Crossroads
The ACM Student Magazine

Programming Perl

An interview with Larry Wall

by***Lorrie Faith Cranor***

The first thing you notice about the Perl book is the camel. The hardy beast is standing nonchalantly and grinning ever so slyly. This is not just any old desert dwelling dromedary. This is a camel with an attitude. And that's the way Perl creator and *Programming Perl* co-author Larry Wall wanted it to be. Wall likens the camel-with-an-attitude to his free **Practical Extraction and Report Language -- Perl**. ``It's ugly but useful,'' he says.

Since 1986 when Wall first developed Perl -- an interpreted data reduction language -- to solve a problem that *awk* could not handle, the language has grown to include a variety of built-in functions, associative arrays, regular expression handling, object-oriented features, and some of the most useful bits of several other programming languages. It is now a popular tool for system administrators and programmers who want to develop code to manipulate strings, files, and processes. Recently it has become the language of choice for programming interactive applications on the World Wide Web.

As Wall has developed and improved Perl, utility has always been a higher priority than theoretical elegance. He criticizes other modern languages that have been developed by people who ``try to define their languages such that you can't do anything bad.'' As a result, he says, ``the act of programming becomes joyless.'' But, not so with Perl. ``The thing that people really like about Perl is that they have fun.'' This is perhaps

why Perl has become so popular.

With the help of volunteers from around the world, Wall put the finishing touches on Perl version 5 in mid-October. Perl 5 introduces not only new features -- including simple object oriented ideas and a more readable syntax -- but also a new philosophy. Wall explains that as Perl developed, new features were continuously added, causing the language to change constantly. ``With Perl 5 I've sort of come to the realization that we need to actually stabilize the language itself and provide an official way of extending it.'' Therefore, Perl 5 includes mechanisms that allow programmers to add reusable modules to extend the Perl language. The object-oriented features were necessary to add this extensibility. In addition, they allow Perl and C++ to be easily used within the same program.

As with versions 1 through 4, the new features of version 5 have been designed so that they can be learned incrementally. Unlike other computer languages ``where you have to practically know the whole language before you can use any of it usefully,'' Wall says that new Perl programmers can learn the subset of Perl they need to solve a particular problem without having to learn the rest of the language. Because of this property, Wall suggests Perl might be a good introductory language for beginning computer science students.

While Wall hopes the new philosophy will help Perl will remain stable for a while, he predicts the continued development of new programming languages, particularly those that make programming accessible to ordinary people. ``I think to the extent that ordinary people are going to be programming, they're not going to be programming in C++ -- because you practically have to have a degree in computer science to do that.'' However, he says that simply adding more English syntax to programming languages is not likely to make programming easier. ``When they first came out with Cobol they said `this is a very English-like language,' but I think they confused the process of just throwing a few English phrases into the language and ending their sentences with periods with some of the deeper aspects of natural language. Our computer languages will start functioning more and more like natural languages, but there is a large amount of misunderstanding in the current computer science community about how natural languages actually work.''

And Wall, a linguist by training, is certainly qualified to talk about natural languages. In fact, he says his background in linguistics and his observations about natural language have influenced his programming language development work. ``Linguists realize the

limitations of language and realize that when you are trying to solve a problem in a natural language you sort of whack at it until you get the thing into the shape that you want it to be,'' he explains. ``I think people program similarly. They rough things out and hack things in. I think a lot of computer scientists believe programmers are omniscient and can figure out before hand all the design criteria, get all the ducks in a row -- and then magically the program just writes itself because it will be obvious. I don't think real people think that way. I don't think they talk that way, or write that way.''

Wall also credits his linguistic background for his views on ambiguity in programming languages. ``My view on local ambiguity is that it's okay because people deal with it well. On the level of phrases and sentences people are very good at figuring out exactly what's going on -- so I think a computer language ought to be allowed to have ambiguity at that level.'' He adds that humans use various natural language clues to disambiguate sentences. In Perl, special characters serve to disambiguate variable types. For example, variable names that begin with a $ refer to singular variables (scalars) while variable names that begin with a @ or a % refer to plural variables (arrays). As with English where a member of a group is described with singular words, an array element is referenced with the singular $.

In addition, Wall has observed that unlike most computer languages, ``natural languages don't have any `theoretical axes' to grind.'' He adds, ``Natural languages typically don't have any shame in borrowing things from other languages, unless of course you're French.'' He pauses for a moment and looks through his notes for a statement he likes so much he has it written down: ``For most people the perceived usefulness of a computer language is inversely proportional to the number of theoretical axes the language intends to grind.''

But while Wall intends Perl to be useful, he does not intend it to replace other programming languages that are also useful for solving certain types of problems. ``I've tried not to reinvent any other language with Perl. So if you see some particular problem that C is good at, I've not tried to make Perl that good in that realm. For low level bit diddling, C is still going to be better. For very large, difficult to manage projects, C++ is going to better. For just starting and stopping processes, shells are going to be better. These are all sort of LegoLand type problems. The place where Perl excels is not where you want **Legos**, but where you want glue -- where you don't want Lincoln Logs so much as you want pipe cleaners. Its sort of officially a chewing gum

and baling wire language.''

In fact, Wall says, ``The Perl slogan is, `There is more than one way to do it.' People have often taken that to mean that there is more than one way to do it within Perl. But I apply the same thing outside of Perl also. I know a lot of people use Perl for what it's useful for. But I don't expect them to take themselves off to a monastery and just write Perl scripts all day.''

Wall says Perl's biggest competitors -- **REXX**, **Tcl**, **Python**, and **Scheme** -- are useful for similar things that Perl is useful for. Although he prefers Perl because of it's efficiency, language structure, and lack of theoretical axes, Wall says he will not engage in language wars. ``I have a firm policy against making enemies,'' he states.

Despite his training as a linguist, Wall turned to computer science because of its more lucrative job prospects. However, this was not a major career change, as Wall had spent three of his eight years as an undergraduate at Seattle Pacific University working at the school's computer center. And although he met his first computer (a PDP-11) at Seattle Pacific, Wall began programming in 1972 when his high school got a programmable calculator. ``You could program 120 steps into [it] and I very nearly taught it to play tic-tac-toe -- but I just couldn't get that squeezed down to 120 steps.''

While at Seattle Pacific, Wall says he ``was vaguely acquainted with Bill Gates.'' Wall explains, ``[Gates] was still programming for the experimental college at the University of Washington and they had been jacking up the computer rates on him. So he came over and was using our little PDP 11 because we'd give him cheaper computer time.''

Since graduating from Seattle Pacific, Wall attended graduate school at U.C. Berkeley and U.C.L.A., and worked at Unisys and the Jet Propulsion Laboratory. In his spare time he developed several free UNIX programs, including the rn news reader, metaconfig, and Perl. He currently holds the position of Senior Scientist (``oldest hacker'') at NetLabs and resides in Mountain View, California with his wife, Gloria, and their four children.

Wall says there have been a number of people who have inspired him or served as role models throughout his life. Among them are 19th century writer George Macdonald, his grandmother who earned her Ph.D. in comparative literature when she was 77, his

parents, and his wife.

Wall describes himself as "one of these people who's been interested in too many things.'' It may be his diverse sets of interests that have led him to take on the development of Perl and other programs. Or perhaps it is what he refers to in *Programming Perl* as the ``three great virtues of a programmer: laziness, impatience, and hubris.'' But Wall says there are three things that have motivated him to invest the enormous amount of time and effort necessary to keep improving Perl. Besides a simple desire to help people and to create something with some lasting significance, Wall says he is motivated by his notion of creativity and how he believes God views the world as a work of art. "If its okay for [God] to make certain kinds of amoral decisions than it's okay for us to be artists also and sling some paint around and make some pretty pictures. I think that I just feel deep down that its fitting in with the overall scheme of things.'' He pauses before adding, ``No pun intended.''