A guide to intermediate awk scripting

Learn how to structure commands into executable scripts.

11 Nov 2019 | Seth Kenlon (Red Hat) (/users/seth) | Dave Morriss (/users/dave-morriss) | Robert Young (/users/ryoung29) | 76 | 2 comments

This article explores awk's capabilities, which are easier to use now that you know how to structure your command into an executable script.

Logical operators and conditionals

You can use the logical operators **and** (written **&&**) and **or** (written ||) to add specificity to your conditionals.

For example, to select and print only records with the string "purple" in the second column *and* an amount less than five in the third column:

```
$2 == "purple" && $3 < 5 {print $1}
```

If a record has "purple" in column two but a value greater than or equal to 5 in column three, then it is *not* selected. Similarly, if a record matches column three's requirement but lacks "purple" in column two, it is also *not* selected.

Next command

Say you want to select every record in your file where the amount is greater than or equal to eight and print a matching record with two asterisks (**). You also want to flag every record with a value between five (inclusive) and eight with only one asterisk (*). There are a few ways to do this, and one way is to use the **next** command to instruct awk that after it takes an action, it should stop scanning and proceed to the *next* record.

Here's an example:

```
$3 >= 8 {
  printf "%s\t%s\n", $0, "**";
  next;
}

$3 >= 5 {
  printf "%s\t%s\n", $0, "*";
  next;
}

$3 < 5 {
  print $0;
}
```

BEGIN command

The **BEGIN** command lets you print and set variables before awk starts scanning a text file. For instance, you can set the input and output field separators inside your awk script by defining them in a **BEGIN** statement. This example adapts the simple script from the previous article for a file with fields delimited by commas instead of whitespace:

```
#!/usr/bin/awk -f
#
# Print each record EXCEPT
# IF the first record contains "raspberry",
# THEN replace "red" with "pi"

BEGIN {
        FS=",";
}

$1 == "raspberry" {
```

END command

The **END** command, like **BEGIN**, allows you to perform actions in awk after it completes its scan through the text file you are processing. If you want to print cumulative results of some value in all records, you can do that only after all records have been scanned and processed.

The **BEGIN** and **END** commands run only once each. All rules between them run zero or more times on *each record*. In other words, most of your awk script is a loop that is executed at every new line of the text file you're processing, with the exception of the **BEGIN** and **END** rules, which run before and after the loop.

Here is an example that wouldn't be possible without the **END** command. This script accepts values from the output of the **df** Unix command and increments two custom variables (**used** and **available**) with each new record.

```
$1 != "tempfs" {
    used += $3;
    available += $4;
}

END {
    printf "%d GiB used\n%d GiB available\n", used/2^20, available/2^
}
```

Save the script as **total.awk** and try it:

```
df -l | awk -f total.awk
```

their type, and you add values to them at will. At the end of the loop, the script adds the records in the respective columns together and prints the totals.

Math

As you can probably tell from all the logical operators and casual calculations so far, awk does math quite naturally. This arguably makes it a very useful calculator for your terminal. Instead of struggling to remember the rather unusual syntax of **bc**, you can just use awk along with its special **BEGIN** function to avoid the requirement of a file argument:

```
$ awk 'BEGIN { print 2*21 }'
42
$ awk 'BEGIN {print 8*log(4) }'
11.0904
```

Admittedly, that's still a lot of typing for simple (and not so simple) math, but it wouldn't take much effort to write a frontend, which is an exercise for you to explore.

This article is adapted from an episode of [Hacker Public Radio (http://hackerpublicradio.org/eps.php?id=2129)](http://hackerpublicradio.org/eps.php?id=2129), a community technology podcast.

Getting started with awk, a powerful text-parsing tool (/article/19/10/intro-awk)

Let's jump in and start using it.

Seth Kenlon (Red Hat) (/users/seth) | Dave Morriss (/users/dave-morriss) | Robert Young (/users/ryoung29)

 (/article/19/11/fields-records-variables-awk)

Fields, records, and variables in awk (/article/19/11/fields-records-variables-awk)

In the second article in this intro to awk series, learn about fields, records, and some powerful awk variables.

Seth Kenlon (Red Hat) (/users/seth)

 (/article/18/7/cheat-sheet-awk)

Extracting and displaying data with awk (/article/18/7/cheat-sheet-awk)

Get our awk cheat sheet.

Jim Hall (Correspondent) (/users/jim-hall)

(/users/seth)

About the author

Seth Kenlon - Seth Kenlon is an independent multimedia artist, free culture advocate, and UNIX geek. He has worked in the film (http://www.imdb.com/name/nm1244992) and computing (http://people.redhat.com/skenlon) industry, often at the same time. He is one of the maintainers of the Slackware-based multimedia production project, http://slackermedia.info (http://slackermedia.info)

• More about me (/users/seth)

(/users/ryoung29)

About the author

Robert Young -

• More about me (/users/ryoung29)

(/users/dave-morriss)

About the author

Dave Morriss - Dave Morriss is a retired IT Manager based in Edinburgh, Scotland. He worked in the UK higher education sector providing IT services to students and staff. He has a BSc degree in Zoology and spent some time working towards a PhD in Animal Behaviour. However, the prospect of working in IT proved to be more attractive and he moved to Lancaster University (in Lancashire, UK) in the mid 1970's where he worked as a Systems Programmer for five years. At Lancaster he worked with the University's ICL...

• More about me (/users/dave-morriss)

Recommended reading

Program hardware from the Linux command line (/article/20/9/hardware-command-line?utm_campaign=intrel)

An introduction to using tcpdump at the Linux command line (/article/18/10/introduction-tcpdump?utm_campaign=intrel)

Shrink PDF size with this command line trick (/article/20/8/reduce-pdf?utm_campaign=intr

Use a Linux terminal on your Android phone (/article/20/8/termux?utm_campaign=intrel)

How I customize my Mac terminal with open source tools (/article/20/8/iterm2-zsh?utm_campaign=intrel)

How I balance features and performance in my Linux terminal (/article/20/7/performance-linux-

2 Comments

Mark_Stewart_1337 on 11 Nov 2019 0

Seth Kenlon (/users/seth) on 25 Nov 2019 0

Thanks, Mark_Steward_1337, you're right. I've added your clarification.