Perl pack/unpack Summary

Usage:

- **pack TEMPLATE, LIST**
- **unpack TEMPLATE, EXPR**

Template parameter summary:

Parameter	Description
a	A string with arbitrary binary data, will be null padded.
A	A text (ASCII) string, will be space padded.
Z	A null terminated (ASCIZ) string, will be null padded.
b	A bit string (ascending bit order inside each byte, like vec()).
B	A bit string (descending bit order inside each byte).
h	A hex string (low nibble first).
H	A hex string (high nibble first).
c	A signed char value.
C	An unsigned char value. Only does bytes. See U for Unicode.
s	A signed short value.
S	An unsigned short value. (This 'short' is _exactly_ 16 bits, which may differ from what a local C compiler calls 'short'. If you want native-length shorts, use the '!' suffix.)
i	A signed integer value.
I	An unsigned integer value. (This 'integer' is _at_least_ 32 bits wide. Its exact size depends on what a local C compiler calls 'int', and may even be larger than the 'long' described in the next item.)
l	A signed long value.
L	An unsigned long value. (This 'long' is _exactly_ 32 bits, which may differ from what a local C compiler calls 'long'. If you want native-length longs, use the '!' suffix.)
n	An unsigned short in "network" (big-endian) order.
N	An unsigned long in "network" (big-endian) order.
v	An unsigned short in "VAX" (little-endian) order.
V	An unsigned long in "VAX" (little-endian) order. (These 'shorts' and 'longs' are _exactly_ 16 bits and _exactly_ 32 bits, respectively.)
q	A signed quad (64-bit) value.
Q	An unsigned quad value. (Quads are available only if your system supports 64-bit integer values _and_ if Perl has been compiled to support those. Causes a fatal error otherwise.)
j	A signed integer value (a Perl internal integer, IV).
J	An unsigned integer value (a Perl internal unsigned integer, UV).
f	A single-precision float in the native format.
d	A double-precision float in the native format.
F	A floating point value in the native native format. (a Perl internal floating point value, NV).
D	A long double-precision float in the native format. (Long doubles are available only if your system supports long double values _and_ if Perl has been compiled to support those. Causes a fatal error otherwise.)
p	A pointer to a null-terminated string.
P	A pointer to a structure (fixed-length string).
u	A uuencoded string.
U	A Unicode character number. Encodes to UTF-8 internally (or UTF-EBCDIC in EBCDIC platforms).
w	A BER compressed integer (not an ASN.1 BER, see perlpacktut for details). Its bytes represent an unsigned integer in base 128, most significant digit first, with as few digits as possible. Bit eight (the high bit) is set on each byte except the last.
x	A null byte.
X	Back up a byte.
@	Null fill to absolute position, counted from the start of the innermost ()-group.
(	Start of a ()-group.

Perl [s]printf Summary

Usage:

- **printf FILEHANDLE FORMAT, LIST**
- **printf FORMAT, LIST**
- **sprintf FORMAT, LIST**

Format summary:

Format	Description
% %	a percent sign
% c	a character with the given number
% s	a string
% d	a signed integer, in decimal
% u	an unsigned integer, in decimal
% o	an unsigned integer, in octal
% x	an unsigned integer, in hexadecimal
% e	a floating-point number, in scientific notation
% f	a floating-point number, in fixed decimal notation
% g	a floating-point number, in %e or %f notation
% X	like %x, but using upper-case letters
% E	like %e, but using an upper-case "E"
% G	like %g, but with an upper-case "E" (if applicable)
% b	an unsigned integer, in binary
% p	a pointer (outputs the Perl value's address in hexadecimal)
% n	special: *stores* the number of characters output so far into the next variable in the parameter list
% l	a synonym for %d
% D	a synonym for %ld
% U	a synonym for %lu
% O	a synonym for %lo
% F	a synonym for %f

Between the % and the format letter, you may specify a number of **additional attributes** controlling the interpretation of the format.

Attribute summary:

<table>
<tr><th>Attribute</th><th>Description</th></tr>
<tr><td>format parameter index</td><td>An explicit format parameter index, such as 2$. By default sprintf will format the next unused argument in the list, but this allows you to take the arguments out of order, e.g.:
<code>printf '%3$d %d %1$d', 1, 2, 3; # prints "3 1 1"</code></td></tr>
<tr><td>flags</td><td>
<table>
<tr><th>Flag</th><th>Description</th></tr>
<tr><td>space</td><td>prefix positive number with a space</td></tr>
<tr><td>+</td><td>prefix positive number with a plus sign</td></tr>
<tr><td>-</td><td>left-justify within the field</td></tr>
<tr><td>0</td><td>use zeros, not spaces, to right-justify</td></tr>
<tr><td>#</td><td>prefix non-zero octal with "0", non-zero hex with "0x", non-zero binary with "0b"</td></tr>
</table>
</td></tr>
<tr><td>vector flag</td><td>This flag tells perl to interpret the supplied string as a vector of integers, one for each character in the string. Perl applies the format to each integer in turn, then joins the resulting strings with a separator (a dot . by default).
Put an asterisk * before the v to override the string to use to separate num’s.</td></tr>
<tr><td>(minimum) width</td><td>Arguments are usually formatted to be only as wide as required to display the given value. You can override the width by putting a number here, or get the width from the next argument (with *) or from a specified argument (with e.g. *2$). If a field width obtained through * is negative, it has the same effect as the - flag: left-justification.</td></tr>
<tr><td>precision, or maximum width</td><td>You can specify a precision (for numeric conversions) or a maximum width (for string conversions) by specifying a . followed by a number. For floating point formats, with the exception of 'g' and 'G', this specifies the number of decimal places to show (the default being 6)</td></tr>
<tr><td>size</td><td>l interpret integer as C type "long" or "unsigned long"
h interpret integer as C type "short" or "unsigned short"
q, L or ll inerpret integer as C type "long long", "unsigned long long" or "quads".</td></tr>
<tr><td>order of arguments</td><td>If the format specification uses * to require additional arguments, these are consumed from the argument list in the order in which they appear in the format specification before the value to format.</td></tr>
</table>

peter@catonmat.net
http://www.catonmat.net
good coders code, great reuse