

자료구조실습 프로젝트 최종 보고서

2022203017 이종훈

1. 프로젝트 명 : 종훈의 게임팩

리눅스 환경에서 ncurses를 사용하여 만든 게임팩이다.

2. 프로젝트 주제에 대한 간단한 설명

Ncurses 라이브러리를 이용한 게임팩이다. 회원 가입과 로그인 시스템을 갖춘 게임 프로그램으로, 2개의 게임이 포함되어 있다. wordle 게임과 snake 게임을 구현하였다.

3. 제안서와 달라진 점과 달라진 이유에 대한 설명

큰 주제 자체는 달라지지 않았다. 게임팩을 만들었고, 회원 가입과 로그인 시스템을 갖추었다. 게임의 종류는 2개를 만들었고, 기존 제안서에 있던 wordle 게임과 제안서에는 없었던 snake 게임을 만들었다.

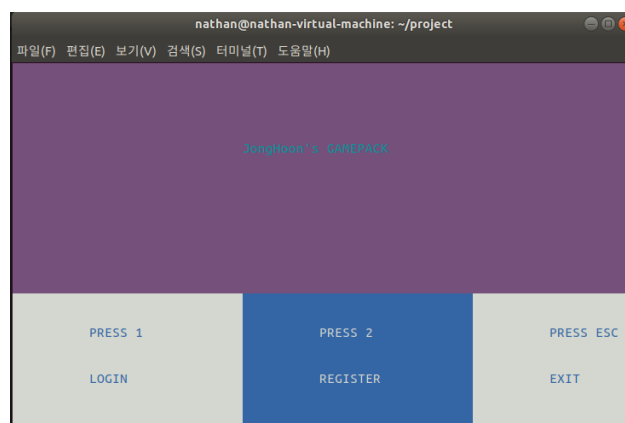
크게 달라진 점은 게임의 종류에 대해서 인데, 제안서에서는 백준 알고리즘 문제 응용과 끝말잇기를 만들 것 같다고 했었다. 하지만 포기한 이유가 있다. 우선 처음으로 끝말잇기는 원래 shared memory같은 IPC 자원을 사용해서 만들려고 했었다. 즉, 2개의 프로세스를 실행하여 만들어야 했는데, 이미 확정된 wordle과 다른 게임들은 전부 다 싱글 플레이인데, 혼자 멀티플레이를 구현하면 편의성 및 일관성을 해친다고 생각하여 제외했고, 백준 알고리즘 문제는 막상 만들어보니 (파이프 자르기 퀴즈) 너무 재미가 없고 게임으로서의 역할을 하지 못하여서 제외했다. 그리고 snake 게임을 대체로 만든 이유는 우선 구현 난이도가 상당히 도전심을 불러일으켰고, 무엇보다 자료구조를 이용해 만들 수 있는 게임 중 제일 흥미로워 보였기 때문에 대체 게임으로 선택하였다.

4. 구현 방법, 사용기술에 대한 설명 및 프로젝트 구현 결과 사진

구현한 것이 꽤 많아 단계별로 나누어서 설명하도록 하겠다. ncurses 인터페이스, 로그인 및 회원가입 시스템, wordle 게임, 뱀 게임이 있다.

(1) ncurses 인터페이스

우선 제일 간단한 ncurses 인터페이스부터 설명해보도록 하겠다. 먼저 실행 시의 화면인데, JongHoon's GAMEPACK이라고 적힌 인터페이스와 1을 누르면 로그인, 2를 누르면 회원가입, ESC를



누르면 EXIT이라고 적힌 화면이 보인다.

이 화면은 간단하게 NCURSES 라이브러리를 통해 구현했다. 뼈대를 만든 함수 mainPage() 함수, 그리고 클릭을 통해 다른 곳으로 이동해주는 함수인 selectMain()까지 말이다. 밑은 그 코드 사진들이다.

```
void selectMain() {
    int choice;

    while (mainBool) {
        int c = getch();

        switch (c) {
            case 49:
                choice = 0;
                break;
            case 50:
                choice = 1;
                break;
            case 27:
                choice = 2;
            }

            if (choice == 0) {
                loginPage();
            } else if (choice == 1) {
                registerPage();
            } else if (choice == 2) {
                endwin();
                exit(0);
            }

            choice = -1;
        }
    }
}
```

