IT Scripting and Automation

Introduction to AWK

Lecturer: Art Ó Coileáin

AWK

- The **awk** is mostly used for pattern scanning and processing.
- It searches one or more files to see if they contains lines that matches with the specified patterns and then perform associated actions.
- Awk views a text file as record and fields.
- Like common programming language, awk has variables, conditionals and loops.
- It has arithmetic and string operators and generate formatted reports.
- **It can only process text files**.

AWK: Syntax

- **Syntax**:

  awk *'/search pattern1/ {Actions}*

  /search pattern2/ {Actions}' file

- This expressions:
- ***Search pattern***-is a regular expression
- ***Action*** – statement (s) to be performed
- Several patterns and actions are possible in awk
- ***File*** - is the inputfile
- Single quotes around program is to avoid shell not to interpret any of its special characters.

AWK: How does it work?

- It reads the input files one line at a time
- For each line, it matches with given pattern in the given order, if matches performs the corresponding action.
- If no pattern matches, no action will be performed.
- In the previous syntax, either search pattern **or** action are optional, but not both.
- If the search pattern is not provided, awk performs the given actions for each line of the input.
- If the action is not given, print all that lines that matches with the given pattern.
- Empty curly brackets ({ }) without any action does nothing.
- Each statement in Actions should be delimited by semicolon (;)

AWK: Example

- Example: Let's have a look at a sample text file called "employee":

```
student@ITSA-Server:~/awk$ cat employee
1 John Manager Sales $ 40000
2 Mark Manager Technology $ 50000
3 Joe Developer Technology $ 30000
4 Sean Officer HR $ 35000
5 David Officer Sales $ 36000
```

- What are the outputs of the following commands?

$ awk '/Manager/' employee

-all the lines with matches with the pattern 'Manager'

$ awk '{print $2, $6;}' employee

-awk splits each record delimited by whitespace character by default it stores it in the $n variables.

$2 represents the second column (field) and

$5 refers to the fifth column (field)in a record.

-$NF ***represents the last field***.

AWK: Syntax (2)

- **Awk** has two special patters which can be specified by the keywords **BEGIN** and **END**.

Syntax:

awk 'BEGIN { Actions}

[/search pattern/] {ACTION} # Action for every line in a file

END { Actions } '

- Actions specified in the BEGIN section will be executed before starts reading the lines from the input.
- Actions specified in END section will be performed after completing the reading and processing the lines from the input.

AWK: Field Separator

- The default field separator in awk is : “ ” (whitespace)
- You can change the Field separator in the **BEGIN** section using FS=“field separator”.

Example:

- **$ awk ‘BEGIN { FS = “,” ;} {print $2;} ’ employee**

- What is the output of the following statements?
  - $ echo “1,2,3,4” | awk ‘BEGIN {FS=“,”;} {print $1;}’
  - $ echo “1,2,3,4” | awk ‘{print $1;}’

AWK Examples

- What are the outputs of the following awk statements?

- **$ awk ‘{sum=sum+$6;} END{print “Total salary: ” sum;}’ employee**

- **$ awk ‘{sum=sum+$6; print “Total salary: ” sum;}’ employee**

- **$ awk ‘{if($6 > 40000){print $2” –“$6; }}’ employee**

- **$ awk ‘$6>40000’ employee**

AWK Exercise

- Create an **awk** program to count the number of ‘managers’ in the file ‘employee’.

AWK Example

- What is the output of the following **awk** statement?

```
$ awk 'BEGIN {print "Name\tRole\tDepartment\tSalary";}
{print $2,"\t",$3,"\t",$4,"\t",$NF;}
END{print “End of Report \n----------"; }’ employee
```

AWK in Shell Scripts

- What does this script do?

(scriptAwk.sh)

```
#!/bin/bash
directory=$1
file=$2
path=$1/$2
cat $path | awk 'BEGIN{FS=":";}{print $1;}' | sort
```

Reference

- The GNU awk User's Guide:

https://www.gnu.org/software/gawk/manual/html_node/index.html#SEC_Contents