Perl

Ed Knapen

"Practical Extraction and Report Language or Pathologically Eclectic Rubbish Lister? Perl has devoted fans and fierce enemies. This paper describes Perl's post modern philosophy, shows some of its exotic operators and explains the link between Perl, Apostle Paul and perlmonks.com."

What is Perl?

Perl is a high-level scripting and programming language originally created by Larry Wall. It derives from the C programming language and to a lesser extent from sed, awk, Unix shells, and at least a dozen other tools and languages. Perl provides few but very powerful sets of data types (numbers, strings, and references) and structures (hashes and lists). Hashes (associative arrays) use strings as indices. Lists are numerically indexable and can also act as stacks, queues, or even double-ended queues. Perl emphasizes support for common application-oriented tasks: important features include built-in regular expressions, "text munging", file I/O, and report generation.

Here are two command line Perl scripts that should give you an impression:

```
ls | perl -pe
 '$i=$.;s//\e[3@{[$i++%7+1]};1m/g;
 END{print "\033[0m"}'

perl -pe "@ph=map {ucfirst(lc)}
 split(/[\s.,-]+/);
 print qq(@ph)" foo.txt
```

Officially, Perl is an acronym for *Practical Extraction and Report Language*, but the alternative *Pathologically Eclectic Rubbish Lister* is often used as well. Its roots are in UNIX but today you will find Perl on a wide range of computing platforms, including Mac, Windows and EPOC.

Not surprising given its origins, Perl is almost the perfect tool for system administrators: it allows the easy manipulation of files and process information, and easy automation of all kinds of tasks.

But Perl's process, file, and text manipulation facilities make it also particularly well-suited for tasks involving quick prototyping, system utilities, software tools, system management tasks, database access, networking, and world wide web programming. Besides system administrators, these strengths make it especially popular with CGI script authors (most CGI programs are written in Perl), but mathematicians, geneticists, journalists, and even managers also use Perl. Maybe you should use it as well.

Who is Larry Wall?

What Linus is to Linux, Larry is to Perl. Son of a pastor in Los Angeles, Larry Wall started off as a programmer and system administrator with a rich heritage of ideas and skills. Among these was the notion that everybody can change the world. He majored in Natural and Artificial Languages and attended grad school in linguistics. After the advent of Perl and the Perl book, which became a best seller, came royalties and a position with the publisher, O'Reilly & Associates. Today, O'Reilly pays Larry to do whatever he likes, as long as it helps Perl. And he generally eats breakfast at lunchtime.

In the beginning

On October 18, 1987, Perl 1.0 was posted to the Usenet group comp.sources. In Larry's own words:

"The beginnings of Perl were directly inspired by running into a problem I couldn't solve with the tools I had. Or rather, that I couldn't easily solve. As the Apostle Paul so succinctly put it, "All things are possible, but not all things are expedient." I could have solved my problem with awk and shell eventually, but I possess a fortuitous surplus of the three chief virtues of a programmer: Laziness, Impatience and Hubris. I was too lazy to do it in awk because it would have been hard to get awk to jump through the hoops I was wanting it to jump through. I was too impatient to wait for awk to finish because it was so slow. And finally, I had the hubris to think I could do better."

Perl was created and has been evolving by combining all cool features from C, sh, csh, grep, sed, awk, Fortran, COBOL, PL/I, BASIC-PLUS, SNOBOL, Lisp, Ada, C++, Python, etc. Or, turning it around, by leaving out all the unwanted features of all these languages.

As for the name, Larry wanted a short name with positive connotations, looked at every three and four-letter word in the dictionary and rejected them all. Eventually he came up with the name "pearl", with the gloss *Practical Extraction and Report Language*. The "a" was dropped because of the existence of some obscure graphics language with the same name.

Post modern

Perl is unique in its aim to be post modern, as opposed to being based on modernism. Post modern is, according to Larry, what the American culture has become, not just in music and literature, but also in fashion, architecture and in overall multi cultural awareness.

To Larry, modernism was based on a kind of arrogance that elevated originality above all else, and led designers to believe that if they thought of something cool, it must be considered universally cool. That is, if something is worth doing, it is worth driving into the ground to the exclusion of all other approaches. Look at the use of parentheses in Lisp or the use of white space as syntax in Python. Or at the mandatory use of objects in many languages, including Java.

In contrast, post modernism allows for cultural and personal context in the interpretation of any work of art. It's the origin of the Perl slogan: "There's More Than One Way To Do It!" The reason Perl gives you more than one way to do anything is a belief that computer programmers want to be creative, and they may have many different reasons for wanting to write code a particular way. What you choose to optimize for is your concern, not Perl's. Perl supplies the paint (be it strings, associative arrays or objects), but the programmer paints the picture.

A second Perl philosophy is its aim for "No Limits". Maximum string or array lengths or similar boundaries are not or hard to find. Usually the only limit is the amount of free memory in your computer: the whole Linux kernel can be read into one (binary) string, patched and written back again using Perl.

Exotica

Perl contains over 50 special variables, most of them a combination of the $-sign (indicating a scalar variable) and a single character, for often used information. So is $_ the default input and pattern-searching space, while $. is the current input line number, $> the effective uid of the process, $+ the last bracket matched by the last search pattern, etc.

