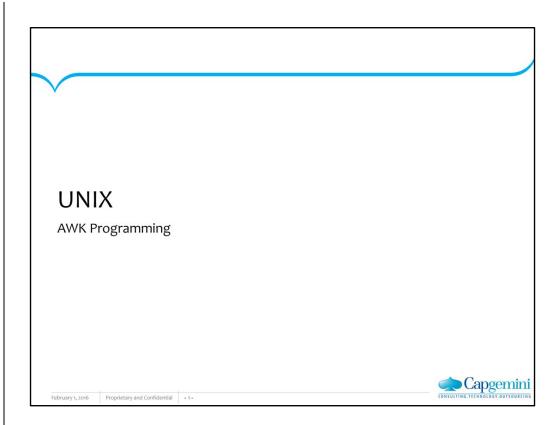
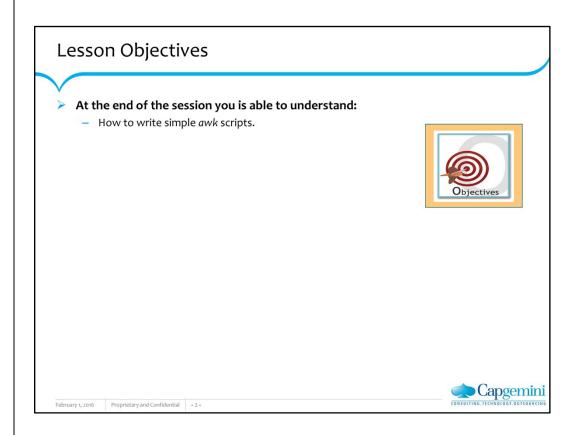
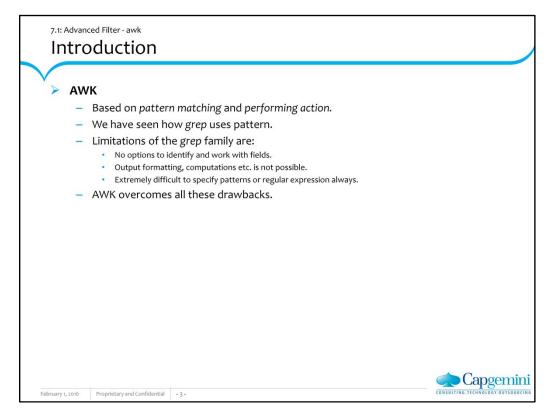
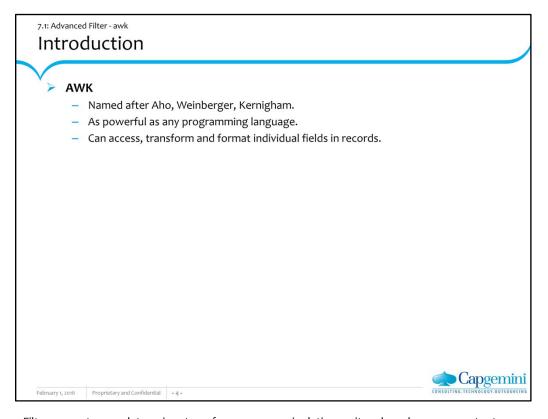
UNIX AWK Programming



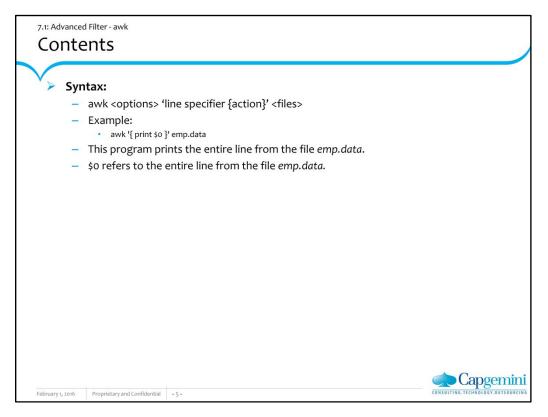




The awk command, named after its authors Aho, Weinberger and Kernighan, is one of the most powerful utilities for text manipulation. It combines features of many other filters. It can access, transform and format individual fields in a record – it is also called as a report writer. It can accept regular expressions for pattern matching, has "C" type programming constructs, variables and inbuilt functions. In fact, awk is nearly as powerful as any other programming language. However, awk programs are generally slow, if any alternative commands are available to do the same use them rather than using AWK.



