Unix Scripting

Week5-Session1

Agenda

- More on Regular Expressions
- Globing shell options
- Extended Globing
- Named Character Classes

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**

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

awk arithmetic

- What does the following command do?
- awk '{inventory+=$5} END {print "Our total inventory has a value of $" inventory}' cars

Observation: What does the following commands do?

- awk ‘BEGIN { print NR }' Sample.txt
- awk 'END { print NR }' Sample.txt
- awk '{inventory+=$5} END {print "Our total inventory has a value of $" inventory}' cars

Globing shell options

- Using wild characters like : * and ?
  - ls t?.*
- Using []
  - ls make.[1-3]
  - ls [^abc]
- Using {}
  - touch myfile{1..10}
  - echo {1..10}
  - echo {1..10..2}

Observation: Globing shell options

- Create a script, myscript.sh, add the followings:
  - echo $10
  - echo ${10}
- Run “ myscript.sh p1 p2 p3 p4 p5 p6 p7 p8 p9 p10
- What happen?
- Run myscript.sh pABC p2 p3 p4 p5 p6 p7 p8 p9 p10
- What happen?

Question:

- What does the following command do?
- echo *
- echo *5

shopt

- **shopt** is a builtin command of the Bash shell which can enable or disable options for the current shell session.
- **shopt** [**-o**] [**-p**] [**-q**] [**-s**] [**-u**] [*optname*...]
- https://www.computerhope.com/unix/bash/shopt.htm

Try the following examples:

- nullglob - non-matching globs are removed, instead of preserved
  echo [0-9]
  shopt -s nullglob
  echo [0-9]
- failglob - non-matching globs cause an error, command is not executed
  echo [0-9]
  shopt -s failglob
  echo [0-9]
- nocaseglob - matches are done ignoring case
  echo file*5
  shopt -s nocaseglob
  echo file*5

Extended Globing

- **extended globbing** may be enabled via a shell option: `shopt -s extglob`, but is on by default
- It allow us to add
  - **?(pattern-list) :**Matches zero or one occurrence of the given patterns
  - ***(pattern-list) :** matches zero or more occurrences of the given patterns
  - **+(pattern-list) :** matches one or more occurrences of the given patterns
  - **@(pattern-list) :** matches one of the given patterns
  - **!(pattern-list) :** matches anything except one of the given patterns

Example

- ls pic*.jp?(e)g
- ls pic*(3).*
- ls pic+(3).*
- ls pic*@(jpg|gif)
- ls pic!(*jpg|*gif)
- More example in Bash Extended Globing: https://www.linuxjournal.com/content/bash-extended-globbing

Named Character Classes

- Named character classes are useful, ensuring that collating sequences are correct regardless of the locale
- [:alnum:] - alphanumeric - same as [:alpha:] and [:digit:]
- Can be used with TR
- can be used within regular expressions, including within the "[[...]]" structure (must be enclosed within a second set of square brackets)

tr command in Linux

- **tr** is used to translate characters to different characters
- **tr a A < filename**
  - translate all characters "a" to "A“
- **tr ' ' '\n' < filename**
  - translate all spaces to newline characters
- **tr -d '\n' < filename**
  - delete all newline characters
- tr "[:lower:]" "[:upper:]" < cars
  - What does this do?