strpos

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

strpos — Encuentra la posición de la primera ocurrencia de un substring en un string

Descripción ¶

strpos(string `$haystack`, mixed `$needle`, int `$offset` = 0): mixed

Encuentra la posición numérica de la primera ocurrencia del `needle` (aguja) en el string `haystack` (pajar).

Parámetros ¶

`haystack`

El string en donde buscar.

`needle`

Si la `needle` no es una cadena, es convertida a integer y se interpreta como el valor ordinal de un carácter.

`offset`

Si se específica, la búsqueda iniciará en éste número de caracteres contados desde el inicio del string. A diferencia de strrpos() y strripos(), el offset no puede ser negativo.

Valores devueltos ¶

Devuelve la posición donde la aguja existe, en relación al inicio del string `haystack` (independiente del offset). También tener en cuenta que las posiciones de inicio de los string empiezan en 0 y no 1.

Devuelve **`false`** si no fue encontrada la aguja.

Advertencia

Esta función puede devolver el valor booleano **`false`**, pero también puede devolver un valor no booleano que se evalúa como **`false`**. Por favor lea la sección sobre Booleanos para más información. Use el operador === para comprobar el valor devuelto por esta función.

Ejemplos ¶

Ejemplo #1 Usando `===`

```
<?php
$mystring = 'abc';
$findme   = 'a';
$pos = strpos($mystring, $findme);

// Nótese el uso de ===. Puesto que == simple no funcionará como se espera
// porque la posición de 'a' está en el 1° (primer) caracter.
if ($pos === false) {
    echo "La cadena '$findme' no fue encontrada en la cadena '$mystring'";
} else {
    echo "La cadena '$findme' fue encontrada en la cadena '$mystring'";
    echo " y existe en la posición $pos";
}
?>
```

Ejemplo #2 Usando !==

```
<?php
$mystring = 'abc';
$findme   = 'a';
$pos = strpos($mystring, $findme);

// El operador !== también puede ser usado. Puesto que != no funcionará como se espera
// porque la posición de 'a' es 0. La declaración (0 != false) se evalúa a
// false.
if ($pos !== false) {
     echo "La cadena '$findme' fue encontrada en la cadena '$mystring'";
         echo " y existe en la posición $pos";
} else {
     echo "La cadena '$findme' no fue encontrada en la cadena '$mystring'";
}
?>
```

Ejemplo #3 Uso de offset

```
<?php
// Se puede buscar por el caracter, ignorando cualquier cosa antes del offset
$newstring = 'abcdef abcdef';
$pos = strpos($newstring, 'a', 1); // $pos = 7, no 0
?>
```

Notas ¶

> **Nota**: Esta función es segura binariamente.

Ver también ¶

- stripos() - Encuentra la posición de la primera aparición de un substring en un string sin considerar mayúsculas ni minúsculas
- strrpos() - Encuentra la posición de la última aparición de un substring en un string
- strripos() - Encuentra la posición de la última aparición de un substring insensible a mayúsculas y minúsculas en un string
- strstr() - Encuentra la primera aparición de un string
- strpbrk() - Buscar una cadena por cualquiera de los elementos de un conjunto de caracteres
- substr() - Devuelve parte de una cadena
- preg_match() - Realiza una comparación con una expresión regular

+ add a note

User Contributed Notes 38 notes

up
down
209
Suggested re-write for pink WARNING box ¶
14 years ago

```
WARNING

As strpos may return either FALSE (substring absent) or 0 (substring at start of string), strict
versus loose equivalency operators must be used very carefully.

To know that a substring is absent, you must use:

=== FALSE

To know that a substring is present (in any position including 0), you can use either of:

!== FALSE  (recommended)
> -1  (note: or greater than any negative number)

To know that a substring is at the start of the string, you must use:

=== 0

To know that a substring is in any position other than the start, you can use any of:

> 0  (recommended)
!= 0  (note: but not !== 0 which also equates to FALSE)
!= FALSE  (disrecommended as highly confusing)

Also note that you cannot compare a value of "" to the returned value of strpos. With a loose
equivalence operator (== or !=) it will return results which don't distinguish between the
substring's presence versus position. With a strict equivalence operator (=== or !==) it will
always return false.
```

up
down
26
martijn at martijnfrazer dot nl ¶
10 years ago

```
This is a function I wrote to find all occurrences of a string, using strpos recursively.

<?php
function strpos_recursive($haystack, $needle, $offset = 0, &$results = array()) {
    $offset = strpos($haystack, $needle, $offset);
    if($offset === false) {
        return $results;
    } else {
        $results[] = $offset;
        return strpos_recursive($haystack, $needle, ($offset + 1), $results);
    }
}
?>

This is how you use it:
```

```
<?php
$string = 'This is some string';
$search = 'a';
$found = strpos_recursive($string, $search);

if($found) {
    foreach($found as $pos) {
        echo 'Found "'.$search.'" in string "'.$string.'" at position <b>'.$pos.'</b><br />';
    }
} else {
    echo '"'.$search.'" not found in "'.$string.'"';
}
?>
```