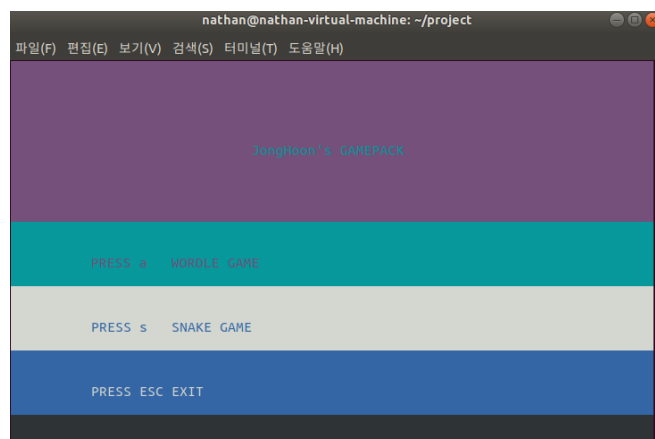
```
void mainPage() {
    WINDOW *win1 = newwin(15, 80, 0, 0);
    WINDOW *win2 = newwin(9, 30, 15, 0);
    WINDOW *win3 = newwin(9, 30, 15, 30);
    WINDOW *win4 = newwin(9, 30, 15, 60);

    wbkgd(win1, COLOR_PAIR(1));
    wbkgd(win2, COLOR_PAIR(3));
    wbkgd(win3, COLOR_PAIR(4));
    wbkgd(win4, COLOR_PAIR(3));

    mvwprintw(win1, 5, 30, "JongHoon's GAMEPACK");
    mvwprintw(win2, 5, 10, "LOGIN");
    mvwprintw(win3, 5, 10, "REGISTER");
    mvwprintw(win4, 5, 10, "EXIT");
    mvwprintw(win2, 2, 10, "PRESS 1");
    mvwprintw(win3, 2, 10, "PRESS 2");
    mvwprintw(win4, 2, 10, "PRESS ESC");

    wrefresh(win1);
    wrefresh(win2);
    wrefresh(win3);
    wrefresh(win4);
}
```

또한 밑의 사진은 로그인 이후 나오는 사진인데, 구현 역시 위와 비슷한 방식으로 했기 때문에 따로 코드 사진은 첨부하지 않도록 하겠다.

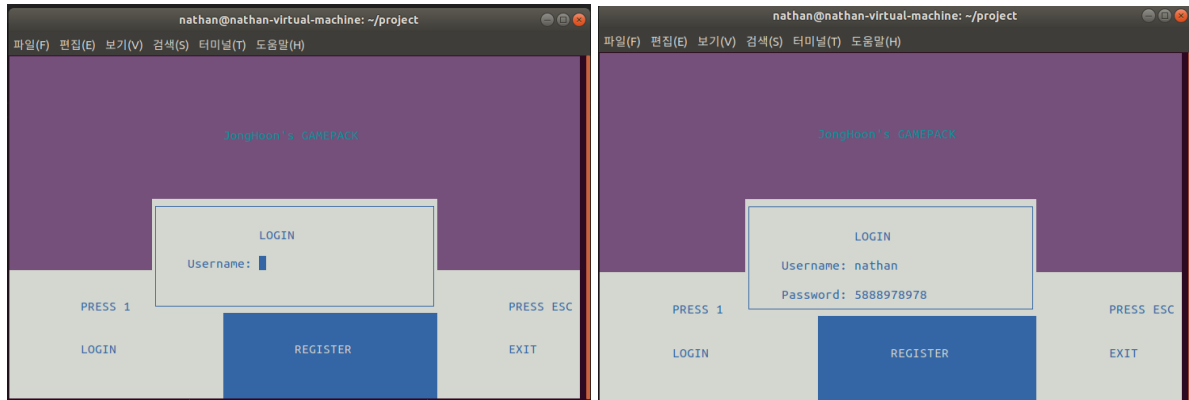


이런 방식으로 시각적으로 도움이 되는 ncurses 인터페이스를 구현하였다.

