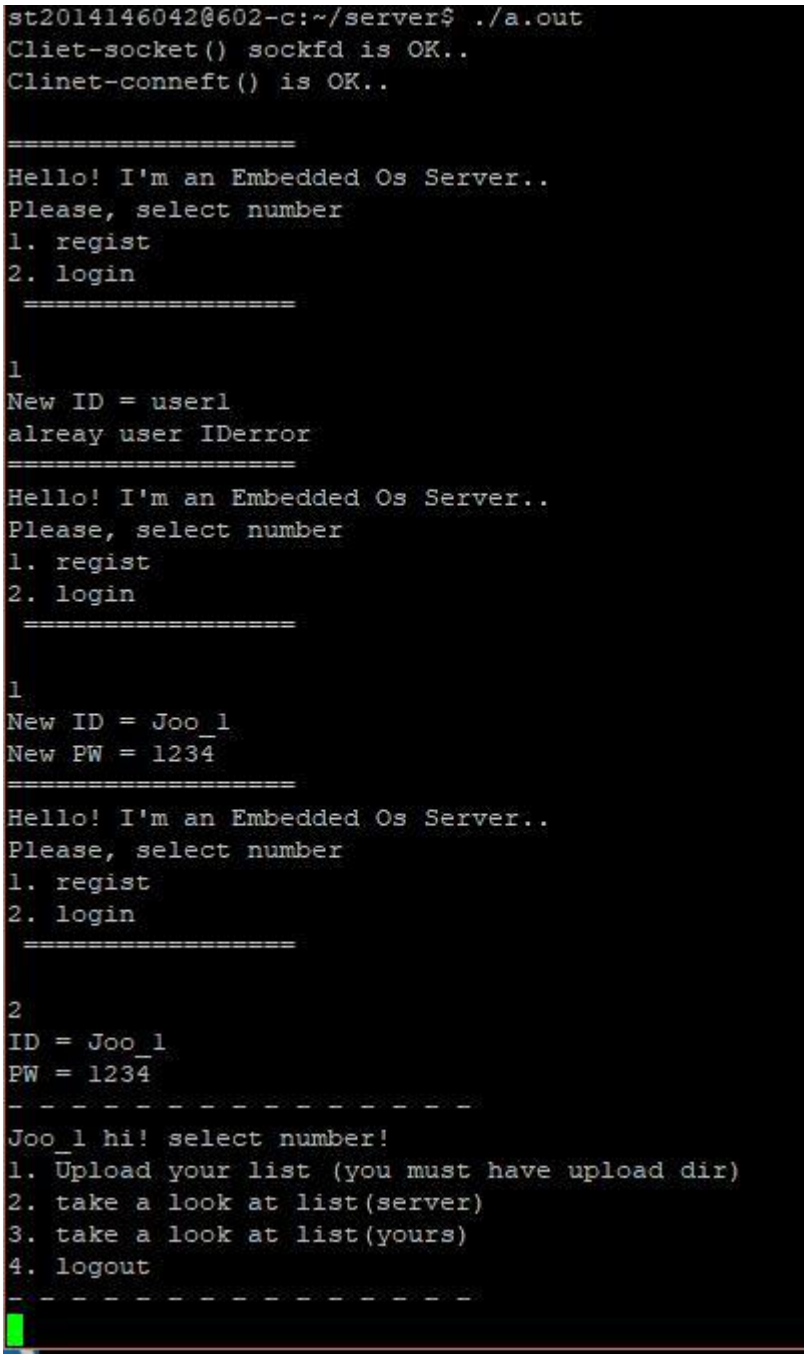
임베디드 운영체제 REPORT #2

조원 1 : 2014146041 조승아
조원 2 : 2014146042 주성민
부제 : 텀프로젝트 #2
교수 : 김 수민 교수님

[실행화면]

1) Joo_1 유저의 회원가입 후 로그인 과정입니다. 중복된 아이디를 검출하는 알고리즘도 구현하였습니다.



