awk—An Advanced Filter

The **awk** command made a late entry into the UNIX system in 1977 to augment the tool kit with suitable report formatting capabilities. Named after its authors, Aho, Weinberger and Kernighan, **awk**, until the advent of **perl**, was the most powerful utility for text manipulation. Like **sed**, it combines features of several filters, though its report writing capability is the most useful. **awk** appears as **gawk** (GNU awk) in Linux.

awk doesn't belong to the do-one-thing-well family of UNIX commands. In fact, it can do several things—and some of them quite well. Unlike other filters, it operates at the *field* level and can easily access, transform and format individual fields in a line. It also accepts extended regular expressions (EREs) for pattern matching, has C-type programming constructs, variables and several built-in functions. Knowing **awk** will help you understand **perl**, which uses most of the **awk** constructs, sometimes in identical manner.

18.1 SIMPLE awk FILTERING

awk is a little awkward to use at first, but if you feel comfortable with **find** and **sed**, then you'll find a friend in **awk**. Even though it is a filter, **awk** resembles **find** in its syntax:

```
awk options 'selection_criteria {action}' file(s)
```

The *selection_criteria* (a form of addressing) filters input and selects lines for the *action* component to act upon. This component is enclosed within curly braces. The *selection_criteria* and *action* constitute an **awk** program that is surrounded by a set of single quotes. These programs are often one line long though they can span several lines as well.

The selection criteria in **awk** have wider scope than in **sed**. Like there, they can be patterns like `/negroponte/` or line addresses that use **awk**'s built-in variable, `NR`. Further, they can also be conditional expressions using the `&&` and `||` operators as used in the shell. You can select lines practically on any condition.

A typically complete **awk** command specifies the selection criteria and action. The following command selects the directors from `emp.lst`:

```
$ awk '/director/ { print }' emp.lst
9876|jai sharma      |director |production|12/03/50|7000
2365|barun sengupta  |director |personnel |11/05/47|7800
1006|chanchal singhvi |director |sales     |03/09/38|6700
6521|lalit chowdury  |director |marketing |26/09/45|8200
```

The *selection_criteria* section (`/director/`) selects lines that are processed in the *action* section (**`{ print }`**). If *selection_criteria* is missing, then *action* applies to all lines. If *action* is missing, the entire line is printed. Either of the two (but not both) is optional, but they must be enclosed within a pair of single (not double) quotes.

The **print** statement, when used without any field specifiers, prints the entire line. Moreover, since printing is the default action of **awk**, all the following three forms could be considered equivalent:

```
awk '/director/' emp.lst                  Printing is the default action
awk '/director/{ print }' emp.lst         Whitespace permitted
awk '/director/ { print $0}' emp.lst      $0 is the complete line
```

For pattern matching, **awk** uses regular expressions in **sed**-style:

```
$ awk -F"|" '/sa[kx]s*ena/' emp.lst
3212|shyam saksena   |d.g.m.   |accounts  |12/12/55|6000
2345|j.b. saxena     |g.m.     |marketing |12/03/45|8000
```

The regular expressions used by **awk** belong to the basic BRE (but not the IRE and TRE) and ERE variety that's used by **grep -E** *(13.3)* or **egrep**. This means that you can also use multiple patterns by delimiting each pattern with a `|`.

Note: An **awk** program must have either the selection criteria or the action, or both, but within single quotes. Double quotes will create problems unless used judiciously.

18.2 SPLITTING A LINE INTO FIELDS

awk uses the special parameter, $0, to indicate the entire line. It also identifies fields by $1, $2, $3. Since these parameters also have a special meaning to the shell, single-quoting an **awk** program protects them from interpretation by the shell.

Unlike the other UNIX filters which operate on fields, **awk** uses a contiguous sequence of spaces and tabs as a *single* delimiter. But the sample database *(12.1)* uses the |, so we must use the -F option to specify it in our programs. You can use **awk** to print the name, designation, department and salary of all the sales people:

```
$ awk -F"|" '/sales/ { print $2,$3,$4,$6 }' emp.lst
a.k. shukla       g.m.        sales      6000
chanchal singhvi  director    sales      6700
s.n. dasgupta     manager     sales      5600
anil aggarwal     manager     sales      5000
```

Notice that a , (comma) has been used to delimit the field specifications. This ensures that each field is separated from the other by a space. If you don't put the comma, the fields will be glued together.

So far, the programs have produced readable output, but that is because the file emp.lst contains fixed-length lines. Henceforth, the input for most **awk** programs used in this chapter will come from the file empn.lst which we created with **sed** in Section 13.10. This file is similar to emp.ls except that the lines are of variable length. A few lines of the file show the total absence of spaces around the |:

```
$ head -n 2 empn.lst
3212|shyam saksena|d.g.m.|accounts|12/12/55|6000|6213
6213|karuna ganguly|g.m.|accounts|05/06/62|6300|6213
```

With this file as input, we'll use **awk** with a line address (single or double) to select lines. If you want to select lines 3 to 6, all you have to do is use the built-in variable NR to specify the line numbers:

```
$ awk -F"|" 'NR == 3, NR == 6 { print NR, $2,$3,$6 }' empn.lst
3 n.k. gupta chairman 5400
4 v.k. agrawal g.m. 9000
5 j.b. saxena g.m. 8000
6 sumit chakrobarty d.g.m. 6000
```

This is **awk**'s way of implementing the **sed** instruction **3,6p**. The statement NR == 3 is really a condition that is being tested, rather than an assignment; this should appear obvious to C programmers. NR is one of those built-in variables used in **awk** programs, and == is one of the many operators employed in comparison tests.

Note: awk is the only filter that uses whitespace as the default delimiter instead of a single space or tab.