NAME

String::Escape – Backslash escapes, quoted phrase, word elision, etc.

SYNOPSIS

This module provides a flexible calling interface to some frequently-performed string conversion functions, including applying and removing backslash escapes like \n and \t, wrapping and removing double-quotes, and truncating to fit within a desired length.

```
use String::Escape qw( printable unprintable );
# Convert control, high-bit chars to \n or \xxx escapes
$output = printable($value);
# Convert escape sequences back to original chars
$value = unprintable($input);

use String::Escape qw( elide );
# Shorten strings to fit, if necessary
foreach (@_) { print elide( $_, 79 ) . "\n"; }

use String::Escape qw( string2list list2string );
# Pack and unpack simple lists by quoting each item
$list = list2string( @list );
@list = string2list( $list );

use String::Escape qw( escape );
# Defer selection of escaping routines until runtime
$escape_name = $use_quotes ? 'qprintable' : 'printable';
@escaped = escape($escape_name, @values);
```

INTERFACE

All of the public functions described below are available as optional exports.

You can either import the specific functions you want, or import only the `escape()` function and pass it the names of the functions to invoke.

Quoting

Each of these functions takes a single simple scalar argument and returns its escaped (or unescaped) equivalent.

quote($value) : `$escaped`
: Add double quote characters to each end of the string.

unquote($value) : `$escaped`
: If the string both begins and ends with double quote characters, they are removed, otherwise the string is returned unchanged.

quote_non_words($value) : `$escaped`
: As above, but only quotes empty, punctuated, and multiword values; simple values consisting of alphanumerics without special characters are not quoted.

singlequote($value) : `$escaped`
: Add single quote characters to each end of the string.

unsinglequote($value) : `$escaped`
: If the string both begins and ends with single quote characters, they are removed, otherwise the string is returned unchanged.

Backslash Escaping Functions

Each of these functions takes a single simple scalar argument and returns its escaped (or unescaped) equivalent.

These functions recognize common whitespace sequences `\r`, `\n`, and `\t`, as well as hex escapes `\x4F` and ocatal `\020`.

When escaping, alphanumeric characters and most punctuation is passed through unchanged; only the return, newline, tab, backslash, dollar, at sign and unprintable control and high-bit characters are escaped.

backslash($value) : `$escaped`
Converts special characters to their backslash-escaped equivalents.

unbackslash($value) : `$escaped`
Converts backslash escape sequences in a string back to their original characters.

qqbackslash($value) : `$escaped`
Converts special characters to their backslash-escaped equivalents and then wraps the results with double quotes.

unqqbackslash($value) : `$escaped`
Strips surrounding double quotes then converts backslash escape sequences back to their original characters.

Here are a few examples:

•

```
print backslash( "\tNow is the time\nfor all good folks\n" );

\tNow is the time\nfor all good folks\n
```

•

```
print unbackslash( '\\tNow is the time\\nfor all good folks\\n' );

      Now is the time
for all good folks
```

Legacy Backslash Functions

In addition to the four functions listed above, there is a corresponding set which use a slightly different set of escape sequences.

These functions do not support as many escape sequences and use a non-standard format for hex escapes. In general, the above `backslash()` functions are recommended, while these functions are retained for legacy compatibility purposes.

printable($value) : `$escaped`
Converts return, newline, tab, backslash and unprintable characters to their backslash-escaped equivalents.

unprintable($value) : `$escaped`
Converts backslash escape sequences in a string back to their original value.

qprintable($value) : `$escaped`
Converts special characters to their backslash-escaped equivalents and then wraps the results with double quotes.

(Note that this is *not* MIME quoted-printable encoding.)

unqprintable($value) : `$escaped`
Strips surrounding double quotes then converts backslash escape sequences back to their original value.

Other Backslash Functions

In addition to the functions listed above, there is also one function that mirrors the behavior of Perl's built-in `quotemeta()` function.

unquotemeta($value) : `$escaped`
Strips out backslashes before any character.

Elision Function

This function extracts the leading portion of a provided string and appends ellipsis if it's longer than the desired maximum excerpt length.

elide($string) : `$elided_string`
elide($string, `$length`) : `$elided_string`
elide($string, `$length`, `$word_boundary_strictness`) : `$elided_string`
elide($string, `$length`, `$word_boundary_strictness`, `$elipses`) : `$elided_string`
Return a single-quoted, shortened version of the string, with ellipsis.

If the original string is shorter than `$length`, it is returned unchanged. At most `$length` characters are returned; if called with a single argument, `$length` defaults to `$DefaultLength`.

Up to `$word_boundary_strictness` additional characters may be ommited in order to make the elided portion end on a word boundary; you can pass 0 to ignore word boundaries. If not provided, `$word_boundary_strictness` defaults to `$DefaultStrictness`.

`$Elipses`
The string of characters used to indicate the end of the excerpt. Initialized to '...'.

`$DefaultLength`
The default target excerpt length, used when the elide function is called with a single argument. Initialized to 60.

`$DefaultStrictness`
The default word-boundary flexibility, used when the elide function is called without the third argument. Initialized to 10.

Here are a few examples:

•

```
$string = 'foo bar baz this that the other';

print elide( $string, 12 );
# foo bar...

print elide( $string, 12, 0 );
# foo bar b...

print elide( $string, 100 );
# foo bar baz this that the other
```

escape()

These functions provide for the registration of string-escape specification names and corresponding functions, and then allow the invocation of one or several of these functions on one or several source string values.

escape($escapes, `$value`) : `$escaped_value`
escape($escapes, `@values`) : `@escaped_values`
Returns an altered copy of the provided values by looking up the escapes string in a registry of string-modification functions.

If called in a scalar context, operates on the single value passed in; if called in a list contact, operates identically on each of the provided values.

