정보통신실험3

중간고사 샘플 코드

Makefile (1)

```
Common_files = addressUtility.o dieWithMessage.o
LIBS = -lsocket -lnsl
CFLAGS = -xc99 -g

mcs_tcp_td : mcTcpServer.o tcpServerUtility.o mcCommon.o mcEncodingText.o delimeterFramer.o
$(Common_files)
        cc -o mcs_tcp_td $(CFLAGS) $(LIBS) mcTcpServer.o tcpServerUtility.o mcCommon.o  mcEncodingText.o
delimeterFramer.o $(Common_files)

mc1_tcp_td : mcTcpClient1.o tcpClientUtility.o mcCommon.o mcEncodingText.o delimeterFramer.o
$(Common_files)
        cc -o mc1_tcp_td $(CFLAGS) $(LIBS) mcTcpClient1.o tcpClientUtility.o mcCommon.o mcEncodingText.o
delimeterFramer.o $(Common_files)

mc2_tcp_td : mcTcpClient2.o tcpClientUtility.o mcCommon.o mcEncodingText.o delimeterFramer.o
$(Common_files)
        cc -o mc2_tcp_td $(CFLAGS) $(LIBS) mcTcpClient2.o tcpClientUtility.o mcCommon.o mcEncodingText.o
delimeterFramer.o $(Common_files)

mcs_tcp_tl : mcTcpServer.o tcpServerUtility.o mcCommon.o mcEncodingText.o lengthFramer.o  $(Common_files)
        cc -o mcs_tcp_tl $(CFLAGS) $(LIBS) mcTcpServer.o tcpServerUtility.o mcCommon.o  mcEncodingText.o
lengthFramer.o $(Common_files)

mc1_tcp_tl : mcTcpClient1.o tcpClientUtility.o mcCommon.o mcEncodingText.o lengthFramer.o $(Common_files)
        cc -o mc1_tcp_tl $(CFLAGS) $(LIBS) mcTcpClient1.o tcpClientUtility.o mcCommon.o mcEncodingText.o
lengthFramer.o $(Common_files)

mc2_tcp_tl : mcTcpClient2.o tcpClientUtility.o mcCommon.o mcEncodingText.o lengthFramer.o $(Common_files)
        cc -o mc2_tcp_tl $(CFLAGS) $(LIBS) mcTcpClient2.o tcpClientUtility.o mcCommon.o mcEncodingText.o
lengthFramer.o $(Common_files)
```

Makefile (2)

```
mcs_tcp_bd : mcTcpServer.o tcpServerUtility.o mcCommon.o mcEncodingBin.o delimeterFramer.o  $(Common_files)
	cc -o mcs_tcp_bd $(CFLAGS) $(LIBS) mcTcpServer.o tcpServerUtility.o mcCommon.o  mcEncodingBin.o
delimeterFramer.o $(Common_files)

mc1_tcp_bd : mcTcpClient1.o tcpClientUtility.o mcCommon.o mcEncodingBin.o delimeterFramer.o $(Common_files)
	cc -o mc1_tcp_bd $(CFLAGS) $(LIBS) mcTcpClient1.o tcpClientUtility.o mcCommon.o mcEncodingBin.o
delimeterFramer.o $(Common_files)

mc2_tcp_bd : mcTcpClient2.o tcpClientUtility.o mcCommon.o mcEncodingBin.o delimeterFramer.o $(Common_files)
	cc -o mc2_tcp_bd $(CFLAGS) $(LIBS) mcTcpClient2.o tcpClientUtility.o mcCommon.o mcEncodingBin.o
delimeterFramer.o $(Common_files)

mcs_tcp_bl : mcTcpServer.o tcpServerUtility.o mcCommon.o mcEncodingBin.o lengthFramer.o  $(Common_files)
	cc -o mcs_tcp_bl $(CFLAGS) $(LIBS) mcTcpServer.o tcpServerUtility.o mcCommon.o  mcEncodingBin.o
lengthFramer.o $(Common_files)

mc1_tcp_bl : mcTcpClient1.o tcpClientUtility.o mcCommon.o mcEncodingBin.o lengthFramer.o $(Common_files)
	cc -o mc1_tcp_bl $(CFLAGS) $(LIBS) mcTcpClient1.o tcpClientUtility.o mcCommon.o mcEncodingBin.o
lengthFramer.o $(Common_files)

mc2_tcp_bl : mcTcpClient2.o tcpClientUtility.o mcCommon.o mcEncodingBin.o lengthFramer.o $(Common_files)
	cc -o mc2_tcp_bl $(CFLAGS) $(LIBS) mcTcpClient2.o tcpClientUtility.o mcCommon.o mcEncodingBin.o
lengthFramer.o $(Common_files)

clean :
	rm *.o
	rm mcs_tcp_* mc1_tcp_* mc2_tcp_*

all : mcs_tcp_td mc1_tcp_td mc2_tcp_td mcs_tcp_tl mc1_tcp_tl mc2_tcp_tl mcs_tcp_bd mc1_tcp_bd mc2_tcp_bd mcs_tcp_bl
mc1_tcp_bl mc2_tcp_bl
```

