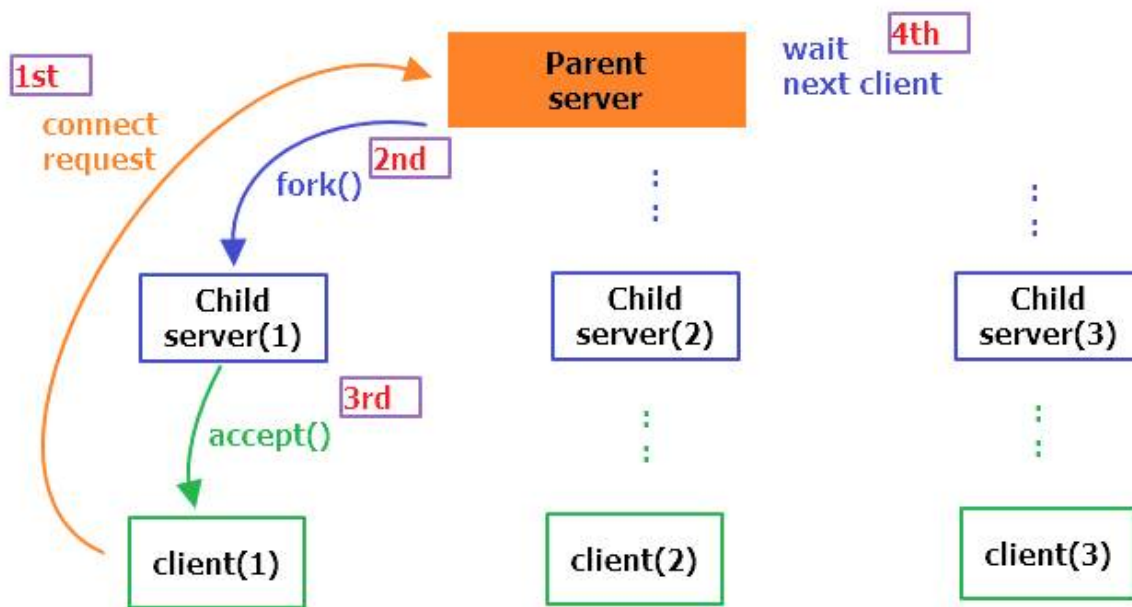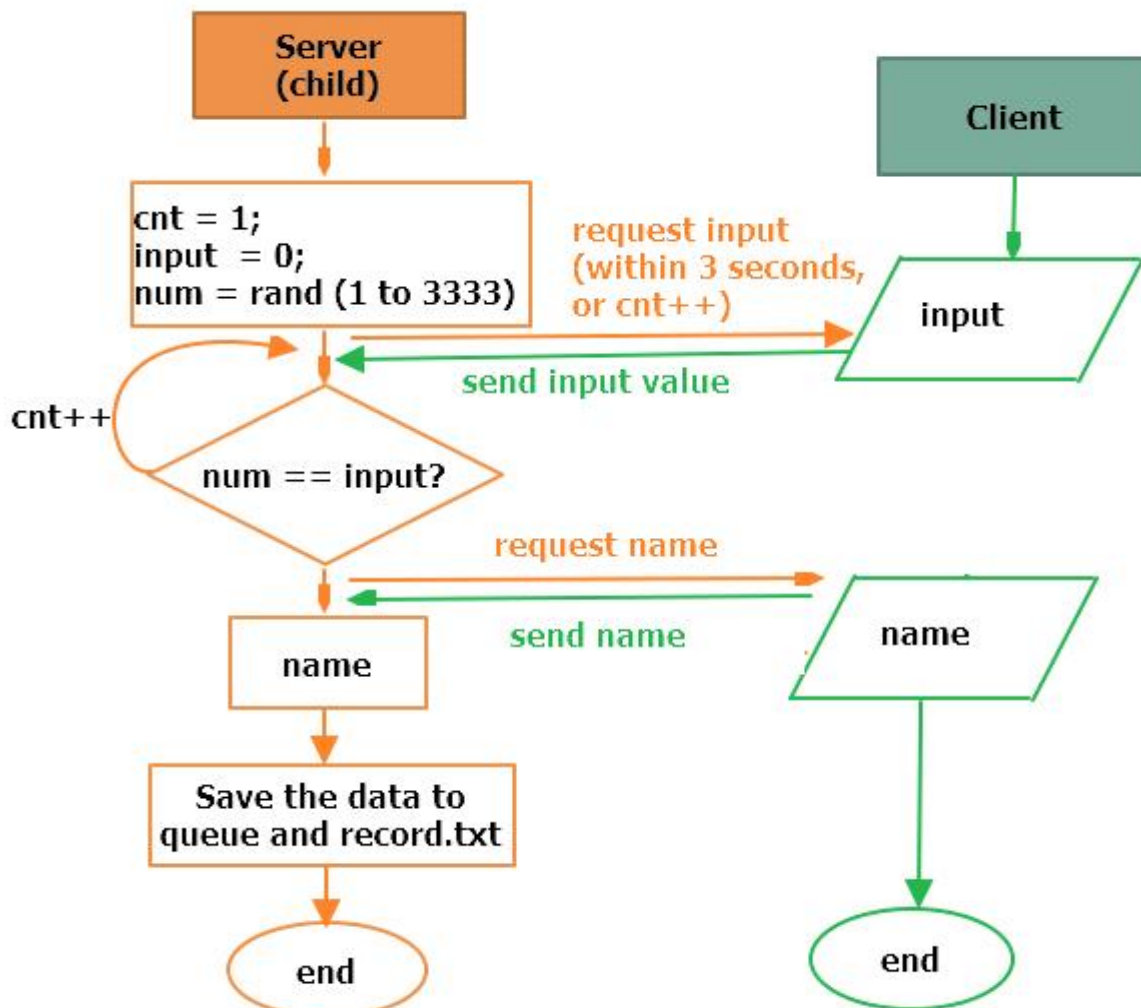**Implement UP&DOWN GAME with socket network.**

