substr_compare »
« strtoupper

- Manual de PHP
- Referencia de funciones
- Procesamiento de texto
- Strings
- Funciones de strings

Change language: Spanish

Submit a Pull Request Report a Bug

strtr

(PHP 4, PHP 5, PHP 7, PHP 8)

strtr — Convierte caracteres o reemplaza substrings

Descripción ¶

strtr(string `$str`, string `$from`, string `$to`): string
strtr(string `$str`, array `$replace_pairs`): string

Si se dan tres argumentos, esta función devuelve una copia de `str`, donde todas las apariciones de cada carácter (byte simple) en `from` han sido convertidas al carácter correspondiente en `to`, es decir, todas las apariciones de `$from[$n]` han sido reemplazadas con `$to[$n]`, donde `$n` es un índice válido en ambos argumentos.

Si `from` y `to` tienen distinta logitud, se ignoran los caracteres extra del string más largo. La longitud de `str` será la misma que la del valor devuelto.

Si se dan dos argumentos, el segundo debería ser un array en la forma `array('from' => 'to', ...)`. El valor devuelto es un string donde todas las apariciones de las claves del array han sido reemplazadas por los valores correspondientes. Las claves más largas se probarán primero. Una vez un substring ha sido reemplazado, su nuevo valor no se buscará de nuevo.

En este caso, las claves y los valores pueden tener cualquier longitud, siempre que no haya claves vacías; además, la longitud del valor devuelto podría diferir de la de `str`. Sin embargo, esta función será la más eficiente cuando todas las claves tienen el mismo tamaño.

Parámetros ¶

`string`

El string a convertir.

`from`

El string a convertir a `to`.

`to`

El string que reemplaza a `from`.

`replace_pairs`

El parámetro `replace_pairs` se podría usar en lugar de `to` y `from`, en cuyo caso sería un array en la forma `array('from' => 'to', ...)`.

Valores devueltos ¶

Devuelve el string convertido.

Si `replace_pairs` contiene una clave que es un string vacío (`""`), devolverá **`false`**. Si `str` no es un escalar, no será convertido a un string, se emitirá una advertencia y devolverá **`null`**.

Ejemplos ¶

Ejemplo #1 Ejemplo de strtr()

```
<?php
//De esta forma, strtr() hace una conversión byte a byte
//Por lo tanto, aquí se asume una codificación de un solo byte:
$addr = strtr($addr, "äåö", "aao");
?>
```

El siguiente ejemplo muestra el comportamiento de **strtr()** cuando se llama con sólo dos argumentos. Observe la preferencia de los reemplazos (`"h"` no se toma porque hay coincidencias más largas) y cómo el texto remplazado no se busca de nuevo.

Ejemplo #2 Ejemplo de strtr() con dos argumentos

```
<?php
$conv = array("h" => "-", "hola" => "hey", "hey" => "hola");
echo strtr("hey, dije hola", $conv);
?>
```

El resultado del ejemplo sería:

```
hola, dije hey
```

Los dos modos de comportamiento son muy diferentes. Con tres argumentos, **strtr()** reemplazará bytes; con dos, podría reemplazar substrings más largos.

Ejemplo #3 Comparación del comportamiento de strtr()

```
<?php
echo strtr("baab", "ab", "01"),"\n";

$conv = array("ab" => "01");
echo strtr("baab", $conv);
?>
```

El resultado del ejemplo sería:

```
1001
ba01
```

Ver también ¶

- str_replace() - Reemplaza todas las apariciones del string buscado con el string de reemplazo
- preg_replace() - Realiza una búsqueda y sustitución de una expresión regular

+ add a note

User Contributed Notes 32 notes

up
down
98
evan dot king at NOSPAM dot example dot com ¶
7 years ago
Here's an important real-world example use-case for strtr where str_replace will not work or will introduce obscure bugs:

```
<?php

$strTemplate = "My name is :name, not :name2.";
$strParams = [
  ':name' => 'Dave',
  'Dave' => ':name2 or :password', // a wrench in the otherwise sensible input
  ':name2' => 'Steve',
  ':pass' => '7hf2348', // sensitive data that maybe shouldn't be here
];

echo strtr($strTemplate, $strParams);
// "My name is Dave, not Steve."

echo str_replace(array_keys($strParams), array_values($strParams), $strTemplate);
// "My name is Steve or 7hf2348word, not Steve or 7hf2348word2."

?>
```

Any time you're trying to template out a string and don't necessarily know what the replacement keys/values will be (or fully understand the implications of and control their content and order), str_replace will introduce the potential to incorrectly match your keys because it does not expand the longest keys first.

Further, str_replace will replace in previous replacements, introducing potential for unintended nested expansions. Doing so can put the wrong data into the "sub-template" or even give users a chance to provide input that exposes data (if they get to define some of the replacement strings).

