/\* AniTool\_F.c \*/

#include "AniTool.h"

#include "Error.h"

void init\_AniList(AniList\* List, Ptr\_AniList\* pHeader)

{

\*pHeader = List;

List->Left = NULL;

List->Right = NULL;

List->number = 1;

List->showTime = INITIAL\_SHOWTIME;

init\_Map(&(List->MapData));

}

AniList\* add\_AniList(AniList\* pCurrent, Ptr\_AniList const Header)

{

AniList\* New = NULL;

New = (AniList\*)malloc(sizeof(AniList));

init\_Map(&(New->MapData));

New->showTime = INITIAL\_SHOWTIME;

New->Left = pCurrent;

New->Right = pCurrent->Right;

if (pCurrent->Right != NULL)

{

pCurrent->Right->Left = New;

}

pCurrent->Right = New;

fill\_Number(Header);

return New;

}

AniList\* copy\_AniList(AniList\* pCurrent, Ptr\_AniList const Header)

{

AniList\* New = NULL;

New = (AniList\*)malloc(sizeof(AniList));

\*New = \*pCurrent;

New->Left = pCurrent;

New->Right = pCurrent->Right;

if (pCurrent->Right != NULL)

{

pCurrent->Right->Left = New;

}

pCurrent->Right = New;

fill\_Number(Header);

return New;

}

AniList\* delete\_AniList(AniList\* pCurrent, Ptr\_AniList\* pHeader)

{

/\* pCurrent에는 이전 pCurrent 또는 자신을 집어넣고,

Header는여전히 맨 앞의 값을 가리키도록함\*/

Ptr\_AniList Temp = NULL;

Ptr\_AniList Temp2 = NULL;

if (pCurrent->Left == NULL)

{ // 첫 번째꺼라면

if (pCurrent->Right == NULL)

{ // 내용물이 한 개일 때

init\_Map(&(pCurrent->MapData));

pCurrent->showTime = INITIAL\_SHOWTIME;

}

else

{

\*pHeader = pCurrent->Right;

(\*pHeader)->Left = NULL;

free(pCurrent);

pCurrent = \*pHeader;

}

}

else if (pCurrent->Right == NULL)

{ // 마지막꺼 라면

Temp = pCurrent->Left;

free(pCurrent);

pCurrent = Temp;

pCurrent->Right = NULL;

}

else

{ // Ordinary

Temp = pCurrent->Left;

Temp2 = pCurrent->Right;

free(pCurrent);

Temp->Right = Temp2;

Temp2->Left = Temp;

pCurrent = Temp;

}

fill\_Number(\*pHeader);

return pCurrent;

}

AniList\* delete\_All(Ptr\_AniList \*pHeader)

{

if (is\_Empty(\*pHeader)) {

usrErr\_Exit(2, " pHeader가 가리키는 값이 없습니다 in delete\_All");

}

Ptr\_AniList Ptr = \*pHeader;

while (Ptr->Right != NULL) { // Ptr이 맨 오른쪽 노드를 가르치게 함

Ptr = Ptr->Right;

}

while (true)

{

if (Ptr->Left != NULL) {

Ptr = delete\_AniList(Ptr, pHeader);

}

else {

Ptr = delete\_AniList(Ptr, pHeader);

break;

}

}

return Ptr;

}

void fill\_Number(Ptr\_AniList const Header)

{

if (is\_Empty(Header)) {

usrErr\_Exit(2, " Header가 가리키는 값이 없습니다 in fill\_Number ");

}

Ptr\_AniList Ptr = Header;

int i = 1;

while (Ptr->Right != NULL) {

Ptr->number = i++;

Ptr = Ptr->Right;

}

Ptr->number = i;

}

AniList\* free\_AllList(Ptr\_AniList\* pHeader)

{

if (is\_Empty(\*pHeader)) {

return NULL; // 비어있다면 그냥 return

}

AniList\* pCurrent = \*pHeader;

AniList\* pRight = NULL;

while (pCurrent->Right != NULL)

{

pRight = pCurrent->Right;

free(pCurrent);

pCurrent = pRight;

} /\* while 문을 빠져 나올때 상황: pRight - 마지막 List, pCurrent - 마지막 List \*/

pRight = NULL;

free(pCurrent);

\*pHeader = NULL;

return NULL;

}

bool is\_Empty(const Ptr\_AniList Header)

{

if (Header == NULL) {

return true;

}

else {

return false;

}

}

void init\_Map(Map\* MapData)

{

int y, x;

for (y = 0; y < Y\_SIZE; y++)

