INDEPTH

GNU Awk 4.0: Teaching an Old Bird Some New Tricks

What's new and nifty in gawk 4.0, with a little history and background along the way. ARNOLD ROBBINS

GNU Awk (gawk) is one of those programs that has been available "since forever", which many people never think about. But, it's a standard and important part of just about every GNU/Linux distribution. In fact, it has been available since even before GCC!

During the past year and a half or so, gawk has undergone a quiet revolution, culminating in the release of gawk 4.0. Although not yet released at the time of this writing, work is in progress and moving forward. By the time you read this article, gawk 4.0 will be a fact and not just a promising code base in the Git repository.

A Little History

The awk language was developed by Al Aho, Peter J. Weinberger and Brian Kernighan, then at Bell Labs (hence the name A.W.K.). It first was released in 1978 with V7 UNIX. It offered the pattern-action programming paradigm, powerful regular expression matching, associative arrays, conventional operators and control structures, and a modest array of built-in numeric and string functions. It was only minimally documented. (So minimally, in fact, that I remember being terribly confused after reading the short paper on awk, and deciding to avoid it!) Nonetheless, the UNIX world accepted it and used it; true UNIX wizards were comfortable writing even large scripts in it.

Circa 1985, the authors started beefing up the language, adding user-defined functions, C-compatible operator precedence, more built-in functions, dynamic regular expressions and a few other minor features. More important though, they then proceeded to write a book about the new version

of awk (*The AWK Programming Language*), which was published in late 1987. This version became available to the world with the UNIX System V Release 3.2.

I bought the book, figuring that now was my chance to learn awk. It was (and remains) a great book. Having an interest in programming languages and an interest in contributing to the world at large, I decided to see whether the GNU project had a version of awk. Indeed, it did, but it implemented only old awk (and poorly, at that). Being single at the time, I decided to get involved and see if I could work to make gawk compatible with new awk. (And, thus, the course of history changed, forever.)

As early as 1988, the GNU developers were corresponding with Brian Kernighan and other awk implementers to make sure that the awk semantics were consistent across implementations. System V Release 4, in 1989, brought a few new features for new awk (the `-v` option, the `ENVIRON` array, the `tolower()` and `toupper()` built-in functions) and the first POSIX standard (circa 1992) introduced the CONVFMT variable.

Starting in December 1993, Brian Kernighan was able to release the code to new awk; it continues to be available (see Resources) and sees minor bug fixes from time to time.

GNU Awk

GNU Awk was first written around 1986 by Jay Rubin and Paul Finlason, with some help from Richard Stallman. It barely implemented the original awk language, was buggy and not particularly fast. It worked by building a parse tree representation of the program and then recursively evaluating the parse tree for each input record.

When I got involved in late 1987, David Trueman already had volunteered to upgrade it to new awk, and I joined the effort, contributing code fixes and doing serious work on the documentation. We worked together until around 1994, when I became the sole maintainer.

Along the way, gawk acquired full compliance with new awk, including POSIX, and it improved in code quality, speed and new features. Throughout the course of more than 20 years though, the basic design remained the same: build the parse tree and recursively evaluate it for each input record.

In 2003, out of the blue, a gentleman named John Haque contacted me. He had rewritten the gawk internals to use a byte-code interpreter and provided an awk-level debugger for awk programs. This was a startling innovation. I worked with him to get his version to the point where it was stable and passed the test suite, but I

The most significant new feature is that the gawk internals have been completely redone.

did not integrate his changes, because they were major, and I wanted to understand them better.

Bad move: John disappeared in early 2004, and the code languished, unused. Finally, in fall 2009, I got a volunteer (Stephen Davies) to start bringing the last version of the byte-code gawk that I had into the present. He and I had things working, pretty much, and I even announced a test release to the world.

Again, out of the blue, John resurfaced in early 2010 and joined the effort to make the byte-code gawk viable. This moved things into high gear, and we made a lot of progress. As I write this, the byte-code version has been merged with my "new features" branch of the code. This is the basis for gawk 4.0.

