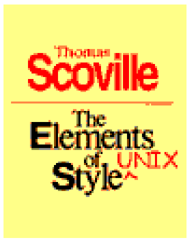
UNIX REVIEW'S PERFORMANCE COMPUTING

Highlights

The Elements Of Style: UNIX As Literature
If there's nothing different about UNIX people, how come so many were liberal-arts majors? It's the love of words that makes UNIX stand out.

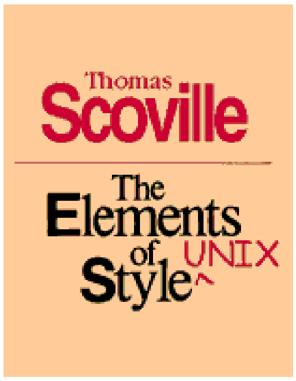


The Elements Of Style: UNIX As Literature

If there's nothing different about UNIX people, how come so many were liberal-arts majors? It's the love of words that makes UNIX stand out.

Thomas Scoville

In the late 1980s, I worked in the advanced R&D arm of the Silicon Valley's regional telephone company. My lab was populated mostly by Ph.D.s and gifted hackers. It was, as you might expect, an all-UNIX shop.



The manager of the group was an exception: no advanced degree, no technical credentials. He seemed pointedly self-conscious about it. We suspected he felt (wrongly, we agreed) underconfident of his education and intellect. One day, a story circulated through the group that confirmed our suspicions: the manager had confided he was indeed intimidated by the intelligence of the group, and was taking steps to remedy the situation. His prescription, though, was unanticipated: "I need to become more of an intellectual," he said. "I'm going to learn UNIX."

Needless to say, we made more than a little fun out of this. I mean, come on: as if UNIX could transform him into a mastermind, like the supplicating scarecrow in "The Wizard of Oz." I uncharitably imagined a variation on the old Charles Atlas ads: "Those senior engineers will never kick sand in my face again."

But part of me was sympathetic: "The boss isn't entirely wrong, is he? There is something different about UNIX people, isn't there?" In the years since, I've come to recognize what my old manager was getting at. I still think he was misguided, but in retrospect I think his belief was more accurate than I recognized at the time.

To be sure, the UNIX community has its own measure of technical parochialism and nerdy tunnel vision, but in my experience there seemed to be a suspicious overrepresentation of polyglots and liberal-arts folks in UNIX shops. I'll admit my evidence is sketchy and anecdotal. For instance, while banging out a line of shell, with a fellow engineer peering over my shoulder, I might make an intentionally obscure literary reference:

```
if test -z ‘ps -fe | grep whom‘
then
echo ^G
fi
# Let’s see for whom the bell tolls.
```

UNIX colleagues were much more likely to recognize and play in a way I'd never expect in the VMS shops, IBM's big-iron data centers, or DOS ghettos on my consulting beat.

Being a liberal-arts type myself (though I cleverly concealed this in my resume), I wondered why this should be true. My original explanation--UNIX's historical association with university computing environments, like UC Berkeley's--didn't hold up over the years; many of the UNIX-philiacs I met came from schools with small or absent computer science departments. There had to be a connection, but I had no plausible hypothesis.

It wasn't until I started regularly asking UNIX refuseniks what they didn't like about UNIX that better explanations emerged.

Some of the prevailing dislike had a distinctly populist flavor--people caught a whiff of snobbery about UNIX and regarded it with the same proletarian resentment usually reserved for highbrow institutions like opera or ballet. They had a point: until recently, UNIX was the *lingua franca* of computing's upper crust. The more harried, practical, and underprivileged of the computing world seemed to object to this aura of privilege. UNIX adepts historically have been a coddled bunch, and tend to be proud of their hard-won knowledge. But these class differences are fading fast in modern computing environments. Now UNIX engineers are more common, and low- or no-cost UNIX variations run on inexpensive hardware. Certainly UNIX folks aren't as coddled in the age of NT.

There was a standard litany of more specific criticisms: UNIX is difficult and time-consuming to learn. There are too many things to remember. It's arcane and needlessly complex.

But the most recurrent complaint was that it was too text-oriented. People really hated the command line, with all the utilities, obscure flags, and arguments they had to memorize. They hated all the typing. One mislaid character and you had to start over. Interestingly, this complaint came most often from users of the GUI-laden Macintosh or Windows platforms. People who had slaved away on DOS batch scripts or spent their days on character-based terminals of multiuser non-UNIX machines were less likely to express the same grievance.

Though I understood how people might be put off by having to remember such willfully obscure utility names like cat and grep, I continued to be puzzled at why they resented typing. Then I realized I could connect the complaint with the scores of "intellectual elite" (as my manager described them) in UNIX shops. The common thread was wordsmithing; a suspiciously high proportion of my UNIX colleagues had already developed, in some prior career, a comfort and fluency with text and printed words. They were adept readers and writers, and UNIX played handily to those strengths. UNIX was, in some sense, literature to them. Suddenly the overrepresentation of polyglots, liberal-arts types, and voracious readers in the UNIX community didn't seem so mysterious, and pointed the way to a deeper issue: in a world increasingly dominated by image culture (TV, movies, *.jpg* files), UNIX remains rooted in the culture of the word.

UNIX programmers express themselves in a rich vocabulary of system utilities and command-line arguments, along with a flexible, varied grammar and syntax. For UNIX enthusiasts, the language becomes second nature. Once, I overheard a conversation in a Palo Alto restaurant: "there used to be a shrimp-and-pasta plate here under ten bucks. Let me see...`cat menu | grep shrimp | test -lt $10`..." though not syntactically correct (and less-than-scintillating conversation), a diner from an NT shop probably couldn't have expressed himself as casually.

With UNIX, text--on the command line, STDIN, STDOUT, STDERR--is the primary interface mechanism: UNIX system utilities are a sort of Lego construction set for word-smiths. Pipes and filters connect one utility to the next, text flows invisibly between. Working with a shell, awk/lex derivatives, or the utility set is literally a word dance.

Working on the command line, hands poised over the keys uninterrupted by frequent reaches for the mouse, is a posture familiar to wordsmiths (especially the really old guys who once worked on teletypes or electric typewriters). It makes some of the same demands as writing an essay. Both require composition skills. Both demand a thorough knowledge of grammar and syntax. Both reward mastery with powerful, compact expression.

At the risk of alienating both techies and writers alike, I also suggest that UNIX offers something else prized in literature: a coherence, a consistent style, something writers call a voice. It doesn't take much exposure to UNIX before you realize that the UNIX core was the creation of a very few well-synchronized minds. I've never met Dennis Ritchie, Brian Kernighan, or Ken Thompson, but after a decade and a half on UNIX I imagine I might greet them as friends, knowing something of the shape of their thoughts.

You might argue that UNIX is as visually oriented as other OSs. Modern UNIX offerings certainly have their fair share of GUI-based OS interfaces. In practice though, the UNIX core subverts them; they end up serving UNIX's tradition of word culture, not replacing it. Take a look at the console of most UNIX workstations: half the windows you see are terminal emulators with command-line prompts or vi jobs running within.

Nowhere is this word/image culture tension better represented than in the contrast between UNIX and NT. When the much-vaunted UNIX-killer arrived a few years ago, backed by the full faith and credit of the Redmond juggernaut, I approached it with an open mind. But NT left me cold. There was something deeply unsatisfying about it. I had that ineffable feeling (apologies to Gertrude Stein) there was no *there* there. Granted, I already knew the major themes of system and network administration from my UNIX days, and I will admit that registry hacking did vex me for a few days, but after my short scramble up the learning curve I looked back at UNIX with the feeling I'd been demoted from a backhoe to a leaf-blower. NT just didn't offer room to move. The one-size-fits-all, point-and-click, we've-already-anticipated-all-your-needs world of NT had me yearning for those obscure command-line flags and `man -k`. I wanted to craft my own solutions from my own toolbox, not have my ideas slammed into the visually homogenous, prepackaged, Soviet world of Microsoft Foundation Classes.

NT was definitely much too close to image culture for my comfort: endless point-and-click graphical dialog boxes, hunting around the screen with the mouse, pop-up after pop-up demanding my attention. The experience was almost exclusively reactive. Every task demanded a GUI-based utility front-end loaded with insidious assumptions about how to visualize (and thus conceptualize) the operation. I couldn’t think "outside the box" because everything literally was a box. There was no opportunity for ad hoc consideration of how a task might alternately be performed.

I will admit NT made my life easier in some respects. I found myself doing less remembering (names of utilities, command arguments, syntax) and more recognizing (solution components associated with check boxes, radio buttons, and pull-downs). I spent much less time typing. Certainly my right hand spent much more time herding the mouse around the desktop. But after a few months I started to get a tired, desolate feeling, akin to the fatigue I feel after too much channel surfing or videogaming: too much time spent reacting, not enough spent in active analysis and expression. In short, image-culture burnout.

The one ray of light that illuminated my tenure in NT environments was the burgeoning popularity of Perl. Perl seemed to find its way into NT shops as a CGI solution for Web development, but people quickly recognized its power and adopted it for uses far outside the scope of Web development: system administration, revision control, remote file distribution, network administration. The irony is that Perl itself is a subset of UNIX features condensed into a quick-and-dirty scripting language. In a literary light, if UNIX is the Great Novel, Perl is the *Cliffs Notes*.

Mastery of UNIX, like mastery of language, offers real freedom. The price of freedom is always dear, but there’s no substitute. Personally, I’d rather pay for my freedom than live in a bitmapped, pop-up-happy dungeon like NT. I’m hoping that as IT folks become more seasoned and less impressed by superficial convenience at the expense of real freedom, they will yearn for the kind of freedom and responsibility UNIX allows. When they do, UNIX will be there to fill the need.

Thomas Scoville has been wrestling with UNIX since 1983. He currently works at Expert Support Inc. in Mountain View, CA.

BACK TO TOP