The cat -v discussion is irrelevant

David M. Tilbrook

Imperial Software Technology

ABSTRACT

"This feels like a Republican victory party ..." -- Vic Vyssotsky, 1985

The so-called UNIX-philosophy has been preached from the pulpits by the high-priests of orthodoxy at many a UNIX conference. Does this zealous fervour have any connection with the failure of UNIX to make any significant advances in recent years? Why are there large areas of computer science that seem to be ignored by the UNIX world and why is that when some areas are attempted on UNIX they prove to be as unworkable or cumbersome as they were on the more traditional environments? This paper is a highly personal view by one who has been dismayed by the failure of the UNIX community (himself included) to make any significant advances in real-time systems (whatever they may be), software engineering (something palatable at least) and a variety of other problems.

> "The first clue something is wrong with APL is that whenever two APL users/programmers get together they start discussing extensions to the language." -- Tom Duff, 1975

This is not a attack on the 'cat -v' talk or ideas. For the most part I agree with Mr. Pike's main points. However, I do feel that there have been two major problems with respect to the original talk:

1) Many people reacted with misdirected hostility to BSD, a system that has made tremendous contributions to the community in some areas and certainly provides a much superior environment to other available environments for certain applications;

2) The arguments were largely expending energy in the wrong direction, concentrating on minor issues (i.e., stylistic points such as flags to commands) rather than fundamental problems with current UNIX implementations and uses.

I am not going to defend the first point. There are large parts of the BSD system that I find dismaying. However, my experience with the commercially available alternatives has been far from pleasant [1]. Rather, I would like to put the case that the arguments and discussions about UNIX are highly reminiscent of the APL hacks discussing the ravel operator. The fundamental limitations of the system are not going to be overcome through either stylistic adherence to a set of loosely defined principles or the power coding of myopic hackers in universities of vulture capital shops.

It is high time that the UNIX research community recognize and accept that it is time to apply one of the so-called UNIX philosophy principles; that one should be prepared to throw out tools and start again, and the tool that should be thrown away is UNIX itself.

To defend this position is difficult and unpopular. Such a move threatens many people. There is a huge investment in UNIX at both the individual and corporate levels. Indeed I am not proposing that everyone rip up their licenses. There is a large segment of the UNIX world that must continue to use UNIX and will do so for a long time. However, UNIX's successor is not going to be reached by constant enhancement of the current system: neither is it going to evolve in parallel with what is becoming a major obsession within

[1] When the Cambridge conference panel was asked to choose between 4.2bsd and System V, five of the six members chose 4.2bsd. The sixth chose V8, which is not commercially available.

the UNIX community, that of satisfying the market place.

It is essential to recognize that the evolution of UNIX thus far has been less than graceful and the 'real' progress less than spectacular [2]. I recognize that there have been improvements in some ideas and facilities and that some have definitely been of major importance to the computer science community (e.g., Make). However, there is little evidence that there will be any change in the way UNIX evolves and a great deal of evidence that progress is going to be hampered by commercial interests.

This gradual decrease of progress and the accompanying increase in complexity and problems (i.e., bugs) in a system's 'middle age' is not unprecedented. In fact it seems inevitable. What sufficed or was deemed unessential in a system's beginning in the interest of meeting initial requirements or objectives inevitably become insufficient and essential as the system's use and its users' objectives change.

In the case of UNIX, the initial developers created a system that was the well designed integration of four or five good ideas. It was relatively conventional in approach, on a popular machine, fairly small and understandable, etc., etc., etc. It was not designed to do everything. In fact certain areas were deliberately excluded (e.g., Real-time, IPC, data-base support) and it was there that the problem began.

There were many efforts to shoe-horn the missing parts into the UNIX environment. This was done out of a need to handle certain applications (e.g., real-time in MERT) while simultaneously taking full advantage of the tools and facilities offered by the base system. Sometimes UNIX was sold into areas which were completely outside the UNIX realm and required substantial developments to satisfy the needs of those areas. For example many of us are guilty of building applications which are ill-suited to UNIX, but required if UNIX was to be available to us for our own uses (e.g., COBOL compilers, RT-11 emulators, data-base systems).

From a small two person development, UNIX exploded into a mega-project with rapidly expanding and diversifying objectives and applications, and as is inevitable, rapidly decreasing coherence and quality.