Don't support recursive expansion unless you need it and know it will be safe. When you do support it, do so explicitly by repeating strtr calls until no more expansions are occurring or a sane iteration limit is reached, so that the results never implicitly depend on order of your replacement keys. Also make certain that any user input will expanded in an isolated step after any sensitive data is already expanded into the output and no longer available as input.

Note: using some character(s) around your keys to designate them also reduces the possibility of unintended mangling of output, whether maliciously triggered or otherwise. Thus the use of a colon prefix in these examples, which you can easily enforce when accepting replacement input to your templating/translation system.

up
down
59
Hayley Watson ¶
9 years ago
Since strtr (like PHP's other string functions) treats strings as a sequence of bytes, and since UTF-8 and other multibyte encodings use - by definition - more than one byte for at least some characters, the three-string form is likely to have problems. Use the associative array form to specify the mapping.

```
<?php
// Assuming UTF-8
$str = 'Äbc Äbc'; // strtr() sees this as nine bytes (including two for each Ä)
echo strtr($str, 'Ä', 'a'); // The second argument is equivalent to the string "\xc3\x84" so
"\xc3" gets replaced by "a" and the "\x84" is ignored

echo strtr($str, array('Ä' => 'a')); // Works much better
?>
```

up
down
17

allixsenos at gmail dot com ¶

13 years ago

fixed "normaliza" functions written below to include Slavic Latin characters... also, it doesn't return lowercase any more (you can easily get that by applying strtolower yourself)...

also, renamed to normalize()

```
<?php

function normalize ($string) {
    $table = array(
        'Š'=>'S', 'š'=>'s', 'Đ'=>'Dj', 'đ'=>'dj', 'Ž'=>'Z', 'ž'=>'z', 'Č'=>'C', 'č'=>'c',
'Ć'=>'C', 'ć'=>'c',
        'À'=>'A', 'Á'=>'A', 'Â'=>'A', 'Ã'=>'A', 'Ä'=>'A', 'Å'=>'A', 'Æ'=>'A', 'Ç'=>'C', 'È'=>'E',
'É'=>'E',
        'Ê'=>'E', 'Ë'=>'E', 'Ì'=>'I', 'Í'=>'I', 'Î'=>'I', 'Ï'=>'I', 'Ñ'=>'N', 'Ò'=>'O', 'Ó'=>'O',
'Ô'=>'O',
        'Õ'=>'O', 'Ö'=>'O', 'Ø'=>'O', 'Ù'=>'U', 'Ú'=>'U', 'Û'=>'U', 'Ü'=>'U', 'Ý'=>'Y', 'Þ'=>'B',
'ß'=>'Ss',
        'à'=>'a', 'á'=>'a', 'â'=>'a', 'ã'=>'a', 'ä'=>'a', 'å'=>'a', 'æ'=>'a', 'ç'=>'c', 'è'=>'e',
'é'=>'e',
        'ê'=>'e', 'ë'=>'e', 'ì'=>'i', 'í'=>'i', 'î'=>'i', 'ï'=>'i', 'ð'=>'o', 'ñ'=>'n', 'ò'=>'o',
'ó'=>'o',
        'ô'=>'o', 'õ'=>'o', 'ö'=>'o', 'ø'=>'o', 'ù'=>'u', 'ú'=>'u', 'û'=>'u', 'ý'=>'y', 'ý'=>'y',
'þ'=>'b',
        'ÿ'=>'y', 'Ŕ'=>'R', 'ŕ'=>'r',
    );

    return strtr($string, $table);
}

?>
```

up
down
13

dot dot dot dot dot alexander at gmail dot com ¶

14 years ago

OK, I debugged the function (had some errors)
Here it is:

```
if(!function_exists("stritr")){
    function stritr($string, $one = NULL, $two = NULL){
/*
stritr - case insensitive version of strtr
Author: Alexander Peev
Posted in PHP.NET
*/
```

```
        if(  is_string( $one )  ){
            $two = strval( $two );
            $one = substr(  $one, 0, min( strlen($one), strlen($two) )  );
            $two = substr(  $two, 0, min( strlen($one), strlen($two) )  );
            $product = strtr(  $string, ( strtoupper($one) . strtolower($one) ), ( $two . $two )
);
            return $product;
        }
        else if(  is_array( $one )  ){
            $pos1 = 0;
            $product = $string;
            while(  count( $one ) > 0  ){
                $positions = array();
                foreach(  $one as $from => $to  ){
                    if(   (  $pos2 = stripos( $product, $from, $pos1 )  ) === FALSE   ){
                        unset(  $one[ $from ]  );
                    }
                    else{
                        $positions[ $from ] = $pos2;
                    }
                }
                if(  count( $one ) <= 0  )break;
                $winner = min( $positions );
                $key = array_search(  $winner, $positions  );
                $product = (   substr(  $product, 0, $winner  ) . $one[$key] . substr(  $product,
( $winner + strlen($key) )  )   );
                $pos1 = (  $winner + strlen( $one[$key] )  );
            }
            return $product;
        }
        else{
            return $string;
        }
    }/* endfunction stritr */
}/* endfunction exists stritr */
```

up
down
4

joeldegan AT yahoo ¶

16 years ago

After battling with strtr trying to strip out MS word formatting from things pasted into forms I ended up coming up with this..

it strips ALL non-standard ascii characters, preserving html codes and such, but gets rid of all the characters that refuse to show in firefox.