[회원가입 알고리즘 설명(서버)]

```
if(strcmp(msgg,"1")==0)
{
    listf = fopen("ID_LIST.txt","r+");
    send(new_fd,"New ID = ",sizeof(msgg),0);
    rcv_byte = recv(new_fd,new_id,sizeof(new_id),0);
    while(1)
    {
        fgets(comp_id,sizeof(comp_id),listf);
        if(strcmp(comp_id,"\0") == 0)
        {
            id_accept_flag = 1;
            send(new_fd,"New PW = ",sizeof(msgg),0);
            rcv_byte = recv(new_fd,new_pw,sizeof(new_pw),0);
            break;
        }
        comp_id[strlen(comp_id) - 1] = '\0';
        if(strcmp(new_id,comp_id) == 0)
        {
            printf("already used ID\n");
            send(new_fd,"alreay user ID",sizeof(msgg),0);
            break;
        }
    }
    if(id_accept_flag == 1)
    {
        id_accept_flag = 0;
        sprintf(id_file_name,"%s.txt",new_id);
        datalist = fopen(id_file_name,"w");
        fprintf(datalist,"%s\n",new_id);
        fprintf(datalist,"%s\n",new_pw);
        fprintf(listf,"%s\n",new_id);
        fclose(datalist);
    }
    fclose(listf);
}
```

먼저 ID_LIST.txt 파일을 열어준뒤 클라이언트에서 회원가입 할 아이디를 입력받습니다. 클라이언트 자체에서 “New ID = ”를 직접 출력할 수 있지만, 파일을 먼저 열고나서 입력을 받게 하는 순서를 지켜주기위해 이런 방식으로 구현하였습니다.

ID 중복검사를 수행하는 알고리즘입니다. 첫 번째 break 조건은 서버에 ID_LIST.txt에 동일한 ID가 있다면 반복문을 빠져나가게 되고, 두 번째 break 조건은 ID_LIST.txt에 더 이상 내용이 없을 때입니다. 마지막줄 까지 읽었는데도 new_id와 중복되는 ID가 없다면 id_accept_flag를 1로 설정하여 비밀번호 입력 알고리즘으로 이어집니다.

비밀번호를 입력받은 뒤 유저의 개인 파일을 만들어서 아이디와 비밀번호를 저장해줍니다. 비밀번호를 입력받으면 아이디와 함께 저장한 뒤, fclose함수로 FILE 구조체를 닫아주고 회원가입 알고리즘을 종료합니다. 간단한 flag로 조건을 걸어두고 조건안에서 flag를 초기화 했습니다.

[클라이언트]

```
while(1)
{
    rcv_byte = recv(sockfd,buf,sizeof(buf),0);//2
    printf("%s\n",buf);
    scanf("%s",msg);//로 그 인 , 회 원 가 입  선 택
    send(sockfd,msg,strlen(msg)+1,0);//3
    if(strcmp(msg,"1") == 0)
    {
        memset(msg,0,40);
        rcv_byte = recv(sockfd,msg,sizeof(buf),0);
        printf("%s",msg);
        memset(msg,0,40);
        scanf("%s",id);
        send(sockfd,id,strlen(id)+1,0);
        rcv_byte = recv(sockfd,msg,sizeof(buf),0);
        printf("%s",msg);
        if(strcmp(msg,"New PW = ") == 0 )
        {
            memset(msg,0,40);
            scanf("%s",pw);
            send(sockfd,pw,strlen(pw)+1,0);
        }
        else
        {
            printf("error\n");
            memset(msg,0,40);
        }
    }
```

첫 시작 메시지를 받으면서 소켓 통신이 시작됩니다. 회원가입과 로그인 메뉴를 선택하게 되며, 데모 실행화면과 관련된 회원가입 알고리즘이 시작되는 부분이다. 회원가입 선택 시 먼저 아이디 중복검사를 실행하게 됩니다. 서버에서 아이디 중복검사 알고리즘을 실행 후 회원가입이 가능한 아이디라면 “New PW =”문자열을 보내주고 이미 사용중인 아이디라면 “already use ID”문자열을 보냅니다.(코드에 문자열 스펠링 오타가 있어서 최종 코드에는 수정했습니다.) 클라이언트에서 문자열을 감지하여 회원가입을 계속 진행할 수 있다면 비밀번호를 입력 후 회원가입을 마치게 됩니다.