up
down
29
fabio at naoimporta dot com ¶
6 years ago

It is interesting to be aware of the behavior when the treatment of strings with characters using different encodings.

```
<?php
# Works like expected. There is no accent
var_dump(strpos("Fabio", 'b'));
#int(2)

# The "á" letter is occupying two positions
var_dump(strpos("Fábio", 'b')) ;
#int(3)

# Now, encoding the string "Fábio" to utf8, we get some "unexpected" outputs. Every letter that is
no in regular ASCII table, will use 4 positions(bytes). The starting point remains like before.
# We cant find the characted, because the haystack string is now encoded.
var_dump(strpos(utf8_encode("Fábio"), 'á'));
#bool(false)

# To get the expected result, we need to encode the needle too
var_dump(strpos(utf8_encode("Fábio"), utf8_encode('á')));
#int(1)

# And, like said before, "á" occupies 4 positions(bytes)
var_dump(strpos(utf8_encode("Fábio"), 'b'));
#int(5)
```

up
down
17
mtroy dot student at gmail dot com ¶
10 years ago

when you want to know how much of substring occurrences, you'll use "substr_count".
But, retrieve their positions, will be harder.
So, you can do it by starting with the last occurrence :

```
function strpos_r($haystack, $needle)
{
    if(strlen($needle) > strlen($haystack))
        trigger_error(sprintf("%s: length of argument 2 must be <= argument 1", __FUNCTION__),
E_USER_WARNING);
```

```
    $seeks = array();
    while($seek = strrpos($haystack, $needle))
    {
        array_push($seeks, $seek);
        $haystack = substr($haystack, 0, $seek);
    }
    return $seeks;
}

it will return an array of all occurrences a the substring in the string

Example :

$test = "this is a test for testing a test function... blah blah";
var_dump(strpos_r($test, "test"));

// output

array(3) {
  [0]=>
  int(29)
  [1]=>
  int(19)
  [2]=>
  int(10)
}

Paul-antoine
Malézieux.
```

up
down
18
rjeggens at ijskoud dot org ¶
10 years ago

```
I lost an hour before I noticed that strpos only returns FALSE as a boolean, never TRUE.. This
means that

strpos() !== false

is a different beast then:

strpos() === true

since the latter will never be true. After I found out, The warning in the documentation made a
lot more sense.
```

up
down
5
greg at spotx dot net ¶
4 years ago

```
Warning:
this is not unicode safe

strpos($word,'?') in e?ez-> 1
strpos($word,'?') in è?ent-> 2
```

up
down

8

jexy dot ru at gmail dot com ¶

5 years ago

```
Docs are missing that WARNING is issued if needle is '' (empty string).

In case of empty haystack it just return false:

For example:

<?php
var_dump(strpos('foo', ''));

var_dump(strpos('', 'foo'));

var_dump(strpos('', ''));
?>

will output:

Warning: strpos(): Empty needle in /in/lADCh on line 3
bool(false)

bool(false)

Warning: strpos(): Empty needle in /in/lADCh on line 7
bool(false)

Note also that warning text may differ depending on php version, see https://3v4l.org/lADCh
```

up

down

3

m.m.j.kronenburg ¶

6 years ago

```
<?php

/**
* Find the position of the first occurrence of one or more substrings in a
* string.
*
* This function is simulair to function strpos() except that it allows to
* search for multiple needles at once.
*
* @param string $haystack    The string to search in.
* @param mixed $needles      Array containing needles or string containing
*                            needle.
* @param integer $offset     If specified, search will start this number of
*                            characters counted from the beginning of the
*                            string.
* @param boolean $last       If TRUE then the farthest position from the start
*                            of one of the needles is returned.
*                            If FALSE then the smallest position from start of
*                            one of the needles is returned.
**/
function mstrpos($haystack, $needles, $offset = 0, $last = false)
{
  if(!is_array($needles)) { $needles = array($needles); }
  $found = false;
  foreach($needles as $needle)
```

```
    {
      $position = strpos($haystack, (string)$needle, $offset);
      if($position === false) { continue; }
      $exp = $last ? ($found === false || $position > $found) :
        ($found === false || $position < $found);
      if($exp) { $found = $position; }
    }
    return $found;
  }

  /**
  * Find the position of the first (partially) occurrence of a substring in a
  * string.
  *
  * This function is simulair to function strpos() except that it wil return a
  * position when the substring is partially located at the end of the string.
  *
  * @param string $haystack    The string to search in.
  * @param mixed $needle       The needle to search for.
  * @param integer $offset     If specified, search will start this number of
  *                            characters counted from the beginning of the
  *                            string.
  **/
  function pstrpos($haystack, $needle, $offset = 0)
  {
    $position = strpos($haystack, $needle, $offset);
    if($position !== false) { return $position; }

    for($i = strlen($needle); $i > 0; $i--)
    {
      if(substr($needle, 0, $i) == substr($haystack, -$i))
      { return strlen($haystack) - $i; }
    }
    return false;
  }

  /**
  * Find the position of the first (partially) occurrence of one or more
  * substrings in a string.
  *
  * This function is simulair to function strpos() except that it allows to
  * search for multiple needles at once and it wil return a position when one of
  * the substrings is partially located at the end of the string.
  *
  * @param string $haystack    The string to search in.
  * @param mixed $needles      Array containing needles or string containing
  *                            needle.
  * @param integer $offset     If specified, search will start this number of
  *                            characters counted from the beginning of the
  *                            string.
  * @param boolean $last       If TRUE then the farthest position from the start
  *                            of one of the needles is returned.
  *                            If FALSE then the smallest position from start of
  *                            one of the needles is returned.
  **/
  function mpstrpos($haystack, $needles, $offset = 0, $last = false)
  {
    if(!is_array($needles)) { $needles = array($needles); }
  $found = false;
  foreach($needles as $needle)
  {
    $position = pstrpos($haystack, (string)$needle, $offset);
    if($position === false) { continue; }
    $exp = $last ? ($found === false || $position > $found) :
      ($found === false || $position < $found);
    if($exp) { $found = $position; }
  }
  return $found;
}

?>
```

