NAME
regcomp, regexec, regerror, regfree - POSIX regex functions

SYNOPSIS
#include <sys/types.h>
#include <regex.h>

int regcomp(regex_t **preg***, const char ****regex***, int** *cflags***);**

int regexec(const regex_t **preg***, const char ****string***, size_t** *nmatch***,**
 regmatch_t *pmatch[]***, int** *eflags***);**

size_t regerror(int *errcode***, const regex_t ****preg***, char ****errbuf***,**
 size_t *errbuf_size***);**

void regfree(regex_t **preg***);**

DESCRIPTION
POSIX regex compiling
regcomp() is used to compile a regular expression into a form that is suitable for subsequent **regexec**()
searches.

regcomp() is supplied with *preg*, a pointer to a pattern buffer storage area; *regex*, a pointer to the null-terminated string and *cflags*, flags used to determine the type of compilation.

All regular expression searching must be done via a compiled pattern buffer, thus **regexec**() must always be
supplied with the address of a **regcomp**() initialized pattern buffer.

cflags may be the bitwise-**or** of zero or more of the following:

REG_EXTENDED
> Use **POSIX** Extended Regular Expression syntax when interpreting *regex*. If not set, **POSIX**
> Basic Regular Expression syntax is used.

REG_ICASE
> Do not differentiate case. Subsequent **regexec**() searches using this pattern buffer will be case
> insensitive.

REG_NOSUB
> Do not report position of matches. The *nmatch* and *pmatch* arguments to **regexec**() are ignored if
> the pattern buffer supplied was compiled with this flag set.

REG_NEWLINE
> Match-any-character operators don't match a newline.
>
> A nonmatching list ([^**...**]) not containing a newline does not match a newline.
>
> Match-beginning-of-line operator (**ˆ**) matches the empty string immediately after a newline,
> regardless of whether *eflags*, the execution flags of **regexec**(), contains **REG_NOTBOL**.
>
> Match-end-of-line operator (**$**) matches the empty string immediately before a newline, regardless
> of whether *eflags* contains **REG_NOTEOL**.

POSIX regex matching
regexec() is used to match a null-terminated string against the precompiled pattern buffer, *preg*. *nmatch*
and *pmatch* are used to provide information regarding the location of any matches. *eflags* may be the bitwise-**or** of one or both of **REG_NOTBOL** and **REG_NOTEOL** which cause changes in matching behavior described below.

REG_NOTBOL
> The match-beginning-of-line operator always fails to match (but see the compilation flag
> **REG_NEWLINE** above). This flag may be used when different portions of a string are passed to
> **regexec**() and the beginning of the string should not be interpreted as the beginning of the line.

REG_NOTEOL

>   The match-end-of-line operator always fails to match (but see the compilation flag **REG_NEW-LINE** above).

Byte offsets

>   Unless **REG_NOSUB** was set for the compilation of the pattern buffer, it is possible to obtain match addressing information. *pmatch* must be dimensioned to have at least *nmatch* elements. These are filled in by **regexec**() with substring match addresses. The offsets of the subexpression starting at the *i*th open parenthesis are stored in *pmatch[i]*. The entire regular expression's match addresses are stored in *pmatch[0]*. (Note that to return the offsets of *N* subexpression matches, *nmatch* must be at least *N+1*.) Any unused structure elements will contain the value -1.

>   The *regmatch_t* structure which is the type of *pmatch* is defined in *<regex.h>*.

```
typedef struct {
regoff_t rm_so;
regoff_t rm_eo;
} regmatch_t;
```

>   Each *rm_so* element that is not -1 indicates the start offset of the next largest substring match within the string. The relative *rm_eo* element indicates the end offset of the match, which is the offset of the first character after the matching text.

POSIX error reporting

>   **regerror**() is used to turn the error codes that can be returned by both **regcomp**() and **regexec**() into error message strings.

>   **regerror**() is passed the error code, *errcode*, the pattern buffer, *preg*, a pointer to a character string buffer, *errbuf*, and the size of the string buffer, *errbuf_size*. It returns the size of the *errbuf* required to contain the null-terminated error message string. If both *errbuf* and *errbuf_size* are nonzero, *errbuf* is filled in with the first *errbuf_size - 1* characters of the error message and a terminating null byte ('\0').

POSIX pattern buffer freeing

>   Supplying **regfree**() with a precompiled pattern buffer, *preg* will free the memory allocated to the pattern buffer by the compiling process, **regcomp**().

RETURN VALUE

>   **regcomp**() returns zero for a successful compilation or an error code for failure.

>   **regexec**() returns zero for a successful match or **REG_NOMATCH** for failure.

ERRORS

>   The following errors can be returned by **regcomp**():

REG_BADBR

>   Invalid use of back reference operator.

REG_BADPAT

>   Invalid use of pattern operators such as group or list.

REG_BADRPT

>   Invalid use of repetition operators such as using '*' as the first character.

REG_EBRACE

>   Un-matched brace interval operators.

REG_EBRACK

>   Un-matched bracket list operators.

REG_ECOLLATE

>   Invalid collating element.

REG_ECTYPE
> Unknown character class name.

REG_EEND
> Nonspecific error. This is not defined by POSIX.2.

REG_EESCAPE
> Trailing backslash.

REG_EPAREN
> Un-matched parenthesis group operators.

REG_ERANGE
> Invalid use of the range operator; for example, the ending point of the range occurs prior to the starting point.

REG_ESIZE
> Compiled regular expression requires a pattern buffer larger than 64Kb. This is not defined by POSIX.2.

REG_ESPACE
> The regex routines ran out of memory.

REG_ESUBREG
> Invalid back reference to a subexpression.

ATTRIBUTES

For an explanation of the terms used in this section, see attributes(7).

Interface	Attribute	Value
regcomp(), **regexec**()	Thread safety	MT-Safe locale
regerror()	Thread safety	MT-Safe env
regfree()	Thread safety	MT-Safe

CONFORMING TO

POSIX.1-2001, POSIX.1-2008.

SEE ALSO

grep(1), regex(7)

The glibc manual section, *Regular Expressions*

COLOPHON

This page is part of release 4.10 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man-pages/.