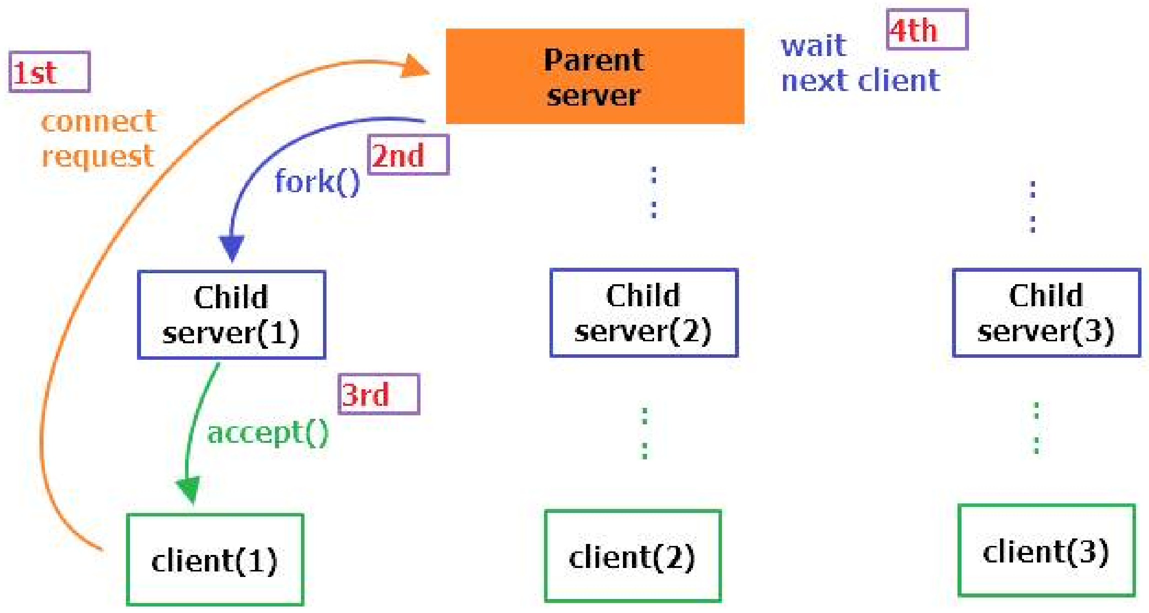
TI DSP, MCU, Xilinx Zynq FPGA Based Programming Expert Program

