

## NAME

perlfaq1 - General Questions About Perl (\$Revision: 1.17 \$, \$Date: 2005/01/31 15:52:15 \$)

## DESCRIPTION

This section of the FAQ answers very general, high-level questions about Perl.

### What is Perl?

Perl is a high-level programming language with an eclectic heritage written by Larry Wall and a cast of thousands. It derives from the ubiquitous C programming language and to a lesser extent from sed, awk, the Unix shell, and at least a dozen other tools and languages. Perl's process, file, and text manipulation facilities make it particularly well-suited for tasks involving quick prototyping, system utilities, software tools, system management tasks, database access, graphical programming, networking, and world wide web programming. These strengths make it especially popular with system administrators and CGI script authors, but mathematicians, geneticists, journalists, and even managers also use Perl. Maybe you should, too.

### Who supports Perl? Who develops it? Why is it free?

The original culture of the pre-populist Internet and the deeply-held beliefs of Perl's author, Larry Wall, gave rise to the free and open distribution policy of perl. Perl is supported by its users. The core, the standard Perl library, the optional modules, and the documentation you're reading now were all written by volunteers. See the personal note at the end of the README file in the perl source distribution for more details. See *perlhists* (new as of 5.005) for Perl's milestone releases.

In particular, the core development team (known as the Perl Porters) are a rag-tag band of highly altruistic individuals committed to producing better software for free than you could hope to purchase for money. You may snoop on pending developments via the archives at <http://www.xray.mpe.mpg.de/mailling-lists/perl5-porters/> and <http://archive.develooper.com/perl5-porters@perl.org/> or the news gateway [nntp://nntp.perl.org/perl.perl5.porters](mailto:nntp://nntp.perl.org/perl.perl5.porters) or its web interface at <http://nntp.perl.org/group/perl.perl5.porters>, or read the faq at <http://simon-cozens.org/writings/p5p-faq>, or you can subscribe to the mailing list by sending [perl5-porters-request@perl.org](mailto:perl5-porters-request@perl.org) a subscription request (an empty message with no subject is fine).

While the GNU project includes Perl in its distributions, there's no such thing as "GNU Perl". Perl is not produced nor maintained by the Free Software Foundation. Perl's licensing terms are also more open than GNU software's tend to be.

You can get commercial support of Perl if you wish, although for most users the informal support will more than suffice. See the answer to "Where can I buy a commercial version of perl?" for more information.

### Which version of Perl should I use?

(contributed by brian d foy)

There is often a matter of opinion and taste, and there isn't any one answer that fits anyone. In general, you want to use either the current stable release, or the stable release immediately prior to that one. Currently, those are perl5.8.x and perl5.6.x, respectively.

Beyond that, you have to consider several things and decide which is best for you.

- If things aren't broken, upgrading perl may break them (or at least issue new warnings).
- The latest versions of perl have more bug fixes.
- The Perl community is geared toward supporting the most recent releases, so you'll have an easier time finding help for those.
- Versions prior to perl5.004 had serious security problems with buffer overflows, and in some

cases have CERT advisories (for instance, <http://www.cert.org/advisories/CA-1997-17.html> ).

- The latest versions are probably the least deployed and widely tested, so you may want to wait a few months after their release and see what problems others have if you are risk averse.
- The immediate, previous releases (i.e. perl5.6.x ) are usually maintained for a while, although not at the same level as the current releases.
- No one is actively supporting perl4.x. Five years ago it was a dead camel carcass (according to this document). Now it's barely a skeleton as its whitewashed bones have fractured or eroded.
- There is no perl6.x for the next couple of years. Stay tuned, but don't worry that you'll have to change major versions of Perl soon (i.e. before 2006).
- There are really two tracks of perl development: a maintenance version and an experimental version. The maintenance versions are stable, and have an even number as the minor release (i.e. perl5.8.x, where 8 is the minor release). The experimental versions may include features that don't make it into the stable versions, and have an odd number as the minor release (i.e. perl5.9.x, where 9 is the minor release).

### What are perl4, perl5, or perl6?

(contributed by brian d foy)

In short, perl4 is the past, perl5 is the present, and perl6 is the future.

The number after perl (i.e. the 5 after perl5) is the major release of the perl interpreter as well as the version of the language. Each major version has significant differences that earlier versions cannot support.

The current major release of Perl is perl5, and was released in 1994. It can run scripts from the previous major release, perl4 (March 1991), but has significant differences. It introduced the concept of references, complex data structures, and modules. The perl5 interpreter was a complete re-write of the previous perl sources.

Perl6 is the next major version of Perl, but it's still in development in both its syntax and design. The work started in 2002 and is still ongoing. Many of the most interesting features have shown up in the latest versions of perl5, and some perl5 modules allow you to use some perl6 syntax in your programs. You can learn more about perl6 at <http://dev.perl.org/perl6/> .

See *perlhist* for a history of Perl revisions.

