NAME

PCRE - Perl-compatible regular expressions

#include <pcre.h>

PCRE 16-BIT API BASIC FUNCTIONS

pcre16 *pcre16_compile(PCRE_SPTR16 *pattern***, int** *options***,**
const char ** *errptr***, int *** *erroffset***,**
const unsigned char * *tableptr***);**

pcre16 *pcre16_compile2(PCRE_SPTR16 *pattern***, int** *options***,**
int * *errorcodeptr***,**
const char ** *errptr***, int *** *erroffset***,**
const unsigned char * *tableptr***);**

pcre16_extra *pcre16_study(const pcre16 * *code***, int** *options***,**
const char ** *errptr***);**

void pcre16_free_study(pcre16_extra * *extra***);**

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

int pcre16_dfa_exec(const pcre16 * *code***, const pcre16_extra *** *extra***,**
PCRE_SPTR16 *subject***, int** *length***, int** *startoffset***,**
int *options***, int *** *ovector***, int** *ovecsize***,**
int * *workspace***, int** *wscount***);**

PCRE 16-BIT API STRING EXTRACTION FUNCTIONS

int pcre16_copy_named_substring(const pcre16 * *code***,**
PCRE_SPTR16 *subject***, int *** *ovector***,**
int *stringcount***, PCRE_SPTR16** *stringname***,**
PCRE_UCHAR16 * *buffer***, int** *buffersize***);**

int pcre16_copy_substring(PCRE_SPTR16 *subject***, int *** *ovector***,**
int *stringcount***, int** *stringnumber***, PCRE_UCHAR16 *** *buffer***,**
int *buffersize***);**

int pcre16_get_named_substring(const pcre16 * *code***,**

PCRE_SPTR16 *subject*, **int** **ovector*,
int *stringcount*, **PCRE_SPTR16** *stringname*,
PCRE_SPTR16 **stringptr*);

int pcre16_get_stringnumber(const pcre16 **code*,
PCRE_SPTR16 *name*);

int pcre16_get_stringtable_entries(const pcre16 **code*,
PCRE_SPTR16 *name*, **PCRE_UCHAR16** ***first*, **PCRE_UCHAR16** ***last*);

int pcre16_get_substring(PCRE_SPTR16 *subject*, **int** **ovector*,
int *stringcount*, **int** *stringnumber*,
PCRE_SPTR16 **stringptr*);

int pcre16_get_substring_list(PCRE_SPTR16 *subject*,
int **ovector*, **int** *stringcount*, **PCRE_SPTR16** ***listptr*);

void pcre16_free_substring(PCRE_SPTR16 *stringptr*);

void pcre16_free_substring_list(PCRE_SPTR16 **stringptr*);

PCRE 16-BIT API AUXILIARY FUNCTIONS

pcre16_jit_stack *pcre16_jit_stack_alloc(int *startsize*, **int** *maxsize*);

void pcre16_jit_stack_free(pcre16_jit_stack **stack*);

void pcre16_assign_jit_stack(pcre16_extra **extra*,
pcre16_jit_callback *callback*, **void** **data*);

const unsigned char *pcre16_maketables(void);

int pcre16_fullinfo(const pcre16 **code*, **const pcre16_extra** **extra*,
int *what*, **void** **where*);

int pcre16_refcount(pcre16 **code*, **int** *adjust*);

int pcre16_config(int *what*, **void** **where*);

const char *pcre16_version(void);

int pcre16_pattern_to_host_byte_order(pcre16 ****code*,**
pcre16_extra ****extra*, const unsigned char ****tables***);**

PCRE 16-BIT API INDIRECTED FUNCTIONS

void *(*pcre16_malloc)(size_t);

void (*pcre16_free)(void *);

void *(*pcre16_stack_malloc)(size_t);

void (*pcre16_stack_free)(void *);

int (*pcre16_callout)(pcre16_callout_block *);

PCRE 16-BIT API 16-BIT-ONLY FUNCTION

int pcre16_utf16_to_host_byte_order(PCRE_UCHAR16 ****output*,**
PCRE_SPTR16 *input***, int** *length***, int ****byte_order***,**
int *keep_boms***);**

THE PCRE 16-BIT LIBRARY

Starting with release 8.30, it is possible to compile a PCRE library that supports 16-bit character strings, including UTF-16 strings, as well as or instead of the original 8-bit library. The majority of the work to make this possible was done by Zoltan Herczeg. The two 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 library. This page describes what is different when you use the 16-bit library.