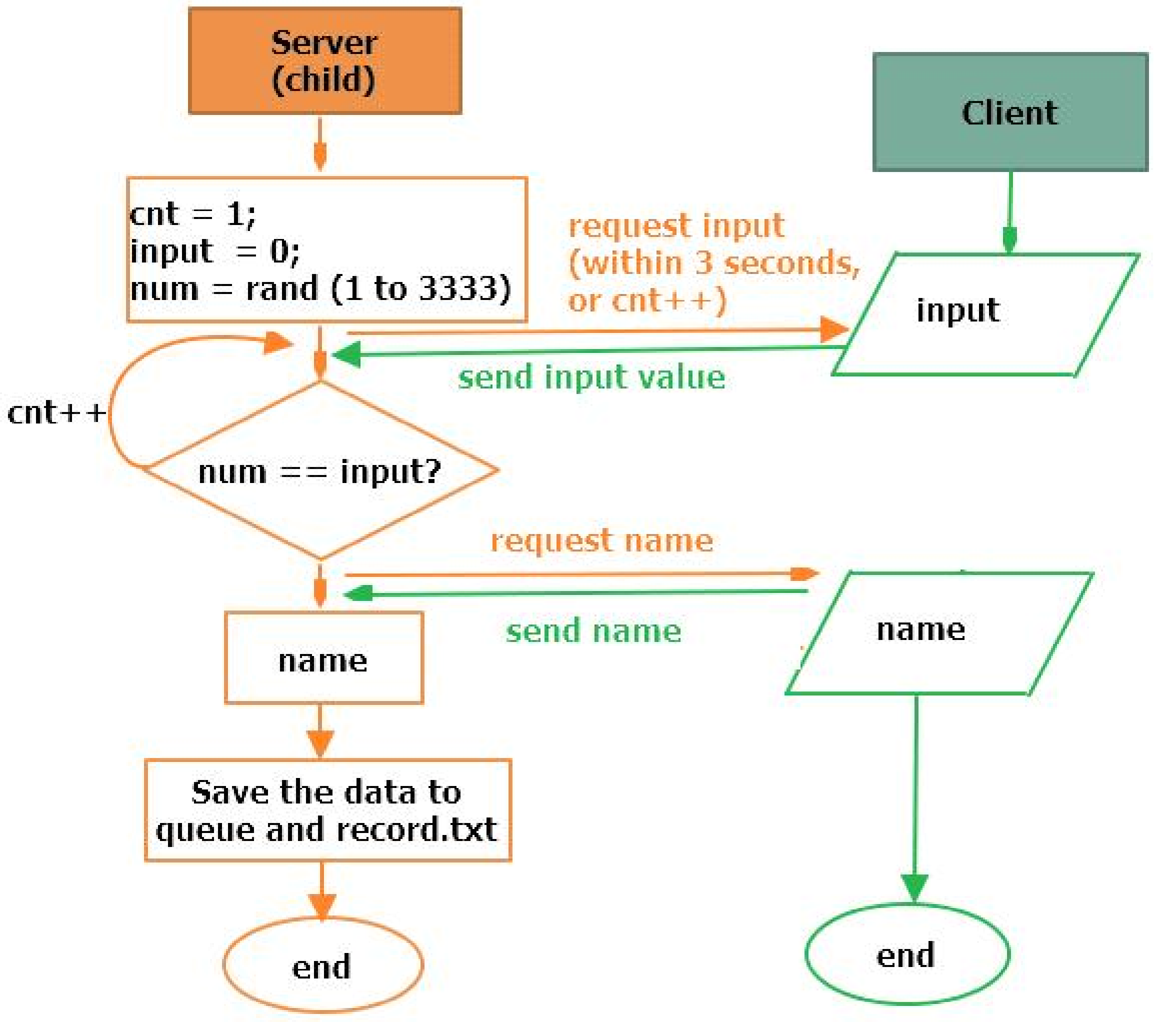
Instructor – Innova Lee(Sanghoon Lee)
gcccompil3r@gmail.com
Student – Hyungju Kim
mihaelkel@naver.com

Implement UP&DOWN GAME with socket network.

CS structure flowchart.