up
down
3
usulaco at gmail dot com ¶
12 years ago

```
Parse strings between two others in to array.

<?php
function g($string,$start,$end){
     preg_match_all('/' . preg_quote($start, '/') . '(.*?)'. preg_quote($end, '/').'/i', $string,
$m);
     $out = array();

     foreach($m[1] as $key => $value){
       $type = explode('::',$value);
       if(sizeof($type)>1){
          if(!is_array($out[$type[0]]))
             $out[$type[0]] = array();
          $out[$type[0]][] = $type[1];
       } else {
          $out[] = $value;
       }
     }
  return $out;
}
print_r(g('Sample text, [/text to extract/] Rest of sample text [/WEB::http://google.com/] bla bla
bla. ','[/','/]'));
?>

results:
Array
(
    [0] => text to extract
    [WEB] => Array
        (
            [0] => http://google.com
        )

)

Can be helpfull to custom parsing :)
```

up
down
11
akarmenia at gmail dot com ¶

11 years ago

```
My version of strpos with needles as an array. Also allows for a string, or an array inside an
array.

<?php
function strpos_array($haystack, $needles) {
    if ( is_array($needles) ) {
        foreach ($needles as $str) {
            if ( is_array($str) ) {
                $pos = strpos_array($haystack, $str);
            } else {
                $pos = strpos($haystack, $str);
            }
            if ($pos !== FALSE) {
                return $pos;
            }
        }
    } else {
        return strpos($haystack, $needles);
    }
}

// Test
echo strpos_array('This is a test', array('test', 'drive')); // Output is 10

?>
```

up
down
1
eef dot vreeland at gmail dot com ¶
5 years ago

```
To prevent others from staring at the text, note that the wording of the 'Return Values' section
is ambiguous.

Let's say you have a string $myString containing 50 'a's except on position 3 and 43, they contain
'b'.
And for this moment, forget that counting starts from 0.

strpos($myString, 'b', 40) returns 43, great.

And now the text: "Returns the position of where the needle exists relative to the beginning of
the haystack string (independent of offset)."

So it doesn't really matter what offset I specify; I'll get the REAL position of the first
occurrence in return, which is 3?

... no ...

"independent of offset" means, you will get the REAL positions, thus, not relative to your
starting point (offset).

Substract your offset from strpos()'s answer, then you have the position relative to YOUR offset.
```

up
down
5
ohcc at 163 dot com ¶
8 years ago

```
Be careful when the $haystack or $needle parameter is an integer.
If you are not sure of its type, you should  convert it into a string.
<?php
    var_dump(strpos(12345,1));//false
    var_dump(strpos(12345,'1'));//0
    var_dump(strpos('12345',1));//false
    var_dump(strpos('12345','1'));//0
    $a = 12345;
    $b = 1;
    var_dump(strpos(strval($a),strval($b)));//0
    var_dump(strpos((string)$a,(string)$b));//0
?>
```

up
down
4
***bishop* ¶**
18 years ago

```
Code like this:
<?php
if (strpos('this is a test', 'is') !== false) {
    echo "found it";
}
?>

gets repetitive, is not very self-explanatory, and most people handle it incorrectly anyway. Make
your life easier:

<?php
function str_contains($haystack, $needle, $ignoreCase = false) {
    if ($ignoreCase) {
        $haystack = strtolower($haystack);
        $needle   = strtolower($needle);
    }
    $needlePos = strpos($haystack, $needle);
    return ($needlePos === false ? false : ($needlePos+1));
}
?>

Then, you may do:
<?php
// simplest use
if (str_contains('this is a test', 'is')) {
    echo "Found it";
}

// when you need the position, as well whether it's present
$needlePos = str_contains('this is a test', 'is');
if ($needlePos) {
    echo 'Found it at position ' . ($needlePos-1);
}

// you may also ignore case
$needlePos = str_contains('this is a test', 'IS', true);
if ($needlePos) {
    echo 'Found it at position ' . ($needlePos-1);
}
?>
```