If you look at this page in firefox you will see a ton of "question mark" characters and so it is not possible to copy and paste those to remove them from strings.. (this fixes that issue nicely, though I admit it could be done a bit better)

```
<?
function fixoutput($str){
    $good[] = 9;  #tab
    $good[] = 10; #nl
    $good[] = 13; #cr
    for($a=32;$a<127;$a++){
        $good[] = $a;
    }
```

```
    $len = strlen($str);
    for($b=0;$b < $len+1; $b++){
        if(in_array(ord($str[$b]), $good)){
            $newstr .= $str[$b];
        }//fi
    }//rof
    return $newstr;
}
?>
```

up
down
7

Michael Schuijff ¶

11 years ago

I found that this approach is often faster than strtr() and won't change the same thing in your string twice (as opposed to str_replace(), which will overwrite things in the order of the array you feed it with):

```
<?php
function replace ($text, $replace) {
    $keys = array_keys($replace);
    $length = array_combine($keys, array_map('strlen', $keys));
    arsort($length);

    $array[] = $text;
    $count = 1;
    reset($length);
    while ($key = key($length)) {
        if (strpos($text, $key) !== false) {
            for ($i = 0; $i < $count; $i += 2) {
                if (($pos = strpos($array[$i], $key)) === false) continue;
                array_splice($array, $i, 1, array(substr($array[$i], 0, $pos), $replace[$key],
substr($array[$i], $pos + strlen($key))));
                $count += 2;
            }
        }
        next($length);
    }
    return implode($array);
}
?>
```

up
down
4

horak.jan AT centrum.cz ¶

15 years ago

Here is a function to convert middle-european windows charset (cp1250) to the charset, that php script is written in:

```
<?php
    function cp1250_to_utf2($text){
        $dict  = array(chr(225) => 'á', chr(228) =>  'ä', chr(232) => 'č', chr(239) => 'ď',
            chr(233) => 'é', chr(236) => 'ě', chr(237) => 'í', chr(229) => 'Í', chr(229) => 'Ĺ',
            chr(242) => 'ň', chr(244) => 'ô', chr(243) => 'ó', chr(154) => 'š', chr(248) => 'ř',
            chr(250) => 'ú', chr(249) => 'ů', chr(157) => 'ť', chr(253) => 'ý', chr(158) => 'ž',
            chr(193) => 'Á', chr(196) => 'Ä', chr(200) => 'Č', chr(207) => 'Ď', chr(201) => 'É',
            chr(204) => 'Ě', chr(205) => 'Í', chr(197) => 'Ĺ',    chr(188) => 'Ľ', chr(210) =>
'Ň',
```

```
            chr(212) => 'Ô', chr(211) => 'Ó', chr(138) => 'Š', chr(216) => 'Ř', chr(218) => 'Ú',
            chr(217) => 'Ů', chr(141) => 'Ť', chr(221) => 'Ý', chr(142) => 'Ž',
            chr(150) => '-');
        return strtr($text, $dict);
    }
?>
```

up
down
6

dcz at phpbb-seo dot com ¶

9 years ago

strstr will issue a notice when $replace_pairs contains an array, even unused, with php 5.5.0.

It was not the case with version at least up to 5.3.2, but I'm not sure the notice was added with exactly 5.5.0.

```
<?php
$str = 'hi all, I said hello';
$replace_pairs = array(
      'all' => 'everybody',
    'unused' => array('somtehing', 'something else'),
     'hello' => 'hey',
);
// php 5.5.0 Notice: Array to string conversion in test.php on line 8
echo strtr($str, $replace_pairs); // hi everybody, I said hey
?>
```

since the result is still correct, @strstr seems a working solution.

