CST8177 – Linux II

awk

Introduction

- What is **awk?**

  Fully functional programming language written for processing text and numbers

  Small, fast, and simple

  Works field by field

  - As opposed to grep, which works line by line

  - **Field** = a column of data, separated by a delimiter (e.g., a space, a comma, etc.)
  - **Record** = line of input

  Meant for processing column-oriented data, like tables

Usage

- **awk [-F *delim*][-v *var=value*] ‘*pattern* {*action*; *action* }’ filename**
  - **Basic idea:** match a **pattern** → perform **action(s)**
    - If no pattern → apply action(s) to all records/lines
    - If no action → default is to print entire record
  - ***Options and option arguments:***
    - **-F** = specify field **delimiter** (default: whitespace)
    - **-v** = define a **variable with its value** to be used within awk
      - E.g., -v bird=5
  - Reads from STDIN and writes to STDOUT
    - Can be used as a filter!

Accessing Columns/Fields

- In awk, the **nth field** is referred to by the variable **$n**

  E.g., $1 → first field, $2 = second field

- **$0** → refers to the **whole line**

- ***Examples:***

  awk '{print $1}'

  - Prints first field of each line

  awk '{print $0}'

  - Prints each line

Changing the Delimiter

- To use tabs as the delimiter:

  awk –F '\t' '{print $4}'

- Another example, this time using colons as delimiter:

  awk –F: '{print $6}' /etc/passwd

  - Prints home directory (sixth field) for each user

Printing Multiple Fields

- You can print multiple fields as well:

  awk –F\t '{print $1,$4}

  - Prints first and fourth field **separated by a space**
  - **Comma here → space**

  awk –F\t '{print $1 $4}

  - Prints first and fourth field **WITHOUT a space**
  - **No comma → no space**

Printing Other Text

- Basically, you can print variables and other text concatenated together:

  awk –F\t ‘{print “The”, $1, “weighed”,$4,”!”}’

  - Spaces inserted where commas are
  - NOTICE:
    - **Single quotes** for the **outside**!
    - ***Double quotes*** for the ***inside***!

Other Useful Variables

- NR = Current record (line) number
- NF = Number of fields on the current line (columns)
  - $NF = Last field
- FS = field separator (defaults to white space)
- OFS = output field separator
  - Allows you to change output delimiter

Quick Example

- Run the following:
  - **awk ‘{print “Current line:”,NR,”Field Count:”,NF}’**
- Awk is now waiting for text (or EOF, which when read will cause awk to stop)

- Enter some numbers separated by whitespace, then hit Enter
- Awk should then print the current line number and field count
- CTRL+D → send EOF

Regular Expression Patterns

- You can also check for a regular expression pattern
  - Must be enclosed with //
  - Matches somewhere in the line (similar to grep)

- Examples

```
awk –F\t '/abc/ {print $1}'
```

Prints first field if line contains "abc"