HW4

2021115744_권구태

1. UDP 프로토콜을 이용한 알파벳 맞추기 게임

[코드 설명**]**

udp_server.c

```
int serv_sock;
   int str_len;
   socklen_t clnt_adr_sz;
   struct sockaddr_in serv_adr, clnt_adr;
   REQ_PACKET req_packet;
   RES_PACKET res_packet;
   char str_arr[BOARD_SIZE][BOARD_SIZE] = {'\0'};

   if (argc != 2) {
       printf("Usage : %s <port>\n", argv[0]);
       exit(1);
   }
   serv_sock = socket(PF_INET, SOCK_DGRAM, 0);

   if (serv_sock == -1) {
       error_handling("UDP socket creation error");
   }

   memset(&serv_adr, 0, sizeof(serv_adr));
   serv_adr.sin_family = AF_INET;
   serv_adr.sin_addr.s_addr = htonl(INADDR_ANY);
   serv_adr.sin_port = htons(atoi(argv[1]));

   if (bind(serv_sock, (struct sockaddr*)&serv_adr, sizeof(serv_adr))
== -1) {
       error_handling("bind() error");
   }
```

UDP 통신 설정. socket함수의 두번째 인자에 SOCK_DGRAM 사용.

```
    srand(time(NULL));

    initOriginalMatrix(str_arr);
    initCopyMatrix(res_packet.board);
```

랜덤 값 생성 위해 srand 함수 호출.
좌측에 출력할 랜덤 매트릭스 초기화(랜덤한 알파벳 삽입)
우측에 출력할 맞춘 매트릭스 '\0'으로 초기화

```
    printf("-------------------------------------\n");
    printf("    Finding Alphabet Game Server    \n");
    printf("-------------------------------------\n");

    printFrame(str_arr, res_packet.board);
```

매트릭스 두개 출력하는 함수 호출

```
clnt_adr_sz = sizeof(clnt_adr);
        str_len = recvfrom(serv_sock, &req_packet, sizeof(REQ_PACKET),
0,
                           (struct sockaddr *)&clnt_adr, &clnt_adr_sz);
        printf("[Server] Rx cmd=%d, ch=%c\n", req_packet.cmd,
req_packet.ch);
```

발신지의 주소의 크기를 설정하고, client로부터 정보를 받음.
첫번째 인자엔 생성한 소켓, 두번째 인자엔 데이터를 저장할 구조체(void로써 어떠한 형식이든 가능하다.), 세번째 인자에 두번째 인자의 크기, 4번째 인자에는 client의 주소, 5번째는 client의 주소의 크기.

정상적으로 받았으면 cmd값과, alphabet값을 출력한다.

```
        // 다 채웠으면 끝내려고 얼마나 비었는지 체크
        int space_count = spaceCount(res_packet.board);
```

```
if (space_count == 0) // 다 맞춤
        {

            // GAME_END 패킷 전송
            res_packet.cmd = GAME_END;
            res_packet.result = 0;

            printf("[Server] Tx cmd=%d, result=%d\n", res_packet.cmd,
res_packet.result);
```

```
            printf("No empty space. Exit this program.\n");

            sendto(serv_sock, &res_packet, sizeof(RES_PACKET), 0,
                (struct sockaddr *)&clnt_adr, clnt_adr_sz);

            break;
        }
```

빈자리 없으면(space_count == 0), cmd = 3 보내서 게임 끝내고, 끝났다는 멘트 출력

```
        if (req_packet.cmd == GAME_REQ)
        {
            int count = countAndInsertAlphabet(str_arr,
res_packet.board, &req_packet.ch);

            res_packet.cmd = GAME_RES;
            res_packet.result = count;

            printf("[Server] Tx cmd=%d, result=%d\n", res_packet.cmd,
res_packet.result);

            printFrame(str_arr, res_packet.board);

            sendto(serv_sock, &res_packet, sizeof(RES_PACKET), 0,
                (struct sockaddr *)&clnt_adr, clnt_adr_sz);
            sleep(1);
        }
```

그게 아니라면, cmd가 1이라면(빈자리가 있다는 얘기), client로부터 받은 알파벳을 복사 배열에(board[][])에 넣고, 넣은 만큼 count함. 그리고 그 넣은 행렬을 아래에 출력한다. 이후 client에 board[][]가 담긴 구조체를 보냄.

```
        else
        {
            printf("Invalid cmd: %d\n", req_packet.cmd);
        }
```

