WIKIPEDIA

Perl

Perl is a family of two high-level, general-purpose, interpreted, dynamic programming languages. "Perl" refers to Perl 5, but from 2000 to 2019 it also referred to its redesigned "sister language", Perl 6, before the latter's name was officially changed to Raku in October 2019.[9][10]

Though Perl is not officially an acronym,[11] there are various backronyms in use, including "Practical Extraction and Reporting Language".[12] Perl was developed by Larry Wall in 1987 as a general-purpose Unix scripting language to make report processing easier.[13] Since then, it has undergone many changes and revisions. Raku, which began as a redesign of Perl 5 in 2000, eventually evolved into a separate language. Both languages continue to be developed independently by different development teams and liberally borrow ideas from each other.

The Perl languages borrow features from other programming languages including C, shell script (sh), AWK, and sed;[14] They provide text processing facilities without the arbitrary data-length limits of many contemporary Unix command line tools.[15] Perl 5 gained widespread popularity in the late 1990s as a CGI scripting language, in part due to its unsurpassed regular expression and string parsing abilities.[16][17][18][19]

In addition to CGI, Perl 5 is used for system administration, network programming, finance, bioinformatics, and other applications, such as for GUIs. It has been nicknamed "the Swiss Army chainsaw of scripting languages" because of its flexibility and power,[20] and also its ugliness.[21] In 1998, it was also referred to as the "duct tape that holds the Internet together," in reference to both its ubiquitous use as a glue language and its perceived inelegance.[22]

Perl is a highly expressive programming language: source code for a given algorithm can be short and highly compressible.[23][24]

Contents

Perl

Paradigm	Multi-paradigm: functional, imperative, object-oriented (class-based), reflective
Designed by	Larry Wall
Developer	Larry Wall
First appeared	February 1, 1988[1]
Stable release	5.34.0[2] / 20 May 2021 5.32.1[3] / 23 January 2021
Preview release	5.35.5[4] / October 21, 2021
Typing discipline	Dynamic
Implementation language	C
OS	Cross-platform
License	Artistic License 1.0[5][6] or GNU General Public License[7]
Filename extensions	.plx, .pl, .pm, .xs, .t, .pod, .cgi
Website	www.perl.org (https://www.perl.org)
Influenced by	
AWK, BASIC, C, C++, Lisp, sed,	

Unix shell[8]
Influenced
CoffeeScript, Groovy, JavaScript, Julia, LPC, PHP, Python, Raku, Ruby, PowerShell
Perl Programming at Wikibooks

Name

Perl was originally named "Pearl". Wall wanted to give the language a short name with positive connotations. Wall discovered the existing PEARL programming language before Perl's official release and changed the spelling of the name.[25]

When referring to the language, the name is capitalized: *Perl*. When referring to the program itself, the name is uncapitalized (*perl*) because most Unix-like file systems are case-sensitive. Before the release of the first edition of *Programming Perl*, it was common to refer to the language as *perl*. Randal L. Schwartz, however, capitalized the language's name in the book to make it stand out better when typeset. This case distinction was subsequently documented as canonical.[26]

The name is occasionally expanded as a backronym: *Practical Extraction and Report Language*[27] and Wall's own *Pathologically Eclectic Rubbish Lister* which is in the manual page for perl.[28]

History

Early versions

Larry Wall began work on Perl in 1987, while working as a programmer at Unisys,[15] and version 1.0 was released to the comp.sources.unix newsgroup on February 1, 1988.[1] The language expanded rapidly over the next few years.

Perl 2, released in 1988, featured a better regular expression engine. Perl 3, released in 1989, added support for binary data streams.

Originally, the only documentation for Perl was a single lengthy man page. In 1991, *Programming Perl*, known to many Perl programmers as the "Camel Book" because of its cover, was published and became the *de facto* reference for the language. At the same time, the Perl version number was bumped to 4, not to mark a major change in the language but to identify the version that was well documented by the book.

Early Perl 5

Perl 4 went through a series of maintenance releases, culminating in Perl 4.036 in 1993, whereupon Wall abandoned Perl 4 to begin work on Perl 5. Initial design of Perl 5 continued into 1994. The *perl5-porters* mailing list was established in May 1994 to coordinate work on porting Perl 5 to different platforms. It remains the primary forum for development, maintenance, and porting of Perl 5.[29]

Perl 5.000 was released on October 17, 1994.[30] It was a nearly complete rewrite of the interpreter, and it added many new features to the language, including objects, references, lexical (my) variables, and modules. Importantly, modules provided a mechanism for extending the language without modifying the interpreter. This allowed the core interpreter to stabilize, even as it enabled ordinary Perl programmers to add new language features. Perl 5 has been in active development since then.

Perl 5.001 was released on March 13, 1995. Perl 5.002 was released on February 29, 1996 with the new prototypes feature. This allowed module authors to make subroutines that behaved like Perl builtins. Perl 5.003 was released June 25, 1996, as a security release.

One of the most important events in Perl 5 history took place outside of the language proper and was a consequence of its module support. On October 26, 1995, the Comprehensive Perl Archive Network (CPAN) was established as a repository for the Perl language and Perl modules; as of May 2017, it carries over 185,178 modules in 35,190 distributions, written by more than 13,071 authors, and is mirrored worldwide at more than 245 locations.[31]

Perl 5.004 was released on May 15, 1997, and included, among other things, the UNIVERSAL package, giving Perl a base object from which all classes were automatically derived and the ability to require versions of modules. Another significant development was the inclusion of the CGI.pm module,[32] which contributed to Perl's popularity as a CGI scripting language.[33]

Perl 5.004 added support for Microsoft Windows, Plan 9, QNX, and AmigaOS.[32]

Perl 5.005 was released on July 22, 1998. This release included several enhancements to the regex engine, new hooks into the backend through the `B::*` modules, the `qr//` regex quote operator, a large selection of other new core modules, and added support for several more operating systems, including BeOS.[34]

2000–2020

Major version[35]	Latest update[36]
5.4	1999-04-29
5.5	1999-03-29
5.6	2003-11-15
5.8	2008-12-14
5.10	2009-08-23
5.12	2012-11-10
5.14	2013-03-10
5.16	2013-03-11
5.18	2014-10-02
5.20	2015-09-12
5.22	2017-07-15
5.24	2018-04-14
5.26	2018-11-29
5.28	2020-06-01
5.30	2020-06-01
5.32	2021-01-23
5.34	2021-05-20
5.35	2021-05-21
5.36	2022-05-20
7.0	2022?

Old version
Older version, still maintained
Current stable version
Latest preview version
Future release

Perl 5.6 was released on March 22, 2000. Major changes included 64-bit support, Unicode string representation, support for files over 2 GiB, and the "our" keyword.[37][38] When developing Perl 5.6, the decision was made to switch the versioning scheme to one more similar to other open source projects; after 5.005_63, the next version became 5.5.640, with plans for development versions to have odd numbers and stable versions to have even numbers.

In 2000, Wall put forth a call for suggestions for a new version of Perl from the community. The process resulted in 361 RFC (request for comments) documents that were to be used in guiding development of Perl 6. In 2001,[39] work began on the "Apocalypses" for Perl 6, a series of documents meant to summarize the change requests and present the design of the next generation of Perl. They were presented as a digest of the RFCs, rather than a formal document. At this point, Perl 6 existed only as a description of a language.

Perl 5.8 was first released on July 18, 2002, and had nearly yearly updates since then. Perl 5.8 improved Unicode support, added a new I/O implementation, added a new thread implementation, improved numeric accuracy, and added several new modules.[40] As of 2013 this version still remains the most popular version of Perl and is used by Red Hat 5, Suse 10, Solaris 10, HP-UX 11.31 and AIX 5.

In 2004, work began on the "Synopses" – documents that originally summarized the Apocalypses, but which became the specification for the Perl 6 language. In February 2005, Audrey Tang began work on Pugs, a Perl 6 interpreter written in Haskell.[41] This was the first concerted effort towards making Perl 6 a reality. This effort stalled in 2006.[42]

PONIE is an acronym for Perl On New Internal Engine. The PONIE Project existed from 2003 until 2006 and was to be a bridge between Perl 5 and Perl 6. It was an effort to rewrite the Perl 5 interpreter to run on Parrot, the Perl 6 virtual machine. The goal was to ensure the future of the millions of lines of Perl 5 code at thousands of companies around the world.[43] The PONIE project ended in 2006 and is no longer being actively developed. Some of the improvements made to the Perl 5 interpreter as part of PONIE were folded into that project.[44]

On December 18, 2007, the 20th anniversary of Perl 1.0, Perl 5.10.0 was released. Perl 5.10.0 included notable new features, which brought it closer to Perl 6. These included a switch statement (called "given"/"when"), regular expressions updates, and the '*smart match operator* (~~).[45][46] Around this same time, development began in earnest on another implementation of Perl 6 known as Rakudo Perl, developed in tandem with the Parrot virtual machine. As of November 2009, Rakudo Perl has had regular monthly releases and now is the most complete implementation of Perl 6.

A major change in the development process of Perl 5 occurred with Perl 5.11; the development community has switched to a monthly release cycle of development releases, with a yearly schedule of stable releases. By that plan, bugfix point releases will follow the stable releases every

three months.

On April 12, 2010, Perl 5.12.0 was released. Notable core enhancements include new `package NAME VERSION` syntax, the Yada Yada operator (intended to mark placeholder code that is not yet implemented), implicit strictures, full Y2038 compliance, regex conversion overloading, DTrace support, and Unicode 5.2.[47] On January 21, 2011, Perl 5.12.3 was released; it contains updated modules and some documentation changes.[48] Version 5.12.4 was released on June 20, 2011. The latest version of that branch, 5.12.5, was released on November 10, 2012.

On May 14, 2011, Perl 5.14 was released with JSON support built-in.[49]

On May 20, 2012, Perl 5.16 was released. Notable new features include the ability to specify a given version of Perl that one wishes to emulate, allowing users to upgrade their version of Perl, but still run old scripts that would normally be incompatible.[50] Perl 5.16 also updates the core to support Unicode 6.1.[50]

On May 18, 2013, Perl 5.18 was released. Notable new features include the new dtrace hooks, lexical subs, more CORE:: subs, overhaul of the hash for security reasons, support for Unicode 6.2.[51]

On May 27, 2014, Perl 5.20 was released. Notable new features include subroutine signatures, hash slices/new slice syntax, postfix dereferencing (experimental), Unicode 6.3, rand() using consistent random number generator.[52]

Some observers credit the release of Perl 5.10 with the start of the Modern Perl movement.[53] In particular, this phrase describes a style of development that embraces the use of the CPAN, takes advantage of recent developments in the language, and is rigorous about creating high quality code.[54] While the book "Modern Perl"[55] may be the most visible standard-bearer of this idea, other groups such as the Enlightened Perl Organization[56] have taken up the cause.

