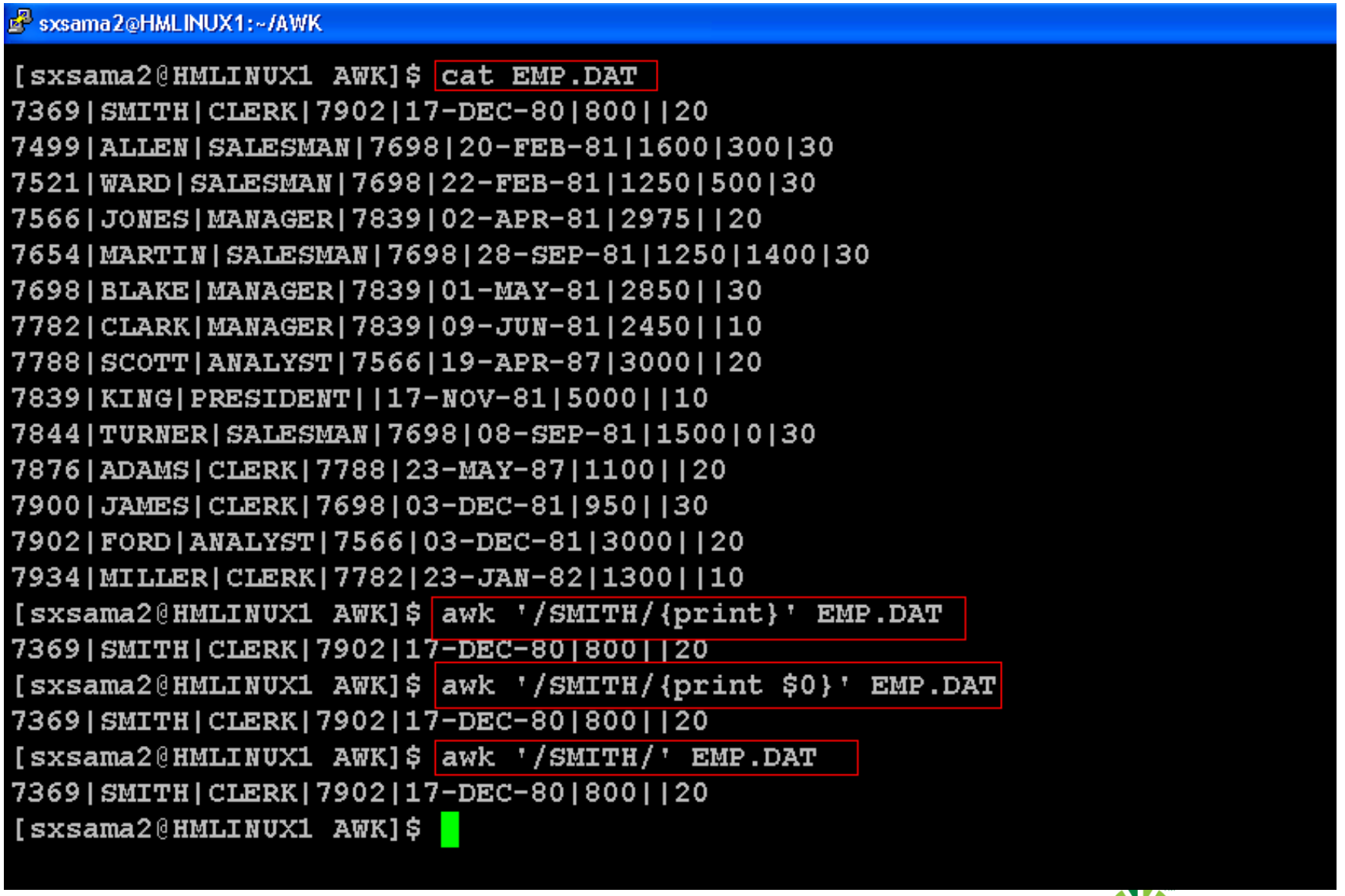
```
awk 'pattern {action}' FILENAME
```

- if pattern is missing, action is applied to all lines
- print is the default action
- if action is missing, the matched line is printed
- must have either pattern or action

Example:

```
awk '/SMITH/{print}' EMP.DAT
awk '/SMITH/' EMP.DAT
awk '/SMITH/{print $0}' EMP.DAT
```

Basic awk Program



Arithmetic Expressions

Operator	MEANING
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo
Space	Concatenation

Unary arithmetic operators :

The "+" and "-" operators can be used before variables and numbers

Auto increment and Auto decrement :

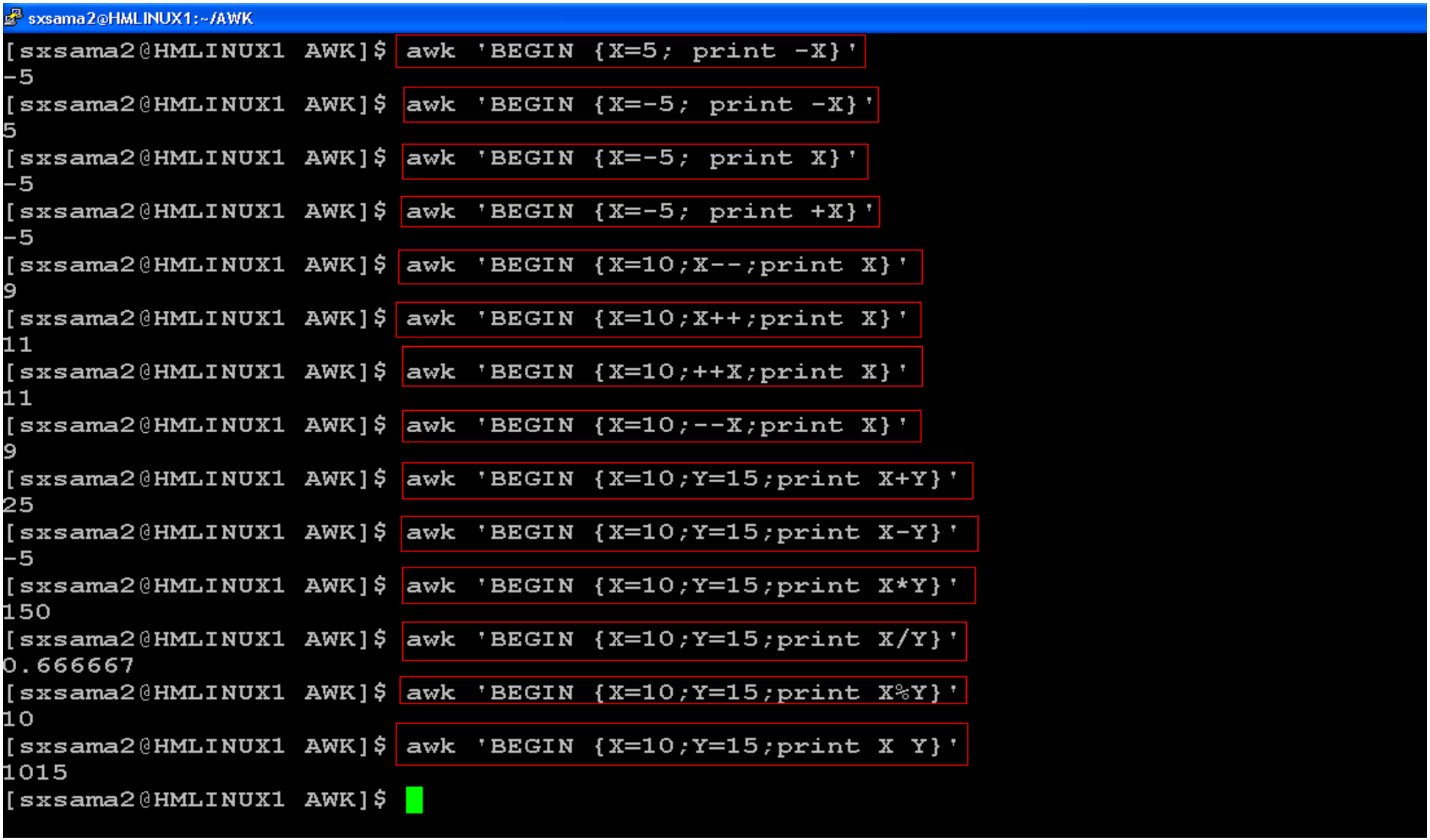
AWK supports the "++" and "--" operators to auto increment and auto decrement the value respectively.

Arithmetic Expressions

Example :

```
awk 'BEGIN {X=5; print -X}'
awk 'BEGIN {X=-5; print -X}'
awk 'BEGIN {X=-5; print X}'
awk 'BEGIN {X=-5; print +X}'
awk 'BEGIN {X=10;X--;print X}'
awk 'BEGIN {X=10;X++;print X}'
awk 'BEGIN {X=10;++X;print X}'
awk 'BEGIN {X=10;--X;print X}'
awk 'BEGIN {X=10;Y=15;print X+Y}'
awk 'BEGIN {X=10;Y=15;print X-Y}'
awk 'BEGIN {X=10;Y=15;print X*Y}'
awk 'BEGIN {X=10;Y=15;print X/Y}'
awk 'BEGIN {X=10;Y=15;print X%Y}'
awk 'BEGIN {X=10;Y=15;print X Y}'
```

Arithmetic Expressions



Conditional expressions AND Regular expressions

Conditional Expressions:

The following operators are used to compare strings to regular expressions.

OPERATOR	MEANING
==	Is equal
!=	Is not equal to
>	Is greater than
>=	Is greater than or equal to
<	Is less than
<=	Is less than or equal to

Regular Expressions:

The following operators are used to compare strings to regular expressions.

OPERATOR	MEANING
~	Matches
!~	Doesn't match

How to filter line based on search pattern ?

To search for lines based on search pattern ,we generally use grep/egrep .

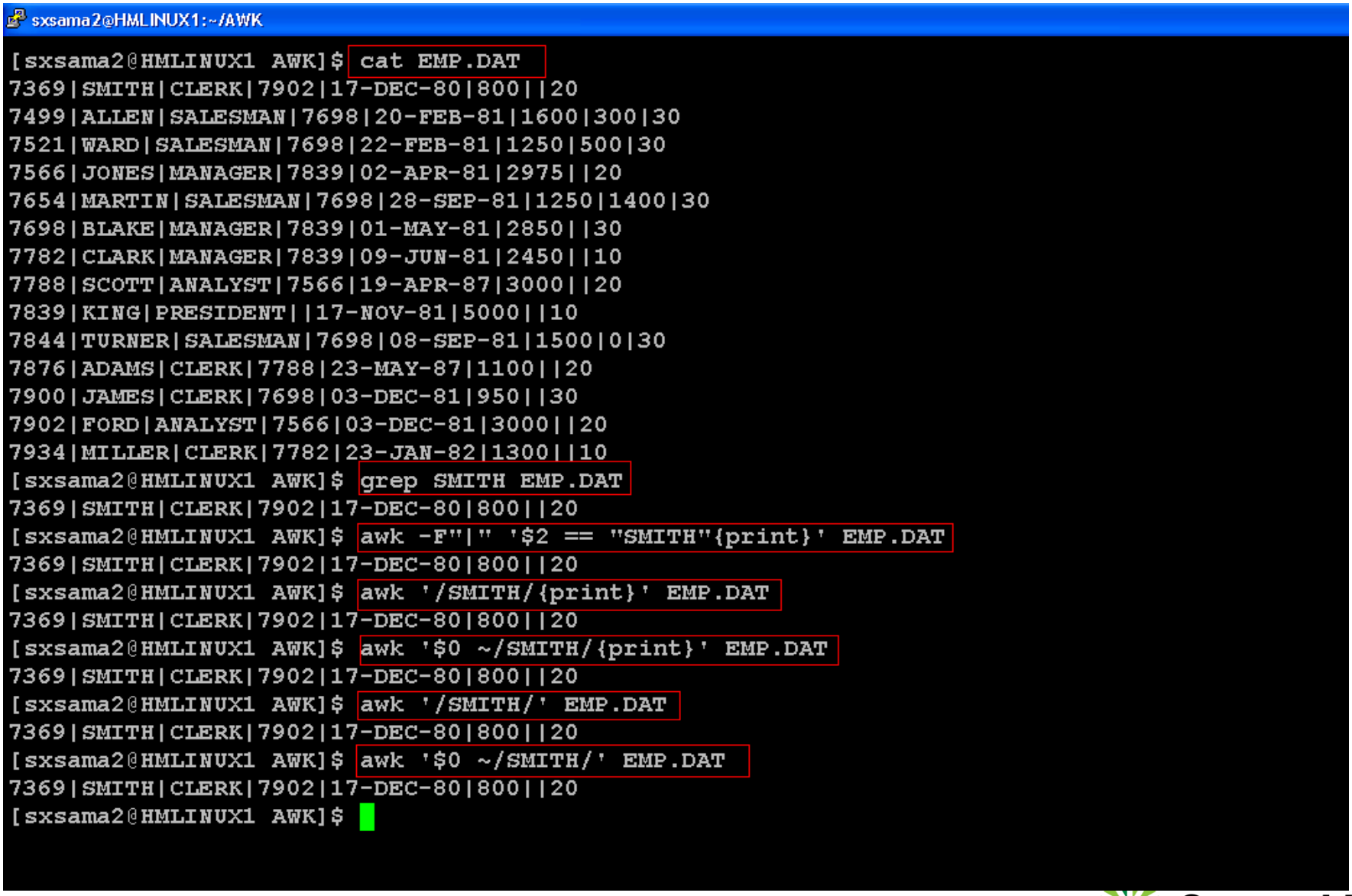
Example :

grep SMITH EMP.DAT

The awk one liner to filter the lines which contains SMITH :

awk '/SMITH/{print}' EMP.DAT
awk '$0 ~/SMITH/{print}' EMP.DAT
awk '/SMITH/' EMP.DAT
awk '$0 ~/SMITH/' EMP.DAT

How to filter line based on search pattern ?



