CS197U: A Hands on Introduction to Unix

Lecture 5: Programming & Scripting

Tian Guo
University of Massachusetts Amherst – CICS

Reminders

- Assignment 3 is due this Thursday Oct 1 at 3:45 pm

- Last time:
  - There are lots of Linux distributions
  - Virtualization is a technique to let you run multiple OS at once

Outline

- **File Processing Utilities**
  - Awk
  - Sed
- Shell Scripting
  - automating command line tasks
- Programming in Linux
  - compilers and accessories

`awk` - processing columns of data

- Awk is a utility for processing **structured data**
  - Anything that has multiple entries containing the same fields
  - Example:
    - ‘Contacts’ file where each contact has a name & phone number
      - Awk could be used to print just the phone numbers
- Awk is a line-based utility
  - For each line, awk checks if certain conditions are met. If so, awk performs some actions.
  - Useful for processing log files, experimental data, etc

`awk` - processing columns of data

awk [-F fs] [-v var=value] ['prog' | -f progfile] [file ...]

field separator

variables

script in ‘ ’ or in file

data file

- Awk program from command line:

```
awk -F “,”   '{print $2}'   data.txt
```

- Awk program from a script:

```
awk -F “,”   -f script.awk   data.txt
```

`awk` - processing columns of data

awk [-F fs] [-v var=value] ['prog' | -f progfile] [file ...]

field separator

variables

script in ‘ ’ or in file

data file

- Use Awk as an interpreter:

./script.awk data.txt

- As shell script:

./script.sh data.txt

`awk` command line examples

- Value of a column = $1, $2, $3 for column 1, 2, 3, etc
- To print out the second column:

```
awk    '{print $2}'   data.txt
```

- Output:

```
10
20
30
40
50
```

data.txt

```
1 10 xyz 100
2 20 abc 200
3 30 def 300
4 40 ade 400
5 50 f2d 500
```

- Remember: Awk operates on lines
- How about $0?

`awk` command line examples

- Can print arbitrary characters within quotes:

```
awk   '{print $2 "," $4}'   data.txt
```

- Output:

```
10, 100
20, 200
30, 300
40, 400
50, 500
```

data.txt

```
1 10 xyz 100
2 20 abc 200
3 30 def 300
4 40 ade 400
5 50 f2d 500
```

```
awk   '{print "col1=" $1 " and col4=" $4}'   data.txt
```

single quotes around program

double quotes for text

`awk` script file examples

- Can also put awk script into a file

```
awk    -f simple.awk   data.txt
```

- Output is the same as the previous example

simple.awk

```
{print $2}
```

- Can have multiple sections of code
  - Separate with braces {}
  - BEGIN, END for start, end of file

sections.awk

```
BEGIN {print “Column 2,4”}
{print $2“, ”$4}
END {print “The end.”}
```

```
Column 2,4
10, 100
20, 200
30, 300
40, 400
50, 500
The end.
```

`awk` exercise

awk '{print $2 “,” $4}' data.txt

sections.awk

```
BEGIN {print “Column 2,4”}
{print $2“, ”$4}
END {print “The end.”}
```

data.txt

```
1 10 xyz 100
2 20 abc 200
3 30 def 300
4 40 ade 400
5 50 f2d 500
```

- Write the awk command to print the word “Sum:”, followed by the sum of columns 1 and 4

`awk` exercise

awk '{print $2 “,” $4}' data.txt

sections.awk

```
BEGIN {print “Column 2,4”}
{print $2“, ”$4}
END {print “The end.”}
```

data.txt

```
1 10 xyz 100
2 20 abc 200
3 30 def 300
4 40 ade 400
5 50 f2d 500
```

- Write the awk command to print the word “Sum:”, followed by the sum of columns 1 and 4

awk 'BEGIN{print "Sum:"}{print $1+$4}' data.txt

Sum:
101
202
303
404
505

`awk` script file examples

- Can define and track variables by giving them a name
  - By default, set to 0

totals.awk

```
{
tot1=tot1+$1
tot4=tot4+$4
print $1 “ “ $4
}
END {print “TOTAL: ” tot1 “ “ tot4}
```

- This example prints the sum of columns 1 and 4

`awk` Build-in Variables

- FS: field separator
  - By default: white space;
  - Can use –F or FS to set the new field separator.
  - FS is more flexible
    - can be more than one character;
    - Can switch between different separators “easily”
  - Example: passwd file, semi-unstructured data;
- OFS: output field separator
  - By default: white space
  - Can be any number of characters
  - Example: delete a column from the file

`awk` Build-in Variables

- NF: num. of field variables
  - Counts how many fields per line
  - Example: access the last field
- NR: num. of records
  - Select certain lines
  - Example: cat | tail
- RS: record separator
  - Redefine “line” to awk
  - Examples: a word a line

`awk` Build-in Variables

- ORS: output record separator
  - Redefine the output format
  - Example: add CRLF for windows file
- FILENAME: the current file name
  - Example: process multiple files at the same time