In late 2012 and 2013, several projects for alternative implementations for Perl 5 started: Perl5 in Perl6 by the Rakudo Perl team,[57] *moe* by Stevan Little and friends,[58] *p2*[59] by the Perl11 team under Reini Urban, *gperl* by goccy,[60] and *rperl,* a Kickstarter project led by Will Braswell and affiliated with the Perll11 project.[61]

2020 onwards

In June 2020, Perl 7 was announced as the successor to Perl 5.[62] Perl 7 was to initially be based on Perl 5.32 with a release expected in first half of 2021, and release candidates sooner.[63] This plan was revised in May 2021, without any release timeframe or version of Perl 5 for use as a baseline specified.[64] When Perl 7 is released, Perl 5 will go into long term maintenance. Supported Perl 5 versions however will continue to get important security and bug fixes.[65]

Symbols

Camel

Programming Perl, published by O'Reilly Media, features a picture of a dromedary camel on the cover and is commonly called the "Camel Book".[66] This image has become an unofficial symbol of Perl as well as a general hacker emblem, appearing on T-shirts and other clothing items.

O'Reilly owns the image as a trademark but licenses it for non-commercial use, requiring only an acknowledgement and a link to www.perl.com. Licensing for commercial use is decided on a case-by-case basis.[67] O'Reilly also provides "Programming Republic of Perl" logos for non-commercial sites and "Powered by Perl" buttons for any site that uses Perl.[67]

The Camel symbol used by O'Reilly Media

Onion

The Perl Foundation owns an alternative symbol, an onion, which it licenses to its subsidiaries, Perl Mongers, PerlMonks, Perl.org, and others.[68] The symbol is a visual pun on pearl onion.[69]

The onion logo used by The Perl Foundation

Raptor

Sebastian Riedel, the creator of Mojolicious, created a logo depicting a raptor dinosaur, which is available under a CC-SA License, Version 4.0.[70] The analogue of the raptor comes from a series of talks given by Matt S Trout beginning in 2010.[71]

Overview

Alternative Perl 5 Logo

According to Wall, Perl has two slogans. The first is "There's more than one way to do it," commonly known as TMTOWTDI. The second slogan is "Easy things should be easy and hard things should be possible".[15]

Features

The overall structure of Perl derives broadly from C. Perl is procedural in nature, with variables, expressions, assignment statements, brace-delimited blocks, control structures, and subroutines.[72]

Perl also takes features from shell programming. All variables are marked with leading sigils, which allow variables to be interpolated directly into strings. However, unlike the shell, Perl uses sigils on all accesses to variables, and unlike most other programming languages that use sigils, the sigil doesn't denote the type of the variable but the type of the expression. So for example, while an array is denoted by the sigil "@" (for example `@arrayname`), an individual member of the array is denoted by the scalar sigil "$" (for example `$arrayname[3]`). Perl also has many built-in functions that provide tools often used in shell programming (although many of these tools are implemented by programs external to the shell) such as sorting, and calling operating system facilities.

Perl takes hashes ("associative arrays") from AWK and regular expressions from sed. These simplify many parsing, text-handling, and data-management tasks. Shared with Lisp is the implicit return of the last value in a block, and all statements are also expressions which can be used in larger expressions themselves.

Perl 5 added features that support complex data structures, first-class functions (that is, closures as values), and an object-oriented programming model. These include references, packages, class-based method dispatch, and lexically scoped variables, along with compiler directives (for

example, the `strict` pragma). A major additional feature introduced with Perl 5 was the ability to package code as reusable modules. Wall later stated that "The whole intent of Perl 5's module system was to encourage the growth of Perl culture rather than the Perl core."[73]

All versions of Perl do automatic data-typing and automatic memory management. The interpreter knows the type and storage requirements of every data object in the program; it allocates and frees storage for them as necessary using reference counting (so it cannot deallocate circular data structures without manual intervention). Legal type conversions — for example, conversions from number to string — are done automatically at run time; illegal type conversions are fatal errors.

Design

The design of Perl can be understood as a response to three broad trends in the computer industry: falling hardware costs, rising labor costs, and improvements in compiler technology. Many earlier computer languages, such as Fortran and C, aimed to make efficient use of expensive computer hardware. In contrast, Perl was designed so that computer programmers could write programs more quickly and easily.

Perl has many features that ease the task of the programmer at the expense of greater CPU and memory requirements. These include automatic memory management; dynamic typing; strings, lists, and hashes; regular expressions; introspection; and an `eval()` function. Perl follows the theory of "no built-in limits,"[66] an idea similar to the Zero One Infinity rule.

Wall was trained as a linguist, and the design of Perl is very much informed by linguistic principles. Examples include Huffman coding (common constructions should be short), good end-weighting (the important information should come first), and a large collection of language primitives. Perl favors language constructs that are concise and natural for humans to write, even where they complicate the Perl interpreter.[74]

Perl's syntax reflects the idea that "things that are different should look different."[75] For example, scalars, arrays, and hashes have different leading sigils. Array indices and hash keys use different kinds of braces. Strings and regular expressions have different standard delimiters. This approach can be contrasted with a language such as Lisp, where the same basic syntax, composed of simple and universal symbolic expressions, is used for all purposes.

Perl does not enforce any particular programming paradigm (procedural, object-oriented, functional, or others) or even require the programmer to choose among them.

There is a broad practical bent to both the Perl language and the community and culture that surround it. The preface to *Programming Perl* begins: "Perl is a language for getting your job done."[15] One consequence of this is that Perl is not a tidy language. It includes many features, tolerates exceptions to its rules, and employs heuristics to resolve syntactical ambiguities. Because of the forgiving nature of the compiler, bugs can sometimes be hard to find. Perl's function documentation remarks on the variant behavior of built-in functions in list and scalar contexts by saying, "In general, they do what you want, unless you want consistency."[76]

No written specification or standard for the Perl language exists for Perl versions through Perl 5, and there are no plans to create one for the current version of Perl. There has been only one implementation of the interpreter, and the language has evolved along with it. That interpreter, together with its functional tests, stands as a *de facto* specification of the language. Perl 6, however, started with a specification,[77] and several projects[78] aim to implement some or all of the specification.

Applications

Perl has many and varied applications, compounded by the availability of many standard and third-party modules.

Perl has chiefly been used to write CGI scripts: large projects written in Perl include cPanel, Slash, Bugzilla, RT, TWiki, and Movable Type; high-traffic websites that use Perl extensively include Priceline.com, Craigslist,[79] IMDb,[80] LiveJournal, DuckDuckGo,[81][82] Slashdot and Ticketmaster. It is also an optional component of the popular LAMP technology stack for Web development, in lieu of PHP or Python. Perl is used extensively as a system programming language in the Debian Linux distribution.[83]

Perl is often used as a glue language, tying together systems and interfaces that were not specifically designed to interoperate, and for "data munging,"[84] that is, converting or processing large amounts of data for tasks such as creating reports. In fact, these strengths are intimately linked. The combination makes Perl a popular all-purpose language for system administrators, particularly because short programs, often called "one-liner programs," can be entered and run on a single command line.

Perl code can be made portable across Windows and Unix; such code is often used by suppliers of software (both COTS and bespoke) to simplify packaging and maintenance of software build- and deployment-scripts.

Perl/Tk and wxPerl are commonly used to add graphical user interfaces to Perl scripts.

Implementation

Perl is implemented as a core interpreter, written in C, together with a large collection of modules, written in Perl and C. As of 2010, the interpreter is 150,000 lines of C code and compiles to a 1 MB executable on typical machine architectures. Alternatively, the interpreter can be compiled to a link library and embedded in other programs. There are nearly 500 modules in the distribution, comprising 200,000 lines of Perl and an additional 350,000 lines of C code (much of the C code in the modules consists of character encoding tables).

The interpreter has an object-oriented architecture. All of the elements of the Perl language—scalars, arrays, hashes, coderefs, file handles—are represented in the interpreter by C structs. Operations on these structs are defined by a large collection of macros, typedefs, and functions; these constitute the Perl C API. The Perl API can be bewildering to the uninitiated, but its entry points follow a consistent naming scheme, which provides guidance to those who use it.

The life of a Perl interpreter divides broadly into a compile phase and a run phase.[85] In Perl, the **phases** are the major stages in the interpreter's life-cycle. Each interpreter goes through each phase only once, and the phases follow in a fixed sequence.

Most of what happens in Perl's compile phase is compilation, and most of what happens in Perl's run phase is execution, but there are significant exceptions. Perl makes important use of its capability to execute Perl code during the compile phase. Perl will also delay compilation into the run phase. The terms that indicate the kind of processing that is actually occurring at any moment are **compile time** and **run time**. Perl is in compile time at most points during the compile phase, but compile time may also be entered during the run phase. The compile time for code in a string argument passed to the `eval` built-in occurs during the run phase. Perl is often in run time during the compile phase and spends most of the run phase in run time. Code in `BEGIN` blocks executes at run time but in the compile phase.

At compile time, the interpreter parses Perl code into a syntax tree. At run time, it executes the program by walking the tree. Text is parsed only once, and the syntax tree is subject to optimization before it is executed, so that execution is relatively efficient. Compile-time optimizations on the syntax tree include constant folding and context propagation, but peephole optimization is also performed.

Perl has a Turing-complete grammar because parsing can be affected by run-time code executed during the compile phase.[86] Therefore, Perl cannot be parsed by a straight Lex/Yacc lexer/parser combination. Instead, the interpreter implements its own lexer, which coordinates with a modified GNU bison parser to resolve ambiguities in the language.

It is often said that "Only perl can parse Perl,"[87] meaning that only the Perl interpreter (`perl`) can parse the Perl language (*Perl*), but even this is not, in general, true. Because the Perl interpreter can simulate a Turing machine during its compile phase, it would need to decide the halting problem in order to complete parsing in every case. It is a longstanding result that the halting problem is undecidable, and therefore not even perl can always parse Perl. Perl makes the unusual choice of giving the user access to its full programming power in its own compile phase. The cost in terms of theoretical purity is high, but practical inconvenience seems to be rare.

Other programs that undertake to parse Perl, such as source-code analyzers and auto-indenters, have to contend not only with ambiguous syntactic constructs but also with the undecidability of Perl parsing in the general case. Adam Kennedy's PPI project focused on parsing Perl code as a document (retaining its integrity as a document), instead of parsing Perl as executable code (that not even Perl itself can always do). It was Kennedy who first conjectured that "parsing Perl suffers from the 'halting problem',"[88] which was later proved.[89]

Perl is distributed with over 250,000 functional tests for core Perl language and over 250,000 functional tests for core modules. These run as part of the normal build process and extensively exercise the interpreter and its core modules. Perl developers rely on the functional tests to ensure that changes to the interpreter do not introduce software bugs; additionally, Perl users who see that the interpreter passes its functional tests on their system can have a high degree of confidence that it is working properly.

