The journal of Apple technology.

Volume Number: 14 (1998)
Issue Number: 2
Column Tag: Alternative Environments

MacPerl: A Developer's Overview

by Rich Morin

The Power of Perl, the ease of Macintosh

Overview

Perl (and, by extension, MacPerl) is a convenient and powerful language for administrative programming, CGI scripting on the World Wide Web, data analysis and filtering (such as error checking and reformatting), network programming, and more. In short, Perl can be used for almost any programming project you may have in mind.

On Unix systems, Perl is well on its way toward taking over all substantial scripting functions, supplanting traditional tools such as sh, awk, and sed. On Macintosh systems, Perl can be used just as readily (with the added attraction that there is nothing to "unlearn").

Perl was created ten years ago when its author, Larry Wall, decided that existing scripting languages were insufficiently powerful for the distributed, bug reporting project he was working on. Seeing the potential in his new tool, Larry was gracious enough to release Perl as freeware (freely redistributable software in source and binary forms).

Other programmers picked it up, tried it, liked what they saw, and suggested enhancements and modifications. In a few years, Perl grew substantially in capabilities and in adherents, and was well on its way to becoming one of the most powerful and popular computer languages in use today. Although Perl was originally written for the Unix operating system, it has since been ported to many different systems.

The recent rise in popularity of the World Wide Web has assured the popularity of Perl for some time to come. Perl is not a "strongly-hyped language" like Java, but it has shown itself to be an indispensible tool for creating and maintaining Web sites. Perl is used for CGI scripting, site management, and many other duties.

MacPerl (ported by Matthias Neeracher) has also been in existence for several years, but its popularity has not increased at the same rate, and certainly not to the level I feel it deserves. The Macintosh is a friendly, easy to use, and very popular computer system. MacPerl is an elegant and friendly Macintosh adaptation of an extraordinarily powerful (and popular) programming language. Why hasn't it taken off as quickly as Perl has?

Many potential MacPerl users are unaware that Perl (let alone a Macintosh version) exists! Most Macintosh magazines, rightly or wrongly, shy away from programming articles. Also, lacking any commercial reference material or distribution CD for MacPerl, many prospective users may have felt apprehensive about getting involved. PTF's MacPerl product (and articles like this one!) should resolve these issues, helping the community to grow substantially.

Language Summary

Perl syntax and fundamental capabilities are reminiscent of those found in C. The following bit of code, for instance, would work in either language:

```
printf("hello, world\n");
```

Ignoring a few dollar signs (indicating that the keyword is being used to name a scalar variable), most Perl code looks quite a bit like C code:

```
$cnt=$sum=0;
for ($i=$lo; $i<$hi; $i++) {
  if ($xyz[$i] >= 0) {
    $cnt++;
    $sum += $xyz[$i];
  }
}
printf("\$cnt=%d, \$sum=%d\n", $cnt, $sum);
```

Perl adds syntax and capabilities from several other languages, however. Here are some Perl commands that might be more familiar to an awk, sed, or shell scripter:

```
$month = $months{"Jan"};     # hash (associative array)
$upper =~ tr/[a-z]/[A-Z]/;  # character substitution
$path  =~ s@/@:@g;           # regular expression
$wd    = `pwd`;              # subprocess invocation
$cfile = "$base.c";          # variable interpolation
print "hello, world\n";      # unformatted print command
```

In fact, the basic Perl language offers a wealth of features unmatched in any other popular programming language, including:

- anonymous functions, defined at run-time
- arbitrary-length strings and data structures
- associative (possibly persistent) arrays (hashes)
- automatic garbage collection
- compound data structures (such as queues of arrays)
- dynamic storage allocation
- file name globbing (wild card expansion)
- late binding of data and functions
- lists: deque, queue, indexed array, stack, and more
- coercion between numbers and strings
- object-oriented features (such as inheritance)
- regular expressions for matching and substitution
- run-time evaluation of arbitrary code
- run-time tracing and control of external data
- sparse arrays (indexed and hashed)

The core language is supplemented by a wide variety of predefined objects. Some of these are included in the base

distribution; others can be found on the Comprehensive Perl Archive Network (CPAN), an international set of FTP mirror sites. By looking around a little, you can find objects for arbitrary-precision arithmetic, CGI scripting, genetic sequence manipulation, network administration... well, you get the idea.

Perl's language features and object definitions work together in very powerful ways. A Perl-based CGI script can, quite trivially

- accept information from a user, checking it for validity
- retrieve data from selected files or a remote database
- perform arbitrary, user-specified calculations on the data
- generate a graph from the results, storing it as an image file
- output HTML to display the graph, with annotations

What's more, this entire operation can be performed in a demonstrably secure manner, using Perl's mechanisms for data-flow tracing, safe run-time evaluation of code, and more.

Macintosh Ease of Use

MacPerl can run as an application under the Finder or as a tool under MPW. Because most Mac users do not have MPW, the Finder version tends to dominate. I am told, however, that the MPW version acts much like any other MPW tool, supporting command-line options, ToolServer, etc.

The MacPerl application normally operates as an interactive development environment, displaying edit and interaction windows. When a MacPerl document (script) is double-clicked, it will either start up an edit/debug session or a batch program, depending on a user-definable preference. It also is possible to create "droplets", MacPerl scripts which support the Macintosh drag-and-drop protocol.

