The awk tool

Enrique Arias
Universidad de Castilla–La Mancha

Contents

Contents

Main objectives

- Programming and automation of tasks using the awk tool
- Awk reference: https://www.computerhope.com/unix/uawk.htm

Contents

Contents

Introduction

- Awk is both a programming language and text processor that can be used to manipulate text data in very useful ways.
- awk [-F fs] [-v var=value] ['prog' | -f progfile] [file ...]
- Awk options
  - -F fs → Sets the input field separator to the regular expression fs.
  - -v var=value → Assigns the value value to the variable var before executing the awk program.
  - 'prog‘ → An awk program.
  - -f progfile → Specify a file, progfile, which contains the awk program to be executed.
  - file ...→ A file to be processed by the specified awk program.

Introduction

- awk '/search_pattern/ { action_to_take_on_matches; another_action; }' file_to_parse
  - You can omit either the search portion or the action portion
  - By default, the action taken if the "action" portion is not given is "print". This simply prints all lines that match.
  - If the search portion is not given, awk performs the action listed on each line.
  - If both are given, awk uses the search portion to decide if the current line reflects the pattern, and then performs the actions on matches.
  - The special patterns BEGIN and END may be used to capture control before the first input line is read and after the last. BEGIN and END do not combine with other patterns.
  - Patterns are arbitrary Boolean combinations (with ! || &&) of regular expressions and relational expressions (>,<,==,>=,<=, !=, ~, !~

Introduction

- awk '/search_pattern/ { action_to_take_on_matches; another_action; }' file_to_parse
  - More about actions
    - If(expression) statement [else statement]
    - while(expression) statement
    - for(expression ; expression ; expression) statement
    - for(var in array) statement
    - do statement while(expression)
    - break
    - continue
    - { [statement ...] }
    - expression
    - print [expression-list] [> file]
    - printf format [, expression-list] [> file]
    - return [expression]
    - next

Introduction

- awk '/search_pattern/ { action_to_take_on_matches; another_action; }' file_to_parse
  - Predefined functions
    - Length → The length of its argument taken as a string, or of $0 if no argument.
    - substr(s, m, n) → The n-character substring of s that begins at position m counted from 1.
    - index(s, t) → The position in s where the string t occurs, or 0 if it does not.
    - system(cmd) → Executes cmd and returns its exit status.
    - getline → sets $0 to the next input record from the current input fil

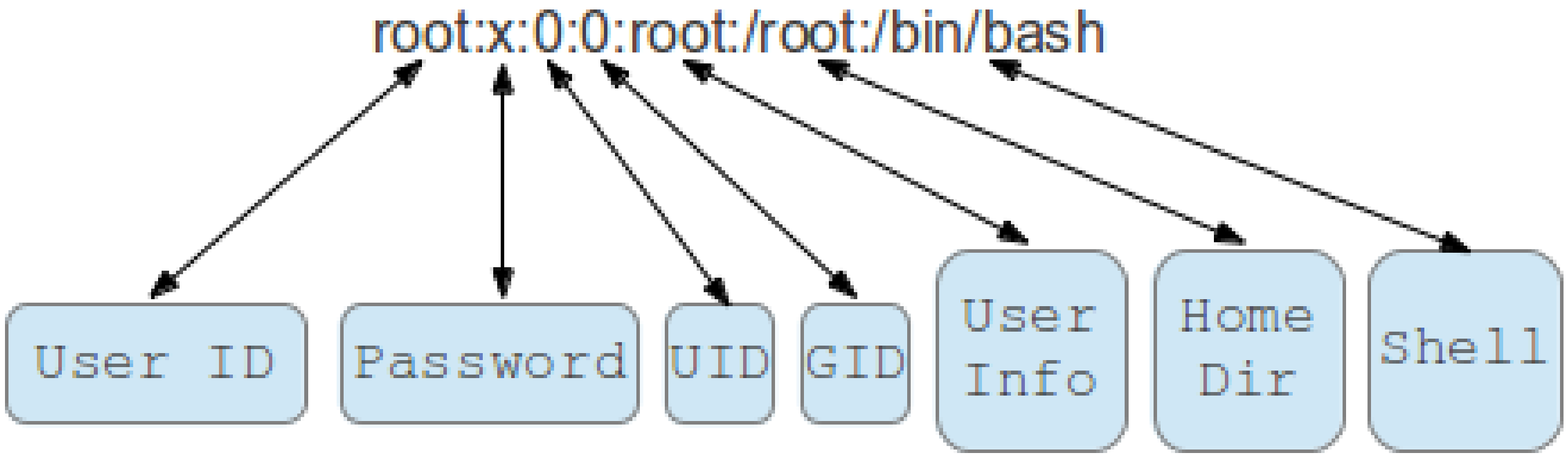
Introduction

- Awk Internal Variables
  - FILENAME: References the current input file.
  - FNR: References the number of the current record relative to the current input file. For instance, if you have two input files, this would tell you the record number of each file instead of as a total.
  - FS: The current field separator used to denote each field in a record. By default, this is set to whitespace.
  - NF: The number of fields in the current record.
  - NR: The number of the current record.
  - OFS: The field separator for the outputted data. By default, this is set to whitespace.
  - ORS: The record separator for the outputted data. By default, this is a newline character.
  - RS: The record separator used to distinguish separate records in the input file. By default, this is a newline character.
  - $0: Complete record
  - $i>0: Filed I on current record

Contents

Examples

- /etc/passwd File Format



Kike:x:1000:1000:"Enrique Arias” :/home/Kike:/bin/bash

Examples

- Print the file /etc/passwd

```
#!/bin/bash
awk '{print $0}' /etc/passwd
```

- Print the file /etc/passwd numbering each record

```
awk '{print NR,$0}' /etc/passwd
```