Availability

Perl is dual licensed under both the Artistic License 1.0[5][6] and the GNU General Public License.[7] Distributions are available for most operating systems. It is particularly prevalent on Unix and Unix-like systems, but it has been ported to most modern (and many obsolete) platforms. With only six reported exceptions, Perl can be compiled from source code on all POSIX-compliant, or otherwise-Unix-compatible, platforms.[90]

Because of unusual changes required for the classic Mac OS environment, a special port called MacPerl was shipped independently.[91]

The Comprehensive Perl Archive Network carries a complete list of supported platforms with links to the distributions available on each.[92] CPAN is also the source for publicly available Perl modules that are not part of the core Perl distribution.

Windows

Users of Microsoft Windows typically install one of the native binary distributions of Perl for Win32, most commonly Strawberry Perl or ActivePerl. Compiling Perl from source code under Windows is possible, but most installations lack the requisite C compiler and build tools. This also

makes it difficult to install modules from the CPAN, particularly those that are partially written in C.

ActivePerl is a closed-source distribution from ActiveState that has regular releases that track the core Perl releases.[93] The distribution previously included the Perl package manager (PPM),[94] a popular tool for installing, removing, upgrading, and managing the use of common Perl modules; however, this tool was discontinued as of ActivePerl 5.28.[95] Included also is PerlScript, a Windows Script Host (WSH) engine implementing the Perl language. Visual Perl is an ActiveState tool that adds Perl to the Visual Studio .NET development suite. A VBScript-to-Perl converter, as well as a Perl compiler for Windows, and converters of awk and sed to Perl have also been produced by this company and included on the *ActiveState CD for Windows*, which includes all of their distributions plus the Komodo IDE and all but the first on the Unix/Linux/Posix variant thereof in 2002 and subsequently.[96]

Strawberry Perl is an open-source distribution for Windows. It has had regular, quarterly releases since January 2008, including new modules as feedback and requests come in. Strawberry Perl aims to be able to install modules like standard Perl distributions on other platforms, including compiling XS modules.

The Cygwin emulation layer is another way of running Perl under Windows. Cygwin provides a Unix-like environment on Windows, and both Perl and CPAN are available as standard pre-compiled packages in the Cygwin setup program. Since Cygwin also includes gcc, compiling Perl from source is also possible.

A perl executable is included in several Windows Resource kits in the directory with other scripting tools.

Implementations of Perl come with the MKS Toolkit, Interix (the base of earlier implementations of Windows Services for Unix), and UWIN.

Database interfaces

Perl's text-handling capabilities can be used for generating SQL queries; arrays, hashes, and automatic memory management make it easy to collect and process the returned data. For example, in Tim Bunce's Perl DBI application programming interface (API), the arguments to the API can be the text of SQL queries; thus it is possible to program in multiple languages at the same time (e.g., for generating a Web page using HTML, JavaScript, and SQL in a here document). The use of Perl variable interpolation to programmatically customize each of the SQL queries, and the specification of Perl arrays or hashes as the structures to programmatically hold the resulting data sets from each SQL query, allows a high-level mechanism for handling large amounts of data for post-processing by a Perl subprogram.[97] In early versions of Perl, database interfaces were created by relinking the interpreter with a client-side database library. This was sufficiently difficult that it was done for only a few of the most-important and most widely used databases, and it restricted the resulting `perl` executable to using just one database interface at a time.[98]

In Perl 5, database interfaces are implemented by Perl DBI modules. The DBI (Database Interface) module presents a single, database-independent interface to Perl applications, while the DBD (Database Driver) modules handle the details of accessing some 50 different databases; there are DBD drivers for most ANSI SQL databases.[99]

DBI provides caching for database handles and queries, which can greatly improve performance in long-lived execution environments such as mod perl,[100] helping high-volume systems avert load spikes as in the Slashdot effect.[101]

In modern Perl applications, especially those written using web frameworks such as Catalyst, the DBI module is often used indirectly via object-relational mappers such as DBIx::Class, Class::DBI[102] or Rose::DB::Object[103] that generate SQL queries and handle data transparently to the application author.[104]

Comparative performance

The Computer Language Benchmarks Game compares the performance of implementations of typical programming problems in several programming languages.[105] The submitted Perl implementations typically perform toward the high end of the memory-usage spectrum and give varied speed results. Perl's performance in the benchmarks game is typical for interpreted languages.[106]

Large Perl programs start more slowly than similar programs in compiled languages because perl has to compile the source every time it runs. In a talk at the YAPC::Europe 2005 conference and subsequent article "A Timely Start," Jean-Louis Leroy found that his Perl programs took much longer to run than expected because the perl interpreter spent significant time finding modules within his over-large include path.[107] Unlike Java, Python, and Ruby, Perl has only experimental support for pre-compiling.[108] Therefore, Perl programs pay this overhead penalty on every execution. The run phase of typical programs is long enough that amortized startup time is not substantial, but benchmarks that measure very short execution times are likely to be skewed due to this overhead.[109]

A number of tools have been introduced to improve this situation. The first such tool was Apache's mod perl, which sought to address one of the most-common reasons that small Perl programs were invoked rapidly: CGI Web development. ActivePerl, via Microsoft ISAPI, provides similar performance improvements.[110]

Once Perl code is compiled, there is additional overhead during the execution phase that typically isn't present for programs written in compiled languages such as C or C++. Examples of such overhead include bytecode interpretation, reference-counting memory management, and dynamic type-checking.[111]

Optimizing

The most critical routines can be written in other languages (such as C), which can be connected to Perl via simple Inline modules or the more complex, but flexible, XS mechanism.[112]

Perl 5

Perl 5, the language usually referred to as "Perl", continues to be actively developed. Perl 5.12.0 was released in April 2010 with some new features influenced by the design of Perl 6,[47][113] followed by Perl 5.14.1 (released on June 17, 2011), Perl 5.16.1 (released on August 9, 2012.[114]), and Perl 5.18.0 (released on May 18, 2013). Perl 5 development versions are released on a monthly basis, with major releases coming out once per year.[115]

The relative proportion of Internet searches for "Perl programming", as compared with similar searches for other programming languages, steadily declined from about 10% in 2005 to about 2% in 2011, to about 0.7% in 2020.[116]

Raku (Perl 6)

At the 2000 Perl Conference, Jon Orwant made a case for a major new language-initiative.[118] This led to a decision to begin work on a redesign of the language, to be called Perl 6. Proposals for new language features were solicited from the Perl community at large, which submitted more than 300 RFCs.[119]

Camelia, the logo for the Perl 6 project[117]

Wall spent the next few years digesting the RFCs and synthesizing them into a coherent framework for Perl 6. He presented his design for Perl 6 in a series of documents called "apocalypses" – numbered to correspond to chapters in *Programming Perl*. As of January 2011, the developing specification of Perl 6 was encapsulated in design documents called Synopses – numbered to correspond to Apocalypses.[120]

Thesis work by Bradley M. Kuhn, overseen by Wall, considered the possible use of the Java virtual machine as a runtime for Perl.[121] Kuhn's thesis showed this approach to be problematic. In 2001, it was decided that Perl 6 would run on a cross-language virtual machine called Parrot. This will mean that other languages targeting the Parrot will gain native access to CPAN, allowing some level of cross-language development.

In 2005, Audrey Tang created the Pugs project, an implementation of Perl 6 in Haskell. This acted as, and continues to act as, a test platform for the Perl 6 language (separate from the development of the actual implementation) – allowing the language designers to explore. The Pugs project spawned an active Perl/Haskell cross-language community centered around the Libera Chat #raku IRC channel. Many functional programming influences were absorbed by the Perl 6 design team.[122]

In 2012, Perl 6 development was centered primarily on two compilers:[123]

1. Rakudo, an implementation running on the Parrot virtual machine and the Java virtual machine.[124]
2. Niecza, which targets the Common Language Runtime.

In 2013, MoarVM ("Metamodel On A Runtime"), a C language-based virtual machine designed primarily for Rakudo was announced.[125]

In October 2019, Perl 6 was renamed to Raku.[126]

As of 2017 only the Rakudo implementation and MoarVM are under active development, and other virtual machines, such as the Java Virtual Machine and JavaScript, are supported.[127]

Perl 7

Perl 7 was announced on 24 June 2020 at "The Perl Conference in the Cloud" as the successor to Perl 5.[128][129] Based on Perl 5.32, Perl 7 is designed to be backwards compatible with modern Perl 5 code; Perl 5 code, without boilerplate (pragma) header needs adding `use compat::perl5;` to stay compatible, but modern code can drop some of the boilerplate.

Perl community

Perl's culture and community has developed alongside the language itself. Usenet was the first public venue in which Perl was introduced, but over the course of its evolution, Perl's community was shaped by the growth of broadening Internet-based services including the introduction of the

World Wide Web. The community that surrounds Perl was, in fact, the topic of Wall's first "State of the Onion" talk.[130]

State of the Onion

State of the Onion is the name for Wall's yearly keynote-style summaries on the progress of Perl and its community. They are characterized by his hallmark humor, employing references to Perl's culture, the wider hacker culture, Wall's linguistic background, sometimes his family life, and occasionally even his Christian background.[131]

Each talk is first given at various Perl conferences and is eventually also published online.

Perl pastimes

JAPHs
: In email, Usenet, and message board postings, "Just another Perl hacker" (JAPH) programs are a common trend, originated by Randal L. Schwartz, one of the earliest professional Perl trainers.[132] In the parlance of Perl culture, Perl programmers are known as Perl hackers, and from this derives the practice of writing short programs to print out the phrase "Just another Perl hacker". In the spirit of the original concept, these programs are moderately obfuscated and short enough to fit into the signature of an email or Usenet message. The "canonical" JAPH as developed by Schwartz includes the comma at the end, although this is often omitted.[133]

Perl golf
: Perl "golf" is the pastime of reducing the number of characters (key "strokes") used in a Perl program to the bare minimum, much in the same way that golf players seek to take as few shots as possible in a round. The phrase's first use[134] emphasized the difference between pedestrian code meant to teach a newcomer and terse hacks likely to amuse experienced Perl programmers, an example of the latter being JAPHs that were already used in signatures in Usenet postings and elsewhere. Similar stunts had been an unnamed pastime in the language APL in previous decades. The use of Perl to write a program that performed RSA encryption prompted a widespread and practical interest in this pastime.[135] In subsequent years, the term "code golf" has been applied to the pastime in other languages.[136] A Perl Golf Apocalypse was held at Perl Conference 4.0 in Monterey, California in July 2000.

