NAME

Text::ParseWords - parse text into an array of tokens or array of arrays

SYNOPSIS

```
use Text::ParseWords;
@lists = nested_quotewords($delim, $keep, @lines);
@words = quotewords($delim, $keep, @lines);
@words = shellwords(@lines);
@words = parse_line($delim, $keep, $line);
@words = old_shellwords(@lines); # DEPRECATED!
```

DESCRIPTION

The &nested_quotewords() and &quotewords() functions accept a delimiter (which can be a regular expression) and a list of lines and then breaks those lines up into a list of words ignoring delimiters that appear inside quotes. &quotewords() returns all of the tokens in a single long list, while &nested_quotewords() returns a list of token lists corresponding to the elements of @lines. &parse_line() does tokenizing on a single string. The &*quotewords() functions simply call &parse_line(), so if you're only splitting one line you can call &parse_line() directly and save a function call.

The $keep argument is a boolean flag. If true, then the tokens are split on the specified delimiter, but all other characters (quotes, backslashes, etc.) are kept in the tokens. If $keep is false then the &*quotewords() functions remove all quotes and backslashes that are not themselves backslash-escaped or inside of single quotes (i.e., &quotewords() tries to interpret these characters just like the Bourne shell). NB: these semantics are significantly different from the original version of this module shipped with Perl 5.000 through 5.004. As an additional feature, $keep may be the keyword "delimiters" which causes the functions to preserve the delimiters in each string as tokens in the token lists, in addition to preserving quote and backslash characters.

&shellwords() is written as a special case of &quotewords(), and it does token parsing with whitespace as a delimiter-- similar to most Unix shells.

EXAMPLES

The sample program:

```
  use Text::ParseWords;
  @words = quotewords('\s+', 0, q{this   is "a test" of\ quotewords \"for
you});
  $i = 0;
  foreach (@words) {
      print "$i: <$_>\n";
      $i++;
  }
```

produces:

```
  0: <this>
  1: <is>
  2: <a test>
  3: <of quotewords>
  4: <"for>
  5: <you>
```

demonstrating:

0 a simple word

1 multiple spaces are skipped because of our $delim

2 use of quotes to include a space in a word

3 use of a backslash to include a space in a word

4 use of a backslash to remove the special meaning of a double-quote

5 another simple word (note the lack of effect of the backslashed double-quote)

Replacing `quotewords('\s+', 0, q{this  is...})` with `shellwords(q{this is...})` is a simpler way to accomplish the same thing.

SEE ALSO

Text::CSV - for parsing CSV files

AUTHORS

Maintainer: Alexandr Ciornii <alexchornyATgmail.com>.

Previous maintainer: Hal Pomeranz <pomeranz@netcom.com>, 1994-1997 (Original author unknown). Much of the code for &parse_line() (including the primary regexp) from Joerk Behrends <jbehrends@multimediaproduzenten.de>.

Examples section another documentation provided by John Heidemann <johnh@ISI.EDU>

Bug reports, patches, and nagging provided by lots of folks-- thanks everybody! Special thanks to Michael Schwern <schwern@envirolink.org> for assuring me that a &nested_quotewords() would be useful, and to Jeff Friedl <jfriedl@yahoo-inc.com> for telling me not to worry about error-checking (sort of-- you had to be there).