Functions


```
/***************************************************************************
 connector.c  -  description
 -------------------
 begin                : Mon Mar 12 2001
 copyright            : (C) 2001 by Zvika Brakerski, Asaf Koren
 email                : zvika@eng.tau.ac.il
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/


 #include "connector.h"
 #include "handlers.h"
 #include "contacts.h"

 /*
  connector_mod_gen_arg

  Generates argument for the function activated by the thread
  create call.
  */
 void *connector_mod_gen_arg(int accepted_sock, struct sockaddr_in *remote_addr, int
sin_size)
 {
                connector_arg *arg = (connector_arg *) malloc(sizeof(connector_arg));

                if (!arg)
                {
                        close(accepted_sock);
                        return NULL;
                }

                arg->sock = accepted_sock;
                arg->remote_addr = *remote_addr;
                arg->sin_size = sin_size;

                return ((void *) arg);
 }

 /*
  connector_mod

  Calls the connector function with the correct parameters.
  */
 void *connector_mod(void *inp)
 {
                connector_arg *arg = (connector_arg *) inp;
                int s;
                struct sockaddr_in remote_addr;
                int sin_size;

                if (!arg) return NULL;

                s = arg->sock;
                remote_addr = arg->remote_addr;
                sin_size = arg->sin_size;

                free(inp);

                return ((void *) connector(s, &remote_addr, sin_size));
 }


 /*
  connector

  Main function for the connector application.
  */
```

Functions


```
75   int connector(int s, struct sockaddr_in *remote_addr, int sin_size)
76   {
77                   link_env env; // The link's environment
78                   fd_set readfds, exceptfds;
79                   struct timeval tv;
80                   time_t last_keepalive_time; // Time last keepalive sent
81
82                   announce_connection(remote_addr);
83
84                   env.s_client = s;
85                   env.b_active = 0; // We are not yet connected to the network
86                   //env.b_writing = 0;
87                   env.b_halt = 0;
88                   env.data = NULL;
89                   init_contacts(&env);
90
91                   last_keepalive_time = time(NULL);
92
93                   while (!env.b_halt)
94                   {
95
96                           tv.tv_sec = SELECT_WAIT_SEC;
97                           tv.tv_usec = SELECT_WAIT_USEC;
98
99                           if (env.b_active)
100                          {
101                                  if ((time(NULL)-last_keepalive_time) >= TIME_BETWEEN_
     KEEPALIVES_SEC)
102                                  {
103                                          icq_KeepAlive(&env.link);
104                                          last_keepalive_time=time(NULL);
105                                  }
106                                  icq_Main(&env.link);
107                          }
108
109  #ifdef DEBUG2
110                          fprintf(stderr, "Before selecting\n");
111  #endif
112
113
114                  FD_ZERO(&readfds);
115                    FD_SET(env.s_client, &readfds);
116                  FD_ZERO(&exceptfds);
117                    FD_SET(env.s_client, &exceptfds);
118                    select(env.s_client+1, &readfds, NULL, &exceptfds, &tv);
119
120  #ifdef DEBUG2
121                          fprintf(stderr, "Before isset\n");
122  #endif
123
124                  if(FD_ISSET(env.s_client, &exceptfds))
125                          env.b_halt = 1;
126                  else
127                          if(FD_ISSET(env.s_client, &readfds))
128                          {
129                          if ((env.data = receive_message(env.s_client)))
130                                  handle_message(&env);
131                          else
132                                  env.b_halt = 1;
133                          }
134                  }
135
136                  clean_env(&env);
137
138                  return 0;
139   }
140
```

```
#ifndef __CONNECTOR_H__
#define __CONNECTOR_H__


/***************************************************************************
 connector.h  -  description
 -------------------
 begin            : Mon Mar 12 2001
 copyright        : (C) 2001 by Zvika Brakerski, Asaf Koren
 email            : zvika@eng.tau.ac.il
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/


 /*
  This file includes prototypes and definitions of the connector algorithm.
  */

 #include <icq.h>
 #include "protocol.h"
 #include "services.h"
 #include "contactlist.h"

 // Time to wait on select

 #define SELECT_WAIT_SEC        5
 #define SELECT_WAIT_USEC       0


 // Time to wait between keepalive messages

 #define TIME_BETWEEN_KEEPALIVES_SEC    50

 // Prototypes for functions of the connector application
 void *connector_mod_gen_arg(int accepted_sock, struct sockaddr_in *remote_addr, int
sin_size);
 void *connector_mod(void *inp);
 int connector(int s, struct sockaddr_in *remote_addr, int sin_size);

 /*
  Following is a struct which wraps all arguments for the connector
  application.
  */

 typedef struct {
                int sock;
                struct sockaddr_in remote_addr;
                int sin_size;
 } connector_arg;


 /*
  Following is a struct that is used as a link's "environment".
  It contains all things that the link needs to know about in the handler functions.
  */

 typedef struct {
                ICQLINK link;           // The link itself
                int v_contacts;         // Is the contact list valid?
                clist contacts;         // User's contact list
                int b_active;                   // Is the link active?
                char *data;                             // Additional required data
                int s_client;                   // The socket that is connected to th
e client
                int b_halt;                             // Do we want to env link act
ivity?
        } link_env;
```

```
73
74
75      #endif   //#ifndef __CONNECTOR_H__
76
```

Functions	
cl_aux_compare_entry	1:74
cl_aux_compose_entry	1:31
cl_aux_decompose_entry	1:55

```
/***************************************************************************
 contactlist.c  -  description
 -------------------
 begin                : Sun Apr 8 2001
 copyright            : (C) 2001 by Zvika Brakerski, Asaf Koren
 email                : zvika@eng.tau.ac.il
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

 /*
  This file implements the contact list functionality.
  */

 #include "contactlist.h"

 /*
  cl_aux_compose_entry

  This function composes a contact list entry from the given dtails.
  It handles all memory allocations.
  */

centry *cl_aux_compose_entry(char *cname, unsigned long cuin)
{
                centry *new_entry = (centry *) malloc(sizeof(centry));

                if (!new_entry) return NULL;

                if (!(new_entry->name = (char *) malloc(strlen(cname)+1)))
                {
                        free(new_entry);
                        return NULL;
                }

                strcpy(new_entry->name, cname);
                new_entry->uin = cuin;

                return new_entry;
}

 /*
  cl_aux_decompose_entry

  This function decomposes a contact list entry, freeing all allocated memory.
  */

void cl_aux_decompose_entry(centry *ce)
{
                if (ce)
                {
                        if (ce->name) free(ce->name);
                        free(ce);
                }

                return;
}

 /*
  cl_aux_compare_entry

  This function compares a contact list entry with a given name and uin.
  If the requested name is NULL it matches everything.
  If the requested uin is 0, it matches everything.
  */

int cl_aux_compare_entry(centry *ce, char *cname, unsigned long cuin)
{
```

