NAME

PCRE - Perl-compatible regular expressions (original API)

PLEASE TAKE NOTE

This document relates to PCRE releases that use the original API, with library names libpcre, libpcre16, and libpcre32. January 2015 saw the first release of a new API, known as PCRE2, with release numbers starting at 10.00 and library names libpcre2-8, libpcre2-16, and libpcre2-32. The old libraries (now called PCRE1) are now at end of life, and 8.45 is the final release. New projects are advised to use the new PCRE2 libraries.

INTRODUCTION

The PCRE library is a set of functions that implement regular expression pattern matching using the same syntax and semantics as Perl, with just a few differences. Some features that appeared in Python and PCRE before they appeared in Perl are also available using the Python syntax, there is some support for one or two .NET and Oniguruma syntax items, and there is an option for requesting some minor changes that give better JavaScript compatibility.

Starting with release 8.30, it is possible to compile two separate PCRE libraries: the original, which supports 8-bit character strings (including UTF-8 strings), and a second library that supports 16-bit character strings (including UTF-16 strings). The build process allows either one or both to be built. The majority of the work to make this possible was done by Zoltan Herczeg.

Starting with release 8.32 it is possible to compile a third separate PCRE library that supports 32-bit character strings (including UTF-32 strings). The build process allows any combination of the 8-, 16- and 32-bit libraries. The work to make this possible was done by Christian Persch.

The three libraries contain identical sets of functions, except that the names in the 16-bit library start with **pcre16_** instead of **pcre_**, and the names in the 32-bit library start with **pcre32_** instead of **pcre_**. To avoid over-complication and reduce the documentation maintenance load, most of the documentation describes the 8-bit library, with the differences for the 16-bit and 32-bit libraries described separately in the **pcre16** and **pcre32** pages. References to functions or structures of the form *pcre[16|32]_xxx* should be read as meaning "*pcre_xxx* when using the 8-bit library, *pcre16_xxx* when using the 16-bit library, or *pcre32_xxx* when using the 32-bit library".

The current implementation of PCRE corresponds approximately with Perl 5.12, including support for UTF-8/16/32 encoded strings and Unicode general category properties. However, UTF-8/16/32 and Unicode support has to be explicitly enabled; it is not the default. The Unicode tables correspond to Unicode release 6.3.0.

In addition to the Perl-compatible matching function, PCRE 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 **pcrematching** page.

PCRE is written in C and released as a C library. A number of people have written wrappers and interfaces of various kinds. In particular, Google Inc. have provided a comprehensive C++ wrapper for the 8-bit library. This is now included as part of the PCRE distribution. The **pcrecpp** page has details of this interface. Other people's contributions can be found in the *Contrib* directory at the primary FTP site, which is:

ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre

Details of exactly which Perl regular expression features are and are not supported by PCRE are given in separate documents. See the **pcrepattern** and **pcrecompat** pages. There is a syntax summary in the **pcresyntax** page.

Some features of PCRE can be included, excluded, or changed when the library is built. The **pcre_config()** function makes it possible for a client to discover which features are available. The features themselves are described in the **pcrebuild** page. Documentation about building PCRE 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 "_pcre_" or "_pcre16_" or "_pcre32_", 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 PCRE 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, provided that PCRE was built with UTF support. For example, an 8-bit pattern that begins with "(*UTF8)" or "(*UTF)" turns on UTF-8 mode, which interprets patterns and subjects as strings of UTF-8 characters 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 **pcre_fullinfo()** function to check the compiled pattern's options for UTF. Alternatively, from release 8.33, you can set the PCRE_NEVER_UTF option at compile time. This causes a compile time error if a pattern contains a UTF-setting sequence.

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 PCRE_NO_UTF[8|16|32]_CHECK option for the second and subsequent matches to save redundant checks.

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. PCRE provides some protection against this: see the PCRE_EXTRA_MATCH_LIMIT feature in the **pcreapi** page.

USER DOCUMENTATION

The user documentation for PCRE 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 **pcregrep** and **pcretest** programs are in files called **pcregrep.txt** and **pcretest.txt**, respectively. The remaining sections, except for the **pcredemo** section (which is a program listing), are concatenated in **pcre.txt**, for ease of searching. The sections are as follows:

pcre	this document
pcre-config	show PCRE installation configuration information
pcre16	details of the 16-bit library
pcre32	details of the 32-bit library
pcreapi	details of PCRE’s native C API
pcrebuild	building PCRE
pcrecallout	details of the callout feature
pcrecompat	discussion of Perl compatibility
pcrecpp	details of the C++ wrapper for the 8-bit library
pcredemo	a demonstration C program that uses PCRE
pcregrep	description of the **pcregrep** command (8-bit only)
pcrejit	discussion of the just-in-time optimization support
pcrelimits	details of size and other limits
pcrematching	discussion of the two matching algorithms
pcrepartial	details of the partial matching facility
pcrepattern	syntax and semantics of supported regular expressions
pcreperform	discussion of performance issues
pcreposix	the POSIX-compatible C API for the 8-bit library
pcreprecompile	details of saving and re-using precompiled patterns
pcresample	discussion of the pcredemo program
pcrestack	discussion of stack usage
pcresyntax	quick syntax reference

pcretest description of the **pcretest** testing command
pcreunicode discussion of Unicode and UTF-8/16/32 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
University Computing Service
Cambridge CB2 3QH, England.

Putting an actual email address here seems to have been a spam magnet, so I’ve taken it away. If you want to email me, use my two initials, followed by the two digits 10, at the domain cam.ac.uk.

REVISION

Last updated: 14 June 2021
Copyright (c) 1997-2021 University of Cambridge.