up
down
5

troelskn at gmail dot com ¶

14 years ago

Here's another transcribe function. This one converts cp1252 (aka. Windows-1252) into iso-8859-1 (aka. latin1, the default PHP charset). It only transcribes the few exotic characters, which are unique to cp1252.

```
function transcribe_cp1252_to_latin1($cp1252) {
  return strtr(
    $cp1252,
    array(
      "\x80" => "e",  "\x81" => " ",    "\x82" => "'", "\x83" => 'f',
      "\x84" => '"',  "\x85" => "...",  "\x86" => "+", "\x87" => "#",
      "\x88" => "^",  "\x89" => "0/00", "\x8A" => "S", "\x8B" => "<",
      "\x8C" => "OE", "\x8D" => " ",    "\x8E" => "Z", "\x8F" => " ",
      "\x90" => " ",  "\x91" => "`",    "\x92" => "'", "\x93" => '"',
      "\x94" => '"',  "\x95" => "*",    "\x96" => "-", "\x97" => "--",
      "\x98" => "~",  "\x99" => "(TM)", "\x9A" => "s", "\x9B" => ">",
      "\x9C" => "oe", "\x9D" => " ",    "\x9E" => "z", "\x9F" => "Y"));
```

up
down
4

elloromtz at gmail dot com ¶

12 years ago

If you supply 3 arguments and the 2nd is an array, strtr will search the "A" from "Array" (because you're treating it as a scalar string) and replace it with the 3rd argument:

```
strtr('Analogy', array('x'=>'y'),  '_'); //'_nalogy'
```

so in reality the above code has the same affect as:

```
strtr('Analogy', 'A' , '_');
```

up
down
4
Annubis ¶
4 years ago

Since I was having a lot of trouble finding a multibyte safe strtr and the solutions I found didn't help, I came out with this function, I don't know how it works with non latin chars but it works for me using spanish/french utf8, I hope it helps someone...

```
<?php
if(!function_exists('mb_strtr')) {
    function mb_strtr ($str, $from, $to = null) {
        if(is_array($from)) {
            $from = array_map('utf8_decode', $from);
            $from = array_map('utf8_decode', array_flip ($from));
            return utf8_encode (strtr (utf8_decode ($str), array_flip ($from)));
        }
        return utf8_encode (strtr (utf8_decode ($str), utf8_decode($from), utf8_decode ($to)));
    }
}
?>
```

up
down
5
dot dot dot dot dot alexander at gmail dot com ¶
14 years ago

Here is the stritr I always needed... I wrote it in 15 minutes... But only after the idea struck me. Hope you find it helpful, and enjoy...

```
<?php
if(!function_exists("stritr")){
    function stritr($string, $one = NULL, $two = NULL){
/*
stritr - case insensitive version of strtr
Author: Alexander Peev
Posted in PHP.NET
*/
        if(  is_string( $one )  ){
            $two = strval( $two );
            $one = substr(  $one, 0, min( strlen($one), strlen($two) )  );
            $two = substr(  $two, 0, min( strlen($one), strlen($two) )  );
            $product = strtr(  $string, ( strtoupper($one) . strtolower($one) ), ( $two . $two )
);
            return $product;
        }
        else if(  is_array( $one )  ){
            $pos1 = 0;
            $product = $string;
            while(  count( $one ) > 0  ){
                $positions = array();
                foreach(  $one as $from => $to  ){
                    if(   (  $pos2 = stripos( $product, $from, $pos1 )  ) === FALSE   ){
                        unset(  $one[ $from ]  );
                    }
                    else{
                        $positions[ $from ] = $pos2;
```

```
                    }
                }
                $winner = min( $positions );
                $key = array_search(  $winner, $positions  );
                $product = (   substr(  $product, 0, $winner  ) . $positions[$key] . substr(
$product, ( $winner + strlen($key) )  )   );
                $pos1 = (  $winner + strlen( $positions[$key] )  );
            }
            return $product;
        }
        else{
            return $string;
        }
    }/* endfunction stritr */
}/* endfunction exists stritr */
?>
```

up
down
6
Tedy ¶
10 years ago

Since strtr() is twice faster than strlwr I decided to write my own lowering function which also handles UTF-8 characters.

```
<?php

function strlwr($string, $utf = 1)
{
    $latin_letters = array('Ă' => 'a',
                           'Â' => 'a',
                           'Î' => 'i',
                           'Ș' => 's',
                           'Ş' => 's',
                           'Ț' => 't',
                           'Ţ' => 't');

    $utf_letters = array('Ă' => 'ă',
                         'Â' => 'â',
                         'Î' => 'î',
                         'Ș' => 'ș',
                         'Ş' => 'ş',
                         'Ț' => 'ț',
                         'Ţ' => 'ț');

    $letters = array('A' => 'a',
                     'B' => 'b',
                     'C' => 'c',
                     'D' => 'd',
                     'E' => 'e',
                     'F' => 'f',
                     'G' => 'g',
                     'H' => 'h',
                     'I' => 'i',
                     'J' => 'j',
                     'K' => 'k',
                     'L' => 'l',
                     'M' => 'm',
                     'N' => 'n',
```

```
                    'O' => 'o',
                    'P' => 'p',
                    'Q' => 'q',
                    'R' => 'r',
                    'S' => 's',
                    'T' => 't',
                    'U' => 'u',
                    'V' => 'v',
                    'W' => 'w',
                    'X' => 'x',
                    'Y' => 'y',
                    'Z' => 'z');

    return ($utf == 1) ? strtr($string, array_merge($utf_letters, $letters)) : strtr($string,
array_merge($latin_letters, $letters));
}

?>
```

This allows you to lower every character (even UTF-8 ones) if you don't set the second parameter, or just lower the UTF-8 ones into their specific latin characters (used when making friendly-urls for example).

I used romanian characters but, of course, you can add your own local characters.

Feel free to use/modify this function as you wish. Hope it helps.

up
down
4

martin[dot]pelikan[at]gmail[dot]com ¶

16 years ago

```
// if you are upset with windows' ^M characters at the end of the line,
// these two lines are for you:
$trans = array("\x0D" => "");
$text = strtr($orig_text,$trans);

// note that ctrl+M (in vim known as ^M) is hexadecimally 0x0D
```

up
down
4

doydoy44 ¶

9 years ago

The example of VOVA (http://www.php.net/manual/fr/function.strtr.php#111968) is good but the result is false:
His example dont replace the string.

```
<?php
function f1_strtr() {
  for($i=0; $i<1000000; ++$i) {
    $new_string = strtr("aboutdealers.com", array(".com" => ""));
  }
  return $new_string;
}
function f2_str_replace() {
  for($i=0; $i<1000000; ++$i) {
    $new_string = str_replace( ".com", "", "aboutdealers.com");
  }
  return $new_string;
```

```
}
$start = microtime(true);
$strtr = f1_strtr();
$stop = microtime(true);
$time_strtr = $stop - $start;

$start = microtime(true);
$str_replace = f2_str_replace();
$stop = microtime(true);
$time_str_replace = $stop - $start;


echo 'time strtr       : ' . $time_strtr       . "\tresult :" . $strtr       . "\n";
echo 'time str_replace : ' . $time_str_replace . "\tresult :" . $str_replace . "\n";
echo 'time strtr > time str_replace => ' . ($time_strtr > $time_str_replace);
?>
---------------------------------------
time strtr       : 3.9719619750977      result :aboutdealers
time str_replace : 2.9930369853973      result :aboutdealers
time strtr > time str_replace => 1

str_replace is faster than strtr
```

up
down
4

qeremy [atta] gmail [dotta] com ¶

9 years ago

```
Weird, but strtr corrupting chars, if used like below and if file is encoded in UTF-8;

<?php
$str = 'Äbc Äbc';
echo strtr($str, 'Ä', 'a');
// output: a�bc a�bc
?>

And a simple solution;

<?php
function strtr_unicode($str, $a = null, $b = null) {
    $translate = $a;
    if (!is_array($a) && !is_array($b)) {
        $a = (array) $a;
        $b = (array) $b;
        $translate = array_combine(
            array_values($a),
            array_values($b)
        );
    }
    // again weird, but accepts an array in this case
    return strtr($str, $translate);
}

$str = 'Äbc Äbc';
echo strtr($str, 'Ä', 'a') ."\n";
echo strtr_unicode($str, 'Ä', 'a') ."\n";
echo strtr_unicode($str, array('Ä' => 'a')) ."\n";
// outputs
// a�bc a�bc
```

```
// abc abc
// abc abc
?>
```

up
down
5

Sidney Ricardo ¶

14 years ago

This work fine to me:

```
<?php
function normaliza ($string){
    $a = 'ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖØÙÚÛÜÝÞ
ßàáâãäåæçèéêëìíîïðñòóôõöøùúûýýþÿŔŕ';
    $b = 'aaaaaaaceeeeiiiidnoooooouuuuy
bsaaaaaaaceeeeiiiidnoooooouuuyybyRr';
    $string = utf8_decode($string);
    $string = strtr($string, utf8_decode($a), $b);
    $string = strtolower($string);
    return utf8_encode($string);
}
?>
```

up
down
1

gabi at unica dot edu ¶

20 years ago

To convert special chars to their html entities strtr you can use strtr in conjunction with get_html_translation_table(HTML_ENTITIES) :

```
$trans = get_html_translation_table(HTML_ENTITIES);
$html_code = strtr($html_code, $trans);

This will replace in $html_code the ? by &Aacute; , etc.
```

up
down
1

Anonymous ¶

17 years ago

If you are going to call strtr a lot, consider using str_replace instead, as it is much faster. I cut execution time in half just by doing this.

```
<?
// i.e. instead of:
$s=strtr($s,$replace_array);

// use:
foreach($replace_array as $key=>$value) $s=str_replace($key,$value,$s);
?>
```

