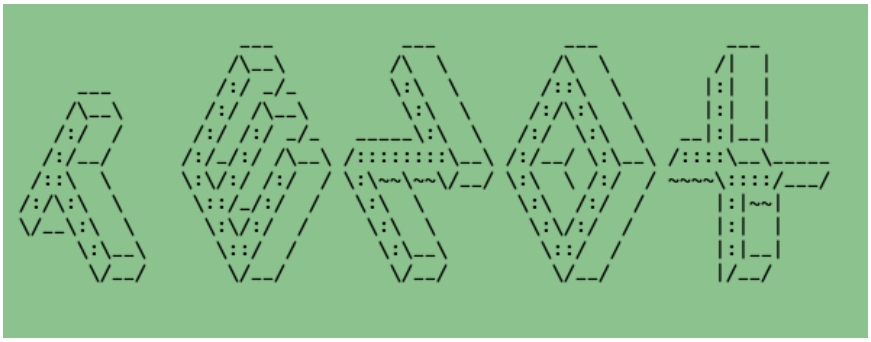
System Programming

TERM Project _report

컴퓨터학부 심화컴퓨터전공
2020114658 이윤서

I. 프로젝트 개요



숫자 10 을 나타내는 TEN 과 상자를 의미하는 BOX 의 합성어인 TENOX 는, 주어진 숫자들 위의 상자를 그려 모두 0 을 만드는 게임이다.
상자 안의 숫자들의 합이 10 이 되면 숫자들은 모두 0 이 된다. 이를 반복하며 각 모드에 따라 정해진 종료 조건에 도달하면 게임은 끝이 난다.
숫자들은 1 에서부터 9 까지, 총 9 개의 숫자가 각각 10 개씩 랜덤 순서로 10 * 9 배열로 배치된다.

각 모드에 따른 종료 조건은 다음과 같다.

+ ***MakeZERO*** 모든 숫자들을 0 으로 만들면 종료, 그때까지 걸린 시간을 카운트
+ ***TimeKILL*** 주어진 시간 (100 초) 동안 0 으로 만든 숫자들의 개수를 점수로 환산 (만점이 100 점)
  (시작할 때 숫자 총합 450 – 남은 숫자들의 합) * 100 /450

! 필수 옵션 모두 구현하였으며, 옵션 1 / 옵션 3 구현하였습니다. 각 기능이 구현된 부분은 **파란색 볼드체**로 표시해 두었습니다.

II. 상세 구조 및 동작

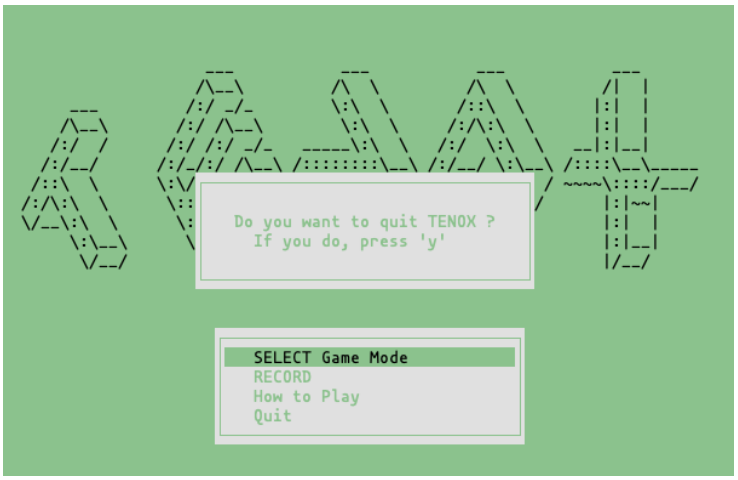
`./tenox.sh` 명령어를 통해 게임을 실행하면 다음과 같은 화면이 출력된다. (**쉘스크립트 구현**)

표준 윈도우인 `stdscr` 위에 게임 로고가 출력되고,
시작 버튼을 입력하라는 문구는 `WINDOW* start` 위에서 출력된다.