[로그인 알고리즘 설명(서버, 클라이언트)]

```
else if(strcmp(msgg,"2") ==0)
{
    rcv_byte = recv(new_fd,id,sizeof(id),0);
    rcv_byte = recv(new_fd,pw,sizeof(pw),0);
    listf = fopen("ID_LIST.txt","r");
    while(1)
    {
        if(fgets(comp_id,sizeof(comp_id),listf)==NULL)
        {
            printf("id error\n");
            send(new_fd,"login fail...plz check your ID" ,sizeof(msgg),0);
            break;
        }
        else
        {
            comp_id[strlen(comp_id) - 1] = '\0';
            if(strcmp(id,comp_id) == 0)
            {
                sprintf(id_file_name,"%s.txt",id);
                f = fopen(id_file_name,"r");
                fgets(comp_id,sizeof(comp_id),f);
                fgets(comp_pw,sizeof(comp_pw),f);
                comp_id[strlen(comp_id)-1] = '\0';
                comp_pw[strlen(comp_pw)-1] = '\0';

                if(strcmp(pw,comp_pw) != 0)
                {
                    printf("PW error\n");
                    send(new_fd,"login fail...plz check your PW" ,sizeof(msgg),0);
                }
                else
                {
                    printf("%s login!\n",id);
                    sprintf(msg,"- - - - - - - - - - - - - - -\n%s hi! select numbe
                    send(new_fd,msg,sizeof(msg),0);
                    memset(msgg,0,40);
                    while(1)
```

클라이언트 초기화면에서 2번을 선택 시 로그인 코드를 수행하게 됩니다. 모든 유저들의 아이디 목록을 열어서 유효한 아이디인지 확인 후 회원가입이 되어있는 유효한 아이디라면 반복문이 끝나기 전에 조건문에 들어가게 됩니다. 만약 끝까지 회원가입 된 아이디 목록 중 입력받은 아이디를 찾지 못한다면 서버에는“id error”라는 에러문구와 함께 ID를 확인해달라는 메시지를 보내게 됩니다.