Obfuscation
: As with C, obfuscated code competitions were a well known pastime in the late 1990s. The Obfuscated Perl Contest was a competition held by The Perl Journal from 1996 to 2000 that made an arch virtue of Perl's syntactic flexibility. Awards were given for categories such as "most powerful"—programs that made efficient use of space—and "best four-line signature" for programs that fit into four lines of 76 characters in the style of a Usenet signature block.[137]

Poetry
: Perl poetry is the practice of writing poems that can be compiled as legal Perl code, for example the piece known as Black Perl. Perl poetry is made possible by the large number of English words that are used in the Perl language. New poems are regularly submitted to the community at PerlMonks.[138]

Perl on IRC

There are a number of IRC channels that offer support for the language and some modules.

IRC Network	Channels
irc.libera.chat	#perl #raku
irc.perl.org	#moose #poe #catalyst #dbix-class #perl-help #distzilla #epo #corehackers #sdl #win32 #toolchain #padre #dancer
irc.slashnet.org	#perlmonks
irc.oftc.net	#perl #debian-perl (packaging Perl modules for Debian)
irc.efnet.net	#perlhelp
irc.rizon.net	#perl

CPAN Acme

There are also many examples of code written purely for entertainment on the CPAN. `Lingua::Romana::Perligata`, for example, allows writing programs in Latin.[139] Upon execution of such a program, the module translates its source code into regular Perl and runs it.

The Perl community has set aside the "Acme" namespace for modules that are fun in nature (but its scope has widened to include exploratory or experimental code or any other module that is not meant to ever be used in production). Some of the Acme modules are deliberately implemented in amusing ways. This includes `Acme::Bleach`, one of the first modules in the `Acme::` namespace,[140] which allows the program's source code to be "whitened" (i.e., all characters replaced with whitespace) and yet still work.

Example code

In older versions of Perl, one would write the Hello World program as:

```
print "Hello, World!\n";
```

Here is a more complex Perl program, that counts down seconds from a given starting value:

```
#!/usr/bin/perl
use strict;
use warnings;

my ( $remaining, $total );

$remaining = $total = shift(@ARGV);

STDOUT->autoflush(1);

while ( $remaining ) {
    printf ( "Remaining %s/%s \r", $remaining--, $total );
    sleep 1;
}

print "\n";
```

The perl interpreter can also be used for one-off scripts on the command line. The following example (as invoked from an sh-compatible shell, such as Bash) translates the string "Bob" in all files ending with .txt in the current directory to "Robert":

```
$ perl -i.bak -lp -e 's/Bob/Robert/g' *.txt
```

Criticism

Perl has been referred to as "line noise" and a write-only language by its critics. The earliest such mention was in the first edition of the book *Learning Perl*, a Perl 4 tutorial book written by Randal L. Schwartz,[141] in the first chapter of which he states: "Yes, sometimes Perl looks like line noise to the uninitiated, but to the seasoned Perl programmer, it looks like checksummed line noise with a mission in life."[142] He also stated that the accusation that Perl is a write-only language could be avoided by coding with "proper care".[142] The Perl overview document *perlintro* states that the names of built-in "magic" scalar variables "look like punctuation or line noise".[143] However, the English module provides both long and short English alternatives. *perlstyle* document states that line noise in regular expressions could be mitigated using the `/x` modifier to add whitespace.[144]

According to the *Perl 6 FAQ*, Perl 6 was designed to mitigate "the usual suspects" that elicit the "line noise" claim from Perl 5 critics, including the removal of "the majority of the punctuation variables" and the sanitization of the regex syntax.[145] The *Perl 6 FAQ* also states that what is sometimes referred to as Perl's line noise is "the actual syntax of the language" just as gerunds and prepositions are a part of the English language.[145] In a December 2012 blog posting, despite claiming that "Rakudo Perl 6 has failed and will continue to fail unless it gets some adult supervision", chromatic stated that the design of Perl 6 has a "well-defined grammar" as well as an "improved type system, a unified object system with an intelligent metamodel, metaoperators, and a clearer system of context that provides for such niceties as pervasive laziness".[146] He also stated that "Perl 6 has a coherence and a consistency that Perl 5 lacks."[146]

See also

- Outline of Perl
- Perl Data Language
- Perl Object Environment
- Plain Old Documentation

References

1. "v13i001: Perl, a "replacement" for awk and sed, Part01/10" (https://www.tuhs.org/Usenet/comp.sources.unix/1988-February/006962.html). comp.sources.unix archives. Retrieved August 11, 2021.
2. "Perl 5.34.0 is now available!" (https://www.nntp.perl.org/group/perl.perl5.porters/2021/05/msg260110.html). www.nntp.perl.org. Retrieved May 21, 2021.
3. "Perl 5.32.1 is now available!" (https://www.nntp.perl.org/group/perl.perl5.porters/2021/01/msg258868.html). www.nntp.perl.org. Archived (https://web.archive.org/web/20210124130938/https://www.nntp.perl.org/group/perl.perl5.porters/2021/01/msg258868.html) from the original on January 24, 2021. Retrieved January 24, 2021.
4. "Perl 5.35.5 is now available!" (https://www.nntp.perl.org/group/perl.perl5.porters/2021/10/msg261779.html). www.nntp.perl.org. Retrieved October 23, 2021.
5. "The "Artistic License" - dev.perl.org" (http://dev.perl.org/licenses/artistic.html). *dev.perl.org*. Archived (https://web.archive.org/web/20180724213601/http://dev.perl.org/licenses/artistic.html) from the original on July 24, 2018. Retrieved June 24, 2016.
6. Artistic (http://perl5.git.perl.org/perl.git/blob/HEAD:/Artistic) Archived (https://web.archive.org/web/20180725033309/http://perl5.git.perl.org/perl.git/blob/HEAD:/Artistic) July 25, 2018, at the Wayback Machine - file on the Perl 5 git repository

7. "Perl Licensing" (http://dev.perl.org/licenses). dev.perl.org. Archived (https://web.archive.org/web/20110122175123/http://dev.perl.org/licenses/) from the original on January 22, 2011. Retrieved January 8, 2011.
8. Larry Wall (December 12, 2007). "Programming is Hard, Let's Go Scripting..." (https://www.perl.com/pub/2007/12/06/soto-11.html/) Archived (https://web.archive.org/web/20170728023959/http://www.perl.com/pub/2007/12/06/soto-11.html) from the original on July 28, 2017. Retrieved April 14, 2019. "All language designers have their occasional idiosyncracies. I'm just better at it than most."
9. "About Perl" (https://www.perl.org/about.html). perl.org. Archived (https://web.archive.org/web/20151106051931/https://www.perl.org/about.html) from the original on November 6, 2015. Retrieved April 20, 2013. ""Perl" is a family of languages, "Perl 6" is part of the family, but it is a separate language that has its own development team. Its existence has no significant impact on the continuing development of "Perl 5"."
10. "Path to Raku" (https://github.com/Raku/problem-solving/blob/master/solutions/language/Path-to-Raku.md). GitHub. Archived (https://web.archive.org/web/20201112035821/https://github.com/Raku/problem-solving/blob/master/solutions/language/Path-to-Raku.md) from the original on November 12, 2020. Retrieved January 14, 2021. "This document describes the steps to be taken to effectuate a rename of Perl 6 to Raku"
11. Lapworth, Leo. "General Questions About Perl" (http://learn.perl.org/faq/perlfaq1.html#Whats-the-difference-between-perl-and-Perl). *Perl FAQ*. Perl.org. Archived (https://www.webcitation.org/6HZ9yBwbW?url=http://learn.perl.org/faq/perlfaq1.html#Whats-the-difference-between-perl-and-Perl) from the original on June 22, 2013. Retrieved February 24, 2012.
12. "perl(1): Practical Extraction/Report Language - Linux man page" (http://linux.die.net/man/1/perl). Linux.die.net. Archived (https://www.webcitation.org/6HZ9z0o0I?url=http://linux.die.net/man/1/perl) from the original on June 22, 2013. Retrieved July 23, 2013.
13. Sheppard, Doug (October 16, 2000). "Beginner's Introduction to Perl" (http://www.perl.com/pub/2000/10/begperl1.html). dev.perl.org. Archived (https://web.archive.org/web/20110605130400/http://www.perl.com/pub/2000/10/begperl1.html) from the original on June 5, 2011. Retrieved January 8, 2011.
14. Ashton, Elaine (1999). "The Timeline of Perl and its Culture (v3.0_0505)" (http://history.perl.org/PerlTimeline.html). Archived (https://web.archive.org/web/20130111100906/http://history.perl.org/PerlTimeline.html) from the original on January 11, 2013. Retrieved March 12, 2004.
15. Wall, Larry; Christiansen, Tom; Orwant, Jon (July 2000). *Programming Perl, Third Edition*. O'Reilly Media. ISBN 978-0-596-00027-1.
16. "Language Evaluations" (http://www.catb.org/esr/writings/taoup/html/ch14s04.html#perl). Archived (https://web.archive.org/web/20150310123855/http://www.catb.org/esr/writings/taoup/html/ch14s04.html#perl) from the original on March 10, 2015. Retrieved January 30, 2015. "Perl's strongest point is its extremely powerful built-in facilities for pattern-directed processing of textual, line-oriented data formats; it is unsurpassed at this."
17. "You Used Perl to Write WHAT?!" (http://www.cio.com/article/2437271/developer/you-used-perl-to-write-what--.html). January 24, 2008. Archived (https://web.archive.org/web/20150204175543/http://www.cio.com/article/2437271/developer/you-used-perl-to-write-what--.html) from the original on February 4, 2015. Retrieved February 4, 2015. "perl has always been the go-to language for any task that involves pattern-matching input"
18. "The Importance of Perl" (https://web.archive.org/web/20150202010003/http://archive.oreilly.com/pub/a/oreilly/perl/news/importance_0498.html). Archived from the original (http://archive.oreilly.com/pub/a/oreilly/perl/news/importance_0498.html) on February 2, 2015. Retrieved February 4, 2015. "Perl's unparalleled ability to process text..."
19. Smith, Roderick W. (June 21, 2002). *Advanced Linux Networking* (https://archive.org/details/linux00libg_999). Addison-Wesley Professional. p. 594 (https://archive.org/details/linux00libg_999/page/n595). ISBN 978-0-201-77423-8.