- Print first field of /etc/passswd file. Default separator “”

```
awk '{print $1}' /etc/passwd
```

Examples

- Print file /etc/passwd
  - Fields 1 and 5.
  - Field 1 has a width of 10 characters, left justification.
  - Filed separator is : (see the file)

```
awk -F: ’{printf "%-10s %s\n",$1,$5}’ /etc/passwd
```

- Print file /etc/passwd
  - Records from 3 to 12
  - Numbering the records

```
awk ’NR==3,NR==12{print NR,$0}’ /etc/passwd
or
awk ’FNR>=3 && FNR<=12{print NR,$0}’ /etc/passwd
```

Examples

- Print file /etc/passwd
  - Fields 1 and 5.
  - Field 1 has a width of 10 characters, left justification.
  - Filed separator is : (see the file)
  - Only those users whose name stars by r

```
awk -F: ’/^r/{printf "%-10s %s\n",$1,$5}’ /etc/passwd
```

- Print file /etc/passwd
  - Fields 1, 5 and 7.
  - Field 1 has a width of 10 characters, left justification.
  - Filed separator is : (see the file)
  - Only those users which use the bash shell

```
awk -F: ’$7==“/bin/bash”{printf "%-10s %s %s\n",$1,$5,$7}’ /etc/passwd
```

Examples

- Print file /etc/passwd
  - Fields 1, 5 and 7.
  - Field 1 has a width of 10 characters, left justification.
  - Filed separator is : (see the file)
  - Only those users whose name stars by r and use the bash shell

```
awk -F: '$7~/^\/bin\/bash$/ && /^r/{printf "%-10s %s %s\n",$1,$5,$7}' /etc/passwd
```

Examples

- Print the number of records of file /etc/passwd

```
awk ’BEGIN{num=0}
        {num++}
        END{print "Number of records:",num}’/etc/passwd
```

- Print the field 1 of record 27 of file /etc/passwd

```
awk -F: -v record=27 -v field=1
        ’BEGIN{num=0; l=record; c=field}
         NR==l{print $c}
         END{print “Filed "c" of record "l” has been printed}’
/etc/passwd
```

Examples

- Print a field of a record of file /etc/passwd (use paramenters in the script)

```
awk -F: -v record=$1 -v field=$2
        ’BEGIN{num=0; l=record; c=field}
         NR==l{print $c}
         END{print “Field "c" of record "l” has been
printed”}’ /etc/passwd
```

Examples

- Print file /etc/passwd
  - Fields 1 an 5
  - Filed separator is : (see the file)
  - Only those users whose name stars by r only if field 5 is not empty

```
awk -F: ’/^r/{if ($5!="") print $1"\n-->\t"$5}’ passwd
```

Examples

- Print file /etc/passwd
  - Fields 1
  - Filed separator is : (see the file)
  - Only if the length of field 1 is greater than the given by parameter to the script
  - Print length and field

```
awk -F: -v long=$1
        ’BEGIN{l=long}
        {if (length($1)>l) print length($1),$1}’ /etc/passwd
```

Examples

- Print file /etc/passwd
  - Fields 1
  - Alphabetical order

awk -F: ’{print $1}’ /ect/passwd | sort

- Print file /etc/passwd
  - Redirect the output to a file
  - Execute the command cat to visualize the new file and more for paging the output

awk ’{print $0 >"borrar"} END{system("cat borrar | more")}’ /etc/passwd

Contents

- Main objectives
- Introduction
- Examples
- Exercises

Exercises

- Exercise 1
  - Create a script containing an awk program that from the list of active processes in the system, show only the process owner user (UID), the process identifier (PID) and the command executed (CMD).
  - Modify the above command so that it only shows the processes of the users whose name begins with "r".
- Exercise 2
  - Create a script containing an awk program which receives as parameter the name of a user and must responds for each active process: “The user *username* has an active process whith PID XXX”
  - Modified the previous script in case the user has not active processes must print “The user *username* has not active processes”

Exercises

- Exercise 3
  - Create a file with the following structure

Record one

Record two

(empty)

Record four

(empty)

Record six

Exercises

- Exercise 3
  - Create a script containing an awk program showing the content of a file according to the following steps
    - Shown all records independly they are empty or not.
    - Numbered all records
    - Use 4 characters for the number, one carácter use for whitespace and the text

```
1 Record one
2 Record two
3
4 Record four
5
6 Record six
```

Exercises

- Exercise 3
  - Create a script containing an awk program showing the content of a file according to the following steps
    - Shown only no empty records
    - Numbered only the no empty records
    - Use 4 characters for the number, one carácter use for whitespace and the text

```
1 Record one
2 Record two
4 Record four
6 Record six
```

Exercises

- Exercise 3
  - Create a script containing an awk program showing the content of a file according to the following steps
    - Shown both empty and no empty records
    - Numbered all records but independently
    - Use 4 characters for the number, one carácter use for whitespace and the text
    - At the end there is a summary

```
1 Record one
2 Record two
1
3 Record four
2
4 Record six
Empty records: 2. No empty records: 4
```

Exercises

- Exercise 4
  - On the basis of the previous exercise, create an script which receives a file and a option.
  - The file contains the records to be processed and the option one of the 3 options of exercise 3.
  - You must check if the number of parameters is correct (2), if the file exists and if the option is between 1 and 3. Provide convenient messages

Contents

- Main objectives
- Introduction
- Examples
- Exercises

The awk tool

Enrique Arias
Universidad de Castilla–La Mancha