Functions


```
76                  if (cname)
77                          if (strcmp(ce->name, cname))
78                                  return 0; // No match
79
80                  if (cuin)
81                          if (ce->uin != cuin)
82                                  return 0; // No match
83
84                  return 1; // Match
85  }
86
87
88  /*
89   cl_add
90
91   This function adds a contact to the contact list.
92   Note that the name string is copied (and therefore it should not
93   be allocated beforehand).
94   The function returns a pointer to the new contact.
95   */
96
97  centry *cl_add(clist *list, char *cname, unsigned long cuin)
98  {
99                  centry *new_entry = cl_aux_compose_entry(cname, cuin);
100
101                 if (!new_entry) return NULL;
102
103                 new_entry->next = list->first;
104                 list->first = new_entry;
105
106                 return new_entry;
107 }
108
109
110 /*
111  cl_find
112
113  Finds an entry in the list.
114  */
115
116 centry *cl_find(clist *list, char *cname, unsigned long cuin)
117 {
118                 centry *current;
119
120                 for(current=list->first; current; current = current->next)
121                         if (cl_aux_compare_entry(current, cname, cuin))
122                                 return current;
123
124                 return NULL;
125 }
126
127
128
129 /*
130  cl_delete
131
132  Searches for a certain contact and deletes its entry from the list.
133  An entry can be indicated by name, uin or both.
134  To specify a search by uin only, set the name field to NULL.
135  To specify a search by name only, set the uin field to 0.
136  */
137 centry *cl_delete(clist *list, char *cname, unsigned long cuin)
138 {
139                 centry *previous, *current;
140
141                 if (!(list->first)) return NULL;
142
143                 if (cl_aux_compare_entry(list->first, cname, cuin))
144                 {
145                         current = list->first;
146                         list->first = list->first->next;
147                         cl_aux_decompose_entry(current);
148                         return list->first;
149                 }
150
```

Functions


```
151                     /* Going over the whole list */
152
153                     for(previous=list->first, current=previous->next; curr
    = current, current=current->next)
154                             if (cl_aux_compare_entry(current, cname, cuin)
155                             {
156                                     previous->next = current->next;
157                                     cl_aux_decompose_entry(current);
158                                     return previous->next;
159                             }
160
161                     return NULL;
162     }
163
164
165     /*
166      cl_read
167
168      Reads a complete contact list from a file.
169      The file is opened beforehand but it is NOT locked (the lock is
170      implemented in this function).
171      The file format is very simple, each entry is represented
172      by two lines: the first contains the uin and the second contains
173      the name.
174      The name must not exceed MAX_CNAME_SIZE characters.
175      */
176
177     void cl_read(clist *list, FILE *in_file)
178     {
179             char temp[MAX_CNAME_SIZE+2];
180             char rname[MAX_CNAME_SIZE+2];
181             unsigned long ruin;
182
183             list->first = NULL;
184
185             flock(fileno(in_file), LOCK_SH); // Locking the file for reading
186
187             while (fgets(temp, MAX_CNAME_SIZE+2, in_file))
188             {
189                             sscanf(temp, "%lu", &ruin);
190                             fgets(rname, MAX_CNAME_SIZE+1, in_file);
191                             rname[strlen(rname)-1] = '\0'; // Remove newline from name
192                             cl_add(list, rname, ruin);
193             }
194
195             flock(fileno(in_file), LOCK_UN);
196
197             return;
198
199     }
200
201
202     /*
203      cl_write
204
205      This function writes the contact list to a given file.
206      This function handles the file locking.
207      */
208
209     void cl_write(clist *list, FILE *out_file)
210     {
211             centry *current;
212
213             flock(fileno(out_file), LOCK_EX);
214
215             for(current=list->first; current; current = current->next)
216             {
217                     fprintf(out_file, "%lu\n", current->uin);
218                     fprintf(out_file, "%s\n", current->name);
219             }
220
221             flock(fileno(out_file), LOCK_UN);
222
223             return;
224
```

```
225     }
226
227     /*
228      cl_init
229
230      Initializes a contact list.
231      */
232
233     void cl_init(clist *list)
234     {
235             list->first = NULL;
236             return;
237     }
238
239     /*
240      cl_clean
241
242      Cleans all entries in a contact list.
243      */
244
245     void cl_clean(clist *list)
246     {
247             centry *current, *suc;
248
249             current=list->first;
250
251             while (current)
252             {
253                     suc = current->next;
254                     cl_aux_decompose_entry(current);
255                     current = suc;
256             }
257
258             list->first = NULL;
259
260             return;
261
262     }
263
264
```

Functions	
cl_add	2:97
cl_aux_compare_entry	1:74
cl_aux_compose_entry	1:31
cl_aux_decompose_entry	1:55
cl_clean	4:245
cl_delete	2:137
cl_find	2:116
cl_init	4:233
cl_read	3:177
cl_write	3:209

```
#ifndef __CONTACTLIST_H__
#define __CONTACTLIST_H__

/***************************************************************************
 contactlist.h  -  description
 -------------------
 begin                : Sun Apr 8 2001
 copyright            : (C) 2001 by Zvika Brakerski, Asaf Koren
 email                : zvika@eng.tau.ac.il
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/


/*
 Header file for the contact list implementation.

 Note that icqlib implements a contact list mechanism that does not
 support saving the name of the contact (only the uin) and therefore
 we require an external mechanism for that purpose.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/file.h>

// Maximal size for a contact name.
#define MAX_CNAME_SIZE 256

typedef struct contact_entry {  // An entry in the contact list
        struct contact_entry *next;
        char *name;
        unsigned long uin;
} centry;

typedef struct {  // The contact list itself
        centry *first;
} clist;

// Prototypes for other modules

centry *cl_add(clist *list, char *cname, unsigned long cuin);
centry *cl_find(clist *list, char *cname, unsigned long cuin);
centry *cl_delete(clist *list, char *cname, unsigned long cuin);
void cl_read(clist *list, FILE *in_file);
void cl_write(clist *list, FILE *out_file);
void cl_init(clist *list);
void cl_clean(clist *list);

#endif   //#ifndef __CONTACTLIST_H__
```