mcProtocol.h

```
 1  #include <stdbool.h>
 2
 3  #define MAX_WIRE_SIZE                   60
 4  #define MAX_MENU_NUMBER                 10
 5  #define MAX_MENU_NAME_LENGTH            40
 6
 7  #define DATA_FILE_NAME          "mcMenuName.dat"
 8  #define RECORD_FILE_NAME        "mcOrderList.dat"
 9
10  typedef struct OrderInfo {
11    int menu_no;
12    char menu_name[MAX_MENU_NAME_LENGTH];
13    int number;
14    bool isRequestByNumber;
15    bool isRequestByName;
16    bool isResponse;
17  } OrderInfo;
18
19  void PrintOrderInfo(OrderInfo *info);
20  void InitMenuTable(char nameList[][MAX_MENU_NAME_LENGTH], int *no_of_item);
21  void PrintMenuTable(char nameList[][MAX_MENU_NAME_LENGTH], int no_of_item);
22  int SearchByName(char nameList[][MAX_MENU_NAME_LENGTH], char* target);
23  void RecordOrderInfo (FILE *fd, OrderInfo *info);
```

mcCommon.c (1)

```
 1  #include <stdio.h>
 2  #include <sys/types.h>
 3  #include <string.h>
 4  #include <time.h>
 5  #include "mcProtocol.h"
 6
 7  /*****************************************************************************/
 8  void InitMenuTable(char nameList[][MAX_MENU_NAME_LENGTH], int *no_of_items)
 9  {
10          FILE *data_f = fopen(DATA_FILE_NAME, "r");
11
12          int i;
13          for (i = 0; i < MAX_MENU_NUMBER ; i++) {
14                  if (!fgets(nameList[i], MAX_MENU_NAME_LENGTH, data_f))
15                          break;  // EOF
16                  nameList[i][strlen(nameList[i]) - 1] = '\0'; // remove newline
17          }
18
19          *no_of_items = i;
20          fclose(data_f);
21  }
22
```

mcCommon.c (2)

```
23  /*************************************************************************/
24  void PrintMenuTable(char nameList[][MAX_MENU_NAME_LENGTH], int no_of_items)
25  {
26          printf("\nMcDowell Menu ########\n");
27          printf("No \t Name \n");
28          for (int i = 1; i < no_of_items; i++) {
29                  printf("%2d \t %s \n", i, nameList[i]);
30          }
31  }
32
33  /*************************************************************************/
34  int SearchByName(char nameList[][MAX_MENU_NAME_LENGTH], char* target)
35  {
36          int index = 0;
37
38          for (int i = 1; !index && nameList[i]; i++) {
39                  if (strncmp(nameList[i], target, MAX_MENU_NAME_LENGTH) == 0) {
40                          fprintf(stderr, "FOUND\n");
41                          index = i;
42                  }
43          }
44
45          return index;
46  }
```