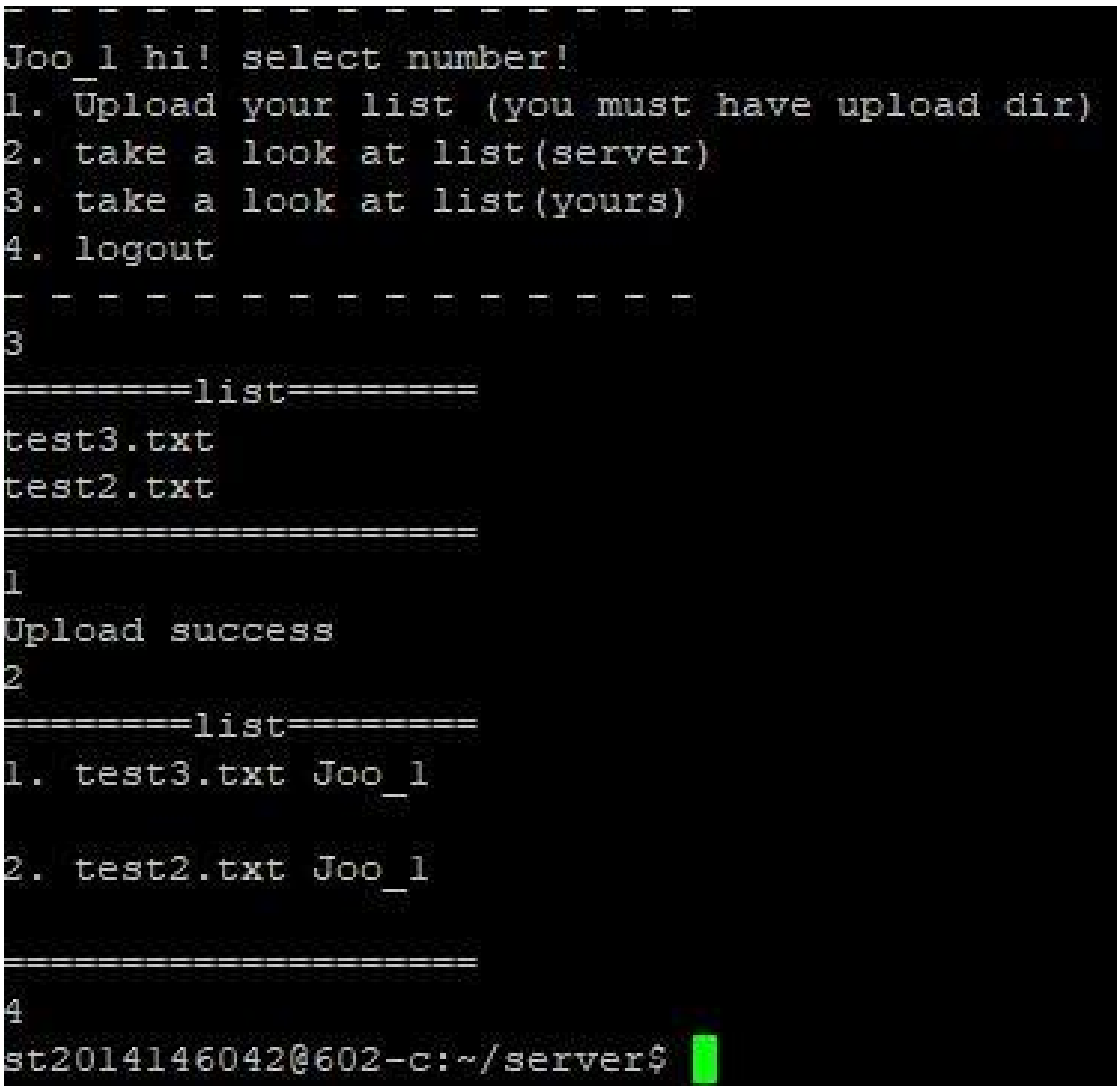
일치하는 아이디를 찾았다면 해당 아이디의 텍스트파일 (data 파일)을 열어서 두 번째 줄에 저장된 비밀번호와 입력받은 비밀번호를 비교하게 되고 일치한다면 로그인을 성공하게 되며 로그인 알고리즘이 종료됩니다.

```
else if(strcmp(msg,"2") ==0)
{
    printf("ID = ");
    memset(id,0,20);
    scanf("%s",id);
    printf("PW = ");
    memset(pw,0,20);
    scanf("%s",pw);
    send(sockfd,id,sizeof(id),0);
    send(sockfd,pw,sizeof(pw),0);
    memset(buf,0,512);
    rcv_byte = recv(sockfd,buf,sizeof(buf),0);
    printf("%s\n",buf);
    if(strstr(buf,"hi!") == NULL)
    {
        break;
    }
    else
    {
        memset(msg,0,40);
        memset(buf,0,512);
        while(1)
        {
            scanf("%s",msg);
            send(sockfd,msg,sizeof(msg),0);
            if(strcmp(msg,"1") == 0)
            {
```

클라이언트에서는 서버에비해 단순한 코드가 실행됩니다. 아이디와 비밀번호를 입력 후 서버의 응답을 기다립니다. 최종적으로 로그인 문구가 전송됐는지만 비교하고, 로그인 문구가 아니라면 수신한 에러 메세지를 그대로 출력된 뒤 반복문에서 빠져나와서 로그인에 실패하게 됩니다. 로그인 메시지를 받는다면 다음 코드를 순차적으로 진행하게 됩니다.

[실행화면]

2) Joo_1 유저의 업로드 가능 리스트 확인과 서버에 업로드를하는 과정입니다.