게임을 시작하기 위해 ENTER 키를 누르면 메뉴화면으로 넘어가고, ESC 나 q 키를 누르면 곧바로 종료된다.


```
void main() {

    /* Screen Setting */
    init_scr();
    bkgd(COLOR_PAIR(1));
    signal(SIGINT, check_quit);
```

또한, 이 화면에서부터 signal (timer signal 이외) 로 등록해 둔 `Ctrl + c (SIGINT)` 키가 입력되면
핸들러 함수인 `quit_check()` 가 호출되고 확인 화면이 뜬다.



여기서 y 키를 입력하면 곧바로 종료된다.

방향키로 각 메뉴를 조종할 수 있는데, 첫번째 메뉴인 SELECT Game Mode 를 선택하면 (선택은 ENTER 키) 각 게임 모드를 선택할 수 있고, 이 역시 방향키로 메뉴 선택이 가능하도록 구현하였다. BACK 을 선택하면 다시 원래의 메뉴를 선택할 수 있다.

두번째 메뉴를 선택하면 앞선 게임 플레이 기록을 확인할 수 있다.

세번째 메뉴를 선택하면 짧은 게임 설명이 출력된다.

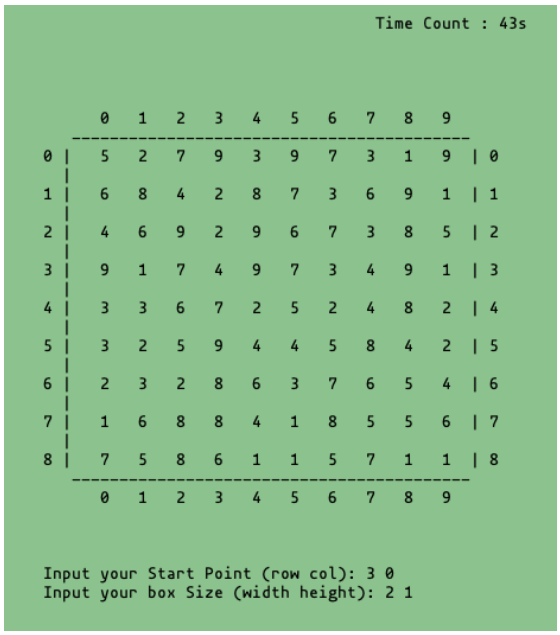
메뉴의 구현은 WINDOW * 의 배열을 생성하여 선택된 인덱스에 따라 배경색 / 글자색 출력을 다르게 함으로써 구현하였다.
`menu_list()` 와 `game_list()` 의 함수 구조가 유사하여, 예로 `menu_list()` 코드를 가져왔다.
이 때 `menu_item` 은 char* 의 리스트로써, 전역에 선언되어 있다.

```
char *menu_item[] = {"", "    SELECT Game Mode", "    RECORD" , "    How to Play", "    Quit"};
```

```
WINDOW ** menu_list() {

    WINDOW **items;
    items = (WINDOW **)malloc(5 * sizeof(WINDOW *));

    items[0] = newwin(6, 32, 18, 25);
    wbkgd(items[0], COLOR_PAIR(2));
    box(items[0], ACS_VLINE, ACS_HLINE);

    items[1] = subwin(items[0], 1, 30, 19, 26);
    items[2] = subwin(items[0], 1, 30, 20, 26);
    items[3] = subwin(items[0], 1, 30, 21, 26);
    items[4] = subwin(items[0], 1, 30, 22, 26);

    for (int i=1 ; i < 5 ; i++)
        wprintw(items[i], "%s", menu_item[i]);
    wbkgd(items[1], COLOR_PAIR(1));
    wrefresh(items[0]);
    return items;
}
```

스크롤 기능 역시 `scroll_menu()` 함수를 구현하여, 매개변수로 받은 WINDOW* 배열과 메뉴 개수에 따라 인덱스 계산을 하여 구현하였다.

