

제출일 : 2022-12-16



담당교수

김용혁 교수님

학 번

2021203034

학 과

소프트웨어학부

이 름

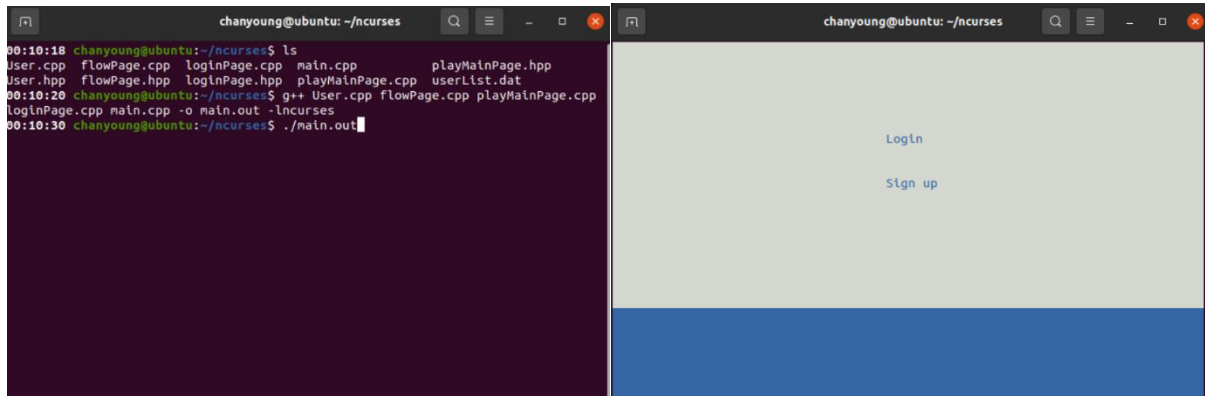
허찬영

KWANGWOON UNIVERSITY

메신저 프로그램(Project_02)

□ 컴파일 및 실행

Client



“g++ User.cpp flowPage.cpp playMainPage.cpp loginPage.cpp main.cpp -o main.out -lncurses” 명령어를 사용해서 컴파일 한다. 프로그램 실행은 ./main.out을 명령어를 이용해서 한다.

Server

“g++ User.cpp server.cpp -o server.out” 명령어를 사용해서 컴파일 한다. 프로그램 실행은 ./server.out 명령어를 이용해서 한다.

□ 소스 코드

Client는 앞서 만든 프로젝트와 유사하기 때문에 설명을 생략하겠습니다

server.cpp

```
const char *C2SFIFO = "./tmp/2021203034/C2S";
const char *S2CFIFO = "./tmp/2021203034/S2C";
const char *filepath = "./tmp/2021203034/userList.dat";
const char *messagePath = "./tmp/2021203034/messagePath.dat";

mkfifo(C2SFIFO, PERMS);
mkfifo(S2CFIFO, PERMS);

c2s = open(C2SFIFO, O_RDONLY);
s2c = open(S2CFIFO, O_WRONLY);

Client Cbuf("", "", "", 0, 0, 0);
Server Sbuf("", "", "", 0, 0, 0);
```

서버와 클라이언트와 통신하기 위해 FIFO를 사용했다. 서버 읽기, 클라이언트 쓰기 기능이 있는 C2S와 클라이언트 읽기, 서버 쓰기 기능이 있는 S2C를 사용해서 통신을 했다. 그리고 Client의 데이터를 담는 Client 클래스와 Server의 데이터를 담는 Server 클래스를 선언해서 데이터를 전달했다.

```
class Client{
public:
    Client();
    Client(std::string name, std::string id, std::string passwd, int login, int sign, int message);

    void setId(std::string id);
    void setName(std::string name);
    void setPasswd(std::string score);

    std::string getName(void);
    std::string getId(void);
    std::string getPasswd(void);
    int getLogin(void);
    int getSign(void);
    int getMessage(void);

private:
    char name[11];
    char id[11];
    char passwd[11];
    int login;
    int sign;
    int message;
};
```

client 클래스는 다음과 같다.

name과 id, passwd, 그리고 로그인 상태, 회원가입 상태, 메시지 전달 상태를 저장한다.

```
private:
    char sendId[21];
    char receiveId[21];
    char content[21];
    int login;
    int sign;
    int message;
```

server 클래스는 다음과 같다. 발신자의 ID, 수신자의 ID, 메시지 내용, 로그인 상태, 회원가입 상태, 메시지 전달 상태를 저장한다.

```
enum response
{
    SEREXIST = 0,
    SERSUCCESS = 1,
    LOGNOTID = 2,
    LOGNOTPW = 3,
    LOGSUCCESS = 4,
    MESSAGECOMPLETE = 5,
    ZEROMESSAGECOMPLETE = 6,
    INPUTUSER = 7,
    MAINNOTID = 8,
    MAINSELF = 9,
    MESSAGESUCCESS = 10,
    FMESSAGESUCCESS = 11
};
```

그리고 enum을 이용하여 서버와 클라이언트가 주고받을 여러가지 상태를 저장했다.

```
while (1) {
    if (read(c2s, &Cbuf, sizeof(Cbuf)) > 0) {
        if (Cbuf.getSign() == 1) {
            const char *tmp = Cbuf.getId().c_str();
            printf("<%s> try to sign up...", tmp);
            fflush(stdout);
            int fd = open(filepath, O_RDWR);
            if (fd > 0)
            {
                uniqueId = 0;
                while (true)
                {
                    int rSize = read(fd, &userReadInfo, sizeof(User));
                    if (rSize == 0)
                        break;
                }
            }
        }
    }
}
```

다시 server로 돌아와 설명을 하면 서버는 클라이언트의 입력을 받아 파일 입출력을 하고 다시 클라이언트로 보내는 역할을 한다. 이 코드에서 클라이언트의 입력을 받고 만약 회원가입을 요청하는 코드를 보낸다면 프로젝트1과 같이 예외를 검사하고 예외가 없으면 회원가입이 성공적으로 진행된 것을 클라이언트로 보낸다. 예외가 발생하면 어떤 예외가 발생했는지 클라이언트로 보내고 다음 작업은 클라이언트에서 수행하게 된다.

```

User saveUser(sname, sID, spasswd);
if (write(fd, &saveUser, sizeof(User)) == -1)
{
    perror("write() error!");
    return -1;
}

close(fd);
Server Sbuf("", "", "", 0, SERSUCCESS, 0);
printf("success\n");

write(s2c, &Sbuf, sizeof(Sbuf));
}
sflag = false;

```

이 코드는 회원가입이 성공했다는 것을 클라이언트로 보내는 코드이다. enum에서 정의한 상수를 이용하여 s2c FIFO로 클라이언트로 보낸다.

```

int c2s = 0, s2c = 0;
Client Cbuf(userInfo.getName(), userInfo.getId(), userInfo.getPasswd(), 0, 1, 0);
Server Sbuf;

const char* writeFIFO = "./tmp/2021203034/C2S";
const char* readFIFO = "./tmp/2021203034/S2C";

c2s = open(writeFIFO, O_WRONLY);
s2c = open(readFIFO, O_RDONLY);

write(c2s, &Cbuf, sizeof(Cbuf));
while (1) {
    if (read(s2c, &Sbuf, sizeof(Sbuf)) > 0) {
        if (Sbuf.getSign() == SERSUCCESS)
            break;
        else if (Sbuf.getSign() == SEREXIST){
            mvwprintw(win2, 0, 25, "Already exists ID");
            wrefresh(win2);

            key = getch();
            return 1;
        }
    }
}

```

s2c FIFO를 이용하여 서버로부터 받은 데이터를 읽고 서버에서 보낸 데이터에 따라 다음 작업을 수행한다.

로그인 부분은 회원가입과 거의 방식이 비슷하므로 생략한다.

```
else if (Cbuf.getMessage() == 1) {
    const char *tmp = Cbuf.getId().c_str();
    printf("<%s> loading user list...", tmp);
    fflush(stdout);
    int fd = open(messagePath, O_CREAT | O_APPEND | O_RDWR, 0644);
    if (fd > 0)
    {
        while (true)
        {
            messageUser messageInfo ("", "", "");
            int rSize = read(fd, &messageInfo, sizeof(messageUser));
            if (rSize == 0){
                Server Sbuf("", "", "", 0, 0, ZEROMESSAGECOMPLETE);
                write(s2c, &Sbuf, sizeof(Sbuf));
                break;
            }
            else if (rSize < 0){
                perror("error!");
                break;
            }

            std::string sendId = messageInfo.getSendId();
            std::string receiveId = messageInfo.getReceiveId();
            std::string content = messageInfo.getContent();
        }
    }
}
```

```
class messageUser{
public:
    messageUser();
    messageUser(std::string sendId, std::string receiveId, std::string content);
    void setSendId(std::string sendId);
    void setReceiveId(std::string receiveId);
    void setContent(std::string content);

    std::string getSendId(void);
    std::string getReceiveId(void);
    std::string getContent(void);
private:
    char sendId[21];
    char receiveId[21];
    char content[21];
};
```

다음은 메인 화면에서 사용자 목록을 만들고 사용자를 선택하는 코드이다. 이를 위해 messageUser라는 class를 정의하여 구현했다. 클라이언트가 메인 화면으로 들어오게 되면 서버가 이진 파일에 저장되어 있는 데이터를 읽어 사용자 목록을 확인하고 클라이언트에 전달한다.

```

        Server Sbuf(sendId, receiveId, content, 0, 0, MESSAGECOMPLETE);
        write(s2c, &Sbuf, sizeof(Sbuf));
    }
    printf("end\n");
    close(fd);
}
else if (Cbuf.getMessage() == INPUTUSER)
{
    const char *tmp = Cbuf.getId().c_str();
    int cnt = 0;
    bool flag = false;
    printf("<%s> enter message window...", tmp);
    fflush(stdout);
    int saveFd = open(messagePath, O_CREAT | O_APPEND | O_WRONLY, 0644);
    if (saveFd < 0){
        perror("error!");
    }
    int readFd = open(filepath, O_RDONLY);
    if (readFd < 0){
        perror("error!");
    }
}

```

만약 성공적으로 처리 됐다면 처리 됐다는 문구와 함께 클라이언트로 성공했음을 알린다.

그리고 다음 else if (Cbuf.getMessage() == INPUTUSER) 코드는 클라이언트에서 입력한 데이터를 받아 사용자를 선택하는 코드다.

```

if (Cbuf.getId() == Cbuf.getPasswd())
{
    flag = true;
}
else
{
    while (true)
    {
        User checkUser("", "", "");
        int rSize = read(readFd, &checkUser, sizeof(User));
        if (rSize == 0)
        {
            break;
        }
        else if (rSize < 0)
        {
            perror("error!");
            break;
        }

        std::string tmpName = checkUser.getName();
        std::string tmpId = checkUser.getId();
        bool fflag = false;
    }
}

```

만약 사용자가 입력한 데이터가 자신의 ID랑 같으면 예외 처리를 하고 그렇지 않다면 사용자의 정보가 저장된 파일을 열어 입력된 해당 ID가 있는지 확인한다.

```

if (Cbuf.getPasswd() == tmpId)
{
    cnt++;
    int readMessageFd = open(messagePath, O_RDONLY);

    messageUser tmpUsr("", "", "");
    while (true)
    {
        if (read(readMessageFd, &tmpUsr, sizeof(messageUser)) > 0){
            if (Cbuf.getPasswd() == tmpUsr.getSendId() || Cbuf.getPasswd() == ""){
                fflag = true;
            }
        }
        else
            break;
    }

    if (fflag == false){
        messageUser saveMessageUser(Cbuf.getPasswd(), Cbuf.getId(), "");
        if (write(saveFd, &saveMessageUser, sizeof(messageUser)) == -1)
        {

```



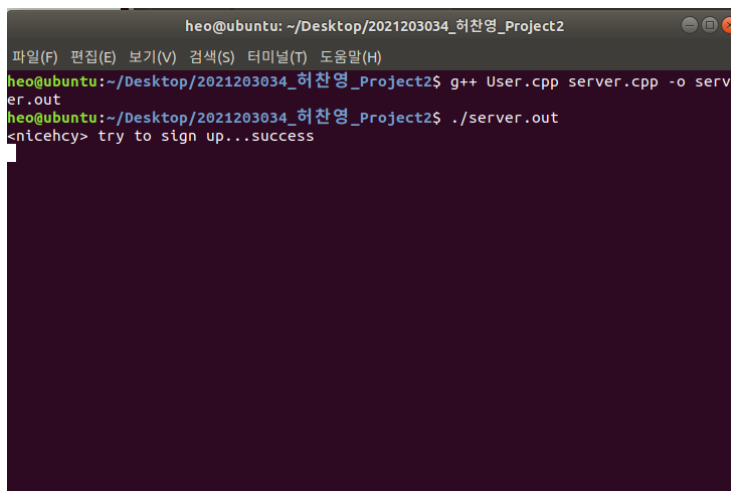
```

if (fflag == false){
    messageUser saveMessageUser(Cbuf.getPasswd(), Cbuf.getId(), "");
    if (write(saveFd, &saveMessageUser, sizeof(messageUser)) == -1)
    {
        perror("write() error!");
        return -1;
    }
    Server Sbuf("", "", "", 0, 0, MESSAGESUCCESS);
    write(s2c, &Sbuf, sizeof(Server));
    printf("success\n");
    break;
}
else{
    Server Sbuf("", "", "", 0, 0, FMESSAGESUCCESS);
    write(s2c, &Sbuf, sizeof(Server));
    printf("success\n");
    break;
}
}