CS structure flowchart.

Game flowchart

result

1. right after executing client

```
기록보기(r), 시작하기(s), 종료하기(q) :
```

2. result from entering 'r' key

```
rank     name      cnt      tagetnum
1        정유경     10       935
2        황수정     10       2029
3        최대성     11       2686
4        김민호     11       962
5        이우석     11       2384
6        유형준     12       2705
7        김형주     12       34
8        신민철     12       1731
9        김시윤     12       1830
10       이호성     13       2386
11       정한별     13       1640
12       장성환     13       1258
13       윤연성     13       359
14       썜        14       3275
15       하성용     14       1752
16       박현우     14       1315
17       문한나     14       431
18       문지회     15       752
19       정상용     15       2429
20       이유성     16       442
21       윤지완     16       2555
22       안상재     17       948
23       은태영     21       3303
기록보기(r), 시작하기(s), 종료하기(q) :
```

3.playing (after press 's' key)

```
1000 보다 더 낮은 숫자야(2번째)
숫자를 맞춰봐!
```

4.after winning the game

```
정답! 592, 21번 시도
이름을 입력하세요!
```

5.record.txt file.

```
1   1    정유경     10    935
2   2    황수정     10    2029
3   3    최대성     11    2686
4   4    김민호     11    962
5   5    이우석     11    2384
6   6    유형준     12    2705
7   7    김형주     12    34
8   8    신민철     12    1731
9   9    김시윤     12    1830
10  10   이호성     13    2386
11  11   정한별     13    1640
12  12   장성환     13    1258
13  13   윤연성     13    359
14  14   썜        14    3275
15  15   하성용     14    1752
16  16   박현우     14    1315
17  17   문한나     14    431
18  18   문지회     15    752
19  19   정상용     15    2429
20  20   이유성     16    442
21  21   윤지완     16    2555
22  22   안상재     17    948
23  23   은태영     21    3303
```

```
Client.c
1    #include <stdio.h>
2    #include <stdlib.h>
3    #include <string.h>
4    #include <unistd.h>
5    #include <arpa/inet.h>
6    #include <sys/socket.h>
7    #include <stdlib.h>
8    typedef struct sockaddr_in     si;
9    typedef struct sockaddr *      sap;
10
11   #define BUF_SIZE        1024
12   #define RLT_SIZE        4
13   #define OPSZ            4
14
15   void err_handler(char *msg)
16   {
17       fputs(msg, stderr);
18       fputc('\n', stderr);
19       exit(1);
20   }
21
22   int main(int argc, char **argv)
23   {
24       int i, sock, result, opnd_cnt, nread;
25       char buf[BUF_SIZE] = {0};
26       char opmsg[BUF_SIZE] = {0};
27       si serv_addr;
28
29       if(argc != 3)
30       {
31           printf("use: %s <IP> <port>\n", argv[0]);
32           exit(1);
33       }
34
35       sock = socket(PF_INET, SOCK_STREAM, 0);
36
37       if(sock == -1)
38           err_handler("socket() error");
39
40       memset(&serv_addr, 0, sizeof(serv_addr));
41       serv_addr.sin_family = AF_INET;
42       serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
43       serv_addr.sin_port = htons(atoi(argv[2]));
44
45       if(connect(sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)
46           err_handler("connect() error");
47       else
48           puts("Connected ......");
49
50       char input[32];
51       int send_len;
52
53       for(;;)
54       {
55           system("clear");
56           nread = read(sock, buf, BUF_SIZE);
57           write(1, buf, nread);
58
```

```c
            //end messeage :
            if(!strncmp(buf," 정답!",4)){
                break;
            }
            else if(!strncmp(buf," 종료!",4)){
                goto exit;
            }
            send_len = read(0,input, sizeof(input));
            write(sock, input, send_len);

        }
        //send player's name to server.
        send_len = read(0,input, sizeof(input));
        write(sock, input, send_len);

        fflush(stdin);

        nread = read(0, buf, BUF_SIZE);
        write(sock, buf, nread);

        fflush(stdin);
        printf("\n");
        nread = read(sock, buf, BUF_SIZE);
        write(1, buf, nread);
        close(sock);

exit:
        return 0;
}
```