{

for (x = 0; x < X\_SIZE; x++)

{

MapData->shapeOfMap[y][x] = off;

}

}

}

void init\_Cur(Point\* Cur)

{

Cur->x = X\_START;

Cur->y = Y\_START;

}

void gotoxy(const int x, const int y)

{

printf("\033[%d;%df", y, x);

fflush(stdout); // 즉시 작동하게 함

}

void gotoUnder(const int num)

{

printf("\033[%dB", num);

fflush(stdout);

}

void gotoRight(const int num)

{

printf("\033[%dC", num);

fflush(stdout);

}

void gotoLeft(const int num)

{

printf("\033[%dD", num);

fflush(stdout);

}

void hide\_Cur()

{

printf("\033[?25l");

fflush(stdout);

}

void reveal\_Cur()

{

printf("\033[?25h");

fflush(stdout);

}

void default\_Cur()

{

printf("\033[0m");

fflush(stdout);

}

void set\_Color(const color\_t color)

{

switch (color) {

case \_default:

default\_Cur();

break;

case \_blue:

printf("\033[36m"); // 사실 cyan

fflush(stdout);

break;

case \_red:

printf("\033[31m");

fflush(stdout);

break;

case \_yellow:

printf("\033[33m");

fflush(stdout);

break;

case \_green:

printf("\033[32m");

fflush(stdout);

break;

default:

usrErr\_Exit(2, "not color error in set\_Color");

}

}

void clear()

{

printf("\033[2J");

fflush(stdout);

}

void erase\_Line(const int line)

{

int space;

gotoxy(1, line);

for (space = 1; space < HORIZON - 1; space++)

{

printf(" ");

}

}

void erase\_Back(const int fromleft)

{

int i;

gotoLeft(fromleft);

for (i = 0; i < fromleft; i++) {

printf(" ");

}

gotoLeft(fromleft);

fflush(stdout);

}

void screen\_Tool(const AniList List, const Point Cur)

{

clear();

print\_Menual();

print\_Map(List, Cur);

print\_Menu(List, Cur);

}

AniList\* screen\_Excute(const AniList List, const Ptr\_AniList Header, const Point Cur)

{

AniList\* Next = NULL;

if (List.Right == NULL) {

Next = Header;

}

else {

Next = List.Right;

}

clear();

print\_Map(List, Cur);

sleep(List.showTime);

// usleep( List.showTime );

return Next;

}

void print\_Map(const AniList List, const Point Cur)

{

int x;

int y;

gotoxy(X\_START - 1, Y\_START - 1);

printf("┌");

for (x = 0; x<X\_SIZE; x++)

{

printf("─");

}

printf("┐");

gotoxy(X\_START - 1, Y\_START);

for (y = 0; y < Y\_SIZE; y++)

{

printf("│");

print\_a\_Line(List.MapData, Cur, y);

printf("│");

gotoUnder(1);

gotoLeft(X\_SIZE + 2);

}

printf("└");

for (x = 0; x < X\_SIZE; x++)

{

printf("─");

}

printf("┘");

fflush(stdout);

}

void print\_a\_Line(const Map MapData, const Point Cur, const int y)

{

int x;

for (x = 0; x < X\_SIZE; x++)

{

switch (get\_Shape(MapData, Cur, x, y))

{

case off:

printf(" ");

break;

case on:

printf("\*");

break;

case cur\_off:

printf("▽");

break;

case cur\_on:

printf("▼");

break;

}

}

}

void print\_Menu(const AniList List, const Point Cur)

{

gotoxy(MENU\_X + 1, MENU\_ONE\_Y);

printf("Save");

gotoUnder(1);

gotoLeft(4);

printf("Add");

gotoUnder(1);

gotoLeft(3);

printf("Copy");

gotoUnder(1);

gotoLeft(4);

printf("Erase");

gotoUnder(1);

gotoLeft(5);

printf("Delete");

gotoUnder(1);

gotoLeft(6);

printf("Load");

gotoUnder(1);

gotoLeft(4);

printf("Quit");

gotoUnder(1);

gotoLeft(4);

printf("Reset");

gotoxy(MENU\_X + 1, MENU\_TAIL\_Y);

printf("%dp", List.number);

const int whereCur = 7; // 오른쪽 메뉴의 커서 위치

if (Cur.x == MENU\_X)

{

if (Cur.y == MENU\_ONE\_Y) {

gotoxy(MENU\_X + whereCur, MENU\_ONE\_Y);

printf("◀");

