NAME

PCRE2 - Perl-compatible regular expressions (revised API)

INTRODUCTION

PCRE2 is the name used for a revised API for the PCRE library, which is a set of functions, written in C, that implement regular expression pattern matching using the same syntax and semantics as Perl, with just a few differences. After nearly two decades, the limitations of the original API were making development increasingly difficult. The new API is more extensible, and it was simplified by abolishing the separate "study" optimizing function; in PCRE2, patterns are automatically optimized where possible. Since forking from PCRE1, the code has been extensively refactored and new features introduced. The old library is now obsolete and is no longer maintained.

As well as Perl-style regular expression patterns, some features that appeared in Python and the original PCRE before they appeared in Perl are available using the Python syntax. There is also some support for one or two .NET and Oniguruma syntax items, and there are options for requesting some minor changes that give better ECMAScript (aka JavaScript) compatibility.

The source code for PCRE2 can be compiled to support strings of 8-bit, 16-bit, or 32-bit code units, which means that up to three separate libraries may be installed, one for each code unit size. The size of code unit is not related to the bit size of the underlying hardware. In a 64-bit environment that also supports 32-bit applications, versions of PCRE2 that are compiled in both 64-bit and 32-bit modes may be needed.

The original work to extend PCRE to 16-bit and 32-bit code units was done by Zoltan Herczeg and Christian Persch, respectively. In all three cases, strings can be interpreted either as one character per code unit, or as UTF-encoded Unicode, with support for Unicode general category properties. Unicode support is optional at build time (but is the default). However, processing strings as UTF code units must be enabled explicitly at run time. The version of Unicode in use can be discovered by running

```
pcre2test -C
```

The three libraries contain identical sets of functions, with names ending in _8, _16, or _32, respectively (for example, **pcre2_compile_8()**). However, by defining PCRE2_CODE_UNIT_WIDTH to be 8, 16, or 32, a program that uses just one code unit width can be written using generic names such as **pcre2_compile()**, and the documentation is written assuming that this is the case.

In addition to the Perl-compatible matching function, PCRE2 contains an alternative function that matches the same compiled patterns in a different way. In certain circumstances, the alternative function has some advantages. For a discussion of the two matching algorithms, see the **pcre2matching** page.

Details of exactly which Perl regular expression features are and are not supported by PCRE2 are given in separate documents. See the **pcre2pattern** and **pcre2compat** pages. There is a syntax summary in the **pcre2syntax** page.

Some features of PCRE2 can be included, excluded, or changed when the library is built. The **pcre2_config()** function makes it possible for a client to discover which features are available. The features themselves are described in the **pcre2build** page. Documentation about building PCRE2 for various operating systems can be found in the **README** and **NON-AUTOTOOLS_BUILD** files in the source distribution.

The libraries contains a number of undocumented internal functions and data tables that are used by more than one of the exported external functions, but which are not intended for use by external callers. Their names all begin with "_pcre2", which hopefully will not provoke any name clashes. In some environments, it is possible to control which external symbols are exported when a shared library is built, and in these cases the undocumented symbols are not exported.

SECURITY CONSIDERATIONS

If you are using PCRE2 in a non-UTF application that permits users to supply arbitrary patterns for compilation, you should be aware of a feature that allows users to turn on UTF support from within a pattern. For example, an 8-bit pattern that begins with "(*UTF)" turns on UTF-8 mode, which interprets patterns and subjects as strings of UTF-8 code units instead of individual 8-bit characters. This causes both the pattern and any data against which it is matched to be checked for UTF-8 validity. If the data string is very long, such a check might use sufficiently many resources as to cause your application to lose performance.

One way of guarding against this possibility is to use the **pcre2_pattern_info()** function to check the compiled pattern’s options for PCRE2_UTF. Alternatively, you can set the PCRE2_NEVER_UTF option when calling **pcre2_compile()**. This causes a compile time error if the pattern contains a UTF-setting sequence.

The use of Unicode properties for character types such as \d can also be enabled from within the pattern, by specifying "(*UCP)". This feature can be disallowed by setting the PCRE2_NEVER_UCP option.

If your application is one that supports UTF, be aware that validity checking can take time. If the same data string is to be matched many times, you can use the PCRE2_NO_UTF_CHECK option for the second and subsequent matches to avoid running redundant checks.

The use of the \C escape sequence in a UTF-8 or UTF-16 pattern can lead to problems, because it may leave the current matching point in the middle of a multi-code-unit character. The

PCRE2_NEVER_BACKSLASH_C option can be used by an application to lock out the use of \C, causing a compile-time error if it is encountered. It is also possible to build PCRE2 with the use of \C permanently disabled.

Another way that performance can be hit is by running a pattern that has a very large search tree against a string that will never match. Nested unlimited repeats in a pattern are a common example. PCRE2 provides some protection against this: see the **pcre2_set_match_limit()** function in the **pcre2api** page. There is a similar function called **pcre2_set_depth_limit()** that can be used to restrict the amount of memory that is used.

USER DOCUMENTATION

The user documentation for PCRE2 comprises a number of different sections. In the "man" format, each of these is a separate "man page". In the HTML format, each is a separate page, linked from the index page. In the plain text format, the descriptions of the **pcre2grep** and **pcre2test** programs are in files called **pcre2grep.txt** and **pcre2test.txt**, respectively. The remaining sections, except for the **pcre2demo** section (which is a program listing), and the short pages for individual functions, are concatenated in **pcre2.txt**, for ease of searching. The sections are as follows:

pcre2	this document
pcre2-config	show PCRE2 installation configuration information
pcre2api	details of PCRE2's native C API
pcre2build	building PCRE2
pcre2callout	details of the pattern callout feature
pcre2compat	discussion of Perl compatibility
pcre2convert	details of pattern conversion functions
pcre2demo	a demonstration C program that uses PCRE2
pcre2grep	description of the **pcre2grep** command (8-bit only)
pcre2jit	discussion of just-in-time optimization support
pcre2limits	details of size and other limits
pcre2matching	discussion of the two matching algorithms
pcre2partial	details of the partial matching facility
pcre2pattern	syntax and semantics of supported regular expression patterns
pcre2perform	discussion of performance issues
pcre2posix	the POSIX-compatible C API for the 8-bit library
pcre2sample	discussion of the pcre2demo program
pcre2serialize	details of pattern serialization
pcre2syntax	quick syntax reference
pcre2test	description of the **pcre2test** command
pcre2unicode	discussion of Unicode and UTF support

In the "man" and HTML formats, there is also a short page for each C library function, listing its arguments and results.

AUTHOR

Philip Hazel
Retired from University Computing Service
Cambridge, England.

Putting an actual email address here is a spam magnet. If you want to email me, use my two names separated by a dot at gmail.com.

REVISION

Last updated: 27 August 2021