```
split(/([,-])/, "1-10,20", 3);
```

produces the list value

```
(1, '-', 10, ',', 20)
```

If you had the entire header of a normal Unix email message in $header, you could split it up into fields and their values this way:

```
$header =~ s/\n\s+/ /g;  # fix continuation lines
%hdrs   =  (UNIX_FROM => split /^(\S*?):\s*/m, $header);
```

The pattern /PATTERN/ may be replaced with an expression to specify patterns that vary at runtime. (To do runtime compilation only once, use /$variable/o.)

As a special case, specifying a PATTERN of space (' ') will split on white space just as split with no arguments does. Thus, split(' ') can be used to emulate **awk**'s default behavior, whereas split(/ /) will give you as many null initial fields as there are leading spaces. A split on /\s+/ is like a split(' ') except that any leading whitespace produces a null first field. A split with no arguments really does a split(' ', $_) internally.

A PATTERN of /^/ is treated as if it were /^/m, since it isn't much use otherwise.

Example:

```
open(PASSWD, '/etc/passwd');
while (<PASSWD>) {
    chomp;
    ($login, $passwd, $uid, $gid,
     $gcos, $home, $shell) = split(/:/);
    #...
}
```

As with regular pattern matching, any capturing parentheses that are not matched in a split() will be set to undef when returned:

```
@fields = split /(A)|B/, "1A2B3";
# @fields is (1, 'A', 2, undef, 3)
```

sprintf FORMAT, LIST

Returns a string formatted by the usual printf conventions of the C library function sprintf. See below for more details and see *sprintf*(3) or *printf*(3) on your system for an explanation of the general principles.

For example:

```
# Format number with up to 8 leading zeroes
$result = sprintf("%08d", $number);

# Round number to 3 digits after decimal point
$rounded = sprintf("%.3f", $number);
```

Perl does its own sprintf formatting–it emulates the C function sprintf, but it doesn't use it (except for floating-point numbers, and even then only the standard modifiers are allowed). As a result, any non-standard extensions in your local sprintf are not available from Perl.

Unlike printf, sprintf does not do what you probably mean when you pass it an array as your first argument. The array is given scalar context, and instead of using the 0th element of the array as the format, Perl will use the count of elements in the array as the format, which is almost never useful.

Perl's sprintf permits the following universally-known conversions: