Old Dogs, New Tricks

maddog reminds us of some old school programs that are still alive and kicking on modern systems. *By Jon "maddog" Hall*

Some discussions evoke passion in the FOSS world, such as Free Software vs. Open Source, Gnome vs. KDE, BSD vs. Linux, and MySQL vs. PostgreSQL. But one discussion has gone on for at least 30 years: Emacs vs. Vi.

As far as I am concerned, the discussion ended a long time ago. I just say they are both great text editors, and you should use whichever one you want to use. But when this email from the SAGE mailing list (Usenix special interest group for sys admins) showed up,

Subject: [SAGE] EMACS sucks

Since we're on the topics of load balance appliances and UTC time - I just wanted to say that vi rules and EMACS sucks.

That is all.

I was suckered into reading the thread. (I have purposely left the author anonymous to save them some grief.) Interestingly, the thread did not turn into the typical flame war of Emacs vs. Vi(m). The "graybeards" of SAGE, having been in computing a *long* time, instead turned it into a trip down memory lane of all the text editors they had used over the years, recalling the fondness they had for each one.

Although ed(1) was the first line-oriented text editor I used on Unix, it was not the first text editor I ever used. That honor belonged to a "dot editor" on the PDP-8. The four thousand 12-bit words of memory in the PDP-8 had to hold not only your application, but all of your data, so that first text editor was very, very simple.

Another reason the text editor was simple was because the user was typically on a very slow (5cps) paper-oriented terminal (e.g., an ASR-33), and the paper was expensive for a college student. Given the slowness and the cost, the editor would not print out *anything* until you told it to print. You had to keep in your head where you were in the file and just print every once in a while to make sure you really were where you thought you were.

People on the SAGE mailing list rapidly discussed many text editors: joe, nano, xedit (with REXX as its macro language), elvis, stevie, EDIT/EDT, EDIT/TPU, EVE/TPU, and others. Then I mentioned the "killer": TECO.

After that, the conversation deteriorated into people talking about how they did not really use text editors, but instead used the command string

```
cat > filename.txt
```

to create their files; others talked about using

```
cat >/vmlinux
```

to patch the kernel (somewhat in jest).

At times, no text editor was available, and the system would not come up without a specific file in place, usually a configuration file. People who knew what that file should contain might create it with the cat(1) command. Been there, done that.

However, what sparked this column was not just the stories about old text editors but thinking about how long some programs have been in existence and how they have morphed into something that is still useful in this age of modern-day computing. A lot of these early editors and programs were the beneficiaries of an exchange of ideas that made them even better.

TeX, a typesetting system first released in 1978, still exists; it has migrated up to version 3.x but has also spun off many front ends (LaTeX, KbibTeX, LyX, to name a few) that make TeX easier to use while extending its life and usefulness. TeX is still considered a premier typesetting program for books, particularly on technical subjects such as mathematics.

Even in the world of digital multimedia, some rather ancient design elements come into play. Command-line programs often do the majority of the work, whereas the GUI leads the user through a process that allows them to choose the right arguments for the command.

Another good example of this is nmh(1) – new MH message system. mh(1) was a set of programs for the command line that implemented the Rand MH mail system. I read mail for years with the mh set of command-line programs, then I started using xmh, a graphical front end for mh. Xmh, converted to use Motif, mxmh, was eventually changed to exmh with the incorporation of MIME and a series of other "modern" additions. mh-e allows mh to integrate with Emacs, as does almost everything else in the world.

I love this aspect of FOSS, in which features from one program are incorporated into another, extending the life of many fine programs. ■■■