**When Is a Rose Not a Rose?**

Having studied Linus's behavior and formed a theory about why it was successful, I made a conscious decision to test this theory on my new (admittedly much less complex and ambitious) project.

But the first thing I did was reorganize and simplify popclient a lot.

Carl Harris's implementation was very sound, but exhibited a kind of unnecessary complexity common to many C programmers.

He treated the code as central and the data structures as support for the code.

As a result, the code was beautiful but the data structure design ad-hoc and rather ugly (at least by the high standards of this veteran LISP hacker).

I had another purpose for rewriting besides improving the code and the data structure design, however.

That was to evolve it into something I understood completely.

It's no fun to be responsible for fixing bugs in a program you don't understand.

For the first month or so, then, I was simply following out the implications of Carl's basic design. The first serious change I made was to add IMAP support.

I did this by reorganizing the protocol machines into a generic driver and three method tables (for POP2, POP3, and IMAP).

This and the previous changes illustrate a general principle that's good for programmers to keep in mind, especially in languages like C that don't naturally do dynamic typing:

1. Smart data structures and dumb code works a lot better than the other way around.

Brooks, Chapter 9: ``Show me your flowchart and conceal your tables, and I shall continue to be mystified. Show me your tables, and I won't usually need your flowchart; it'll be obvious.'' Allowing for thirty years of terminological/cultural shift, it's the same point.

At this point (early September 1996, about six weeks from zero) I started thinking that a name change might be in order—after all, it wasn't just a POP client any more. But I hesitated, because there was as yet nothing genuinely new in the design. My version of popclient had yet to develop an identity of its own.

That changed, radically, when popclient learned how to forward fetched mail to the SMTP port.

I'll get to that in a moment. But first: I said earlier that I'd decided to use this project to test my theory about what Linus Torvalds had done right. How (you may well ask) did I do that? In these ways:

* + I released early and often (almost never less often than every ten days; during periods of intense development, once a day).
  + I grew my beta list by adding to it everyone who contacted me about fetchmail.
  + I sent chatty announcements to the beta list whenever I released, encouraging people to participate.
  + And I listened to my beta-testers, polling them about design decisions and stroking them whenever they sent in patches and feedback.

The payoff from these simple measures was immediate.

From the beginning of the project, I got bug reports of a quality most developers would kill for, often with good fixes attached.

I got thoughtful criticism, I got fan mail, I got intelligent feature suggestions. Which leads to:

1. If you treat your beta-testers as if they're your most valuable resource, they will respond by becoming your most valuable resource.

One interesting measure of fetchmail's success is the sheer size of the project beta list, fetchmail-friends. At the time of latest revision of this paper (November 2000) it has 287 members and is adding two or three a week.

Actually, when I revised in late May 1997 I found the list was beginning to lose members from its high of close to 300 for an interesting reason. Several people have asked me to unsubscribe them because fetchmail is working so well for them that they no longer need to see the list traffic! Perhaps this is part of the normal life-cycle of a mature bazaar-style project.