up

down
1
yasindagli at gmail dot com ¶
13 years ago

```
This function finds postion of nth occurence of a letter starting from offset.

<?php
function nth_position($str, $letter, $n, $offset = 0){
    $str_arr = str_split($str);
    $letter_size = array_count_values(str_split(substr($str, $offset)));
    if( !isset($letter_size[$letter])){
        trigger_error('letter "' . $letter . '" does not exist in ' . $str . ' after ' . $offset .
'. position', E_USER_WARNING);
        return false;
    } else if($letter_size[$letter] < $n) {
        trigger_error('letter "' . $letter . '" does not exist ' . $n .' times in ' . $str . '
after ' . $offset . '. position', E_USER_WARNING);
        return false;
    }
    for($i = $offset, $x = 0, $count = (count($str_arr) - $offset); $i < $count, $x != $n; $i++){
        if($str_arr[$i] == $letter){
            $x++;
        }
    }
    return $i - 1;
}

echo nth_position('foobarbaz', 'a', 2); //7
echo nth_position('foobarbaz', 'b', 1, 4); //6
?>
```

up
down
2
Anonymous ¶
9 years ago

```
The most straightforward way to prevent this function from returning 0 is:

  strpos('x'.$haystack, $needle, 1)

The 'x' is simply a garbage character which is only there to move everything 1 position.
The number 1 is there to make sure that this 'x' is ignored in the search.
This way, if $haystack starts with $needle, then the function returns 1 (rather than 0).
```

up
down
2
ilaymyhat-rem0ve at yahoo dot com ¶
14 years ago

```
This might be useful.

<?php
class String{

    //Look for a $needle in $haystack in any position
    public static function contains(&$haystack, &$needle, &$offset)
    {
        $result = strpos($haystack, $needle, $offset);
        return $result !== FALSE;
    }
    //intuitive implementation .. if not found returns -1.
    public static function strpos(&$haystack, &$needle, &$offset)
    {
        $result = strpos($haystack, $needle, $offset);
        if ($result === FALSE )
        {
            return -1;
        }
        return $result;
    }

}//String
?>
```

up
down
0

Jean ¶

3 years ago

```
When a value can be of "unknow" type, I find this conversion trick usefull and more readable than
a formal casting (for php7.3+):

<?php
$time = time();
$string = 'This is a test: ' . $time;
echo (strpos($string, $time) !== false ? 'found' : 'not found');
echo (strpos($string, "$time") !== false ? 'found' : 'not found');
?>
```

up
down
0

amolocaleb at gmail dot com ¶

4 years ago

```
Note that strpos() is case sensitive,so when doing a case insensitive search,use stripos()
instead..If the latter is not available,subject the string to strlower() first,otherwise you may
end up in this situation..
<?php
//say we are matching url routes and calling access control middleware depending on the route

$registered_route = '/admin' ;
//now suppose we want to call the authorization middleware before accessing the admin route
if(strpos($path->url(),$registered_route) === 0){
     $middleware->call('Auth','login');
}
?>
and the auth middleware is as follows
<?php
class Auth{

function login(){
   if(!loggedIn()){
       return redirect("path/to/login.php");
}
return true;
}
}

//Now suppose:
```

```
$user_url = '/admin';
//this will go to the Auth middleware for checks and redirect accordingly

//But:
$user_url = '/Admin';
//this will make the strpos function return false since the 'A' in admin is upper case and user
will be taken directly to admin dashboard authentication and authorization notwithstanding
?>
Simple fixes:
<?php
//use stripos() as from php 5
if(stripos($path->url(),$registered_route) === 0){
     $middleware->call('Auth','login');
}
//for those with php 4
if(stripos(strtolower($path->url()),$registered_route) === 0){
     $middleware->call('Auth','login');
}
//make sure the $registered_route is also lowercase.Or JUST UPGRADE to PHP 5>
```

up
down
0
marvin_elia at web dot de ¶
4 years ago

```
Find position of nth occurrence of a string:

    function strpos_occurrence(string $string, string $needle, int $occurrence, int $offset =
null) {
        if((0 < $occurrence) && ($length = strlen($needle))) {
            do {
            } while ((false !== $offset = strpos($string, $needle, $offset)) && --$occurrence &&
($offset += $length));
            return $offset;
        }
        return false;
    }
```

up
down
1
digitalpbk [at] gmail.com ¶
13 years ago

```
This function raises a warning if the offset is not between 0 and the length of string:

Warning: strpos(): Offset not contained in string in %s on line %d
```

up
down
1
Achintya ¶
13 years ago

```
A function I made to find the first occurrence of a particular needle not enclosed in
quotes(single or double). Works for simple nesting (no backslashed nesting allowed).

<?php
function strposq($haystack, $needle, $offset = 0){
    $len = strlen($haystack);
    $charlen = strlen($needle);
    $flag1 = false;
    $flag2 = false;
    for($i = $offset; $i < $len; $i++){
        if(substr($haystack, $i, 1) == "'"){
            $flag1 = !$flag1 && !$flag2 ? true : false;
        }
        if(substr($haystack, $i, 1) == '"'){
            $flag2 = !$flag1 && !$flag2 ? true : false;
        }
        if(substr($haystack, $i, $charlen) == $needle && !$flag1 && !$flag2){
            return $i;
        }
    }
    return false;
}

echo strposq("he'llo'character;\"'som\"e;crap", ";"); //16
?>
```