How to get first n lines from a file ?

To get the first n lines from a file generally we use head command.

Example :

head -5 EMP.DAT

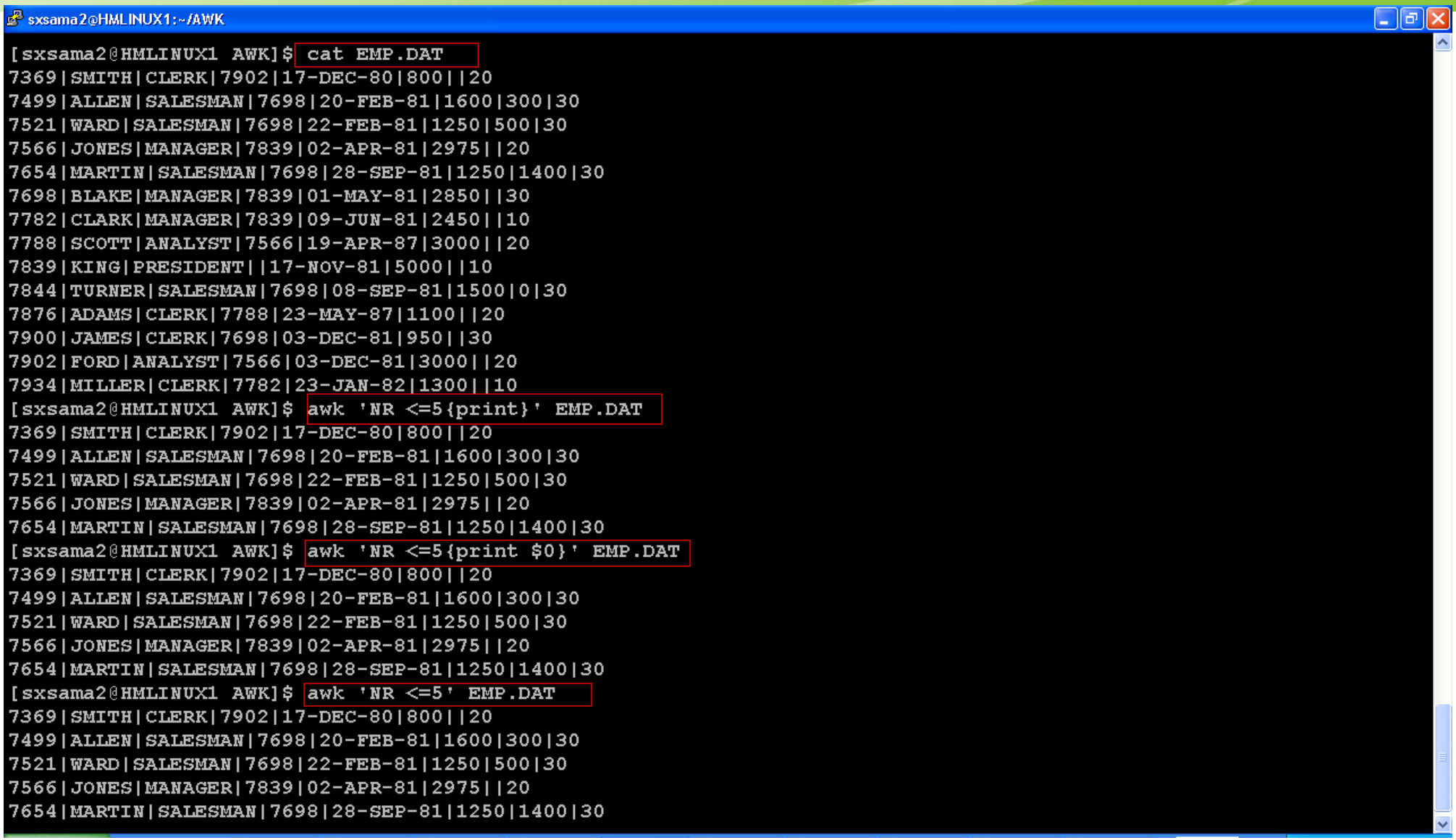
The awk oneliner to get the first 5 lines :

awk ‘NR <=5 {print}‘ EMP.DAT

awk ‘NR <=5 ‘ EMP.DAT

NR represents the record/line number

How to get first n lines from a file ?



CenturyLink™

How to print the Line number at the starting of each record ?

Usually we use cat ,nl command to add the line numbers .

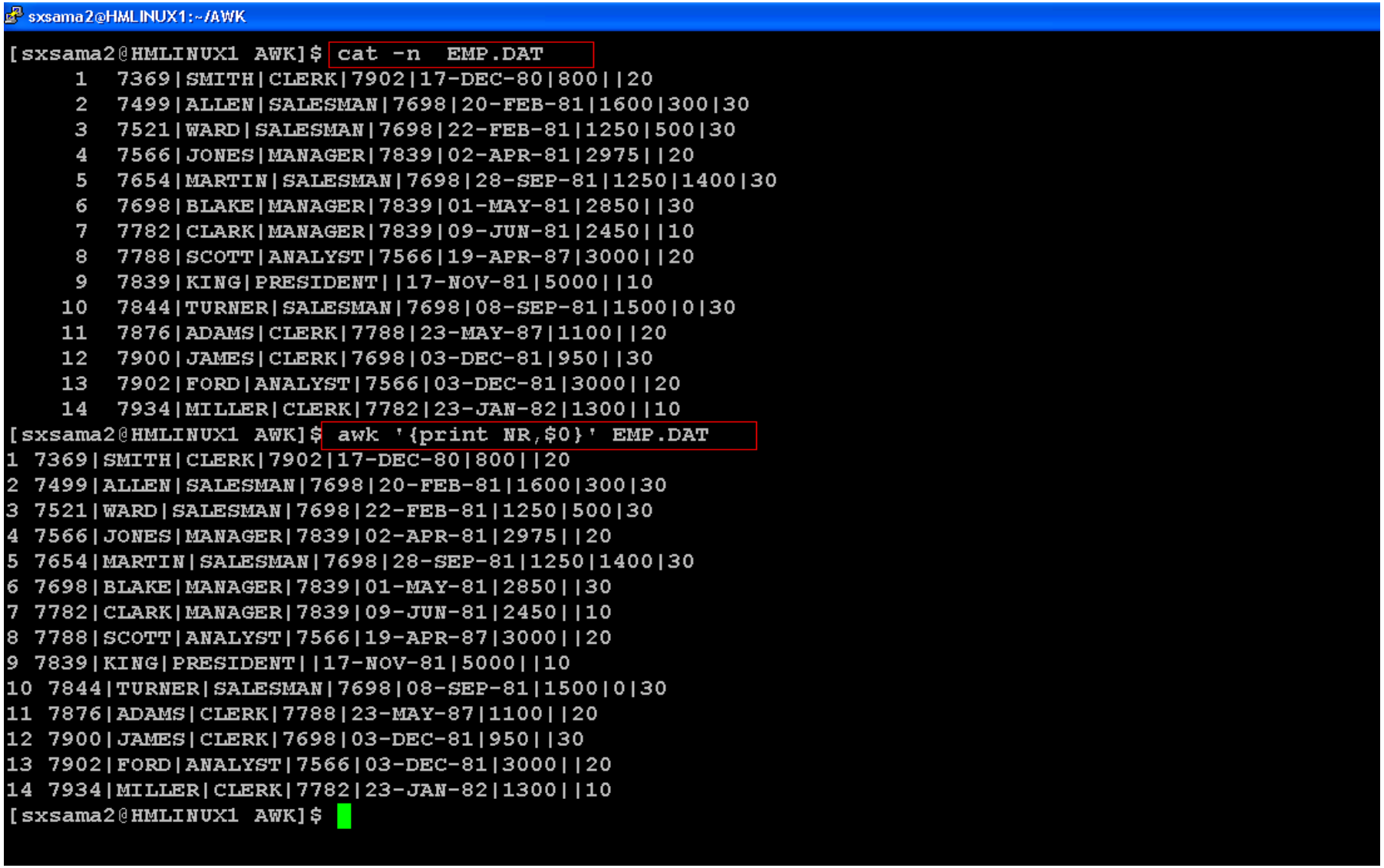
Example :

```
cat –n EMP.DAT
nl EMP.DAT
```

The awk one liner to add the line numbers :

```
awk '{ print  NR,$0}' EMP.DAT
```

How to print the Line number at the starting of each record ?



CenturyLink™

How to get the range of Lines/records from a FILE?

To get the range of records/lines from a file generally we use the combination of head & tail command.

Example: To get the line number 5 – 10 from a File :

```
head -10 EMP.DAT |tail +5
```

The awk one liner to get the record number 5 -10 from a file :

```
awk 'NR >=5 && NR <=10 {print NR,$0}' EMP.DAT
awk 'NR >=5 && NR <=10 {print}' EMP.DAT
awk 'NR >=5 && NR <=10' EMP.DAT
```

How to get the range of Lines/records from a FILE?

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ awk 'NR >=5 && NR <=10 {print NR,$0}' EMP.DAT
5 7654|MARTIN|SALESMAN|7698|28-SEP-81|1250|1400|30
6 7698|BLAKE|MANAGER|7839|01-MAY-81|2850||30
7 7782|CLARK|MANAGER|7839|09-JUN-81|2450||10
8 7788|SCOTT|ANALYST|7566|19-APR-87|3000||20
9 7839|KING|PRESIDENT||17-NOV-81|5000||10
10 7844|TURNER|SALESMAN|7698|08-SEP-81|1500|0|30
[sxsama2@HMLINUX1 AWK]$ awk 'NR >=5 && NR <=10 {print}' EMP.DAT
7654|MARTIN|SALESMAN|7698|28-SEP-81|1250|1400|30
7698|BLAKE|MANAGER|7839|01-MAY-81|2850||30
7782|CLARK|MANAGER|7839|09-JUN-81|2450||10
7788|SCOTT|ANALYST|7566|19-APR-87|3000||20
7839|KING|PRESIDENT||17-NOV-81|5000||10
7844|TURNER|SALESMAN|7698|08-SEP-81|1500|0|30
[sxsama2@HMLINUX1 AWK]$ awk 'NR >=5 && NR <=10' EMP.DAT
7654|MARTIN|SALESMAN|7698|28-SEP-81|1250|1400|30
7698|BLAKE|MANAGER|7839|01-MAY-81|2850||30
7782|CLARK|MANAGER|7839|09-JUN-81|2450||10
7788|SCOTT|ANALYST|7566|19-APR-87|3000||20
7839|KING|PRESIDENT||17-NOV-81|5000||10
7844|TURNER|SALESMAN|7698|08-SEP-81|1500|0|30
[sxsama2@HMLINUX1 AWK]$ awk 'NR==1 || NR ==2' EMP.DAT
7369|SMITH|CLERK|7902|17-DEC-80|800||20
7499|ALLEN|SALESMAN|7698|20-FEB-81|1600|300|30
[sxsama2@HMLINUX1 AWK]$ awk 'NR==1 || NR ==2{print NR,$0}' EMP.DAT
1 7369|SMITH|CLERK|7902|17-DEC-80|800||20
2 7499|ALLEN|SALESMAN|7698|20-FEB-81|1600|300|30
[sxsama2@HMLINUX1 AWK]$
```

Logical Operators

OPERATOR	MEANING
&&	Logical AND
||	Logical OR
!	NOT

Examples:

```
awk -F"|" '$3=="MANAGER" || $3=="ANALYST"' EMP.DAT

awk -F"|" '$3=="MANAGER" && $4=="7839"{print}' EMP.DAT

awk -F"|" '$3!="MANAGER" && $3!="SALESMAN"{print $0}' EMP.DAT

awk -F"|" '$3!="MANAGER"' EMP.DAT
```

Logical Operators

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ awk -F"|" '$3=="MANAGER" || $3=="ANALYST"' EMP.DAT
7566|JONES|MANAGER|7839|02-APR-81|2975||20
7698|BLAKE|MANAGER|7839|01-MAY-81|2850||30
7782|CLARK|MANAGER|7839|09-JUN-81|2450||10
7788|SCOTT|ANALYST|7566|19-APR-87|3000||20
7902|FORD|ANALYST|7566|03-DEC-81|3000||20
[sxsama2@HMLINUX1 AWK]$ awk -F"|" '$3=="MANAGER" && $4=="7839"{print}' EMP.DAT
7566|JONES|MANAGER|7839|02-APR-81|2975||20
7698|BLAKE|MANAGER|7839|01-MAY-81|2850||30
7782|CLARK|MANAGER|7839|09-JUN-81|2450||10
[sxsama2@HMLINUX1 AWK]$ awk -F"|" '$3!="MANAGER" && $3!="SALESMAN"{print $0}' EMP.DAT
7369|SMITH|CLERK|7902|17-DEC-80|800||20
7788|SCOTT|ANALYST|7566|19-APR-87|3000||20
7839|KING|PRESIDENT||17-NOV-81|5000||10
7876|ADAMS|CLERK|7788|23-MAY-87|1100||20
7900|JAMES|CLERK|7698|03-DEC-81|950||30
7902|FORD|ANALYST|7566|03-DEC-81|3000||20
7934|MILLER|CLERK|7782|23-JAN-82|1300||10
[sxsama2@HMLINUX1 AWK]$ awk -F"|" '$3!="MANAGER"' EMP.DAT
7369|SMITH|CLERK|7902|17-DEC-80|800||20
7499|ALLEN|SALESMAN|7698|20-FEB-81|1600|300|30
7521|WARD|SALESMAN|7698|22-FEB-81|1250|500|30
7654|MARTIN|SALESMAN|7698|28-SEP-81|1250|1400|30
7788|SCOTT|ANALYST|7566|19-APR-87|3000||20
7839|KING|PRESIDENT||17-NOV-81|5000||10
7844|TURNER|SALESMAN|7698|08-SEP-81|1500|0|30
7876|ADAMS|CLERK|7788|23-MAY-87|1100||20
7900|JAMES|CLERK|7698|03-DEC-81|950||30
7902|FORD|ANALYST|7566|03-DEC-81|3000||20
7934|MILLER|CLERK|7782|23-JAN-82|1300||10
[sxsama2@HMLINUX1 AWK]$
```

Awk programming

awk scripts/programs are divided into three major blocks

BEGIN {ACTION }	**PRE PROCESSING**
PATTERN{ACTION} PATTERN{ACTION} PATTERN{ACTION} PATTERN{ACTION} PATTERN{ACTION}	**BODY**
END {ACTION }	**POST PROCESSING**

- Actions specified in BEGIN block will be executed before program starts reading the lines from the standard input /input file.

- Actions specified in END block will be executed after completing the reading and processing the lines from the standard input /input file.

- Actions specified in the BODY will be executed for each record .

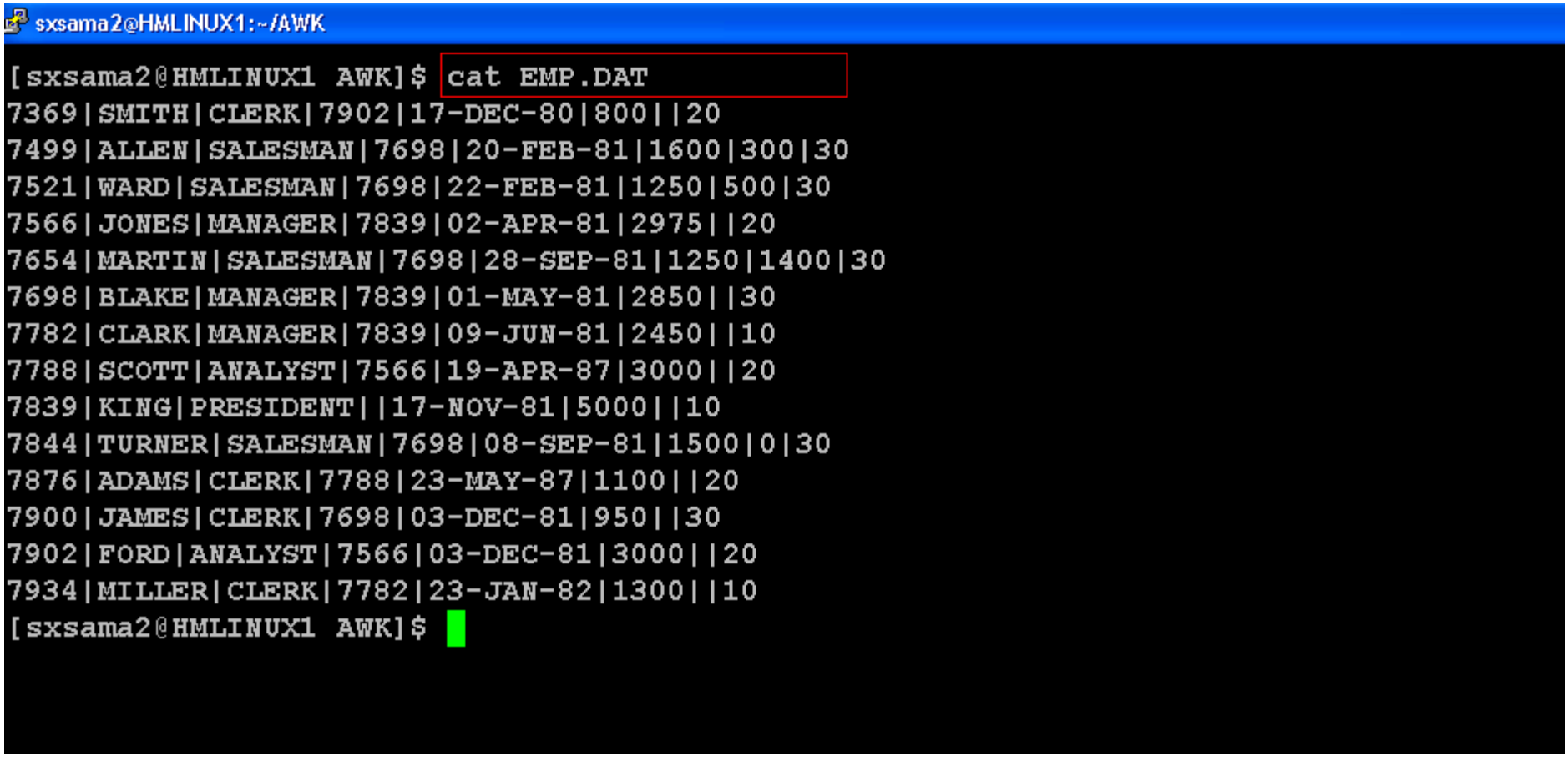
Basic AWK program/script

```
$ awk -F"|" 'BEGIN{print "EMPID \t NAME \t JOIN_DATE"} \
{print $1,"\t",$2,"\t",$5;} \
END{ print "------- REPORT ----------";}' EMP.DAT

$ cat EMP_REPORT.AWK
BEGIN{print "EMPID \t NAME \t JOIN_DATE"}
{print $1,"\t",$2,"\t",$5;}
END{ print "------- REPORT ----------";}

$ awk -F"|" -f EMP_REPORT.AWK EMP.DAT
```

Basic AWK program/script - EXAMPLE



CenturyLink™

Basic AWK program/script - EXAMPLE

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ awk -F"|" 'BEGIN{print "EMPID \t NAME \t JOIN_DATE"}
> {print $1,"\t",$2,"\t",$5;}
> END{ print "------- REPORT ----------";}' EMP.DAT
EMPID     NAME     JOIN_DATE
7369      SMITH    17-DEC-80
7499      ALLEN    20-FEB-81
7521      WARD     22-FEB-81
7566      JONES    02-APR-81
7654      MARTIN           28-SEP-81
7698      BLAKE    01-MAY-81
7782      CLARK    09-JUN-81
7788      SCOTT    19-APR-87
7839      KING     17-NOV-81
7844      TURNER           08-SEP-81
7876      ADAMS    23-MAY-87
7900      JAMES    03-DEC-81
7902      FORD     03-DEC-81
7934      MILLER           23-JAN-82
------- REPORT ----------
[sxsama2@HMLINUX1 AWK]$
```

Basic AWK program/script - EXAMPLE

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ cat EMP_REPORT.AWK
BEGIN{print "EMPID \t NAME \t JOIN_DATE"}
{print $1,"\t",$2,"\t",$5;}
END{ print "------- REPORT ----------";}

[sxsama2@HMLINUX1 AWK]$ awk -F"|" -f EMP_REPORT.AWK EMP.DAT
EMPID     NAME     JOIN_DATE
7369      SMITH    17-DEC-80
7499      ALLEN    20-FEB-81
7521      WARD     22-FEB-81
7566      JONES    02-APR-81
7654      MARTIN           28-SEP-81
7698      BLAKE    01-MAY-81
7782      CLARK    09-JUN-81
7788      SCOTT    19-APR-87
7839      KING     17-NOV-81
7844      TURNER           08-SEP-81
7876      ADAMS    23-MAY-87
7900      JAMES    03-DEC-81
7902      FORD     03-DEC-81
7934      MILLER           23-JAN-82
------- REPORT ----------
[sxsama2@HMLINUX1 AWK]$
```

AWK BUILTIN VARIABLES

VARIABLE NAME	DESCRIPTION
FS	Field separator (default=whitespace)
RS	Record separator (default=\n)
NF	Number of fields in current record
NR	Number of the current record
OFS	Output field separator (default=space)
ORS	Output record separator (default=\n)
FILENAME	Current filename
ARGC	Command Line Argument count (Available in GAWK/NAWK)
ARGV	Used to retrieve the COMMAND line PARAM values (Available in GAWK/NAWK)

FS – Field Separator

- The field separator is represented by the built-in variable FS.
- Awk does not use the name IFS which is used by the shell.
- You can change the value of FS in the awk program with the assignment operator, `=‘
- The right time to do this is at the beginning of execution (begin block)

Example :

```
awk 'BEGIN { FS = “|" } ; { print $2 }'  EMP.DAT
awk 'BEGIN { FS = “," } ; { print $2,$1 }'  EMP.CSV
```

Try this :

```
awk -F"|" '{ print $2 }'  EMP.DAT
awk -F"," '{ print $2,$1 }'  EMP.CSV
```

Example : FS – Filed separator

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN { FS = "|" } ; { print $2 }'  EMP.DAT
SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
SCOTT
KING
TURNER
ADAMS
JAMES
FORD
MILLER
[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN { FS = "," } ; { print $2,$1 }'  EMP.CSV
SMITH 7369
ALLEN 7499
WARD 7521
JONES 7566
MARTIN 7654
BLAKE 7698
CLARK 7782
SCOTT 7788
KING 7839
TURNER 7844
ADAMS 7876
JAMES 7900
FORD 7902
MILLER 7934
[sxsama2@HMLINUX1 AWK]$
```

Example : FS – Filed separator

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ cat FS_EX.awk
BEGIN{FS="|";print "---------EMPLOYEE RECORD--------" ;}
{print $2,"\t",$1;}
END{ print "-------- END EMPLOYEE RECORD  --------";}

[sxsama2@HMLINUX1 AWK]$ awk -f FS_EX.awk EMP.DAT
---------EMPLOYEE RECORD--------
SMITH    7369
ALLEN    7499
WARD     7521
JONES    7566
MARTIN   7654
BLAKE    7698
CLARK    7782
SCOTT    7788
KING     7839
TURNER   7844
ADAMS    7876
JAMES    7900
FORD     7902
MILLER   7934
-------- END EMPLOYEE RECORD  --------
[sxsama2@HMLINUX1 AWK]$
```

RS – Record Separator

- Record separator (default=\n)
- AWK reads one line at a time, and breaks up the line into fields.
- We can set the "RS" variable to change AWK's definition of a record/line

Example :

```
awk 'BEGIN{RS="|"}; NR==3{print $0}' EMP_RECORD
```

RS – Record Separator

```
$ cat EMP_RECORD
7369
SMITH
CLERK|
7499
ALLEN
SALESMAN|
7521
WARD
SALESMAN|
7566
JONES
MANAGER|
7654
MARTIN
SALESMAN|
$ awk 'BEGIN{RS="|"}; NR==3{print $0}' EMP_RECORD

7521
WARD
SALESMAN
```

Example : RS – Record Separator

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ cat EMP_RECORD
7369
SMITH
CLERK|
7499
ALLEN
SALESMAN|
7521
WARD
SALESMAN|
7566
JONES
MANAGER|
7654
MARTIN
SALESMAN|
[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN{RS="|"}; NR==3{print $0}' EMP_RECORD

7521
WARD
SALESMAN
[sxsama2@HMLINUX1 AWK]$
```

NF - Number of fields in current record

- **NF** gives the total number of fields in a record

- Can be used to check the number of fields are regular in a File ?

Example :