Functions	
push_contact_list	1:31

```
/***************************************************************************
 contacts.c  -  description
 -------------------
 begin                : Thu Apr 12 2001
 copyright            : (C) 2001 by Zvika Brakerski, Asaf Koren
 email                : zvika@eng.tau.ac.il
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

#include "contacts.h"

 /*
  This file contains functions that handle the contacts section of the
  environment.
  */

 /*
  push_contact_list

  Pushes contact list to client.
  */

 void push_contact_list(link_env *penv)
 {
                char message[4+4+4+4+MAX_CNAME_SIZE+1];
                centry *current;

#ifdef DEBUG
                fprintf(stderr, "%lu: Pushing contacts.\n", penv->link.icq_Uin);
#endif

#ifdef DO_LOG
                fprintf(LOG_FILE, "%lu\tPushing contacts:\n", penv->link.icq_Uin);
#endif


                for (current=penv->contacts.first; current; current = current->next)
                {

#ifdef DEBUG
                fprintf(stderr, "%lu: Pushing contact (%lu, %s) to user\n", penv->lin
k.icq_Uin, current->uin, current->name);
#endif

#ifdef DO_LOG
                fprintf(LOG_FILE, "%lu\tPushing contact (%lu, %s)\n", penv->link.icq_
Uin, current->uin, current->name);
#endif
                                (*(unsigned long *) message) = htonl(4+4+4+4+strlen(c
urrent->name)+1);
                                (*(unsigned long *) (message+4)) = htonl(STOC(PSH_CNT
CT));
                                (*(unsigned long *) (message+8)) = htonl(current->uin
);
                                (*(unsigned long *) (message+12)) = htonl(strlen(curr
ent->name)+1);
                                strcpy(message+16, current->name);

                                if (send_message(penv->s_client, message, 4+4+4+4+str
len(current->name)+1) == -1)
                                {
                                        penv->b_halt = 1;
                                        return;
                                }

                                icq_ContactAdd(&penv->link, current->uin);
                }
```

Functions	
another_contact	2:142
init_contacts	2:81
push_contact_list	1:31
start_contact_list	2:94
term_contact_list	2:124

```
69
70                      return;
71      }
72
73
74
75      /*
76       init_contacts
77
78       Initializes contacts
79       */
80
81      void init_contacts(link_env *penv)
82      {
83                      cl_init(&penv->contacts);
84                      penv->v_contacts = 0;
85                      return;
86      }
87
88      /*
89       start_contact_list
90
91       Brings contact list from database
92       */
93
94      void start_contact_list(link_env *penv)
95      {
96                       if (penv->v_contacts)
97                                       return;
98
99      #ifdef DEBUG
100                     fprintf(stderr, "Going to get contact list.\n");
101     #endif
102
103                      get_contact_list(penv->link.icq_Uin, &penv->contacts);
104
105     #ifdef DEBUG
106                     fprintf(stderr, "Finished getting contact list.\n");
107     #endif
108
109
110     #ifdef DEBUG
111                     fprintf(stderr, "%lu: Got contact list from file.\n", penv->link.icq_
        Uin);
112     #endif
113                      penv->v_contacts = 1;
114                      push_contact_list(penv);
115                      return;
116      }
117
118      /*
119       term_contact_list
120
121       Writes contact list to database and clears it
122       */
123
124      void term_contact_list(link_env *penv)
125      {
126                      if (!(penv->v_contacts))
127                                      return;
128
129                      put_contact_list(penv->link.icq_Uin, &penv->contacts);
130                      cl_clean(&penv->contacts);
131                      cl_init(&penv->contacts);
132                      penv->v_contacts = 0;
133                      return;
134      }
135
136      /*
137       another_contact
138
139       Adds a contact to contact list.
140       */
141
142      void another_contact(link_env *penv, char *cname, unsigned long cuin)
```

Functions


```
143     {
144             centry *item;
145
146             if (!(penv->v_contacts))
147                     return;
148
149             item = cl_find(&penv->contacts, NULL, cuin);
150
151             if (item) // If the uin is already there, we just change the name
152             {
153                     char *temp = (char *) malloc(strlen(cname)+1);
154
155                     if (!temp)
156                             return;
157
158                     if (item->name)
159                             free(item->name);
160
161                     item->name = temp;
162                     strcpy(item->name, cname);
163             }
164             else
165                     cl_add(&penv->contacts, cname, cuin);
166
167             return;
168     }
169
170     /*
171      remove_contact
172
173      Removes a contact from the contact list.
174      */
175
176     void remove_contact(link_env *penv, unsigned long cuin)
177     {
178             if (!(penv->v_contacts))
179                     return;
180
181             cl_delete(&penv->contacts, NULL, cuin);
182
183             return;
184     }
185
186
187
```

```
#ifndef __CONTACTS_H__
#define __CONTACTS_H__

/***************************************************************************
 contacts.h  -  description
 -------------------
 begin                : Thu Apr 12 2001
 copyright            : (C) 2001 by Zvika Brakerski, Asaf Koren
 email                : zvika@eng.tau.ac.il
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

  /*
   This file contains prototypes and definitions for handling the
   contacts part of the environment.
   */

#include "db.h"
#include "connector.h"


// Prototypes required by other modules
void init_contacts(link_env *penv);
void start_contact_list(link_env *penv);
void term_contact_list(link_env *penv);
void another_contact(link_env *penv, char *cname, unsigned long cuin);
void remove_contact(link_env *penv, unsigned long cuin);


#endif   //#ifndef __CONTACTS_H__

```

Functions


```
/***************************************************************************
 db.c  -  description
 -------------------
 begin                : Wed Apr 11 2001
 copyright            : (C) 2001 by Zvika Brakerski, Asaf Koren
 email                : zvika@eng.tau.ac.il
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

#include "db.h"

 /*
  This file implements all database features.
  */

 /*
  compose_db_path

  Assigns the path of the database into the given string.
  */

 void compose_db_path(char *result, const char *file)
 {
  strcpy(result, HOME_DIR);
        strcat(result, DB_PATH);

        if (file)
                strcat(result, file);

        return;
 }


 /*
  initialize_db

  Makes sure database subdirectory exists and creates it otherwise.
  Returns zero if successful, otherwise returns the value of errno.
  */

 int initialize_db()
 {
        int e_val=0; // error value
        char db_name[DB_PATH_LEN+1]; // The name of the directory

        compose_db_path(db_name, NULL);

        if (mkdir(db_name, DB_PERMISSIONS) == -1)
        {
                e_val = errno;

                if (e_val == EEXIST)
                        e_val = 0;
        }

        return e_val;

 }


 /*
  get_contact_list

  Takes the contact list from the database and puts it into
  the given contact list.
  */
```

