Awk Simple If Statement

**Single Action:** Simple If statement is used to check the conditions, if the condition returns true, it performs its corresponding action(s).

Syntax:

if (conditional-expression)

action

* if is a keyword
* conditional-expression – expression to check conditions
* action – any awk statement to perform action.

**Multiple Action:** If the conditional expression returns true, then action will be performed. If more than one action needs to be performed, the actions should be enclosed in curly braces, separating them into a new line or semicolon as shown below.

Syntax:

if (conditional-expression)

{

action1;

action2;

}

If the condition is true, all the actions enclosed in braces will be performed in the given order. After all the actions are performed it continues to execute the next statements.

Awk If Else Statement

In the above simple awk If statement, there is no set of actions in case if the condition is false. In the awk If Else statement you can give the list of action to perform if the condition is false. If the condition returns true action1 will be performed, if the condition is false action 2 will be performed.

Syntax:

if (conditional-expression)

action1

else

action2

Awk also has conditional operator i.e **ternary operator** ( ?: ) whose feature is similar to the awk If Else Statement. If the conditional-expression is true, action1 will be performed and if the conditional-expression is false action2 will be performed.

Syntax:

conditional-expression ? action1 : action2 ;

Awk If Else If ladder

if(conditional-expression1)

action1;

else if(conditional-expression2)

action2;

else if(conditional-expression3)

action3;

.

.

else

action n;

* If the conditional-expression1 is true then action1 will be performed.
* If the conditional-expression1 is false then conditional-expression2 will be checked, if its true, action2 will be performed and goes on like this. Last else part will be performed if none of the conditional-expression is true.

Now let us create the sample input file which has the student marks.

$cat student-marks

Jones 2143 78 84 77

Gondrol 2321 56 58 45

RinRao 2122 38 37

Edwin 2537 87 97 95

Dayan 2415 30 47

1. Awk If Example: Check all the marks are exist

$ awk '{

if ($3 =="" || $4 == "" || $5 == "")

print "Some score for the student",$1,"is missing";'

}' student-marks

Some score for the student RinRao is missing

Some score for the student Dayan is missing

$3, $4 and $5 are test scores of the student. If test score is equal to empty, it throws the message. || operator is to check any one of marks is not exist, it should alert.

2. Awk If Else Example: Generate Pass/Fail Report based on Student marks in each subject

$ awk '{

if ($3 >=35 && $4 >= 35 && $5 >= 35)

print $0,"=>","Pass";

else

print $0,"=>","Fail";

}' student-marks

Jones 2143 78 84 77 => Pass

Gondrol 2321 56 58 45 => Pass

RinRao 2122 38 37 => Fail

Edwin 2537 87 97 95 => Pass

Dayan 2415 30 47 => Fail

The condition for Pass is all the test score mark should be greater than or equal to 35. So all the test scores are checked if greater than 35, then it prints the whole line and string “Pass”, else i.e even if any one of the test score doesn’t meet the condition, it prints the whole line and prints the string “Fail”.

3. Awk If Else If Example: Find the average and grade for every student

$ cat grade.awk

{

total=$3+$4+$5;

avg=total/3;

if ( avg >= 90 ) grade="A";

else if ( avg >= 80) grade ="B";

else if (avg >= 70) grade ="C";

else grade="D";

print $0,"=>",grade;

}

$ awk -f grade.awk student-marks

Jones 2143 78 84 77 => C

Gondrol 2321 56 58 45 => D

RinRao 2122 38 37 => D

Edwin 2537 87 97 95 => A

Dayan 2415 30 47 => D

In the above awk script, the variable called ‘avg’ has the average of the three test scores. If the average is greater than or equal to 90, then grade is A, or if the average is greater than or equal to 80 then grade is B, if the average is greater than or equal to 70, then the grade is C. Or else the grade is D.

4. Awk Ternary ( ?: ) Example: Concatenate every 3 lines of input with a comma.

$ awk 'ORS=NR%3?",":"\n"' student-marks

Jones 2143 78 84 77,Gondrol 2321 56 58 45,RinRao 2122 38 37

Edwin 2537 87 97 95,Dayan 2415 30 47,

We discussed about [awk ORS built-in variable](http://www.thegeekstuff.com/2010/01/8-powerful-awk-built-in-variables-fs-ofs-rs-ors-nr-nf-filename-fnr/) earlier. This variable gets appended after every line that gets output. In this example, it gets changed on every 3rd line from a comma to a newline. For lines 1, 2 it’s a comma, for line 3 it’s a newline, for lines 4, 5 it’s a comma, for line 6 a newline, etc.

Recommended Reading