mcCommon.c (3)

```
48  /*****************************************************************************/
49  void PrintOrderInfo (OrderInfo *info)
50  {
51          if (info->isRequestByNumber) {
52                  printf("Request Menu : %d \t Number : %d\n",
53                                  info->menu_no, info->number);
54          }
55          else if (info->isRequestByName) {
56                  printf("Request Menu : %s \t Number : %d\n",
57                                  info->menu_name, info->number);
58          }
59          else if (info->isResponse) {
60                  printf("Response Menu : %d \t Name: %s \t Number : %d\n",
61                                  info->menu_no, info->menu_name, info->number);
62          }
63          else {
64                  printf("Invalid data\n");
65          }
66  }
67
```

mcCommon.c (4)

```
68  /*****************************************************************************/
69  void RecordOrderInfo (FILE *fd, OrderInfo *info)
70  {
71          time_t now;
72
73          time(&now);
74          fprintf(fd, "%s \t Menu No : %2d \t %40s %d\n",
75                              ctime(&now), info->menu_no, info->menu_name, info->number);
76          fflush(fd);
77  }
```

mcEncoding.h

```
1  #include <stdint.h>
2  #include <stdbool.h>
3  #include <unistd.h>
4
5  bool Decode(uint8_t *inBuf, size_t mSize, OrderInfo *v);
6  size_t Encode(OrderInfo *v, uint8_t *outBuf, size_t bufSize);
```

mcEncodingText.c (1)

```
 1  #include <string.h>
 2  #include <stdint.h>
 3  #include <stdbool.h>
 4  #include <stdlib.h>
 5  #include <stdio.h>
 6  #include <string.h>
 7  #include "Practical.h"
 8  #include "mcProtocol.h"
 9
10  static const char *MAGIC = "mcOrder";
11  static const char *RQ_NAME_STR = "N";
12  static const char *RQ_NUMBER_STR = "I";
13  static const char *RSP_STR = "R";
14  static const char *DELIMSTR = " ";
15  enum {
16    BASE = 10
17  };
18
```

mcEncodingText.c (2)

```
19  /* Encode voting message info as a text string.
20   * WARNING: Message will be silently truncated if buffer is too small!
21   * Invariants (e.g. 0 <= candidate <= 1000) not checked.
22   */
23  size_t Encode(const OrderInfo *oi, uint8_t *outBuf, const size_t bufSize)
24  {
25          uint8_t *bufPtr = outBuf;
26          long size = (size_t) bufSize;
27          int rv = snprintf((char *) bufPtr, size, "%s %s %d %s %d",
28                            MAGIC,
29                            (oi->isResponse ? RSP_STR : (oi->isRequestByName ?
RQ_NAME_STR : RQ_NUMBER_STR)),
30                            oi->menu_no,
31                            oi->menu_name,
32                            oi->number);
33
34          return (size_t)(rv);
35  }
36
```

mcEncodingText.c (3)

```
37  /* Extract message information from given buffer.
38   * Note: modifies input buffer.
39   */
40  bool Decode(uint8_t *inBuf, const size_t mSize, OrderInfo *oi)
41  {
42
43          char *token;
44
45          token = strtok((char *) inBuf, DELIMSTR);
46          // Check for magic
47          if (token == NULL || strcmp(token, MAGIC) != 0)
48                  return false;
49
50          // Get { request by name /request by number / response } indicator
51          token = strtok(NULL, DELIMSTR);
52          if (token == NULL)
53                  return false; // Message too short
54
55          if (strcmp(token, RSP_STR) == 0) { // Response flag present
56                  oi->isResponse = true;
57                  oi->isRequestByNumber = false;
58                  oi->isRequestByName = false;
59          } else if (strcmp(token, RQ_NUMBER_STR) == 0) { // Request by number
60                  oi->isResponse = false;
61                  oi->isRequestByNumber = true;
62                  oi->isRequestByName = false;
63          } else if (strcmp(token, RQ_NAME_STR) == 0) { // Request by name
64                  oi->isResponse = false;
65                  oi->isRequestByNumber = false;
66                  oi->isRequestByName = true;
67          } else
68                  return false;
69
```

