Seneca | SCHOOL OF INFORMATION AND COMMUNICATIONS TECHNOLOGY

Unix Scripting

Lecturer: Shahdad Shariatmadari

June 2020

What we have learned…

- Introduction to Shell Scripting
  - Categories of variables
  - Conditional Statements
  - Loops
- stdin, stdout, stderr Redirection and piping
- File descriptor
- Filtering
  - Simple filter commands: head, tail, cut, sort, wc
  - grep utility
- Multiple commands : () vs {}
- Sed, Awk, Grep

Agenda

- More on Regular Expressions

Activity

- Try the sample queries in the BB (cars.txt)

AWK syntax

- awk [options] '/re/ {execution}' filename
- Options:
  -f scriptfilename (**.awk**, execute from script)
  -F";"   (sets default delimiter)
- AWK Simple form:
  - **awk** *search pattern* **{** *program actions* **} filename**

Example

- awk '/gold/ {print $5,$6,$7,$8}' sample.txt
- The $5, $6, $7, and $8 are **fields**
  - WHICH is defined as any string of printing characters separated by spaces.

- Note that Awk recognizes the field variable $0 as representing the entire line.

Introduction to awk Utility

- **Variables**
  - The following is a list of common variables used with awk:

$0	Entire Record
$n	*Field number* “n” in Record (eg. $1, $2, $3)
NF	Number of fields in record
NR	Record number of current record
FS	Input Field Separator (default space / tab)
OFS	Output Field Separator (default space)
RS	Input Record Separator (default new line)
ORS	Output Record Separator (default new line)
FILENAME	Name of current input file

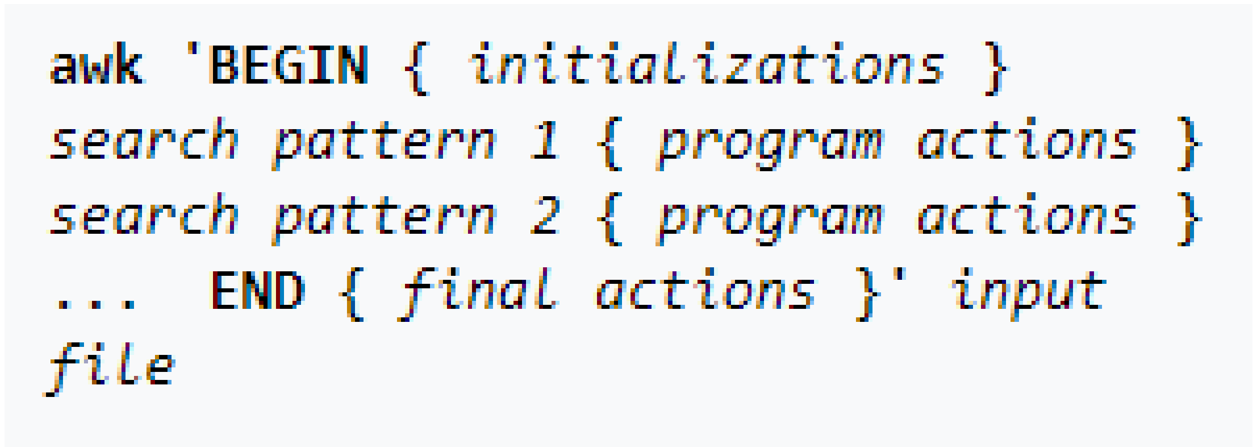
Print and printf

- Here are some common commands that can be used in the execution of awk (contained in braces { }):
- print   Can use variables like $1,$2, etc. When using those types of variables separate with a comma (no spaces). The comma represents the default output field separator.

