AWK From My Perspective

Ramachandran Subramanian

Computational Sciences Club

September 21, 2016

Assumptions and disclaimer

- Assumptions: Familiarity with Linux/Unix/Bash commands and basic programming knowledge.

Assumptions and disclaimer

- Assumptions: Familiarity with Linux/Unix/Bash commands and basic programming knowledge.
- Disclaimer: This is strictly AWK from my perspective. I might not cover all different aspects of it in detail (gawk, regular expressions, arrays, debugging, output formatting, built-in and user-defined functions etc.) and what I do cover might not be the most efficient way to do a certain task. This workshop is therefore intended as an introductory workshop to give you a flavor of the convenience and versatility of AWK.

Assumptions and disclaimer

- Assumptions: Familiarity with Linux/Unix/Bash commands and basic programming knowledge.
- Disclaimer: This is strictly AWK from my perspective. I might not cover all different aspects of it in detail (gawk, regular expressions, arrays, debugging, output formatting, built-in and user-defined functions etc.) and what I do cover might not be the most efficient way to do a certain task. This workshop is therefore intended as an introductory workshop to give you a flavor of the convenience and versatility of AWK.
- Reference and further reading: https://www.gnu.org/software/gawk/manual/gawk.html

Overview of the workshop

- Basics of using AWK - syntax, BEGIN/END blocks and predefined variables
- AWK in the command line. Input from files and pipes. Few use and throw examples that demonstrate the power of AWK.
- Writing AWK scripts in a separate file.

AWK basics

- Syntax: awk '/pattern/ command' input-stream.
- Reads the entire input stream (file(s) or pipe) and looks for the given pattern and executes the command on that record.
- Usually every line is considered a record and every column is considered a field.
- Default: record separator is the newline character and the field separator is white-space sequences (spaces, tabs and newlines).
- Field values are referenced using the dollar symbol '$'. $0 represents the entire record. 'print $0' (is the same as 'print') is the default if no command is specified.

Some predefined variables - command line examples

- NR - total number of records processed until this point. Gets updated each time AWK reads a record.
- NF - total number of fields in the current record. Value of NF gets reset to 0 before processing the next record.
- FS - field separator (default: white-space sequences).
- RS - record separator (default: newline character).

Key aspects - command line examples

- BEGIN block - gets executed only once before the first record is read. All predefined variables are either 0, not defined or null depending on context.
- END block - gets executed only once after processing all the input.
- The 'next' statement - stops reading current record and proceeds to the next record.
- Assigning a value to a variable.

```
awk '
    BEGIN {
        <some commands here>
    }
    {
        <some commands here>
    }
    END {
        <some commands here>
    }' input-file
```

Pipe as input instead of a file - command line examples

- Using BEGIN block as a calculator.
- Total size of files in a directory.
- Canceling specific jobs in CCR.

Multiple input files

Order of execution of statements:

```
awk '
        BEGIN {

                <some commands here>

        }
        {

                <some commands here>

        }
        END {

                <some commands here>

        }' input-file1 input-file2 ...
```

Multiple input files - command line examples

- FNR - total number of records processed until this point in the current file. If there are multiple input files this value gets reset to 0 at the beginning of a new file. Common first column example.
- FILENAME - contains the name of the file currently being read. If there are multiple input files this value gets updated at the beginning of a new file. Examples with grep for simple cases and AWK for more complicated cases.
- Assigning multiple values to the same variable for multiple files.

AWK script files

- Create a separate file with lots of commands.
- Can be run using two methods. 1) awk -f source-file input-file1 input-file2 ... 2) Make the source file executable using chmod.
- Comments - lines starting with the '#' symbol.

AWK script files

More predefined variables:

- ARGC - total number of arguments in the command line (including the key word awk and any variables that you might have declared).
- ARGV - array containing all the command line arguments.
- ARGIND - index of command line argument.

AWK script files for processing data

- Few examples combining shell commands with AWK.

When to use AWK

"Now that youve seen some of what awk can do, you might wonder how awk could be useful for you. By using utility programs, advanced patterns, field separators, arithmetic statements, and other selection criteria, you can produce much more complex output. The awk language is very useful for producing reports from large amounts of raw data, such as summarizing information from the output of other utility programs like ls."

Source: https://www.gnu.org/software/gawk/manual/gawk.html