Functions	
compose_db_path	1:30
get_contact_list	2:76
initialize_db	1:49
put_contact_list	2:128

```
76   void get_contact_list(unsigned long uin, clist *list)
77   {
78                   char file[UIN_LEN+DB_EXT_LEN+1]; // The name of the file
79                   char db_entry_path[DB_PATH_LEN+UIN_LEN+DB_EXT_LEN+1];
     hat contains the path of the database entry
80                   FILE *db_entry; // The database entry of the given uin
81
82                   cl_init(list);
83
84                   sprintf(file, "%lu%s", uin, DB_EXT);
85
86   #ifdef DEBUG
87                   fprintf(stderr, "file: %s\n", file);
88   #endif
89
90                   compose_db_path(db_entry_path, file);
91
92   #ifdef DEBUG
93                   fprintf(stderr, "db_entry_path: %s\n", db_entry_path);
94   #endif
95
96                   db_entry = fopen(db_entry_path, "r");
97
98   #ifdef DEBUG
99                   fprintf(stderr, "Opened database entry. Null pointer? %c\n", (db_entr
     y)?'n':'y');
100  #endif
101
102                  if (!db_entry)
103                  {
104  #ifdef DEBUG
105                          fprintf(stderr, "Database entry is null.\n");
106  #endif
107                          return;
108                  }
109
110  #ifdef DEBUG
111                          fprintf(stderr, "Database entry is not null.\n");
112  #endif
113
114                  cl_read(list, db_entry);
115
116                  fclose(db_entry);
117
118                  return;
119  }
120
121
122   /*
123    put_contact_list
124
125    Writes contact list to database.
126    */
127
128  void put_contact_list(unsigned long uin, clist *list)
129  {
130                  char file[UIN_LEN+DB_EXT_LEN+1]; // The name of the file
131                  char db_entry_path[DB_PATH_LEN+UIN_LEN+DB_EXT_LEN+1]; // The string t
     hat contains the path of the database entry
132                  FILE *db_entry; // The database entry of the given uin
133
134                  sprintf(file, "%lu%s", uin, DB_EXT);
135                  compose_db_path(db_entry_path, file);
136
137                  db_entry = fopen(db_entry_path, "w");
138                  if (!db_entry)
139                          return;
140
141                  cl_write(list, db_entry);
142
143                  fclose(db_entry);
144
145                  return;
146  }
147
```

Functions	
compose_db_path	1:30
get_contact_list	2:76
initialize_db	1:49
put_contact_list	2:128

```
#ifndef __DB_H__
#define __DB_H__

/***************************************************************************
 db.h  -  description
 -------------------
 begin                : Tue Apr 10 2001
 copyright            : (C) 2001 by Zvika Brakerski, Asaf Koren
 email                : zvika@eng.tau.ac.il
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/


  /*
   This file contains prototypes and definitions for the database related issues.
   */

#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <stdio.h>
#include <string.h>
#include "contactlist.h"

// The subdirectory in which the database resides
#define DB_PATH                                 ".csicq_db/"
// The permissions that the subdirectory should get
#define DB_PERMISSIONS  0700

// The length of the database path
#define DB_PATH_LEN                     (strlen(HOME_DIR)+strlen(DB_PATH))
// An upper bound on the size of a uin
#define UIN_LEN                                 20

// The extension of files in the database
#define DB_EXT                                  ".cl"
// The length of the extension of files in the database
#define DB_EXT_LEN                      3


// Prototypes for functions used by other modules
int initialize_db();
void get_contact_list(unsigned long uin, clist *list);
void put_contact_list(unsigned long uin, clist *list);



#endif   //#ifndef __DB_H__

```

Functions	
icq_handle_logged_in	1:35

```
/***************************************************************************
 handlers.c  -  description
 -------------------
 begin                : Tue Mar 13 2001
 copyright            : (C) 2001 by Zvika Brakerski, Asaf Koren
 email                : zvika@eng.tau.ac.il
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

 #include "handlers.h"

 /*
  Implementations of handler functions
  */


 /*
  Following are handlers for events on the icq socket
  */

 /*
  icq_handle_logged_in

  ICQ informed us we're logged in
  */

 void icq_handle_logged_in(struct icq_link *link)
 {
                link_env *penv = (link_env *) ENV(link); // Link's environment
                char buf[8];
                unsigned long *pL;

#ifdef DEBUG
                fprintf(stderr, "%lu: Connection established!\n", link->icq_Uin);
#endif

#ifdef DO_LOG
                fprintf(LOG_FILE, "%lu\tConnected to icq network.\n", link->icq_Uin);
#endif


                pL = (unsigned long *) buf;

                *pL = htonl(8);
                pL++;
                *pL = htonl(STOC(SV(LOGIN_REQ)));

                if (send_message(penv->s_client, buf, 8) == -1)
                        penv->b_halt = 1;

#ifdef DEBUG
                fprintf(stderr, "%lu: Starting contact list...\n", link->icq_Uin);
#endif

                start_contact_list(penv);

                // Sending contact list to server
                icq_SendContactList(&penv->link);

#ifdef DEBUG
                fprintf(stderr, "%lu: Contact list started.\n", link->icq_Uin);
#endif

                return;

 }
```

Functions


```
76      /*
77       icq_handle_not_logged_in
78
79       ICQ informed us we can't login
80       */
81
82      void icq_handle_not_logged_in(struct icq_link *link)
83      {
84                      link_env *penv = (link_env *) ENV(link); // Link's environment
85                      char buf[8];
86                      unsigned long *pL;
87
88      #ifdef DEBUG
89                      fprintf(stderr, "%lu: Connection refused!\n", link->icq_Uin);
90      #endif
91
92      #ifdef DO_LOG
93                      fprintf(LOG_FILE, "%lu\tConnection to the ICQ server refused.\n", lin
        k->icq_Uin);
94      #endif
95
96
97                      pL = (unsigned long *) buf;
98
99                      *pL = htonl(8);
100                     pL++;
101                     *pL = htonl(STOC(SX(LOGIN_REQ)));
102
103                     if (send_message(penv->s_client, buf, 8) == -1)
104                             penv->b_halt = 1;
105
106                     return;
107
108     }
109
110      /*
111       icq_handle_disconnect
112
113      ICQ informed us we're disconnected
114       */
115
116     void icq_handle_disconnect(struct icq_link *link)
117     {
118                     link_env *penv = (link_env *) ENV(link); // Link's environment
119                     char buf[8];
120                     unsigned long *pL;
121
122     #ifdef DEBUG
123                     fprintf(stderr, "%lu: Disconnected from server.\n", link->icq_Uin);
124     #endif
125
126     #ifdef DO_LOG
127                     fprintf(LOG_FILE, "%lu\tDisconnected from ICQ server.\n", link->icq_U
        in);
128     #endif
129
130                     pL = (unsigned long *) buf;
131
132                     *pL = htonl(8);
133                     pL++;
134                     *pL = htonl(STOC(INF_DISCNCT));
135
136                     if (send_message(penv->s_client, buf, 8) == -1)
137                             penv->b_halt = 1;
138
139                     // Attempting to login again
140                     icq_Connect(&penv->link, ICQ_HOST, ICQ_PORT);
141                     icq_Login(&penv->link, STATUS_ONLINE);
142
143                     return;
144     }
145
146      /*
147       icq_handle_recv_message
148
```