현재 서버에 Joo_1의 file list만 있습니다.

[알고리즘 설명(서버) - 서버에 리스트 등록]

```
if(strcmp(msgg,"1") == 0)
{
    sprintf(msgg,"./userslist/%s_list.txt",id);
    upload = fopen(msgg,"w");
    memset(buf,0,512);
    memset(msgg,0,40);
    i = 1;
    while(1)
    {
        rcv_byte = recv(new_fd,buf,sizeof(buf),0);
        if(strcmp(buf,"END")==0)
        {
            break;
        }
        sprintf(msgg,"%s %s",buf,id);
        fprintf(upload,"%s\n",msgg);
        memset(msgg,0,40);
    }
    fclose(upload);
    memset(msgg,0,40);
    dir_info = opendir("./userslist");
    upload = fopen("Total_list.txt","w");
    if(NULL != dir_info)
    {
        while(dir_entry = readdir(dir_info))
        {
            if(strstr(dir_entry->d_name,".txt") != NULL)
            {
                sprintf(msgg,"./userslist/%s",dir_entry->d_name);
                f_dir  = fopen(msgg,"r");
                memset(msgg,0,40);
                while(1)
                {
                    if(fgets(msgg,sizeof(msgg),f_dir)==NULL)
                    {
                        memset(msgg2,0,40);
                        memset(msgg,0,40);
                        break;
                    }
                    else
                    {
                        sprintf(msgg2,"%d. %s",i,msgg);
                        fprintf(upload,"%s",msgg2);
                        memset(msgg2,0,40);
                        memset(msgg,0,40);
                        i++;
                    }
                }
            }
        }
    }
    closedir(dir_info);
    fclose(upload);
    fclose(f_dir);
    sprintf(msgg,"%s's upload success",id);
    printf("%s\n",msgg);
    memset(msgg,0,40);
}
```

클라이언트에서 “1”을 입력받게 되면 먼저 msgg에 “userslist/유저의 이름_list,txt” 경로를 넣어주고, 해당경로의 파일을 열어줍니다.(없다면 만들어줍니다.)
클라이언트에서 파일 리스트를 한 개씩 입력받아서 해당 유저의 list파일을 서버에서 만들어 줍니다.
만약 클라이언트에서 END라는 명령어를 보낸다면 반복문을 탈출하게 됩니다.

유저의 개인 리스트를 만들어준 뒤 서버에서는 Total_list.txt 파일을 열어서 서버 내부의 userlist dir에서 유저들 개인의 리스트를 한 개씩 읽어옵니다. 최종적으로 Total_list.txt 에는 모든 유저들의 upload dir에있는 파일들의 리스트가 만들어집니다. 더 이상 읽을 유저의 리스트가 없다면 break 함수로 반복문을 탈출하게 됩니다.
strstr()함수를 사용하여 “.txt” 파일을 구별하였습니다.

실제 서버에 로그에 남는 것처럼 유저의 기록을 남기게 하였습니다.

[알고리즘 설명(서버) 그 외 기능들]

```
else if(strcmp(msgg,"2") == 0)
{
    upload = fopen("Total_list.txt","r");
    memset(msgg,0,40);
    while(1)
    {
        if(fgets(msgg,sizeof(msgg),upload)==NULL)
        {
            send(new_fd,"END" ,sizeof(msgg),0);
            memset(msgg,0,40);
            break;
        }
        else
        {
            send(new_fd,msgg,sizeof(msgg),0);
            memset(msgg,0,40);
        }
    }
    fclose(upload);
```

```
else if(strcmp(msgg,"3") == 0)
{
    memset(msgg,0,40);
    sprintf(msgg,"%s just take a look %s's list",id,id);
    printf("%s\n",msgg);
    memset(msgg,0,40);
}
else if(strcmp(msgg,"4") == 0)
{
    memset(msgg,0,40);
    break;//elseif "4" == logout
```

클라이언트에서 “2”를 입력받게 되면 “Total_list.txt”를 열어 한줄 씩 명령창에 출력해주게 됩니다. 텀프로젝트-3단계에서 다운로드 기능이 구현 될 구간입니다. 리스트를 다 읽게되면(혹은 리스트가 없으면) “END”문자열을 보내 서버의 리스트 입출력이 끝났다는 것을 알려주게 됩니다. 클라이언트에서 처리하는 과정은 뒤쪽 코드 설명에 되어있습니다.