WARNING: A single application can be linked with both 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 **pcre16_compile()**, you must do so with **pcre16_study()**, not **pcre_study()**, and you must free the study data with **pcre16_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 16-bit library is called **libpcre16**, and can normally be accesss by adding **-lpcre16** 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 16-bit library, strings are passed as vectors of unsigned 16-bit quantities. The macro PCRE_UCHAR16 specifies an appropriate data type, and PCRE_SPTR16 is defined as "const PCRE_UCHAR16 *". In very many environments, "short int" is a 16-bit data type. When PCRE is built, it defines PCRE_UCHAR16 as "unsigned short int", but checks that it really is a 16-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 16-bit patterns and JIT stacks are **pcre16** and **pcre16_jit_stack** respectively. The type of the user-accessible structure that is returned by **pcre16_study()** is **pcre16_extra**, and the type of the structure that is used for passing data to a callout function is **pcre16_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 16-bit instead of 8-bit types.

16-BIT FUNCTIONS

For every function in the 8-bit library there is a corresponding function in the 16-bit library with a name that starts with **pcre16_** instead of **pcre_**. The prototypes are listed above. In addition, there is one extra function, **pcre16_utf16_to_host_byte_order()**. This is a utility function that converts a UTF-16 character string to host byte order if necessary. The other 16-bit functions expect the strings they are passed to be in host byte order.

The *input* and *output* arguments of **pcre16_utf16_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 16-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 16-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 16-bit data units, and the offsets within subject strings that are returned by the matching functions are in also 16-bit units rather than bytes.

NAMED SUBPATTERNS

The name-to-number translation table that is maintained for named subpatterns uses 16-bit characters. The **pcre16_get_stringtable_entries()** function returns the length of each entry in the table as the number of 16-bit data units.

OPTION NAMES

There are two new general option names, PCRE_UTF16 and PCRE_NO_UTF16_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-16 strings in the **pcreunicode** page.

For the **pcre16_config()** function there is an option PCRE_CONFIG_UTF16 that returns 1 if UTF-16 support is configured, otherwise 0. If this option is given to **pcre_config()** or **pcre32_config()**, or if the PCRE_CONFIG_UTF8 or PCRE_CONFIG_UTF32 option is given to **pcre16_config()**, the result is the PCRE_ERROR_BADOPTION error.

CHARACTER CODES

In 16-bit mode, when PCRE_UTF16 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 0xffff 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-16 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 used in pairs to encode values greater than 0xffff.

A UTF-16 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 **pcre16_utf16_to_host_byte_order()** is provided to help with this (see above).

ERROR NAMES

The errors PCRE_ERROR_BADUTF16_OFFSET and PCRE_ERROR_SHORTUTF16 correspond to

their 8-bit counterparts. 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 **pcre16_exec()**.

There are new error codes whose names begin with PCRE_UTF16_ERR for invalid UTF-16 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-16 errors are:

PCRE_UTF16_ERR1	Missing low surrogate at end of string
PCRE_UTF16_ERR2	Invalid low surrogate follows high surrogate
PCRE_UTF16_ERR3	Isolated low surrogate
PCRE_UTF16_ERR4	Non-character

ERROR TEXTS

If there is an error while compiling a pattern, the error text that is passed back by **pcre16_compile()** or **pcre16_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 16-bit vectors.

TESTING

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

Not all the features of the 8-bit library are available with the 16-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