Awk
Instructor Handout

Author: Reuben Francis Cornel

1 What is AWK?

- AWK is a programable filter developed with the aim of using it to generate formatted reports from data.
- Althought is uses addresses like sed to perform operations it differs from sed in the way data is handled.

sed handles data as plain text, But for AWK all lines are records having fields

2 Why use AWK instead of Perl?

- Perl can be used to handle most of the operations which AWK can do. To be precise AWK is a subset of perl.
- The main reason why AWK comes in more handy than perl is because of its ease of use and clean syntax.

3 Uses Of AWK

- Apart from report generation, AWK can be used as a pseudo C interpreter.

- As a result it can used to prototype small tools.
- A less known aspect of AWK is that it can be used in AI programming, specially because of its ability to act on patterns.

4 Basic Structure Of AWK Programs

- The basic organisation of awk programs follows the pattern shown below. **Show example1**

$$/pattern/ \{actions\}$$

- The pattern could be a regular expression or a numeric comparison.
- Either the pattern or the action can be left out.
  - If the pattern is left out. The action is applied to every line
  - Can you guess what happens if the pattern is left out? The pattern is copied to the output
  - There are two special cases for patterns
    - BEGIN { }: This statement is executed at the start of processing **Show example4**
    - END { }: This statement is executed at end of processing

5 Running AWK programs

- There are 3 major ways of running AWK programs.
  - One Shot Programs: Example shown before was a one shot program
  - Long AWK programs: Here we use the -f option of awk. **Show example12 after removing the interpreter line**
  - Executable AWK programs: We set executable permission of the awk program and add the interpreter line.

6 Syntax Elements Of Awk Programs

6.1 Awk Variables

- Awk gives us three types of variables
  - User Defined Variables
  - Positional Variables
  - Special Variables

6.1.1 User Defined Variables

- This is the variable you create.
- Creating these variables require no prior declaration like C for example you want to create a variable all you have to do is assign a value to that variable name. For example `test=1` **Show Example3**
- But what happens if you refer to a variable which you have not defined previously? **Show Example3a**
- Now what would happen if we define a variable which has a name of a inbuilt function **Show Example3b**

6.1.2 Positional Variables

- A positional variable is one which is prefixed with the $ sign. This represents a particular field in the record. show example 1 and explain
- Whitespaces are used as delimiters for fields
- There is a special positional variable $0. This represents the entire line read in by awk.
- When you do a `print $0` you print the whole line. But is there another way to print the same? just `print` does the job

6.2 Operators

- Awk provides us with three types of operators
  - Math Operators
  - Relational Operators
  - Regular Expression Operators
- The table below is the list of mathematical operators

Operator	Meaning
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`%`	Modulo
`++`	Auto increment
`--`	Auto decrement
`+=`	Add result to variable
`-=`	Subtract result from variable
`*=`	Multiply variable by result
`/=`	Divide variable by result
`%=`	Apply modulo to variable

- **Show example13**
- The table below is a list of relational operators

Operator	Meaning
`==`	Is equal to
`!=`	Is not equal to
`>`	Greater than
`>=`	Greater than
`<`	Lesser than
`<=`	Lesser than

- Finally we have operators for regular expressions

Operator	Meaning
~	Pattern matches String
!~	Pattern Does not match

- **Mention ! is used for negation. Show Example14**

6.3 Awk Keywords

- Most awk commands have been taken over from C. Therefore most of you would be familiar with most of the basic syntax. Nonetheless the table below is a reference to the syntax.

Syntax	Meaning
if(conditional) statement [else statment]	If Else Construct
while (conditional) statement	While Construct
for (expression ; conditional ; expression) statement	Typical For Construct
for (variable in array) statement	This is a variation of the above for, similar to the shell for command
break	Break a loop
continue	Get on with the next iteration
{ [statement] ...}	Blocks
variable=expression	Assignment
print [expression-list]	Standard Print
printf format [, expression-list]	formatted output
exit	Exits the interpreter

7 Special Variables

- Awk provides us with special variables using which we can change certian properties of awk, such as the delimiter that seperates fields or records.