```
awk '{print $0"\t:"NF}' EMP2.DAT
awk -F"|" '{print $0"\t:"NF}' EMP.DAT
ls -l |awk '{print $0"\t:"NF}‘
df -h |tail +2|grep "%" |awk '{print $NF}‘
df -h |tail +2|grep "%" |awk '{print $NF,$(NF-1)}‘
df -h |tail +2|grep "%" |awk '{print $NF,"\t",$(NF-1)}'
```

NF - Number of fields in current record

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ awk '{print $0"\t:"NF}' EMP2.DAT
7369    SMITH   CLERK           17-DEC-80       :4
7499    ALLEN   SALESMAN        20-FEB-81       :4
7521    WARD    SALESMAN        22-FEB-81       :4
7782    CLARK   MANAGER         09-JUN-81       :4
7788    SCOTT   ANALYST         19-APR-87       :4
7839    KING    PRESIDENT       17-NOV-81       :4
7900    JAMES   CLERK           03-DEC-81       :4
7902    FORD    ANALYST         03-DEC-81       :4
[sxsama2@HMLINUX1 AWK]$ awk -F"|" '{print $0"\t:"NF}' EMP.DAT
7369|SMITH|CLERK|7902|17-DEC-80|800||20 :8
7499|ALLEN|SALESMAN|7698|20-FEB-81|1600|300|30  :8
7521|WARD|SALESMAN|7698|22-FEB-81|1250|500|30   :8
7566|JONES|MANAGER|7839|02-APR-81|2975||20      :8
7654|MARTIN|SALESMAN|7698|28-SEP-81|1250|1400|30        :8
7698|BLAKE|MANAGER|7839|01-MAY-81|2850||30      :8
7782|CLARK|MANAGER|7839|09-JUN-81|2450||10      :8
7788|SCOTT|ANALYST|7566|19-APR-87|3000||20      :8
7839|KING|PRESIDENT||17-NOV-81|5000||10 :8
7844|TURNER|SALESMAN|7698|08-SEP-81|1500|0|30   :8
7876|ADAMS|CLERK|7788|23-MAY-87|1100||20        :8
7900|JAMES|CLERK|7698|03-DEC-81|950||30 :8
7902|FORD|ANALYST|7566|03-DEC-81|3000||20       :8
7934|MILLER|CLERK|7782|23-JAN-82|1300||10       :8
[sxsama2@HMLINUX1 AWK]$ ls -l |awk '{print $0"\t:"NF}'
total 76        :2
-rw-r--r--  1 sxsama2 users  822 Oct 16 23:04 A :9
-rw-r--r--  1 sxsama2 users  234 Oct 15 22:24 EMP1.DAT  :9
-rw-r--r--  1 sxsama2 users  234 Oct 16 22:34 EMP2.DAT  :9
-rw-r--r--  1 sxsama2 users  605 Oct  9 18:52 EMP.CSV   :9
-rw-r--r--  1 sxsama2 users  605 Oct  9 18:50 EMP.DAT   :9
```

NF - Number of fields in current record

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                       72G   36G   33G  52% /
/dev/sda1              99M  9.5M   85M  11% /boot
none                  502M     0  502M   0% /dev/shm
[sxsama2@HMLINUX1 AWK]$ df -h |tail +2|grep "%" |awk '{print $NF}'
/
/boot
/dev/shm
[sxsama2@HMLINUX1 AWK]$ df -h |tail +2|grep "%" |awk '{print $NF,$(NF-1)}'
/ 52%
/boot 11%
/dev/shm 0%
[sxsama2@HMLINUX1 AWK]$ df -h |tail +2|grep "%" |awk '{print $NF,"\t",$(NF-1)}'
/        52%
/boot    11%
/dev/shm         0%
[sxsama2@HMLINUX1 AWK]$ df -h |tail +2|grep "%" |awk '{print $NF,"\t\t",$(NF-1)}'
/                52%
/boot            11%
/dev/shm                 0%
[sxsama2@HMLINUX1 AWK]$
```

NR - Number of the current record

NR - The number of record / the line number.

Example :

```
awk ‘print { NR,$0}’ EMP.DAT
```

The awk one liner to get the record number 5 -10 from a file :

```
awk 'NR >=5 && NR <=10 {print NR,$0}' EMP.DAT
awk 'NR >=5 && NR <=10 {print}' EMP.DAT
awk 'NR >=5 && NR <=10' EMP.DAT
```

NR - Number of the current record

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ awk '{ print  NR,$0}' EMP.DAT
1 7369|SMITH|CLERK|7902|17-DEC-80|800||20
2 7499|ALLEN|SALESMAN|7698|20-FEB-81|1600|300|30
3 7521|WARD|SALESMAN|7698|22-FEB-81|1250|500|30
4 7566|JONES|MANAGER|7839|02-APR-81|2975||20
5 7654|MARTIN|SALESMAN|7698|28-SEP-81|1250|1400|30
6 7698|BLAKE|MANAGER|7839|01-MAY-81|2850||30
7 7782|CLARK|MANAGER|7839|09-JUN-81|2450||10
8 7788|SCOTT|ANALYST|7566|19-APR-87|3000||20
9 7839|KING|PRESIDENT||17-NOV-81|5000||10
10 7844|TURNER|SALESMAN|7698|08-SEP-81|1500|0|30
11 7876|ADAMS|CLERK|7788|23-MAY-87|1100||20
12 7900|JAMES|CLERK|7698|03-DEC-81|950||30
13 7902|FORD|ANALYST|7566|03-DEC-81|3000||20
14 7934|MILLER|CLERK|7782|23-JAN-82|1300||10
[sxsama2@HMLINUX1 AWK]$ awk 'NR >=5 && NR <=10 {print NR,$0}' EMP.DAT
5 7654|MARTIN|SALESMAN|7698|28-SEP-81|1250|1400|30
6 7698|BLAKE|MANAGER|7839|01-MAY-81|2850||30
7 7782|CLARK|MANAGER|7839|09-JUN-81|2450||10
8 7788|SCOTT|ANALYST|7566|19-APR-87|3000||20
9 7839|KING|PRESIDENT||17-NOV-81|5000||10
10 7844|TURNER|SALESMAN|7698|08-SEP-81|1500|0|30
[sxsama2@HMLINUX1 AWK]$ awk 'NR >=5 && NR <=10 {print}' EMP.DAT
7654|MARTIN|SALESMAN|7698|28-SEP-81|1250|1400|30
7698|BLAKE|MANAGER|7839|01-MAY-81|2850||30
7782|CLARK|MANAGER|7839|09-JUN-81|2450||10
7788|SCOTT|ANALYST|7566|19-APR-87|3000||20
7839|KING|PRESIDENT||17-NOV-81|5000||10
7844|TURNER|SALESMAN|7698|08-SEP-81|1500|0|30
[sxsama2@HMLINUX1 AWK]$ awk 'NR >=5 && NR <=10' EMP.DAT
7654|MARTIN|SALESMAN|7698|28-SEP-81|1250|1400|30
```

OFS – Output field Separator

- The output field separator is represented by the built-in variable OFS.
- You can change the value of OFS in the awk program with the assignment operator, `=‘
- The right time to do this is at the beginning of execution (begin block)

Example :

```
awk -F"|"  'BEGIN { OFS = ":" } ; { print $2,$3 }'  EMP.DAT
awk -F","  'BEGIN { OFS = "+" } ; { print $2,$3 }'  EMP.CSV
```

Try this :

```
awk 'BEGIN { FS="," ;OFS = "+" } ; { print $2,$3 }'  EMP.CSV
awk 'BEGIN { FS="|" ;OFS = ":" } ; { print $2,$3 }'  EMP.DAT
```

OFS – Output field Separator

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ awk -F"|"  'BEGIN { OFS = ":" } ; { print $2,$3 }'  EMP.DAT
SMITH:CLERK
ALLEN:SALESMAN
WARD:SALESMAN
JONES:MANAGER
MARTIN:SALESMAN
BLAKE:MANAGER
CLARK:MANAGER
SCOTT:ANALYST
KING:PRESIDENT
TURNER:SALESMAN
ADAMS:CLERK
JAMES:CLERK
FORD:ANALYST
MILLER:CLERK
[sxsama2@HMLINUX1 AWK]$ awk -F","  'BEGIN { OFS = "+" } ; { print $2,$3 }'  EMP.CSV
SMITH+CLERK
ALLEN+SALESMAN
WARD+SALESMAN
JONES+MANAGER
MARTIN+SALESMAN
BLAKE+MANAGER
CLARK+MANAGER
SCOTT+ANALYST
KING+PRESIDENT
TURNER+SALESMAN
ADAMS+CLERK
JAMES+CLERK
FORD+ANALYST
MILLER+CLERK
[sxsama2@HMLINUX1 AWK]$
```

OFS – Output field Separator

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN { FS="," ;OFS = "+" } ; { print $2,$3 }'  EMP.CSV
SMITH+CLERK
ALLEN+SALESMAN
WARD+SALESMAN
JONES+MANAGER
MARTIN+SALESMAN
BLAKE+MANAGER
CLARK+MANAGER
SCOTT+ANALYST
KING+PRESIDENT
TURNER+SALESMAN
ADAMS+CLERK
JAMES+CLERK
FORD+ANALYST
MILLER+CLERK
[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN { FS="|" ;OFS = ":" } ; { print $2,$3 }'  EMP.DAT
SMITH:CLERK
ALLEN:SALESMAN
WARD:SALESMAN
JONES:MANAGER
MARTIN:SALESMAN
BLAKE:MANAGER
CLARK:MANAGER
SCOTT:ANALYST
KING:PRESIDENT
TURNER:SALESMAN
ADAMS:CLERK
JAMES:CLERK
FORD:ANALYST
MILLER:CLERK
[sxsama2@HMLINUX1 AWK]$
```

ORS – Output Record Separator

- Output Record separator (default=\n)

- AWK reads one line at a time, and breaks up the line into fields.

- We can set the “ORS" variable to change AWK's definition of a record/line

Example :

```
awk 'BEGIN{RS="|";ORS="+"}; NR==3{print $0}' EMP_RECORD
awk 'BEGIN{RS="|";ORS="+"};NR==2||NR==3{print}' EMP_RECORD
awk 'BEGIN{RS="|";ORS="+"};{print}' EMP_RECORD
```

Try this :

```
seq 10|awk 'BEGIN{ORS="|"}{print}‘
seq 10|awk 'BEGIN{ORS="+"}{print}'
```

ORS – Output Record Separator

```
$ cat EMP_RECORD
7369
SMITH
CLERK|
7499
ALLEN
SALESMAN|
7521
WARD
SALESMAN|
7566
JONES
MANAGER|
7654
MARTIN
SALESMAN|
$awk 'BEGIN{RS="|";ORS="+"};NR==2||NR==3{print}' EMP_RECORD

7499
ALLEN
SALESMAN+
7521
WARD
SALESMAN+
```

ORS – Output Record Separator

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ cat EMP_RECORD
7369
SMITH
CLERK|
7499
ALLEN
SALESMAN|
7521
WARD
SALESMAN|
7566
JONES
MANAGER|
7654
MARTIN
SALESMAN|
[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN{RS="|";ORS="+"};NR==2||NR==3{print}' EMP_RECORD

7499
ALLEN
SALESMAN+
7521
WARD
SALESMAN+[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN{RS="|";ORS="+"};NR==3{print}' EMP_RECORD

7521
WARD
SALESMAN+[sxsama2@HMLINUX1 AWK]$
```

FILENAME

FILENAME stores the name of the file which is currently being read by awk.

Example :

```
$ cat FILENAME.awk
#!/bin/awk -f
BEGIN {FN="";}
{       if (FN != FILENAME) {
                print "------------: PROCESSING", FILENAME " :------------";
                FN=FILENAME;
        }
        if (FNR==3){
        print;
        }
}
```

NOTE: FNR will work in gawk

Observe the value of **NR** incase of multiple input files

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ awk '{print NR,$0}' EMP.DAT EMP.CSV
1 7369|SMITH|CLERK|7902|17-DEC-80|800||20
2 7499|ALLEN|SALESMAN|7698|20-FEB-81|1600|300|30
3 7521|WARD|SALESMAN|7698|22-FEB-81|1250|500|30
4 7566|JONES|MANAGER|7839|02-APR-81|2975||20
5 7654|MARTIN|SALESMAN|7698|28-SEP-81|1250|1400|30
6 7698|BLAKE|MANAGER|7839|01-MAY-81|2850||30
7 7782|CLARK|MANAGER|7839|09-JUN-81|2450||10
8 7788|SCOTT|ANALYST|7566|19-APR-87|3000||20
9 7839|KING|PRESIDENT||17-NOV-81|5000||10
10 7844|TURNER|SALESMAN|7698|08-SEP-81|1500|0|30
11 7876|ADAMS|CLERK|7788|23-MAY-87|1100||20
12 7900|JAMES|CLERK|7698|03-DEC-81|950||30
13 7902|FORD|ANALYST|7566|03-DEC-81|3000||20
14 7934|MILLER|CLERK|7782|23-JAN-82|1300||10
15 7369,SMITH,CLERK,7902,17-DEC-80,800,,20
16 7499,ALLEN,SALESMAN,7698,20-FEB-81,1600,300,30
17 7521,WARD,SALESMAN,7698,22-FEB-81,1250,500,30
18 7566,JONES,MANAGER,7839,02-APR-81,2975,,20
19 7654,MARTIN,SALESMAN,7698,28-SEP-81,1250,1400,30
20 7698,BLAKE,MANAGER,7839,01-MAY-81,2850,,30
21 7782,CLARK,MANAGER,7839,09-JUN-81,2450,,10
22 7788,SCOTT,ANALYST,7566,19-APR-87,3000,,20
23 7839,KING,PRESIDENT,,17-NOV-81,5000,,10
24 7844,TURNER,SALESMAN,7698,08-SEP-81,1500,0,30
25 7876,ADAMS,CLERK,7788,23-MAY-87,1100,,20
26 7900,JAMES,CLERK,7698,03-DEC-81,950,,30
27 7902,FORD,ANALYST,7566,03-DEC-81,3000,,20
28 7934,MILLER,CLERK,7782,23-JAN-82,1300,,10
[sxsama2@HMLINUX1 AWK]$
```

