제출일 :

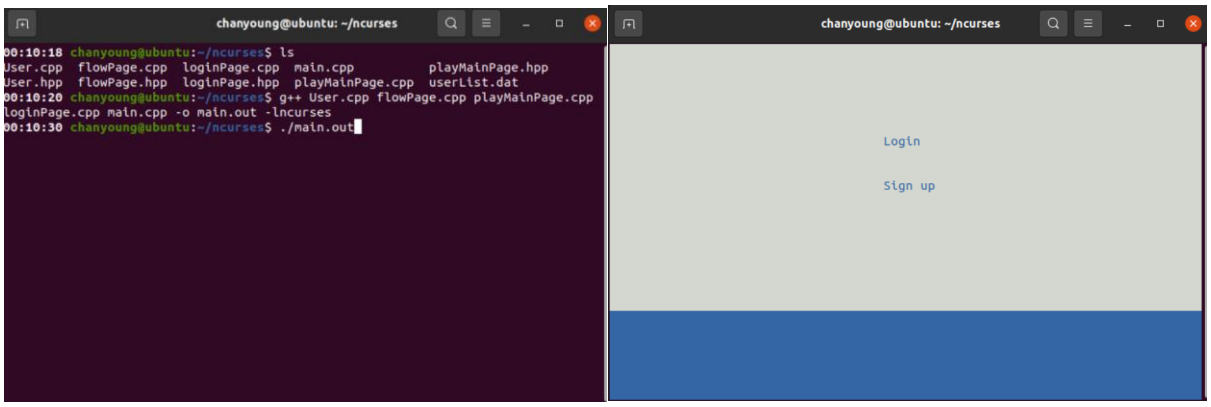
KWANGWOON UNIVERSITY

담당교수	김용혁 교수님	학 번	2021203034
학 과	소프트웨어학부	이 름	허찬영

KWANGWOON UNIVERSITY

메신저 프로그램(Project_01)

□ 컴파일 및 실행



"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.

□ 소스 코드

(소스 코드 사진은 미완성 코드로 키보드 오류가 있어 임시로 숫자4(52)를 아래 방향키, 숫자6(54)를 윗 방향키로 설정했습니다)

```
int main(int argc, char const *argv[]) {
    // 라이브러리를 초기화하고, 전체 화면을 나타나는 창 객체를 반환함.
    initscr();

    // terminal에서 색상 사용 가능 여부를 확인하고 가능 하지 않으면 오류 반환
    if (has_colors() == FALSE) {
        puts("Terminal does not support colors!");
        endwin();
        return -1;
    } else {
        // 색상을 사용하기 위해 호출하는 함수.
        start_color();
        // 색상 쌍을 정의
        // 인자 (변경할 색상 쌍 번호, 나타나는 색상, 배경 색상)
        init_pair(1, COLOR_BLUE, COLOR_WHITE);
        init_pair(2, COLOR_WHITE, COLOR_BLUE);
    }
    // 화면을 갱신하는 함수
    refresh();
```

main 함수에선 ncurses 라이브러리에서 제공하는 함수를 이용하여 라이브러리를 초기화하고 전체 화면을 나타나는 창 객체를 반환한다. 그리고 기본적인 색상을 정의하고 화면을 갱신한다.

```
mainPage();
mainMoveKey = getch();
if (mainMoveKey == 27)
    break;
else if (mainMoveKey == KEY_UP || mainMoveKey == KEY_DOWN) {
    checkInputESC = startPage(mainMoveKey);
}

// startPage() 함수에서 esc 키를 입력하면 함수 종료
if (checkInputESC == 1)
    break;

// Window를 지우는 함수. clear() 호출 이후, refresh()를 사용해야함.
clear();
// 화면의 창을 갱신(화면의 출력을 갱신하기 위한 필수 함수)
refresh();
```

기본적인 창을 초기화하고 mainPage() (다음 페이지에 자세히 설명)를 호출하여 기본적인 화면을 출력한다. 그 후, getch() 함수를 통해 키보드로부터 문자 하나를 입력 받고 방향키가 입력될 경우 startPage() 함수를 호출하고 esc키가 입력될 경우 반복문에서 빠져 나온다. startPage()의 반환값이 1일 경우 반복문에서 빠져 나온다. 반복문에서 빠져 나오면 endwin() 함수를 호출하여 라이브러리를 해제한 뒤 프로그램을 종료한다.

```
void mainPage() {
    curs_set(0);
    // 새로운 window를 할당
    WINDOW *win1 = newwin(24, 80, 0, 0);
    WINDOW *win2 = newwin(6, 80, 18, 0);

    // window에 색상 쌍을 입력
    // (window, COLOR_PAIR(color_pair_number))
    wbkgd(win1, COLOR_PAIR(1));
    wbkgd(win2, COLOR_PAIR(2));

    // 특정 window의 특정 위치에 문자열 출력
    mvwprintw(win1, 6, 37, "Login");
    mvwprintw(win1, 9, 37, "Sign up");

    // 특정 window를 출력
    wrefresh(win1);
    wrefresh(win2);
}
```

다음은 flowPage.cpp 파일에 있는 mainPage()함수다. flowPage.cpp와 flowPage.hpp엔 프로그램의 기본적인 실행 흐름과 시작 화면, 회원가입 화면과 기능이 구현되어 있다. mainPage()는 프로그램의 시작 화면을 구현했다. 모든 화면을 구현하는 함수는 위와 비슷한 방식으로 구현했다.

```
// login, register 항목을 선택하고 이동시켜주는 함수(전체적인 프로그램의 실행
// 흐름을 관리함)
// precondition: main함수에서 getch로 입력 받은 키를 인자로 받음
// postcodition: esc 입력 시 1를 반환, 정상적으로 반환될 경우 0을 반환, 오류
// 발생 시 -1을 반환
int startPage(char);
```

다음은 startPage() 함수이다. 함수의 기능은 위의 주석과 동일하다.

```
// 항목 이동
switch (key) {
case 54: {
    if (flag == true) {
        mvwprintw(win1, 6, 35, "> ");
    } else if (flag == false)
        mvwprintw(win1, 9, 35, "> ");
    break;
}
case 52: {
    if (flag == true) {
        mvwprintw(win1, 6, 35, "> ");
    } else if (flag == false)
        mvwprintw(win1, 9, 35, "> ");
    break;
}
// 항목 선택 및 실행
case 10: {
```

(54와 52는 키보드 입력 오류가 생겨 임시로 작성한 코드, 키보드 숫자 6과 숫자 4를 의미)

startPage() 함수는 do-while문을 사용해서 esc키가 입력될 때까지 반복된다. 그리고 매개 변수로 받은 key와 flag를 이용하여 항목 이동을 한다. 예를 들어 아래 방향키(52)를 입력하고 flag가 false일 경우 회원가입 항목이 선택 된다.

```
case 10: {
    if (flag == false) {
        do {
            checkLogin = loginPage();
        } while (checkLogin == 1);
        if (checkLogin != 27)
            playMainPage();
        mainPage();
        checkLogin = 0;
    }
```

```
if (flag == true) {
    int checkSignUp = registerPage();
    if (checkSignUp == 1) {
        do {
            checkLogin = loginPage();
        } while (checkLogin == 1);
        if (checkLogin != 27)
            playMainPage();
    }
    mainPage();
    checkLogin = 0;
}
break;
```

특정 항목을 선택하고 엔터 키를 입력할 경우 특정 항목으로 이동한다.

flag가 false일 경우 login 화면으로 이동한다. loginPage()의 값이 1이 반환될 때까지(정상적으로 로그인 될 때까지) 반복되고 정상적으로 로그인이 되면 playMainPage() 함수를 호출하여 메인 화면으로 이동한다. 메인 화면에서 동작이 끝나면 mainPage() 함수를 호출하여 시작화면으로 이동한다.