Filters accept some data as input, perform some manipulation on it and produce some output. Some simple filters have been discussed in the previous chapters. This chapter discusses advanced filter –awk.



Advanced Filter awk.

The general syntax of the awk command is:

awk <options> 'line specifier {action}' <file(s)>

Simple awk Filtering

Following is an example of a simple awk command:

It prints all lines from file books. Ist in which pattern 'Computer' is found.

\$ awk '/Computer/ {print}' books.lst Output:

 1001 | Learning Unix
 | Computers
 | 01/01/1998 | 575

 1003 | XML Unleashed
 | Computers
 | 20/02/2000 | 398

 1004 | Unix Device Drivers
 | Computers
 | 09/08/1995 | 650

 1007 | Unix Shell Programming
 | Computers
 | 03/02/1993 | 536

UNIX AWK Programming

7.1: Advanced Filter - awk

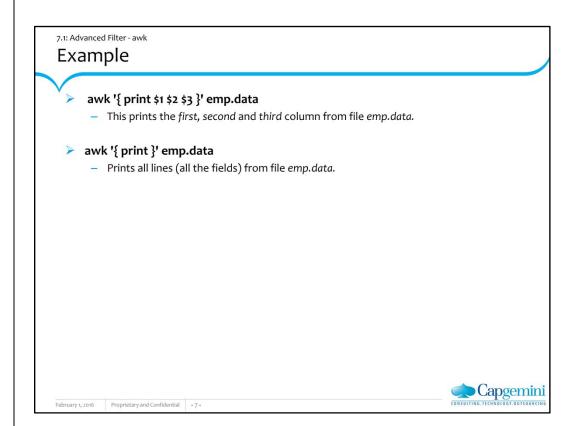
AWK variables

Variable List:

- \$0: Contains contents of the full current record.
- \$1..\$n: Holds contents of individual fields in the current record.
- NF: Contains number of fields in the input record.
- NR: Contains number of record processed so far.
- **FS:** Holds the field separator. Default is *space* or *tab.*
- OFS: Holds the output field separator.
- RS: Holds the input record separator. Default is a new line.
- FILENAME: Holds the input file name.
- ARGC: Holds the number of Arguments on Command line
- ARGV: The list of arguments

February 1, 2016 Proprietary and Confidential - 6 -





7.2 AWK variables

Overview

- Line specifier and action option are optional, either of them needs to be specified.
- If line specifier is not specified, it indicates that all lines are to be selected.
- {action} omitted, indicates print (default).
- Fields are identified by special variable \$1, \$2,;
- Default delimiter is a contiguous string of spaces.
- Explicit delimiter can be specified using -F option
 - Example: awk -F "|" '/sales/{print \$3, \$4}' emp.lst
- Regular expression of egrep can be used to specify the pattern.

February 1, 2016 Proprietary and Confidential - 8 -



The line specifier uses a context address to specify the lines that need to be taken up for processing in the action section. If the line specifier is missing, then the action is applicable to all lines of the file.

In the action part, statement {print} indicates that selected lines are to be printed. The statement {print} is equivalent to {print \$o} - \$o being the variable for the entire line. The awk command is also capable of breaking each line into fields – each field is identified as \$1, \$2 etc.

For the purpose of identification of fields, awk uses a contiguous string of spaces as field delimiter. However, it is possible to use a different delimiter. This is specified using the –F option in awk.

\$ awk -F"|" '/Computer/ {print \$2,\$5}' books.lst

Output:

Learning Unix 575 XML Unleashed 398 Unix Device Drivers 650 Unix Shell Programming 536

For pattern matching, awk uses regular expressions of egrep variety.