Observe the value of **FNR** incase of multiple input files

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ awk '{print FNR,$0}' EMP.DAT EMP.CSV
1 7369|SMITH|CLERK|7902|17-DEC-80|800||20
2 7499|ALLEN|SALESMAN|7698|20-FEB-81|1600|300|30
3 7521|WARD|SALESMAN|7698|22-FEB-81|1250|500|30
4 7566|JONES|MANAGER|7839|02-APR-81|2975||20
5 7654|MARTIN|SALESMAN|7698|28-SEP-81|1250|1400|30
6 7698|BLAKE|MANAGER|7839|01-MAY-81|2850||30
7 7782|CLARK|MANAGER|7839|09-JUN-81|2450||10
8 7788|SCOTT|ANALYST|7566|19-APR-87|3000||20
9 7839|KING|PRESIDENT||17-NOV-81|5000||10
10 7844|TURNER|SALESMAN|7698|08-SEP-81|1500|0|30
11 7876|ADAMS|CLERK|7788|23-MAY-87|1100||20
12 7900|JAMES|CLERK|7698|03-DEC-81|950||30
13 7902|FORD|ANALYST|7566|03-DEC-81|3000||20
14 7934|MILLER|CLERK|7782|23-JAN-82|1300||10
1 7369,SMITH,CLERK,7902,17-DEC-80,800,,20
2 7499,ALLEN,SALESMAN,7698,20-FEB-81,1600,300,30
3 7521,WARD,SALESMAN,7698,22-FEB-81,1250,500,30
4 7566,JONES,MANAGER,7839,02-APR-81,2975,,20
5 7654,MARTIN,SALESMAN,7698,28-SEP-81,1250,1400,30
6 7698,BLAKE,MANAGER,7839,01-MAY-81,2850,,30
7 7782,CLARK,MANAGER,7839,09-JUN-81,2450,,10
8 7788,SCOTT,ANALYST,7566,19-APR-87,3000,,20
9 7839,KING,PRESIDENT,,17-NOV-81,5000,,10
10 7844,TURNER,SALESMAN,7698,08-SEP-81,1500,0,30
11 7876,ADAMS,CLERK,7788,23-MAY-87,1100,,20
12 7900,JAMES,CLERK,7698,03-DEC-81,950,,30
13 7902,FORD,ANALYST,7566,03-DEC-81,3000,,20
14 7934,MILLER,CLERK,7782,23-JAN-82,1300,,10
[sxsama2@HMLINUX1 AWK]$
```

FILENAME - Example

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ cat FILENAME.awk
#!/bin/awk -f
BEGIN {FN="";}

{       if (FN != FILENAME) {
                print "------------: PROCESSING", FILENAME " :------------";
                FN=FILENAME;
        }
        if (FNR==3){
        print;
        }
}

[sxsama2@HMLINUX1 AWK]$ awk -f FILENAME.awk EMP.DAT EMP.CSV
------------: PROCESSING EMP.DAT :------------
7521|WARD|SALESMAN|7698|22-FEB-81|1250|500|30
------------: PROCESSING EMP.CSV :------------
7521,WARD,SALESMAN,7698,22-FEB-81,1250,500,30
[sxsama2@HMLINUX1 AWK]$
```

ARGC AND ARGV (**This is available in gawk/nawk**)

- In awk the array **ARGV** contains the elements **ARGV[0]**, . . ., **ARGV[ARGC-1]**
- **ARGC** is the count of command line parameters.
- **ARGV[0]** is the name of the program (generally **awk**)
- The remaining arguments are whatever was provided (excluding the program and any optional arguments)

Example :

$ **cat Sum.awk**

```
#!/bin/awk -f
BEGIN{
        print "TOTAL NUMBER OF ARGUMENTS",ARGC ;
        print "VALUE OF ARGV[0]" ,ARGV[0];
        print "VALUE OF ARGV[1]" ,ARGV[1];
        print "VALUE OF ARGV[2]" ,ARGV[2];
        print ARGV[1]+ARGV[2];
}
```

$ **chmod +x Sum.awk**

$ **./Sum.awk 23 22**

ARGC AND ARGV (This is available in gawk/nawk)

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ cat Sum.awk
#!/bin/awk -f
BEGIN{
        print "TOTAL NUMBER OF ARGUMENTS",ARGC ;
        print "VALUE OF ARGV[0]" ,ARGV[0];
        print "VALUE OF ARGV[1]" ,ARGV[1];
        print "VALUE OF ARGV[2]" ,ARGV[2];
        print ARGV[1]+ARGV[2];
}
[sxsama2@HMLINUX1 AWK]$ chmod +x Sum.awk
[sxsama2@HMLINUX1 AWK]$ ./Sum.awk 23 22
TOTAL NUMBER OF ARGUMENTS 3
VALUE OF ARGV[0] awk
VALUE OF ARGV[1] 23
VALUE OF ARGV[2] 22
45
[sxsama2@HMLINUX1 AWK]$
```

FUNCTIONS

FUNCTION NAME	DESCRIPTION
length()	Calculates the length of a string
index()	Used to search for specific character inside a string
substr()	Used to extract a portion of a string
split()	Splits the string into pieces based on the defined field separator and stores the pieces in to an array
toupper()	To convert lowercase to uppercase (Available in GAWK)
tolower()	To convert uppercase to lowercase (Available in GAWK)
system()	To execute any program /command (Available in GAWK & NAWK)
systime()	Returns number of seconds since Midnight, January 1, 1970 (Available in GAWK)

length Function :

- The *length*() function calculates the length of a string.

- length() function can be used to calculate the length of the each record or any specific filed .

- length() function can be used to validate the length of a specific field / record

Example :

```
awk -F "|" '{print length($2),$2}' EMP.DAT
awk -F "|" 'length($2)==4{print $2}' EMP.DAT
awk -F "|" 'length($2)==4{print length($0),$0}' EMP.DAT
```

length Function :

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ awk -F "|" '{print length($2),$2}' EMP.DAT
5 SMITH
5 ALLEN
4 WARD
5 JONES
6 MARTIN
5 BLAKE
5 CLARK
5 SCOTT
4 KING
6 TURNER
5 ADAMS
5 JAMES
4 FORD
6 MILLER
[sxsama2@HMLINUX1 AWK]$ awk -F "|" 'length($2)==4{print $2}' EMP.DAT
WARD
KING
FORD
[sxsama2@HMLINUX1 AWK]$ awk -F "|" 'length($2)==4{print length($0),$0}' EMP.DAT
45 7521|WARD|SALESMAN|7698|22-FEB-81|1250|500|30
39 7839|KING|PRESIDENT||17-NOV-81|5000||10
41 7902|FORD|ANALYST|7566|03-DEC-81|3000||20
[sxsama2@HMLINUX1 AWK]$
```

index Function :

The *index*() function is used to search for specific characters inside a string.

index(“*Centurylink”*, “*link”*) : It will search for the first occurrence of string “link” in “Centurylink” and returns the position where the string “link” begins.

index(“*Centurylink”*, “*t”*) : It will search for the first occurrence of character “t” in string “Centurylink” and returns the position.

Example:

```
$ awk 'BEGIN{print index("Centurylink","link")}'
8
$ awk 'BEGIN{print index("Centurylink","t")}'
4
$ awk '/SMITH/{print index($0,"SMITH")}' EMP.DAT
6
$ awk '/SMITH/{print}' EMP.DAT
7369|SMITH|CLERK|7902|17-DEC-80|800||20
```

index Function :

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN{print index("Centurylink","link")}'
8
[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN{print index("Centurylink","t")}'
4
[sxsama2@HMLINUX1 AWK]$ awk '/SMITH/{print index($0,"SMITH")}' EMP.DAT
6
[sxsama2@HMLINUX1 AWK]$ awk '/SMITH/{print}' EMP.DAT
7369|SMITH|CLERK|7902|17-DEC-80|800||20
[sxsama2@HMLINUX1 AWK]$
```

substr Function :

- The *substr*() function is used to extract a portion of a string.

- One common use is to split a string into two parts based on a special character.

- **substr(STR, m, n)** : Returns **n** number of chars from string **STR,** starting at position **m**.

- **substr(STR, m)** : Returns all the characters staring from character **m** to the end from string **STR**

substr Function - Example

EXAMPLES :

```
$ awk 'BEGIN{print substr("Centurylink",8)}'
link
$ awk 'BEGIN{print substr("Centurylink",4,8)}'
turylink
$ awk 'BEGIN{print substr("Centurylink",1,6)}'
Centur
$ awk 'BEGIN{print substr("Centurylink",1,7)}'
Century
$ awk '/SMITH/{print}' EMP.DAT
7369|SMITH|CLERK|7902|17-DEC-80|800||20
$ awk '/SMITH/{print substr($0,index($0,"|")+1,5)}' EMP.DAT
SMITH
```

substr Function - Example

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN{print substr("Centurylink",8)}'
link
[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN{print substr("Centurylink",1,7)}'
Century
[sxsama2@HMLINUX1 AWK]$ awk '/SMITH/{print}' EMP.DAT
7369|SMITH|CLERK|7902|17-DEC-80|800||20
[sxsama2@HMLINUX1 AWK]$ awk '/SMITH/{print substr($0,index($0,"|")+1,5)}' EMP.DAT
SMITH
[sxsama2@HMLINUX1 AWK]$ awk '/SMITH/{print}' EMP.CSV
7369,SMITH,CLERK,7902,17-DEC-80,800,,20
[sxsama2@HMLINUX1 AWK]$ awk '/SMITH/{print substr($0,index($0,",")+1,5)}' EMP.CSV
SMITH
```

split Function :

split() function is used to split the string.

It takes three arguments: The string, an array to store the result, and the filed separator.

Example :

```
$ awk '/SMITH/{print}' EMP.DAT
7369|SMITH|CLERK|7902|17-DEC-80|800||20
$ awk '/SMITH/{split($0,EMPREC,"|"); print EMPREC[2],EMPREC[3]}' EMP.DAT
SMITH CLERK
$ awk '/SMITH/{split($0,EMPREC,"|"); print EMPREC[1],EMPREC[2]}' EMP.DAT
7369 SMITH
$ awk '/SMITH/{split($0,EMPREC,"|"); print EMPREC[2],EMPREC[5]}' EMP.DAT
SMITH 17-DEC-80
$ awk '/SMITH/{print split($0,EMPREC,"|")}' EMP.DAT
8
```

Size of Array

```
$ echo "a|b|c|d" |awk '{n=split($0,arr,"|")}END{print arr[n]}'
d
$ echo "a|b|c|d" |awk '{n=split($0,arr,"|")}END{print n}'
4
```

split Function - Example

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ awk '/SMITH/{print}' EMP.DAT
7369|SMITH|CLERK|7902|17-DEC-80|800||20
[sxsama2@HMLINUX1 AWK]$ awk '/SMITH/{split($0,EMPREC,"|"); print EMPREC[2],EMPREC[3]}' EMP.DAT
SMITH CLERK
[sxsama2@HMLINUX1 AWK]$ awk '/SMITH/{split($0,EMPREC,"|"); print EMPREC[1],EMPREC[2]}' EMP.DAT
7369 SMITH
[sxsama2@HMLINUX1 AWK]$ awk '/SMITH/{split($0,EMPREC,"|"); print EMPREC[2],EMPREC[5]}' EMP.DAT
SMITH 17-DEC-80
[sxsama2@HMLINUX1 AWK]$ awk '/SMITH/{print split($0,EMPREC,"|")}' EMP.DAT
8
[sxsama2@HMLINUX1 AWK]$
```

tolower - This is available in gawk

tolower(string) : Converts the upper case character to lower case.

Generally we use the tr command to translate the upper case characters to lowercase .

Example :

```
cat EMP2.DAT |tr '[:upper:]' '[:lower:]'
```

Awk one liner :

```
cat EMP2.DAT |awk '{print tolower($0)}'
```

tolower - This is available in gawk

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ cat EMP2.DAT
7369    SMITH   CLERK           17-DEC-80
7499    ALLEN   SALESMAN        20-FEB-81
7521    WARD    SALESMAN        22-FEB-81
7782    CLARK   MANAGER         09-JUN-81
7788    SCOTT   ANALYST         19-APR-87
7839    KING    PRESIDENT       17-NOV-81
7900    JAMES   CLERK           03-DEC-81
7902    FORD    ANALYST         03-DEC-81
[sxsama2@HMLINUX1 AWK]$ cat EMP2.DAT |tr '[:upper:]' '[:lower:]'
7369    smith   clerk           17-dec-80
7499    allen   salesman        20-feb-81
7521    ward    salesman        22-feb-81
7782    clark   manager         09-jun-81
7788    scott   analyst         19-apr-87
7839    king    president       17-nov-81
7900    james   clerk           03-dec-81
7902    ford    analyst         03-dec-81
[sxsama2@HMLINUX1 AWK]$ cat EMP2.DAT |awk '{print tolower($0)}'
7369    smith   clerk           17-dec-80
7499    allen   salesman        20-feb-81
7521    ward    salesman        22-feb-81
7782    clark   manager         09-jun-81
7788    scott   analyst         19-apr-87
7839    king    president       17-nov-81
7900    james   clerk           03-dec-81
7902    ford    analyst         03-dec-81
[sxsama2@HMLINUX1 AWK]$
```

toupper - This is available in gawk

toupper(string) : Converts the lower-case character(s) to upper-case.

