NAME

PCRE - Perl-compatible regular expressions

#include <pcre.h>

PCRE 32-BIT API BASIC FUNCTIONS

pcre32 *pcre32_compile(PCRE_SPTR32 *pattern***, int** *options***,**
const char *******errptr*, int ****erroffset***,**
const unsigned char ****tableptr*);**

pcre32 *pcre32_compile2(PCRE_SPTR32 *pattern***, int** *options***,**
int ****errorcodeptr*,**
const unsigned char ****tableptr*);**

pcre32_extra *pcre32_study(const pcre32 ****code*, int** *options***,**
const char *******errptr*);**

void pcre32_free_study(pcre32_extra ****extra*);**

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

int pcre32_dfa_exec(const pcre32 ****code*, const pcre32_extra ****extra***,**
PCRE_SPTR32 *subject***, int** *length***, int** *startoffset***,**
int *options***, int ****ovector***, int** *ovecsize***,**
int ****workspace*, int** *wscount***);**

PCRE 32-BIT API STRING EXTRACTION FUNCTIONS

int pcre32_copy_named_substring(const pcre32 ****code*,**
PCRE_SPTR32 *subject***, int ****ovector***,**
int *stringcount***, PCRE_SPTR32** *stringname***,**
PCRE_UCHAR32 ****buffer*, int** *buffersize***);**

int pcre32_copy_substring(PCRE_SPTR32 *subject***, int ****ovector***,**
int *stringcount***, int** *stringnumber***, PCRE_UCHAR32 ****buffer***,**
int *buffersize***);**

int pcre32_get_named_substring(const pcre32 ****code*,**
PCRE_SPTR32 *subject***, int ****ovector***,**

int *stringcount*, **PCRE_SPTR32** *stringname*,
**PCRE_SPTR32 ***stringptr*);

**int pcre32_get_stringnumber(const pcre32 ***code*,
PCRE_SPTR32 *name*);

**int pcre32_get_stringtable_entries(const pcre32 ***code*,
PCRE_SPTR32 *name*, **PCRE_UCHAR32 *****first*, **PCRE_UCHAR32 *****last*);

int pcre32_get_substring(PCRE_SPTR32 *subject*, **int ***ovector*,
int *stringcount*, **int** *stringnumber*,
**PCRE_SPTR32 ***stringptr*);

int pcre32_get_substring_list(PCRE_SPTR32 *subject*,
int ***ovector*, **int *stringcount*, **PCRE_SPTR32 *****listptr*);

void pcre32_free_substring(PCRE_SPTR32 *stringptr*);

**void pcre32_free_substring_list(PCRE_SPTR32 ***stringptr*);

PCRE 32-BIT API AUXILIARY FUNCTIONS

pcre32_jit_stack *pcre32_jit_stack_alloc(int *startsize*, **int** *maxsize*);

**void pcre32_jit_stack_free(pcre32_jit_stack ***stack*);

**void pcre32_assign_jit_stack(pcre32_extra ***extra*,
pcre32_jit_callback *callback*, **void ***data*);

const unsigned char *pcre32_maketables(void);

**int pcre32_fullinfo(const pcre32 ***code*, **const pcre32_extra ***extra*,
int *what*, **void ***where*);

int pcre32_refcount(pcre32 ***code*, **int *adjust*);

int pcre32_config(int *what*, **void ***where*);

const char *pcre32_version(void);

**int pcre32_pattern_to_host_byte_order(pcre32 ***code*,

pcre32_extra ***extra*, const unsigned char ***tables***);**

PCRE 32-BIT API INDIRECTED FUNCTIONS

void *(*pcre32_malloc)(size_t);

void (*pcre32_free)(void *);

void *(*pcre32_stack_malloc)(size_t);

void (*pcre32_stack_free)(void *);

int (*pcre32_callout)(pcre32_callout_block *);

PCRE 32-BIT API 32-BIT-ONLY FUNCTION

int pcre32_utf32_to_host_byte_order(PCRE_UCHAR32 ***output*,**
PCRE_SPTR32 *input***, int** *length***, int ***byte_order***,**
int *keep_boms***);**