20. Sheppard, Doug (October 16, 2000). "Beginner's Introduction to Perl" (http://www.perl.com/pub/a/2000/10/begperl1.html). O'Reilly Media. Archived (https://web.archive.org/web/20080604140740/http://www.perl.com/pub/a/2000/10/begperl1.html) from the original on June 4, 2008. Retrieved July 27, 2008.
21. Raymond, Eric (December 23, 2003). "Swiss Army chainsaw" (http://www.catb.org/jargon/html/S/Swiss-Army-chainsaw.html). *The Jargon File*. Archived (https://web.archive.org/web/20180814171034/http://www.catb.org/jargon/html/S/Swiss-Army-chainsaw.html) from the original on August 14, 2018. Retrieved November 29, 2014.
22. Leonard, Andrew. "The joy of Perl" (http://www.salon.com/1998/10/13/feature_269/). *Salon.com*. Archived (https://web.archive.org/web/20120706094345/http://www.salon.com/1998/10/13/feature_269/) from the original on July 6, 2012. Retrieved June 5, 2012.
23. "How programs are measured | Computer Language Benchmarks Game" (https://benchmarksgame-team.pages.debian.net/benchmarksgame/how-programs-are-measured.html#source-code). *benchmarksgame-team.pages.debian.net*. Archived (https://web.archive.org/web/20200712000728/https://benchmarksgame-team.pages.debian.net/benchmarksgame/how-programs-are-measured.html#source-code) from the original on July 12, 2020. Retrieved October 5, 2020.
24. "RSA in 3 lines of perl - Everything2.com" (https://everything2.com/title/RSA+in+3+lines+of+perl). *everything2.com*. Archived (https://web.archive.org/web/20201008120935/https://everything2.com/title/RSA+in+3+lines+of+perl) from the original on October 8, 2020. Retrieved October 5, 2020.
25. Richardson, Marjorie (May 1, 1999). "Larry Wall, the Guru of Perl" (http://www.linuxjournal.com/article/3394). *Linux Journal*. Archived (https://www.webcitation.org/6HZA1Vm75?url=http://www.linuxjournal.com/article/3394) from the original on June 22, 2013. Retrieved January 3, 2011.
26. "perlfaq1: What's the difference between "perl" and "Perl"?" (http://perldoc.perl.org/perlfaq1.html#What's-the-difference-between-%22perl%22-and-%22Perl%22%3f). *perldoc.perl.org - Perl 5 version 12.2 documentation*. Archived (https://www.webcitation.org/6GYqliwy7?url=http://perldoc.perl.org/perlfaq1.html#What's-the-difference-between-%22perl%22-and-%22Perl%22%3f) from the original on May 12, 2013. Retrieved June 4, 2007.
27. Schwartz, Randal; foy, brian; Phoenix, Tom (June 16, 2011). *Learning Perl* (https://archive.org/details/learningperl00schw_278). O'Reilly Media, Inc. p. 4 (https://archive.org/details/learningperl00schw_278/page/n27). ISBN 978-1449313142. "Perl is sometimes called the "Practical Extraction and Report Language," although it has also been called a "Pathologically Eclectic Rubbish Lister," among other expansions. It's actually a backronym, not an acronym, since Larry Wall, Perl's creator, came up with the name first and the expansion later. That's why "Perl" isn't in all caps. There's no point in arguing that expansion is correct: Larry endorses both."
28. Wall, Larry. "perl - The Perl language interpreter" (http://perldoc.perl.org/perl.html#BUGS). *Perl 5 version 12.2 documentation*. Archived (https://www.webcitation.org/6HZA2aDcr?url=http://perldoc.perl.org/perl.html#BUGS) from the original on June 22, 2013. Retrieved January 26, 2011.
29. "perl.perl5.porters archive" (http://www.nntp.perl.org/group/perl.perl5.porters/). perl.org. Archived (https://web.archive.org/web/20110501081803/http://www.nntp.perl.org/group/perl.perl5.porters/) from the original on May 1, 2011. Retrieved January 13, 2011.
30. "perlhist — the Perl history records" (http://perldoc.perl.org/perlhist.html). *Perl 5 version 12.2 documentation*. perldoc.perl.org. Archived (https://web.archive.org/web/20110113030100/http://perldoc.perl.org/perlhist.html) from the original on January 13, 2011. Retrieved January 21, 2011.
31. "CPAN" (https://www.cpan.org/). CPAN. Archived (https://web.archive.org/web/20191003040107/https://www.cpan.org/) from the original on October 3, 2019. Retrieved May 8, 2017.

32. "perl5004delta — what's new for perl5.004" (http://perldoc.perl.org/perl5004delta.html). *Perl 5 version 12.2 documentation*. perldoc.perl.org. Archived (https://web.archive.org/web/20110227200616/http://perldoc.perl.org/perl5004delta.html) from the original on February 27, 2011. Retrieved January 8, 2011.
33. Patwardhan, Nathan; Siever, Ellen; Spainhour, Stephen (2002). *Perl in a Nutshell, Second Edition* (https://archive.org/details/perlinnutshell00patw). O'Reilly Media. ISBN 978-0-596-00241-1.
34. "perl5005delta - what's new for perl5.005" (http://perldoc.perl.org/perl5005delta.html). *Perl 5 version 12.2 documentation*. perldoc.perl.org. Archived (https://web.archive.org/web/20110203100249/http://perldoc.perl.org/perl5005delta.html) from the original on February 3, 2011. Retrieved January 21, 2011.
35. "perlhist - the Perl history records" (https://perldoc.perl.org/perlhist.html). www.cpan.org. Archived (https://web.archive.org/web/20200916211248/https://perldoc.perl.org/perlhist.html) from the original on September 16, 2020. Retrieved June 2, 2020.
36. "Perl Source" (http://www.cpan.org/src/). www.cpan.org. Archived (https://web.archive.org/web/20170601222517/http://www.cpan.org/src/) from the original on June 1, 2017. Retrieved June 2, 2020.
37. "perl56delta - what's new for perl v5.6.0" (http://perldoc.perl.org/perl56delta.html). *Perl 5 version 12.2 documentation*. perldoc.perl.org. Archived (https://web.archive.org/web/20110202135358/http://perldoc.perl.org/perl56delta.html) from the original on February 2, 2011. Retrieved January 21, 2011.
38. "perl56delta - what's new for perl v5.6.x" (http://perldoc.perl.org/perl561delta.html). *Perl 5 version 12.2 documentation*. perldoc.perl.org. Archived (https://web.archive.org/web/20101118101544/http://perldoc.perl.org/perl561delta.html) from the original on November 18, 2010. Retrieved January 21, 2011.
39. Wall, Larry. "Apocalypse 1: The Ugly, the Bad, and the Good" (http://dev.perl.org/perl6/doc/design/apo/A01.html). Archived (https://web.archive.org/web/20101123182201/http://dev.perl.org/perl6/doc/design/apo/A01.html) from the original on November 23, 2010. Retrieved January 8, 2011.
40. "perl58delta - what is new for perl v5.8.0" (http://perldoc.perl.org/perl58delta.html). *Perl 5 version 12.2 documentation*. perldoc.perl.org. Archived (https://web.archive.org/web/20101121023149/http://perldoc.perl.org/perl58delta.html) from the original on November 21, 2010. Retrieved January 21, 2011.
41. "A Plan for Pugs" (http://www.perl.com/pub/a/2005/03/03/pugs_interview.html). O'Reilly Media. March 3, 2005. Archived (https://archive.today/20120908200150/http://www.perl.com/pub/2005/03/03/pugs_interview.html) from the original on September 8, 2012. Retrieved January 27, 2011.
42. Tang, Audrey (April 21, 2010). "Re: How to Implement Perl 6 in Ten Years" (http://www.perlmonks.org/?node_id=835936). PerlMonks. Archived (https://web.archive.org/web/20110511190417/http://www.perlmonks.org/?node_id=835936) from the original on May 11, 2011. Retrieved January 3, 2011.
43. Geoff Broadwell (August 8, 2005), *OSCON 4.4: Inside Ponie, the Bridge from Perl 5 to Perl 6* (http://www.oreillynet.com/onlamp/blog/2005/08/oscon_44_inside_ponie_the_brid.html), O'Reilly ONLamp Blog, archived (https://web.archive.org/web/20120314013450/http://www.oreillynet.com/onlamp/blog/2005/08/oscon_44_inside_ponie_the_brid.html) from the original on March 14, 2012, retrieved June 27, 2016
44. Jesse Vincent (August 23, 2006), *Ponie has been put out to pasture* (https://web.archive.org/web/20090627091007/http://news.perlfoundation.org/2006/08/ponie_has_been_put_out_to_past.html), The Perl Foundation, archived from the original (http://news.perlfoundation.org/2006/08/ponie_has_been_put_out_to_past.html) on June 27, 2009, retrieved January 15, 2019

45. "perl5100delta - what is new for perl 5.10.0" (http://perldoc.perl.org/perl5100delta.html). *Perl 5 version 12.2 documentation*. perldoc.perl.org. Archived (https://web.archive.org/web/20101221024004/http://perldoc.perl.org/perl5100delta.html) from the original on December 21, 2010. Retrieved January 8, 2011.
46. "perlsyn - Perl syntax" (http://perldoc.perl.org/perlsyn.html#Smart-matching-in-detail). *Perl 5 version 12.2 documentation*. perldoc.perl.org. Archived (https://web.archive.org/web/20130826100652/http://perldoc.perl.org/perlsyn.html#Smart-matching-in-detail) from the original on August 26, 2013. Retrieved January 21, 2011.
47. "perl5120delta - what is new for perl v5.12.0" (http://perldoc.perl.org/perl5120delta.html). *Perl 5 version 12.2 documentation*. perldoc.perl.org. Archived (https://web.archive.org/web/20110104093548/http://perldoc.perl.org/perl5120delta.html) from the original on January 4, 2011. Retrieved January 8, 2011.
48. "perldelta - what is new for perl v5.12.3" (http://perldoc.perl.org/perl5123delta.html). *Perl 5 version 12.2 documentation*. perldoc.perl.org. Archived (https://web.archive.org/web/20110608102751/http://perldoc.perl.org/perl5123delta.html) from the original on June 8, 2011. Retrieved January 8, 2011.
49. "perl5140delta - what is new for perl v5.14.0 - metacpan.org" (https://metacpan.org/pod/distribution/perl/pod/perl5140delta.pod). *metacpan.org*. Archived (https://web.archive.org/web/20180725004523/https://metacpan.org/pod/distribution/perl/pod/perl5140delta.pod) from the original on July 25, 2018. Retrieved July 22, 2017.
50. "perldelta - what is new for perl v5.16.0" (https://metacpan.org/module/RJBS/perl-5.16.0/pod/perldelta.pod). *Perl 5 version 16.0 documentation*. metacpan.org. Archived (https://web.archive.org/web/20200728080612/https://metacpan.org/pod/release/RJBS/perl-5.16.0/pod/perldelta.pod) from the original on July 28, 2020. Retrieved May 21, 2012.
51. "perl5180delta - what is new for perl v5.18.0 - Perl programming language" (https://metacpan.org/pod/release/RJBS/perl-5.18.1/pod/perl5180delta.pod). *Perl 5 version 18.0 documentation*. metacpan.org. Archived (https://web.archive.org/web/20131029224638/https://metacpan.org/pod/release/RJBS/perl-5.18.1/pod/perl5180delta.pod) from the original on October 29, 2013. Retrieved October 27, 2013.
52. "perl5200delta - what is new for perl v5.20.0 - Perl programming language" (https://metacpan.org/source/RJBS/perl-5.20.0/pod/perldelta.pod). *Perl 5 version 20.0 documentation*. metacpan.org. Archived (https://archive.today/20140527190905/https://metacpan.org/source/RJBS/perl-5.20.0/pod/perldelta.pod) from the original on May 27, 2014. Retrieved May 27, 2014.
53. Milestones in the Perl Renaissance - Modern Perl Programming (http://www.modernperlbooks.com/mt/2009/07/milestones-in-the-perl-renaissance.html) Archived (https://web.archive.org/web/20121027105918/http://www.modernperlbooks.com/mt/2009/07/milestones-in-the-perl-renaissance.html) October 27, 2012, at the Wayback Machine. Modernperlbooks.com. Retrieved on 2013-07-17.
54. Preface (Modern Perl 2011-2012) (http://modernperlbooks.com/books/modern_perl/) Archived (https://web.archive.org/web/20120928232457/http://modernperlbooks.com/books/modern_perl/) September 28, 2012, at the Wayback Machine. Modernperlbooks.com. Retrieved on 2013-07-17.
55. Modern Perl 2011-2012 edition by chromatic | Onyx Neon Press (http://onyxneon.com/books/modern_perl/) Archived (https://web.archive.org/web/20111222045417/http://onyxneon.com/books/modern_perl/) December 22, 2011, at the Wayback Machine. Onyxneon.com. Retrieved on 2013-07-17.
56. "Enlightened Perl" (https://web.archive.org/web/20140205015536/http://www.enlightenedperl.org/). *Enlightened Perl*. Archived from the original (http://www.enlightenedperl.org/) on February 5, 2014. Retrieved September 28, 2012.
57. "YAPC::NA 2013 - June 3-5, Austin, Texas" (https://web.archive.org/web/20130622201417/http://www.yapcna.org/yn2013/talk/4725). Yapcna.org. June 4, 2013. Archived from the original (http://www.yapcna.org/yn2013/talk/4725) on June 22, 2013. Retrieved April 11, 2014.