up
down
1
spinicrus at gmail dot com ¶
16 years ago

```
if you want to get the position of a substring relative to a substring of your string, BUT in
REVERSE way:

<?php

function strpos_reverse_way($string,$charToFind,$relativeChar) {
    //
    $relativePos = strpos($string,$relativeChar);
    $searchPos = $relativePos;
    $searchChar = '';
    //
    while ($searchChar != $charToFind) {
        $newPos = $searchPos-1;
        $searchChar = substr($string,$newPos,strlen($charToFind));
        $searchPos = $newPos;
    }
    //
    if (!empty($searchChar)) {
        //
        return $searchPos;
        return TRUE;
    }
    else {
        return FALSE;
    }
    //
}

?>
```

up
down
0
lairdshaw at yahoo dot com dot au ¶
7 years ago

```
<?php
/*
* A strpos variant that accepts an array of $needles - or just a string,
 * so that it can be used as a drop-in replacement for the standard strpos,
 * and in which case it simply wraps around strpos and stripos so as not
 * to reduce performance.
 *
 * The "m" in "strposm" indicates that it accepts *m*ultiple needles.
 *
 * Finds the earliest match of *all* needles. Returns the position of this match
 * or false if none found, as does the standard strpos. Optionally also returns
 * via $match either the matching needle as a string (by default) or the index
 * into $needles of the matching needle (if the STRPOSM_MATCH_AS_INDEX flag is
 * set).
 *
 * Case-insensitive searching can be specified via the STRPOSM_CI flag.
 * Note that for case-insensitive searches, if the STRPOSM_MATCH_AS_INDEX is
 * not set, then $match will be in the haystack's case, not the needle's case,
 * unless the STRPOSM_NC flag is also set.
 *
 * Flags can be combined using the bitwise or operator,
 * e.g. $flags = STRPOSM_CI|STRPOSM_NC
 */
define('STRPOSM_CI'            , 1); // CI => "case insensitive".
define('STRPOSM_NC'            , 2); // NC => "needle case".
define('STRPOSM_MATCH_AS_INDEX', 4);
function strposm($haystack, $needles, $offset = 0, &$match = null, $flags = 0) {
    // In the special case where $needles is not an array, simply wrap
    // strpos and stripos for performance reasons.
    if (!is_array($needles)) {
        $func = $flags & STRPOSM_CI ? 'stripos' : 'strpos';
        $pos = $func($haystack, $needles, $offset);
        if ($pos !== false) {
            $match = (($flags & STRPOSM_MATCH_AS_INDEX)
                      ? 0
                      : (($flags & STRPOSM_NC)
                         ? $needles
                         : substr($haystack, $pos, strlen($needles))
                        )
                      );
            return $pos;
        } else    goto strposm_no_match;
    }

    // $needles is an array. Proceed appropriately, initially by...
    // ...escaping regular expression meta characters in the needles.
    $needles_esc = array_map('preg_quote', $needles);
    // If either of the "needle case" or "match as index" flags are set,
    // then create a sub-match for each escaped needle by enclosing it in
    // parentheses. We use these later to find the index of the matching
    // needle.
    if (($flags & STRPOSM_NC) || ($flags & STRPOSM_MATCH_AS_INDEX)) {
        $needles_esc = array_map(
            function($needle) {return '('.$needle.')';},
            $needles_esc
        );
    }
    // Create the regular expression pattern to search for all needles.
    $pattern = '('.implode('|', $needles_esc).')';
    // If the "case insensitive" flag is set, then modify the regular
    // expression with "i", meaning that the match is "caseless".
    if ($flags & STRPOSM_CI) $pattern .= 'i';
    // Find the first match, including its offset.
    if (preg_match($pattern, $haystack, $matches, PREG_OFFSET_CAPTURE, $offset)) {
        // Pull the first entry, the overall match, out of the matches array.
        $found = array_shift($matches);
        // If we need the index of the matching needle, then...
        if (($flags & STRPOSM_NC) || ($flags & STRPOSM_MATCH_AS_INDEX)) {
            // ...find the index of the sub-match that is identical
            // to the overall match that we just pulled out.
            // Because sub-matches are in the same order as needles,
            // this is also the index into $needles of the matching
            // needle.
            $index = array_search($found, $matches);
        }
        // If the "match as index" flag is set, then return in $match
        // the matching needle's index, otherwise...
        $match = (($flags & STRPOSM_MATCH_AS_INDEX)
          ? $index
          // ...if the "needle case" flag is set, then index into
          // $needles using the previously-determined index to return
          // in $match the matching needle in needle case, otherwise...
          : (($flags & STRPOSM_NC)
             ? $needles[$index]
             // ...by default, return in $match the matching needle in
             // haystack case.
             : $found[0]
          )
        );
        // Return the captured offset.
        return $found[1];
    }

strposm_no_match:
    // Nothing matched. Set appropriate return values.
    $match = ($flags & STRPOSM_MATCH_AS_INDEX) ? false : null;
    return false;
}
?>
```

up
down
0
qrworld.net ¶
8 years ago

I found a function in this post http://softontherocks.blogspot.com/2014/11/buscar-multiples-textos-en-un-texto-con.html
that implements the search in both ways, case sensitive or case insensitive, depending on an input parameter.

The function is:

```
function getMultiPos($haystack, $needles, $sensitive=true, $offset=0){
    foreach($needles as $needle) {
        $result[$needle] = ($sensitive) ? strpos($haystack, $needle, $offset) : stripos($haystack,
$needle, $offset);
    }
    return $result;
}
```

It was very useful for me.

up
down
0

Lurvik ¶

8 years ago

Don't know if already posted this, but if I did this is an improvement.