Furthermore, due to the nature of the UNIX community, some problems have been tolerated due to cost of rectification, and certain application areas have either been ignored (as not being of interest) or developed by teams who lacked sufficient UNIX-experience to ensure that their implementations were compatible with the rest of the system [3].

To catalogue all the sins and follies would take too long. The list of things UNIX does not handle or handles poorly should be obvious to anyone who has used it for any period of time or who has tried to bend it further than it yields (e.g., trying to ensure reliability when signals must be handled). My own major concerns relate to controlling multiple process applications (what facilities exist are primitive, undisciplined and complicated [4]), to documentation which is largely unchanged in 15 years (why do people still insist on orienting documentation to paper output [5]) and a seemingly inability of suppliers to adequately test their products (I mention no names). These are just some of many areas where UNIX, whilst offering a good solution for 80% of the problem, seems sadly deficient for the remaining 20%. Others have expressed views about its inadequacy on large systems (it was initially a small-machine system) and the problems with security and reliability which are likely to remain unsolved.

---

[2] I ignore the commercial and marketing success as being of little interest to the researcher except in that it has meant increased availability.

[3] Signal handling is an area that is largely unchanged since the early days of UNIX despite the fact that it is extremely difficult to create reliable systems that avoid all possible race conditions. SCCS is a splendid example of a non-UNIX tool in the way files are named, and input and flags are handled. Worst of all is the difficulty encountered in managing large numbers of related source files, facility of UNIX that has proved to be so important.

[4] There is a glimmer of hope that streams can offer some relief in the area of multiple process control and communication. However, it is unlikely, when commercially available, that a style of use will have evolved that will provide the 'standard' way of building systems from components. Part of UNIX's success is due to the fact that the developers used and tuned it for a number of years before its release to the outside world thus had time to experiment sufficiently to develop (perhaps unconsciously) a 'standard' approach to software and its combination.

[5] The answer is of course so they can sell books to a captive readership and documentation reading software packages to those who recognize paper is inadequate.

Having complained a great deal about the system, I should now propose a solution. However, I recognize that as I am a long-time UNIX user I am no longer qualified to create such a system. It is unlikely that the next generation of the research computing environment can come from the UNIX community itself. I hope that the developers will be aware of UNIX's strengths, however, a team whose primary background is UNIX will probably not recognize a solution to a problem (unless it is called grep) without trying to cast it into the UNIX mould.

If this is the case, what role can the UNIX research community play in the advance to the 'UNIX' replacement?

Our first priority is to recognize our true requirements for a computing environment. Such an evaluation should be done without regard to actual implementation considerations. This is an ambitious assignment, which must be done, however, to evaluate other solutions with respect to their acceptability. Such an evaluation should be done on the basis of how well it fulfills our particular needs, not our prejudices towards particular solutions.

Secondly we must understand how the basic UNIX features or facilities succeed or fail in fulfilling our needs. To demand the existence of a facility without an appreciation of its true importance or value can be dangerous. For example, it is unlikely that any system that does not provide some sort of hierarchical file organization would be acceptable to most of us. But why? Can we honestly reject a system that doesn't provide such a file system on that basis alone? Far too often users fail to recognize the difference between a requirement and a partial fulfillment of that requirement.

Finally we must be prepared to honestly accept UNIX's short-comings and be prepared to evaluate alternatives without prejudice. To continue to design or code around UNIX short-comings is too costly and it is unlikely that some of the rare 'real' advances (e.g., streams, C++) will be available without unacceptable costs [6].

The last task is the most difficult to accomplish. The UNIX community has worked very hard at promoting UNIX for their own needs and in many cases are now unable to separate the marketing hype from the true value. We have been telling each other how wonderful the system is for so long that we are in danger of suffering from the symptoms discussed in the following article from The Guardian, Saturday, August 31, 1985, entitled: "What the team thinks is wrong"

> "Group think" can be self-defeating, said Dr. Pat Shipley, an occupational psychologist, at a session on ergonomics, the science of the workplace.
>
> "Group think was used to describe the British Cabinet's deliberations leading to the Falklands war," said Dr. Shipley.
>
> The irrational dimensions of group think underlying impaired critical judgement included: the group protecting itself from adverse information with self appointed mind guards; stereotyping the enemy as too stupid to be a threat, or too evil to negotiate with; dealing with challenges to cherished values and assumptions by ignoring them or rationalising them away.

I hope that I am wrong.

[6] It has been announced that future releases of UNIX for the VAX will not be forthcoming from the original supplier and, from my experience, the alternative offered is not a pleasant one.