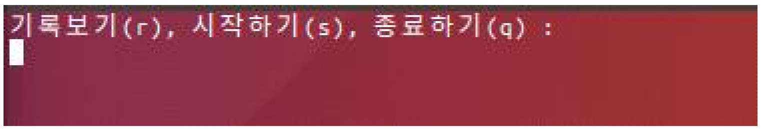


Game flowchart



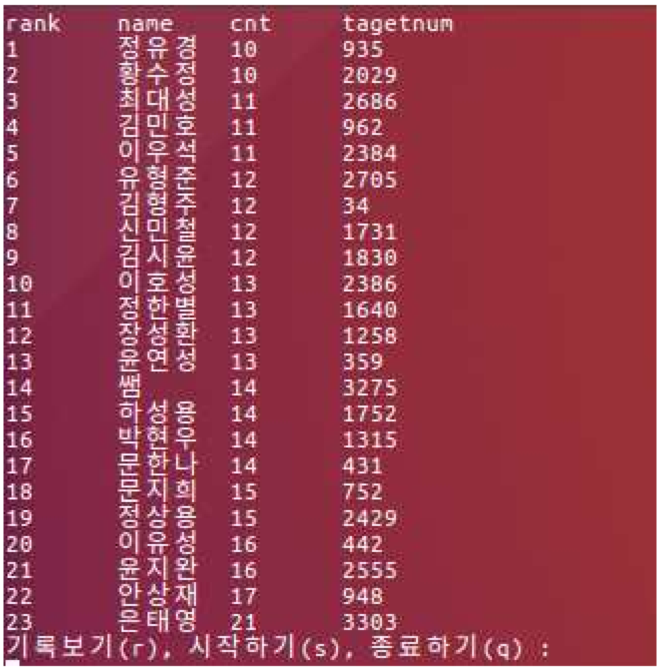
result

1. right after executing client



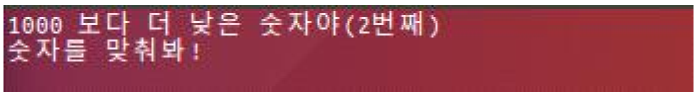
```
기록보기(r), 시작하기(s), 종료하기(q) :
```

2. result from entering ‘r’ key



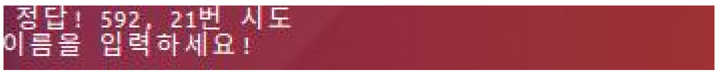
```
rank    name    cnt     tagetnum
1       정유경  10      935
2       황수정  10      2029
3       최대성  11      2686
4       김민호  11      962
5       이우석  11      2384
6       유형준  12      2705
7       김형주  12      34
8       신민철  12      1731
9       김시윤  12      1830
10      이호성  13      2386
11      정한별  13      1640
12      장성환  13      1258
13      윤연성  13      359
14      쌤      14      3275
15      하성용  14      1752
16      박현우  14      1315
17      문한나  14      431
18      문지희  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)



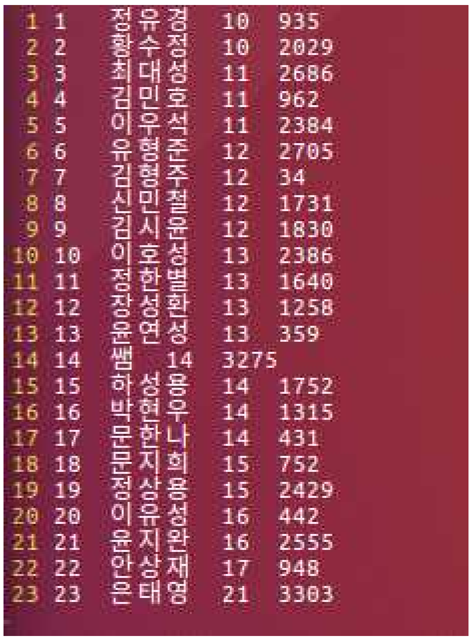
```
1000 보다 더 낮은 숫자야(2번째)
숫자를 맞춰봐!
```

4.after winning the game



```
 정답! 592, 21번 시도
이름을 입력하세요!
```

5.record.txt file.



