

```

1  /*
2   * Group Number: Group D
3   * Group member name: Haidar Musaqalab
4   * Email: haidar.musaqalab@okstate.edu
5   */
6
7  // header inclusion
8  #include <stdio.h>
9  #include <sys/types.h>
10 #include <sys/socket.h>
11 #include <stdlib.h>
12 #include <unistd.h>
13 #include <arpa/inet.h>
14 #include <strings.h>
15 #include <string.h>
16 #include <pthread.h>
17 #include "Query.h"
18 #include "Record.h"
19
20 #define ID_NAME_FILE "IdName.txt"
21 #define SALARIES_FILE "Salaries.txt"
22 #define SATISFACTION_LEVEL_FILE "SatisfactionLevel.txt"
23 #define PORT 8000
24 #define LENGTH 1024
25
26 int getId(Query query) {
27     // function to get id from employee name
28     char buffer[LENGTH];
29     int id = -1;
30     FILE *fp;
31     int matches[32];
32     int count;
33     char temp[256];
34     char *token;
35     char *rest = buffer;
36     const char delim[2] = "\t";
37
38     fp = fopen(ID_NAME_FILE, "r"); // open file for read
39     if(fp == NULL) { // File couldn't be opened
40         return id;
41     }
42
43     while(fgets(buffer, LENGTH, fp)) {
44         if(strstr(buffer, query.employee_name)) { // found match
45             sscanf(buffer, "%d\t", &id);
46             matches[count] = id;
47             count++;
48         }
49     }
50     fclose(fp);
51
52     if (count <= 1)
53         return id;
54     else{
55         fp = fopen(SALARIES_FILE, "r");
56
57         // Look for possible matches in salaries file
58         while(fgets(buffer, LENGTH, fp)) {
59             token = strtok_r(buffer, delim, &rest);
60             id = (int)atoi(token);
61             token = strtok_r(NULL, delim, &rest);
62             strcpy(temp, token);
63
64             // Checking possible matches
65             for (int i = 0 ; i < count; i++){
66                 if (id == matches[i] && strcmp(query.job_title, temp) == 0){

```

```

67         fclose(fp);
68         return id;
69     }
70 }
71 }
72 }
73
74 printf("no match found\n");
75 fclose(fp);
76 return -1; // get the id
77 }
78
79 void *SatisfactionLevelSearch(void *arg) {
80     // thread function to get satisfaction level details
81     Record *record = (Record *)arg;
82     char buffer[LENGTH];
83     const char delim[2] = "\t";
84     char *token;
85     char *rest = buffer;
86     int i = 0;
87     FILE *fp;
88
89     fp = fopen(SATISFACTION_LEVEL_FILE, "r");
90     if(fp == NULL) { // File couldn't be opened
91         record->id = -1;
92     }
93     else { // file opened
94         while(fgets(buffer, LENGTH, fp)) { // get the data parsed with strtok
95             token = strtok_r(buffer, delim, &rest);
96             if(atoi(token) == record->id) {
97                 token = strtok_r(NULL, delim, &rest); // split with delim
98                 record->satisfaction_level = (float)atof(token);
99                 token = strtok_r(NULL, delim, &rest); // split with delim
100                record->number_project = atoi(token);
101                token = strtok_r(NULL, delim, &rest); // split with delim
102                record->average_monthly_hours = atoi(token);
103                token = strtok_r(NULL, delim, &rest); // split with delim
104                record->time_spend_company_in_yrs = atoi(token);
105                token = strtok_r(NULL, delim, &rest); // split with delim
106                record->work_accident = atoi(token);
107                token = strtok_r(NULL, delim, &rest); // split with delim
108                record->promotion_last_5years = atoi(token);
109                i = 1;
110            }
111        }
112    }
113    if(i == 0)
114        record->id = -1; // not found
115
116    }
117    fclose(fp);
118    pthread_exit(NULL); // exit thread
119 }
120
121 void *SalariesSearch(void *arg) {
122     // thread function to get salaries details
123     Record *record = (Record *)arg;
124     char buffer[LENGTH];
125     const char delim[2] = "\t";
126     char *token;
127     char *rest = buffer;
128     int i = 0;
129     FILE *fp;
130
131     fp = fopen(SALARIES_FILE, "r");
132     if(fp == NULL) { // File couldn't be opened

