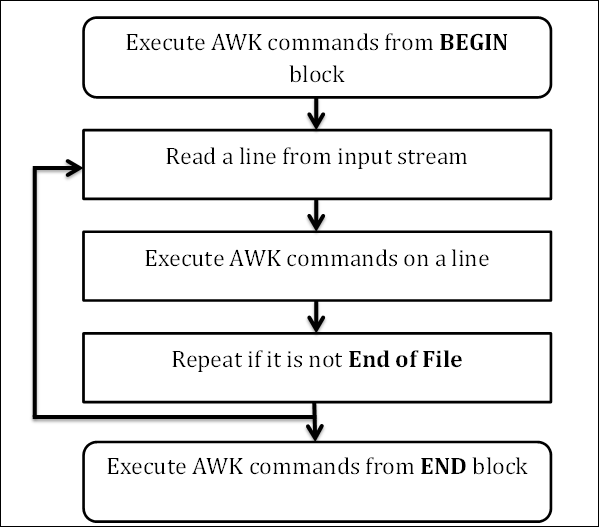
<https://www.tutorialspoint.com/awk/awk_basic_syntax.htm>

# Awk Workflow



BEGIN {awk-commands}

/pattern/ {awk-commands}

END {awk-commands}

awk 'BEGIN{printf "f1\tf2\tf3\textra\_info\n"}{print}' f.csv

Awk ‘{print}’ f.csv

awk [options] -f awkscript.awk file.csv #read

# Create an awk file script and then run it with -f

Echo “{print}” >> command.awk

Awk –f command.awk f.csv

# Run awk script but with custom delimiters

awk -F'[-,]' -f command.awk temp.csv

# Assign value to a variable and use it

awk -v name=Jerry 'BEGIN{printf "Name = %s\n", name}'

# Print only certain columns i.e. columns 3 and 4

Awk ‘{print $3 “,” $4}’ f.csv

# Print by custom delimiters. Good for if there’s multiple columns with different delimiters (print columns 1,2 and 3 separated by comma delimiters) with FS. The default value for FS is space

Awk ‘{ FS = “,|”} {print $1 “,” $2 “,” $3}’ f.csv

# Read file with multiple delimiters and print with different delimiters

awk -F'[-,]' '{print $1 "--" $2 "--" $3}' temp.csv

# Print 3rd and 4th field when a pattern match succeeds

awk '/pattern/{ print $0 }' file

awk '/a/ {print $3 "\t" $4}' marks.txt

# Count and print the number of lines for which a pattern match succeeded

$ awk '/a/{++cnt} END {print "Count = ", cnt}' f.csv

# Where /a/ is the pattern. The above prints the number of lines

# Print lines with more than 18 characters

awk 'length($0) > 18' marks.txt

# Set awk to become case-insensitive, then search for a pattern

awk 'BEGIN{IGNORECASE=1} /no internet service/' f.csv

# Some regex patterns

Cat f.csv | awk ‘pattern.’

/./ >> any single character except end of line character

Cat f.csv | awk ‘/^2/’ >> All lines that start with 2

‘/n$/’ >> end of lines that end with the letter n using $ sign

‘/[CT]all/’ >> match character set e.g. Call and Tall

‘/[^CT]all/’ >> excludes match character set e.g. Call and Tall but will print Ball

‘/Call|Ball/’ >> where line matches Call or Ball

? >> optional character e.g. for cases like color and colour, u can be optional

‘/cat\*/’ >> asterisk means 0 or more occurrences e.g. cat, catergory

+ >> plus means 1 or more occurrence

‘/Apple (Juice|Cake)/’ >> Apple juice or apple cake. Brackets used for grouping and | is used for OR

# If statement

If (condition) action

Awk ‘BEGIN {num = 10; if (num % 2 == 0) printf “%d is even number”, num}’

# If else

If (condition) action elseif (condition) action2 else action3

awk 'BEGIN {

a = 30;

if (a==10)

print "a = 10";

else if (a == 20)

print "a = 20";

else if (a == 30)

print "a = 30";

}'

# For loop

For (initialization; condition; increment/decrement)

awk 'BEGIN { for (i = 1; i <= 5; ++i) print i }'

# While loop

While (condition) action

Awk ‘BEGIN {i=1; while(i < 6) {print i; ++i }}’

# Do while loop

awk 'BEGIN {i = 1; do { print i; ++i } while (i < 6) }'

# Break statement

awk 'BEGIN {

sum = 0; for (i = 0; i < 20; ++i) {

sum += i; if (sum > 50) **break**; else print "Sum =", sum

}

}'

# Continue statement

awk 'BEGIN {

for (i = 1; i <= 20; ++i) {

if (i % 2 == 0) print i ; else **continue**

}

}'

# Exit statement

awk 'BEGIN {

sum = 0; for (i = 0; i < 20; ++i) {

sum += i; if (sum > 50) exit(10); else print "Sum =", sum

}

}'

Exit(n) is the exit status code for the AWK process. If none give, default is zero

# User defined functions example

# Returns minimum number

function find\_min(num1, num2){

if (num1 < num2)

return num1

return num2

}

# Returns maximum number

function find\_max(num1, num2){

if (num1 > num2)

return num1

return num2

}

# Main function

function main(num1, num2){

# Find minimum number

result = find\_min(10, 20)

print "Minimum =", result

# Find maximum number

result = find\_max(10, 20)

print "Maximum =", result

}

# Script execution starts here

BEGIN {

main(10, 20)

}

# Output to file >

Print f.csv > outputfile.csv

awk 'BEGIN { print "Hello, World !!!" > "/tmp/message.txt" }'

# Append to file >>

awk 'BEGIN { print "Hello, World !!!" >> "/tmp/message.txt" }'

# Convert to uppercase

cat f.csv | awk '{print | "tr [a-z] [A-Z]"}'

# Convert to uppercase and output to a file >

cat f.csv | awk '{print | "tr [a-z] [A-Z]"}' > temp.csv

# convert to uppercase and append to a file >>

cat f.csv | awk '{print | "tr [a-z] [A-Z]"}' >> temp.csv

# |& - 2 way communication

# Print out a file

Awk ‘{print}’ f.csv

# Search a field for a pattern

awk -F, '{if($2 == "Male") print $0}' f.csv

# AND condition using &&

awk -F, '{if($2 == "Male" && $9 == "Fiber optic") print $0}' f.csv

# OR logic using || note the bracketing!

NOT logic using !(CONDITION)

awk -F, '{if ( !($2 == "Male" && $9 == "Fiber optic")) print $0}' f.csv

# You can tell awk how fields are separated using the –F option in the command line

Awk –F “,” ‘print $1’ file.csv

# Print range of columns given a list of delimiters

awk -F'[-,]' -v f=1 -v t=3 '{for(i=f;i<=t;i++) printf("%s%s",$i,(i==t)?"\n":OFS)}' temp.csv

# Exclude columns i.e exclude column 3

awk '{$3=""; print $0}' file.csv

# Exclude columns 1 and 2 but print the rest

awk '{$1=$2=""; print $0}' file.csv

# Print or exclude a range of columns

awk -v f=2 -v t=4 '{for(i=f;i<=t;i++) printf("%s%s",$i,(i==t)?"\n":OFS)}' FILE

# Exclude a column range from the second till the fourth and print the rest

$ awk -v f=2 -v t=4 '{for(i=1;i<=NF;i++)if(i>=f&&i<=t)continue;else printf("%s%s",$i,(i!=NF)?OFS:ORS)}' FILE