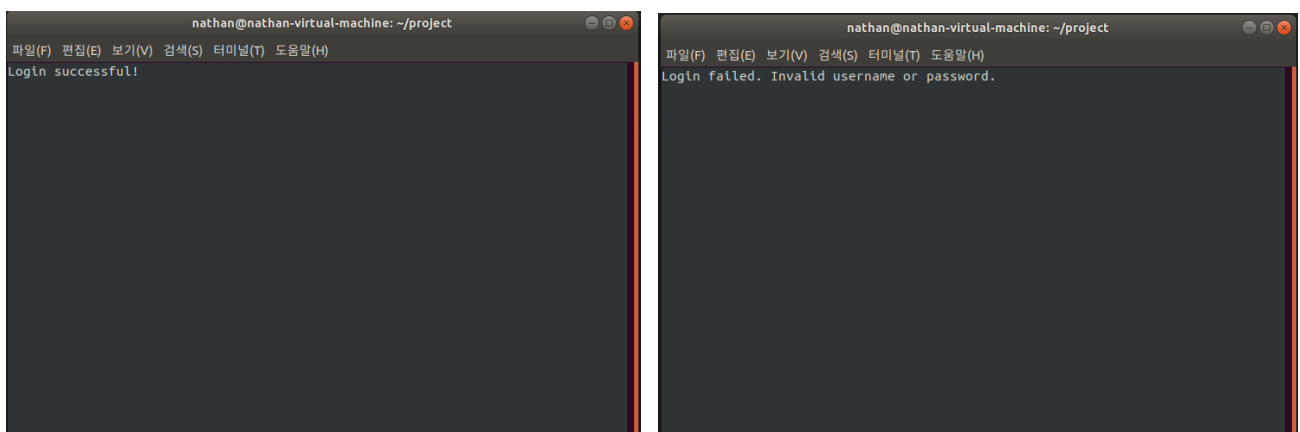
사용된 기술 : ncurses

(2) 로그인 및 회원가입 시스템

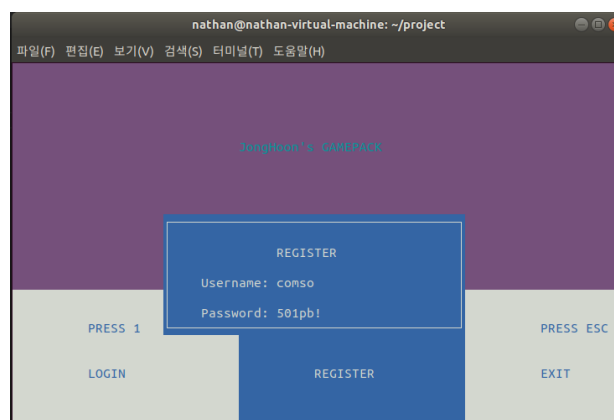
기능 설명 이후, 코드와 함께 구현 방법을 하도록 하겠다. 맨 처음 프로그램 실행 시의 사진을 보면 로그인, 회원가입, 나가기 메뉴가 있는 것을 볼 수 있다. 여기서 키보드에서 숫자 1 키패드를 누르면 이런 창이 나온다. 아이디를 입력하면 비밀번호 입력창도 나오는 것을 볼 수 있다.



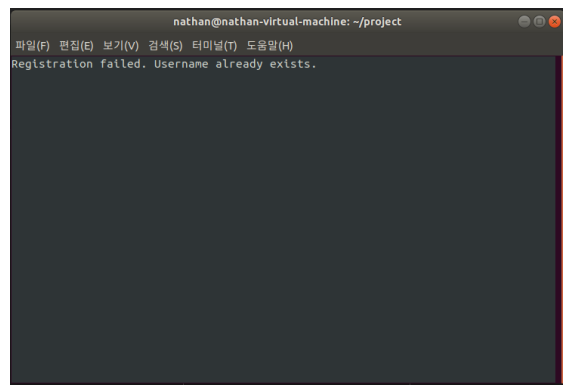
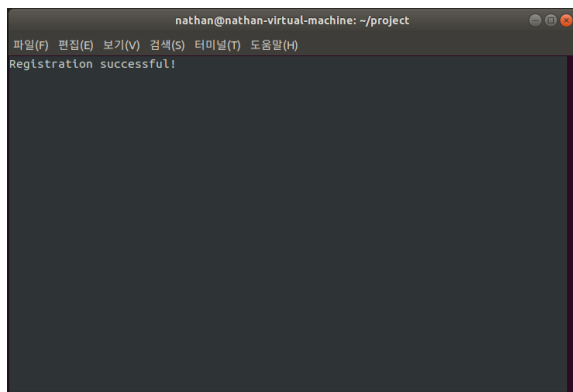
만약 로그인에 성공했다면 로그인에 성공했다는 메시지가 나오고, 실패했다면 실패했다는 메시지가 나온다. 밑의 사진처럼 말이다.



로그인에 성공하려면 우선 아이디와 비밀번호가 있어야 하는데, 그건 회원가입 (키패드 2번)을 누르면 할 수 있다.



로그인과 같은 방식으로 작동되고, 회원가입에 성공하면 성공했다고, 실패하면 (이미 있는 아이디), 실패했다고 나온다. 밑의 사진처럼 말이다.

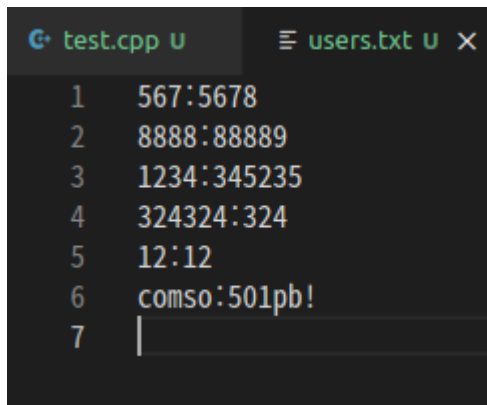


이제부터 구현 방법에 대해 설명하도록 하겠다.

이렇게 작동이 가능하게 만드는 기술은 바로 파일 입출력이다. 물론 `fstream`을 사용하지 않았다. 모든 코드를 보고서에 넣기에는 아무래도 무리가 있어 핵심 함수만 사진을 첨부하도록 하겠다.

```
if (dprintf(fileDescriptor, "%s:%s\n", username, password) == -1) {  
    cerr << "Error writing to file for user registration.\n";  
    close(fileDescriptor);  
    return false;  
}
```

위 사진은 `addUser()` 함수의 일부분인데, `users.txt`라는 파일에 정보를 아이디:비밀번호 식으로 적어 놓았다. 회원가입 설명 사진의 회원가입에 사용된 `comso, 501pb!`가 그대로 들어가 있는 것을 볼 수 있다. 이렇게 회원 가입을 구현하였다.



```
if (addUser(username, password)) {  
    clear();  
    printf("Registration successful!\n");  
    refresh();  
    getch();  
}
```

이제 로그인을 설명할 차례인데, `authenticateUser()`라는 함수를 이용하여 실제로 이 아이디와 비밀번호가 존재하는지 확인하고, 그 다음 단계로 넘어갈 수 있게 해주는 기능을 가지고 있다.

```
if (authenticateUser(username, password)) {  
    clear();  
    printf("Login successful!\n");  
    refresh();  
    getch();  
    delwin(loginWin);  
    mainBool = false;  
    gameSelectPage();  
}
```

```

while ((bytesRead = read(fileDescriptor, buffer, sizeof(buffer))) > 0) {
    char *line = strtok(buffer, "\n");

    while (line != nullptr) {

        char *colonPos = strchr(line, ':');
        if (colonPos != nullptr) {
            *colonPos = '\0';
            string userID = line;
            string password1 = colonPos + 1;

            if (userID == username && password1 == password) {
                close(fileDescriptor);
                return true;
            }
        }
        line = strtok(nullptr, "\n");
    }
}

```

사진은 이 user가 실제로 회원가입이 된 아이디를 가지고 있는지 확인하는 함수이다.

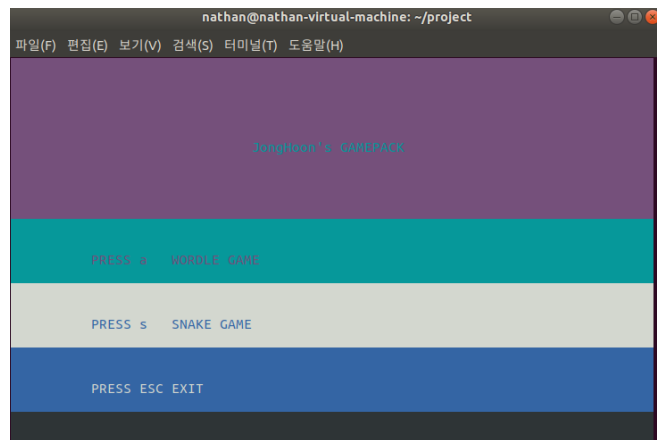
이렇게 파일 입출력 함수를 통해 로그인과 회원가입 기능을 구현하였다.

사용된 기술 : 시스템소프트웨어의 파일 입출력 함수들 (open, close, lseek 등등...)

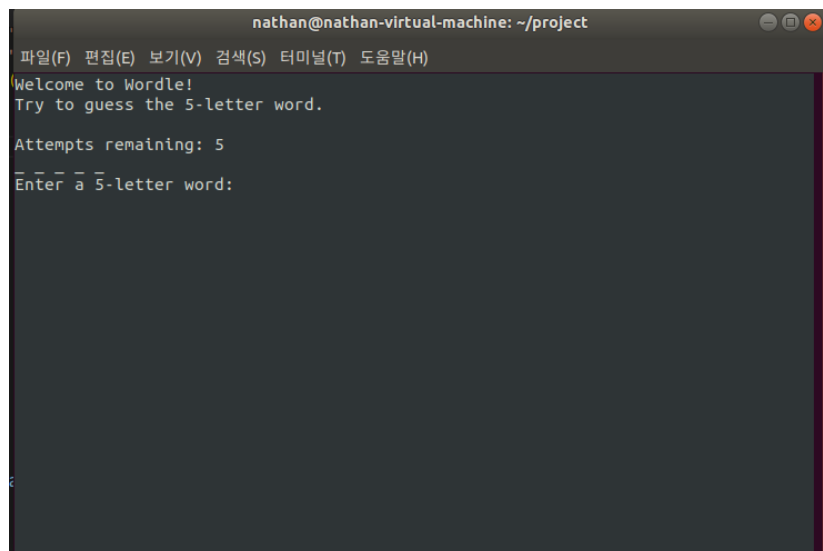
기술 사용 이유 : 로그인 시스템을 만들기에는 파일 입출력으로 하기가 제일 효율적이었기 때문.