This function will check if a string contains a needle. It _will_ work with arrays and multidimensional arrays (I've tried with a > 16 dimensional array and had no problem).

```
<?php
function str_contains($haystack, $needles)
{
    //If needles is an array
    if(is_array($needles))
    {
        //go trough all the elements
        foreach($needles as $needle)
        {
            //if the needle is also an array (ie needles is a multidimensional array)
            if(is_array($needle))
            {
                //call this function again
                if(str_contains($haystack, $needle))
                {
                    //Will break out of loop and function.
                    return true;
                }

                return false;
            }

            //when the needle is NOT an array:
                //Check if haystack contains the needle, will ignore case and check for whole
words only
            elseif(preg_match("/\b$needle\b/i", $haystack) !== 0)
            {
                return true;
            }
        }
    }
    //if $needles is not an array...
    else
    {
        if(preg_match("/\b$needles\b/i", $haystack) !== 0)
        {
            return true;
        }
    }

    return false;
}
?>
```

up
down
0

gambajaja at yahoo dot com ¶

12 years ago

```
<?php
$my_array = array ('100,101', '200,201', '300,301');
$check_me_in = array ('100','200','300','400');
foreach ($check_me_in as $value_cmi){
    $is_in=FALSE; #asume that $check_me_in isn't in $my_array
    foreach ($my_array as $value_my){
        $pos = strpos($value_my, $value_cmi);
        if ($pos===0)
            $pos++;
        if ($pos==TRUE){
            $is_in=TRUE;
            $value_my2=$value_my;
            }
    }
    if ($is_in) echo "ID $value_cmi in \$check_me_in I found in value '$value_my2' \n";
}
?>

The above example will output
ID 100 in $check_me_in I found in value '100,101'
ID 200 in $check_me_in I found in value '200,201'
ID 300 in $check_me_in I found in value '300,301'
```

up
down
0

teddanzig at yahoo dot com ¶

13 years ago

```
routine to return -1 if there is no match for strpos

<?php
//instr function to mimic vb instr fucntion
function InStr($haystack, $needle)
{
    $pos=strpos($haystack, $needle);
    if ($pos !== false)
    {
        return $pos;
    }
    else
    {
        return -1;
    }
}
?>
```

up
down
0

Tim ¶

14 years ago

```
If you would like to find all occurences of a needle inside a haystack you could use this function
strposall($haystack,$needle);. It will return an array with all the strpos's.

<?php
/**
* strposall
*
```

```
* Find all occurrences of a needle in a haystack
*
* @param string $haystack
* @param string $needle
* @return array or false
*/
function strposall($haystack,$needle){

    $s=0;
    $i=0;

    while (is_integer($i)){

        $i = strpos($haystack,$needle,$s);

        if (is_integer($i)) {
            $aStrPos[] = $i;
            $s = $i+strlen($needle);
        }
    }
    if (isset($aStrPos)) {
        $return $aStrPos;
    }
    else {
        return false;
    }
}
?>
```

up
down
0
user at nomail dot com ¶
15 years ago

```
This is a bit more useful when scanning a large string for all occurances between 'tags'.

<?php
function getStrsBetween($s,$s1,$s2=false,$offset=0) {
    /*====================================================================
    Function to scan a string for items encapsulated within a pair of tags

    getStrsBetween(string, tag1, <tag2>, <offset>

    If no second tag is specified, then match between identical tags

    Returns an array indexed with the encapsulated text, which is in turn
    a sub-array, containing the position of each item.

    Notes:
    strpos($needle,$haystack,$offset)
    substr($string,$start,$length)

    ====================================================================*/

    if( $s2 === false ) { $s2 = $s1; }
    $result = array();
    $L1 = strlen($s1);
    $L2 = strlen($s2);
    if( $L1==0 || $L2==0 ) {
        return false;
    }

    do {
        $pos1 = strpos($s,$s1,$offset);

        if( $pos1 !== false ) {
            $pos1 += $L1;

            $pos2 = strpos($s,$s2,$pos1);

            if( $pos2 !== false ) {
                $key_len = $pos2 - $pos1;

                $this_key = substr($s,$pos1,$key_len);

                if( !array_key_exists($this_key,$result) ) {
                    $result[$this_key] = array();
                }

                $result[$this_key][] = $pos1;

                $offset = $pos2 + $L2;
            } else {
                $pos1 = false;
            }
        }
    } while($pos1 !== false );

    return $result;
}
?>
```

up
down
-1
ah dot d at hotmail dot com ¶
13 years ago

```
A strpos modification to return an array of all the positions of a needle in the haystack

<?php
function strallpos($haystack,$needle,$offset = 0){
    $result = array();
    for($i = $offset; $i<strlen($haystack); $i++){
        $pos = strpos($haystack,$needle,$i);
        if($pos !== FALSE){
            $offset =  $pos;
            if($offset >= $i){
                $i = $offset;
                $result[] = $offset;
            }
        }
    }
    return $result;
}
?>

example:-
```

```
<?php
$haystack = "ASD is trying to get out of the ASDs cube but the other ASDs told him that his
behavior will destroy the ASDs world";

$needle = "ASD";

print_r(strallpos($haystack,$needle));

//getting all the positions starting from a specified position

print_r(strallpos($haystack,$needle,34));
?>
```