### What is Ponie?

At The O'Reilly Open Source Software Convention in 2003, Artur Bergman, Fotango, and The Perl Foundation announced a project to run perl5 on the Parrot virtual machine named Ponie. Ponie stands for Perl On New Internal Engine. The Perl 5.10 language implementation will be used for Ponie, and there will be no language level differences between perl5 and ponie. Ponie is not a complete rewrite of perl5.

For more details, see <http://www.poniecode.org/>

### What is perl6?

At The Second O'Reilly Open Source Software Convention, Larry Wall announced Perl6 development would begin in earnest. Perl6 was an oft used term for Chip Salzenberg's project to rewrite Perl in C++ named Topaz. However, Topaz provided valuable insights to the next version of Perl and its implementation, but was ultimately abandoned.

If you want to learn more about Perl6, or have a desire to help in the crusade to make Perl a better place then peruse the Perl6 developers page at <http://dev.perl.org/perl6/> and get involved.

Perl6 is not scheduled for release yet, and Perl5 will still be supported for quite awhile after its release. Do not wait for Perl6 to do whatever you need to do.

"We're really serious about reinventing everything that needs reinventing." --Larry Wall

### How stable is Perl?

Production releases, which incorporate bug fixes and new functionality, are widely tested before release. Since the 5.000 release, we have averaged only about one production release per year.

Larry and the Perl development team occasionally make changes to the internal core of the language, but all possible efforts are made toward backward compatibility. While not quite all perl4 scripts run flawlessly under perl5, an update to perl should nearly never invalidate a program written for an earlier version of perl (barring accidental bug fixes and the rare new keyword).

### Is Perl difficult to learn?

No, Perl is easy to start learning--and easy to keep learning. It looks like most programming languages you're likely to have experience with, so if you've ever written a C program, an awk script, a shell script, or even a BASIC program, you're already partway there.

Most tasks only require a small subset of the Perl language. One of the guiding mottos for Perl development is "there's more than one way to do it" (TMTOWTDI, sometimes pronounced "tim toady"). Perl's learning curve is therefore shallow (easy to learn) and long (there's a whole lot you can do if you really want).

Finally, because Perl is frequently (but not always, and certainly not by definition) an interpreted language, you can write your programs and test them without an intermediate compilation step, allowing you to experiment and test/debug quickly and easily. This ease of experimentation flattens the learning curve even more.

Things that make Perl easier to learn: Unix experience, almost any kind of programming experience, an understanding of regular expressions, and the ability to understand other people's code. If there's something you need to do, then it's probably already been done, and a working example is usually available for free. Don't forget the new perl modules, either. They're discussed in Part 3 of this FAQ, along with CPAN, which is discussed in Part 2.

### How does Perl compare with other languages like Java, Python, REXX, Scheme, or Tcl?

Favorably in some areas, unfavorably in others. Precisely which areas are good and bad is often a personal choice, so asking this question on Usenet runs a strong risk of starting an unproductive Holy War.

Probably the best thing to do is try to write equivalent code to do a set of tasks. These languages have their own newsgroups in which you can learn about (but hopefully not argue about) them.

Some comparison documents can be found at <http://www.perl.com/doc/FMTEYEWTK/versus/> if you really can't stop yourself.

### Can I do [task] in Perl?

Perl is flexible and extensible enough for you to use on virtually any task, from one-line file-processing tasks to large, elaborate systems. For many people, Perl serves as a great replacement for shell scripting. For others, it serves as a convenient, high-level replacement for most of what they'd program in low-level languages like C or C++. It's ultimately up to you (and possibly your management) which tasks you'll use Perl for and which you won't.

If you have a library that provides an API, you can make any component of it available as just another Perl function or variable using a Perl extension written in C or C++ and dynamically linked into your main perl interpreter. You can also go the other direction, and write your main program in C or C++, and then link in some Perl code on the fly, to create a powerful application. See *perlembed*.

That said, there will always be small, focused, special-purpose languages dedicated to a specific

problem domain that are simply more convenient for certain kinds of problems. Perl tries to be all things to all people, but nothing special to anyone. Examples of specialized languages that come to mind include prolog and matlab.

### When shouldn't I program in Perl?

When your manager forbids it--but do consider replacing them :-).

Actually, one good reason is when you already have an existing application written in another language that's all done (and done well), or you have an application language specifically designed for a certain task (e.g. prolog, make).

For various reasons, Perl is probably not well-suited for real-time embedded systems, low-level operating systems development work like device drivers or context-switching code, complex multi-threaded shared-memory applications, or extremely large applications. You'll notice that perl is not itself written in Perl.

The new, native-code compiler for Perl may eventually reduce the limitations given in the previous statement to some degree, but understand that Perl remains fundamentally a dynamically typed language, not a statically typed one. You certainly won't be chastised if you don't trust nuclear-plant or brain-surgery monitoring code to it. And Larry will sleep easier, too--Wall Street programs not withstanding. :-)

### What's the difference between "perl" and "Perl"?

One bit. Oh, you weren't talking ASCII? :-) Larry now uses "Perl" to signify the language proper and "perl" the implementation of it, i.e. the current interpreter. Hence Tom's quip that "Nothing but perl can parse Perl." You may or may not choose to follow this usage. For example, parallelism means "awk and perl" and "Python and Perl" look OK, while "awk and Perl" and "Python and perl" do not. But never write "PERL", because perl is not an acronym, apocryphal folklore and post-facto expansions notwithstanding.