If you don't yet have gawk 4.0, see Resources for information on where to download the source and how to build it; building from source is very easy.

New Stuff in gawk 4.0

With all the background out of the way, let's look at the cool stuff. Due to space considerations, this is just a quick tour; see the documentation (listed in Resources) for details.

New Internals

The most significant new feature is that the gawk internals have been completely redone. The parser now builds a linked list of "instructions". Each instruction contains a code indicating what it is and a few members with needed information, such as the next instruction and which instruction to jump to if a jump is needed. This list then is interpreted for each record by a big switch statement running inside a for loop that traverses the list. Data for operations are pushed and popped off a runtime stack.

This implementation performs no worse than the original recursive evaluator, and in many cases, it performs better. But what's really cool is that John added an awk-level *debugger*!

Since 1978 when awk was first introduced into the world, the only debugging tool was the `print` statement. Now, gawk has a full debugger, with breakpoints, watchpoints, stepping by statement or instruction, the ability to step into and out of functions, and

many other features.

The debugger is a separately compiled program named dgawk. It is a line-oriented debugger modeled after GDB (the GNU Debugger). If you're familiar with GDB, it will be very easy to learn the gawk debugger. In addition, the debugger is fully documented in the gawk.texi file in the gawk distribution.

New Language-Level Features

At the language level, there are several new features.

1. gawk now provides a built-in file inclusion mechanism. Lines that begin with `@include` and have a filename in double quotes cause gawk to include that file, using the same path searching mechanism as the `-f` option. Nested includes are supported, and gawk will not include the same file twice. This effectively obsoletes the `igawk` script that has come with gawk for many years.

2. New patterns named `BEGINFILE` and `ENDFILE` provide "hooks" into gawk's automatic "read a record and process it" loop. The action for `BEGINFILE` is called before the first record is read from each input file. Normally, when a file cannot be opened, gawk exits with a fatal error (such as if you provide a directory on the command line). When a program has a `BEGINFILE` pattern, instead, gawk sets the `ERRNO` variable to a string indicating the problem, so that you can tell if the file is problematic. If it is, use the `nextfile` keyword to just skip it. `ENDFILE` actions let you do easy per-file cleanup actions.

3. You now can call a function indirectly. By setting a variable to the name of the function you wish to call and using special syntax, gawk will "indirect" through the variable and call the desired function:

```
function f1(a, b) { .... }
function f2(c, d) { .... }

{ fun = "f1";    @fun(2, 3)    # calls f1()
  fun = "f2";    @fun(4, 5) }  # calls f2()
```

4. gawk now sports true multidimensional arrays! Regular awk simulates multidimensional arrays (a[x, y]) using string concatenation of the index values. gawk now provides multidimensional arrays (a[x][y]) but does not require that arrays be rectangular (as in C or other compiled languages). Code like this is valid:

```
a[1] = 1
a[2][1] = 21
```

It is up to the programmer to track the type stored at any given index: scalar or array. Subarrays can be passed

to functions, as long as the function knows what to expect.

5. The `switch/case` statement is enabled by default. gawk has had `switch/case` for a long time, but it had to be enabled at build time, and the default was not to do so; now it's enabled automatically.

6. gawk now supports defining fields based on field content, instead of based on the separators between fields. A new variable, `FPAT`, is used. When you assign a string containing a regular expression to `FPAT`, gawk begins splitting fields such that each field is the text that matched `FPAT`. (Normal field splitting is based on the text in between fields matching the regular expression in `FS`.) This is useful for many kinds of data where `FS`-based matching just doesn't work.

The new `patsplit()` built-in function provides access to this functionality for strings besides the input record. It is the analogue of awk's regular `split()` function. Additionally, `patsplit()` lets you capture the text of the separators between fields.