up
down
-1
Lhenry ¶
5 years ago

```
note that strpos( "8 june 1970"  ,  1970 ) returns FALSE..

add quotes to the needle
```

up
down
-1
ds at kala-it dot de ¶
2 years ago

```
Note this code example below in PHP 7.3
<?php
$str = "17,25";

if(FALSE !== strpos($str, 25)){
    echo "25 is inside of str";
} else {
    echo "25 is NOT inside of str";
}
?>

Will output "25 is NOT inside of str" and will throw out a deprication message, that non string
needles will be interpreted as strings in the future.

This just gave me some headache since the value I am checking against comes from the database as
an integer.
```

up
down
-1
philip ¶
18 years ago

```
Many people look for in_string which does not exist in PHP, so, here's the most efficient form of
in_string() (that works in both PHP 4/5) that I can think of:
<?php
function in_string($needle, $haystack, $insensitive = false) {
    if ($insensitive) {
        return false !== stristr($haystack, $needle);
    } else {
        return false !== strpos($haystack, $needle);
    }
}
?>
```

up
down
-2
gjh42 - simonokewode at hotmail dot com ¶
11 years ago

```
A pair of functions to replace every nth occurrence of a string with another string, starting at
any position in the haystack. The first works on a string and the second works on a single-level
array of strings, treating it as a single string for replacement purposes (any needles split over
two array elements are ignored).

Can be used for formatting dynamically-generated HTML output without touching the original
generator: e.g. add a newLine class tag to every third item in a floated list, starting with the
fourth item.

<?php
/* String Replace at Intervals   by Glenn Herbert (gjh42)    2010-12-17
*/

//(basic locator by someone else - name unknown)
//strnposr() - Find the position of nth needle in haystack.
function strnposr($haystack, $needle, $occurrence, $pos = 0) {
    return ($occurrence<2)?strpos($haystack, $needle,
$pos):strnposr($haystack,$needle,$occurrence-1,strpos($haystack, $needle, $pos) + 1);
}

//gjh42
//replace every nth occurrence of $needle with $repl, starting from any position
function str_replace_int($needle, $repl, $haystack, $interval, $first=1, $pos=0) {
  if ($pos >= strlen($haystack) or substr_count($haystack, $needle, $pos) < $first) return
$haystack;
  $firstpos = strnposr($haystack, $needle, $first, $pos);
  $nl = strlen($needle);
  $qty = floor(substr_count($haystack, $needle, $firstpos + 1)/$interval);
  do { //in reverse order
    $nextpos = strnposr($haystack, $needle, ($qty * $interval) + 1, $firstpos);
    $qty--;
    $haystack = substr_replace($haystack, $repl, $nextpos, $nl);
  } while ($nextpos > $firstpos);
  return $haystack;
}
  //$needle = string to find
  //$repl = string to replace needle
  //$haystack = string to do replacing in
  //$interval = number of needles in loop
  //$first=1 = first occurrence of needle to replace (defaults to first)
  //$pos=0 = position in haystack string to start from (defaults to first)

//replace every nth occurrence of $needle with $repl, starting from any position, in a single-
level array
function arr_replace_int($needle, $repl, $arr, $interval, $first=1, $pos=0, $glue='|+|') {
  if (!is_array($arr))  return $arr;
  foreach($arr as $key=>$value){
    if (is_array($arr[$key])) return $arr;
  }
  $haystack = implode($glue, $arr);
  $haystack = str_replace_int($needle, $repl, $haystack, $interval, $first, $pos);
  $tarr = explode($glue, $haystack);
  $i = 0;
  foreach($arr as $key=>$value){
    $arr[$key] = $tarr[$i];
    $i++;
  }
  return $arr;
}
?>
If $arr is not an array, or a multilevel array, it is returned unchanged.
```

up
down
-2
sunmacet at gmail dot com ¶
1 year ago

```
To check that a substring is present.

Confusing check if position is not false:

if ( strpos ( $haystack , $needle ) !== FALSE )

Logical check if there is position:

if ( is_int ( strpos ( $haystack , $needle ) ) )
```