Functions	
icq_handle_disconnect	2:116
icq_handle_logged_in	1:35
icq_handle_not_logged_i	2:82
icq_handle_recv_messa	3:152
icq_handle_update_stat	3:204

```
149     We got a message on ICQ
150     */
151
152     void icq_handle_recv_message(struct icq_link *link, unsigned long uin, u
        hour,
153
                                unsigned char minute, unsigned char day, unsigned char month,
154
                                unsigned short year, const char *msg)
155     {
156                     link_env *penv = (link_env *) ENV(link);
157                     unsigned long length;
158                     char *buf, *pC;
159
160     #ifdef DEBUG
161                     fprintf(stderr, "%lu: Received message. [%s]\n", link->icq_Uin, msg);
162     #endif
163
164     #ifdef DO_LOG
165                     fprintf(LOG_FILE, "%lu\tReceived message from %lu:\n{begin}\n%s\n{end
        }\n", link->icq_Uin, uin, msg);
166     #endif
167
168
169                     length = 4+4+4+4+strlen(msg)+1;
170                     while (!(pC = malloc(length)))
171                             sleep(1);
172
173                     buf = pC;
174
175                     *((unsigned long *) pC) = htonl(length);
176                     pC+=4;
177
178                     *((unsigned long *) pC) = htonl(STOC(RECV_MSG));
179                     pC+=4;
180
181                     *((unsigned long *) pC) = htonl(uin);
182                     pC+=4;
183
184                     *((unsigned long *) pC) = htonl(strlen(msg)+1);
185                     pC+=4;
186
187                     strcpy(pC, msg);
188
189
190                     if (send_message(penv->s_client, buf, length) == -1)
191                             penv->b_halt = 1;
192
193                     free(buf);
194
195                     return;
196     }
197
198
199     /*
200      icq_handle_update_status
201
202      Handles an update and the status of a user in the contact list
203      */
204     void icq_handle_update_status(struct icq_link *link, unsigned long uin, unsigned lon
        g status)
205     {
206                     link_env *penv = (link_env *) ENV(link);
207                     char buf[16], *pC=buf;
208
209     #ifdef DEBUG
210                     fprintf(stderr, "%lu: User %lu changed status to %lu.\n", link->icq_U
        in, uin, status);
211     #endif
212
213     #ifdef DO_LOG
214                     fprintf(LOG_FILE, "%lu\tUser %lu changed status to %lu.\n", link->icq
        _Uin, uin, status);
215     #endif
216
```

Functions


```
217                 *((unsigned long *) pC) = htonl(16);
218                 pC+=4;
219
220                 *((unsigned long *) pC) = htonl(STOC(UPDATE_USR));
221                 pC+=4;
222
223                 *((unsigned long *) pC) = htonl(uin);
224                 pC+=4;
225
226                 *((unsigned long *) pC) = htonl(status);
227
228                 if (send_message(penv->s_client, buf, 16) == -1)
229                         penv->b_halt = 1;
230
231                 return;
232  }
233
234  /*
235   icq_handle_user_online
236
237   Handles an event of user coming online
238   */
239  void icq_handle_user_online(struct icq_link *link, unsigned long uin, unsigned long
     status,
240
                             unsigned long ip, unsigned short port, unsigned long real_ip,
241
                             unsigned char tcp_flag)
242  {
243
244 #ifdef DEBUG
245                 fprintf(stderr, "%lu: User %lu came online.\n", link->icq_Uin, uin);
246 #endif
247
248 #ifdef DO_LOG
249                 fprintf(LOG_FILE, "%lu\tUser %lu came online.\n", link->icq_Uin, uin)
     ;
250 #endif
251
252                 icq_handle_update_status(link, uin, STATUS_ONLINE);
253                 return;
254  }
255
256
257  /*
258   icq_handle_user_offline
259
260   Handles an event of user going offline
261   */
262  void icq_handle_user_offline(struct icq_link *link, unsigned long uin)
263  {
264 #ifdef DEBUG
265                 fprintf(stderr, "%lu: User %lu went offline.\n", link->icq_Uin, uin);
266 #endif
267
268 #ifdef DO_LOG
269                 fprintf(LOG_FILE, "%lu\tUser %lu went offline.\n", link->icq_Uin, uin
     );
270 #endif
271
272                 icq_handle_update_status(link, uin, STATUS_OFFLINE);
273                 return;
274  }
275
276
277  /*
278   icq_handle_srv_ack
279
280   For debug reasons
281   */
282  void icq_handle_srv_ack(struct icq_link *link, unsigned short seq)
283  {
284 #ifdef DEBUG
285                 fprintf(stderr, "%lu: Got SrvAck %d.\n", link->icq_Uin, seq);
286 #endif
```