flag가 true일 경우 Sign Up 화면으로 이동한다. registerPage() 함수를 호출하여 회원가입 화면으로 이동하고 정상적으로 회원가입이 성공되면 로그인 화면으로 이동한다. 사용자가 로그인을 정상적으로 하면 메인 화면으로 이동 한 뒤, 메인 화면의 동작이 끝나면 시작화면으로 이동한다.

```
// 회원가입 화면을 구현하는 함수
// postcondition: 정상적으로 회원가입 성공 시 1반환, esc 입력시 0 반환
int registerPage();
```

다음은 registerPage()이다. 함수의 기능은 위의 사진과 동일하다.

```
// cursor setting
curs_set(1);
move(6, 40);

std::string tmpName = "";
std::string tmpId = "";
std::string tmpPasswd = "";
```

registerPage() 함수는 curs_set()함수와 move()함수를 통해 커서의 위치를 설정한다.

그리고 name, id, passwd를 저장할 string 타입의 변수를 선언한다.

```
switch (arrowKeyValue) {
case 0: {
    curs_set(1);
    move(6, 40 + nCnt);

    // 이름 입력
    while (true) {
        iptUserInfo = getch();

        // Key Exception
        if (iptUserInfo == 27) {
            return 0;
        } else if (iptUserInfo == 52) {
            arrowKeyValue = 1;
            break;
        } else if (iptUserInfo == 54) {
            arrowKeyValue = 3;
            break;
```

```
if (iCnt < 10) {
    mvwprintw(win1, 8, 38 + iCnt, "%c", iptUserInfo);
    wrefresh(win1);

    tmpId.push_back(iptUserInfo);

    iCnt++;
}
```

do-while문을 사용하여 registerButtonPage가 1을 반환(정상적으로 회원가입이 안됨)하지 않을 때까지 반복한다. arrowKeyValue 변수를 이용하여 항목을 이동한다. case 0, 1, 2는 startPage()에 구현한 방식과 비슷하다. getch() 함수로 입력 받은 문자를 윈도우에 출력하고 push_back 함수를 통해 앞에서 정의한 string 변수에 Name, Id, Passwd를 저장한다. 그리고 Name, Id, Passwd는 최대 10개의 문자만 입력할 수 있도록 구현했다.

```
case 3:
    curs_set(0);
    mvwprintw(win1, 14, 35, "> ");
    wrefresh(win1);

    User userInfo(tmpName, tmpId, tmpPasswd);
    int cnt = 0;

    iptUserInfo = getch();
    if (iptUserInfo == 52) {
        arrowKeyValue = 0;
    } else if (iptUserInfo == 54) {
        arrowKeyValue = 2;
```

```
} else if (iptUserInfo == 10) {
    // checkSignUp이 1이면 반복, 0이면 종료
    checkSignUp = registerButtonPage(userInfo, nCnt, iCnt, pCnt);
    if (checkSignUp == 1) {
        nCnt = 0, iCnt = 0, pCnt = 0, arrowKeyValue = 0;
        char iptUserInfo = '\0';

        // Window Setting
        WINDOW *win1 = newwin(18, 80, 0, 0);
        WINDOW *win2 = newwin(6, 80, 18, 0);

        wbkgd(win1, COLOR_PAIR(1));
        wbkgd(win2, COLOR_PAIR(2));

        mvwprintw(win1, 6, 33, "name: ");
        mvwprintw(win1, 8, 33, "ID: ");
        mvwprintw(win1, 10, 33, "PW: ");
        mvwprintw(win1, 14, 37, "Sign up");
```

arrowKeyValue의 값이 3일 경우 Sign UP 버튼을 표시하는 것으로 커서를 화면에 표시하지 않는다. 그리고 앞에서 정의한 string 변수를 User Class 변수에 저장한다. 그 후 엔터 키가 입력되면 registerButtonPage()를 호출하여 회원가입을 위한 조건을 검사한다. 정상적으로 회원가입이 될 경우 0을 반환하고 정상적으로 회원가입이 되지 않을 경우(심각한 에러는 -1 반환 후 프로그램 종

료) 1을 반환 한 뒤 계속 do -while문 루프를 돈다.

```
// User에 입력된 ID와 같은 ID가 파일에 있는지 확인하는 함수
    (파일입출력을 활용함)
// precondition: User 변수, (int)문자의
// 길이를 받음 postcondition: 오류 발생 시 -1 반환, 예외
    처리 시 1 반환,
// 정상적으로 종료할 경우 0반환
int registerButtonPage(User, int, int, int);
```

다음은 registerButtonPage이다. User Class 변수와 int 세 개를 매개변수로 받는다.

```
if (nCnt == 0 || iCnt == 0 || pCnt == 0) {
    mvwprintw(win2, 0, 25, "Please fill out all blanks");
    wrefresh(win2);

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

nCnt(name에 저장된 문자의 수), iCnt(id에 저장된 문자의 수), pCnt(패스워드에 저장된 문자의 수) 중 하나라도 0일 경우 에러 메시지를 출력하고 1을 반환한다.

```
int fd = open(filepath.c_str(), O_RDWR);
if (fd > 0) {
    while (true) {
        int rSize = read(fd, &userReadInfo, sizeof(User));
        if (rSize == 0)
            break;
        else if (rSize < 0)
            break;
        std::string name = userReadInfo.getName();
        std::string id = userReadInfo.getId();
        std::string passwd = userReadInfo.getPasswd();

        if (id == cid) {
            uniqueId++;
        }
    }
    close(fd);
```

```
if (uniqueId > 0) {
    mvwprintw(win2, 0, 25, "Already exists ID");
    wrefresh(win2);

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

회원가입 창에 빈 칸이 없을 경우 open 함수를 통해 해당 경로에 있는 파일을 읽고 쓰기 권한으로 오픈한다. 그리고 read 함수를 통해 파일에 저장된 데이터를 읽고 저장한다. 만약 사용자가 입력한 id가 파일에 존재하면 에러 메시지를 출력하고 1을 반환한다.

```
fd = open(filepath.c_str(), O_CREAT | O_APPEND | O_WRONLY, 0644);
if (fd == -1) {
    perror("open() error!");
    return -1;
}

if (write(fd, &userInfo, sizeof(User)) == -1) {
    perror("write() error!");
    return -1;
}
close(fd);

return 0;
```

사용자가 정상적으로 사용자 정보를 입력하면 open 함수를 통해 파일을 다시 열고 write 함수를 통해 사용자의 정보를 파일에 저장한다. 그 후 0을 반환한다.

```
extern loginUser lUser;

// 로그인 화면을 구현
// postcondition: 정상적으로 종료될 경우 0을 반환, 로그인 실패 시 1 반환, esc
// 입력시 27을 반환
int loginPage();

// User에 입력된 ID와 같은 ID가 파일에 있는지 확인하고 있으면 패스워드가
// 동일하는지 확인하고 정수를 반환하는 함수(파일입출력을 활용함)
// precondiotion: 사용자가 입력한 User 변수를 매개변수로 받음
// postcondition: 정상적으로 입력할 경우 1을 반환, 예외 발생 시 0을 반환
int loginButtonPage(User);
```

```
// 로그인된 정보를 저장하는 변수
loginUser lUser("", "", "");
```

다음은 loginPage.cpp에 구현한 함수이다. loginPage() 함수와 loginButtonPage() 함수는 로그인을 구현한 함수로 자세한 정보는 위의 주석에 표기 되어 있다. 그리고 loginUser class 전역 변수를 사용하여 로그인한 사용자의 정보를 저장한다.

```
// 정상적으로 비밀번호를 입력했는지 확인
if (id == cid) {
    uniqueId++;
    if (cpw == passwd) {
        lUser.setLoginName(name);
        lUser.setLoginId(id);
        lUser.setLoginPasswd(passwd);
        close(fd);
        return 1;
    } else {
        mvwprintw(win2, 0, 25, "Incorrect Password");
        wrefresh(win2);

        key = getch();
        return 0;
    }
```

```
mvwprintw(win2, 0, 25, "ID doesn't exist");
wrefresh(win2);

key = getch();
return 0;
```

loginPage()의 기본적인 작동 방식은 registerPage()와 거의 동일하다. 단지 출력되는 문자와 기능이 조금 다를 뿐이다. loginButtonPage()는 로그인 조건을 만족하는지 확인 하는 함수다. 사용자가

입력한 아이디가 파일에 있다면 해당 아이디와 같이 저장된 패스워드를 찾는다. 만약 사용자가 입력한 패스워드가 파일에 있는 패스워드와 동일하다면 로그인이 정상적으로 된 것이고 1을 반환한다. 패스워드가 동일하지 않거나 존재하지 않는 ID일 경우 0을 반환한다.

```
std::string id = lUser.getLoginId();
for (int i = 0; i < id.length(); i++) {
    cid[i] = id[i];
}

std::string name = lUser.getLoginName();
for (int i = 0; i < name.length(); i++) {
    cname[i] = name[i];
}

for (cnt = 0; cnt < id.length(); cnt++) {
    mvwprintw(win2, 0, 4 + cnt, "%c", cid[cnt]);
}
mvwprintw(win2, 0, 73, "Logout");
mvwprintw(win2, 1, 0, "Name: ");
for (cnt = 0; cnt < name.length(); cnt++) {
    mvwprintw(win2, 1, 6 + cnt, "%c", cname[cnt]);
}
```

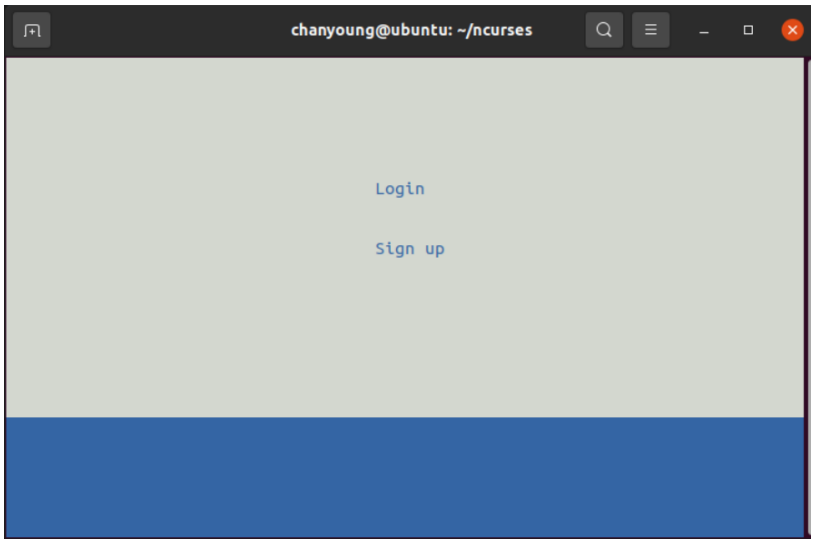
다음은 playMainPage.cpp에 있는 playMainPage() 함수이다. playMainPage 함수는 메인 화면을 출력하는 기능을 한다. 함수의 기본적인 동작은 앞에서 설명한 함수들과 거의 동일하다. 사용자가 로그인한 정보가 저장된 lUser 전역 변수의 정보를 char형으로 저장한다. 그리고 char형으로 저장된 ID와 Name을 출력한다.

```
if (iptMove == 27) {
    mvwprintw(win1, 0, 0, "  ");
    mvwprintw(win1, 1, 0, "  ");
    mvwprintw(win1, 2, 0, "  ");
    mvwprintw(win1, 3, 0, "  ");
    mvwprintw(win2, 0, 71, ">");
    wrefresh(win1);
    wrefresh(win2);
    iptMove = getch();

    if (iptMove == 27)
        arrowKeyValue = 0;
    else if (iptMove == 10)
        ESCFlag = true;
}
```

함수 진행 도중 esc가 입력되면 한 번 더 입력을 받는다. 만약 다시 esc가 입력 될 경우 계속해서 반복문의 루프를 돌고 엔터 키가 입력될 경우 함수를 반환한다.