THE PCRE 32-BIT LIBRARY

Starting with release 8.32, it is possible to compile a PCRE library that supports 32-bit character strings, including UTF-32 strings, as well as or instead of the original 8-bit library. This work was done by Christian Persch, based on the work done by Zoltan Herczeg for the 16-bit library. All three libraries contain identical sets of functions, used in exactly the same way. Only the names of the functions and the data types of their arguments and results are different. To avoid over-complication and reduce the documentation maintenance load, most of the PCRE documentation describes the 8-bit library, with only occasional references to the 16-bit and 32-bit libraries. This page describes what is different when you use the 32-bit library.

WARNING: A single application can be linked with all or any of the three libraries, but you must take care when processing any particular pattern to use functions from just one library. For example, if you want to study a pattern that was compiled with **pcre32_compile()**, you must do so with **pcre32_study()**, not **pcre_study()**, and you must free the study data with **pcre32_free_study()**.

THE HEADER FILE

There is only one header file, **pcre.h**. It contains prototypes for all the functions in all libraries, as well as definitions of flags, structures, error codes, etc.

THE LIBRARY NAME

In Unix-like systems, the 32-bit library is called **libpcre32**, and can normally be accesss by adding **-lpcre32** to the command for linking an application that uses PCRE.

STRING TYPES

In the 8-bit library, strings are passed to PCRE library functions as vectors of bytes with the C type "char *". In the 32-bit library, strings are passed as vectors of unsigned 32-bit quantities. The macro PCRE_UCHAR32 specifies an appropriate data type, and PCRE_SPTR32 is defined as "const PCRE_UCHAR32 *". In very many environments, "unsigned int" is a 32-bit data type. When PCRE is built, it defines PCRE_UCHAR32 as "unsigned int", but checks that it really is a 32-bit data type. If it is not, the build fails with an error message telling the maintainer to modify the definition appropriately.

STRUCTURE TYPES

The types of the opaque structures that are used for compiled 32-bit patterns and JIT stacks are **pcre32** and **pcre32_jit_stack** respectively. The type of the user-accessible structure that is returned by **pcre32_study()** is **pcre32_extra**, and the type of the structure that is used for passing data to a callout function is **pcre32_callout_block**. These structures contain the same fields, with the same names, as their 8-bit counterparts. The only difference is that pointers to character strings are 32-bit instead of 8-bit types.

32-BIT FUNCTIONS

For every function in the 8-bit library there is a corresponding function in the 32-bit library with a name that starts with **pcre32_** instead of **pcre_**. The prototypes are listed above. In addition, there is one extra function, **pcre32_utf32_to_host_byte_order()**. This is a utility function that converts a UTF-32 character string to host byte order if necessary. The other 32-bit functions expect the strings they are passed to be in host byte order.

The *input* and *output* arguments of **pcre32_utf32_to_host_byte_order()** may point to the same address, that is, conversion in place is supported. The output buffer must be at least as long as the input.

The *length* argument specifies the number of 32-bit data units in the input string; a negative value specifies a zero-terminated string.

If *byte_order* is NULL, it is assumed that the string starts off in host byte order. This may be changed by byte-order marks (BOMs) anywhere in the string (commonly as the first character).

If *byte_order* is not NULL, a non-zero value of the integer to which it points means that the input starts off in host byte order, otherwise the opposite order is assumed. Again, BOMs in the string can change this. The final byte order is passed back at the end of processing.

If *keep_boms* is not zero, byte-order mark characters (0xfeff) are copied into the output string. Otherwise they are discarded.

The result of the function is the number of 32-bit units placed into the output buffer, including the zero terminator if the string was zero-terminated.

SUBJECT STRING OFFSETS

The lengths and starting offsets of subject strings must be specified in 32-bit data units, and the offsets within subject strings that are returned by the matching functions are in also 32-bit units rather than bytes.