Functions	
handle_login_req	5:308
icq_handle_disconnect	2:116
icq_handle_logged_in	1:35
icq_handle_not_logged_i	2:82
icq_handle_recv_messa	3:152
icq_handle_srv_ack	4:282
icq_handle_update_stat	3:204
icq_handle_user_offline	4:262
icq_handle_user_online	4:239

```
287
288                     return;
289     }
290
291
292     /* ========================================== */
293
294
295
296   /*
297    Following are handlers for messages received from the client
298    */
299
300
301
302
303     /*
304      handle_login_req
305
306      Client asks to login
307      */
308     void handle_login_req(link_env *penv)
309     {
310                     login_req req;
311                     char *pC = penv->data;
312
313                     if (penv->b_active)
314                             return; // you need to disconnect before you try to login aga
    in
315
316                     pC += 4; // The first field is message length which isn't interesting
317
318                     req.type = ntohl(* (unsigned long *) pC);
319                     pC += 4;
320
321                     req.uin = ntohl(* (unsigned long *) pC);
322                     pC += 4;
323
324                     req.password.length = ntohl(* (unsigned long *) pC);
325                     pC += 4;
326
327                     req.password.text = (char *) pC;
328                     pC += req.password.length;
329
330                     req.nick.length = ntohl(* (unsigned long *) pC);
331                     pC += 4;
332
333                     req.nick.text = (char *) pC;
334                     pC += req.nick.length;
335
336
337   #ifdef DEBUG
338                     fprintf(stderr, "Initializing connection. UIN: %lu, Password: ***, Ni
    ck: %s\n", req.uin, req.nick.text);
339   #endif
340
341   #ifdef DO_LOG
342                     fprintf(LOG_FILE, ">>Initializing connection: UIN=%lu\tPassword=*****
    ***\tNick=%s\n", req.uin, req.nick.text);
343   #endif
344
345
346             icq_Init(&penv->link, req.uin, req.password.text, req.nick.text);
347
348             ENV((&(penv->link))) = (char *) penv; // Putting a link to the environment
349             penv->b_active = 1;
350
351               penv->link.icq_Logged = &icq_handle_logged_in;
352               penv->link.icq_Disconnected = &icq_handle_disconnect;
353               penv->link.icq_RecvMessage = &icq_handle_recv_message;
354               penv->link.icq_WrongPassword = &icq_handle_not_logged_in;
355               penv->link.icq_InvalidUIN = &icq_handle_not_logged_in;
356               penv->link.icq_UserStatusUpdate = &icq_handle_update_status;
357               penv->link.icq_UserOnline = &icq_handle_user_online;
358               penv->link.icq_UserOffline = &icq_handle_user_offline;
```

```
359             penv->link.icq_SrvAck = &icq_handle_srv_ack;
360
361
362                     icq_Connect(&penv->link, ICQ_HOST, ICQ_PORT);
363                     icq_Login(&penv->link, STATUS_ONLINE);
364
365                     return;
366     }
367
368
369     /*
370      handle_disconnect
371
372      Client wants to disconnect
373      */
374
375     void handle_disconnect(link_env *penv)
376     {
377                     if (!(penv->b_active))
378                             return; // You are already disconnected!
379
380   #ifdef DEBUG
381                     fprintf(stderr, "%lu: Terminating connection.\n", penv->link.icq_Uin)
    ;
382   #endif
383
384   #ifdef DO_LOG
385                     fprintf(LOG_FILE, "%lu\tConnection terminated.\n", penv->link.icq_Uin
    );
386   #endif
387
388
389                     penv->b_active = 0;
390                     icq_Logout(&penv->link);
391                     icq_Disconnect(&penv->link);
392                     icq_Done(&penv->link);
393
394                     close(penv->s_client); // Closing socket
395                     penv->b_halt = 1;
396
397                     return;
398     }
399
400     /*
401      handle_send_msg
402
403      The client wants to send a message
404      */
405     void handle_send_msg(link_env *penv)
406     {
407                     char *pC = penv->data;
408                     unsigned long uin;
409
410                     pC += 8; // We don't care about the length and type fields
411                     uin = ntohl(* (unsigned long *) pC);
412                     pC += 8;
413
414   #ifdef DEBUG
415                     fprintf(stderr, "%lu: Sending message to %lu [%s]\n", penv->link.icq_
    Uin, uin, pC);
416   #endif
417
418   #ifdef DO_LOG
419                     fprintf(LOG_FILE, "%lu\tSending message to %lu:\n{begin}\n%s\n{end}\n
    ", penv->link.icq_Uin, uin, pC);
420   #endif
421
422                     icq_SendMessage(&penv->link, uin, pC, ICQ_SEND_BESTWAY);
423
424                     return;
425     }
426
427     /*
428      handle_add_usr
429
```

Functions

```
 The client wants a uin to be added into the contact list
 */
void handle_add_usr(link_env *penv)
{
                char *pC = penv->data;
                unsigned long uin;


                uin = ntohl(* (unsigned long *) (pC + 8));
                pC += 16;

#ifdef DEBUG
                fprintf(stderr, "%lu: Adding user %lu to contact list.
.icq_Uin, uin);
#endif

#ifdef DO_LOG
                fprintf(LOG_FILE, "%lu\tAdding user (%lu, %s) to contact list.\n", pe
nv->link.icq_Uin, uin, pC);
#endif


                icq_ContactAdd(&penv->link, uin);
                icq_SendNewUser(&penv->link, uin);

                another_contact(penv, pC, uin);

                return;
 }


 /*
  handle_del_usr

  The client wants a uin to be added into the contact list
  */
 void handle_del_usr(link_env *penv)
 {
                char *pC = penv->data;
                unsigned long uin;

                uin = ntohl(* (unsigned long *) (pC + 8));
                pC += 16;

#ifdef DEBUG
                fprintf(stderr, "%lu: Removing user %lu from contact list.\n", penv->
link.icq_Uin, uin);
#endif

#ifdef DO_LOG
                fprintf(LOG_FILE, "%lu\tRemoving user %lu from contact list.\n", penv
->link.icq_Uin, uin);
#endif

                icq_ContactRemove(&penv->link, uin);

                remove_contact(penv, uin);

                return;
 }


 /* ========================================== */

 /*
  handle_message

  Handles a message received from the client
  */
 int handle_message(link_env *penv)
 {
                unsigned long type;

                if (!(penv->data))
```

Functions	
handle_add_usr	7:432
handle_del_usr	7:464
handle_disconnect	6:375
handle_login_req	5:308
handle_message	7:496
handle_send_msg	6:405
icq_handle_disconnect	2:116
icq_handle_logged_in	1:35
icq_handle_not_logged_i	2:82
icq_handle_recv_messa	3:152
icq_handle_srv_ack	4:282
icq_handle_update_stat	3:204
icq_handle_user_offline	4:262
icq_handle_user_online	4:239

```
                        return 1;

                type = message_type(penv->data);

#ifdef DEBUG
                fprintf(stderr, "Received message of type %lu from cli
#endif

                switch (type)
                {
                        case (CTOS(LOGIN_REQ)):
                                handle_login_req(penv);
                                break;

                        case (CTOS(SEND_MSG)):
                                handle_send_msg(penv);
                                break;

                        case (CTOS(ADD_USR)):
                                handle_add_usr(penv);
                                break;

                        case (CTOS(DEL_USR)):
                                handle_del_usr(penv);
                                break;

                        case (CTOS(DISCONNECT)):
                                handle_disconnect(penv);
                                break;
                        }

                        if (penv->data)
                        {
                                clean_buf(penv->data);
                                penv->data = NULL;
                        }

                        return 0;
}



/* ============================================ */

/*
 clean_env

 Cleans the environment when we are finish
 */

int clean_env(link_env *penv)
{
                if (penv->data)
                {
                        clean_buf(penv->data);
                        penv->data = NULL;
                }

                close(penv->s_client);

                if (penv->b_active)
                {
                        icq_Logout(&penv->link);
                        icq_Disconnect(&penv->link);
                        icq_Done(&penv->link);
                        penv->b_active = 0;
                }

                term_contact_list(penv);

                return 0;

}
```