up
down
1

tomhmambo at seznam dot cz ¶

16 years ago

```
<?
// Windows-1250 to ASCII
// This function replace all Windows-1250 accent characters with
// thier non-accent ekvivalents. Useful for Czech and Slovak languages.
```

```
function win2ascii($str)    {

    $str = StrTr($str,
        "\xE1\xE8\xEF\xEC\xE9\xED\xF2",
        "\x61\x63\x64\x65\x65\x69\x6E");

    $str = StrTr($str,
        "\xF3\xF8\x9A\x9D\xF9\xFA\xFD\x9E\xF4\xBC\xBE",
        "\x6F\x72\x73\x74\x75\x75\x79\x7A\x6F\x4C\x6C");

    $str = StrTr($str,
        "\xC1\xC8\xCF\xCC\xC9\xCD\xC2\xD3\xD8",
        "\x41\x43\x44\x45\x45\x49\x4E\x4F\x52");

    $str = StrTr($str,
        "\x8A\x8D\xDA\xDD\x8E\xD2\xD9\xEF\xCF",
        "\x53\x54\x55\x59\x5A\x4E\x55\x64\x44");

    return $str;
}
?>
```

up
down
0

Romain ¶

8 years ago

```
<?php
    /**
     * Clean string,
     * minimize and remove space, accent and other
     *
     * @param string $string
     * @return string
     */
    public function mb_strtoclean($string){
        // Valeur a nettoyer (conversion)
        $unwanted_array = array(    'Š'=>'S', 'š'=>'s', 'Ž'=>'Z', 'ž'=>'z', 'À'=>'A', 'Á'=>'A',
'Â'=>'A', 'Ã'=>'A', 'Ä'=>'A', 'Å'=>'A', 'Æ'=>'A', 'Ç'=>'C', 'È'=>'E', 'É'=>'E',
                                    'Ê'=>'E', 'Ë'=>'E', 'Ì'=>'I', 'Í'=>'I', 'Î'=>'I', 'Ï'=>'I',
'Ñ'=>'N', 'Ò'=>'O', 'Ó'=>'O', 'Ô'=>'O', 'Õ'=>'O', 'Ö'=>'O', 'Ø'=>'O', 'Ù'=>'U',
                                    'Ú'=>'U', 'Û'=>'U', 'Ü'=>'U', 'Ý'=>'Y', 'Þ'=>'B', 'ß'=>'Ss',
'à'=>'a', 'á'=>'a', 'â'=>'a', 'ã'=>'a', 'ä'=>'a', 'å'=>'a', 'æ'=>'a', 'ç'=>'c',
                                    'è'=>'e', 'é'=>'e', 'ê'=>'e', 'ë'=>'e', 'ì'=>'i', 'í'=>'i',
'î'=>'i', 'ï'=>'i', 'ð'=>'o', 'ñ'=>'n', 'ò'=>'o', 'ó'=>'o', 'ô'=>'o', 'õ'=>'o',
                                    'ö'=>'o', 'ø'=>'o', 'ù'=>'u', 'ú'=>'u', 'û'=>'u', 'ý'=>'y',
'ý'=>'y', 'þ'=>'b', 'ÿ'=>'y',
                                    ' ' => '', '_' => '', '-' => '', '.'=> '', ',' => '', ';' =>
'');

        return mb_strtolower(strtr($string, $unwanted_array ));
    }
```

up
down
0

ktogias at math dot upatras dot gr ¶

18 years ago

```
This function is usefull for
accent insensitive regexp
searches into greek (iso8859-7) text:
(Select View -> Character Encoding -> Greek (iso8859-7)
at your browser to see the correct greek characters)

function gr_regexp($mystring){
        $replacement=array(
                array("?","?","?","?"),
                array("?","?","?","?"),
                array("?","?","?","?"),
                array("?","?","?","?","?","?"),
                array("?","?","?","?"),
                array("?","?","?","?","?","?"),
                array("?","?","?","?")
        );
        foreach($replacement as $group){
                foreach($group as $character){
                        $exp="[";
                        foreach($group as $expcharacter){
                                $exp.=$expcharacter;
                        }
                        $exp.="]";
                        $trans[$character]=$exp;
                }
        }
        $temp=explode(" ", $mystring);
        for ($i=0;$i<sizeof($temp);$i++){
                $temp[$i]=strtr($temp[$i],$trans);
                $temp[$i]=addslashes($temp[$i]);
        }
        return implode(".*",$temp);
}

$match=gr_regexp("???????????????????? ??? ????????");

//The next query string can be sent to MySQL
through mysql_query()
$query=
      "Select `column` from `table` where `column2` REGEXP
                        '".$match."'";
```