58. Little, Stevan (February 8, 2013). "What is Moe (a clarification) | Stevan Little" (http://blogs.perl.org/users/stevan_little/2013/02/what-is-moe-a-clarification.html). Blogs.perl.org. Archived (https://web.archive.org/web/20131219185710/http://blogs.perl.org/users/stevan_little/2013/02/what-is-moe-a-clarification.html) from the original on December 19, 2013. Retrieved April 11, 2014.
59. "p2 on potion" (http://perl11.org/p2/). Perl11.org. February 7, 2004. Archived (https://web.archive.org/web/20130924023845/http://perl11.org/p2/) from the original on September 24, 2013. Retrieved April 11, 2014.
60. "goccy/gperl 路 GitHub" (https://github.com/goccy/gperl/). GitHub.com. Archived (https://web.archive.org/web/20150223170215/https://github.com/goccy/gperl) from the original on February 23, 2015. Retrieved April 11, 2014.
61. "rperl" (http://rperl.org/faq.html). RPerl.org. Archived (https://web.archive.org/web/20141018002115/http://rperl.org/faq.html) from the original on October 18, 2014. Retrieved August 11, 2014.
62. "Perl 7 announced at Perl Conference in the Cloud" (https://news.perlfoundation.org/post/perl_7_announced_sawyerx_conference). perlfoundation.org. June 24, 2020. Archived (https://web.archive.org/web/20200626213418/https://news.perlfoundation.org/post/perl_7_announced_sawyerx_conference) from the original on June 26, 2020. Retrieved June 24, 2020.
63. "Announcing Perl 7" (https://www.perl.com/article/announcing-perl-7/). perl.com. June 24, 2020. Archived (https://web.archive.org/web/20200624160531/https://www.perl.com/article/announcing-perl-7/) from the original on June 24, 2020. Retrieved June 24, 2020.
64. Nicholas Clark (May 9, 2021). "Steering Council meeting #019 2021-05-06" (https://www.nntp.perl.org/group/perl.perl5.porters/2021/05/msg260050.html). *www.nntp.perl.org*. Archived (https://web.archive.org/web/20210518015233/https://www.nntp.perl.org/group/perl.perl5.porters/2021/05/msg260050.html) from the original on May 18, 2021. Retrieved May 17, 2021. "The plan remains that there will be a Perl 7 bump, but not immediately after 5.34.0 is released. ... We don't think that we can deliver on [Perl 7] in 12 months."
65. *Preparing for Perl 7d* (https://leanpub.com/preparing_for_perl7). leanpub.com. June 24, 2020. Archived (https://web.archive.org/web/20200625004047/https://leanpub.com/preparing_for_perl7) from the original on June 25, 2020. Retrieved June 24, 2020.
66. Schwartz, Randal L; Phoenix, Tom; Foy, Brian (December 6, 2007). *Learning Perl, Third Edition* (https://archive.org/details/learningperl00schw). ISBN 978-0-596-00132-2.
67. "The Perl Camel Usage and Trademark Information" (https://web.archive.org/web/20180425080044/http://archive.oreilly.com/pub/a/oreilly/perl/usage). O'Reilly Media. Archived from the original (http://archive.oreilly.com/pub/a/oreilly/perl/usage) on April 25, 2018. Retrieved January 9, 2011.
68. "Perl Trademark" (https://web.archive.org/web/20110503211915/http://www.perlfoundation.org/perl_trademark). The Perl Foundation. Archived from the original (http://www.perlfoundation.org/perl_trademark) on May 3, 2011. Retrieved January 9, 2011.
69. Gillmore, Dan (October 25, 1998). "Republic Of Perl" (http://articles.chicagotribune.com/1998-10-25/news/9810250094_1_programmers-open-source-movement-programming-community). *Chicago Tribune*. Archived (https://web.archive.org/web/20110430031425/http://articles.chicagotribune.com/1998-10-25/news/9810250094_1_programmers-open-source-movement-programming-community) from the original on April 30, 2011. Retrieved January 10, 2011.
70. Riedel, Sebastian (January 18, 2012). "Perl 5 Raptor" (https://github.com/kraih/perl-raptor). *Sebastian Riedel*. Archived (https://web.archive.org/web/20180611021629/https://github.com/kraih/perl-raptor) from the original on June 11, 2018. Retrieved November 12, 2017.
71. Trout, Matt (June 16, 2005). "State of the Velociraptor - Phase two" (https://shadow.cat/blog/matt-s-trout/sotv-vi/). *Shadowcat Systems Limited*. Archived (https://web.archive.org/web/20171113003519/https://shadow.cat/blog/matt-s-trout/sotv-vi/) from the original on November 13, 2017. Retrieved November 12, 2017.
72. Nagpal, D.P. (2010). *Web Design Technology*. India: S. Chand. p. 700. ISBN 978-8121927635.

73. "title unknown". May 10, 1997. Usenet: 199705101952.MAA00756@wall.org (news:19970510 1952.MAA00756@wall.org).
74. Wall, Larry. "perl - The Perl 5 language interpreter - Perldoc Browser" (https://perldoc.perl.org/perl). *perldoc.perl.org*. Retrieved June 24, 2021.
75. Wall, Larry (March 1, 1997). "Wherefore Art, Thou?" (http://www.linuxjournal.com/article/2070). *Linux Journal*. Archived (https://web.archive.org/web/20101209021107/http://www.linuxjournal.com/article/2070) from the original on December 9, 2010. Retrieved March 13, 2011.
76. "perlfunc - Perl builtin functions" (http://perldoc.perl.org/perlfunc.html). *Perl 5 version 12.2 documentation*. perldoc.perl.org. Archived (https://web.archive.org/web/20110106003034/http://perldoc.perl.org/perlfunc.html) from the original on January 6, 2011. Retrieved January 10, 2011.
77. "Perl 6 Specification" (http://www.perl6.org/specification). The Perl 6 Project. Archived (https://web.archive.org/web/20091202073507/http://www.perl6.org/specification/) from the original on December 2, 2009. Retrieved January 27, 2011.
78. "Perl 6 Compilers" (http://www.perl6.org/compilers/). The Perl 6 Project. Archived (https://web.archive.org/web/20091202073302/http://www.perl6.org/compilers/) from the original on December 2, 2009. Retrieved January 27, 2011.
79. Gilmore, W. J. (2010). *Beginning PHP and MySQL: From Novice to Professional, Fourth Edition* (https://archive.org/details/beginningphpmysq00gilm_240). Apress. p. 484 (https://archive.org/details/beginningphpmysq00gilm_240/page/n519). ISBN 978-1-4302-3114-1.
80. "IMDb Helpdesk: What software/hardware are you using to run the site?" (https://www.imdb.com/help/search?domain=helpdesk_faq&index=1&file=techinfo). *Internet Movie Database*. Archived (https://web.archive.org/web/20170309083513/http://www.imdb.com/help/search?domain=helpdesk_faq&index=1&file=techinfo) from the original on March 9, 2017. Retrieved February 12, 2011.
81. DuckDuckGo handles a large amount of search queries at 4.5 million queries per day . https://duckduckgo.com/traffic.html Archived (http://webarchive.loc.gov/all/20120215210604/https://duckduckgo.com/traffic.html) February 15, 2012, at the Library of Congress Web Archives
82. DuckDuckGo uses Perl https://web.archive.org/web/20101231135106/http://www.gabrielweinberg.com/blog/2009/03/duckduck-go-architecture.html
83. "Perl FAQ" (https://wiki.debian.org/PerlFAQ). Archived (https://web.archive.org/web/20190806111259/https://wiki.debian.org/PerlFAQ) from the original on August 6, 2019. Retrieved August 6, 2019. "Perl is used quite extensively in Debian. Not only are some core functions written in Perl, but there are over 700 packages in unstable that have perl in their name (Mar 2004)."
84. "Perl Books - Book: Data Munging with Perl" (http://books.perl.org/book/95). *Perl.org*. Archived (https://web.archive.org/web/20110907020511/http://books.perl.org/book/95) from the original on September 7, 2011. Retrieved December 30, 2010.
85. A description of the Perl 5 interpreter can be found in *Programming Perl*, 3rd Ed., chapter 18. See particularly page 467, which carefully distinguishes run phase and compile phase from run time and compile time. Perl "time" and "phase" are often confused.
86. Schwartz, Randal. "On Parsing Perl" (http://www.perlmonks.org/index.pl?node_id=44722). Archived (https://web.archive.org/web/20070927000827/http://www.perlmonks.org/index.pl?node_id=44722) from the original on September 27, 2007. Retrieved January 3, 2007.
87. "The Perl Journal #19/9.26" (ftp://ftp.ora.com/pub/labs/tpj/tpj2.pdf) (PDF). O'Reilly Media. Retrieved February 4, 2011.
88. Kennedy, Adam (2006). "PPI—Parse, Analyze and Manipulate Perl (without perl)" (https://metacpan.org/module/PPI). CPAN. Archived (https://web.archive.org/web/20130903091241/https://metacpan.org/module/PPI) from the original on September 3, 2013. Retrieved September 16, 2013.

