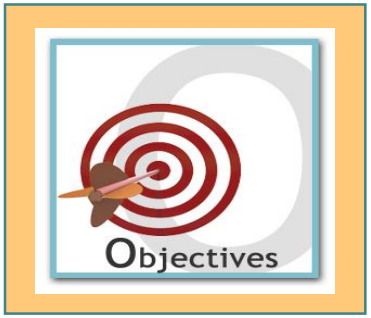
UNIX

AWK Programming

Lesson Objectives

- **At the end of the session you is able to understand:**
  - How to write simple *awk* scripts.



Introduction

- **AWK**
  - Based on *pattern matching* and *performing action.*
  - We have seen how *grep* uses pattern.
  - Limitations of the *grep* family are:
    - No options to identify and work with fields.
    - Output formatting, computations etc. is not possible.
    - Extremely difficult to specify patterns or regular expression always.
  - AWK overcomes all these drawbacks.

Introduction

- **AWK**
  - Named after Aho, Weinberger, Kernigham.
  - As powerful as any programming language.
  - Can access, transform and format individual fields in records.

Contents

- **Syntax:**
  - awk <options> ‘line specifier {action}’ <files>
  - Example:
    - awk '{ print $0 }' emp.data
  - This program prints the entire line from the file *emp.data*.
  - $0 refers to the entire line from the file *emp.data.*

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

Example

- **awk '{ print $1 $2 $3 }' emp.data**
  - This prints the *first*, *second* and *third* column from file *emp.data.*

- **awk '{ print }' emp.data**
  - Prints all lines (all the fields) from file *emp.data.*

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.**

Examples

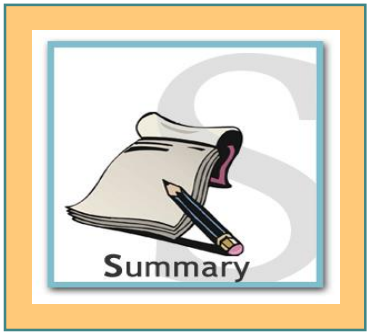
- **awk '$3 > 0 { print $1, $2 * $3 }' emp.data**
  - Checks for $3 (third field) value. If it is greater than 0, then it prints the first column and the multiplication of the second and the third columns.

Examples

- **Line numbers can be selected using NR built-in variable.**
  - awk -F “|” ‘NR ==3, NR ==6 {print NR, $0}’ emp.lst
  - awk '{ print NF, $1, NR }' emp.data
  - awk '$3 == 0' emp.data
  - awk '{ print NR, $0 }' emp.data
  - awk ' $1 == "Susie" ' emp.data

Summary

- **AWK is based on pattern matching and performing action.**
- **Various built in variable of AWK**
- **Extracting field from file using AWK**



Review Questions

- **Which variable is used to print number of records processed by AWK ?**
- **How many times action block is executed in AWK?**
- **Print $0 prints whole record**
  - TRUE
  - FALSE