```
$line = <STDIN>;
$line = readline(*STDIN);           # same thing
```

If readline encounters an operating system error, `$!` will be set with the corresponding error message. It can be helpful to check `$!` when you are reading from filehandles you don't trust, such as a tty or a socket. The following example uses the operator form of `readline`, and takes the necessary steps to ensure that `readline` was successful.

```
for (;;) {
    undef $!;
    unless (defined( $line = <> )) {
        die $! if $!;
        last; # reached EOF
    }
    # ...
}
```

readlink EXPR

readlink

Returns the value of a symbolic link, if symbolic links are implemented. If not, gives a fatal error. If there is some system error, returns the undefined value and sets `$!` (errno). If EXPR is omitted, uses `$_`.

readpipe EXPR

EXPR is executed as a system command. The collected standard output of the command is returned. In scalar context, it comes back as a single (potentially multi-line) string. In list context, returns a list of lines (however you've defined lines with `$/` or `$INPUT_RECORD_SEPARATOR`). This is the internal function implementing the `qx/EXPR/` operator, but you can use it directly. The `qx/EXPR/` operator is discussed in more detail in I/O Operators in *perlop*.

recv SOCKET,SCALAR,LENGTH,FLAGS

Receives a message on a socket. Attempts to receive LENGTH characters of data into variable SCALAR from the specified SOCKET filehandle. SCALAR will be grown or shrunk to the length actually read. Takes the same flags as the system call of the same name. Returns the address of the sender if SOCKET's protocol supports this; returns an empty string otherwise. If there's an error, returns the undefined value. This call is actually implemented in terms of recvfrom(2) system call. See UDP: Message Passing in *perlipc* for examples.

Note the *characters*: depending on the status of the socket, either (8-bit) bytes or characters are received. By default all sockets operate on bytes, but for example if the socket has been changed using binmode() to operate with the `:utf8` I/O layer (see the `open` pragma, *open*), the I/O will operate on UTF-8 encoded Unicode characters, not bytes. Similarly for the `:encoding` pragma: in that case pretty much any characters can be read.

redo LABEL

redo

The `redo` command restarts the loop block without evaluating the conditional again. The `continue` block, if any, is not executed. If the LABEL is omitted, the command refers to the innermost enclosing loop. This command is normally used by programs that want to lie to themselves about what was just input:

```
# a simpleminded Pascal comment stripper
# (warning: assumes no { or } in strings)
LINE: while (<STDIN>) {
    while (s|({.*}.*){.*}|$1 |) {}
    s|{.*}| |;
    if (s|{.*| |) {
        $front = $_;
        while (<STDIN>) {
```