```

만약 있으면 클라이언트가 입력한 사용자를 messageUser를 통해 다른 이진 파일에 저장하고 클라이언트에게 성공했음을 보낸다. 만약 없다면 실패했음을 보낸다.

□ 프로그램 실행 결과



```

heo@ubuntu: ~/Desktop/2021203034_허찬영_Project2
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
heo@ubuntu:~/Desktop/2021203034_허찬영_Project2$ g++ User.cpp server.cpp -o server.out
heo@ubuntu:~/Desktop/2021203034_허찬영_Project2$ ./server.out
<nicehcy> try to sign up...success

```

회원 가입이 성공했음을 출력한다.

```
heo@ubuntu: ~/Desktop/2021203034_허찬영_Project2
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
heo@ubuntu:~/Desktop/2021203034_허찬영_Project2$ g++ User.cpp server.cpp -o server.out
heo@ubuntu:~/Desktop/2021203034_허찬영_Project2$ ./server.out
<nicehcy> try to sign up...success
<nicehcy> try to sign up...denied -Already exists ID
```

만약 이미 아이디가 있으면 오류 메시지를 출력한다. (로그인도 비슷해서 생략하겠습니다)

```
heo@ubuntu: ~/Desktop/2021203034_허찬영_Project2
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
heo@ubuntu:~/Desktop/2021203034_허찬영_Project2$ g++ User.cpp server.cpp -o server.out
heo@ubuntu:~/Desktop/2021203034_허찬영_Project2$ ./server.out
<nicehcy> try to sign up...success
<nicehcy> try to sign up...denied -Already exists ID
<ekqls> try to sign up...success
<nich> try to login...denied -ID doesn't exist
<nicehcy2> try to login...denied -ID doesn't exist
<nicehcy2> try to login...denied -ID doesn't exist
<nicehcy> try to login...denied -ID doesn't exist
<nicehcy> try to login...success
<nicehcy> loading user list...end
```

클라이언트가 메인 화면으로 들어가면 클라이언트에게 유저 리스트를 보내고 완료 됐음을 출력한다.

```
heo@ubuntu: ~/Desktop/2021203034_허찬영_Project2
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
heo@ubuntu:~/Desktop/2021203034_허찬영_Project2$ g++ User.cpp server.cpp -o server.out
heo@ubuntu:~/Desktop/2021203034_허찬영_Project2$ ./server.out
<nicehcy> try to sign up...success
<nicehcy> try to sign up...denied -Already exists ID
<ekqls> try to sign up...success
<nich> try to login...denied -ID doesn't exist
<nicehcy2> try to login...denied -ID doesn't exist
<nicehcy2> try to login...denied -ID doesn't exist
<nicehcy> try to login...denied -ID doesn't exist
<nicehcy> try to login...success
<nicehcy> loading user list...end
<nicehcy> enter message window...success
█
```

클라이언트가 특정 사용자를 올바르게 입력해서 메시지 윈도우에 들어가게 되면 성공 했음을 알리고 예외가 발생하면 예외가 발생했음을 출력한다.

□ 고찰 리포트

느낀 점:

오랜 만에 간단한 프로그램을 제작해 보았는데 상당히 까다롭고 복잡했다. 원인을 알 수 없는 오류가 너무 많았고 예외적인 상황이 많아 코드가 복잡해진 것 같다.

프로젝트 수행 과정:

Incurses 라이브러리를 이해하기 위해 강의 자료와 인터넷에서 정보를 찾아보고 직접 코드를 작성해보면서 라이브러리를 이해할 수 있었다. 그리고 코드를 작성하였고 예상하지 못한 오류를 수정하면서 프로그램을 제작했다.