Generally we use the tr command to translate the lower case characters to upper case.

Example :

```
cat EMP1.DAT |tr '[:lower:]'  '[:upper:]'
```

Awk one liner :

```
cat EMP1.DAT |awk '{print toupper($0)}'
```

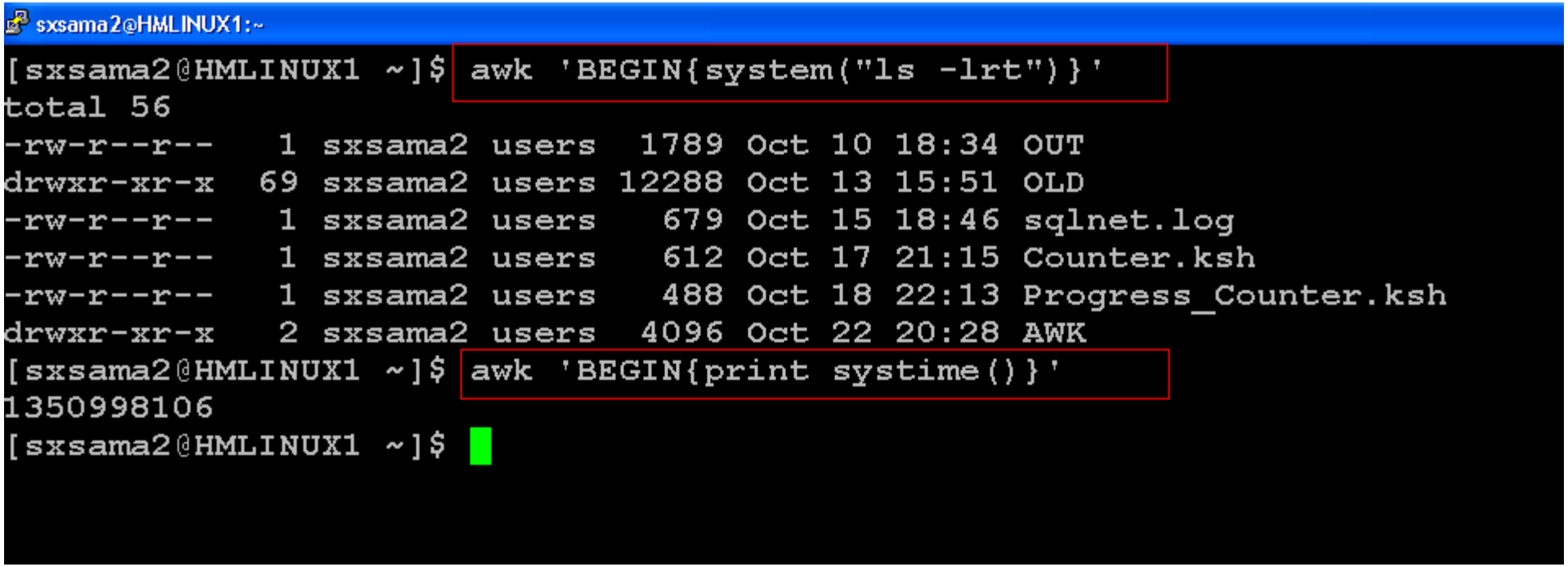
toupper - This is available in gawk

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ cat EMP1.DAT
7369    smith   clerk           17-dec-80
7499    allen   salesman        20-feb-81
7521    ward    salesman        22-feb-81
7782    clark   manager         09-jun-81
7788    scott   analyst         19-apr-87
7839    king    president       17-nov-81
7900    james   clerk           03-dec-81
7902    ford    analyst         03-dec-81
[sxsama2@HMLINUX1 AWK]$ cat EMP1.DAT |tr '[:lower:]'  '[:upper:]'
7369    SMITH   CLERK           17-DEC-80
7499    ALLEN   SALESMAN        20-FEB-81
7521    WARD    SALESMAN        22-FEB-81
7782    CLARK   MANAGER         09-JUN-81
7788    SCOTT   ANALYST         19-APR-87
7839    KING    PRESIDENT       17-NOV-81
7900    JAMES   CLERK           03-DEC-81
7902    FORD    ANALYST         03-DEC-81
[sxsama2@HMLINUX1 AWK]$ cat EMP1.DAT |awk '{print toupper($0)}'
7369    SMITH   CLERK           17-DEC-80
7499    ALLEN   SALESMAN        20-FEB-81
7521    WARD    SALESMAN        22-FEB-81
7782    CLARK   MANAGER         09-JUN-81
7788    SCOTT   ANALYST         19-APR-87
7839    KING    PRESIDENT       17-NOV-81
7900    JAMES   CLERK           03-DEC-81
7902    FORD    ANALYST         03-DEC-81
[sxsama2@HMLINUX1 AWK]$
```

system() & *systime*() Function (Available in GAWK/NAWK)

system() - To execute any program /command

systime() - Returns number of seconds since Midnight, January 1, 1970

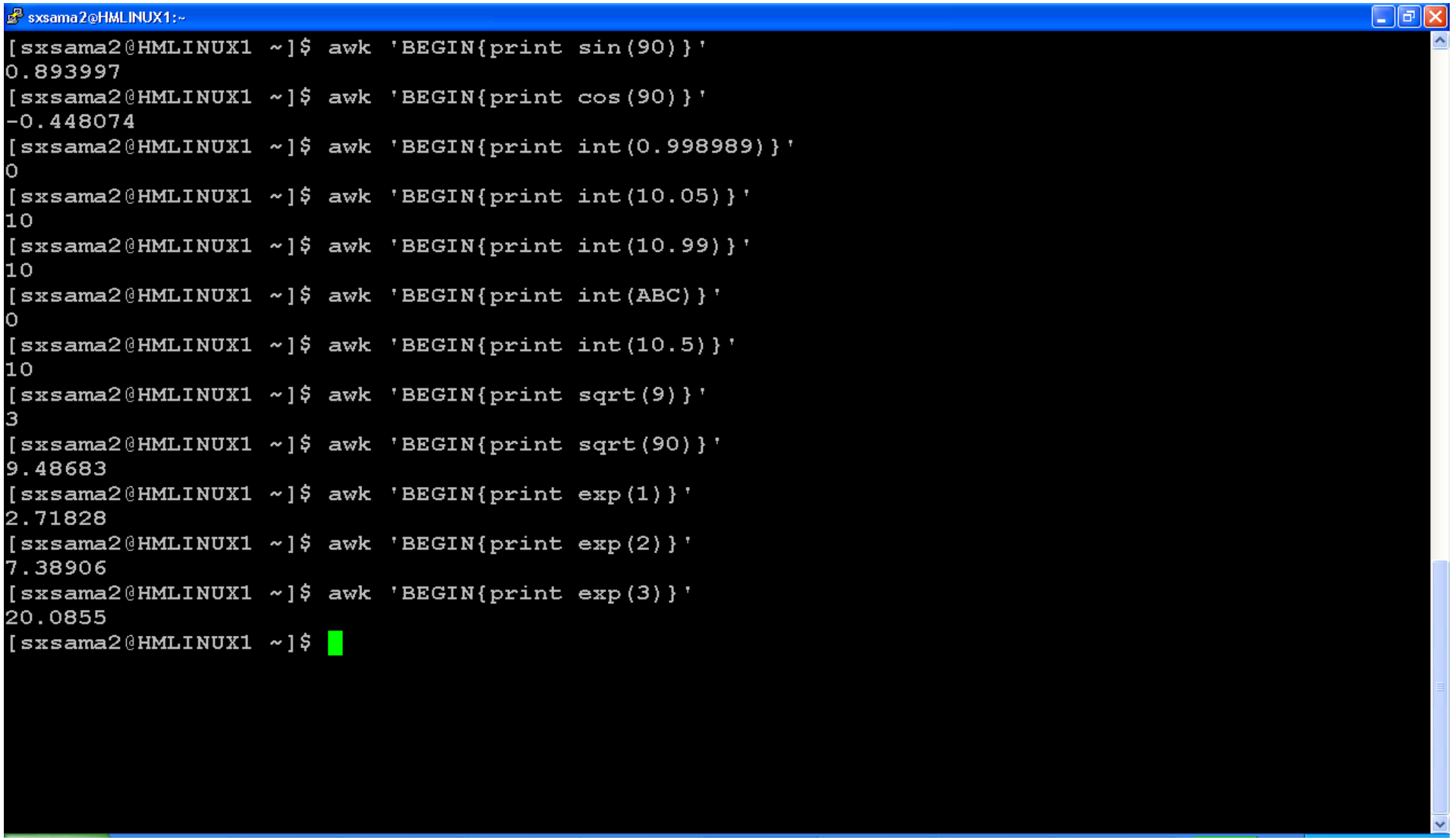


CenturyLink™

Arithmetic Functions

FUNCTION NAME	DESCRIPTION
cos(x)	Return cosine of x, where x is in radians.
sin(x)	Return sine of x, where x is in radians.
exp(x)	Return the exponential function of x.
log(x)	Return the natural logarithm of x.
sqrt(x)	Return the square root of x.
int(x)	Truncate its argument to an integer.

Arithmetic Functions



Programming in AWK

if (*condition*) *statement* **else** *statement*
while (*condition*) *statement*
for (initialization; *condition* ; *increment*) *statement*
for (*variable* in *array*) *statement*
getline
next
break
continue
exit

Simple If statement

Syntax:

```
if (conditional-expression)
{
        Statement1;
        Statement2;
 }
```

Simple If statement - Example

```
$ cat If_Example.awk
#!/bin/awk -f
## PRE_PROCESSING BLOCK ##
BEGIN{
        FS="|";
        print "---------------- MANAGERS DETAILS ----------------";
    }
## PROCESSING BLOCK ###
{
        if ( $3== "MANAGER")
        {
                printf "%-20s%-20s%-20s\n" ,$1,$2,$5;
        }
}
## END BLOCK/POST PROCESSING BLOCK ##
END{
        print "-----------------------------------------------";
  }
```

Simple If statement - Example

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ cat If_Example.awk
#!/bin/awk -f
## PRE_PROCESSING BLOCK ##
BEGIN{
        FS="|";
        print "---------------- MANAGERS DETAILS ----------------";
    }
## PROCESSING BLOCK ###
{
        if ( $3== "MANAGER")
        {
                printf "%-20s%-20s%-20s\n" ,$1,$2,$5;
        }
}
## END BLOCK/POST PROCESSING BLOCK ##
END{
        print "-------------------------------------------------";
   }
[sxsama2@HMLINUX1 AWK]$ awk -f If_Example.awk EMP.DAT
---------------- MANAGERS DETAILS ----------------
7566                JONES               02-APR-81
7698                BLAKE               01-MAY-81
7782                CLARK               09-JUN-81
-------------------------------------------------
[sxsama2@HMLINUX1 AWK]$
```

If else statement

Syntax:

```
if (conditional-expression)
{
        Statements;
 }
else
{
        Statements;
}
```

If else statement - Example

```
#!/bin/awk -f
## PRE_PROCESSING BLOCK ##
BEGIN{
        FS="|";
        print "-------------------------------------------------------------------";
        print "--                MANAGERS DETAILS                    --";
        print "-------------------------------------------------------------------";
        printf "%-20s%-20s%-20s%-10s\n" ,"EMP_ID","ENAME","JOINING_DATE","BONUS_CHK";
        print "-------------------------------------------------------------------";
    }
## PROCESSING BLOCK ###
{
        if ( $3== "MANAGER")
        {
                printf "%-20s%-20s%-20s%-10s\n" ,$1,$2,$5,"YES";
        }
        else
        {
                printf "%-20s%-20s%-20s%-10s\n" ,$1,$2,$5,"NO";
        }
}
## END BLOCK/POST PROCESSING BLOCK ##
END{
        print "-------------------------------------------------------------------";
   }
```

If else statement - Example **(INPUT FILE)**

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ cat EMP.DAT
7369|SMITH|CLERK|7902|17-DEC-80|800||20
7499|ALLEN|SALESMAN|7698|20-FEB-81|1600|300|30
7521|WARD|SALESMAN|7698|22-FEB-81|1250|500|30
7566|JONES|MANAGER|7839|02-APR-81|2975||20
7654|MARTIN|SALESMAN|7698|28-SEP-81|1250|1400|30
7698|BLAKE|MANAGER|7839|01-MAY-81|2850||30
7782|CLARK|MANAGER|7839|09-JUN-81|2450||10
7788|SCOTT|ANALYST|7566|19-APR-87|3000||20
7839|KING|PRESIDENT||17-NOV-81|5000||10
7844|TURNER|SALESMAN|7698|08-SEP-81|1500|0|30
7876|ADAMS|CLERK|7788|23-MAY-87|1100||20
7900|JAMES|CLERK|7698|03-DEC-81|950||30
7902|FORD|ANALYST|7566|03-DEC-81|3000||20
7934|MILLER|CLERK|7782|23-JAN-82|1300||10
[sxsama2@HMLINUX1 AWK]$
```

If else statement - Example **(Program)**

sxsama2@HMLINUX1:~/AWK