89. "Rice's Theorem". *The Perl Review*. **4** (3): 23–29. Summer 2008. and "Perl is Undecidable". *The Perl Review*. **5**: 7–11. Fall 2008., available online at Kegler, Jeffrey. "Perl and Undecidability" (http://www.jeffreykegler.com/Home/perl-and-undecidability). Archived (https://web.archive.org/web/20090817183115/http://www.jeffreykegler.com/Home/perl-and-undecidability) from the original on August 17, 2009. Retrieved January 4, 2009.
90. Hietaniemi, Jarkko (1998). "Perl Ports (Binary Distributions)" (http://www.cpan.org/ports/). CPAN.org. Archived (https://web.archive.org/web/20060418115903/http://www.cpan.org/ports/) from the original on April 18, 2006. Retrieved April 16, 2006.
91. "The MacPerl Pages" (http://www.macperl.com/). Prime Time Freeware. 1997. Archived (https://web.archive.org/web/20060118125208/http://www.macperl.com/) from the original on January 18, 2006. Retrieved January 18, 2006.
92. "Perl Ports (Binary Distributions)" (http://www.cpan.org/ports/). CPAN. Archived (https://web.archive.org/web/20060418115903/http://www.cpan.org/ports/) from the original on April 18, 2006. Retrieved January 27, 2011.
93. "ActivePerl is Perl for Windows, Mac, Linux, AIX, HP-UX & Solaris" (https://web.archive.org/web/20160331201814/http://www.activestate.com/activeperl). ActiveState Software. Archived from the original (http://www.activestate.com/activeperl) on March 31, 2016. Retrieved January 9, 2011.
94. "Using PPM" (http://docs.activestate.com/activeperl/5.12/faq/ActivePerl-faq2.html). ActiveState Software. Archived (https://web.archive.org/web/20100827110749/http://docs.activestate.com/activeperl/5.12/faq/ActivePerl-faq2.html) from the original on August 27, 2010. Retrieved January 9, 2011.
95. "Goodbye PPM, Hello State Tool" (https://www.activestate.com/blog/goodbye-ppm-hello-state-tool/). *activestate.com*. July 23, 2019. Archived (https://web.archive.org/web/20200728080612/https://www.activestate.com/blog/goodbye-ppm-hello-state-tool/) from the original on July 28, 2020. Retrieved April 16, 2020.
96. readme.txt
97. Descartes, Alligator; Bunce, Tim (2000). *Programming the Perl DBI : [database programming with Perl]* (https://archive.org/details/programmingperld00desc) (1 ed.). Beijing [u.a.]: O'Reilly. ISBN 978-1-56592-699-8.
98. "Perl Programming - Principles of Programming Languages" (https://sites.google.com/a/principlesofprogram.com/www/perl). *sites.google.com*. Retrieved May 18, 2021.
99. Bunce, Tim; Descartes, Alligator (February 4, 2000). *Programming the Perl DBI: Database programming with Perl* (https://books.google.com/books?id=WfoOrfuwcb8C&q=The+DBI+(Database+Interface)+module+presents+a+single,+database-independent+interface+to+Perl+applications,+while+the+DBD+(Database+Driver)+modules+handle+the+details+of+accessing+some+50+different+databases;+there+are+DBD+drivers+for+most+ANSI+SQL+databases). "O'Reilly Media, Inc.". ISBN 978-1-4493-1536-8.
100. Bekman, Stas. "Efficient Work with Databases under mod_perl" (http://perl.apache.org/docs/1.0/guide/performance.html#Efficient_Work_with_Databases_under_mod_perl). Archived (https://web.archive.org/web/20070822162513/http://perl.apache.org/docs/1.0/guide/performance.html#Efficient_Work_with_Databases_under_mod_perl) from the original on August 22, 2007. Retrieved September 1, 2007.
101. Pachev, Sasha (April 10, 2007). *Understanding MySQL Internals: Discovering and Improving a Great Database* (https://books.google.com/books?id=vz6PcTdo8VUC&q=DBI+provides+caching+for+database+handles+and+queries,+which+can+greatly+improve+performance+in+long-lived+execution+environments+such+as+mod+perl,%5B100%5D+helping+high-volume+systems+avert+load+spikes+as+in+the+Slashdot+effect). "O'Reilly Media, Inc.". ISBN 978-0-596-55280-0.
102. "Class::DBI - Simple Database Abstraction - metacpan.org" (https://metacpan.org/pod/Class::DBI). *metacpan.org*. Archived (https://web.archive.org/web/20200806100410/https://metacpan.org/pod/Class::DBI) from the original on August 6, 2020. Retrieved April 8, 2020.

103. "Rose::DB::Object - Extensible, high performance object-relational mapper (ORM). - metacpan.org" (https://metacpan.org/pod/Rose::DB::Object). *metacpan.org*. Archived (https://web.archive.org/web/20200806085143/https://metacpan.org/pod/Rose::DB::Object) from the original on August 6, 2020. Retrieved April 8, 2020.
104. "T sql querying developer reference" (http://pdfpremiumfree.com/download/t-sql-querying-developer-reference-pdf/). *pdfpremiumfree.com*. Retrieved May 18, 2021.
105. "Alioth: The Computer Language Benchmarks Game: Project Info" (https://web.archive.org/web/20130325192723/https://alioth.debian.org/projects/benchmarksgame/). Alioth. Archived from the original (http://alioth.debian.org/projects/benchmarksgame/) on March 25, 2013. Retrieved January 13, 2011.
106. "Which programs are fastest?" (https://web.archive.org/web/20130517145336/http://benchmarksgame.alioth.debian.org/u32/which-programs-are-fastest.php?v8=on&lua=on&jruby=on&php=on&python3=on&yarv=on&perl=on). *Computer Language Benchmarks Game*. Alioth. Archived from the original (http://benchmarksgame.alioth.debian.org/u32/which-programs-are-fastest.php?v8=on&lua=on&jruby=on&php=on&python3=on&yarv=on&perl=on) on May 17, 2013. Retrieved January 13, 2011.
107. Leroy, Jean-Louis (December 1, 2005). "A Timely Start" (http://www.perl.com/pub/a/2005/12/21/a_timely_start.html). O'Reilly. Archived (https://web.archive.org/web/20060613025623/http://www.perl.com/pub/a/2005/12/21/a_timely_start.html) from the original on June 13, 2006. Retrieved May 22, 2006.
108. Beattie, Malcolm & Enache Adrian (2003). "B::Bytecode Perl compiler's bytecode backend" (https://metacpan.org/module/NWCLARK/perl-5.8.8/ext/B/B/Bytecode.pm#KNOWN-BUGS). CPAN. Archived (https://web.archive.org/web/20210330011632/https://metacpan.org/pod/release/NWCLARK/perl-5.8.8/ext/B/B/Bytecode.pm#KNOWN-BUGS) from the original on March 30, 2021. Retrieved September 16, 2013.
109. Schwartz, Randal; foy, brian; Phoenix, Tom (June 23, 2011). *Learning Perl* (https://books.google.com/books?id=va1PSgaO4xIC&q=Therefore,+Perl+programs+pay+this+overhead+penalty+on+every+execution.+The+run+phase+of+typical+programs+is+long+enough+that+amortized+startup+time+is+not+substantial,+but+benchmarks+that+measure+very+short+execution+times+are+likely+to+be+skewed+due+to+this+overhead.). "O'Reilly Media, Inc.". ISBN 978-1-4493-0358-7.
110. Stein, Lincoln; MacEachern, Doug (1999). *Writing Apache Modules with Perl and C: The Apache API and Mod_perl* (https://books.google.com/books?id=qyzTI_eAeHUC&q=A+number+of+tools+have+been+introduced+to+improve+this+situation.+The+first+such+tool+was+Apache's+mod+perl,+which+sought+to+address+one+of+the+most-common+reasons+that+small+Perl+programs+were+invoked+rapidly:+CGI+Web+development.+ActivePerl,+via+Microsoft+ISAPI,+provides+similar+performance+improvements.). "O'Reilly Media, Inc.". ISBN 978-1-56592-567-0.
111. Bekman, Stas; Cholet, Eric (2003). *Practical Mod_perl* (https://books.google.com/books?id=UDabAgAAQBAJ&q=Once+Perl+code+is+compiled,+there+is+additional+overhead+during+the+execution+phase+that+typically+isn't+present+for+programs+written+in+compiled+languages+such+as+C+or+C++.+Examples+of+such+overhead+include+bytecode+interpretation,+reference-counting+memory+management,+and+dynamic+type-checking.). "O'Reilly Media, Inc.". ISBN 978-0-596-00227-5.
112. Ingerson, Brian. "Inline - metacpan.org" (https://web.archive.org/web/20130613163723/https://metacpan.org/module/Inline). CPAN. Archived from the original (https://metacpan.org/module/Inline) on June 13, 2013. Retrieved January 26, 2011.
113. "Perl 5.12.0 released - Update" (https://web.archive.org/web/20100419023449/http://www.h-online.com/open/news/item/Perl-5-12-0-released-Update-976919.html). Heise Media UK. April 13, 2010. Archived from the original (http://www.h-online.com/open/news/item/Perl-5-12-0-released-Update-976919.html) on April 19, 2010. Retrieved January 8, 2011.