(3) wordle 게임

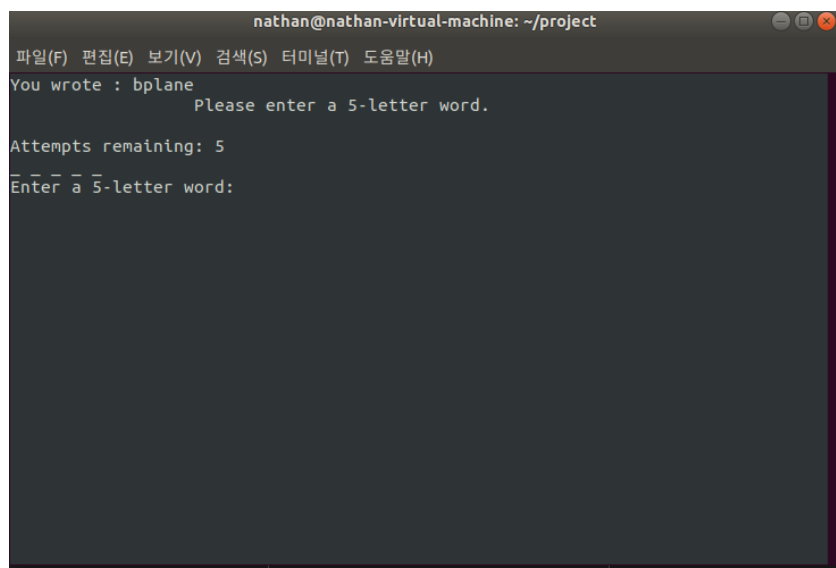
기능 설명 이후, 코드와 함께 구현 방법 설명을 하도록 하겠다.



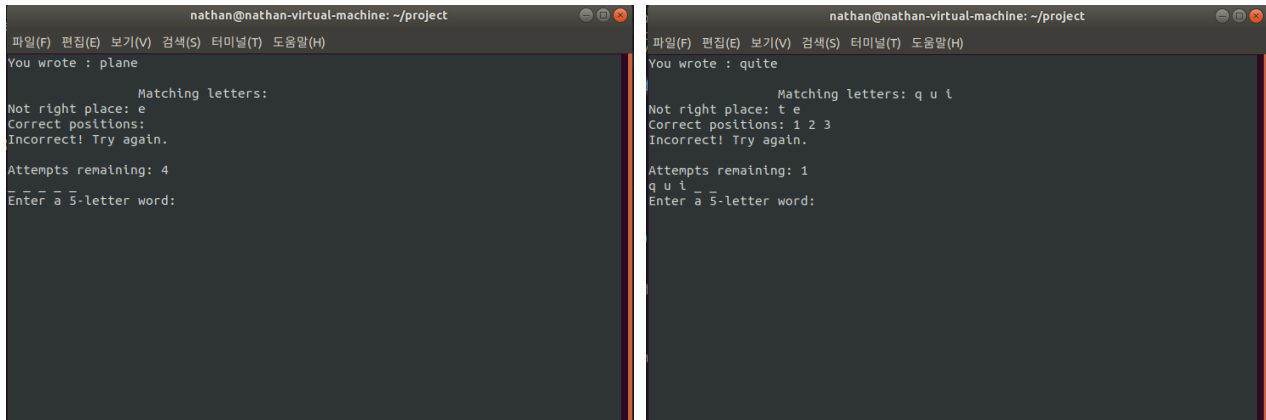
이 화면에서 a를 누르면 wordle game으로 이동이 되고, wordle game이 실행이 된다.



실행한 뒤의 화면이다. 다섯 글자를 누르라고 적혀있고, 기회는 총 5번이다. 5글자 글자를 맞추는 게임이다. 5글자가 아니면 도전 횟수가 줄어들지 않는다.



