

NAME

PCRE - Perl-compatible regular expressions

SYNOPSIS

```
#include <pcre.h>
```

```
int pcre_jit_exec(const pcre *code, const pcre_extra *extra,
    const char *subject, int length, int startoffset,
    int options, int *ovector, int ovecsize,
    pcre_jit_stack *jstack);
```

```
int pcre16_jit_exec(const pcre16 *code, const pcre16_extra *extra,
    PCRE_SPTR16 subject, int length, int startoffset,
    int options, int *ovector, int ovecsize,
    pcre_jit_stack *jstack);
```

```
int pcre32_jit_exec(const pcre32 *code, const pcre32_extra *extra,
    PCRE_SPTR32 subject, int length, int startoffset,
    int options, int *ovector, int ovecsize,
    pcre_jit_stack *jstack);
```

DESCRIPTION

This function matches a compiled regular expression that has been successfully studied with one of the JIT options against a given subject string, using a matching algorithm that is similar to Perl's. It is a "fast path" interface to JIT, and it bypasses some of the sanity checks that **pcre_exec()** applies. It returns offsets to captured substrings. Its arguments are:

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

The allowed options are:

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_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

However, the **PCRE_NO_UTF[8|16|32]_CHECK** options have no effect, as this check is never applied. 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 JIT API in the **pcrejit** page.