\$ awk -F"|" '/XML|Unix/ {print \$2, \$5}' books.lst

Output:

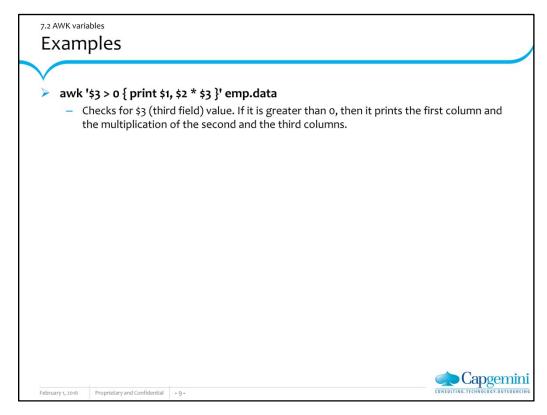
Learning Unix 575 XML Unleashed 398 Unix Device Drivers 650 XML Applications 630 Unix Shell Programming 536

It is possible to specify line numbers in file using the inbuilt NR variable. Also, awk can use the C-like printf statement to format the output.

\$ awk -F"|" '\$1=="1002" {printf "%2d,%-20s",NR,\$2}' books.lst Output:

2, Moby Dick

The –f option of awk is useful if you wish to store the line specifier or action in a separate file.



The logical operators \parallel (or) and && (and) are used by the awk command to combine conditions in the line specifier. A relational operator can be used in the line specifier, also in the action component.

The operators == (equal) and != (not equal) can handle only fixed length strings and not regular expressions. To match regular expressions, ~ (match) and !~ (negate) are used. The characters ^ and \$ can be used for looking for a pattern in the beginning and end of field.

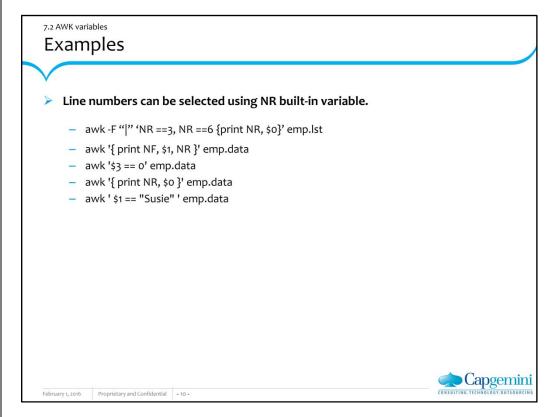
To work with numbers, operators like < (less than), > (greater than), <= (less than or equal), >= (greater than or equal), == (equal) and != (not equal) can be used.

It is possible to perform computations on numbers using C like arithmetic operators (+, -, *, /, *, ++, -, += etc).

\$ awk -F"|" '/Unix/ && \$5 < 600 {printf "%s,%d\n",\$2,\$5}' books.lst

Output:

Learning Unix ,575



awk -F "|" 'NR ==3, NR ==6 {print NR, \$0}' emp.lst

In this example, NR represents record number. It prints records from third record to sixth record.

-F is use to specify field separator.

awk '{ print NF, \$1, NR }' emp.data

This prints number of fields, contents of field 1 and record number for all records.

awk '\$3 == o' emp.data

It prints all lines in which the value in the third field is o.

awk '{ print NR, \$0 }' emp.data

It will print record number and record for all records

awk ' \$1 == "Susie" ' emp.data

It prints all lines in which the value in the first field is Susie.

7.3 Logical and Relational operators

Logical and Relational operator

Logical Operator &&, ||

```
$awk - F "|" '$3 =="director" || $3 == "chairman" {printf "%-20s",$2}' emp.lst
```

Relational Operators : <, <=, ==, !=, >=, >, ~, !~

```
\ $awk -F "|" '$6>7500 {printf "%20s", $2}' emp.lst $awk -F "|" '$3 == "director" || $6>7500 {print $0}' emp.lst
```

ebruary 1, 2016 Proprietary and Confidential - 11 -



The logical operators \parallel (or) and && (and) are used by the awk command to combine conditions in the line specifier. A relational operator can be used in the line specifier, also in the action component.

The operators == (equal) and != (not equal) can handle only fixed length strings and not regular expressions. To match regular expressions, \sim (match) and ! \sim (negate) are used. The characters $^$ and $^$ can be used for looking for a pattern in the beginning and end of field.

For working with numbers, operators like < (less than), > (greater than), <= (less than or equal), >= (greater than or equal), == (equal) and != (not equal) can be used.

It is possible to perform computations on numbers using C like arithmetic operators (+, -, *, /, *, ++, -, += etc).