이제 첫번째 게임 모드 (MakeZERO) 를 선택하여 플레이를 해 보면,

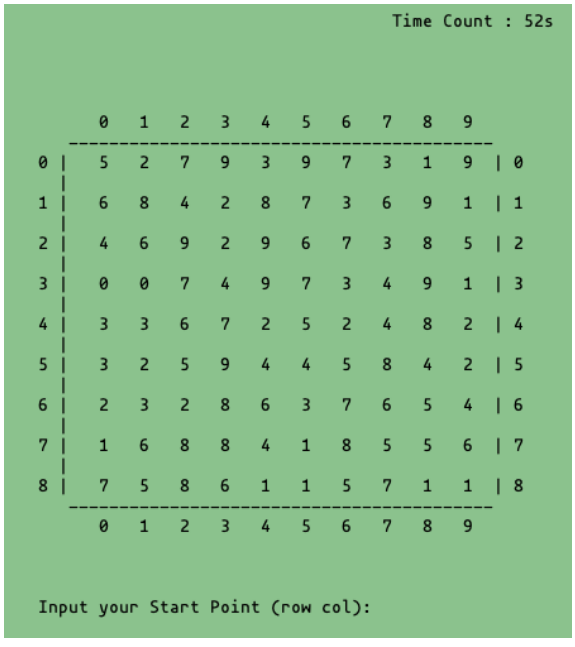


thread 로 구현된 `time WINDOW` 에서 시간이 1 초 단위로 측정되고 있고,

```c
/* Start game */
if (selected_game == 1) {

    pthread_t t1;
    pthread_create(&t1, NULL, time_count, (void*)time);
    gameStart_makeZero(gameWin);
}
```

1 ~ 9 까지의 숫자가 각각 10 개 씩 랜덤으로 좌표와 함께 출력된다.

시작 좌표와 box 의 사이즈를 입력하면
`calculate()` 함수에서 합을 계산하고, 이 합이 10 이라면 해당 box 안의 숫자들을 0 으로 만든다.



TimeKILL 모드에서도 이와 비슷하게 작동하는데,
비슷하게 thread 를 사용하여 이번에는 설정된 시간 (100 초) 부터 0 초까지를 카운트 해 나간다.

```c
if (selected_game == 2) {

    pthread_t t2;
    pthread_create(&t2, NULL, time_check, (void *)time);
    gameStart_timeKill(gameWin, score);

}
```

```c
void *time_check(void *time) {

    WINDOW* timeWin = (WINDOW*) time;
    signal(SIGALRM, game_finish);
    alarm(100);

    for (timeCount = 100; timeCount > 0; timeCount--) {
        werase(timeWin);
        wprintw(timeWin, "Time left : %ds", timeCount);
        sleep(1);
        wrefresh(timeWin);
    }

}
```

이 코드에서 알 수 있듯이,

`alarm()` 함수와 `SIGALRM` 을 사용하여 설정한 시간이 지나면 게임을 종료하는 함수를 호출한다. (timer signal)

제한 시간이 지나면 점수와 함께 기록되었다는 창이 뜨며 게임이 종료된다.

기록을 저장하는 과정에서, **fork()** 를 사용하여 **두 개의 프로세스를 구현**하였고 **pipe()** 를 사용하여 부모 - 자식간의 정보를 전달하였다.
부모 프로세스에서 정보를 얻어 특정한 양식으로 자식 프로세스에게 기록을 전달하면, 자식 프로세스에서 파일에 작성한다.