Functions	
clean_env	8:553
handle_add_usr	7:432
handle_del_usr	7:464
handle_disconnect	6:375
handle_login_req	5:308
handle_message	7:496
handle_send_msg	6:405
icq_handle_disconnect	2:116
icq_handle_logged_in	1:35
icq_handle_not_logged_i	2:82
icq_handle_recv_messa	3:152
icq_handle_srv_ack	4:282
icq_handle_update_stat	3:204
icq_handle_user_offline	4:262
icq_handle_user_online	4:239

```
#ifndef __HANDLERS_H__
#define __HANDLERS_H__

/***************************************************************************
 handlers.h  -  description
 -------------------
 begin                : Tue Mar 13 2001
 copyright            : (C) 2001 by Zvika Brakerski, Asaf Koren
 email                : zvika@eng.tau.ac.il
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/


/*
 Header file for message handlers
 */

#include <netinet/in.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>

#include "connector.h"
#include "services.h"
#include "contacts.h"


// A macro to getting a link's environment pointer
#define ENV(pL) ((pL)->icq_ProxyHost)


// Prototype for functions used in other modules
int handle_message(link_env *penv);
int clean_env(link_env *penv);


#endif   //#ifndef __HANDLERS_H__

```

Functions	
listener	1:25

```
/***************************************************************************
 listener.c  -  description
 -------------------
 begin                : Sat Mar 10 2001
 copyright            : (C) 2001 by Zvika Brakerski, Asaf Koren
 email                : zvika@eng.tau.ac.il
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

  /*
   This file includes the functionality of the listener program.
   */

  #include "listener.h"
  #include "connector.h"

  int listener(listener_options opt)
  {
                int listening_sock;
   // Socket we'll be listening on
                int accepted_sock;
   // Socket for incomming connections
      struct sockaddr_in my_addr;                          // A struct to hold local add
resses
      struct sockaddr_in remote_addr;              // A struct to hold remote addresses
      int sin_size;
   // An integer to hold size of struct
      pthread_t thread;
// Details of created thread
      int yes=1;

      // Initializing the database
                if (initialize_db())
                {
                        perror("mkdir()");
                        return -1;
                }


      if ((listening_sock = socket(AF_INET, SOCK_STREAM, 0)) == -1)
      {
          perror("socket()");
          return -1;
      }


      if (setsockopt(listening_sock, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(int)) == -1
)
      {
          perror("setsockopt");
          exit(1);
      }


      my_addr.sin_family = AF_INET;
      my_addr.sin_port = htons(CSICQ_PORT);
      my_addr.sin_addr.s_addr = INADDR_ANY;
      memset(&(my_addr.sin_zero), '\0', 8);

      if (bind(listening_sock, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)) ==
 -1)
      {
          perror("bind()");
          return -1;
      }

      if (listen(listening_sock, BACKLOG) == -1)
```

Functions	
listener	1:25

```
69          {
70              perror("listen()");
71              close(listening_sock);
72              return -1;
73          }
74
75
76          sin_size = sizeof(struct sockaddr_in);
77          while ((accepted_sock = accept(listening_sock, &remote_addr, &sin_size)) != -1)
78          {
79              /* Spawning new thread */
80
81              pthread_create(&thread, NULL, &connector_mod, connector_mod_gen_arg(accepted_
    sock, &remote_addr, sin_size));
82
83              /* End of creating new thread */
84
85              sin_size = sizeof(struct sockaddr_in);
86          }
87
88          if (accepted_sock == -1)
89          {
90              perror("accept()");
91              return -1;
92          }
93
94          return 0;
95      }
96
```

```
#ifndef __LISTENER_H__
#define __LISTENER_H__

/***************************************************************************
 listener.h  -  description
 -------------------
 begin                : Sat Mar 10 2001
 copyright            : (C) 2001 by Zvika Brakerski, Asaf Koren
 email                : zvika@eng.tau.ac.il
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

  /*
    This is the header file for the listener program. This program listens on
    the CSICQ_PORT and awaits for incoming connections. For each such connection
    it forks out a process to take care of it.
    */


#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <pthread.h>
#include "db.h"

#define BACKLOG 10

// The port csicq servers listen on
#define CSICQ_PORT 4040

// An enumeration of options for the listener (currently empty)
typedef enum {
        NONE
} listener_options;

// Prototypes for functions used in other modules
int listener(listener_options opt);


#endif   //#ifndef __LISTENER_H__

```

Functions	
main	1:27

```
/***************************************************************************
 main.c  -  description
 -------------------
 begin                : Sat Mar 10 17:50:53 IST 2001
 copyright            : (C) 2001 by Zvika Brakerski, Asaf Koren
 email                : zvika@eng.tau.ac.il
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

#ifdef HAVE_CONFIG_H
#include <config.h>
#endif

#include <stdio.h>
#include <stdlib.h>

#include "listener.h"

int main(int argc, char *argv[])
{
  printf("Welcome to %s, version %s\n", PACKAGE, VERSION);
  listener(0);

  return EXIT_SUCCESS;
}

```

```
#ifndef __PROTOCOL_H__
#define __PROTOCOL_H__

/***************************************************************************
 protocol.h  -  description
 -------------------
 begin                : Sat Mar 10 2001
 copyright            : (C) 2001 by Zvika Brakerski, Asaf Koren
 email                : zvika@eng.tau.ac.il
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

  /*
   This file specifies the protocol for sent and received messages in the csicq
   project.
   */

// Messages from client to server
 #define CTOS_MSG 100
 #define CTOS(x) (CTOS_MSG+(x))
// Messages from server to client
 #define STOC_MSG 200
 #define STOC(x) (STOC_MSG+(x))

// C->S Messages
#define LOGIN_REQ       1
#define SEND_MSG                2
#define ADD_USR                 3
#define DEL_USR                 4
#define DISCONNECT      5


// S->C Messages
#define LOGIN_AUTH      1
#define RECV_MSG                2
#define UPDATE_USR      3
#define INF_DISCNCT     4
#define PSH_CNTCT               7

// Server authorizes request
#define S_AUTH                  50
#define SV(x)                           (S_AUTH+(x))

// Server refuses request
#define S_REFUSE                60
#define SX(x)                           (S_REFUSE+(x))


typedef struct {  // A struct to hold text fields of messages
        unsigned long length;
        char *text;
} text_field;


/*
 Client -> Server requests

 Following are definitions of structs that are used to hold requests from client
 to server.
 */

typedef struct {  // A struct to hold login requests
        unsigned long type;
        unsigned long uin;
        text_field password;
        text_field nick;
} login_req;
```

```
76
77  typedef struct {  // A struct to hold send requests
78          unsigned long type;
79          unsigned long uin;
80          text_field message;
81  } send_req;
82
83  typedef struct {  // A struct to hold add user requests
84          unsigned long type;
85          unsigned long uin;
86          text_field name;
87  } add_usr_req;
88
89  typedef struct {  // A struct to hold remove user requests
90          unsigned long type;
91          unsigned long uin;
92  } rem_usr_req;
93
94  typedef struct {  // A struct to hold disconnection requests
95          unsigned long type;
96  } disconnect_req;
97
98
99  /*
100  Server -> Client messages
101
102  Following are definitions of structs that are used to hold messages from the server
103  to the client.
104  */
105
106 typedef struct {  // A struct to hold authorize login message
107         unsigned long type;
108 } auth_login;
109
110 typedef struct {  // A struct to hold message received message
111         unsigned long type;
112         unsigned long from_uin;
113         text_field message;
114 } recv_message;
115
116 typedef struct {  // A struct to hold update status message
117         unsigned long type;
118         unsigned long uin;
119         unsigned long status;
120 } update_usr_status;
121
122 typedef struct {  // A struct to hold inform of disconnection message
123         unsigned long type;
124 } inform_disconnect;
125
126
127
128 /*
129  Following is a general union of all message types.
130  */
131 typedef union {  // a union to hold all possible messages
132         /* C->S */
133         login_req                                       l_req;
134         send_req                                                s_req;
135         add_usr_req                                     au_req;
136         rem_usr_req                                     ru_req;
137         disconnect_req                  d_req;
138
139         /* S->C */
140         auth_login                                      auth_msg;
141         recv_message                            recv_msg;
142         update_usr_status               update_msg;
143         inform_disconnect               inform_msg;
144
145 } generic_msg;
146
147
148 /*
149  Following are definitions concerning the icq network
150  */
```

```
151
152 #define ICQ_HOST "icq.mirabilis.com"
153 #define ICQ_PORT 4000
154
155
156
157 #endif  //#ifndef __PROTOCOL_H__
158
```

