```
 1: // $Id: bcat.c,v 1.2 2012-02-01 13:40:11-08 - - $
 2:
 3: //
 4: // NAME
 5: //    bcat - concatenate and display files
 6: //
 7: // SYNOPSIS
 8: //    bcat [filename...]
 9: //
10: // DESCRIPTION
11: //    The ‘bcat’ utility reads each file in sequence and writes it
12: //    to stdout.  If any filename is given as the single character
13: //    ‘-’, stdin is read at that point.  If no filenames are given
14: //    then stdin is read as the only file.
15: //
16:
17: #include <errno.h>
18: #include <libgen.h>
19: #include <stdarg.h>
20: #include <stdio.h>
21: #include <stdlib.h>
22: #include <string.h>
23:
24: //
25: // cat -
26: // Copy the contents of an already-opened file to stdout.
27: //
28:
29: void catfile (FILE *input) {
30:    for (;;) {
31:       int byte = getc (input);
32:       if (byte == EOF) break;
33:       (void) putchar (byte);
34:    };
35: }
36:
```

```
37:
38: //
39: // main -
40: // Loop over files, if any, and cat each of them to stdout.
41: // Print error messages if appropriate.
42: //
43:
44: int main (int argc, char **argv) {
45:    int exit_status = EXIT_SUCCESS;
46:    char *progname = basename (argv[0]);
47:    if (argc == 1) {
48:       catfile (stdin);
49:    }else{
50:       for (int argi = 1; argi < argc; ++argi) {
51:          if (strcmp (argv[argi], "-") == 0) {
52:             catfile (stdin);
53:          }else{
54:             FILE *input = fopen (argv[argi], "r");
55:             if (input == NULL) {
56:                fflush (NULL);
57:                fprintf (stderr, "%s: %s: %s\n", progname,
58:                         argv[argi], strerror (errno));
59:                fflush (NULL);
60:                exit_status = EXIT_FAILURE;
61:             }else{
62:                catfile (input);
63:                fclose (input);
64:             };
65:          };
66:       };
67:    };
68:    return exit_status;
69: }
70:
```

```
 1: /*
 2: ****************************************************************
 3:
 4: Whenever a man page is referenced, read it online.  For example,
 5: when we refer to 'stdio(3c)', you can read it with ''man -s 3C
 6: stdio''.
 7:
 8: As described in stdio(3c), there are three FILE* handles that
 9: are always opened when a program starts:  'stdin', 'stdout', and
10: 'stderr'.  These are, respectively, the standard input, standard
11: output, and standard error.  Normal output is written to stdout,
12: while error messages are written to stderr.
13:
14: The usual format of an error message is something like:
15: .   progname: object_or_function: reason
16: The reason a system call has failed is given in the external
17: variable 'errno'.  This can be translated into English via
18: strerror(3c).
19:
20: 'fopen(3c)' opens a file and returns a pointer to a 'FILE',
21: given a filename.  'fclose(3c)' closes that file, given a
22: FILE*.  'putchar(3c)' writes one byte to stdout.  'getc(3c)'
23: reads one byte from the FILE* given as an argument and returns
24: an int containing the character, if one exists.  If not, returns
25: EOF (-1).  Note that end of line is signalled by '\n'.  To
26: signal EOF from a Unix terminal, type Control/D as the first
27: character on a line.
28:
29: Strings are represented as arrays of characters.  Each string
30: ends with a null plug ('\0').  'strcmp(3c)' compares two such
31: character strings and returns a number that is <, =, or > 0,
32: depending on the relationship.  See Java's compareTo function.
33:
34: Some functions return values instead of void, but we often don't
35: care what these values are, so we use the function in a
36: statement context.  This causes lint(1) to complain:  ''function
37: returns value which is always ignored''.  So we explicitly cast
38: the results of these functions to (void) in order to suppress
39: this error.  Alternately we could have use a drop-in macro to
40: replace them.
41:
42: The call fflush (NULL) causes all opened FILE* handles to be
43: flushed.  When a program writes data, it is buffered in memory
44: before being written to the disk.  This causes immediate writing
45: instead of waiting until the buffer is full.  We do this so that
46: anything written to stdout and stderr are properly interleaved.
47:
48: ****************************************************************
49: */
```

```
1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting bcat.c
2: bcat.c: $Id: bcat.c,v 1.2 2012-02-01 13:40:11-08 - - $
3: gcc -g -O0 -Wall -Wextra -std=gnu99 bcat.c -o bcat -lm
4: lint -Xa -fd -m -u -x -errchk=%all bcat.c
5:
6: function returns value which is always ignored
7:     fclose              fflush              fprintf
8: rm -f bcat.o
9: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished bcat.c
```