Print all lines with Unix pattern in the line and value of 5 th field should be < 600:

\$ awk -F"|" '/Unix/ && \$5 < 600 {printf "%s,%d\n",\$2,\$5}' books.lst

Output

```
Learning Unix ,575
Unix Shell Programming ,536
$ awk -F"|" '$2=="Learning Unix"' books.lst
$ awk -F"|" '$2~/Learning Unix/' books.lst
1001|Learning Unix | Computers | 01/01/1998| 575
$ awk -F"|" '$5<500 {
> cnt=cnt+1
> printf "%d %s\n",cnt,$2}' books.lst
1 Moby Dick
2 XML Unleashed
```

Example of awk command using relational, logical expressions and computations.

7.3 Logical and Relational operators

Logical and Relational operators

== tries to find perfect match.

- String may have trailing spaces.
- To overcome this you can use "~" and "!~" (match and negate of match) with the regular expression.
- Example:

 $\$ \\$awk -F "|" '\\$2\"/director/||\\$2\"/g.m/\{printf \\$0\}' emp.\]st \\$3\"/\g.m/ #Beginning with g.m

February 1, 2016 Proprietary and Confidential - 12 -



7.4 Arithmetic operators

Arithmetic operators

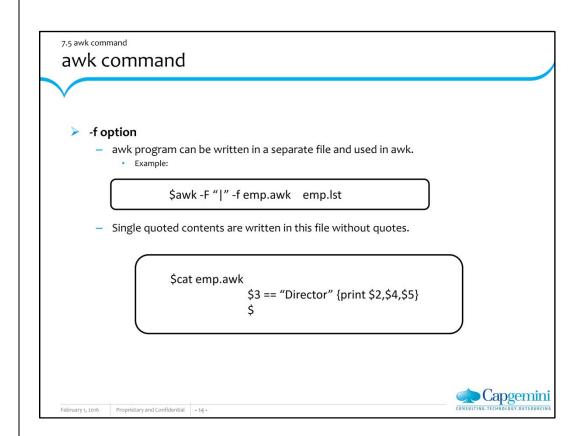
- Computation on numbers is done:
 - +, -, *, /, % operator available.
 - No type-declaration for variables.
 - Variables are initialized to zero.

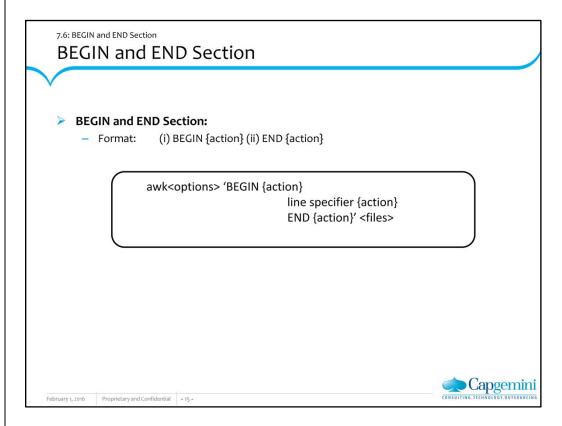
```
$awk -F "|" '$3 == "director" || $6>7500 {
             >kount = kount+1
             >printf "%3d%-20s\n", kount,$2}' emp.lst
```

Can also use C constructs like kount++, Kount+=sum etc.

February 1, 2016 Proprietary and Confidential - 13 -







BEGIN and END Sections

In case, something is to be printed before processing the first line begins, BEGIN section can be used. Similarly, to print at the end of processing, END section can be used.

These sections are optional. When present, they are delimited by the body of the awk program.

Normally if you want to print any header lines or want to set field separator then use BEGIN section

And If you want to display total or any summarized information at the end of the report then use END section