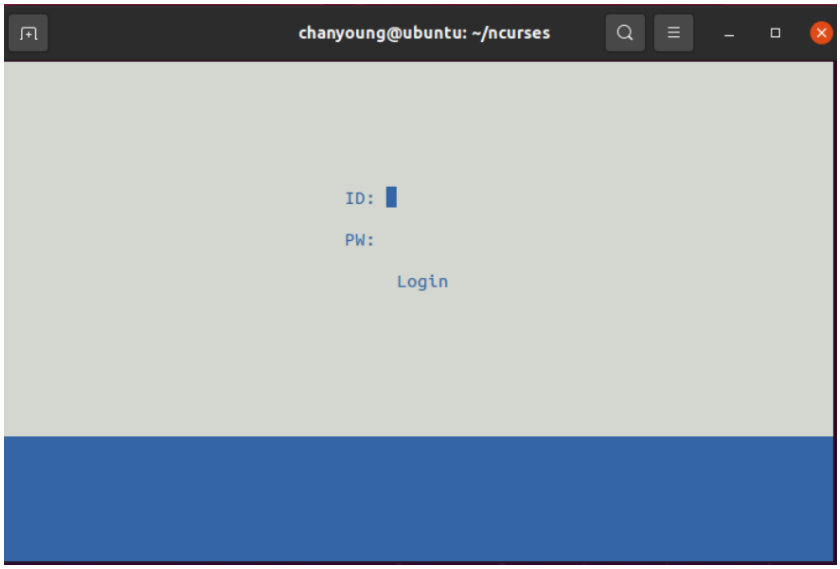
```
class User {
  public:
    User();
    User(std::string name, std::string id, std::string passwd);

    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);

  private:
    char name[11];
    char id[11];
    char passwd[11];
};
```

□ 프로그램 실행 결과



(시작화면)

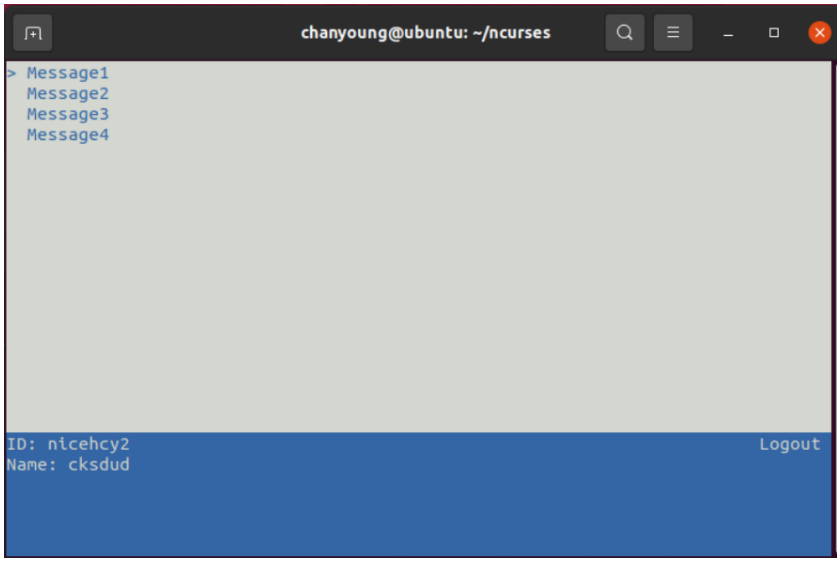
방향키를 통해 항목을 이동하고 엔터 키를 통해 항목을 선택할 수 있다.



