LibLouisAPH

Reference

API

Functions

LibLouisAPH keeps a current path list that is used to search for table and conversion files. It is a string containing directory names separated by the path separator. The path separator is `:` for POSIX systems (Linux and Macs), and `;` for Windows systems.

```
void louis_get_version(char *version,
                       const int version_max)
```

`version` Stored a copy of the library's version.

`version_max` The maximum number of characters that can be stored in `version`.

```
void louis_set_log_callback(void (*callback)(const int level,
                                             const char *message))
```

`callback` Function called when `message` is written to log with `level`, may be `NULL`.

```
int louis_get_paths(unsigned short *paths,
                    const int paths_max)
```

`paths` Stored a copy of the current path list.

`paths_max` The maximum number of characters that can be stored in `paths`.

Returns the number of characters stored in `paths`.

```
int louis_set_paths(const unsigned short *paths)
```

`paths` Sets the current path list. Clears the current path list if `paths` is `NULL`.

Returns the number of characters stored in the current path list, or 0 if an error occurred

```
int louis_add_paths(const unsigned short *paths)
```

`paths` Appended to the current path list with path separator.

Returns the length of the current path list.

```
int louis_translate_forward(unsigned short *dots,
                            const int dots_len,
                            const unsigned short *chars,
                            const int chars_len,
                            const char *table_names,
                            const char *conversion_name,
                            int *chars_to_dots_map,
                            int *dots_to_chars_map,
                            int *cursor)
```

`dots`	Stored the resulting forward translation in UTF-16LE.
`dots_len`	Maximum number of characters that can be stored in dots.
`chars`	Text to be translated. The text must be UTF-16LE.
`chars_len`	Number of characters in chars.
`table_names`	Table file name list used for translation.
`conversion_name`	Conversion file name.
`chars_to_dots_map`	Buffer to store mapping from chars to dots, may be `NULL` if `dots_to_chars_map` is also `NULL`.
`dots_to_chars_map`	Buffer to store mapping from dots to chars, may be `NULL` if `chars_to_dots_map` is also `NULL`.
`cursor`	Stores the pre-translation cursor position and reset to the post-translation cursor position. May be `NULL`.

Returns the length of the resulting forward translation, 0 if no processing was done and no forward translation was performed, and -1 if an error occurred.

```
int louis_translate_backward(unsigned short *chars,
                             const int chars_len,
                             const unsigned short *dots,
                             const int dots_len,
                             const char *table_names,
                             const char *conversion_name,
                             int *chars_to_dots_map,
                             int *dots_to_chars_map,
                             int *cursor)
```

`chars`	Stored the resulting backward translation UTF-16LE .
`chars_len`	Maximum number of characters that can be stored in dots.
`dots`	Text to be translated. The text must be UTF-16LE.
`dots_len`	Number of characters in chars.
`table_names`	Table file name list used for translation.
`conversion_name`	Conversion file name.
`chars_to_dots_map`	Buffer to store mapping from chars to dots, may be `NULL` if `dots_to_chars_map` is also `NULL`.
`dots_to_chars_map`	Buffer to store mapping from dots to chars, may be `NULL` if `chars_to_dots_map` is also `NULL`.
`cursor`	Stores the pre-translation cursor position and reset to the post-translation cursor position. May be `NULL`.

Returns the length of the resulting backward translation, 0 if no processing was done and no backward translation was performed, and -1 if an error occurred.

```
int louis_convert_to(unsigned short *chars,
                     const int chars_len)
```

`chars`	Characters to be converted.
`chars_len`	The number of characters in `chars`.
`conversion_name`	Conversion file name.

Returns 1 if successful, 0 if an error occurred.

```
int louis_convert_from(unsigned short *dots,
                       const int dots_len)
```

`dot`	Characters to be converted.
`dots_len`	The number of characters in `dot`.
`conversion_name`	Conversion file name.

Returns 1 if successful, 0 if an error occurred.