7. Standard awk provides only one-way pipelines, either to or from another process. gawk provides a notation for opening a two-way pipeline to a co-process. gawk uses the same notation with special, internally recognized file-names, to provide TCP/IP communication over sockets. This feature has been available for a long time.

gawk 4.0 enhances the networking by providing explicit filenames to indicate IPv4 or IPv6 connections. Filenames are of the form /inet4/*protocol*/*local-portt*/*remote-host*/*remote-port* or /inet6/*protocol*/*local-port*/*remote-host*/*remote-port*. Plain /inet/*protocol*/*local-port*/*remote-host*/*remote-port* is what gawk supplied up to now and continues to be supported: it now means "use the system default". Most likely, this will continue to be IPv4 for many years.

8. gawk now provides a short (single-letter) option for every long option that it has. This finally makes it possible to use almost every feature from a !# script. It does somewhat bloat the manual page. (gawk has too many options, but that's a different problem; nonetheless, I did remove a few redundant long options.)

9. Interval expressions now are available by default. An interval expression is an enhanced regular expression syntax, such as `(foo|bar){2,4}`, which matches anywhere from two to four occurrences of either foo or bar. The part between the curly braces is the interval expression. POSIX added them to awk many years ago for compatibility with `egrep`'s regular expressions. But most awks didn't implement them. For historical compatibility, gawk's default was to

disable them, unless running in POSIX mode. Today, compatibility with POSIX has gained enough importance for enough users that interval expressions now are available by default.

10. Finally, for this release, the code has been reviewed and cleaned up. gawk now requires a full C 89 environment to compile and run. It will not work with K&R compilers or if __STDC__ is defined but less than 1. The code for many obsolete and unsupported systems has been removed completely. This slightly decreases the size of the distribution, but mainly it reduces useless clutter in the source. The documentation also has been reviewed and cleaned up.

Source Code Management

For many years, I was the only one with access to gawk source while it was being worked on. Circa 2006, I made both the stable and development versions available via CVS from savannah.gnu.org. This was a good move; it gave the user community access to all my bug fixes and to my development code base.

In late 2010, I moved to git. I am

expecting greater productivity from using git and better ease of use for the user community. And, it's nice to be using 21st-century tools.

Future Work

Some further interesting development remains to be done.

1. The XMLGawk Project (see Resources) is a fork of gawk based on 3.1.6 that provides better facilities for loading dynamic extensions and several very interesting extensions to go with those features. These should be merged into the main gawk code base and distribution, respectively.

2. Although gawk has had the ability to load extensions dynamically for many years, the API has not been stable or easy to use. I have designed an API for C functions that can be called from an awk program that is considerably better, but I have not implemented it yet. This should be done.

3. Currently, the gawk distribution builds three separate executables: regular gawk, pgawk (for profiling awk programs) and dgawk for debugging them. The new internals enable the possibility of making just one executable that could perform all three functions (based on command-line options). This should simplify the build process and definitely will reduce the total installation "footprint".

4. The documentation could use further cleanup. Some of the examples cause the documentation to show its age. (Who uses dial-up BBS systems anymore?)

Acknowledgements

Thanks to Brian Kernighan, Stephen Davies and John Haque for reviewing this article.■

Arnold Robbins is a programmer, technical author, husband and father. A native of Atlanta, Georgia, he and his family have been living in Israel since 1997, where he now works writing software for a very large semiconductor manufacturing company. He has been involved with GNU Awk since 1987(!). In his non-copious spare time, he maintains gawk and its documentation, among other activities. Arnold is also the author or co-author of a number of UNIX- and Linux-related books from O'Reilly and Prentice Hall, which he hopes that all readers of this article will now run out and buy. For more information, see www.skeeve.com.

Resources

Gawk Home Page at the FSF: **www.gnu.org/software/gawk**

Gawk Project Home Page at Savannah, with Links and Instructions for Using Git: **savannah.gnu.org/projects/gawk**

Gawk Download Directory: **ftp.gnu.org/gnu/gawk**

Gawk Documentation: **www.gnu.org/software/gawk/manual**

Installation Instructions: **www.gnu.org/software/gawk/manual/html_node/Installation.html#Installation**

Brian Kernighan's "one true awk": **www.cs.princeton.edu/~bwk/btl.mirror**

The XMLGawk Download Page: **sourceforge.net/projects/xmlgawk**