예외처리.

```
    printf("Exit Server Program\n");
    close(serv_sock);
```

끝나면 종료

udp_client.c

```
    int sock;
    int str_len;
    socklen_t adr_sz;
    struct sockaddr_in serv_adr, from_adr;
    REQ_PACKET req_packet;
    RES_PACKET res_packet;

    if (argc != 3) {
        printf("Usage : %s <port>\n", argv[0]);
        exit(1);
    }
    sock = socket(PF_INET, SOCK_DGRAM, 0);

    if (sock == -1) {
        error_handling("UDP socket creation error");
    }

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

udp 통신 초기 설정

```
req_packet.cmd = GAME_REQ;
    req_packet.ch = get_random_alphabet();

    sendto(sock, &req_packet, sizeof(REQ_PACKET), 0,
        (struct sockaddr *)&serv_adr, sizeof(serv_adr));

    printf("[Client] Tx cmd=%d, ch=%c\n", req_packet.cmd,
req_packet.ch);
```

cmd 값 1로 설정하고(게임 진행 알림)
랜덤하게 알파벳 하나 불러온 후, 서버로 보냄. 그리고 어떤 것을 보냈는지 출력한다.

```
adr_sz = sizeof(from_adr);
        str_len = recvfrom(sock, &res_packet, sizeof(RES_PACKET), 0,
```

```
            (struct sockaddr *)&from_adr, &adr_sz);

        printf("[Client] Rx cmd=%d, result=%d\n", res_packet.cmd,
res_packet.result);
```

recvfrom 함수 호출하기 위해 해당 길이를 sizeof연산으로 구함. socklen_t 형임.
길이가 담긴 변수의 주소값을 넘겨줘야 함
서버로부터, boar[][]가 담긴 res_packet 구조체를 받고 어떤 값을 받았는지 각각 출력.

```
if (res_packet.cmd == GAME_RES)
        {

            printFrame(res_packet.board);
            req_packet.cmd = GAME_REQ;
            req_packet.ch = get_random_alphabet();

            printf("[Client] Tx cmd=%d, ch=%c\n", req_packet.cmd,
req_packet.ch);

            sendto(sock, &req_packet, sizeof(REQ_PACKET), 0,
                (struct sockaddr *)&serv_adr, sizeof(serv_adr));

        }
```

만약 값이 2라면 board[][]를 출력하고, 게임을 계속 진행하기 위해 req_packet의 cmd값과 ch을 각각 1과 랜덤한 값으로 업데이트 한 후, 해당 값을 출력하고 서버로 보냄.

```
        else if (res_packet.cmd == GAME_END)
        {
            printf("No empty space. Exit this program.\n");
            break;
        }
```

3이라면 board[][]배열이 가득 찼다는 얘기고, 알파벳을 모두 맞춰 게임이 종료된다.