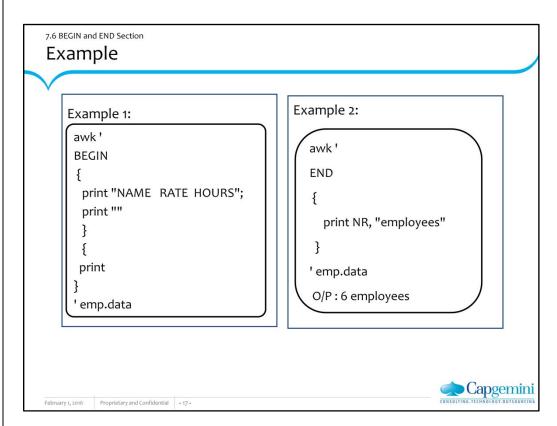
```
$cat emp.awk

BEGIN { printf "\n\t Employee details \n\n" }
$6>7500{

# increment sr. no. and sum salary
kount++; tot+=$6
printf "%d%-20s%d\n", kount, $2, $6
}

END { printf "\n The Avg. Sal. Is %6d\n", tot/kount
}
$awk -F "|" -f emp.awk emp.lst
```

```
$ cat awk1.awk
BEGIN {
printf "\n\t Unix Book Details \n\n"
}
$2~/Unix/ {
cnt++; tot+=$5
printf "%d %-15s\n",cnt,$2
}
END {
printf "\n Total Cost is %d\n",tot
}
$ awk -F"|" -f awk1.awk books.lst
Output:
    Unix Book Details
1 Learning Unix
2 Unix Device Drivers
3 Unix Shell Programming
Total Cost is 1761
```



Example 1:

In this example It will perform action enclosed in BEGIN section (It prints headings) and then it will perform print action on all lines in the file emp.dat (because no line specifier is mentioned)

Example 2:

In this example BEGIN section is not given. The main action part is also empty hence no action is performed on all the records and then it will execute End section which prints last record number which will give you total number of records.

7.7 Positional parameters and shell variable

Positional parameters and shell variable

- Requires entire awk command to be in the shell script.
- Differentiate positional parameter and field identifier
 - Positional parameter should be single-quoted in an awk program.
 - Example: \$3 > '\$1'



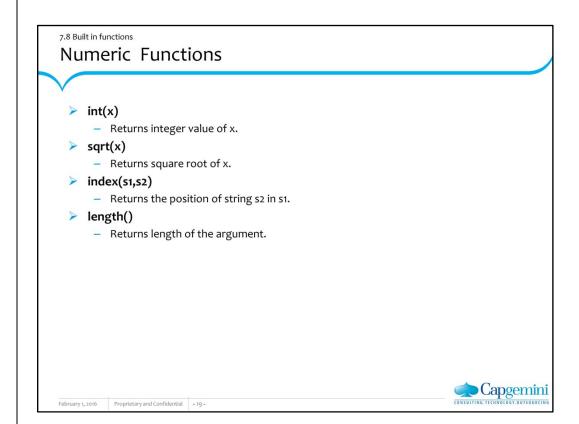
ebruary 1, 2016 Proprietary and Confidential - 18 -

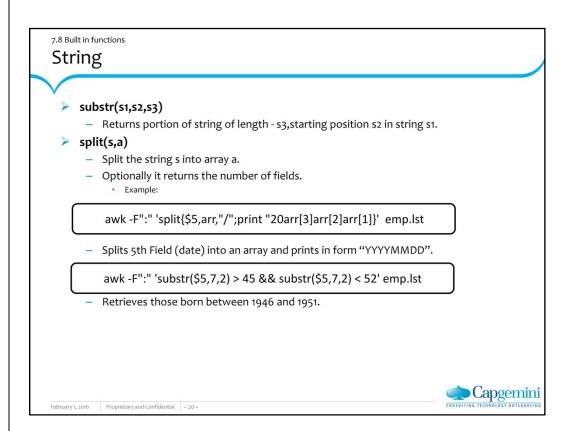
Positional Parameters and Shell Variables

It is possible to store an entire awk command into a file as a shell script, and pass parameters as arguments to the shell script. The shell will identify these arguments as \$1, \$2 etc based on the order of their occurrence on the command line.

Within awk, since \$1, \$2 etc. indicate fields of data file, it is necessary to distinguish between the positional parameters and field identifiers. This is done by using single quotes for the positional parameters used by awk.

```
$ cat awk2.awk
awk -F"|" '$5>'$1' {
cnt++
printf "%d %s %d\n",cnt,$2,$5 }
END {
printf "\n No. of books costing more than specified amount is:%d\n",cnt
}' books.lst
$ awk2.awk 600
Output:
1 Unix Device Drivers 650
2 Complete Works:Sherlock H 1290
3 XML Applications 630
No. of books costing more than specified amount is:3
$ awk2.awk 1000
1 Complete Works:Sherlock H 1290
No. of books costing more than specified amount is:1
(Example of awk command using positional parameter.)
```