Control Characters

The control characters can be changed in the table using the **config** control opcode.

U+F001	USER
U+F002	BEGIN
U+F003	END

Table

When compiling, LibLouisAPH maintains a current rule attributes which is an attribute *INTEGER* that is set for every **rule** or **match**. LibLouisAPH also maintains current rule filters which represents *PATTERN*s that are applied to every **rule** or **match**. The current rule filters are two *FILTER*s, both forward and backwards, each containing two *PATTERN*s that are checked against the characters that are before and after a *TARGET*.

Blank lines and lines starting with # are ignored. Also any characters after an opcode's defined parameters are ignored, unless there are a variable number of parameters. All table files are required to be in UTF-8 format.

Definitions:

CHAR	A single UTF16 character.
CHARS	A string of UTF16 characters.
DOT	A string of digits from 1 to 8 or a single UTF16 character between U+2800 to U+28FF.
DOTS	A combination of strings of digits from 1 to 8 separated by a hyphen, or UTF16 characters between U+2800 to U+28FF.
CHARDOTS	*CHARS* that may contain *DOTS* surrounded by semicolons, that are following the escape character \.
TARGET	*CHARDOTS* that determine whether a rule is to be applied during forward translation. Acts as *REPLACEMENT* during backward translation.
REPLACEMENT	*CHARDOTS* that are to replace *TARGET* when applying a rule during forward translation. Acts as *TARGET* during backward translation. Note that for rule type trans, *DOTS* are expected instead of *CHARDOTS*.
INTEGER	An integer that may be represented as digits, a variable prefixed with a $, or bit## representing a single bit. Digits may be prefixed with a 0 for octal base and 0x for hexadecimal base.
PATTERN	An expression used to filter rules when matching. *PATTERN* may be just a hyphen to specify that no filtering is to be done.
FILTER	Two *PATTERN*s that are checked against the characters that are before and after a given *TARGET*.

Main Opcodes

rule *[OPTIONS] TYPE TARGET REPLACEMENT*

Defines a translation from *TARGET* to *REPLACEMENT*.

OPTIONS

-forward	Forward translation only.
-backward	Backward translation only.
-after	Placed after similar rules.

TYPE

init	
premode	
postmode	
pretrans	
trans	*DOTS* instead of *CHARDOTS* for *REPLACEMENT*.
posttrans	
fini	

match *[OPTIONS] TYPE PATTERN TARGET PATTERN PATTERN REPLACEMENT PATTERN*

Like **rule**, but includes *PATTERN*s to further filtering. *OPTIONS* and *TYPE* are the same as rule. The first *PATTERN* is checked against the characters before *TARGET* during forward translation. The second *PATTERN* is checked against the characters after *TARGET* during forward translation. The third *PATTERN* is checked against the characters before *REPLACEMENT* during backward translation. The fourth *PATTERN* is checked against the characters after *REPLACEMENT* during backward translation. The equal sign may be used for any *PATTERN* to designate that the corresponding portion of the current rule filters should be used.

chars *CHARS INTEGER [INTEGER...]*

Takes the current attribute for each *CHARS* and it is or'ed with *INTEGER*. Multiple *INTEGER*s may be used, with each *INTEGER* being or'ed to the current attribute, and the list can be terminated with a single #.

attrs =|+|- *INTEGER*

Modifies the current rule attributes for every rule that follows.