}

else if (Cur.y == MENU\_TWO\_Y) {

gotoxy(MENU\_X + whereCur, MENU\_TWO\_Y);

printf("◀");

}

else if (Cur.y == MENU\_THREE\_Y) {

gotoxy(MENU\_X + whereCur, MENU\_THREE\_Y);

printf("◀");

}

else if (Cur.y == MENU\_FOUR\_Y) {

gotoxy(MENU\_X + whereCur, MENU\_FOUR\_Y);

printf("◀");

}

else if (Cur.y == MENU\_FIVE\_Y) {

gotoxy(MENU\_X + whereCur, MENU\_FIVE\_Y);

printf("◀");

}

else if (Cur.y == MENU\_SIX\_Y) {

gotoxy(MENU\_X + whereCur, MENU\_SIX\_Y);

printf("◀");

}

else if (Cur.y == MENU\_SEVEN\_Y) {

gotoxy(MENU\_X + whereCur, MENU\_SEVEN\_Y);

printf("◀");

}

else if (Cur.y == MENU\_EIGHT\_Y) {

gotoxy(MENU\_X + whereCur, MENU\_EIGHT\_Y);

printf("◀");

}

}

}

void show\_AtCmdLIne(const char\* commandLine)

{

gotoxy(1, COMMAND\_LINE);

printf("%s", commandLine);

}

void print\_Menual()

{

gotoxy(1, 1);

printf("Unix Team-Project ® 2017\n");

printf("key: ↑ ↓ → ← space [ ] ");

}

shape\_t get\_Shape(const Map MapData, const Point Cur, const int x, const int y)

{

if (MapData.shapeOfMap[y][x] == off)

{

if ((Cur.x == x + X\_START) && (Cur.y == y + Y\_START))

return cur\_off;

else

return off;

}

else

{

if ((Cur.x == x + X\_START) && (Cur.y == y + Y\_START))

return cur\_on;

else

return on;

}

}

int kbhit()

{

/\*

출력 - 입력x : 0

- 입력o : 입력값

\*/

struct termios old\_tm, new\_tm;

/\* 참고

struct termios

struct{

tcflag\_t c\_iflag; // input modes //

tcflag\_t c\_oflag; // ouput modes //

tcflag\_t c\_clfag; // contrl modes // 통신 관련

tcflag\_t c\_lfag; // local flag // 출력 효과, 특수문자 등 제어

cc\_t c\_cc[NCCS];

};

\*/

union Askii ch = { 0 };

int old\_fd; // file descripter

int nread;

fflush(stdout);

tcgetattr(STDIN\_FILENO, &old\_tm); // tcgetatter(): 속성get

new\_tm = old\_tm;

new\_tm.c\_lflag &= ~(ICANON | ECHO);

// ICANON: on되면 입력이 이루어진다

// ECHO: off 되어있으면 echo하지 않는다

tcsetattr(STDIN\_FILENO, TCSANOW, &new\_tm);

// TCSANOW : 호출한 즉시 변경사항 적용

//old\_fd = fcntl( STDIN\_FILENO, F\_GETFL, 0 );

//fcntl( STDIN\_FILENO, F\_SETFL, old\_fd | O\_NONBLOCK );

/\* fcntl(): 파일의 전체나 일부를 다른 프로세스에서 사용하는것을

제한할 수 있도록 하는 함수 \*/

nread = read(STDIN\_FILENO, ch.char\_buf, sizeof(int));

tcsetattr(STDIN\_FILENO, TCSANOW, &old\_tm);

//fcntl( STDIN\_FILENO, F\_SETFL, old\_fd );

// fcntl이 atexit 버그의 원인이었다.

if (nread > 0) {

return (ch.int\_buf);

}

return 0;

}

void set\_default\_tm()

{

tcsetattr(STDIN\_FILENO, TCSANOW, &default\_tm);

}

void up\_Tool(Point\* Cur)

{

if (Cur->x == MENU\_X)

{

if (Cur->y > MENU\_ONE\_Y)

Cur->y--;

}

else

{

if (Cur->y > Y\_START)

Cur->y--;

}

}

void down\_Tool(Point\* Cur)

{

if (Cur->x == MENU\_X)

{

if (Cur->y < MENU\_END\_Y)

Cur->y++;

}

else

{

if (Cur->y < Y\_START + Y\_SIZE - 1)

Cur->y++;

}

}

void left\_Tool(Point\* Cur)

{

if (Cur->x > X\_START)

Cur->x--;

}

void right\_Tool(Point\* Cur)

