NAME

PCRE - Perl-compatible regular expressions

SYNOPSIS

#include <pcre.h>

int pcre_exec(const pcre ****code*, const pcre_extra ****extra***,**
const char ****subject*, int** *length***, int** *startoffset***,**
int *options***, int ****ovector***, int** *ovecsize***);**

int pcre16_exec(const pcre16 ****code*, const pcre16_extra ****extra***,**
PCRE_SPTR16 *subject***, int** *length***, int** *startoffset***,**
int *options***, int ****ovector***, int** *ovecsize***);**

int pcre32_exec(const pcre32 ****code*, const pcre32_extra ****extra***,**
PCRE_SPTR32 *subject***, int** *length***, int** *startoffset***,**
int *options***, int ****ovector***, int** *ovecsize***);**

DESCRIPTION

This function matches a compiled regular expression against a given subject string, using a matching algorithm that is similar to Perl's. It returns offsets to captured substrings. Its arguments are:

```
code        Points to the compiled pattern
extra       Points to an associated pcre[16|32]_extra structure,
              or is NULL
subject     Points to the subject string
length      Length of the subject string
startoffset Offset in the subject at which to start matching
options     Option bits
ovector     Points to a vector of ints for result offsets
ovecsize    Number of elements in the vector (a multiple of 3)
```

The units for *length* and *startoffset* are bytes for **pcre_exec()**, 16-bit data items for **pcre16_exec()**, and 32-bit items for **pcre32_exec()**. The options are:

```
PCRE_ANCHORED          Match only at the first position
PCRE_BSR_ANYCRLF       \R matches only CR, LF, or CRLF
PCRE_BSR_UNICODE       \R matches all Unicode line endings
PCRE_NEWLINE_ANY       Recognize any Unicode newline sequence
PCRE_NEWLINE_ANYCRLF   Recognize CR, LF, & CRLF as newline sequences
PCRE_NEWLINE_CR        Recognize CR as the only newline sequence
PCRE_NEWLINE_CRLF      Recognize CRLF as the only newline sequence
PCRE_NEWLINE_LF        Recognize LF as the only newline sequence
PCRE_NOTBOL            Subject string is not the beginning of a line
PCRE_NOTEOL            Subject string is not the end of a line
PCRE_NOTEMPTY          An empty string is not a valid match
PCRE_NOTEMPTY_ATSTART  An empty string at the start of the subject
                         is not a valid match
PCRE_NO_START_OPTIMIZE Do not do "start-match" optimizations
PCRE_NO_UTF16_CHECK    Do not check the subject for UTF-16
                         validity (only relevant if PCRE_UTF16
                         was set at compile time)
PCRE_NO_UTF32_CHECK    Do not check the subject for UTF-32
                         validity (only relevant if PCRE_UTF32
                          was set at compile time)
  PCRE_NO_UTF8_CHECK     Do not check the subject for UTF-8
                          validity (only relevant if PCRE_UTF8
                          was set at compile time)
  PCRE_PARTIAL           ) Return PCRE_ERROR_PARTIAL for a partial
  PCRE_PARTIAL_SOFT      )   match if no full matches are found
  PCRE_PARTIAL_HARD      Return PCRE_ERROR_PARTIAL for a partial match
                          if that is found before a full match
```

For details of partial matching, see the **pcrepartial** page. A **pcre_extra** structure contains the following fields:

flags	Bits indicating which fields are set
study_data	Opaque data from **pcre[16|32]_study()**
match_limit	Limit on internal resource use
match_limit_recursion	Limit on internal recursion depth
callout_data	Opaque data passed back to callouts
tables	Points to character tables or is NULL
mark	For passing back a *MARK pointer
executable_jit	Opaque data from JIT compilation

The flag bits are PCRE_EXTRA_STUDY_DATA, PCRE_EXTRA_MATCH_LIMIT, PCRE_EXTRA_MATCH_LIMIT_RECURSION, PCRE_EXTRA_CALLOUT_DATA, PCRE_EXTRA_TABLES, PCRE_EXTRA_MARK and PCRE_EXTRA_EXECUTABLE_JIT.

There is a complete description of the PCRE native API in the **pcreapi** page and a description of the POSIX API in the **pcreposix** page.