Space-separated compound specifications like 'quoted uppercase' are expanded to a list of functions to be applied in order.

Valid escape specifications are:

one of the keys defined in `%Escapes`
: The coresponding specification will be looked up and used.

a sequence of names separated by whitespace,
: Each name will be looked up, and each of the associated functions will be applied successively, from left to right.

a reference to a function
: The provided function will be called on with each value in turn.

a reference to an array
: Each item in the array will be expanded as provided above.

A fatal error will be generated if you pass an unsupported escape specification, or if the function is called with multiple values in a scalar context.

String::Escape::names() : `@defined_escapes`
: Returns a list of defined escape specification strings.

String::Escape::add(`$escape_name`, \&escape_function);
: Add a new escape specification and corresponding function.

By default, all of the public functions described below are available as named escape commands, as well as the following built-in functions:

- none: Return the string unchanged.
- uppercase: Calls the built-in uc function.
- lowercase: Calls the built-in lc function.
- initialcase: Calls the built-in lc and ucfirst functions.

Here are a few examples:

- ```
  print escape('qprintable', "\tNow is the time\nfor all good folks\n" );

    "\tNow is the time\nfor all good folks\n"
  ```
- ```
  print escape('uppercase qprintable', "\tNow is the time\nfor all good folks\n" );

    "\tNOW IS THE TIME\nFOR ALL GOOD FOLKS\n"
  ```
- ```
  print join '--', escape('printable', "\tNow is the time\n", "for all good folks\n" );

    \tNow is the time\n--for all good folks\n
  ```
- You can add more escaping functions to the supported set by calling *add()*.

  ```
  String::Escape::add( 'html', \&HTML::Entities::encode_entities );

  print escape('html', "AT&T" );

    AT&T
  ```

Space-separated Lists and Hashes

`@words` = string2list(`$space_separated_phrases`);
: Converts a space separated string of words and quoted phrases to an array;

`$space_sparated_string` = list2string(`@words`);
: Joins an array of strings into a space separated string of words and quoted phrases;
  ```

`%hash` = string2hash(`$string`);
Converts a space separated string of equal-sign-associated key=value pairs into a simple hash.

`$string` = hash2string(`%hash`);
Converts a simple hash into a space separated string of equal-sign-associated key=value pairs.

`%hash` = list2hash(`@words`);
Converts an array of equal-sign-associated key=value strings into a simple hash.

`@words` = hash2list(`%hash`);
Converts a hash to an array of equal-sign-associated key=value strings.

Here are a few examples:

- ```
  print list2string('hello', 'I move next march');

    hello "I move next march"
  ```
- ```
  @list = string2list('one "second item" 3 "four\nlines\nof\ntext"');

  print $list[1];

    second item
  ```
- ```
  print hash2string( 'foo' => 'Animal Cities', 'bar' => 'Cheap' );

    foo="Animal Cities" bar=Cheap
  ```
- ```
  %hash = string2hash('key=value "undefined key" words="the cat in the hat"');

  print $hash{'words'};

    the cat in the hat

  print exists $hash{'undefined_key'} and ! defined $hash{'undefined_key'};

    1
  ```

SEE ALSO

Numerous modules provide collections of string escaping functions for specific contexts.

The string2list function is similar to to the quotewords function in the standard distribution; see Text::ParseWords.

Use other packages to stringify more complex data structures; see Storable, Data::Dumper, or other similar package.

BUGS

The following issues or changes are under consideration for future releases:

- Does this problem with the \r character only show up on Windows? (And is it, in fact, a feature rather than a bug?)

  ```
  http://rt.cpan.org/Public/Bug/Display.html?id=19766
  ```
- Consider changes to word parsing in string2list: Perhaps use \b word-boundary test in elide's regular expression rather than \s|\Z? Perhaps quotes embedded in a word (eg: a@"!a) shouldn't cause phrase breaks?
- Check for possible problems in the use of printable escaping functions and list2hash. For example, are the encoded strings for hashes with high-bit characters in their keys properly unquoted and unescaped?
- We should allow escape specifications to contain = signs and optional arguments, so that users can request certain string lengths with `escape("lowercase elide=20 quoted", @_`.

VERSION

This is version 2010.002.

INSTALLATION

This package should run on any standard Perl 5 installation.

To install this package, download the distribution from a CPAN mirror, unpack the archive file, and execute the standard ‘‘perl Makefile.PL’’, ‘‘make test’’, ‘‘make install’’ sequence or your local equivalent.

SUPPORT

Once installed, this module's documentation is available as a manual page via `perldoc String::Escape` or on CPAN sites such as `http://search.cpan.org/dist/String-Escape`.

If you have questions or feedback about this module, please feel free to contact the author at the address shown below. Although there is no formal support program, I do attempt to answer email promptly. Bug reports that contain a failing test case are greatly appreciated, and suggested patches will be promptly considered for inclusion in future releases.

You can report bugs and request features via the CPAN web tracking system at `http://rt.cpan.org/NoAuth/ReportBug.html?Queue=String-Escape` or by sending mail to `bug-string-escape at rt.cpan.org`.

If you've found this module useful or have feedback about your experience with it, consider sharing your opinion with other Perl users by posting your comment to CPAN's ratings system (`http://cpanratings.perl.org/rate/?distribution=String-Escape`).

For more general discussion, you may wish to post a message on PerlMonks (`http://perlmonks.org/?node=Seekers%20of%20Perl%20Wisdom`) or on the comp.lang.perl.misc newsgroup (`http://groups.google.com/group/comp.lang.perl.misc/topics`).

AUTHOR

Matthew Simon Cavalletto, `<simonm at cavalletto.org>`

Initial versions developed at Evolution Online Systems with Eleanor J. Evans and Jeremy G. Bishop.

LICENSE