PERL PACK FUNCTION

Description

This function evaluates the expressions in LIST and packs them into a binary structure specified by EXPR. The format is specified using the characters shown in Table below −

Each character may be optionally followed by a number, which specifies a repeat count for the type of value being packed.that is nibbles, chars, or even bits, according to the format. A value of * repeats for as many values remain in LIST. Values can be unpacked with the unpack function.

For example, a5 indicates that five letters are expected. b32 indicates that 32 bits are expected. h8 indicates that 8 nybbles *or4bytes* are expected. P10 indicates that the structure is 10 bytes long.

Syntax

Following is the simple syntax for this function −

```
pack EXPR, LIST
```

Return Value

- This function returns a packed version of the data in LIST using TEMPLATE to determine how it is coded.

Here is the table which gives values to be used in TEMPLATE.

Character	Description
a	ASCII character string padded with null characters
A	ASCII character string padded with spaces
b	String of bits, lowest first
B	String of bits, highest first
c	A signed character *rangeusually* − 128*to*127
C	An unsigned character *usually*8*bits*
d	A double-precision floating-point number
f	A single-precision floating-point number
h	Hexadecimal string, lowest digit first
H	Hexadecimal string, highest digit first
i	A signed integer
I	An unsigned integer
l	A signed long integer
L	An unsigned long integer
n	A short integer in network order
N	A long integer in network order
p	A pointer to a string

s	A signed short integer
S	An unsigned short integer
u	Convert to uuencode format
v	A short integer in VAX $little-endian$ order
V	A long integer in VAX order
x	A null byte
X	Indicates "go back one byte"
@	Fill with nulls $ASCII0$

Example

Following is the example code showing its basic usage −

```
#!/usr/bin/perl -w

$bits = pack("c", 65);
# prints A, which is ASCII 65.
print "bits are $bits\n";
$bits = pack( "x" );
# $bits is now a null chracter.
print "bits are $bits\n";
$bits = pack( "sai", 255, "T", 30 );
# creates a seven charcter string on most computers'
print "bits are $bits\n";

@array = unpack( "sai", "$bits" );

#Array now contains three elements: 255, T and 30.
print "Array $array[0]\n";
print "Array $array[1]\n";
print "Array $array[2]\n";
```

When above code is executed, it produces the following result −

```
bits are A
bits are
bits are T
Array 255
Array T
Array 30
```