하지만 다섯 글자를 입력하게 되면 변화가 생긴다. 우리가 예상 정답을 입력하게 되면 다섯 가지 위치에는 각각 위치에 맞는 알파벳, 위치는 틀리지만 존재하는 알파벳, 그리고 존재하지 않는 알파벳이 있을 것이다. 여기서는 기존 워들처럼 색으로 표현은 못해주지만, 각각 정답인 알파벳은 2번째 줄의 Matching letters: 에서 알려주고, 알파벳은 존재하지만 위치가 틀렸으면 그 아래에 Not right place: 에 출력하고, 맞는 위치에 있는 알파벳들의 숫자를 Correct positions: 에 출력한다. 그리고 정답 유무를 다음줄에 알려주고, 그 다음 줄에는 남은 기회, 그 다음 줄에는 시각적으로 여태까지 완성한 단어의 모습을 보여준다.



```
nathan@nathan-virtual-machine: ~/project
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
You wrote : plane

      Matching letters:
Not right place: e
Correct positions: 1 2 3
Incorrect! Try again.

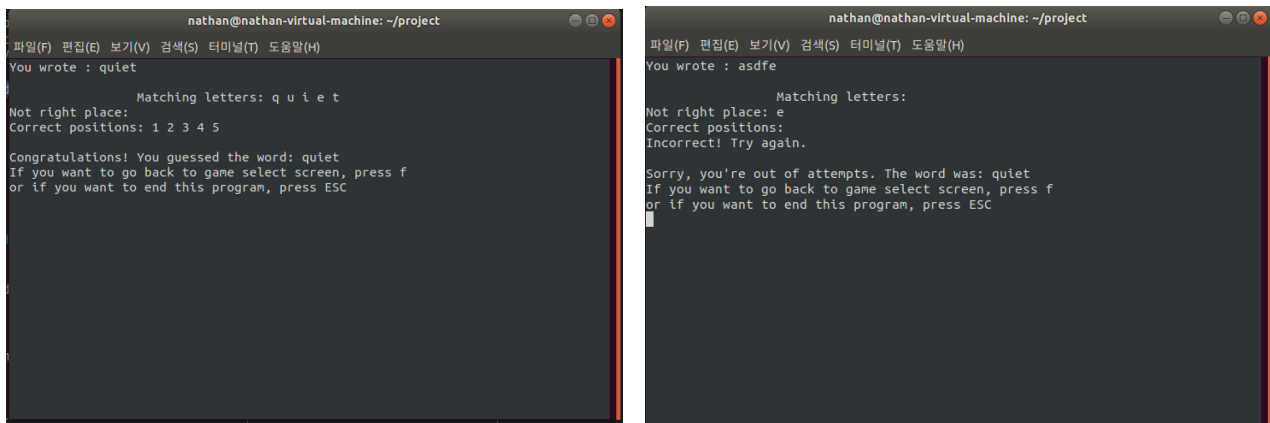
Attempts remaining: 4
Enter a 5-letter word:

nathan@nathan-virtual-machine: ~/project
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
You wrote : quite

      Matching letters: q u i
Not right place: t e
Correct positions: 1 2 3
Incorrect! Try again.

Attempts remaining: 1
q u i _
Enter a 5-letter word:
```

위의 사진들에서 알 수 있는데, 답은 quiet일 때이다. 왼쪽 위는 plane을 입력해 유일하게 맞는 알파벳이 e인데다가, 위치가 맞지 않을 때의 모습이고, 오른쪽 위는 quite를 입력했을 때의 모습이다. qui는 맞고, e t는 맞지 않기 때문에, 사진의 모습이 나온다.



```
nathan@nathan-virtual-machine: ~/project
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
You wrote : quiet

      Matching letters: q u i e t
Not right place:
Correct positions: 1 2 3 4 5

Congratulations! You guessed the word: quiet
If you want to go back to game select screen, press f
or if you want to end this program, press ESC

nathan@nathan-virtual-machine: ~/project
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
You wrote : asdfs

      Matching letters:
Not right place: e
Correct positions:
Incorrect! Try again.

Sorry, you're out of attempts. The word was: quiet
If you want to go back to game select screen, press f
or if you want to end this program, press ESC
```

정답일 때와 정답이 아닐 때의 출력 모습은 위 사진에서 확인할 수 있다. 왼쪽은 맞았을 때, 오른쪽은 틀렸을 때의 모습이다.

결과 창에서 f를 누르면 로그인 이후 맨 처음 화면으로 돌아가게 되고, ESC를 누르면 프로그램이 종료된다.

이제 구현 방법에 대해서 설명해 보도록 하겠다. 구현은 간단하다.

```

printw("\nMatching Letters: ");
for (size_t i = 0; i < 5; ++i) {
    if (guess[i] == secretWord[i]) {
        printw("%c ", guess[i]);
    }
}

printw("\nNot right place: ");
for (size_t i = 0; i < 5; ++i) {
    if (guess[i] != secretWord[i] &&
        secretWord.find(guess[i]) != string::npos) {
        printw("%c ", guess[i]);
    }
}

printw("\nCorrect positions: ");
for (size_t i = 0; i < 5; ++i) {
    if (guessedWord[i] == secretWord[i]) {
        printw("%d ", i + 1);
    }
}
}

```

5번의 시도 동안 각각 알파벳이 5개인 문자열을 받은 뒤, 정답과 각각 대조하여 알파벳의 유무 및 위치의 옳고 그름을 모두 판단 한 뒤 출력하는 식으로 구현하였다. 5글자 정답은 미리 코드 안에 10개를 넣어놓았고, 각 실행 때마다 랜덤으로 정해진다.

```
string chooseWord() {
    string words[] = {"apple", "blood", "plane", "grape", "melon",
                     "peach", "quiet", "table", "quite", "crime"};
    return words[rand() % (sizeof(words) / sizeof(words[0]))];
}
```