{

if (Cur->y <= MENU\_END\_Y)

{

if (Cur->x < MENU\_X)

Cur->x++;

}

else

{

if (Cur->x < MENU\_X - 1)

Cur->x++;

}

}

AniList\* space\_Tool(AniList\* pCurrent, Ptr\_AniList \*pHeader, Point\* Cur)

{

if (Cur->x < MENU\_X)

{

if (pCurrent->MapData.shapeOfMap[Cur->y - Y\_START][Cur->x - X\_START]

== on)

{

pCurrent->MapData.shapeOfMap[Cur->y - Y\_START][Cur->x - X\_START]

= off;

}

else

{

pCurrent->MapData.shapeOfMap[Cur->y - Y\_START][Cur->x - X\_START]

= on;

}

}

else

{ // 오른쪽 메뉴

switch (Cur->y)

{

case MENU\_ONE\_Y: // Save

save(\*pHeader);

break;

case MENU\_TWO\_Y: // Add

pCurrent = add\_AniList(pCurrent, \*pHeader);

break;

case MENU\_THREE\_Y: // Copy

pCurrent = copy\_AniList(pCurrent, \*pHeader);

break;

case MENU\_FOUR\_Y: // Erase

init\_Map(&(pCurrent->MapData));

break;

case MENU\_FIVE\_Y: // Delete

pCurrent = delete\_AniList(pCurrent, pHeader);

break;

case MENU\_SIX\_Y: // Load

pCurrent = load(pCurrent, pHeader);

break;

case MENU\_SEVEN\_Y: // Quit

pthread\_exit((void \*)0);

break;

case MENU\_EIGHT\_Y: // Reset

pCurrent = delete\_All(pHeader);

break;

}

}

return pCurrent;

}

void file\_List(const char\* dirname)

{

DIR\* od = NULL;

struct dirent\* dir = NULL;

struct stat info;

int permi\_R = 0, permi\_W = 0; // permission for regular file

int permi\_X = 0; // permission for excute file

od = opendir(dirname);

if (od == NULL) {

perr\_Exit(1, "opendir error in ls");

}

printf("File list: /\* colored files are not allowed \*/\n");

while ((dir = readdir(od)) != NULL)

{

// 1. ".", ".." 제외

if (!strcmp(".", dir->d\_name) || !strcmp("..", dir->d\_name))

{

continue; // 다음꺼로 넘어감

}

// 2. directory면 남색

lstat(dir->d\_name, &info);

if (S\_ISDIR(info.st\_mode))

{

set\_Color(\_blue);

printf("%s ", dir->d\_name);

set\_Color(\_default);;

}// 3. regular 파일 -> default, red

else if (S\_ISREG(info.st\_mode))

{

permi\_R = access(dir->d\_name, R\_OK);

permi\_W = access(dir->d\_name, W\_OK);

permi\_X = access(dir->d\_name, X\_OK);

if (((permi\_R == 0) && (permi\_W == 0)) && (permi\_X != 0))

{ // 읽고 쓸 수 있으면, default

printf("%s ", dir->d\_name);

}

else if (permi\_X == 0)

{ // 실행할 수 있으면, green

set\_Color(\_green);

printf("%s ", dir->d\_name);

set\_Color(\_default);

}

else

{ // 읽거나 쓸 수 없으면, red

set\_Color(\_red);

printf("%s ", dir->d\_name);

set\_Color(\_default);

}

}// 4. 그 외 파일 -> yellow

else

{

set\_Color(\_yellow);

printf("%s ", dir->d\_name);

set\_Color(\_default);

}

}

printf("\n");

closedir(od);

}

int check\_File(const char\* filename, const char\* dirname)

{

DIR\* od = NULL;

struct dirent\* dir = NULL;

od = opendir(dirname);

if (od == NULL)

perr\_Exit(1, "opendir error in check\_File");

while ((dir = readdir(od)) != NULL)

{

if (!strcmp(dir->d\_name, filename))

{ // 같다면

closedir(od);

return 0;

}

}

closedir(od);

return 1;

}

void default\_wd()

{

chdir(filewd);

}

void save(const Ptr\_AniList Header)

{

char savename[S\_BUFSIZE];

/\* 알고리즘용 변수\*/

int fd;

char wrbuf[F\_BUFSIZE] = "";

int index = 0;

char key = '\0'; // y/n 받는데 사용

union Askii IntData;

const int xys = X\_SIZE \* Y\_SIZE \* sizeof(int);