이에 관한 코드는 다음과 같다.

```c
void make_record(int score, int gametype) {

    int thepipe[2];
    char buf[BUFSIZ];
    char type[BUFSIZ];

    FILE *fp = fopen("./Record/gameRecord.txt", "a");

    if (pipe(thepipe) == -1) {
        perror("Could not make pipe");
        exit(1);
    }

    switch(fork()) {

        case -1:
            perror("fork error");

        /* Child Process */
        case 0:
            close(thepipe[1]);
            read(thepipe[0], buf, BUFSIZ);
            fprintf(fp, "%s", buf);
            break;

        /* Parent Process */
        default:

            close(thepipe[0]);

            char timebuf[BUFSIZ];
            time_t t = time(NULL);
            struct tm* tm_info = localtime(&t);
```

```
            strftime(timebuf, BUFSIZ, "%Y.%m.%d %H:%M", tm_info);

            if (gametype == 1){
                sprintf(type, "MakeZERO mode");
            }
            else {
                sprintf(type, "TimeKILL mode");
            }

            sprintf(buf, "%20s   %5d    %20s\n", type, score, timebuf);

            if(strlen(buf) != write(thepipe[1], buf, strlen(buf))) {
                perror("write");
                exit(3);
            }
            wait(NULL);
            break;
    }
}
```

저장된 기록은 위에서 기술했던 것과 같이 메뉴에서 RECORD 를 선택하면 확인할 수 있다.

크게 중요하지 않은 부분은 생략하였으나, 선언된 함수들은 다음과 같다.

```
/* function Declaration */
void init_scr();
void printLogo();
void printList(WINDOW *scr);
void makeRandomList();
void calculate(int *, int *);
WINDOW ** menu_list();
int scroll_menu(WINDOW **, int , int , int , int);
WINDOW **game_list();
void check_quit();
int totalSum();
int score();
void *time_check(void *);
void *time_count(void *);
void gameStart_makeZero(WINDOW*);
void gameStart_timeKill(WINDOW* , WINDOW* );
void make_record(int , int);
void game_finish(int);
void print_record();
void print_howto();
void wait();
```

III. 결과 및 사용법

II. 에서 함께 기술하였으며, 사용법은 `./tenox.sh` 로 스크립트를 호출하면 된다.

처음에 계획한 그대로 구현한 결과이며, 아직 완벽하게 모든 상황에서 올바르게 실행되지는 않으나 대체적으로 잘 작동된다.

IV. 고찰

먼저 처음에 계획한 대로 구현을 끝까지 완성했다는 점에서 만족스럽다. 중간 중간에 이유모를 core dump 가 발생되어 골머리를 앓았지만, 딱히 수정한 부분이 없는데 어느 순간부터 잘 작동되어 아직도 해결되지 못한 궁금증이 생겼다.

콘솔에서 실행되는 게임이지만 나만의 디자인을 입혀보고 싶어, 수업 시간 때 학습한 curse 라이브러리의 심화 버전인 ncurse 라이브러리를 공부하여 구현해 보았다. 좌표와 박스 사이즈 입력을 숫자 입력 말고, 방향키로 선택할 수 있게 하고 싶은 마음이 있었는데 너무 복잡해질 것 같아서 이번 프로젝트에서는 제외시켰지만, 나중에 기회가 된다면 한번 진화시켜보고 싶다.

아쉬운 점은, 마지막 게임 스코어 창이 도무지 사라지지 않는다는 것이다. 다른 종료창과 코드가 비슷한데 대체 왜 이 친구만 사라지지 않는지 모르겠다. 또한 핸들러로 발생 시킨 함수, 즉 ctrl + c 를 입력하고 나타나는 종료 확인 창에서 y 가 아닌 다른 키를 입력하면 에러가 발생한다. 핸들러 함수가 끝나면 발생 위치로 돌아와야 하는데, 이 방법을 모르겠다. 이런 점이 아쉽긴 하지만, 한 학기 동안 학습한 것을 토대로 나름 만족하는 결과물을 만들어 내서 기쁘다.

V. 참고자료

주로 ncurses 라이브러리에 대한 정보를 참고하였음

1. https://minwook-shin.github.io/basic-ncurses/
2. https://psman2.tistory.com/entry/ncurses-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D