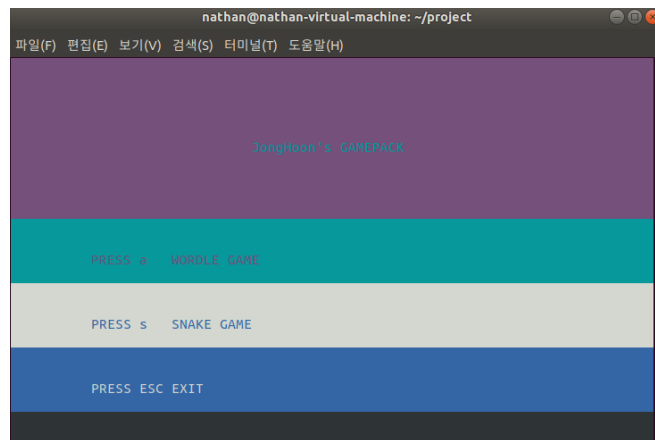
위 사진은 실제 코드에서의 모습이다.

사용된 기술 : ncurses

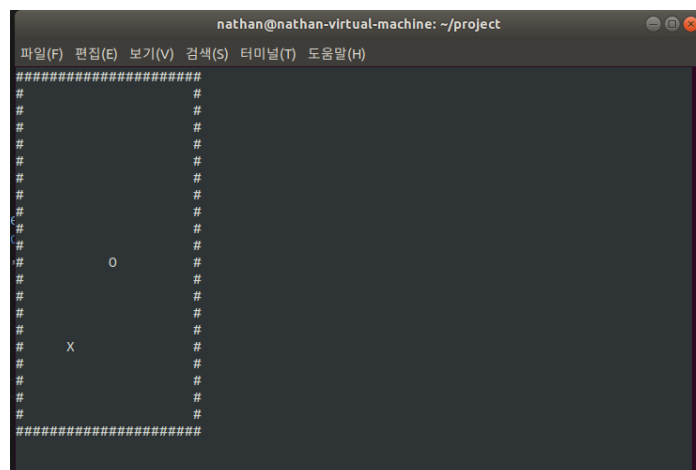
기술 사용 이유 : ncurses로 각각의 기회를 표시하는 것이 효율적이라고 생각했음

(4) snake 게임

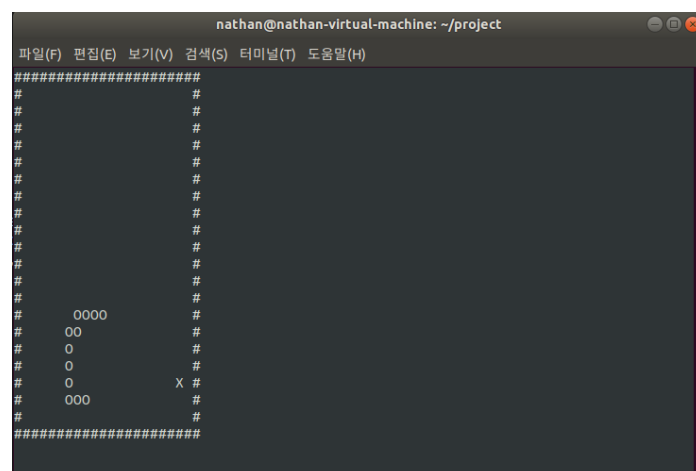
기능 설명 이후, 코드와 함께 구현 방법 설명을 하도록 하겠다.



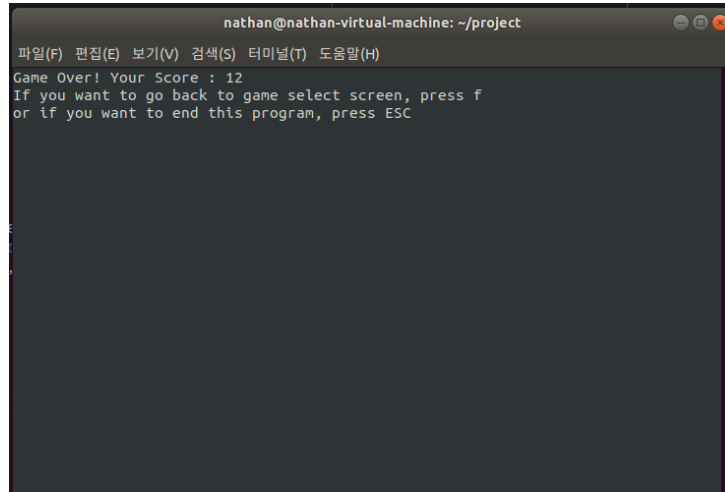
이 화면에서 키보드의 s키를 누르면 snake game 화면으로 이동된다.