Ptr\_AniList CurPtr = Header;

if (check\_File(SAVEDIR, ".")) // 디렉터리 내에 같은파일이 있으면 0

mkdir(SAVEDIR, 0755); // 없다면 1

chdir(SAVEDIR);

gotoxy(1, COMMAND\_LINE);

file\_List(".");

printf("Filename");

scan(savename); // 출력 : Filename << 입력

if (!isAbleToSaveLoad(savename))

{ // 입출력 할 수 있는 프로그램이 아니면

printf("The file can't save.\n");

press\_Key();

chdir(filewd);

return;

}

if (access(savename, F\_OK) == 0)

{ // 파일이 이미 존재한다면

printf("Do you want to overwrite it?(y/n)\n");

while (((key = kbhit()) != Y\_KEY) && (key != N\_KEY))

{

if (key != 0) {

erase\_Back(sizeof(int)); // 화면에 출력된 부분 지우기

}

} // y나 n이 아닐때 무한 반복

if (key == Y\_KEY) { // 지나감

gotoLeft(sizeof(int)); // 다음 출력 대기

}

else if (key == N\_KEY) {

gotoLeft(sizeof(int)); // 다음 출력을 위해

printf("Not Saved.\n");

press\_Key();

chdir(filewd);

return;

}

else {

usrErr\_Exit(2, "Not y/n in save");

}

}

fd = open(savename, O\_WRONLY | O\_CREAT | O\_TRUNC, 0644);

if (fd == -1)

{

printf("The file can't save.\n");

press\_Key();

// press\_Key() : while( !kbhit() ){};

chdir(filewd);

return;

}

/\* 여기부터 저장 알고리즘 \*/

while (CurPtr != NULL)

{

// CurPtr->number 입력

IntData.int\_buf = CurPtr->number;

for (index = 0; index < sizeof(int); index++) {

wrbuf[index] = IntData.char\_buf[index];

}

// CurPtr->MapData 입력

for (index = sizeof(int); index < xys + sizeof(int); index += 4)

{

IntData.int\_buf

= (int)(\*(\*(CurPtr->MapData.shapeOfMap)

+ (index - sizeof(int)) / 4));

wrbuf[index] = IntData.char\_buf[0];

wrbuf[index + 1] = IntData.char\_buf[1];

wrbuf[index + 2] = IntData.char\_buf[2];

wrbuf[index + 3] = IntData.char\_buf[3];

}

// CurPtr->showTime 입력

IntData.int\_buf = CurPtr->showTime;

for (index = xys + sizeof(int); index < (xys + sizeof(int) \* 2);

index++)

{

wrbuf[index] = IntData.char\_buf[index - (xys + sizeof(int))];

}

// 쓰기 작업

if (write(fd, wrbuf, xys + sizeof(int) \* 2) == -1) {

perror("write error in save");

return;

}

CurPtr = CurPtr->Right;

}

close(fd);

chdir(filewd);

printf("The file successfully saved.\n");

press\_Key();

return; // 저장 완료

}

AniList\* load(AniList\* pBefore, Ptr\_AniList\* pHeader)

{

/\* 이전 정보 저장 \*/

Ptr\_AniList OldHead = \*pHeader;

/\* File Open \*/

char loadname[S\_BUFSIZE];

int fd;

if (check\_File(SAVEDIR, ".")) { // 디렉터리 내에 같은파일이 없으면

mkdir(SAVEDIR, 0755);

}

chdir(SAVEDIR);

gotoxy(1, COMMAND\_LINE);

file\_List(".");

printf("Filename");

scan(loadname); // 출력 : Filename <<

if (!isAbleToSaveLoad(loadname))

{ // 입출력 할 수 있는 프로그램이 아니면

printf("The file can't load.\n");

press\_Key();

chdir(filewd);

return pBefore;

}

fd = open(loadname, O\_RDONLY);

if (fd == -1)

{

/\* 비정상 종료 \*/

printf("The file can't load.\n");

press\_Key();

chdir(filewd);

return pBefore;

}

/\* 알고리즘 \*/

union Askii FByte;

union Askii arrByte[X\_SIZE \* Y\_SIZE]; // MapData용

int bytes\_read;

char big\_buf[F\_BUFSIZE]; // MapData용

bool isFirst = true;

int arrIndex = 0;

AniList\* NewNode = NULL;

AniList\* PreNode = NULL;

while (true)

{

/\* 구조체 생성 및 number 입력 \*/

bytes\_read = read(fd, FByte.char\_buf, sizeof(int));

if (bytes\_read != 4)