(로그인 화면)

ID와 패스워드를 입력 하고 정상적으로 입력될 경우 로그인을 한다. ID가 존재하지 않거나 패스워드가 일치하지 않을 경우 에러 메시지를 반환하고 로그인이 되지 않는다. 이 후 아무 키를 입력하면 다시 새로 로그인을 할 수 있다.

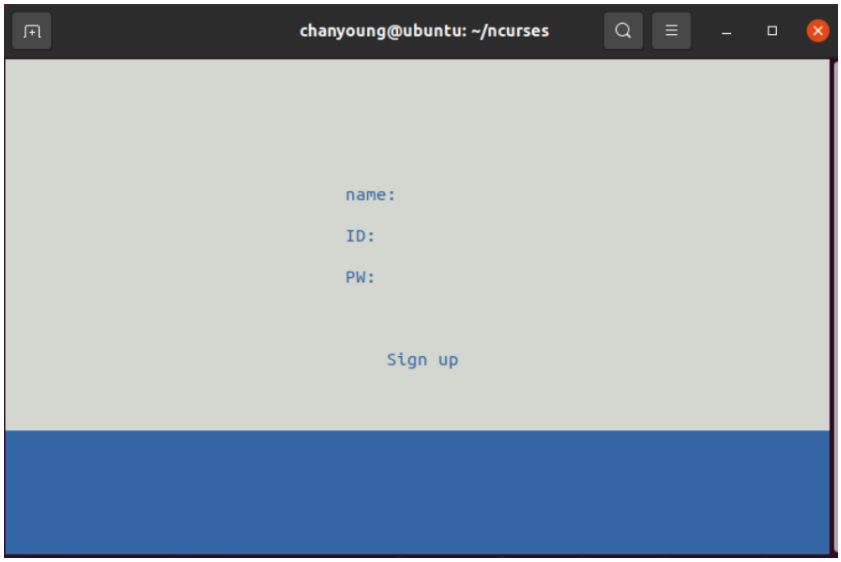
(esc키를 입력하고 시작화면으로 돌아가기 전까지 조금의 딜레이가 있다.)



(메인 화면)

메인 화면이다. 아래 칸에 사용자의 정보가 출력된다. 아직 아무런 기능이 없으며 방향키를 통해 항목을 이동할 수는 있다. ESC키를 입력하면 Logout이 선택되며 다시 ESC키를 누르면 Message 칸으로 이동하고 엔터 키를 입력하면 로그 아웃 되며 시작 화면으로 이동한다.

(esc를 입력하고 화면에 출력되기까지 조금의 딜레이가 있다.)



(회원가입 화면)

회원 가입 화면이다. 정상적으로 입력할 경우 회원가입이 되며 하나라도 입력하지 않을 경우 에러 메시지를 반환한다. 그리고 이미 존재하고 있는 아이디가 있을 경우 에러 메시지를 반환한다. 에러 메시지가 반환된 경우 회원가입이 되지 않으며 아무 키를 누르면 다시 회원가입 정보를 작성할 수 있다.

(회원 가입을 하지 않고 로그인을 하면 에러 메시지가 출력된다)

□ 고찰 리포트

질문 사항:

- ~~방향키를 입력해도 터미널이 종료되진 않았으나 코드가 방향키를 인식하지 못했다. 방향키 이외에도 F1, Home 키와 같은 다른 특수키도 인식하지 못했다~~.

  (해결되었다. 특수키의 값이 char형이 담을 수 있는 값을 넘어가서 오류가 생긴 거였다.)

- 파일 입출력 과정에서 Class의 private 변수를 string 타입으로 작성하면 파일에 정상적으로 값이 저장되지 못했다. 그러나 char 배열 타입으로 작성할 경우 파일에 정상적으로 값이 저장되고 읽을 수 있었다.

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. 그리고 처음 코드를 작성할 때 Incurses 라이브러리를 제대로 이해하지 못해 프로그램을 제작하는데 많은 어려움을 겪었다.

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.