클라이언트에서 “3”를 입력받게 되면 명령을 입력했던 클라이언트는 단순히 자기 자신의 업로드 리스트만 확인하게 되고 서버에서는 별다른 알고리즘없이 로그만 남겨주게 됩니다.

[알고리즘 설명(클라이언트)]

```
if(strcmp(msg,"1") == 0)
{
    dir_info = opendir("./upload");
    if(NULL != dir_info)//잘 못 된  dir이  아 니 라 면 !
    {
        while(dir_entry = readdir(dir_info))//파 일 을  하 나 씩  읽 어 줌
        {
            if(strstr(dir_entry->d_name,".txt") != NULL)
            {
                send(sockfd,dir_entry->d_name,sizeof(msg),0);
                sleep(1);
            }
        }
        closedir(dir_info);
        send(sockfd,"END",sizeof(msg),0);
        printf("Upload success\n");
    }
```

클라이언트에서 “1”을 입력하게 되면 나의 upload dir의 파일목록들을 하나씩 불러와서 문자열로 전송하게 됩니다. 서버에는 전송된 파일의 이름들로 구성된 “유저이름_list.txt파일”이 만들어지게 됩니다. 텀프로젝트 3단계에서는 IP,PORT등의 정보를 추가로 보낼 예정입니다.

```
if(strcmp(msg,"2") == 0)
{
    memset(msg,0,40);
    printf("========list========\n");
    while(1)
    {
        rcv_byte = recv(sockfd,msg,sizeof(msg),0);
        if(strcmp(msg,"END") == 0)
        {
            memset(msg,0,40);
            printf("====================\n");
            break;
        }
        else
        {
            printf("%s\n",msg);
            memset(msg,0,40);
        }
    }
```

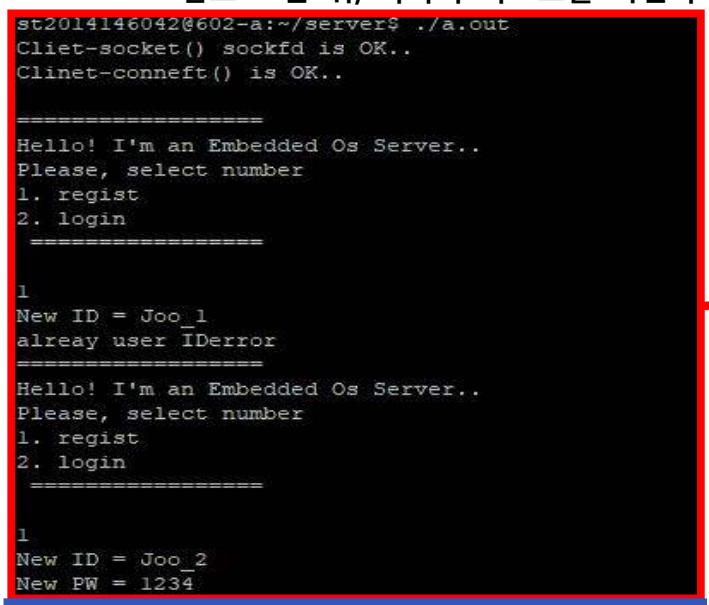
클라이언트에서 “2”를 입력하게 되면 서버의 리스트를 한 줄씩 받아오게 됩니다. “END”신호가 도착하면 반복문을 멈추고 출력되는 리스트의 끝선을 출력해줍니다. 텀프로젝트 3단계에서 다운로드 알고리즘을 구현할 구간입니다.

```
else if(strcmp(msg,"3") == 0)
{
    memset(msg,0,40);
    dir_info = opendir("./upload");
    if(NULL != dir_info)//잘 못 된  dir이  아 니 라 면 !
    {
        printf("========list========\n");
        while(dir_entry = readdir(dir_info))//파 일 을  하 나 씩  읽 어 줌
        {
            if(strstr(dir_entry->d_name,".txt") != NULL)
            {
                printf("%s\n",dir_entry->d_name);
            }
        }
        printf("====================\n");
        closedir(dir_info);
    }
}
```