7.1 FS – Field Seperator

- This variable represents the string which seperates fields in a record. By default this is a single space.
- Since not all records use spaces for delimiters changing this field lets you change the delimiter. **show example5**

7.2 RS – Record Seperator

- This by default is a newline character.
- Sometime we have records which span across multiple lines. So in a we have to change the string which acts like a delimiter. The RS varible helps us do this. **show example6**

7.3 NF – Number of Fields

- This is variable represents the number of fields. This comes in helpful when you have to change operation of the program based on the number of fields avaliable.

show example7

7.4 NR – Number of Records

- This variable represents the number of records that have been processed.

show example8

7.5 OFS – Output Field Seperator & ORS – Output Record Seperator

- Sometimes a program has to provide a input to a filter and that filter which uses a different output field and record seperator. In suc a case you can use awk to act like an adaptor.
- OFS - is the character which is printed as the field delimiter when you print a number of fields
- ORS - is the character which is printed as the record delimiter.

show example9

Emphasise that print $1 $2 is different from print $1, $2. And OFS works only with the later.

8 Associative Arrays

- Instead of the regular arrays, awk provides us with associative arrays.
- Associative arrays unlike conventional arrays can use anything as subscripts.
- This comes in helpful when you have to perform count operations to generate reports.

show example10

9 Formatted Output using `printf`

- Awk provides us with printf so that our output can be formatted to fit, certain specifications.
- Below shown is a list of format specifiers

Format Specifier	Meaning
%c	ASCII Character
%d	Decimal integer
%e	Floating Point number (engineering format)
%f	Floating Point number (fixed point format)
%o	Octal
%s	String
%x	Hexadecimal
%	Literal %

- The basic format specifier syntax is as shown below

```
%[+-]?[0-9]*.?[0-9]*[specifier]
```

- Below are some example how to use the specifiers

Statement	Meaning
x = "Baryshnikov"	
printf("[%16s]",x)	[Baryshnikov]
printf("[%-16s]",x)	[Baryshnikov]
printf("[%.3s]",x)	[Bar]
printf("[%16.3s]",x)	[Bar]
printf("[%-16.3s]",x)	[Bar]
printf("[%016s]",x)	[00000Baryshnikov]
printf("[%-016s]",x)	[Baryshnikov]
x = 312	
printf("[%8d]",x)	[312]
printf("[%-8d]",x)	[312]
printf("[%08d]",x)	[00000312]
printf("[%-08d]",x)	[312]
x = 251.673209	
printf("[%16f]",x)	[251.67309]
printf("[%-16f]",x)	[251.67309]
printf("[%.3f]",x)	[251.673]
printf("[%16.3f]",x)	[251.673]
printf("[%016.3f]",x)	[00000000251.673]

- printf and print also allow you to write the output into a file using the > and the >> symbol. **show example11**

10 String Functions

- It is important to remember that strings in awk are handled as a full data type and not like a array of characters in C.
- Below is a list of commonly used string functions in awk **show example12**

Function	Meaning
index(string,search)	Returns the location of search string in the given string else 0
length(string)	Prints the lenght of the string
split(string,array,separator)	Splits a given string into an array based on supplied seperator
substr(string,position,max)	extracts substring from the given string starting at mentioned position and for a given lenght of characters
sub(regex,replacement,string)	substitutes one regular expression instance in string with replacement
gsub(regex,replacement,string)	substitutes all regular expression instances in string with replacement

11 Command Line Options

- Awk allows you to perform command line initialization of variables, using arguments to the awk command. Some of the most common arguments are given below

Option	Meaning
`-F`	Lets you can specify the feild seperator here
`-f`	File from which the program is to be executed
`-v` *`var=val`*	Lets you can initialise a specific variable from the command line