In addition, there are over 50 operators, like the usual +, ++, +=, etc., but also more unusual ones, like <> to read from a filehandle, <=> for numerical comparison (returning -1, 0, or 1) and =~ for search, substitute or translate.

Example:

```
open (SRC, $_[0]) ||
```

```
      die "Can't find source file";
   while (<SRC>) {
      # match 'type' keyword
      if ( /type\s*=\s*"(\w*)"/ )
         { print $1."\n"; }
   }
   close SRC;
```

Many syntactic elements can be omitted, like parentheses around function arguments. The following two statements are equally valid:

```
print "Hello world";
print("Hello world");
```

Even semantic elements can sometimes be omitted. To read input from a file,

```
while ($_ = <STDIN>) { print $_; }
```

can be abbreviated to

```
while (<>) { print }
```

Perl allows execution of statements to depend on modifiers (if, unless, while, until). The following statements are all equivalent:

```
if ($energy < 0) { $nLives--; }
$nLives-- if ($energy < 0);
$nLives-- unless ($energy >= 0);
unless ($energy >= 0) { $nLives--; }
($energy < 0) && $nLives--;
($energy >= 0) || $nLives--;
```

In Perl, you can use the form that fits best with your ideas about what highlights the most important part of the statement.

As many other string based scripting languages, Perl interprets variables either as numbers or as strings depending on the context. A similar context dependency holds for scalars and arrays. If, for example, an array is assigned to a scalar variable, the length of the array will be assigned.

Passing arguments to a subroutine can only be done by resorting to list-context functions to retrieve the values, like:

```
do processFile($fileName);

sub processFile {
   print "Reading " . $_[0] ." file\n";
   open (SRC, $_[0]) ||
      die "Can't find source file";
   ...
}
```

Applications

Another Perl anecdote from Larry:

"A couple of years ago, I ran into someone at a trade show who was representing the National Security Agency. He mentioned to someone else in passing that he'd written a filter program in Perl, so without telling him who I was, I asked him if I could tell people that the NSA uses Perl. His response was, "Doesn't everyone?" So now I don't tell people the NSA uses Perl. I merely tell people the NSA thinks everyone uses Perl. They should know, after all."

Perl is used on Wall Street, in CGI scripts, in the robots and spiders that navigate the Web and build much of the various on-line databases. If you've ever been spammed, your e-mail address was almost certainly gleaned from the Net using a Perl script. The spam itself was likely sent via a Perl script.

Personally, my first Perl script was a 15-liner called "bgr", which changed the background picture on my OOTI machine every five minutes or so.

There are 800 or so reusable extension modules in the Comprehensive Perl Archive Network (CPAN). Glancing through those modules will give the impression that Perl has interfaces to almost everything in the world.

Comparison

Perl can be compared with other scripting languages like Tcl, Javascript and Python. Of these three, Tcl is the closest relation. Compared to Perl, Tcl's syntax is clean and simple, consisting of only a few building blocks. This makes it easier to teach non-

programmers Tcl. On the other hand, Perl gives you more power of expression. While you can certainly write awful and unreadable Perl programs, Perl's syntax and vocabulary also allow programmers to express exactly what they think, without having to resort to unnecessary constructions.

One advantage of Tcl over Perl is the availability of the graphical toolkit (Tk). This extension of Tcl is so tightly integrated that Tcl is normally referred to as Tcl/Tk. With Tk, three or four lines of code is all it takes to create a window with a clickable button or an editable input field. Since Tcl/Tk is also an interpreted language, you can play around with fonts, colours and dimensions until your interface is just right, without the need for recompilation. It is possible to use graphical extensions in combination with Perl (even the combination Perl/Tk is possible), but these are more awkward to use than the integrated Tcl/Tk pair.

Both Perl and Tcl are implemented in C and can be embedded into your own application code, to extend it with the power to interpret scripts.

Objects were only introduced in version 5 of Perl, which made a seamless integration impossible. If you are an object wizard who wants to write system administration scripts using objects, then maybe a language like Python is a better option for you.

Perl has the largest user base of all scripting languages. For whatever you need, chances are there is a package at CPAN available that does it. This is especially true in the field of CGI scripts. So if you need a quick start for a script that requires database connectivity or XML parsing, then Perl is a good choice.

Future

Last summer, Larry Wall announced the start of the development of Perl 6. In contrast to the first five versions, which followed an evolutionary development, a more organised approached with community input has been set up. If you have a desire to help in the crusade to make Perl a better place then peruse the Perl 6 developers page and get involved. The first alpha is expected by Summer 2001.

Links

For more information on Perl, try the following links:

www.perl.com
www.cpan.org
www.perldoc.com
www.perlmonks.com
www.perl.org/perl6

Biography Born in Elsloo (Limburg), The Netherlands, in 1970, Ed Knapen graduated in computing science from the Eindhoven University of Technology, The Netherlands. In 1995, he graduated in the postgraduate programme on software technology at the Stan Ackermans Institute in Eindhoven. This programme was concluded with a project carried out at the National Aerospace Laboratory (NLR) in Amsterdam, The Netherlands. Since then he has been employed by NLR to work on research and development in the field of application of information and communication technology in airport operations, air transport and air traffic control. His Perl programs are in use at NLR, in European aerospace companies and institutes and by an international chess organisation.