```
[sxsama2@HMLINUX1 AWK]$ cat If_else_Example.awk
#!/bin/awk -f
## PRE_PROCESSING BLOCK ##
BEGIN{
        FS="|";
        print "-------------------------------------------------------------------";
        print "--                  MANAGERS DETAILS                             --";
        print "-------------------------------------------------------------------";
        printf "%-20s%-20s%-20s%-10s\n" ,"EMP_ID","ENAME","JOINING_DATE","BONUS_CHK";
        print "-------------------------------------------------------------------";
     }
## PROCESSING BLOCK ###
{
        if ( $3== "MANAGER")
        {
                printf "%-20s%-20s%-20s%-10s\n" ,$1,$2,$5,"YES";
        }
        else
        {
                printf "%-20s%-20s%-20s%-10s\n" ,$1,$2,$5,"NO";
        }
}
## END BLOCK/POST PROCESSING BLOCK ##
END{
        print "-------------------------------------------------------------------";
   }
[sxsama2@HMLINUX1 AWK]$
```

If else statement - Example **(OUPUT)**

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ awk -f If_else_Example.awk EMP.DAT
-------------------------------------------------------------------
--                  MANAGERS DETAILS                             --
-------------------------------------------------------------------
EMP_ID                ENAME                JOINING_DATE         BONUS_CHK
-------------------------------------------------------------------
7369                  SMITH                17-DEC-80            NO
7499                  ALLEN                20-FEB-81            NO
7521                  WARD                 22-FEB-81            NO
7566                  JONES                02-APR-81            YES
7654                  MARTIN               28-SEP-81            NO
7698                  BLAKE                01-MAY-81            YES
7782                  CLARK                09-JUN-81            YES
7788                  SCOTT                19-APR-87            NO
7839                  KING                 17-NOV-81            NO
7844                  TURNER               08-SEP-81            NO
7876                  ADAMS                23-MAY-87            NO
7900                  JAMES                03-DEC-81            NO
7902                  FORD                 03-DEC-81            NO
7934                  MILLER               23-JAN-82            NO
-------------------------------------------------------------------
[sxsama2@HMLINUX1 AWK]$
```

Awk while Loop

Syntax:

```
while(condition)
statements;
```

Example of a simple while loop :

```
#!/bin/awk -f
BEGIN {

    i=1;
    while (i <= 10)
    {
        printf "The square of %d is %d\n",i,i*i;
        i = i+1;
    }
  }
```

Awk while Loop

sxsama2@HMLINUX1:~/AWK

```
[sxsama2@HMLINUX1 AWK]$ cat While_Loop_Ex.awk
#!/bin/awk -f
BEGIN {

        i=1;
        while (i <= 10)
        {
                printf "The square of %d is %d\n",i,i*i;
                i = i+1;
        }
      }
[sxsama2@HMLINUX1 AWK]$ awk -f While_Loop_Ex.awk
The square of 1 is 1
The square of 2 is 4
The square of 3 is 9
The square of 4 is 16
The square of 5 is 25
The square of 6 is 36
The square of 7 is 49
The square of 8 is 64
The square of 9 is 81
The square of 10 is 100
[sxsama2@HMLINUX1 AWK]$
```

Awk for Loop

Syntax:

for (initialization; *condition* ; *increment*)

statements;

Example of a simple while loop :

```
#!/bin/awk -f
BEGIN {
        for (i=1;i <= 10;i++)
         {
        printf "The square of %d is %d\n",i,i*i;
        }
    }
```

Awk for Loop

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ cat For_Ex.awk
#!/bin/awk -f
BEGIN {
        for (i=1;i <= 10;i++)
         {
                printf "The square of %d is %d\n",i,i*i;
                }
      }

[sxsama2@HMLINUX1 AWK]$ awk -f For_Ex.awk
The square of 1 is 1
The square of 2 is 4
The square of 3 is 9
The square of 4 is 16
The square of 5 is 25
The square of 6 is 36
The square of 7 is 49
The square of 8 is 64
The square of 9 is 81
The square of 10 is 100
[sxsama2@HMLINUX1 AWK]$
```

getline

getline : Set $0 to the next input record from the current input file. getline returns 1 for successful input, 0 for end of file, and -1 for an error.

Example :

```
awk '/SMITH/{getline;getline;print}' EMP.DAT
awk '/SMITH/{print;getline;print;getline;print}' EMP.DAT
```

getline

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ cat EMP.DAT
7369|SMITH|CLERK|7902|17-DEC-80|800||20
7499|ALLEN|SALESMAN|7698|20-FEB-81|1600|300|30
7521|WARD|SALESMAN|7698|22-FEB-81|1250|500|30
7566|JONES|MANAGER|7839|02-APR-81|2975||20
7654|MARTIN|SALESMAN|7698|28-SEP-81|1250|1400|30
7698|BLAKE|MANAGER|7839|01-MAY-81|2850||30
7782|CLARK|MANAGER|7839|09-JUN-81|2450||10
7788|SCOTT|ANALYST|7566|19-APR-87|3000||20
7839|KING|PRESIDENT||17-NOV-81|5000||10
7844|TURNER|SALESMAN|7698|08-SEP-81|1500|0|30
7876|ADAMS|CLERK|7788|23-MAY-87|1100||20
7900|JAMES|CLERK|7698|03-DEC-81|950||30
7902|FORD|ANALYST|7566|03-DEC-81|3000||20
7934|MILLER|CLERK|7782|23-JAN-82|1300||10
[sxsama2@HMLINUX1 AWK]$ awk '/SMITH/' EMP.DAT
7369|SMITH|CLERK|7902|17-DEC-80|800||20
[sxsama2@HMLINUX1 AWK]$ awk '/SMITH/{print;getline;print;getline;print}' EMP.DAT
7369|SMITH|CLERK|7902|17-DEC-80|800||20
7499|ALLEN|SALESMAN|7698|20-FEB-81|1600|300|30
7521|WARD|SALESMAN|7698|22-FEB-81|1250|500|30
[sxsama2@HMLINUX1 AWK]$ awk '/SMITH/{getline;getline;print}' EMP.DAT
7521|WARD|SALESMAN|7698|22-FEB-81|1250|500|30
[sxsama2@HMLINUX1 AWK]$
```

next

- The next statement forces awk to immediately stop processing the current record and go on to the next record.

- This means that no further rules are executed for the current record, and the rest of the current rule's action isn't executed.

- If the next statement causes the end of the input to be reached, then the code in any END rules is executed .

- The next statement is not allowed inside BEGIN and END block.

Next - Example

```
#!/bin/awk -f

{
        if ( NF != 4 )
        {
                print "########################################";
                print "Skipping processing for employee :" $2;
                print $0;
                next;
                ##  IF Number of fileds not equal to 4 , Stop processing here and Move to the next record  ###
        }
        else
        {
                print "%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%" ;
                print "Processing Records for employee :"$2;
                print $0;
        }

        print "Processing completed for employee :"$2;
        print "%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%" ;
}
```

Next - Example

sxsama2@HMLINUX1:~/AWK

```
[sxsama2@HMLINUX1 AWK]$ cat EMP11.DAT
7369    smith   clerk           17-dec-80
7499    allen   salesman
7521    ward    salesman        22-feb-81
7782    clark   manager
7788    scott   analyst         19-apr-87
7839    king    president       17-nov-81
7900    james   clerk
7902    ford    analyst         03-dec-81
[sxsama2@HMLINUX1 AWK]$ cat Next.awk
#!/bin/awk -f

{

        if ( NF != 4 )
        {
                print "########################################";
                print "Skipping processing for employee :" $2;
                print $0;
                next;
        }
        else
        {
                print "%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%" ;
                print "Processing Records for employee :"$2;
                print $0;
        }

        print "Processing completed for employee :"$2;
        print "%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%" ;
}
[sxsama2@HMLINUX1 AWK]$
```

Next - Example

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ awk -f Next.awk EMP11.DAT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Processing Records for employee :smith
7369     smith    clerk            17-dec-80
Processing completed for employee :smith
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
########################################
Skipping processing for employee :allen
7499     allen    salesman
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Processing Records for employee :ward
7521     ward     salesman         22-feb-81
Processing completed for employee :ward
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
########################################
Skipping processing for employee :clark
7782     clark    manager
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Processing Records for employee :scott
7788     scott    analyst          19-apr-87
Processing completed for employee :scott
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Processing Records for employee :king
7839     king     president        17-nov-81
Processing completed for employee :king
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
########################################
Skipping processing for employee :james
7900     james    clerk
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Processing Records for employee :ford
7902     ford     analyst          03-dec-81
Processing completed for employee :ford
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[sxsama2@HMLINUX1 AWK]$
```

break and continue

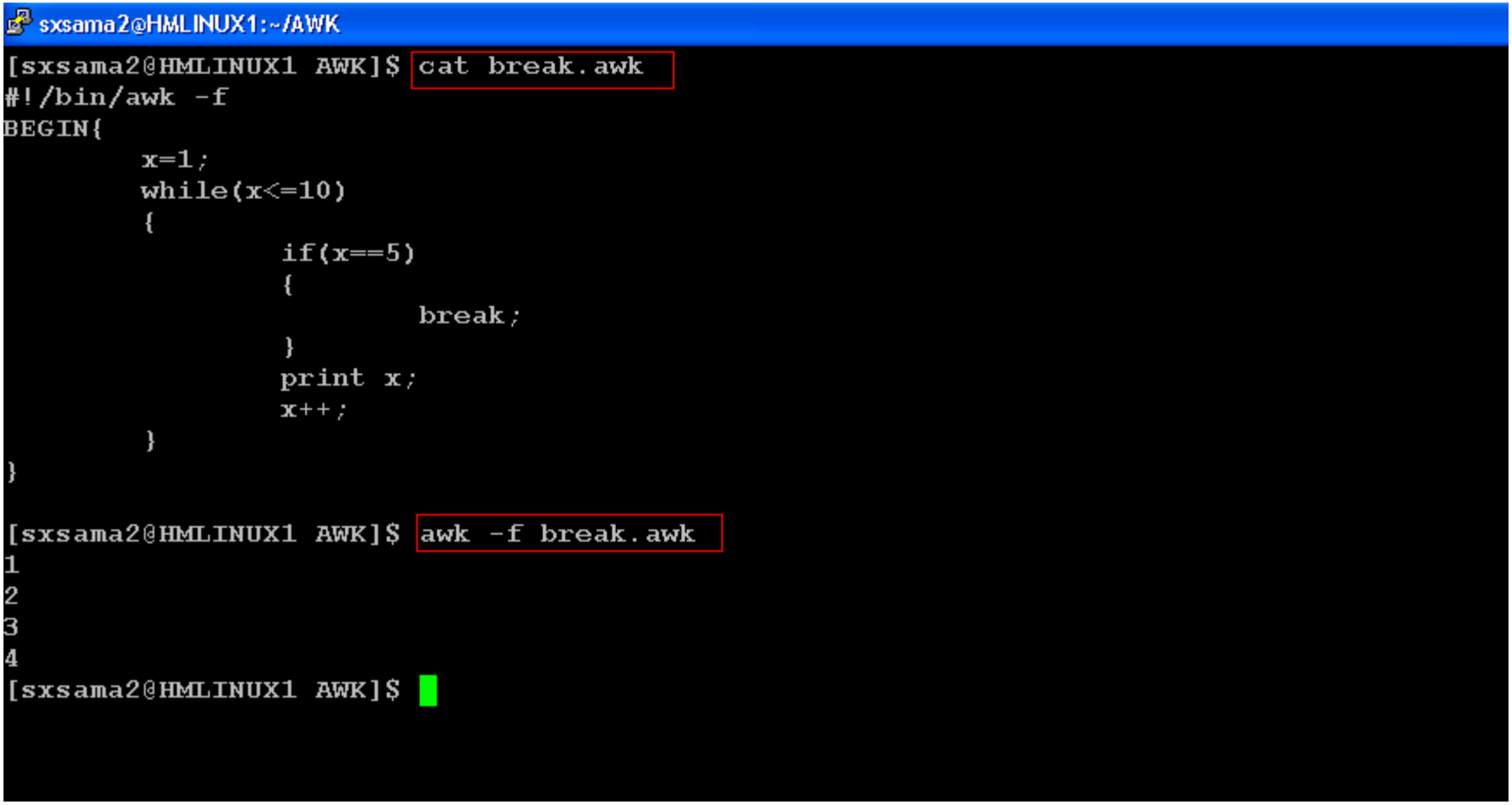
```
#!/bin/awk -f
BEGIN{
        x=1;
        while(x<=10)
        {
                if(x==5)
                {
                        x++;
                        continue;
                }
                print x;
                x++;
        }
}
```

```
#!/bin/awk -f
BEGIN{
        x=1;
        while(x<=10)
        {
                if(x==5)
                {
                        break;
                }
                print x;
                x++;
        }
}
```

continue - Example

```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ cat Continue.awk
#!/bin/awk -f
BEGIN{
        x=1;
        while(x<=10)
        {
                if(x==5)
                {
                        x++;
                        continue;
                }
                print x;
                x++;
        }
}
[sxsama2@HMLINUX1 AWK]$ awk -f Continue.awk
1
2
3
4
6
7
8
9
10
[sxsama2@HMLINUX1 AWK]$
```

break - Example



exit - Example

```
#!/bin/awk
BEGIN{
        FS="|";
        print "-----------------------------------------------------------"
        print "            ANNUAL BONUS CALCULATION                    "
        print "-----------------------------------------------------------"

    }
{
        if ( $3 == "PRESIDENT" )
        {
                print "-----------------------------------------------------------"
                print "Bonus Should not be calculated for - President, Terminating";
                print "Data Error Processing  Input File :" FILENAME ;
                print $0;
                print "-----------------------------------------------------------"
                exit;
        }
        else
        {
                print "Calculating Bonus for employee:" $2;
        }
}
END{
        print "-----------------------------------------------------------"
        print "                 END OF REPORT                      "
        print "-----------------------------------------------------------"
  }
```

exit - Example

sxsama2@HMLINUX1:~/AWK

```
[sxsama2@HMLINUX1 AWK]$ cat Exit.awk
#!/bin/awk
BEGIN{
        FS="|";
        print "-------------------------------------------------------------"
        print "               ANNUAL BONUS CALCULATION                      "
        print "-------------------------------------------------------------"

     }

{
        if ( $3 == "PRESIDENT" )
        {
                print "-------------------------------------------------------------"
                print "Bonus Should not be calculated for - President, Terminating";
                print "Data Error Processing  Input File :" FILENAME ;
                print $0;
                print "-------------------------------------------------------------"
                exit;
        }
        else
        {
                print "Calculating Bonus for employee:" $2;
        }
}

END{
        print "-------------------------------------------------------------"
        print "                      END OF REPORT                          "
        print "-------------------------------------------------------------"
   }
[sxsama2@HMLINUX1 AWK]$
```

exit - Example

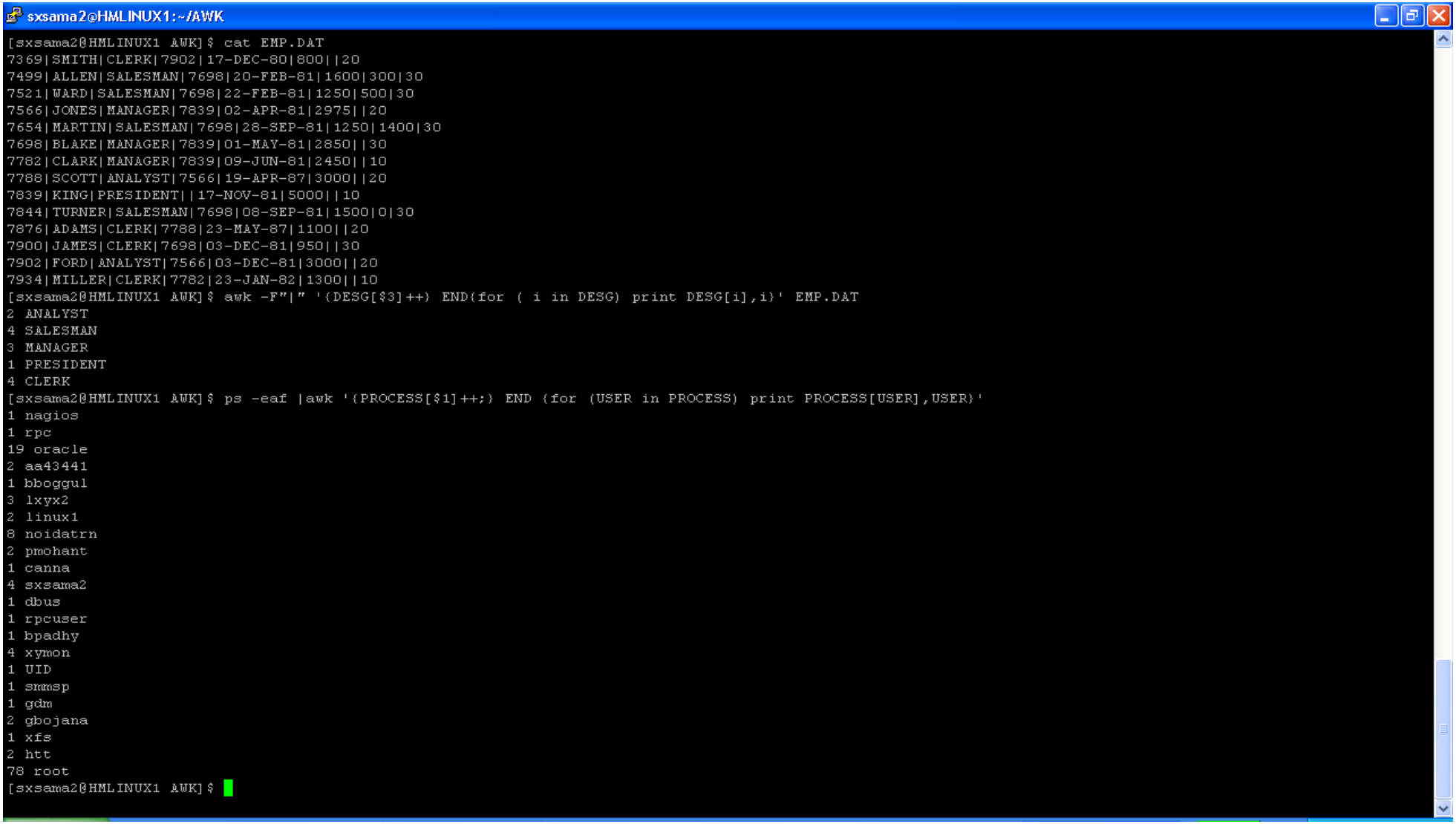
```
sxsama2@HMLINUX1:~/AWK
[sxsama2@HMLINUX1 AWK]$ cat EMP.DAT
7369|SMITH|CLERK|7902|17-DEC-80|800||20
7499|ALLEN|SALESMAN|7698|20-FEB-81|1600|300|30
7521|WARD|SALESMAN|7698|22-FEB-81|1250|500|30
7566|JONES|MANAGER|7839|02-APR-81|2975||20
7654|MARTIN|SALESMAN|7698|28-SEP-81|1250|1400|30
7698|BLAKE|MANAGER|7839|01-MAY-81|2850||30
7782|CLARK|MANAGER|7839|09-JUN-81|2450||10
7788|SCOTT|ANALYST|7566|19-APR-87|3000||20
7839|KING|PRESIDENT||17-NOV-81|5000||10
7844|TURNER|SALESMAN|7698|08-SEP-81|1500|0|30
7876|ADAMS|CLERK|7788|23-MAY-87|1100||20
7900|JAMES|CLERK|7698|03-DEC-81|950||30
7902|FORD|ANALYST|7566|03-DEC-81|3000||20
7934|MILLER|CLERK|7782|23-JAN-82|1300||10
[sxsama2@HMLINUX1 AWK]$ awk -f Exit.awk EMP.DAT
---------------------------------------------------------
              ANNUAL BONUS CALCULATION
---------------------------------------------------------
Calculating Bonus for employee:SMITH
Calculating Bonus for employee:ALLEN
Calculating Bonus for employee:WARD
Calculating Bonus for employee:JONES
Calculating Bonus for employee:MARTIN
Calculating Bonus for employee:BLAKE
Calculating Bonus for employee:CLARK
Calculating Bonus for employee:SCOTT
---------------------------------------------------------
Bonus Should not be calculated for - President, Terminating
Data Error Processing  Input File :EMP.DAT
7839|KING|PRESIDENT||17-NOV-81|5000||10
---------------------------------------------------------
---------------------------------------------------------
                      END OF REPORT
---------------------------------------------------------
[sxsama2@HMLINUX1 AWK]$
```

USE OF IGNORECASE ,ENVIRON

```
[sxsama2@HMLINUX1 AWK]$ awk '/smith/{print}' EMP.DAT
[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN{IGNORECASE=1}/smith/{print}' EMP.DAT
7369|SMITH|CLERK|7902|17-DEC-80|800||20
[sxsama2@HMLINUX1 AWK]$ echo $PWD
/home/sxsama2/AWK
[sxsama2@HMLINUX1 AWK]$ echo $HOME
/home/sxsama2
[sxsama2@HMLINUX1 AWK]$ echo $PATH
/u01/app/oracle/product/10.2.0/db_2/bin:/bin:/usr/bin:/usr/local/bin:/usr/X11R6/bin:/sbin
[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN{print ENVIRON["PWD"]}'
/home/sxsama2/AWK
[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN{print ENVIRON["HOME"]}'
/home/sxsama2
[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN{print ENVIRON["PATH"]}'
/u01/app/oracle/product/10.2.0/db_2/bin:/bin:/usr/bin:/usr/local/bin:/usr/X11R6/bin:/sbin
[sxsama2@HMLINUX1 AWK]$ export COMP=CLINK
[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN{print ENVIRON["COMP"]}'
CLINK
[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN{print "'$COMP'"}'
CLINK
[sxsama2@HMLINUX1 AWK]$ LOC=BANGALORE
[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN{print ENVIRON["LOC"]}'

[sxsama2@HMLINUX1 AWK]$ awk 'BEGIN{print "'$LOC'"}'
BANGALORE
[sxsama2@HMLINUX1 AWK]$
```

Example of Associate Array



User Defined functions

```
$ cat Multi_Table.awk
#!/bin/awk -f
## User Defined function
function usage()
{
      print "Usage: <Programname> <number> | <Programname> <number1> <number2>" ;
      exit;
}

BEGIN{
if (ARGC <2)
{
      usage()
}
if(ARGC==2)
{
      START_NUM=ARGV[1];
      END_NUM=START_NUM;
}
if(ARGC==3)
{
      if ( ARGV[1] < ARGV[2] )
      {
      START_NUM=ARGV[1];
      END_NUM=ARGV[2];
      }
      else
      {
            print "Second number should be higher than first number";
            usage();
      }
}

for(i=START_NUM;i<=END_NUM;i++)
{
      for(j=1;j<=10;j++)
      {
      printf "%-5s",i*j;
      }
      printf "\n";
}

}
```

User Defined functions

awktrn@localhost:~/AWK

```
[awktrn@localhost AWK]$ cat Multi_Table.awk
#!/bin/awk -f
## User Defined function
function usage()
{
        print "Usage: <Programname> <number> | <Programname> <number1> <number2>" ;
        exit;
}

BEGIN{
if (ARGC <2)
{
        usage()
}
if(ARGC==2)
{
        START_NUM=ARGV[1];
        END_NUM=START_NUM;
}
if(ARGC==3)
{
        if ( ARGV[1] < ARGV[2] )
        {
        START_NUM=ARGV[1];
        END_NUM=ARGV[2];
        }
        else
        {
                print "Second number should be higher than first number";
                usage();
        }
}

for(i=START_NUM;i<=END_NUM;i++)
{
        for(j=1;j<=10;j++)
        {
        printf "%-5s",i*j;
        }
        printf "\n";
}

}
[awktrn@localhost AWK]$
```

Filename & Variable as input during runtime

```
$ cat  Pattern_Lookup.awk
BEGIN{
system("clear");
printf "Enter FILE NAME(S): "
# Read the file name and Store the value in MYFILE Variable ##
getline MYFILE < "-"
printf "Enter the Pattern to Look up: "
## Read the pattern and store it in PATT variable ##
getline PATT < "-"
LCNT=0;
### Read the content in a loop ##
## We can also read and store in a variable ##
### Like getline myrec<MYFIL  ###
while(( getline <MYFILE) > 0 ) {
POS=0;
CNT=0;
LCNT++;
for(i=1;i<=NF;i++)
{
        if($i==PATT)
        {
                (POS!=0)?POS=POS","i:POS=i;CNT++;
        }
}
if(CNT !=0)
{
        printf "[Line #"LCNT" : Word Positions- "POS" : "PATT" is present - "CNT" times]\n";
}
}
}
```

Filename & Variable as input during runtime

sxsama2@localhost:~

```
[sxsama2@localhost ~]$
[sxsama2@localhost ~]$ cat  Pattern_Lookup.awk
BEGIN{
system("clear");
printf "Enter FILE NAME(S): "
# Read the file name and Store the value in MYFILE Variable ##
getline MYFILE < "-"
printf "Enter the Pattern to Look up: "
## Read the pattern and store it in PATT variable ##
getline PATT < "-"
LCNT=0;
### Read the content in a loop ##
## We can also read and store in a variable ##
### Like getline myrec<MYFIL  ###
while(( getline <MYFILE) > 0 ) {
POS=0;
CNT=0;
LCNT++;
for(i=1;i<=NF;i++)
{
        if($i==PATT)
        {
                (POS!=0)?POS=POS","i:POS=i;CNT++;
        }
}
if(CNT !=0)
{
        printf "[Line #"LCNT" : Word Positions- "POS" : "PATT" is present - "CNT" times]\n";
}
}
}
[sxsama2@localhost ~]$ awk -f Pattern_Lookup.awk
Enter FILE NAME(S): DATA
Enter the Pattern to Look up: CTL
[Line #1 : Word Positions- 1,3,6 : CTL is present - 3 times]
[Line #3 : Word Positions- 3,7 : CTL is present - 2 times]
[Line #4 : Word Positions- 1,3,4 : CTL is present - 3 times]
[Line #7 : Word Positions- 1 : CTL is present - 1 times]
[sxsama2@localhost ~]$
```

Thank You !!!!

Any Questions Mail me ☺

Subhasis.samantaray@centurylink.com
subhasis.samantaray@gmail.com