```
  if (argc != 3) {
    fprintf(stderr, "usage: fcopy source dest\n");
    exit(EXIT_FAILURE);
  }

  if ((source_fp = fopen(argv[1], "rb")) == NULL) {
    fprintf(stderr, "Can't open %s\n", argv[1]);
    exit(EXIT_FAILURE);
  }

  if ((dest_fp = fopen(argv[2], "wb")) == NULL) {
    fprintf(stderr, "Can't open %s\n", argv[2]);
    fclose(source_fp);
    exit(EXIT_FAILURE);
  }

  while ((ch = getc(source_fp)) != EOF)
    putc(ch, dest_fp);

  fclose(source_fp);
  fclose(dest_fp);
  return 0;
}
```

Using `"rb"` and `"wb"` as the file modes enables `fcopy` to copy both text and binary files. If we used `"r"` and `"w"` instead, the program wouldn't necessarily be able to copy binary files.

22.5 Line I/O

We'll now turn to library functions that read and write lines. These functions are used mostly with text streams, although it's legal to use them with binary streams as well.

Output Functions

```
int fputs(const char * restrict s,
          FILE * restrict stream);
int puts(const char *s);
```

puts We encountered the `puts` function in Section 13.3; it writes a string of characters to `stdout`:

```
puts("Hi, there!");    /* writes to stdout */
```

After it writes the characters in the string, `puts` always adds a new-line character.