- printf   very similar to print but provides formatting options for the display of values (eg. # of decimal places) (refer to examples in Sample Script section of this week's resources…)

Awk

- AWK can have an optional `BEGIN{}` section of commands that are done before processing any content of the file, then the main {} section works on each line of the file, and finally there is an optional `END{}` section of actions that happen after the file reading has finished



```
awk 'BEGIN { initializations }
search pattern 1 { program actions }
search pattern 2 { program actions }
...  END { final actions }' input
file
```

Generating Reports

- The awk utility can use the BEGIN directive in its expression to indicate execution to be formed at the beginning of the report (i.e. before reading in the lines from a file for processing)

- Example:

```
awk 'BEGIN {print “REPORT TITLE”}  /re/ { print }'
filename
```

Generating reports

- The awk utility can also use the END directive in its expression to indicate execution to be formed at the end of the report (i.e. before reading in the lines from a file for processing)

- Example (using both BEGIN and END directive):

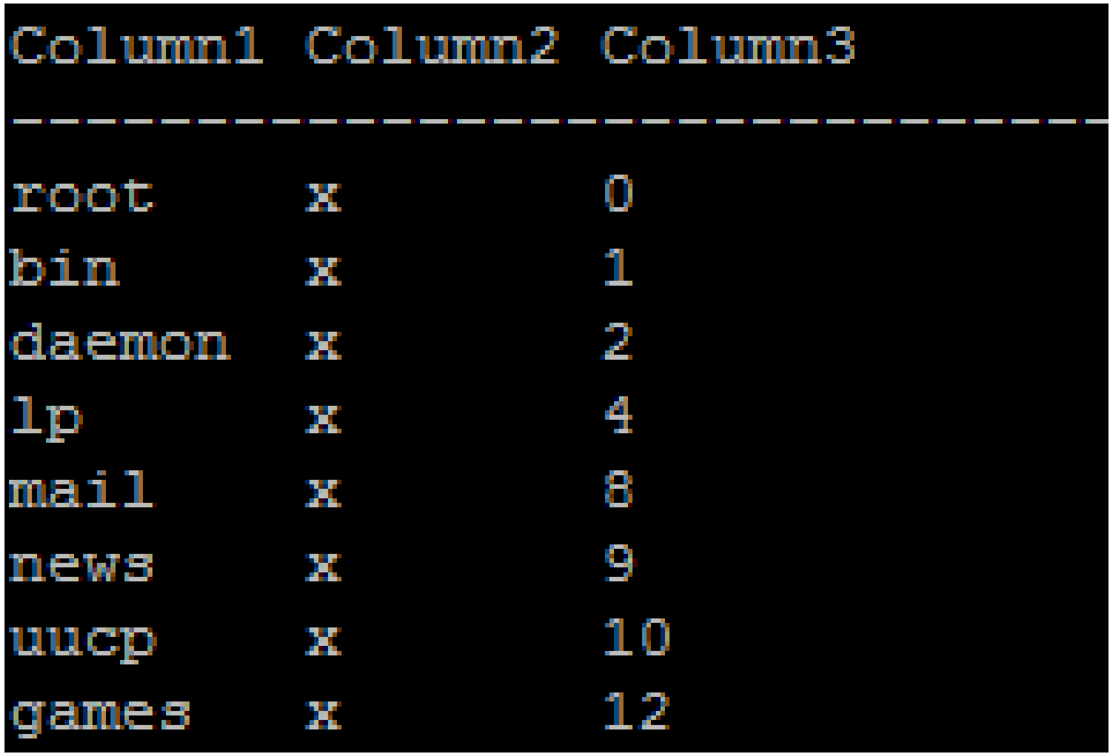
```
awk 'BEGIN {print “REPORT TITLE”} /re/ { execution }
END { print “END OF REPORT” }' filename
```

Activity: What does the following command do? Explain it.

- awk 'END { print NR }' Sample.txt

Activity

- Using awk/grep/sed , Display the first 3 columns of /etc/passwd with Heading titles (column1, column2,..) as follow:



```
Column1 Column2 Column3
-------------------------------------
root    x       0
bin     x       1
daemon  x       2
lp      x       4
mail    x       8
news    x       9
uucp    x       10
games   x       12
```

Activity

- echo "Welcome To The Geek Stuff" | sed 's/\(\b[A-Z]\)/\(\1\)/g‘