vector flag

The vector flag v, optionally specifying the join string to use. This flag tells perl to interpret the supplied string as a vector of integers, one for each character in the string, separated by a given string (a dot . by default). This can be useful for displaying ordinal values of characters in arbitrary strings:

```
printf "version is v%vd\n", $^V;     # Perl's version
```

Put an asterisk * before the v to override the string to use to separate the numbers:

```
printf "address is %*vX\n", ":", $addr;   # IPv6 address
printf "bits are %0*v8b\n", " ", $bits;   # random bitstring
```

You can also explicitly specify the argument number to use for the join string using eg *2$v:

```
printf '%*4$vX %*4$vX %*4$vX', @addr[1..3], ":";   # 3 IPv6 addresses
```

(minimum) width

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 eg *2$):

```
printf '<%s>', "a";       # prints "<a>"
printf '<%6s>', "a";      # prints "<     a>"
printf '<%*s>', 6, "a";   # prints "<     a>"
printf '<%*2$s>', "a", 6; # prints "<     a>"
printf '<%2s>', "long";   # prints "<long>" (does not truncate)
```

If a field width obtained through * is negative, it has the same effect as the - flag: left-justification.

precision, or maximum width

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), eg:

```
# these examples are subject to system-specific variation
printf '<%f>', 1;    # prints "<1.000000>"
printf '<%.1f>', 1;  # prints "<1.0>"
printf '<%.0f>', 1;  # prints "<1>"
printf '<%e>', 10;   # prints "<1.000000e+01>"
printf '<%.1e>', 10; # prints "<1.0e+01>"
```

For 'g' and 'G', this specifies the maximum number of digits to show, including prior to the decimal point as well as after it, eg:

```
# these examples are subject to system-specific variation
printf '<%g>', 1;        # prints "<1>"
printf '<%.10g>', 1;     # prints "<1>"
printf '<%g>', 100;      # prints "<100>"
printf '<%.1g>', 100;    # prints "<1e+02>"
printf '<%.2g>', 100.01; # prints "<1e+02>"
printf '<%.5g>', 100.01; # prints "<100.01>"
printf '<%.4g>', 100.01; # prints "<100>"
```

For integer conversions, specifying a precision implies that the output of the number itself should be zero-padded to this width:

```
printf '<%.6x>', 1;      # prints "<000001>"
printf '<%#.6x>', 1;     # prints "<0x000001>"
printf '<%-10.6x>', 1;   # prints "<000001    >"
```

For string conversions, specifying a precision truncates the string to fit in the specified width: