```
 1: // $Id: buffercp.cpp,v 1.5 2014-06-02 16:10:33-07 - - $
 2:
 3: //
 4: // Queue of buffers and file I/O.
 5: //
 6:
 7: #include <array>
 8: #include <cassert>
 9: #include <cstring>
10: #include <fstream>
11: #include <iostream>
12: #include <memory>
13: #include <queue>
14: using namespace std;
15:
16: string execname;
17: int exit_status {EXIT_SUCCESS};
18:
19: struct exec_error: public runtime_error {
20:    exec_error (const string& message): runtime_error (message) {
21:       assert (execname.size() > 0);
22:    }
23: };
24:
25: struct sys_errno: public exec_error {
26:    sys_errno (const string& obj):
27:               exec_error (execname + ": " + obj + ": "
28:                           + strerror (errno)){}
29: };
30:
31: void usage () {
32:    throw exec_error ("Usage: " + execname + " infile outfile");
33: }
34:
```

```
35:
36: struct buffer {
37:    static constexpr size_t MAX_BYTES = 0x100;
38:    size_t nbytes {};
39:    array<char,MAX_BYTES>  bytes;
40: };
41: using buffer_uptr = unique_ptr<buffer>;
42: using buffer_queue = queue<buffer_uptr>;
43:
44: buffer_queue readfile (const string& filename) {
45:    buffer_queue que;
46:    ifstream infile {filename};
47:    if (infile.fail()) throw sys_errno (filename);
48:    while (not infile.eof()) {
49:       buffer_uptr uptr {new buffer()};
50:       infile.read (uptr->bytes.data(), uptr->MAX_BYTES);
51:       uptr->nbytes = infile.gcount();
52:       que.push (std::move (uptr));
53:    }
54:    return std::move (que);
55: }
56:
57: void writefile (const string& filename, buffer_queue& que) {
58:    ofstream outfile {filename};
59:    if (outfile.fail()) throw sys_errno (filename);
60:    while (not que.empty()) {
61:       buffer_uptr uptr = std::move (que.front());
62:       que.pop();
63:       outfile.write (uptr->bytes.data(), uptr->nbytes);
64:    }
65: }
66:
67: int main (int argc, char** argv) {
68:    execname = basename (argv[0]);
69:    try {
70:       if (argc != 3) usage();
71:       buffer_queue que = readfile (argv[1]);
72:       writefile (argv[2], que);
73:    }catch (exec_error& error) {
74:       cerr << error.what() << endl;;
75:       exit_status = EXIT_FAILURE;
76:    }
77:    return exit_status;
78: }
```