mcEncodingText.c (4)

```
70          token = strtok(NULL, DELIMSTR);
71          if (token == NULL)
72                  return false; // Message too short
73
74          oi->menu_no = atoi(token);
75
76          token = strtok(NULL, DELIMSTR);
77          if (token == NULL)
78                  return false; // Message too short
79
80          strncpy(oi->menu_name, token, MAX_MENU_NAME_LENGTH);
81
82          token = strtok(NULL, DELIMSTR);
83          if (token == NULL)
84                  return false; // Message too short
85
86          oi->number = atoi(token);
87          return true;
88  }
```

mcEncodingBin.c (1)

```
 1  /* Routines for binary encoding of order messages
 2   * Wire Format:
 3   *                               1  1  1  1  1  1  1  1  1  2  2  2  2  2  2  2  2  2  3  3
 4   *  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5  6  7  8  9  0  1
 5   * +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 6   * |                    Magic                      |  Flags |              ZERO                    |
 7   * +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 8   * |                                          menu no                                             |
 9   * +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
10   * |                                                                                              |
11   * |                                          menu name                                           |
12   * |                                                                                              |
13   * |                                       (40 characters)                                        |
14   * |                                                                                              |
15   * +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
16   * |                                           number                                             |
17   * +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
18   *
19   */
20
```

mcEncodingBin.c (2)

```
21  #include <string.h>
22  #include <stdbool.h>
23  #include <stdlib.h>
24  #include <stdint.h>
25  #include <netinet/in.h>
26  #include "Practical.h"
27  #include "mcProtocol.h"
28
29  enum {
30    PACKET_SIZE = (MAX_MENU_NAME_LENGTH + 3 * sizeof(int)),
31    SHIFT_COUNT = 16,
32    MAGIC = 0x5555,
33    RESPONSE_FLAG = 0x8000,
34    RQ_NUMBER_FLAG = 0x4000,
35    RQ_NAME_FLAG = 0x2000,
36  };
37
38  typedef struct orderMsgBin {
39    uint32_t header;
40    int32_t menu_no;
41    char menu_name[MAX_MENU_NAME_LENGTH];
42    int32_t number;
43  } OrderMsgBin;
44
```

mcEncodingBin.c (3)

```
45  /*****************************************************************************/
46  size_t Encode(OrderInfo *oi, uint8_t *outBuf, size_t bufSize) {
47    if (bufSize < PACKET_SIZE)
48      DieWithUserMessage("Output buffer too small", "");
49
50    OrderMsgBin *om = (OrderMsgBin *) outBuf;
51    memset(om, 0, sizeof(OrderMsgBin)); // Be sure
52    om->header = MAGIC << SHIFT_COUNT;
53    if (oi->isResponse)
54      om->header |= RESPONSE_FLAG;
55    if (oi->isRequestByNumber)
56      om->header |= RQ_NUMBER_FLAG;
57    if (oi->isRequestByName)
58      om->header |= RQ_NAME_FLAG;
59    om->header = htonl(om->header); // Byte order
60
61    om->menu_no = htonl(oi->menu_no);
62    strncpy(om->menu_name, oi->menu_name, MAX_MENU_NAME_LENGTH);
63    om->number = htonl(oi->number);
64
65    return PACKET_SIZE;
66  }
67
```

mcEncodingBin.c (4)

```
68  /*****************************************************************************/
69  bool Decode(uint8_t *inBuf, size_t mSize, OrderInfo *oi)
70  {
71
72    OrderMsgBin *om = (OrderMsgBin *) inBuf;
73
74    uint32_t header = ntohl(om->header);
75
76    if ((mSize < PACKET_SIZE) || ((header >> SHIFT_COUNT) != MAGIC))
77      return false;
78    /* message is big enough and includes correct magic number */
79
80    oi->isResponse = ((header & RESPONSE_FLAG) != 0);
81    oi->isRequestByNumber = ((header & RQ_NUMBER_FLAG) != 0);
82    oi->isRequestByName = ((header & RQ_NAME_FLAG) != 0);
83
84    oi->menu_no = ntohl(om->menu_no);
85    strncpy(oi->menu_name, om->menu_name, MAX_MENU_NAME_LENGTH);
86    oi->number = ntohl(om->number);
87
88    return true;
89  }
```

mcTcpServer.c (1)

```
 1  #include <stdio.h>
 2  #include <string.h>
 3  #include <stdbool.h>
 4  #include <stdint.h>
 5  #include <unistd.h>
 6  #include <fcntl.h>
 7  #include <errno.h>
 8  #include <sys/socket.h>
 9  #include <sys/types.h>
10  #include <sys/stat.h>
11  #include <arpa/inet.h>
12  #include "Practical.h"
13  #include "mcProtocol.h"
14  #include "mcEncoding.h"
15  #include "Framer.h"
16
```

mcTcpServer.c (2)

```
17  int main(int argc, char *argv[])
18  {
19          if (argc != 2) // Test for correct number of arguments
20                  DieWithUserMessage("Parameter(s)", "<Server Port/Service>");
21
22          char menu_names[MAX_MENU_NUMBER][MAX_MENU_NAME_LENGTH] = {0};
23          int no_of_items;
24          FILE *outf = fopen(RECORD_FILE_NAME, "a");
25
26          InitMenuTable(menu_names, &no_of_items);
27          PrintMenuTable(menu_names, no_of_items);
28
29          int servSock = SetupTCPServerSocket(argv[1]);
30                  // servSock is now ready to use to accept connections
31
```

mcTcpServer.c (3)

```
32          for (;;) { // Run forever
33
34                  // Wait for a client to connect
35                  int clntSock = AcceptTCPConnection(servSock);
36
37                  // Receive messages until connection closes
38                  int mSize;
39                  uint8_t inBuf[MAX_WIRE_SIZE] = {0};
40                  uint8_t outBuf[MAX_WIRE_SIZE];
41                  OrderInfo oi, result;
42
43                  while ((mSize = GetNextMsg(clntSock, inBuf, MAX_WIRE_SIZE)) > 0) {
44                          memset(&oi, 0, sizeof(oi));             // Clear order information
45                          memset(&result, 0, sizeof(oi));         // Clear order information
46                          printf("Received message (%d bytes)\n", mSize);
47
48                          if (Decode(inBuf, mSize, &oi)) {        // Parse to get OrderInfo
49                                  if (!oi.isResponse) {           // Ignore non-requests
50                                          result.isResponse = true;
51                                          result.isRequestByNumber = false;
52                                          result.isRequestByName = false;
53                                          result.number = oi.number;
54
```

mcTcpServer.c (4)

```
    55
    56                                                  if ((oi.menu_no > 0) && (oi.menu_no
< no_of_items))
    57                                                          result.menu_no = oi.menu_no;
    58                                                  else
    59                                                          result.menu_no = 0;     // Invalid No
    60
    61                                                  strncpy(result.menu_name, menu_names[oi.menu_no],
MAX_MENU_NAME_LENGTH);
    62                                          }
    63                                          else if (oi.isRequestByName) {
    64                                                  result.menu_no = SearchByName(menu_names,
oi.menu_name);
    65                                                  strncpy(result.menu_name,
menu_names[result.menu_no], MAX_MENU_NAME_LENGTH);
    66                                          }
    67                                          else {
    68                                                  fputs("Parse error, closing connection.\n",
stderr);
    69                                                  break;
    70                                          }
    71
    72                                          RecordOrderInfo(outf, &result);
    73                                  }
    74
    75
```

mcTcpServer.c (5)

```
75                                  memset(inBuf, 0, MAX_WIRE_SIZE);
76                                  memset(outBuf, 0, MAX_WIRE_SIZE);
77                                  mSize = Encode(&result, outBuf, MAX_WIRE_SIZE);
78
79                                  if (PutMsg(outBuf, mSize, clntSock) < 0) {
80                                          fputs("Error framing/outputting message\n", stderr);
81                                          break;
82                                  }
83                          }
84                          else {
85                                  fputs("Parse error, closing connection.\n", stderr);
86                                  break;
87                          }
88
89                  }
```

mcTcpClient1.c (1)

```
 1  #include <stdio.h>
 2  #include <string.h>
 3  #include <stdlib.h>
 4  #include <stdint.h>
 5  #include <unistd.h>
 6  #include <errno.h>
 7  #include <sys/socket.h>
 8  #include <netinet/in.h>
 9  #include <arpa/inet.h>
10  #include <netdb.h>
11  #include "Practical.h"
12  #include "mcProtocol.h"
13  #include "Framer.h"
14  #include "mcEncoding.h"
15
16  int main(int argc, char *argv[])
17  {
18          if (argc != 6)  // Test for correct # of args
19                  DieWithUserMessage("Parameter(s)", "<Server Address/Name>
<Server Port/Service> { <-n> <Menu Name> | <-i> <Menu Number> } number ");
20
```

mcTcpClient1.c (2)

```
21          OrderInfo oi, result;
22          memset(&oi, 0, sizeof(oi));
23          oi.isResponse = false;
24
25          char *server = argv[1];     // First arg: server address/name
26          char *service = argv[2];    // Second arg: server port/service
27
28          if (strncmp(argv[3], "-n", 2) == 0) {
29                  strncpy(oi.menu_name, argv[4], MAX_MENU_NAME_LENGTH);
30                  if (strlen(oi.menu_name) >= MAX_MENU_NAME_LENGTH)
31                          oi.menu_name[MAX_MENU_NAME_LENGTH - 1] = '\0';
32                  oi.isRequestByName = true;
33                  oi.isRequestByNumber = false;
34                  oi.menu_no = 0;
35          }
36          else if (strncmp(argv[3], "-i", 2) == 0) {
37                  oi.menu_no = atoi(argv[4]);
38                  if (oi.menu_no < 1 || oi.menu_no > MAX_MENU_NUMBER)
39                          DieWithUserMessage("Value(s)", "Menu number should be 1 ~ MAX_MENU_NUMBER
");
40                  oi.isRequestByNumber = true;
41                  oi.isRequestByName = false;
42                  strncpy(oi.menu_name, "Invalid", MAX_MENU_NAME_LENGTH);
43          }
```

mcTcpClient1.c (3)

```
    44          else
    45                  DieWithUserMessage("Parameter(s)", "<Server Address/Name> <Server
Port/Service> { <-n> <Menu Name> | <-i> <Menu Number> } number ");
    46
    47          oi.number = atoi(argv[5]);
    48
    49          // Create a connected TCP socket
    50          int sock = SetupTCPClientSocket(server, service);
    51          if (sock < 0)
    52                  DieWithUserMessage("SetupTCPClientSocket() failed", "unable to
connect");
    53
    54          // Encode for transmission
    55          uint8_t outbuf[MAX_WIRE_SIZE] = {0};    // by kgu
    56          size_t reqSize = Encode(&oi, outbuf, MAX_WIRE_SIZE);
    57
    58          // Frame and send
    59          if (PutMsg(outbuf, reqSize, sock) < 0)
    60                  DieWithSystemMessage("PutMsg() failed");
    61
```

mcTcpClient1.c (4)

```
62          // Receive and print response
63          uint8_t inbuf[MAX_WIRE_SIZE] = {0};      // by kgu
64          memset(&result, 0, sizeof(result));
65          size_t respSize = GetNextMsg(sock, inbuf, MAX_WIRE_SIZE); // Get the message
66
67          if (Decode(inbuf, respSize, &result)) { // Parse it
68                  PrintOrderInfo(&result);
69          }
70
71          // Close up
72          close(sock);
73          exit(0);
74  }
```

mcTcpClient2.c (1)

```
 1  #include <stdio.h>
 2  #include <string.h>
 3  #include <stdlib.h>
 4  #include <stdint.h>
 5  #include <unistd.h>
 6  #include <errno.h>
 7  #include <sys/socket.h>
 8  #include <netinet/in.h>
 9  #include <arpa/inet.h>
10  #include <netdb.h>
11  #include "Practical.h"
12  #include "mcProtocol.h"
13  #include "Framer.h"
14  #include "mcEncoding.h"
15
16  /*****************************************************************************/
17  int main(int argc, char *argv[])
18  {
19
20          if (argc != 3)  // Test for correct # of args
21                  DieWithUserMessage("Parameter(s)", "<Server Address/Name> <Server Port/Service>");
22
```