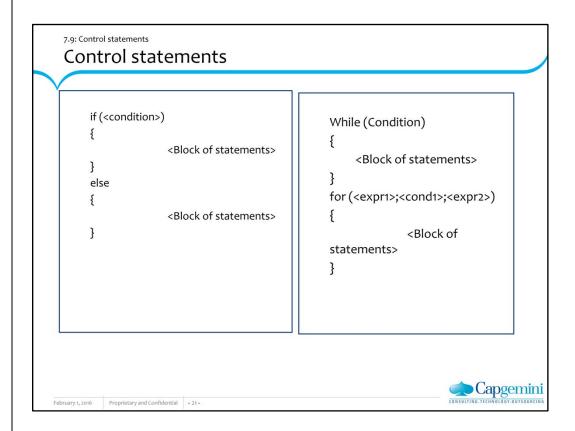


awk -F":" 'length > 1024' emp.lst Retrieves all lines > 1024. awk -F":" 'length(\$2) > 11' emp.lst

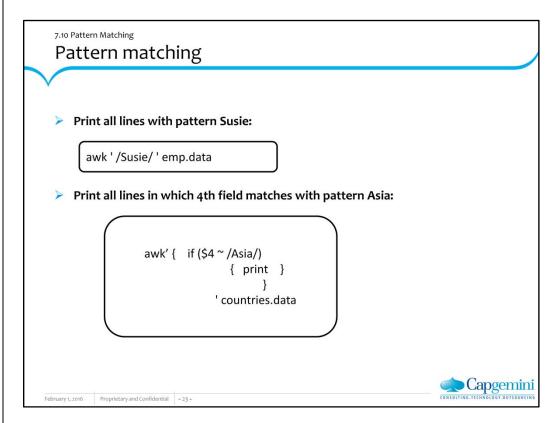
Retrieves all lines whose characters in 2nd col < 11.

Retrieves those born between 1946 and 1951.

awk -F":" 'substr(\$5,7,2) > 45 && substr(\$5,7,2) < 52' emp.lst

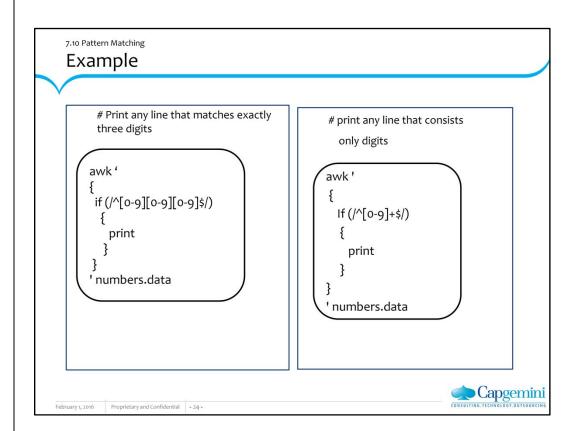


```
7.9: Control statements
Example
                awk '
                {
                  if (NF!= 3) {
                    print $0, "number of fields is not equal to 3"
                  if ($2 < 3.35) {
                    print $0, "rate is below minimum wage" }
                  if ($2 > 10) {
                    print $0, "rate exceeds $ 10 per hour" }
                  if ($3 < 0) {
                    print $0, "negative hours worked" }
                  if ($3 > 60) {
                    print $0, "too many hours worked" }
                }
                ' emp.data
                                                                             Capgemini
February 1, 2016 Proprietary and Confidential - 22 -
```



```
To print all lines in which 4 th field does not matches with pattern Asia: awk'
{
    if ($4!~/Asia/)
    {
        print
    }
}
'countries.data
```

UNIX



Summary

- AWK is based on pattern matching and performing action.
- Commands enclosed in BEGIN section gets executed first.
- Then. it performs main action part on all records.
- Commands enclosed in END section gets executed.



February 1, 2016 Proprietary and Confidential -25 - Capge mini

Add the notes here.

Review Questions In ----- section report header can be printed ------ section helps to print totals Print \$0 prints whole record -- TRUE -- FALSE February 1, 2016 Proprietary and Confidential -26- Proprietary and Confidential -26- Proprietary 2016 Pr

Add the notes here.