Functions	
ready_to_read	1:28
receive_message	1:55

```
/***************************************************************************
 services.c  -  description
 -------------------
 begin                : Mon Mar 12 2001
 copyright            : (C) 2001 by Zvika Brakerski, Asaf Koren
 email                : zvika@eng.tau.ac.il
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/


#include "services.h"



/*
 ready_to_read

 Waits until a socket is ready to read
 */
int ready_to_read(int s)
{
                fd_set readfds;
                struct timeval tv;

                FD_ZERO(&readfds);
                FD_SET(s, &readfds);

                tv.tv_sec = R_GIVEUP_SEC;
                tv.tv_usec = R_GIVEUP_USEC;

                select(s+1, &readfds, NULL, NULL, &tv);

                if (FD_ISSET(s, &readfds))
                        return 1;

                return 0;
}


/*
 receive_message

 This function receives a message and puts it in a buffer.
 The function allocates the buffer itself so when we're done
 with it, we need to call a cleanup function.
 */
char *receive_message(int s)
{
                unsigned long length, *pL;
                char *buf, *pC=(char *) &length;
                int size = 4, rec_size;

                while (size>0) {
                        if (ready_to_read(s))
                                rec_size = recv(s, pC, size, 0);
                        else
                                return NULL;
                        if (rec_size == 0)
                                return NULL;
                        if (rec_size == -1)
                                return NULL;
                        pC += rec_size;
                        size -= rec_size;
                }

#ifdef DEBUG
                        fprintf(stderr, "Going to allocate %lu bytes\n", ntohl(length
```

```
));
#endif

                buf = malloc(ntohl(length));
                pL = (unsigned long *) buf;

                if (!buf) return NULL;

                *pL = length;
                pC = buf+4;
                length = ntohl(length);
                length -= 4;

                while (length>0) {
                        if (ready_to_read(s))
                                rec_size = recv(s, pC, length, 0);
                        else
                                return NULL;
                        if (rec_size == 0)
                                return NULL;
                        if (rec_size == -1)
                                return NULL;
                        pC += rec_size;
                        length -= rec_size;
                }

                return buf;
}

/*
 clean_buf

 This function frees buffers allocated by the receive_message function.
 */
void clean_buf(char *buf)
{
                free(buf);
                return;
}

/*
 message_type

 This function returns the mesage type of the message stored in the buffer
 pointed to by the given pointer.
 */
unsigned long message_type(char *msg)
{
                unsigned long *pUL=(unsigned long *) (msg+4);
                return ntohl(*pUL);
}


/*
 ready_to_write

 Waits until a socket is ready to write
 */
int ready_to_write(int s)
{
                fd_set writefds;
                struct timeval tv;

                FD_ZERO(&writefds);
                FD_SET(s, &writefds);

                tv.tv_sec = W_GIVEUP_SEC;
                tv.tv_usec = W_GIVEUP_USEC;

                select(s+1, NULL, &writefds, NULL, &tv);

                if (FD_ISSET(s, &writefds))
                        return 1;

                return 0;
```

Functions

Functions	
announce_connection	3:185
clean_buf	2:109
message_type	2:121
ready_to_read	1:28
ready_to_write	2:133
receive_message	1:55
send_message	3:159

```
150   }
151
152
153
154   /*
155    send_message
156
157    This function sends a message over a socket.
158    */
159   int send_message(int s, char *msg, unsigned long len)
160   {
161                   int send_size;
162
163                   while (len>0)
164                   {
165                           if (ready_to_write(s))
166                                   send_size = send(s, msg, len, 0);
167                           else
168                                   return -1;
169                           if (send_size == -1)
170                                   return -1;
171                           if (send_size == 0)
172                                   return -1;
173                           msg += send_size;
174                           len -= send_size;
175                   }
176
177                   return 0;
178   }
179
180   /*
181    announce_connection
182
183    Informs of an incoming connection
184    */
185   int announce_connection(struct sockaddr_in *remote_addr)
186   {
187  #ifdef DO_LOG
188                   struct in_addr connected_addr;
189                   connected_addr.s_addr = remote_addr->sin_addr.s_addr;
190                   fprintf(LOG_FILE, ">>New connection from %s:%d\n", inet_ntoa(connecte
    d_addr), ntohs(remote_addr->sin_port));
191  #endif
192
193                   return 0;
194    }
195
```

```c
#ifndef __SERVICES_H__
#define __SERVICES_H__

/***************************************************************************
 services.h  -  description
 -------------------
 begin                : Mon Mar 12 2001
 copyright            : (C) 2001 by Zvika Brakerski, Asaf Koren
 email                : zvika@eng.tau.ac.il
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

#include <netinet/in.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>


// How much time we are willing to wait to read the rest of the information
#define R_GIVEUP_SEC    40
#define R_GIVEUP_USEC   0

// How much time we are willing to wait to write the rest of the information
#define W_GIVEUP_SEC    40
#define W_GIVEUP_USEC   0


// Prototypes
 char *receive_message(int s);
 void clean_buf(char *buf);
 unsigned long message_type(char *msg);
 int send_message(int s, char *msg, unsigned long len);
 int announce_connection(struct sockaddr_in *remote_addr);



#endif  //#ifndef __SERVICES_H__
```