NAMED SUBPATTERNS

The name-to-number translation table that is maintained for named subpatterns uses 32-bit characters. The **pcre32_get_stringtable_entries()** function returns the length of each entry in the table as the number of 32-bit data units.

OPTION NAMES

There are two new general option names, PCRE_UTF32 and PCRE_NO_UTF32_CHECK, which correspond to PCRE_UTF8 and PCRE_NO_UTF8_CHECK in the 8-bit library. In fact, these new options define the same bits in the options word. There is a discussion about the validity of UTF-32 strings in the **pcreunicode** page.

For the **pcre32_config()** function there is an option PCRE_CONFIG_UTF32 that returns 1 if UTF-32 support is configured, otherwise 0. If this option is given to **pcre_config()** or **pcre16_config()**, or if the PCRE_CONFIG_UTF8 or PCRE_CONFIG_UTF16 option is given to **pcre32_config()**, the result is the PCRE_ERROR_BADOPTION error.

CHARACTER CODES

In 32-bit mode, when PCRE_UTF32 is not set, character values are treated in the same way as in 8-bit, non UTF-8 mode, except, of course, that they can range from 0 to 0x7fffffff instead of 0 to 0xff. Character types for characters less than 0xff can therefore be influenced by the locale in the same way as before. Characters greater than 0xff have only one case, and no "type" (such as letter or digit).

In UTF-32 mode, the character code is Unicode, in the range 0 to 0x10ffff, with the exception of values in the range 0xd800 to 0xdfff because those are "surrogate" values that are ill-formed in UTF-32.

A UTF-32 string can indicate its endianness by special code knows as a byte-order mark (BOM). The PCRE functions do not handle this, expecting strings to be in host byte order. A utility function called **pcre32_utf32_to_host_byte_order()** is provided to help with this (see above).

ERROR NAMES

The error PCRE_ERROR_BADUTF32 corresponds to its 8-bit counterpart. The error PCRE_ERROR_BADMODE is given when a compiled pattern is passed to a function that processes

patterns in the other mode, for example, if a pattern compiled with **pcre_compile()** is passed to **pcre32_exec()**.

There are new error codes whose names begin with PCRE_UTF32_ERR for invalid UTF-32 strings, corresponding to the PCRE_UTF8_ERR codes for UTF-8 strings that are described in the section entitled "Reason codes for invalid UTF-8 strings" in the main **pcreapi** page. The UTF-32 errors are:

PCRE_UTF32_ERR1 Surrogate character (range from 0xd800 to 0xdfff)
PCRE_UTF32_ERR2 Non-character
PCRE_UTF32_ERR3 Character > 0x10ffff

ERROR TEXTS

If there is an error while compiling a pattern, the error text that is passed back by **pcre32_compile()** or **pcre32_compile2()** is still an 8-bit character string, zero-terminated.

CALLOUTS

The *subject* and *mark* fields in the callout block that is passed to a callout function point to 32-bit vectors.

TESTING

The **pcretest** program continues to operate with 8-bit input and output files, but it can be used for testing the 32-bit library. If it is run with the command line option **-32**, patterns and subject strings are converted from 8-bit to 32-bit before being passed to PCRE, and the 32-bit library functions are used instead of the 8-bit ones. Returned 32-bit strings are converted to 8-bit for output. If both the 8-bit and the 16-bit libraries were not compiled, **pcretest** defaults to 32-bit and the **-32** option is ignored.

When PCRE is being built, the **RunTest** script that is called by "make check" uses the **pcretest -C** option to discover which of the 8-bit, 16-bit and 32-bit libraries has been built, and runs the tests appropriately.

NOT SUPPORTED IN 32-BIT MODE

Not all the features of the 8-bit library are available with the 32-bit library. The C++ and POSIX wrapper functions support only the 8-bit library, and the **pcregrep** program is at present 8-bit only.

AUTHOR

Philip Hazel
University Computing Service
Cambridge CB2 3QH, England.

REVISION

Last updated: 12 May 2013