```
1    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   쌤    14  3275
15   하성용  14  1752
16   박현우  14  1315
17   문한나  14  431
18   문지희  15  752
19   정상용  15  2429
20   이유성  16  442
21   윤지완  16  2555
22   안상재  17  948
23   은태영  21  3303
```

Client.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <stdlib.h>
typedef struct sockaddr_in      si;
typedef struct sockaddr *     sap;

#define BUF_SIZE         1024
#define RLT_SIZE         4
#define OPSZ             4

void err_handler(char *msg)
{
    fputs(msg, stderr);
    fputc('\n', stderr);
    exit(1);
}

int main(int argc, char **argv)
{
    int i, sock, result, opnd_cnt, nread;
    char buf[BUF_SIZE] = {0};
    char opmsg[BUF_SIZE] = {0};
    si serv_addr;

    if(argc != 3)
    {
        printf("use: %s <IP> <port>\n", argv[0]);
        exit(1);
    }

    sock = socket(PF_INET, SOCK_STREAM, 0);

    if(sock == -1)
        err_handler("socket() error");

    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
    serv_addr.sin_port = htons(atoi(argv[2]));

    if(connect(sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)
        err_handler("connect() error");
    else
        puts("Connected ......");

    char input[32];
    int send_len;

    for(;;)
    {
        system("clear");
        nread = read(sock, buf, BUF_SIZE);
        write(1, buf, nread);

        //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
/*
compile methode :
    gcc -o serv game1_serv.c
    gcc -o clnt game1_clnt.c

execution :
    ./serv PORTNUM                  (./serv 7777)
    ./clnt IPADDRESS PORTNUM     (./clnt 127.0.0.1 7777)
record.txt :
    file format
    rank'\t'name'\t'count'\t'targetnumber'\n'

log.txt :
    this file records the access of all clients with time.
*/
/* For Network */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <stdbool.h>

/* For System */
#include <time.h>
#include <fcntl.h>
#include <signal.h>
#include <stdbool.h>
#include <sys/wait.h>

//data structure : queue
//is for saving the records.
typedef struct _data{
    char name[64];
    int target_num;
    int rank;
    int cnt;
}data;
typedef struct __queue{
    data p;
    struct __queue* link;
}queue;
queue* get_queue_node(void);
void ins_queue_sorted(queue** head,data* p);
void print_queue(int clnt_sock);
typedef struct sockaddr_in     si;
typedef struct sockaddr *     sap;

#define BUF_SIZE        1024
#define OPSZ            4

int cnt;

void err_handler(char *msg);
void init_sock(si* serv_addr,socklen_t* clnt_addr_size,int* serv_sock,char* port,si clnt_addr);

void init_game(int* data);
bool cmp_num(int data,char* input,int clnt_sock,int cnt);
void start_game(int clnt_sock,queue** head);
void time_handler(int signo);
void open_record(int* fd,queue** head);
void re_record(int fd,queue** head);
int main(int argc, char **argv)
{
    pid_t pid;
    int status;

    int serv_sock, clnt_sock;
    char opinfo[BUF_SIZE];
    char* port = argv[1];
    int result, opnd_cnt, i;
    int recv_cnt, recv_len;
    char* start_msg = "기록보기(r), 시작하기(s), 종료하기(q) : \n";
    char* end_msg = " 종료!\n";
    char* wrong_msg = "r, s ,q 중 하나를 입력해주세요\n";
    char ins[32];
    int fd;
    si serv_addr, clnt_addr;
    socklen_t clnt_addr_size;
    queue* head = NULL;


    //misuse alert
    if(argc != 2)
    {
        printf("use: %s <port>\n", argv[0]);
        exit(1);
    }

    //bind(), listen(), set clnt_addr_size value
    init_sock(&serv_addr, &clnt_addr_size, &serv_sock, port, clnt_addr);

    //connection check.
    for(i = 0; i < 30; i++)
    {
        //parent process only does wait for clients connecting.
        clnt_sock = accept(serv_sock, (sap)&clnt_addr, &clnt_addr_size);

        //record.txt to queue
        open_record(&fd,&head);

        if((pid = fork()) > 0){
            printf("i : %d, child pid : %d, parent(me) pid : %ld\n",
                                                        i,pid,(long)getpid());

        }
        else if(pid == 0)
        {

        redo:
            //send the message : "기록보기(r), 시작하기(s), 종료하기(q) :"
            write(clnt_sock, start_msg , strlen(start_msg));
            //receive user's instruction
            read(clnt_sock, ins, sizeof(ins));

            ins[1] = '\0';//flush the buffer('\n')

            switch(ins[0]){
                case 's':
                    //start game through clnt_sock
                    start_game(clnt_sock,&head);
                    break;
                case 'r':
                    //show the record file to player
                    print_queue(clnt_sock);

                    goto redo;
                case 'q':
                    //terminate player
                    write(clnt_sock, end_msg, strlen(end_msg));
                    break;
                default :
                    //undefined instruction, request another letter
                    write(clnt_sock, wrong_msg, strlen(wrong_msg));

                    goto redo;

            }
            break;
        }

        //after game ends, recode.
    }
    close(serv_sock);
    return 0;
}
void err_handler(char *msg)
{
    fputs(msg, stderr);
    fputc('\n', stderr);
    exit(1);
}
void init_sock(si* serv_addr, socklen_t* clnt_addr_size, int* serv_sock,char* port,si clnt_addr){
    *serv_sock = socket(PF_INET, SOCK_STREAM, 0);

