```
// main.cc
//  Bootstrap code to initialize the operating system kernel.
//
//  Allows direct calls into internal operating system functions,
//  to simplify debugging and testing.  In practice, the
//  bootstrap code would just initialize data structures,
//  and start a user program to print the login prompt.
//
//  Most of this file is not needed until later assignments.
//
// Usage: nachos -d <debugflags> -rs <random seed #>
//      -s -x <nachos file> -c <consoleIn> <consoleOut>
//      -f -cp <unix file> <nachos file>
//      -p <nachos file> -r <nachos file> -l -D -t
//              -n <network reliability> -e <network orderability>
//              -m <machine id>
//              -o <other machine id>
//              -z
//
//    -d causes certain debugging messages to be printed (cf. utility.h)
//    -rs causes Yield to occur at random (but repeatable) spots
//    -z prints the copyright message
//
//  USER_PROGRAM
//    -s causes user programs to be executed in single-step mode
//    -x runs a user program
//    -c tests the console
//
//  FILESYS
//    -f causes the physical disk to be formatted
//    -cp copies a file from UNIX to Nachos
//    -p prints a Nachos file to stdout
//    -r removes a Nachos file from the file system
//    -l lists the contents of the Nachos directory
//    -D prints the contents of the entire file system
//    -t tests the performance of the Nachos file system
//
//  NETWORK
//    -n sets the network reliability
//    -e sets the network orderability
//    -m sets this machine's host id (needed for the network)
//    -o runs a simple test of the Nachos network software
//
//  NOTE -- flags are ignored until the relevant assignment.
//  Some of the flags are interpreted here; some in system.cc.
//
// Copyright (c) 1992-1993 The Regents of the University of California.
// All rights reserved.  See copyright.h for copyright notice and limitation
// of liability and disclaimer of warranty provisions.

#define MAIN
#include "copyright.h"
#undef MAIN

#include "utility.h"
```

```
 56 #include "system.h"
 57
 58
 59 // External functions used by this file
 60 extern void ThreadTest(void), Copy(char *unixFile, char *nachosFile);
 61 extern void Print(char *file), PerformanceTest(void);
 62 extern void StartProcess(char *file), ConsoleTest(char *in, char *out);
 63 extern void MailTest(int networkID);
 64 extern void SynchTest(void);
 65
 66 //----------------------------------------------------------------------
 67 // main
 68 //  Bootstrap the operating system kernel.
 69 //
 70 //  Check command line arguments
 71 //  Initialize data structures
 72 //  (optionally) Call test procedure
 73 //
 74 //  "argc" is the number of command line arguments (including the name
 75 //      of the command) -- ex: "nachos -d +" -> argc = 3
 76 //  "argv" is an array of strings, one for each command line argument
 77 //      ex: "nachos -d +" -> argv = {"nachos", "-d", "+"}
 78 //----------------------------------------------------------------------
 79
 80 int
 81 main(int argc, char **argv)
 82 {
 83     int argCount;           // the number of arguments
 84                     // for a particular command
 85
 86     DEBUG('t', "Entering main");
 87     (void) Initialize(argc, argv);
 88
 89     ThreadTest(); //Start ThreadTest();
 90
 91     for (argc--, argv++; argc > 0; argc -= argCount, argv += argCount) {
 92     argCount = 1;
 93         if (!strcmp(*argv, "-z"))               // print copyright
 94             printf (copyright);
 95 #ifdef USER_PROGRAM
 96         if (!strcmp(*argv, "-x")) {         // run a user program
 97         ASSERT(argc > 1);
 98             StartProcess(*(argv + 1));
 99             argCount = 2;
100         } else if (!strcmp(*argv, "-c")) {      // test the console
101         if (argc == 1)
102             ConsoleTest(NULL, NULL);
103         else {
104         ASSERT(argc > 2);
105             ConsoleTest(*(argv + 1), *(argv + 2));
106             argCount = 3;
107         }
108         interrupt->Halt();      // once we start the console, then
109                     // Nachos will loop forever waiting
110                     // for console input
```

```
111     }
112 #endif // USER_PROGRAM
113 #ifdef FILESYS
114     if (!strcmp(*argv, "-cp")) {        // copy from UNIX to Nachos
115         ASSERT(argc > 2);
116         Copy(*(argv + 1), *(argv + 2));
117         argCount = 3;
118     } else if (!strcmp(*argv, "-p")) {  // print a Nachos file
119         ASSERT(argc > 1);
120         Print(*(argv + 1));
121         argCount = 2;
122     } else if (!strcmp(*argv, "-r")) {  // remove Nachos file
123         ASSERT(argc > 1);
124         fileSystem->Remove(*(argv + 1));
125         argCount = 2;
126     } else if (!strcmp(*argv, "-l")) {  // list Nachos directory
127             fileSystem->List();
128     } else if (!strcmp(*argv, "-D")) {  // print entire filesystem
129             fileSystem->Print();
130     } else if (!strcmp(*argv, "-t")) {  // performance test
131             PerformanceTest();
132     }
133 #endif // FILESYS
134 #ifdef NETWORK
135         if (!strcmp(*argv, "-o")) {
136         ASSERT(argc > 1);
137             Delay(2);               // delay for 2 seconds
138                         // to give the user time to
139                         // start up another nachos
140             MailTest(atoi(*(argv + 1)));
141             argCount = 2;
142         }
143 #endif // NETWORK
144     }
145
146     currentThread->Finish();    // NOTE: if the procedure "main"
147                 // returns, then the program "nachos"
148                 // will exit (as any other normal program
149                 // would).  But there may be other
150                 // threads on the ready list.  We switch
151                 // to those threads by saying that the
152                 // "main" thread is finished, preventing
153                 // it from returning.
154     return(0);          // Not reached...
155 }
156
```