MacPerl has a built-in text editor, but it also works well with text editors such as Alpha and BBEdit. Both directly and by means of an Apple Script interface, MacPerl programs can emit and receive Apple events. Because of its interpretive nature, MacPerl provides a pleasant way to interface with the Toolbox (prototyping Dialog Boxes and such).

Although a Perl compiler is under development, current Perl implementations use a hybrid compiler-interpreter. The Perl source code is syntax-checked and parsed, but not turned into the host system's machine language. This approach allows Perl scripts to start up quickly and still run at a reasonable speed. A Perl script typically runs within a factor of three of the speed of a compiled C program.

Legal Issues

MacPerl (like Perl) is free software; it may be used, modified, and redistributed under the terms of the Perl Artistic License. This license, crafted by Larry Wall, is quite flexible. It allows commercial and non-commercial distribution of the program, with fairly minor restrictions. See the license text (included with the distribution) for specific details.

Please note that this definition of "free software" is far broader than that used by many Macintosh "freeware" applications. The fact that MacPerl is available in source code allows any interested party to look over its construction and modify its behavior. This has had a great deal to do with the growth and overall robustness of MacPerl and Perl.

Porting Issues

Perl comes from the Unix community, which does some things rather differently than the Mac OS community. Where

possible, MacPerl makes accomodations, providing "reasonable" behavior. In some cases, however, Unix-derived code will have to be tweaked before it can be used. Finally, some kinds of Perl applications are totally unsuited for use on a Macintosh.

The Macintosh uses a carriage return (\015), rather than Unix's line feed (\012), to separate lines of text. MacPerl accomodates this by emitting a carriage return when a newline (\n) is specified (requests for \015 are, however, taken literally). Similarly, when reading line-oriented text, MacPerl expects a carriage return, rather than a line feed.

As a result, MacPerl does not accept line feeds as delimiters in its input programs or data files. If you want to use a Unix-derived Perl script or textual data file under MacPerl, you must first convert all of its carriage returns to newlines. This is trivial, if slightly annoying.

File naming syntax must also be converted. Slashes (/) must be changed into colons (:), full path names must be modified to include disk names, etc. In addition, if the Unix code depends on "special" files (/dev/*, /proc/*, and such), some modifications will be needed.

Unix supports preemptive multitasking, allowing (nay, encouraging) programmers to invoke separate programs whenever this seems appropriate. Perl follows in this pattern, giving programmers many ways (backquotes, exec, fork, pipes, and such) to start up other processes.

MacPerl makes a few accomodations to multitasking. Certain backquoted commands (such as `pwd`) are silently emulated, handling common Unix idioms. Some multitasking may be performed if ToolServer is present. (Of course, the requested program also must be available!) Code which depends strongly on multitasking may not port smoothly, however, even with the aid of the ToolServer. If you find such a Perl script, you should expect to rework it quite a bit before you can use it on a Mac.

Some Unix-derived Perl scripts will not port readily. Code which depends on multi-tasking may not port smoothly, even with the aid of ToolServer. Code that depends on Unix-specific system calls must be modified or eliminated.

Finally, some Perl scripts rely on binary extensions which are linked into the Perl interpreter at run time. Only a few of these extensions have been ported to MacPerl, so these scripts are unlikely to work without a lot of effort (and Perl-specific knowledge).

MacPerl Resources

The MacPerl Pages http://www.ptf.com/macperl/ are a centralized source of information on MacPerl. They contain information on MacPerl, links to online resources, forms for joining the MacPerl mailing list and submitting materials (such as code samples and war stories), and more.

All modesty aside, the definitive work on MacPerl is "MacPerl: Power and Ease" (Prime Time Freeware, 1998, ISBN 1-881957-32-2, $40 MSRP). This product combines a 350+ page introductory and reference manual with a MacPerl distribution CD-ROM. For more information, visit the Prime Time Freeware web site http://www.ptf.com/.

Perl is served by two main web sites: http://www.perl.com and http://www.perl.org. Try these sites before doing any sort of search: if you can't find what you want through one or the other of these sites, it probably doesn't exist on the Internet. Though there are quite a few books in print on Perl and related subjects, the three definitive books are published by O'Reilly & Associates:

- *Learning Perl*, Second Edition, by Randal L. Schwartz, Tom Christiansen, 1996, ISBN 1-56592-284-0.
- *Programming Perl*, Second Edition, by Larry Wall, et al, 1996, ISBN 1-56592-149-6.
- *Advanced Perl Programming* by Sriram Srinivasan, 1997, ISBN 1-56592-220-4.

A 30-year veteran of the computer industry, **Rich Morin** rdm@ptf.com writes the I/Opener column for SunExpert magazine. His desktop system, Cerberus, is a three-headed Power Mac, networked to several Unixish (FreeBSD, MkLinux, Rhapsody, Solaris, and SunOS) systems.

Rich is also the president of Prime Time Freeware http://www.ptf.com/, which publishes mixed-media (book/CD-ROM) collections of freely redistributable software. PTF's Mac-specific products include MacPerl: Power and Ease and MkLinux: Microkernel Linux for the Power Macintosh. This article contains material adapted from PTF's MacPerl book.