up
down
-4
hu60 dot cn at gmail dot com ¶
3 years ago

```
A more accurate imitation of the PHP function session_start().

Function my_session_start() does something similar to session_start() that has the default
configure, and the session files generated by the two are binary compatible.

The code may help people increase their understanding of the principles of the PHP session.

<?php
error_reporting(E_ALL);
ini_set('display_errors', true);
ini_set('session.save_path', __DIR__);

my_session_start();

echo '<p>session id: '.my_session_id().'</p>';

echo '<code><pre>';
var_dump($_SESSION);
echo '</pre></code>';

$now = date('H:i:s');
if (isset($_SESSION['last_visit_time'])) {
  echo '<p>Last Visit Time: '.$_SESSION['last_visit_time'].'</p>';
}
echo '<p>Current Time: '.$now.'</p>';

$_SESSION['last_visit_time'] = $now;

function my_session_start() {
  global $phpsessid, $sessfile;
  if (!isset($_COOKIE['PHPSESSID']) || empty($_COOKIE['PHPSESSID'])) {
    $phpsessid = my_base32_encode(my_random_bytes(16));
    setcookie('PHPSESSID', $phpsessid, ini_get('session.cookie_lifetime'),
ini_get('session.cookie_path'), ini_get('session.cookie_domain'),
ini_get('session.cookie_secure'), ini_get('session.cookie_httponly'));
  } else {
    $phpsessid = substr(preg_replace('/[^a-z0-9]/', '', $_COOKIE['PHPSESSID']), 0, 26);
  }

  $sessfile = ini_get('session.save_path').'/sess_'.$phpsessid;
  if (is_file($sessfile)) {
    $_SESSION = my_unserialize(file_get_contents($sessfile));
  } else {
    $_SESSION = array();
  }
  register_shutdown_function('my_session_save');
}

function my_session_save() {
  global $sessfile;

  file_put_contents($sessfile, my_serialize($_SESSION));
}

function my_session_id() {
  global $phpsessid;
  return $phpsessid;
}

function my_serialize($data) {
  $text = '';
  foreach ($data as $k=>$v) {
    // key cannot contains '|'
    if (strpos($k, '|') !== false) {
      continue;
    }
    $text.=$k.'|'.serialize($v)."\n";
  }
  return $text;
}

function my_unserialize($text) {
  $data = [];
  $text = explode("\n", $text);
  foreach ($text as $line) {
    $pos = strpos($line, '|');
    if ($pos === false) {
      continue;
    }
    $data[substr($line, 0, $pos)] = unserialize(substr($line, $pos + 1));
  }
  return $data;
}

function my_random_bytes($length) {
  if (function_exists('random_bytes')) {
      return random_bytes($length);
  }
  $randomString = '';
  for ($i = 0; $i < $length; $i++) {
      $randomString .= chr(rand(0, 255));
  }
  return $randomString;
}

function my_base32_encode($input) {
  $BASE32_ALPHABET = 'abcdefghijklmnopqrstuvwxyz234567';
  $output = '';
  $v = 0;
  $vbits = 0;
  for ($i = 0, $j = strlen($input); $i < $j; $i++) {
    $v <<= 8;
    $v += ord($input[$i]);
    $vbits += 8;
    while ($vbits >= 5) {
      $vbits -= 5;
      $output .= $BASE32_ALPHABET[$v >> $vbits];
      $v &= ((1 << $vbits) - 1);
    }
  }
  if ($vbits > 0) {
    $v <<= (5 - $vbits);
    $output .= $BASE32_ALPHABET[$v];
  }
  return $output;
}
```

up
down
-4
binodluitel at hotmail dot com ¶
8 years ago

```
This function will return 0 if the string that you are searching matches i.e. needle matches the
haystack

{code}
echo strpos('bla', 'bla');
{code}

Output: 0
```

up
down
-4
msegit post pl ¶
4 years ago

```
This might be useful, I often use for parsing file paths etc.
(Some examples inside https://gist.github.com/msegu/bf7160257037ec3e301e7e9c8b05b00a )
<?php
/**
* Function 'strpos_' finds the position of the first or last occurrence of a substring in a
string, ignoring number of characters
*
* Function 'strpos_' is similar to 'str[r]pos()', except:
* 1. fourth (last, optional) param tells, what to return if str[r]pos()===false
* 2. third (optional) param $offset tells as of str[r]pos(), BUT if negative (<0) search starts
-$offset characters counted from the end AND skips (ignore!, not as 'strpos' and 'strrpos')
-$offset-1 characters from the end AND search backwards
```

```
 *
 * @param string $haystack Where to search
 * @param string $needle What to find
 * @param int $offset (optional) Number of characters to skip from the beginning (if 0, >0) or from
the end (if <0) of $haystack
 * @param mixed $resultIfFalse (optional) Result, if not found
 *    Example:
 *    positive $offset - like strpos:
 *        strpos_('abcaba','ab',1)==strpos('abcaba','ab',1)==3, strpos('abcaba','ab',4)===false,
strpos_('abcaba','ab',4,'Not found')==='Not found'
 *    negative $offset - similar to strrpos:
 *        strpos_('abcaba','ab',-1)==strpos('abcaba','ab',-1)==3, strrpos('abcaba','ab',-3)==3 BUT
strpos_('abcaba','ab',-3)===0 (omits 2 characters from the end, because -2-1=-3, means search in
'abca'!)
 *
 * @result int $offset Returns offset (or false), or $resultIfFalse
 */
function strpos_($haystack, $needle, $offset = 0, $resultIfFalse = false) {
    $haystack=((string)$haystack);    // (string) to avoid errors with int, float...
    $needle=((string)$needle);
    if ($offset>=0) {
        $offset=strpos($haystack, $needle, $offset);
        return (($offset===false)? $resultIfFalse : $offset);
    } else {
        $haystack=strrev($haystack);
        $needle=strrev($needle);
        $offset=strpos($haystack,$needle,-$offset-1);
        return (($offset===false)? $resultIfFalse : strlen($haystack)-$offset-strlen($needle));
    }
}
?>
```

+ add a note

  - 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