### Is it a Perl program or a Perl script?

Larry doesn't really care. He says (half in jest) that "a script is what you give the actors. A program is what you give the audience."

Originally, a script was a canned sequence of normally interactive commands--that is, a chat script. Something like a UUCP or PPP chat script or an expect script fits the bill nicely, as do configuration scripts run by a program at its start up, such `.cshrc` or `.ircrc`, for example. Chat scripts were just drivers for existing programs, not stand-alone programs in their own right.

A computer scientist will correctly explain that all programs are interpreted and that the only question is at what level. But if you ask this question of someone who isn't a computer scientist, they might tell you that a *program* has been compiled to physical machine code once and can then be run multiple times, whereas a *script* must be translated by a program each time it's used.

Perl programs are (usually) neither strictly compiled nor strictly interpreted. They can be compiled to a byte-code form (something of a Perl virtual machine) or to completely different languages, like C or assembly language. You can't tell just by looking at it whether the source is destined for a pure interpreter, a parse-tree interpreter, a byte-code interpreter, or a native-code compiler, so it's hard to give a definitive answer here.

Now that "script" and "scripting" are terms that have been seized by unscrupulous or unknowing marketers for their own nefarious purposes, they have begun to take on strange and often pejorative meanings, like "non serious" or "not real programming". Consequently, some Perl programmers prefer to avoid them altogether.

### What is a JAPH?

These are the "just another perl hacker" signatures that some people sign their postings with. Randal Schwartz made these famous. About 100 of the earlier ones are available from

<http://www.cpan.org/misc/japh> .

### Where can I get a list of Larry Wall witticisms?

Over a hundred quips by Larry, from postings of his or source code, can be found at <http://www.cpan.org/misc/lwall-quotes.txt.gz> .

### How can I convince my sysadmin/supervisor/employees to use version 5/5.6.1/Perl instead of some other language?

If your manager or employees are wary of unsupported software, or software which doesn't officially ship with your operating system, you might try to appeal to their self-interest. If programmers can be more productive using and utilizing Perl constructs, functionality, simplicity, and power, then the typical manager/supervisor/employee may be persuaded. Regarding using Perl in general, it's also sometimes helpful to point out that delivery times may be reduced using Perl compared to other languages.

If you have a project which has a bottleneck, especially in terms of translation or testing, Perl almost certainly will provide a viable, quick solution. In conjunction with any persuasion effort, you should not fail to point out that Perl is used, quite extensively, and with extremely reliable and valuable results, at many large computer software and hardware companies throughout the world. In fact, many Unix vendors now ship Perl by default. Support is usually just a news-posting away, if you can't find the answer in the *comprehensive* documentation, including this FAQ.

See <http://www.perl.org/advocacy/> for more information.

If you face reluctance to upgrading from an older version of perl, then point out that version 4 is utterly unmaintained and unsupported by the Perl Development Team. Another big sell for Perl5 is the large number of modules and extensions which greatly reduce development time for any given task. Also mention that the difference between version 4 and version 5 of Perl is like the difference between awk and C++. (Well, OK, maybe it's not quite that distinct, but you get the idea.) If you want support and a reasonable guarantee that what you're developing will continue to work in the future, then you have to run the supported version. As of December 2003 that means running either 5.8.2 (released in November 2003), or one of the older releases like 5.6.2 (also released in November 2003; a maintenance release to let perl 5.6 compile on newer systems as 5.6.1 was released in April 2001) or 5.005\_03 (released in March 1999), although 5.004\_05 isn't that bad if you **absolutely** need such an old version (released in April 1999) for stability reasons. Anything older than 5.004\_05 shouldn't be used.

Of particular note is the massive bug hunt for buffer overflow problems that went into the 5.004 release. All releases prior to that, including perl4, are considered insecure and should be upgraded as soon as possible.

In August 2000 in all Linux distributions a new security problem was found in the optional 'suidperl' (not built or installed by default) in all the Perl branches 5.6, 5.005, and 5.004, see <http://www.cpan.org/src/5.0/sperl-2000-08-05/> Perl maintenance releases 5.6.1 and 5.8.0 have this security hole closed. Most, if not all, Linux distribution have patches for this vulnerability available, see <http://www.linuxsecurity.com/advisories/> , but the most recommendable way is to upgrade to at least Perl 5.6.1.

## AUTHOR AND COPYRIGHT

Copyright (c) 1997-2005 Tom Christiansen, Nathan Torkington, and other authors as noted. All rights reserved.

This documentation is free; you can redistribute it and/or modify it under the same terms as Perl itself.

Irrespective of its distribution, all code examples here are in the public domain. You are permitted and encouraged to use this code and any derivatives thereof in your own programs for fun or for profit as you see fit. A simple comment in the code giving credit to the FAQ would be courteous but is not required.