```

```

133     record->id = -1;
134 }
135 else { // file opened
136     while(fgets(buffer, LENGTH, fp)) {
137         token = strtok_r(buffer, delim, &rest);
138         if(atoi(token) == record->id) { // match found
139             token = strtok_r(NULL, delim, &rest); // split with delim
140             strcpy(record->job_title, token);
141             token = strtok_r(NULL, delim, &rest); // split with delim
142             record->base_pay = (float)atof(token);
143             token = strtok_r(NULL, delim, &rest); // split with delim
144             record->overtime_pay = (float)atof(token);
145             token = strtok_r(NULL, delim, &rest); // split with delim
146             record->benefit = (float)atof(token);
147             token = strtok_r(NULL, delim, &rest); // split with delim
148             strcpy(record->status, token);
149             record->status[2] = '\0';
150             i = 1;
151         }
152     }
153     if(i == 0)
154         record->id = -1; // no match
155 }
156 fclose(fp);
157 pthread_exit(NULL); // exit thread
158 }
159
160 int main(int argc, char **argv) {
161     // main function server
162     Record record;
163     Query query;
164     pthread_t tid1, tid2;
165     char buffer[LENGTH];
166     int connFd, fd, len;
167     int id;
168
169     struct sockaddr_in servaddr, cli;
170
171     fd = socket(AF_INET, SOCK_STREAM, 0); // create socket
172     if(fd == -1) { // server creation failed
173         printf(">> Error creating socket. Try again!!\n");
174         exit(-1);
175     }
176
177     // socket details endpoint
178     bzero(&servaddr, sizeof(servaddr));
179     servaddr.sin_family = AF_INET; // protocol
180     servaddr.sin_addr.s_addr = htonl(INADDR_ANY); // any ip
181     servaddr.sin_port = htons(PORT); //port num
182
183     // bind socket name
184     if((bind(fd, (struct sockaddr *)&servaddr, sizeof(servaddr))) != 0) {
185         printf(">> Bind call failed\n"); // bind call failed
186         exit(-1);
187     }
188
189     // listen for connections
190     if((listen(fd, 3)) != 0) {
191         printf(">> Listen call failed\n"); // listen call failed
192         exit(-1);
193     }
194
195     int b1, b2, b3;
196     len = sizeof(cli);
197     connFd = accept(fd, (struct sockaddr *)&cli, &len); // accept conn
198     if(connFd < 0) { // connect call failed

```

```

199     printf(">> Accept call failed\n");
200     exit(-1);
201 }
202 while(1) {
203
204     // communicate with Assistant here
205     if(read(connFd, &query, sizeof(Query)) > 1){
206
207         // End if needed
208         if (strcmp(query.employee_name, "exit") == 0 && strcmp(query.job_title, "exit") == 0 && strcmp(
query.status, "ex") == 0)
209             break;
210
211
212         // Clear existing values
213         bzero(&record, sizeof(Record));
214
215         id = getId(query); // get the id from employee name
216         record.id = id;
217         strcpy(record.employee_name, query.employee_name);
218
219         // create 2 threads
220         pthread_create(&tid1, NULL, SatisfactionLevelSearch, (void *)&record);
221         pthread_create(&tid2, NULL, SalariesSearch, (void *)&record);
222         pthread_join(tid1, NULL); // join thread 1
223         pthread_join(tid2, NULL); // join thread 2
224
225         printf("done.\n");
226         printf("%d %s %s %.2f %.2f %.2f %s %.2f %d %d %d %d %d\n",
record.id, record.employee_name,
227
228             record.job_title,
229             record.base_pay, // not working
230             record.overtime_pay, // not working
231             record.benefit, // not working
232             record.status, // not working
233
234             record.satisfaction_level, record.number_project,
235             record.average_monthly_hours, record.time_spend_company_in_yrs, record.work_accident,
236             record.promotion_last_5years);
237
238
239         // write back to client assistant
240         send(connFd, &record, sizeof(Record), 0); // changed from write to send.. mroyster
241         //close(connFd); // close client conn
242     }
243 }
244 close(fd); // close socket fd
245 return 0;
246 }

```