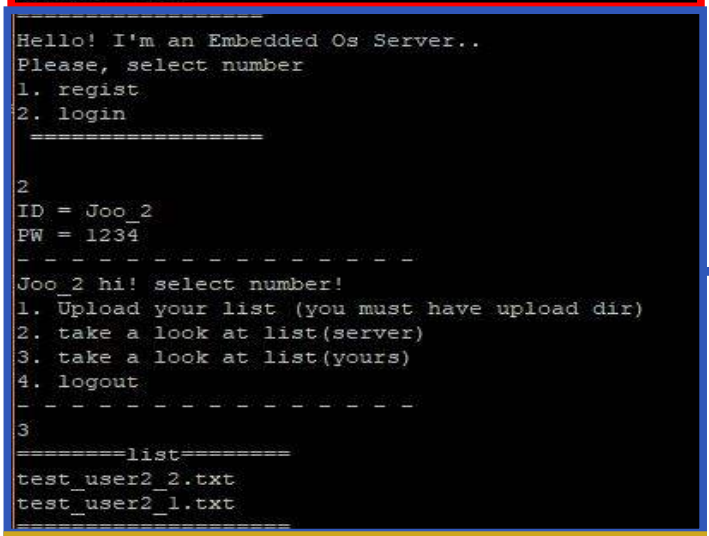
클라이언트에서 “3”을 입력하게 되면 유저는 자신의 upload dir에서 파일들의 이름을 한 개씩 불러와서 txt파일만 출력하게 됩니다. 이 구간에서도 strstr()함수를 사용하여 “.txt” 파일을 구별하였습니다.

[실행화면]

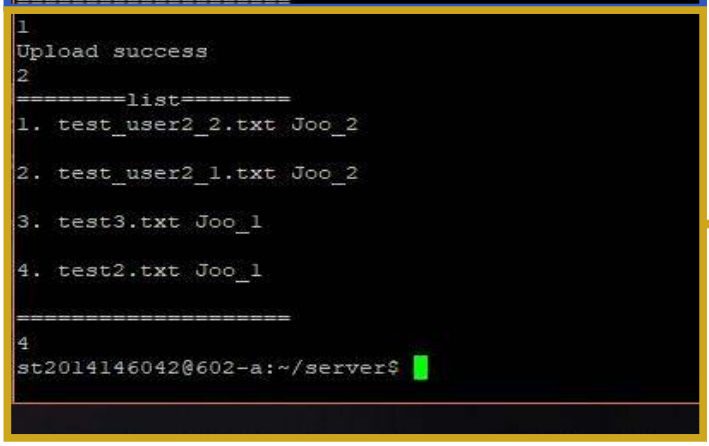
3) 클라이언트2에서 Joo_2 아이디로 회원가입과 로그인하여 서버에 리스트를 업로드 한 뒤, 서버의 리스트를 확인하기 까지의 실행 화면입니다.



중복된 ID를 검출하는 과정이 포함된 회원가입 알고리즘의 실행 화면입니다. Joo_1 아이디가 이미 사용 중이기 때문에 회원가입이 불가능하고 Joo_2로 회원가입을 실행하는 과정입니다.



회원가입을 끝내고 Joo_2 계정으로 로그인하여 먼저 자신의 리스트를 확인해보는 과정입니다.



메뉴에서 1번을 선택해서 Joo_2 계정의 리스트를 서버에 업로드 해주고, 이전 서버리스트에서 Joo_2의 파일이 추가된 리스트를 확인할 수 있습니다.