up
down
0
j at pureftpd dot org ¶
18 years ago

Here's a very useful function to translate Microsoft characters into Latin 15, so that people won't see any more square instead of characters in web pages .

```
function demicrosoftize($str) {
    return strtr($str,
"\x82\x83\x84\x85\x86\x87\x89\x8a" .
"\x8b\x8c\x8e\x91\x92\x93\x94\x95" .
"\x96\x97\x98\x99\x9a\x9b\x9c\x9e\x9f",
"'f\".**^\xa6<\xbc\xb4'" .
"\"\"---~ \xa8>\xbd\xb8\xbe");
}
```

up

down
0
bisqwit at iki dot fi ¶
20 years ago

```
#!/bin/sh
# This shell script generates a strtr() call
# to translate from a character set to another.
# Requires: gnu recode, perl, php commandline binary
#
# Usage:
#  Set set1 and set2 to whatever you prefer
#  (multibyte character sets are not supported)
#  and run the script. The script outputs
#  a strtr() php code for you to use.
#
# Example is set to generate a
# cp437..latin9 conversion code.
#
set1=cp437
set2=iso-8859-15
result="`echo '<? for($c=32;$c<256;$c++)'\
              'echo chr($c);'\
         |php -q|recode -f $set1..$set2`"
echo "// This php function call converts \$string in $set1 to $set2";
cat <<EOF  | php -q
<?php
\$set1='`echo -n "$result"\
   |perl -pe "s/([\\\\']) /\\\\\\\\\\\$1/g"`';
\$set2='`echo -n "$result"|recode -f $set2..$set1\
   |perl -pe "s/([\\\\']) /\\\\\\\\\\\$1/g"`';
\$erase=array();
\$l=strlen(\$set1);
for(\$c=0;\$c<\$l;++\$c)
  if(\$set1[\$c]==\$set2[\$c])\$erase[\$set1[\$c]]='';
if(count(\$erase))
{
  \$set1=strtr(\$set1,\$erase);
  \$set2=strtr(\$set2,\$erase);
}
if(!strlen(\$set1))echo 'IRREVERSIBLE';else
echo "strtr(\\\$string,\n  '",
     ereg_replace('([\\\\\\\'])', '\\\\\\1', \$set2),
     "',\n  '",
     ereg_replace('([\\\\\\\'])', '\\\\\\1', \$set1),
     "');";
EOF
```

up
down
-1
ajitsingh4u at gmail dot com ¶
15 years ago

```
/**
* Replaces special characters with single quote,double quote and comma for charset iso-8859-1
*
* replaceSpecialChars()
* @param string $str
* @return string
*/
```

```
function replaceSpecialChars($str)
{
    //`(96) ’(130) „(132) ‘(145) ’(146) “(147) ”(148) ´(180)   // equivalent ascii values of these
characters.
    $str = strtr($str, "`’„‘’´", "'',''''");
    $str = strtr($str, '“”', '""');
    return $str;
}
```

up
down
-1

patrick at p-roocks dot de ¶
17 years ago

As Daijoubu suggested use str_replace instead of this function for large arrays/subjects. I just tried it with a array of 60 elements, a string with 8KB length, and the execution time of str_replace was faster at factor 20!

Patrick

up
down
-1

ru dot dy at gmx dot net ¶
17 years ago

Posting umlaute here resulted in a mess. Heres a version of the same function that works with preg_replace only:

```
<?php
  function getRewriteString($sString) {
     $string = strtolower(htmlentities($sString));
     $string = preg_replace("/&(.)(uml);/", "$1e", $string);
     $string = preg_replace("/&(.)(acute|cedil|circ|ring|tilde|uml);/", "$1", $string);
     $string = preg_replace("/([^a-z0-9]+)/", "-", html_entity_decode($string));
     $string = trim($string, "-");
     return $string;
  }
?>
```

up
down
-1

symlink23-remove-my-spleen at yahoo dot com ¶
20 years ago

As noted in the str_rot13 docs, some servers don't provide the str_rot13() function. However, the presence of strtr makes it easy to build your own facsimile thereof:

```
if (!function_exists('str_rot13')) {
    function str_rot13($str) {
        $from = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
        $to   = 'nopqrstuvwxyzabcdefghijklmNOPQRSTUVWXYZABCDEFGHIJKLM';

        return strtr($str, $from, $to);
    }
}
```