X는 먹이, O는 스네이크로, 먹이를 먹을수록 뱀은 길이가 길어지며, 본인의 몸통이나 벽에 닿으면 게임 오버가 되고, 머리를 제외한 현재 길이가 점수로 치환되는 게임이다.



이런 식으로 먹이를 먹을수록 길어진 뱀의 모습을 볼 수 있다.
벽에 부딪히거나 본인의 몸에 부딪혀서 게임 오버가 될 경우,



위 사진처럼 점수와 f키를 눌러 메인화면으로 나갈건지, 아니면 ESC를 눌러 프로그램을 종료할 건지를 물어보는 화면이 나온다.

이제 구현 방법에 대해서 설명해 보도록 하겠다.

클래스 형태로 뱀 게임을 제작하였다. 2X2인 2차원 좌표에 대응할 doubly edged queue인 snake와 food 변수를 제작했다.

```
private:
    deque<pair<int, int>> snake;
    pair<int, int> food;
    int direction;
    bool gameOver;
```

그 뒤, displayBoard() 함수를 제작해 판을 만들고,

```
void displayBoard() {
    clear();

    for (auto segment : snake) {
        mvprintw(segment.first + 1, segment.second + 1, "O");
    }

    mvprintw(food.first + 1, food.second + 1, "X");

    for (int i = 0; i < BOARD_SIZE + 2; ++i) {
        mvprintw(0, i, "#");
        mvprintw(BOARD_SIZE + 1, i, "#");
    }

    for (int j = 0; j < BOARD_SIZE + 2; ++j) {
        mvprintw(j, 0, "#");
        mvprintw(j, BOARD_SIZE + 1, "#");
    }

    refresh();
}
```

spawnFood()함수로 랜덤 위치에 음식을 생성하며, checkCollision() 함수로 충돌을 감지시켰다.

```
void spawnFood() {
    srand(time(0));
    int row, col;

    do {
        row = rand() % (BOARD_SIZE - 2) + 1;
        col = rand() % (BOARD_SIZE - 2) + 1;
    } while (isSnakeSegment(row, col));

    food = {row, col};
}
```

```
void checkCollision() {
    if (snake.front().first == 0 || snake.front().first == BOARD_SIZE - 1 ||
        snake.front().second == 0 || snake.front().second == BOARD_SIZE - 1) {
        gameOver = true;
    }

    for (auto it = next(snake.begin()); it != snake.end(); ++it) {
        if (snake.front() == *it) {
            gameOver = true;
            break;
        }
    }
}
```

또한, 제일 중요한 부분인 뱀이 음식을 먹었을 때를 구현한 checkFood() 함수에서 queue를 사용하여 뱀의 길이를 잘 늘여놓았다.

```
void checkFood() {
    if (snake.front() == food) {
        snake.push_back({-1, -1});
        spawnFood();
    }
}

bool isSnakeSegment(int row, int col) {
    for (auto segment : snake) {
        if (segment.first == row && segment.second == col) {
            return true;
        }
    }
    return false;
}
```

사용된 기술 : *deque* (자료구조의 STL queue 사용하였음)

기술 사용 이유 : *snake*의 몸통을 구현하는데 있어서 *deque*가 필요하였음

5. 본인이 만든 프로젝트에 대한 본인의 의견

나름 만족스러웠던 프로젝트이다. ncurses를 제대로 사용해 본 것은 처음이었는데, 나름대로 학습을 통해 꽤 의미 있는 인터페이스를 만든 것 같아 기분이 좋았고, 코딩 실력이 한 단계 업그레이드 된 것 같아 좋았다. 하지만 아쉬운 점도 있었는데, 다 만들고 제출 직전에 생각나는 것이 여러가지 있었다. 예를 들어 로그인 기능에서 숫자와 알파벳을 무조건 하나 이상 씩은 들어가야 한다는 보안 문구 출력 같은 것 말이다. 기능의 부족이 조금 아쉬웠고. 게임 역시 wordle의 기능 자체는 전부 구현했지만, 실제 wordle은 맞는 부분은 초록색, 존재하지만 위치가 틀린 부분은 노란색으로 출력하는, 좀 더 가독성이 좋은 인터페이스를 차용하는데, 그런 것을 따라하지 못해서 조금 단점이 되었다고 생각한다. 하지만, 기존의 목표를 달성하는데 성공하였고, 이 정도 게임들에 로그인 기능은 실질적으로도 충분히 사용 가능하기 때문에, 앞에서 만족했다고 말한 것이다.

6. 사용 기술 정리

시스템프로그래밍 : 파일 입출력, 자료구조 : 큐 사용 (각 설명 밑에 사용 기술 및 이유가 상세히 적혀있음)

7. 결과 시연 영상 링크

<https://youtu.be/6lSYVyUqYV4>