```
        else
        {
            printf("Invalid cmd: %d\n", res_packet.cmd);
        }
```

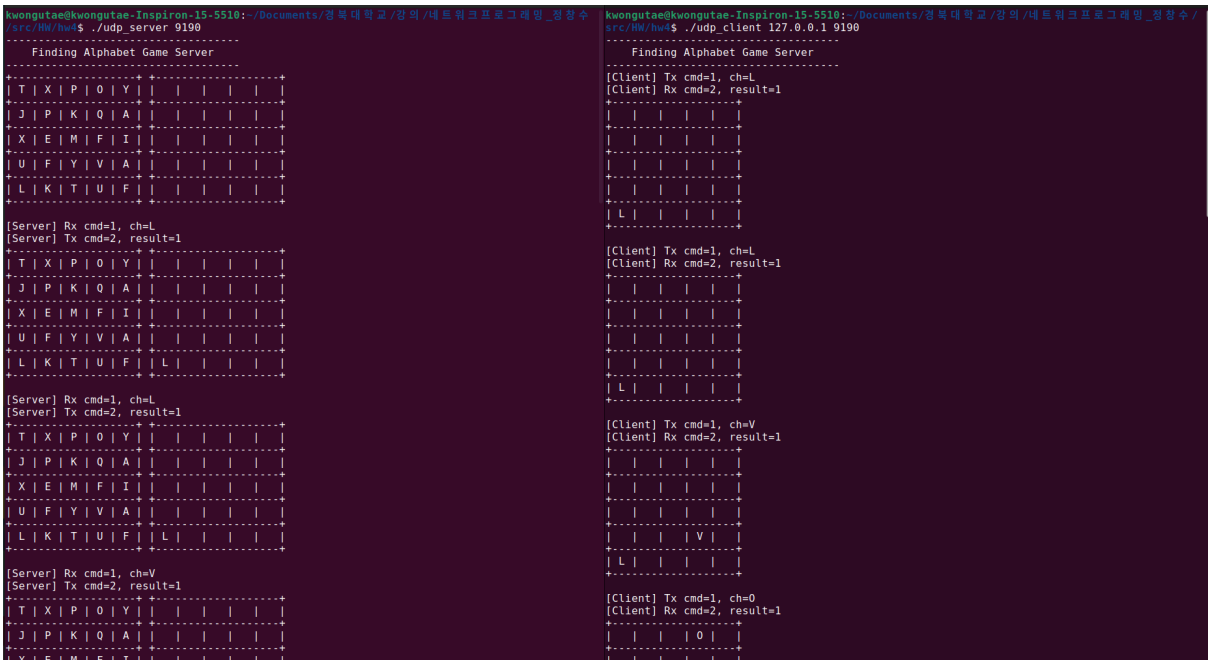
예외처리.

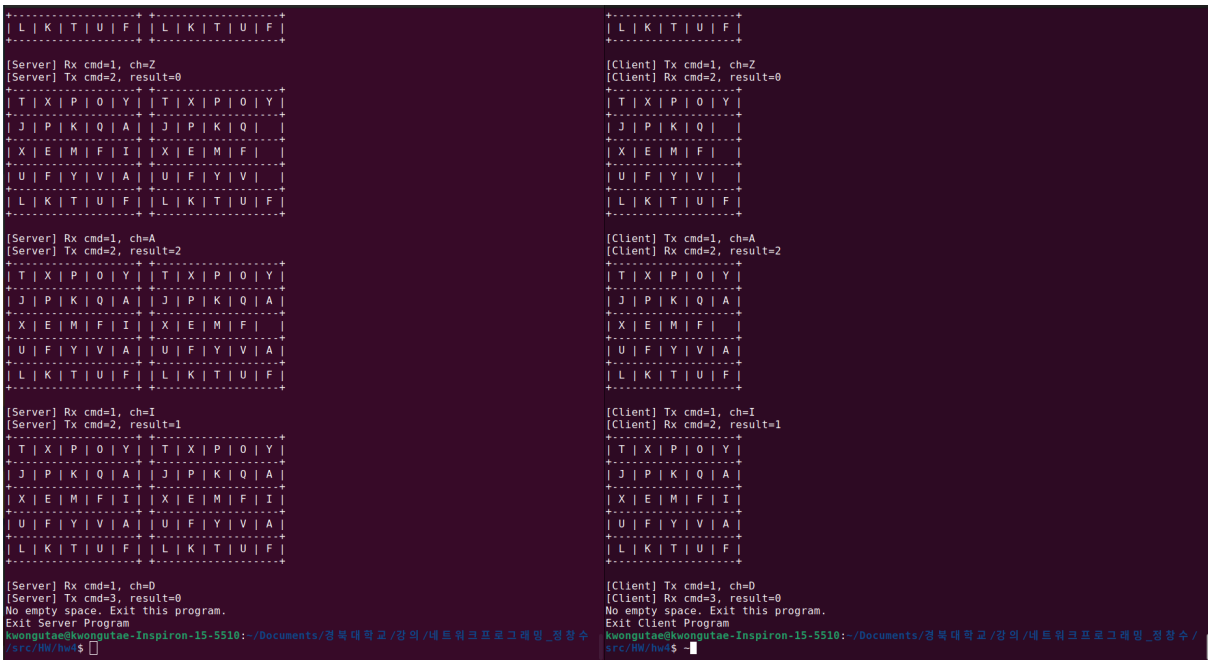
```
    printf("Exit Client Program\n");
```

```
    close(sock);
```

게임 종료

[실행 결과]





[주의할 점 및 새롭게 알게 된 점]

1. recvfrom()함수와 sendto()함수의 인자 값
   recvfrom(소켓, 받을 데이터를 저장할 버퍼의 **주소**, 받을 데이터의 크기(2번째 인자의 크기를 넣으면 됨), 0 넣기, 발신지 주소 정보를 저장할 sockaddr 구조체 "주소값", 5번인자의 길이를 담고 있는 변수의 "주소값");
   sendto(소켓, 보낼 데이터를 담고 있는 버퍼의 **주소**, 보낼 데이터의 크기(2번째 인자의 크기를 넣으면 됨), 0 넣기, 목적지 주소 정보를 저장할 sockaddr 구조체 "주소값", 5번인자의 길이를 담고 있는 변수의 "주소값");

2. 터미널에 뜨는 에러만으로 코드를 디버깅하는 방법
3. 함수인자를 넘길때 포인터 지정 방법
   char *var, char var[] -> 문자열 1개나 문자 하나를 받을 때
   char *var[5] -> 문자열 5개를 담은 배열
   char (*var)[5] -> 크기 5짜리 char 배열의 포인터
   char var[5][5] -> 크기가 지정된 문자열 배열 받을 때
   char **var -> 동적할당에 사용되는 이중 포인터를(문자열 배열) 받을 때

4. srand(time(NULL));을 한번만 사용해야 랜덤한 값이 사용된다는 점.