An incomplete history of the QED Text Editor

Dennis Ritchie

The text editors **ed** and **vi**, still much-used on Unix systems and elsewhere, have a long history, some bits of which are recounted here.

The QED text editor was first written by Butler Lampson and Peter Deutsch for the Berkeley time-sharing system on the SDS 940; see their paper in C. ACM **10** #12 (December, 1967).

Ken Thompson used this QED at Berkeley before he came to Bell Labs, and among the first things he did on arriving was to write a new version for the MIT CTSS system. Written in IBM 7090 assembly language, it differed from the Berkeley version most notably in introducing regular expressions for specifying strings to seek within the document being edited, and to specify a substring for which a substitution should be made. Until that time, text editors could search for a literal string, and substitute for one, but not specify more general strings.

Ken not only introduced a new idea, he found an inventive implementation: on-the-fly compiling. Ken's QED compiled machine code for each regular expression that created a NDFA (non-deterministic finite automaton) to do the search. He published this in C. ACM **11** #6, and also received a patent for the technique: US Patent #3568156. (Who said you couldn't get a software patent until recently?)

While the Berkeley QED was character-oriented, the CTSS version was line-oriented. Ken's CTSS qed adopted from the Berkeley one the notion of multiple buffers to edit several files simultaneously and to move and copy text among them, and also the idea of executing a given buffer as editor commands, thus providing programmability. (TECO, which grew into Emacs, was approximately contemporaneous or just a bit later, and elaborated these ideas independently).

CTSS was used at Bell Labs as part of our participation in the Multics project, and as Ken began to use Multics, he wrote yet another version of QED for that system. This one was in BCPL, and instead of compiling to machine code, regular expressions were represented as trees that were interpreted by the editor. However, the basic technique of simulating an NDFA was still used.

During the same period (ca. 1967), I arrived at Bell Labs, and fairly soon thereafter the company began withdrawing from Multics. Unix was at first nonexistent and then embryonic, and the main computation resource was GE-TSS, GE's time-sharing system for GECOS. I liked QED, and so set out to write a version of it for this system. This, like Ken's CTSS QED, was written in assembly language and compiled its regular expressions to machine code.

This version was described in an internal technical memo, which is obtainable either in a slightly cut-down HTML browsable format, or reproduced as a scanned (and big: 1.1MB) PDF image.

I wrote some incomplete notes of this QED's internal workings. This was intended to be a published paper, but was never finished.

This QED was probably the most baroque and complicated of all its manifestations, particularly in pushing "regular expressions" well beyond what Kleene thought they should do. Other and later offspring of QED and its predecessors kept the central ideas, but were considerably simpler.

For a brief period in the 1970s, the GECOS QED served us as a scripting language; it was in some ways analogous to the Awk or Perl of today. It was used for such tasks as submitting batch jobs, formatting files for the

printer, collecting statistics on a file. A collection of macros to do various useful tasks was put in a commonly available place.

I've been reminded that a bit later, someone at Bell Labs wrote a version approximating this rendition for the IBM TSO system, and it was used at the Labs locations that used 360-style hardware. I don't remember the details of this.

Offspring

The "standard Unix editor" **ed** was first written by Ken Thompson for the PDP-7. It kept the basic text-line orientation, but radically simplified the regular expressions to include only the ***operator: no alternation, no parentheses. Where my QED embraced much of the context-free languages, this version could not even express all regular languages. It was not much of a loss.

Similarly, Ken's Unix **ed** ditched the notion of multiple buffers and of execution of buffers. Subsequent versions of **ed** for Unix (now written in C) began to add back some of the complexity (e.g. back-referencing in "regular" expressions, which now didn't quite include all the regular languages nor context-free languages, but did intrude a bit on the context-sensitive languages.)

Subsequent developments along this line include **vi**, done mostly by Bill Joy at UC Berkeley. The advance here was essentially to adapt the command-set of **ed** to a screen-editor format; instead of doing things on (possibly virtual) paper on a typewriter, **vi** keeps a current view of a piece of a document on the screen, while commands are typed on the bottom line. Keeping a constantly updated window on a part of the text being edited is now, of course, completely standard and accepted. The same program, under the name **ex**, works better on typewriter-like terminals; aside from some fancier commands and better diagnostics, it's essentially **ed**.

The current local state of the QED line of descent is exemplified by **sam**, by Rob Pike (see his "The Text Editor sam," in Software -- Practice and Experience **17** #11, Nov. 1987). It's available for FTP in both Unix and Windows versions; search for "sam editor" on netlib. **Sam** offers cut-and-paste, mouse-and-menu editing on its text windows together with a command window, the language of which is drawn from the QED line of descent.

The current implementation of regular expressions in Bell Labs research software uses algorithms close to those in Ken's CTSS and Multics versions, particularly the latter, because they don't try to compile to machine code. They have been elaborated somewhat in that they deal with a very large character set (Unicode), but they keep both the classical set of operators (*****, |, concatenation, grouping for parentheses) and the NDFA simulation. We've done some fairly elaborate versions of **grep** that construct DFAs dynamically on demand, use Boyer-Moore techniques for fast searching of fixed strings, and so on. In the end simple and solid seems best.

Going back to the 1970s, QED had other offspring. Bob Daley wrote an editor called **qedx** in PL/1 for the Multics project. See, for example, Tom Van Vleck's encyclopedic discussion of Multics things (this pointer being to things beginning with Q). The tart comments about the rebarbativeness of QED as a scripting language are on the mark, though I don't think that Emacs is all that much better.

A traditional (and maybe the nicest) version of QED was done at the University of Toronto by Tom Duff, Rob Pike, Hugh Redelmeier, and Dave Tilbrook; it supports multiple buffers, execution of buffers, and regular expressions with back-referencing. A more recent Canadian version, also a traditional-style descendant, is Fred, from Thinkage Ltd. It, like various earlier versions, is available for Honeywell GCOS systems and is still used as a scripting language as well as a line-editor for TTY-like communications interfaces.