Server.c

```c
1    /*
2    compile methode :
3        gcc -o serv game1_serv.c
4        gcc -o clnt game1_clnt.c
5
6    execution :
7        ./serv PORTNUM                  (./serv 7777)
8        ./clnt IPADDRESS PORTNUM    (./clnt 127.0.0.1 7777)
9    record.txt :
10       file format
11       rank'\t'name'\t'count'\t'targetnumber'\n'
12
13   log.txt :
14       this file records the access of all clients with time.
15   */
16   /* For Network */
17   #include <stdio.h>
18   #include <stdlib.h>
19   #include <string.h>
20   #include <unistd.h>
21   #include <arpa/inet.h>
22   #include <sys/types.h>
23   #include <sys/socket.h>
24   #include <stdbool.h>
25
26   /* For System */
27   #include <time.h>
28   #include <fcntl.h>
29   #include <signal.h>
30   #include <stdbool.h>
31   #include <sys/wait.h>
32
33   //data structure : queue
34   //is for saving the records.
35   typedef struct _data{
36       char name[64];
37       int target_num;
38       int rank;
39       int cnt;
40   }data;
41   typedef struct __queue{
42       data p;
43       struct __queue* link;
44   }queue;
45   queue* get_queue_node(void);
46   void ins_queue_sorted(queue** head,data* p);
47   void print_queue(int clnt_sock);
48   typedef struct sockaddr_in     si;
49   typedef struct sockaddr *      sap;
50
51   #define BUF_SIZE        1024
52   #define OPSZ            4
53
54   int cnt;
55
56   void err_handler(char *msg);
57   void init_sock(si* serv_addr,socklen_t* clnt_addr_size,int* serv_sock,char* port,si clnt_addr);
58
```

```
59    void init_game(int* data);
60    bool cmp_num(int data,char* input,int clnt_sock,int cnt);
61    void start_game(int clnt_sock,queue** head);
62    void time_handler(int signo);
63    void open_record(int* fd,queue** head);
64    void re_record(int fd,queue** head);
65    int main(int argc, char **argv)
66    {
67        pid_t pid;
68        int status;
69
70        int serv_sock, clnt_sock;
71        char opinfo[BUF_SIZE];
72        char* port = argv[1];
73        int result, opnd_cnt, i;
74        int recv_cnt, recv_len;
75        char* start_msg = "기록보기(r), 시작하기(s), 종료하기(q) : \n";
76        char* end_msg = " 종료!\n";
77        char* wrong_msg = "r, s ,q 중 하나를 입력해주세요\n";
78        char ins[32];
79        int fd;
80        si serv_addr, clnt_addr;
81        socklen_t clnt_addr_size;
82        queue* head = NULL;
83
84
85        //misuse alert
86        if(argc != 2)
87        {
88            printf("use: %s <port>\n", argv[0]);
89            exit(1);
90        }
91
92        //bind(), listen(), set clnt_addr_size value
93        init_sock(&serv_addr, &clnt_addr_size, &serv_sock, port, clnt_addr);
94
95        //connection check.
96        for(i = 0; i < 30; i++)
97        {
98            //parent process only does wait for clients connecting.
99            clnt_sock = accept(serv_sock, (sap)&clnt_addr, &clnt_addr_size);
100
101            //record.txt to queue
102            open_record(&fd,&head);
103
104            if((pid = fork()) > 0){
105                printf("i : %d, child pid : %d, parent(me) pid : %ld\n",
106                                                i,pid,(long)getpid());
107
108            }
109            else if(pid == 0)
110            {
111
112        redo:
113                //send the message : "기록보기(r), 시작하기(s), 종료하기(q) :"
114                write(clnt_sock, start_msg , strlen(start_msg));
115                //receive user's instruction
116                read(clnt_sock, ins, sizeof(ins));
117
118                ins[1] = '\0';//flush the buffer('\n')
```

```
119
120                  switch(ins[0]){
121                      case 's':
122                          //start game through clnt_sock
123                          start_game(clnt_sock,&head);
124                          break;
125                      case 'r':
126                          //show the record file to player
127                          print_queue(clnt_sock);
128
129                          goto redo;
130                      case 'q':
131                          //terminate player
132                          write(clnt_sock, end_msg, strlen(end_msg));
133                          break;
134                      default :
135                          //undefined instruction, request another letter
136                          write(clnt_sock, wrong_msg, strlen(wrong_msg));
137
138                          goto redo;
139
140                  }
141              break;
142          }
143
144          //after game ends, recode.
145      }
146      close(serv_sock);
147      return 0;
148 }
149 void err_handler(char *msg)
150 {
151      fputs(msg, stderr);
152      fputc('\n', stderr);
153      exit(1);
154 }
155 void init_sock(si* serv_addr, socklen_t* clnt_addr_size, int* serv_sock,char* port,si clnt_addr){
156      *serv_sock = socket(PF_INET, SOCK_STREAM, 0);
157
158      if(*serv_sock == -1)
159          err_handler("socket() error");
160
161      memset(serv_addr, 0, sizeof(*serv_addr));
162      serv_addr->sin_family = AF_INET;
163      serv_addr->sin_addr.s_addr = htonl(INADDR_ANY);
164      serv_addr->sin_port = htons(atoi(port));
165
166      if(bind(*serv_sock, (sap)serv_addr, sizeof(*serv_addr)) == -1)
167          err_handler("bind() error");
168      if(listen(*serv_sock, 1) == -1)
169          err_handler("listen() error");
170
171      *clnt_addr_size = sizeof(clnt_addr);
172 }
173 void init_game(int* data){
174      *data = rand()%3333 + 1;
175 }
176 bool cmp_num(int data,char* input,int clnt_sock,int cnt){
177      int user = atoi(input);
178      char buf[64];
```

```c
179        if(data == user){
180            sprintf(buf," 정답! %d, %d번 시도\n",user,cnt);
181            write(clnt_sock, buf, strlen(buf));
182            return true;
183        }
184        if(data > user)
185            sprintf(buf,"%d 보다 더 높은 숫자야(%d번째)\n",user,cnt);
186        else if(data < user)
187            sprintf(buf,"%d 보다 더 낮은 숫자야(%d번째)\n",user,cnt);
188
189        write(clnt_sock, buf, strlen(buf));
190        return false;
191 }
192 void start_game(int clnt_sock,queue** head){
193
194        data p;
195        int data;
196        char buf[32] = "숫자를 맞춰봐!\n";
197        char end_msg[32] = "이름을 입력하세요!\n";
198        char input[32];
199        srand(time(NULL));
200        cnt = 0;
201        int recv_len;
202        //set rand number to "data", 1~3333
203        init_game(&data);
204
205        //time limit set
206        signal(SIGALRM,time_handler);
207
208        for(;;)
209        {
210            cnt++;
211
212            //send a message, "숫자를 맞춰봐!"
213            write(clnt_sock, buf , sizeof(buf));
214
215            //time limit : 3 seconds. should the time fly, cnt++
216            alarm(3);
217
218            //receive user's input(number).
219            recv_len = read(clnt_sock, input, sizeof(input));
220
221            //cmp_num returns true when data == input, otherwise, returns false
222            if(true == cmp_num(data, input, clnt_sock, cnt)){
223                p.cnt = cnt;
224                p.target_num = data;
225                break;
226            }
227        }
228        write(clnt_sock,end_msg,strlen(end_msg));
229
230        recv_len = read(clnt_sock, input, sizeof(input));
231        strncpy(p.name,input,recv_len - 1);
232
233        p.name[recv_len] = '\0';
234        p.rank = 1;
235
236        ins_queue_sorted(head,&p);
237
238
```

```
239        int fd;
240        system("mv record.txt record_backup.txt");
241        fd = creat("record.txt",0644);
242        re_record(fd,head);
243
244
245        print_queue(clnt_sock);
246        close(clnt_sock);
247
248    }
249    void time_handler(int signo){
250        //printf("cnt++\n");
251        cnt++;
252        alarm(3);
253    }
254    void open_record(int* fd,queue** head){
255        char buf[1024];
256        char d[32];
257        data p = {"",0,0,0};
258        int ret, tmp;
259        int start_idx = 0,end_idx = 0, i = 0, j = 0, chk;
260        if((*fd = open("record.txt", O_RDONLY)) < 0){
261            printf("서버 오류 : record.txt 파일 손상\n");
262            exit(1);
263        }
264        while((ret = read(*fd,buf,sizeof(buf))) > 0){
265            while(buf[i]){
266                if((buf[i] == '\t') || (buf[i] == '\n') ){
267                    end_idx = i;
268                    strncpy(d,buf+start_idx, end_idx - start_idx);
269                    d[end_idx - start_idx] = '\0';
270
271                    start_idx = i + 1;
272                    chk = j%4;
273
274                    //chk : 0,          1,          2,          3
275                    //     rank,      name,      cnt,      target_num
276                    switch(chk){
277                        case 0:
278                            //rank will be adjusted when insert to queue
279                            p.rank = 1;
280                            break;
281                        case 1:
282                            strncpy(p.name,d,strlen(d));
283                            break;
284                        case 2:
285                            tmp = atoi(d);
286                            p.cnt = tmp;
287                            break;
288                        case 3:
289                            tmp = atoi(d);
290                            p.target_num = tmp;
291                            //read data to queue
292                            ins_queue_sorted(head, &p);
293                            break;
294                    }
295                    j++;
296                }
297                i++;
298            }
```

```c
299        }
300        close(*fd);
301    }
302
303    //data structrue : queue
304    //for save the records.
305    queue* get_queue_node(void){
306        queue* tmp;
307        tmp = (queue*)malloc(sizeof(queue)*1);
308        tmp->link = NULL;
309        tmp->p.rank = 1;
310        return tmp;
311    }
312    void ins_queue_sorted(queue** head,data* p){
313        queue* tmp;
314        tmp = get_queue_node();
315
316        int flag = 0;
317        //insert & quick sort
318        while(*head){
319            if((*head)->p.cnt < p->cnt){
320                head = &(*head)->link;
321                p->rank = p->rank + 1;
322            }
323            else if((*head)->p.cnt >= p->cnt){
324                tmp->link = *head;
325                tmp->p = *p;
326                (*head) = tmp;
327                flag = 1;
328
329                head = &(*head)->link;
330                while(*head){
331                    (*head)->p.rank = (*head)->p.rank + 1;
332                    head = &(*head)->link;
333                }
334            }
335
336        }
337        if(flag == 0){
338            if(!(*head)){
339                (*head) = tmp;
340                tmp->p = *p;
341            }
342            else
343                printf("error\n");
344        }
345    }
346    void print_queue(int clnt_sock){
347    /*
348        queue* tmp = *head;
349        char buf[1024];
350        sprintf(buf,"rank\tname\tcnt\ttagetnum\n");
351        write(clnt_sock,buf,strlen(buf));
352        while(tmp){
353            sprintf(buf,"%d\t%s\t%d\t%d\n",
354                tmp->p.rank,tmp->p.name,tmp->p.cnt,tmp->p.target_num);
355            write(clnt_sock,buf,strlen(buf));
356            tmp = tmp->link;
357        }
358    */
```

```c
        char buf[1024];
        int fd, ret;
        sprintf(buf,"rank\tname\tcnt\ttagetnum\n");
        write(clnt_sock,buf,strlen(buf));

        fd = open("record.txt",O_RDONLY, 0644);
        ret = read(fd,buf,sizeof(buf));
        write(clnt_sock,buf,ret);
        close(fd);
}
void re_record(int fd,queue** head){
        queue* tmp = *head;
        char buf[1024];
        int i = 0, chk = 0, debug = 0;
        while(tmp){
                chk = i % 4;
                switch(chk){
                        case 0:
                                sprintf(buf,"%d\t",tmp->p.rank);
                                printf("%d\t",tmp->p.rank);
                                write(fd,buf,strlen(buf));
                                break;
                        case 1:
                                sprintf(buf,"%s\t",tmp->p.name);
                                printf("%s\t",tmp->p.name);
                                write(fd,buf,strlen(buf));
                                break;
                        case 2:
                                sprintf(buf,"%d\t",tmp->p.cnt);
                                printf("%d\t",tmp->p.cnt);
                                write(fd,buf,strlen(buf));
                                break;
                        case 3:
                                sprintf(buf,"%d\n",tmp->p.target_num);
                                printf("%d\n",tmp->p.target_num);
                                write(fd,buf,strlen(buf));
                                tmp = tmp->link;
                                debug++;
                                break;
                }
                i++;
        }
        printf("%d개 기록\n",debug);
        close(fd);
}
```
Colored by Color Scripter