strncasecmp »
« strnatcasecmp

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

Change language: Spanish

Submit a Pull Request Report a Bug

strnatcmp

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

strnatcmp — Comparación de strings utilizando un algoritmo de "orden natural"

Descripción ¶

strnatcmp(string `$str1`, string `$str2`): int

Esta función implementa un algoritmo de comparación que ordena strings alfanuméricos de la manera en que un humano lo haría, lo cual se describe como "orden natural". Tener en cuenta que esta comparación es sensible a mayúsculas y minúsculas.

Parámetros ¶

`str1`

El primer string.

`str2`

El segundo string.

Valores devueltos ¶

De forma similar a otras funciones de comparación, esta devuelve < 0 si `str1` es menor que `str2`; > 0 si `str1` es mayor que `str2` y 0 si son iguales.

Ejemplos ¶

Un ejemplo de la diferencia entre éste algoritmo y los algoritmos normales de clasificación del computador (usados en strcmp()), se puede ver a continuación

```
<?php
$arr1 = $arr2 = array("img12.png", "img10.png", "img2.png", "img1.png");
echo "Standard string comparison\n";
usort($arr1, "strcmp");
print_r($arr1);
echo "\nNatural order string comparison\n";
usort($arr2, "strnatcmp");
print_r($arr2);
?>
```

El resultado del ejemplo sería:

```
Standard string comparison
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)

Natural order string comparison
Array
(
    [0] => img1.png
    [1] => img2.png
    [2] => img10.png
    [3] => img12.png
)
```

Para más información ver la página de Martin Pool: » Natural Order String Comparison.

Ver también ¶

- preg_match() - Realiza una comparación con una expresión regular
- strcasecmp() - Comparación de string segura a nivel binario e insensible a mayúsculas y minúsculas
- substr() - Devuelve parte de una cadena
- stristr() - strstr insensible a mayúsculas y minúsculas
- strcmp() - Comparación de string segura a nivel binario
- strncmp() - Comparación segura a nivel binario de los primeros n caracteres entre strings
- strncasecmp() - Comparación de los primeros n caracteres de cadenas, segura con material binario e insensible a mayúsculas y minúsculas
- strnatcasecmp() - Comparación de strings, insensible a mayúsculas y minúsculas, utilizando un algoritmo de "orden natural"
- strstr() - Encuentra la primera aparición de un string
- natsort() - Ordena un array usando un algoritmo de "orden natural"
- natcasesort() - Ordenar un array usando un algoritmo de "orden natural" insensible a mayúsculas-minúsculas

+ add a note

User Contributed Notes 4 notes

up
down
3
in dot games dot mq at gmail dot com ¶
5 years ago

Can also be used with combination of a compare for an array nested value, like

```
<?php
$array = array(
    "city" => "xyz",
    "names" => array(
        array(
            "name" => "Ana2",
            "id" => 1
        ) ,
        array(
            "name" => "Ana1",
```

```
            "id" => 2
        )
    )
);
usort($array["names"], function ($a, $b)    {    return strnatcmp($a['name'], $b['name']);} );
```

up
down
3
thomas at uninet dot se ¶
16 years ago

There seems to be a bug in the localization for strnatcmp and strnatcasecmp. I searched the reported bugs and found a few entries which were up to four years old (but the problem still exists when using swedish characters).

These functions might work instead.

```
<?php
function _strnatcasecmp($left, $right) {
  return _strnatcmp(strtolower($left), strtolower($right));
}

function _strnatcmp($left, $right) {
  while((strlen($left) > 0) && (strlen($right) > 0)) {
    if(preg_match('/^([^0-9]*)([0-9].*)$/Us', $left, $lMatch)) {
      $lTest = $lMatch[1];
      $left = $lMatch[2];
    } else {
      $lTest = $left;
      $left = '';
    }
    if(preg_match('/^([^0-9]*)([0-9].*)$/Us', $right, $rMatch)) {
      $rTest = $rMatch[1];
      $right = $rMatch[2];
    } else {
      $rTest = $right;
      $right = '';
    }
    $test = strcmp($lTest, $rTest);
    if($test != 0) {
      return $test;
    }
    if(preg_match('/^([0-9]+)([^0-9].*)?$/Us', $left, $lMatch)) {
      $lTest = intval($lMatch[1]);
      $left = $lMatch[2];
    } else {
      $lTest = 0;
    }
    if(preg_match('/^([0-9]+)([^0-9].*)?$/Us', $right, $rMatch)) {
      $rTest = intval($rMatch[1]);
      $right = $rMatch[2];
    } else {
      $rTest = 0;
    }
    $test = $lTest - $rTest;
    if($test != 0) {
      return $test;
    }
  }
  return strcmp($left, $right);
```

```
}
?>
```

The code is not optimized. It was just made to solve my problem.

up
down
-1

chris at ocproducts dot com ¶
5 years ago

This function has some interesting behaviour on strings consisting of mixed numbers and letters.

One may expect that such a mixed string would be treated as alpha-numeric, but that is not true.

```
var_dump(strnatcmp('23','123')); →
int(-1)
As expected, 23<123   (even though first digit is higher, overall number is smaller)

var_dump(strnatcmp('yz','xyz')); →
int(1)
As expected, yz>xyz   (string comparison, irregardless of string length)

var_dump(strnatcmp('2x','12y')); →
int(-1)
Remarkable, 2x<12y    (does a numeric comparison)

var_dump(strnatcmp('20x','12y'));
int(1)
Remarkable, 20x>12y    (does a numeric comparison)
```

It seems to be splitting what is being compared into runs of numbers and letters, and then comparing each run in isolation, until it has an ordering difference.

up
down
-4

spamspamspam at gmx dot com ¶
4 years ago

Some more remarkable outcomes:

```
var_dump(strnatcmp("0.15m", "0.2m"));
int(1)

var_dump(strnatcmp("0.15m", "0.20m"));
int(-1)
```

It's not about localisation:

```
var_dump(strnatcmp("0,15m", "0,2m"));
int(1)

var_dump(strnatcmp("0,15m", "0,20m"));
int(-1)
```

+ 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
- strstr
- strtok
- strtolower
- strtoupper
- strtr
- substr_compare
- substr_count
- substr_replace
- substr
- trim
- ucfirst
- ucwords
- utf8_decode
- utf8_encode
- vfprintf
- vprintf
- vsprintf
- wordwrap

- Deprecated
  - convert_cyr_string
  - hebrevc