114. "perl 5.16.1 released!" (http://www.perlmonks.org/?node_id=986397). August 9, 2012. Archived (https://web.archive.org/web/20121202012457/http://www.perlmonks.org/?node_id=986397) from the original on December 2, 2012. Retrieved August 26, 2012.
115. "The unstoppable Perl release train?" (https://lwn.net/Articles/484297/). LWN.net. February 29, 2012. Archived (https://web.archive.org/web/20160822222857/http://lwn.net/Articles/484297/) from the original on August 22, 2016. Retrieved March 28, 2012.
116. TIOBE Software Index (2020). "TIOBE Programming Community Index Perl" (https://www.tiobe.com/tiobe-index/). Archived (https://web.archive.org/web/20180225101948/https://www.tiobe.com/tiobe-index/) from the original on February 25, 2018. Retrieved February 6, 2020.
117. "Perl 6" (http://perl6.org/). The Perl 6 Project. Archived (https://web.archive.org/web/20110221094526/http://perl6.org/) from the original on February 21, 2011. Retrieved February 27, 2011.
118. Torkington, Nathan. "Transcription of Larry's talk" (http://www.nntp.perl.org/group/perl.perl6.meta/2000/10/msg424.html). nntp.perl.org. Archived (https://web.archive.org/web/20110501081806/http://www.nntp.perl.org/group/perl.perl6.meta/2000/10/msg424.html) from the original on May 1, 2011. Retrieved January 25, 2011.
119. "Perl6 - The future of Perl" (https://www.java-samples.com/showtutorial.php?tutorialid=1443). *www.java-samples.com*. Retrieved May 18, 2021.
120. "Official Perl 6 Documentation" (https://web.archive.org/web/20090831103918/http://perlcabal.org/syn/). The Perl 6 Project. Archived from the original (http://perlcabal.org/syn/) on August 31, 2009. Retrieved January 25, 2011.
121. Kuhn, Bradley (January 2001). "Considerations on Porting Perl to the Java Virtual Machine" (http://www.ebb.org/bkuhn/writings/technical/thesis/). University of Cincinnati. Archived (https://web.archive.org/web/20080321164747/http://ebb.org/bkuhn/writings/technical/thesis/) from the original on March 21, 2008. Retrieved June 28, 2008.
122. Chromatic (2015). *Modern Perl* (https://books.google.com/books?id=JUjmsgEACAAJ). Pragmatic Bookshelf. ISBN 978-1-68050-088-2.
123. "Feature comparison of Perl 6 compilers" (https://web.archive.org/web/20170811073233/https://perl6.org/compilers/features). Archived from the original (http://perl6.org/compilers/features) on August 11, 2017. Retrieved March 28, 2012.
124. Worthington, Jonathan (July 15, 2013). "Rakudo JVM News: More tests, plus Thread and Promise prototypes" (http://6guts.wordpress.com/2013/07/15/rakudo-jvm-news-more-tests-plus-thread-and-promise-prototypes/). *6guts*. Archived (https://web.archive.org/web/20131005055559/http://6guts.wordpress.com/2013/07/15/rakudo-jvm-news-more-tests-plus-thread-and-promise-prototypes/) from the original on October 5, 2013. Retrieved July 24, 2013.
125. Worthington, Jonathan (May 31, 2013). "MoarVM: A virtual machine for NQP and Rakudo" (http://6guts.wordpress.com/2013/05/31/moarvm-a-virtual-machine-for-nqp-and-rakudo/). *6guts*. Archived (https://web.archive.org/web/20130709185252/http://6guts.wordpress.com/2013/05/31/moarvm-a-virtual-machine-for-nqp-and-rakudo/) from the original on July 9, 2013. Retrieved July 24, 2013.
126. "rename-lwn" (https://web.archive.org/web/20191017155422/https://lwn.net/Articles/802329/). Archived from the original (https://lwn.net/Articles/802329/) on October 17, 2019. Retrieved November 10, 2019.
127. "rakudo/rakudo - GitHub" (https://github.com/rakudo/rakudo/). GitHub.com. Archived (https://web.archive.org/web/20170729084734/https://github.com/rakudo/rakudo/) from the original on July 29, 2017. Retrieved September 21, 2013.
128. "Announcing Perl 7" (https://www.perl.com/article/announcing-perl-7/). perl.com. June 24, 2020. Archived (https://web.archive.org/web/20200624160531/https://www.perl.com/article/announcing-perl-7/) from the original on June 24, 2020. Retrieved June 24, 2020.
129. "Perl 7 announced at Perl Conference in the Cloud" (https://news.perlfoundation.org/post/perl_7_announced_sawyerx_conference). perlfoundation.org. June 24, 2020. Archived (https://web.archive.org/web/20200626213418/https://news.perlfoundation.org/post/perl_7_announced_sawyerx_conference) from the original on June 26, 2020. Retrieved June 24, 2020.

130. Wall, Larry (May 22, 2014). "Perl Culture (AKA the first State of the Onion)" (http://grnlight.net/index.php/programming-articles/100-perl-culture). Archived (https://web.archive.org/web/20140522141559/http://grnlight.net/index.php/programming-articles/100-perl-culture) from the original on May 22, 2014. Retrieved May 22, 2014.
131. Larry Wall. "2nd State of the Onion" (http://www.wall.org/~larry/onion/onion.html). Archived (https://web.archive.org/web/20120717014443/http://www.wall.org/~larry/onion/onion.html) from the original on July 17, 2012. Retrieved October 12, 2012. (Search for 'church')
132. Randal L. Schwartz (May 2, 1999). "Who is Just another Perl hacker?" (https://groups.google.com/forum/#!msg/comp.lang.perl.misc/nK-lswsaMec/DBL87v4FxOwJ). Newsgroup: comp.lang.perl.misc (news:comp.lang.perl.misc). Usenet: m1hfpvh2jq.fsf@halfdome.holdit.com (news:m1hfpvh2jq.fsf@halfdome.holdit.com). Archived (https://archive.today/20120708165748/http://groups.google.com/group/comp.sys.acorn.programmer/browse_thread/thread/b5fd3717bda6a8d0/d4d3e151a783dffa?lnk=gst&q=ioc%23d4d3e151a783dffa#!msg/comp.lang.perl.misc/nK-lswsaMec/DBL87v4FxOwJ) from the original on July 8, 2012. Retrieved December 5, 2014.
133. Schwartz, Randal (March 31, 2005). "Canonical JAPH" (http://www.perlmonks.org/bare/?node_id=443856). PerlMonks. Archived (https://web.archive.org/web/20110722055125/http://www.perlmonks.org/bare/?node_id=443856) from the original on July 22, 2011. Retrieved May 16, 2011.
134. Greg Bacon (May 28, 1999). "Re: Incrementing a value in a slice" (https://groups.google.com/group/comp.lang.perl.misc/msg/7b97c434492c8d20). Newsgroup: comp.lang.perl.misc (news:comp.lang.perl.misc). Usenet: 7imntimjh1@info2.uah.edu (news:7imntimjh1@info2.uah.edu). Archived (https://web.archive.org/web/20110707134412/http://groups.google.com/group/comp.lang.perl.misc/msg/7b97c434492c8d20) from the original on July 7, 2011. Retrieved July 12, 2011.
135. Back, Adam. "RSA in 5 lines of perl" (http://www.cypherspace.org/rsa/pureperl.html). Archived (https://web.archive.org/web/20110119154503/http://www.cypherspace.org/rsa/pureperl.html) from the original on January 19, 2011. Retrieved January 10, 2011.
136. "Code Golf: What is Code Golf?" (https://web.archive.org/web/20120113152453/http://codegolf.com/). 29degrees. 2007. Archived from the original (http://codegolf.com/) on January 13, 2012. Retrieved November 26, 2018.
137. Gallo, Felix (2003). "The Zeroth Obfuscated Perl Contest" (https://web.archive.org/web/20091122114544/http://oreilly.com/catalog/tpj3/chapter/ch43.pdf) (PDF). In Jon Orwant (ed.). *Games, diversions, and Perl culture: best of the Perl journal*. O'Reilly Media. Archived from the original (http://oreilly.com/catalog/tpj3/chapter/ch43.pdf) (PDF) on November 22, 2009. Retrieved January 12, 2011.
138. "Perl Poetry" (http://www.perlmonks.org/?node_id=1590). PerlMonks. Archived (https://web.archive.org/web/20070927000904/http://www.perlmonks.org/?node_id=1590) from the original on September 27, 2007. Retrieved January 27, 2011.
139. Conway, Damian. "Lingua::Romana::Perligata -- Perl for the XXI-imum Century" (http://www.csse.monash.edu.au/~damian/papers/HTML/Perligata.html). Archived (https://web.archive.org/web/20070930165519/http://www.csse.monash.edu.au/~damian/papers/HTML/Perligata.html) from the original on September 30, 2007. Retrieved June 15, 2006.
140. Brocard, Leon (May 22, 2014). "use Perl; Journal of acme" (https://web.archive.org/web/20140522123044/http://grnlight.net/index.php/programming-articles/101-use-perl-journal-of-acme). GrnLight.net. Archived from the original (http://grnlight.net/index.php/programming-articles/101-use-perl-journal-of-acme) on May 22, 2014.
141. "Developer Update". *Dr. Dobb's Developer Update*. Miller-Freeman. **2**. 1995.
142. Schwartz, Randal L. (1993). *Learning Perl*. O'Reilly & Associates. Bibcode:1993lepe.book.....S (https://ui.adsabs.harvard.edu/abs/1993lepe.book.....S).
143. "perlintro" (https://web.archive.org/web/20110109121845/http://perldoc.perl.org/perlintro.html). *Perl 5 version 18.0 documentation*. Perl 5 Porters and perldoc.perl.org. Archived from the original (http://perldoc.perl.org/perlintro.html) on January 9, 2011. Retrieved June 30, 2013.

144. "perlstyle" (https://web.archive.org/web/20130626010707/http://perldoc.perl.org/perlstyle.html). *Perl 5 version 18.0 documentation*. Perl 5 Porters and perldoc.perl.org. Archived from the original (http://perldoc.perl.org/perlstyle.html) on June 26, 2013. Retrieved June 30, 2013.
145. "Perl 6 FAQ" (http://www.perl6.org/archive/faq.html). Perl 6 Project. Archived (https://web.archive.org/web/20130701183900/http://www.perl6.org/archive/faq.html) from the original on July 1, 2013. Retrieved June 30, 2013.
146. chromatic (December 31, 2012). "The Implementation of Perl 5 versus Perl 6" (http://www.modernperlbooks.com/mt/2012/12/the-implementation-of-perl-5-versus-perl-6.html). Archived (https://web.archive.org/web/20130729214209/http://www.modernperlbooks.com/mt/2012/12/the-implementation-of-perl-5-versus-perl-6.html) from the original on July 29, 2013. Retrieved June 30, 2013.

Further reading

- Learning Perl (http://shop.oreilly.com/product/0636920018452.do) 6th Edition (2011), O'Reilly. Beginner-level introduction to Perl.
- Beginning Perl (http://ofps.oreilly.com/titles/9781118013847/index.html) 1st Edition (2012), Wrox. A beginner's tutorial for those new to programming or just new to Perl.
- Modern Perl (http://onyxneon.com/books/modern_perl/) Archived (https://web.archive.org/web/20111222045417/http://onyxneon.com/books/modern_perl/) December 22, 2011, at the Wayback Machine 2nd Edition (2012), Onyx Neon. Describes Modern Perl programming techniques.
- Programming Perl (http://shop.oreilly.com/product/9780596004927.do) 4th Edition (2012), O'Reilly. The definitive Perl reference.
- Effective Perl Programming (http://www.pearsonhighered.com/educator/product/Effective-Perl-Programming-Ways-to-Write-Better-More-Idiomatic-Perl-2E/9780321496942.page) 2nd Edition (2010), Addison-Wesley. Intermediate- to advanced-level guide to writing idiomatic Perl.
- *Perl Cookbook*, ISBN 0-596-00313-7. Practical Perl programming examples.
- Dominus, Mark Jason (2005). *Higher Order Perl* (http://hop.perl.plover.com/book/). Morgan Kaufmann. ISBN 978-1-55860-701-9. Functional programming techniques in Perl.

External links

- Official website (https://www.perl.org/)