    if(*serv_sock == -1)
        err_handler("socket() error");

    memset(serv_addr, 0, sizeof(*serv_addr));
    serv_addr->sin_family = AF_INET;
    serv_addr->sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr->sin_port = htons(atoi(port));

    if(bind(*serv_sock, (sap)serv_addr, sizeof(*serv_addr)) == -1)
        err_handler("bind() error");
    if(listen(*serv_sock, 1) == -1)
        err_handler("listen() error");

    *clnt_addr_size = sizeof(clnt_addr);
}
void init_game(int* data){
    *data = rand()%3333 + 1;
}
bool cmp_num(int data,char* input,int clnt_sock,int cnt){
    int user = atoi(input);
    char buf[64];
```

```c
    if(data == user){
        sprintf(buf," 정답! %d, %d번 시도\n",user,cnt);
        write(clnt_sock, buf, strlen(buf));
        return true;
    }
    if(data > user)
        sprintf(buf,"%d 보다 더 높은 숫자야(%d번째)\n",user,cnt);
    else if(data < user)
        sprintf(buf,"%d 보다 더 낮은 숫자야(%d번째)\n",user,cnt);

    write(clnt_sock, buf, strlen(buf));
    return false;
}
void start_game(int clnt_sock,queue** head){

    data p;
    int data;
    char buf[32] = "숫자를 맞춰봐!\n";
    char end_msg[32] = "이름을 입력하세요!\n";
    char input[32];
    srand(time(NULL));
    cnt = 0;
    int recv_len;
    //set rand number to "data", 1~3333
    init_game(&data);

    //time limit set
    signal(SIGALRM,time_handler);

    for(;;)
    {
        cnt++;

        //send a message, "숫자를 맞춰봐!"
        write(clnt_sock, buf , sizeof(buf));

        //time limit : 3 seconds. should the time fly, cnt++
        alarm(3);

        //receive user's input(number).
        recv_len = read(clnt_sock, input, sizeof(input));

        //cmp_num returns true when data == input, otherwise, returns false
        if(true == cmp_num(data, input, clnt_sock, cnt)){
            p.cnt = cnt;
            p.target_num = data;
            break;
        }
    }
    write(clnt_sock,end_msg,strlen(end_msg));

    recv_len = read(clnt_sock, input, sizeof(input));
    strncpy(p.name,input,recv_len - 1);

    p.name[recv_len] = '\0';
    p.rank = 1;

    ins_queue_sorted(head,&p);


    int fd;
    system("mv record.txt record_backup.txt");
    fd = creat("record.txt",0644);
    re_record(fd,head);


    print_queue(clnt_sock);
    close(clnt_sock);

}
void time_handler(int signo){
    //printf("cnt++\n");
    cnt++;
    alarm(3);
}
void open_record(int* fd,queue** head){
    char buf[1024];
    char d[32];
    data p = {"",0,0,0};
    int ret, tmp;
    int start_idx = 0,end_idx = 0, i = 0, j = 0, chk;
    if((*fd = open("record.txt", O_RDONLY)) < 0){
        printf("서버 오류 : record.txt 파일 손상\n");
        exit(1);
    }
    while((ret = read(*fd,buf,sizeof(buf))) > 0){
        while(buf[i]){
            if((buf[i] == '\t') || (buf[i] == '\n') ){
                end_idx = i;
                strncpy(d,buf+start_idx, end_idx - start_idx);
                d[end_idx - start_idx] = '\0';

                start_idx = i + 1;
                chk = j%4;

                //chk : 0,          1,          2,          3
                //      rank,     name,     cnt,     target_num
                switch(chk){
                    case 0:
                        //rank will be adjusted when insert to queue
                        p.rank = 1;
                        break;
                    case 1:
                        strncpy(p.name,d,strlen(d));
                        break;
                    case 2:
                        tmp = atoi(d);
                        p.cnt = tmp;
                        break;
                    case 3:
                        tmp = atoi(d);
                        p.target_num = tmp;
                        //read data to queue
                        ins_queue_sorted(head, &p);
                        break;
                }
                j++;
            }
            i++;
        }
```

```c
    }
    close(*fd);
}

//data structrue : queue
//for save the records.
queue* get_queue_node(void){
    queue* tmp;
    tmp = (queue*)malloc(sizeof(queue)*1);
    tmp->link = NULL;
    tmp->p.rank = 1;
    return tmp;
}
void ins_queue_sorted(queue** head,data* p){
    queue* tmp;
    tmp = get_queue_node();

    int flag = 0;
    //insert & quick sort
    while(*head){
        if((*head)->p.cnt < p->cnt){
            head = &(*head)->link;
            p->rank = p->rank + 1;
        }
        else if((*head)->p.cnt >= p->cnt){
            tmp->link = *head;
            tmp->p = *p;
            (*head) = tmp;
            flag = 1;

            head = &(*head)->link;
            while(*head){
                (*head)->p.rank = (*head)->p.rank + 1;
                head = &(*head)->link;
            }
        }

    }
    if(flag == 0){
        if(!(*head)){
            (*head) = tmp;
            tmp->p = *p;
        }
        else
            printf("error\n");
    }
}
void print_queue(int clnt_sock){
/*
    queue* tmp = *head;
    char buf[1024];
    sprintf(buf,"rank\tname\tcnt\ttagetnum\n");
    write(clnt_sock,buf,strlen(buf));
    while(tmp){
        sprintf(buf,"%d\t%s\t%d\t%d\n",
            tmp->p.rank,tmp->p.name,tmp->p.cnt,tmp->p.target_num);
        write(clnt_sock,buf,strlen(buf));
        tmp = tmp->link;
    }
*/
```

```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);
}
```