=	Sets the current rule attributes to *INTEGER*.
+	Sets the bits of the current rule attributes that are set in *INTEGER*.
-	Unsets the bits of the current rule attributes that are set in *INTEGER*.

join *INTEGER CHAR CHAR [INTEGER [INTEGER]]*

Connects both *CHARs* at join index *INTEGER*. If present, the bits of the first *INTEGER* are set in the first *CHAR* attributes. If present, the bits of the second *INTEGER* are set in the second *CHAR* attributes.

mode *NAME BEGIN END [WORD TERM SYMBOL LENGTH]*

Sets the indicators for a mode named *NAME*. *NAME* is used to specify the mode when using streaming control characters while translating. There are three special modes that are processed separately:

nocontract

capital

numeric

Supplementary Opcodes

set *NAME INTEGER*

Creates a variable *NAME* with value *INTEGER*. A variable can be used in place of an *INTEGER* via $*NAME*.

pattern *NAME PATTERN*

Defines a subpattern *PATTERN* with *NAME* that can used inside other following patterns.

filter *NAME PATTERN PATTERN*

Creates a filter *NAME*. The first *PATTERN* is checked against a characters before a *TARGET* or *REPLACEMENT*, and the second *PATTERN* is checked against the characters after a *TARGET* or *REPLACEMENT*.

use *FILTER FILTER*

Sets the current rule filters that are applied to every rule that follows. The first *FILTER* is used when translating forward; can be just a hyphen to specify no filtering. The second *FILTER* is used when translating backward; can be just a hyphen or left blank to specify no filtering. Both *FILTER*s can be left blank to specify no filtering.

uses *FILTER FILTER* **rule|match**

Overrides the current rule filters one time for **rule** or **match**. *FILTER* can be just a hyphen to specify no filtering.

macro *NAME*
LINES
...
eom
@NAME [CHARS...]

Defines a macro *NAME* that represents the *LINES* between **macro** and **eom**. The macro can be used via *@NAME*. If present, the *CHARS* replace occurrences of $1, $2, $3, up to $9, in the order they are listed, in *LINES*. Macros cannot define other macros, but can use other macros, provided no macro is used recursively.

include *FILE*

Compiles the rules in *FILE* in place as if they were part of the current file being compiled.

config

Miscellaneous settings.

attrs_chars *CHARS*	String *CHARS* used to represent specific character attribute bits in patterns.
control *TYPE CHAR*	Sets streaming control character *TYPE* to *CHAR*. *TYPE*: user|begin|end|hard|soft|modifier|internal

Pattern Expressions

Definitions:

CHAR	Same as table.
CHARDOTS	Same as table.
ATTR	*CHAR* that represents a specific attribute, more specifically the bit the signifies a specific character attribute that was set via the **chars** opcode. By default, bits 0 to 31 are represented by the characters: 0123456789ABCDEFGHIJKLMNOPQRSTUV. These can be changed using the **config** opcode with attrs_chars.
ATTRS	String of *ATTR*s
NAME	Name of a *PATTERN* that was previously defined using the **pattern** opcode.
EXPR	Sequence of one or more of the following expressions described below.

Expressions:

CHARDOTS		Match exactly the string *CHARDOTS*.
\	*CHAR*	Escape; match literal *CHAR*.
[ ]	[*CHARSDOTS*]	Match one of the characters in the string *CHARSDOTS*.
.		Match any character.
^		Match beginning or end of input.
%	%*ATTR*	Match *ATTR*.
%[ ]	%[*ATTRS*]	Match any of *ATTRS*.
( )	(*EXPR...*)	Treat the group of *EXPR*s as one *EXPR*.
!	!*EXPR*	Not *EXPR*.
*	*EXPR**	Zero or more of *EXPR*.
+	*EXPR*+	One or more of *EXPR.*
?	*EXPR*?	Zero or one of *EXPR.*
|	*EXPR*|*EXPR*	Either first *EXPR* or second *EXPR*.
@[ ]	@[*NAME*]	Use subpattern *NAME* during compilation.
@{ }	@{*NAME*}	**EXPERIMENTAL** Use subpattern *NAME* during checking.

Conversion

Blank lines and lines starting with # are ignored. All conversion files are required to be in UTF-8 format.

Definitions:

CHAR	Same as table.
DOT	Same as table.

convert *DOT CHAR*

Sets the conversion of *DOT* to *CHAR*.

unknown *CHAR*

Sets the conversion of any unknown character to CHAR.