mcTcpClient2.c (2)

```
23          char *server = argv[1];     // First arg: server address/name
24          char *service = argv[2];    // Second arg: server port/service
25
26          char menu_names[MAX_MENU_NUMBER][MAX_MENU_NAME_LENGTH] = {0};
27          int no_of_items;
28
29          InitMenuTable(menu_names, &no_of_items);
30
31          // Create a connected TCP socket
32          int sock = SetupTCPClientSocket(server, service);
33          if (sock < 0)
34                  DieWithUserMessage("SetupTCPClientSocket() failed", "unable to connect");
35
36          OrderInfo oi, result;
37          uint8_t outbuf[MAX_WIRE_SIZE];
38          uint8_t inbuf[MAX_WIRE_SIZE];
39
40          bool end_of_program = false;
41
```

mcTcpClient2.c (3)

```
42          while (!end_of_program) {
43                  memset(&oi, 0, sizeof(oi));
44
45                  oi.isRequestByNumber = true;
46                  oi.isRequestByName = false;
47                  oi.isResponse = false;
48                  strncpy(oi.menu_name, menu_names[0], MAX_MENU_NAME_LENGTH);     // invalid name
49
50                  memset(outbuf, 0, MAX_WIRE_SIZE);
51
52                  PrintMenuTable(menu_names, no_of_items);
53                  printf("Enter menu # and number of items : ");
54                  if (scanf("%d %d", &oi.menu_no, &oi.number) != 2) {
55                          end_of_program = 1;
56                  }
57
58                  if (oi.menu_no <= 0 || oi.menu_no >= no_of_items) {
59                          printf("Invalid menu no !\n");
60                          continue;
61                  }
62
63                  if (oi.number <= 0) {
64                          printf("Invalid number !\n");
65                          continue;
66                  }
67
```

mcTcpClient2.c (4)

```
68                  // Encode for transmission
69                  size_t reqSize = Encode(&oi, outbuf, MAX_WIRE_SIZE);
70
71                  // Frame and send
72                  if (PutMsg(outbuf, reqSize, sock) < 0)
73                          DieWithSystemMessage("PutMsg() failed");
74
75                  // Receive and print response
76                  memset(&result, 0, sizeof(result));
77                  memset(inbuf, 0, MAX_WIRE_SIZE);
78
79                  size_t respSize = GetNextMsg(sock, inbuf, MAX_WIRE_SIZE); // Get the message
80
81                  if (Decode(inbuf, respSize, &result)) { // Parse it
82                          PrintOrderInfo(&result);
83                  }
84          }
85
86          // Close up
87          close(sock);
88          exit(0);
89  }
```

메뉴 이름 파일 (mcMenuName.dat)

```
Invalid
BigMac
Hamburger
Cheeseburger
Chickenburger
Bulgogiburger
Coke
Sprite
Icecream
```

서버 결과 파일(mcOrderList.dat)

```
Fri Jun  8 15:16:32 2012
         Menu No :  2                                  Hamburger 2
Fri Jun  8 15:16:34 2012
         Menu No :  3                               Cheeseburger 3
Fri Jun  8 15:16:35 2012
         Menu No :  4                              Chickenburger 4
Fri Jun  8 15:16:36 2012
         Menu No :  5                              Bulgogiburger 5
Fri Jun  8 15:32:18 2012
         Menu No :  3                               Cheeseburger 4
Fri Jun  8 15:32:20 2012
         Menu No :  2                                  Hamburger 3
Fri Jun  8 15:32:26 2012
         Menu No :  7                                    Sprite 2
Fri Jun  8 15:32:28 2012
         Menu No :  3                               Cheeseburger 3
Fri Jun  8 15:32:29 2012
         Menu No :  1                                    BigMac 1
Fri Jun  8 16:39:21 2012
         Menu No :  2                                  Hamburger 2
```