This is suitable for very light "encryption" such as hiding email addressess from spambots (then unscrambling them in a mail class, for example).

```
$mail_to=str_rot13("$mail_to");
```

up
down

-1

erik at eldata dot se ¶

20 years ago

As an alternative to the not-yet-existing function stritr mentioned in the first note above You can easily do this:

```
strtr("abc","ABCabc","xyzxyz")

or more general:

strtr("abc",
strtoupper($fromchars).strtolower($fromchars),
$tochars.$tochars);

Just a thought.
```

up

down

-3

Fernando "Malk" Piancastelli ¶

19 years ago

Here's a function to replace linebreaks to html <p> tags. This was initially designed to receive a typed text by a form in a "insert new notice" page and put in a database, then a "notice" page could get the text preformatted with paragraph tags instead of linebreaks that won't appear on browser. The function also removes repeated linebreaks the user may have typed in the form.

```
function break_to_tags(&$text) {

	// find and remove repeated linebreaks

	$double_break = array("\r\n\r\n" => "\r\n");
	do {
		$text = strtr($text, $double_break);
		$position = strpos($text, "\r\n\r\n");
	} while ($position !== false);

	// find and replace remanescent linebreaks by <p> tags

	$change = array("\r\n" => "<p>");
	$text = strtr($text, $change);
}

[]'s
Fernando
```

up

down

-3

tekintian#gmail.com ¶

2 years ago

```
/**
*
*  将一个字串中含有全角的数字字符、字母、空格或'%+-()'字符转换为相应半角字符
* @author tekintian#gmail.com
* @param   string       $str         待转换字串
*
* @return  string       $str         处理后字串
*/
function str2byte_convert(String $str):String  {
    $arr = array('０' => '0', '１' => '1', '２' => '2', '３' => '3', '４' => '4',
```

```
        '5' => '5', '6' => '6', '7' => '7', '8' => '8', '9' => '9',
        'A' => 'A', 'B' => 'B', 'C' => 'C', 'D' => 'D', 'E' => 'E',
        'F' => 'F', 'G' => 'G', 'H' => 'H', 'I' => 'I', 'J' => 'J',
        'K' => 'K', 'L' => 'L', 'M' => 'M', 'N' => 'N', 'O' => 'O',
        'P' => 'P', 'Q' => 'Q', 'R' => 'R', 'S' => 'S', 'T' => 'T',
        'U' => 'U', 'V' => 'V', 'W' => 'W', 'X' => 'X', 'Y' => 'Y',
        'Z' => 'Z', 'a' => 'a', 'b' => 'b', 'c' => 'c', 'd' => 'd',
        'e' => 'e', 'f' => 'f', 'g' => 'g', 'h' => 'h', 'i' => 'i',
        'j' => 'j', 'k' => 'k', 'l' => 'l', 'm' => 'm', 'n' => 'n',
        'o' => 'o', 'p' => 'p', 'q' => 'q', 'r' => 'r', 's' => 's',
        't' => 't', 'u' => 'u', 'v' => 'v', 'w' => 'w', 'x' => 'x',
        'y' => 'y', 'z' => 'z',
        '（' => '(', '）' => ')', '〔' => '[', '〕' => ']', '【' => '[',
        '】' => ']', '〖' => '[', '〗' => ']', '“' => '[', '”' => ']',
        '‘' => '[', '’' => ']', '｛' => '{', '｝' => '}', '《' => '<',
        '》' => '>',
        '%' => '%', '＋' => '+', '—' => '-', '－' => '-', '～' => '-',
        '：' => ':', '。' => '.', '、' => ',', '，' => '.', '、' => '.',
        '；' => ',', '？' => '?', '！' => '!', '…' => '-', '‖' => '|',
        '”' => '"', '’' => '`', '‘' => '`', '｜' => '|', '〃' => '"',
        '　' => ' ');

    return strtr($str, $arr);
}
```

up
down
-2

Anonymous ¶
2 years ago

The string, string form cannot remove characters, but the array form can.

+ add a note

- Funciones de strings
  - addcslashes
  - addslashes
  - bin2hex
  - chop
  - chr
  - chunk_split
  - convert_uudecode
  - convert_uuencode
  - count_chars
  - crc32
  - crypt
  - echo
  - explode
  - fprintf
  - get_html_translation_table
  - hebrev
  - hex2bin
  - html_entity_decode
  - htmlentities
  - htmlspecialchars_decode
  - htmlspecialchars
  - implode
  - join
  - lcfirst
  - levenshtein

- localeconv
- ltrim
- md5_file
- md5
- metaphone
- money_format
- nl_langinfo
- nl2br
- number_format
- ord
- parse_str
- print
- printf
- quoted_printable_decode
- quoted_printable_encode
- quotemeta
- rtrim
- setlocale
- sha1_file
- sha1
- similar_text
- soundex
- sprintf
- sscanf
- str_contains
- str_ends_with
- str_getcsv
- str_ireplace
- str_pad
- str_repeat
- str_replace
- str_rot13
- str_shuffle
- str_split
- str_starts_with
- str_word_count
- strcasecmp
- strchr
- strcmp
- strcoll
- strcspn
- strip_tags
- stripcslashes
- stripos
- stripslashes
- stristr
- strlen
- strnatcasecmp
- strnatcmp
- strncasecmp
- strncmp
- strpbrk
- strpos
- strrchr
- strrev
- strripos
- strrpos
- strspn