{

if (((bytes\_read == 0) || (bytes\_read == 1))

&& (isFirst == false))

{

break;

}

else { /\* 비정상 종료 \*/

printf("The file is invalid or corrupted\n");

press\_Key();

\*pHeader = OldHead;

chdir(filewd);

close(fd);

return pBefore;

}

}

NewNode = (AniList\*)malloc(sizeof(AniList));

NewNode->number = FByte.int\_buf;

if (isFirst == true)

{

\*pHeader = NewNode;

isFirst = false;

}

/\* MapData 입력 \*/

bytes\_read

= read(fd, arrByte[0].char\_buf, X\_SIZE \* Y\_SIZE \* sizeof(int));

if (bytes\_read != X\_SIZE \* Y\_SIZE \* sizeof(int))

{ /\* 비정상 종료 \*/

printf("The file is invalid or corrupted\n");

press\_Key();

\*pHeader = OldHead;

chdir(filewd);

close(fd);

return pBefore;

}

for (arrIndex = 0; arrIndex < X\_SIZE \* Y\_SIZE; arrIndex++)

{

\*(\*(NewNode->MapData.shapeOfMap) + arrIndex)

= arrByte[arrIndex].int\_buf;

}

/\* showTime 입력 \*/

bytes\_read = read(fd, FByte.char\_buf, sizeof(int));

if (bytes\_read != 4)

{ /\* 비정상 종료 \*/

printf("The file is invalid or corrupted\n");

press\_Key();

\*pHeader = OldHead;

chdir(filewd);

close(fd);

return pBefore;

}

NewNode->showTime = FByte.int\_buf;

/\* 포인터 입력 \*/

NewNode->Left = PreNode;

if (PreNode != NULL) {

PreNode->Right = NewNode;

}

/\* 다음 루트를 위한 준비 \*/

PreNode = NewNode;

}

NewNode->Right = NULL;

/\* 마무리 작업 \*/

close(fd);

chdir(filewd);

OldHead = free\_AllList(&OldHead);

return \*pHeader;

}

AniList\* control\_Tool(AniList\* pCurrent, Ptr\_AniList \*pHeader, Point\* Cur)

{

int ch = 0;

#ifdef \_ASKII\_DEBUG\_ // 문자코드 확인용

while (!(ch = kbhit()))

{

}

printf("\n%d\n", ch);

#else

while (!(ch = kbhit()))

{

} // 입력이 없을땐 대기

switch (ch)

{

case UP:

up\_Tool(Cur);

break;

case DOWN:

down\_Tool(Cur);

break;

case LEFT:

left\_Tool(Cur);

break;

case RIGHT:

right\_Tool(Cur);

break;

case SPACE:

case ENTER:

pCurrent = space\_Tool(pCurrent, pHeader, Cur);

break;

case SBLEFT:

if (pCurrent->Left != NULL)

pCurrent = pCurrent->Left;

break;

case SBRIGHT:

if (pCurrent->Right != NULL)

pCurrent = pCurrent->Right;

break;

}

#endif

return pCurrent;

}

void press\_Key()

{

while (!kbhit())

{

};

}

void scan(char string[])

{

int index;

printf(" << ");

fflush(stdout);

if (read(STDIN\_FILENO, string, S\_BUFSIZE) < 0) {

perr\_Exit(1, "read error in scan");

}

for (index = 0; index < S\_BUFSIZE; index++)

{

if (string[index] == '\n')

{

string[index] = '\0';

break;

}

}

}

void print\_Opening()

{

clear();

gotoxy(1, 1);

printf(" ┌─────────────┐ \n");

printf(" │ Dot-AniTool │ \n");

printf(" └─────────────┘ \n");

printf("1. NewFile\n");

printf("2. ExcuteFile\n");

printf("3. Quit\n");

}

void print\_OpeningCur(const short int opening\_cur)

{

gotoxy(15, BANNER + 1);

switch (opening\_cur)

{

case 1:

printf("◀");

break;

case 2:

gotoUnder(1);

printf("◀");

break;

case 3:

gotoUnder(2);

printf("◀");

break;

}

if (opening\_cur != 1) {

gotoxy(15, BANNER + 1);

printf(" ");

}

if (opening\_cur != 2) {

gotoxy(15, BANNER + 2);

printf(" ");

}

if (opening\_cur != 3) {

gotoxy(15, BANNER + 3);

printf(" ");

}

}

void cntl\_Opening(short int\* opening\_cur, pthread\_t\* nthread)

{

int ch = 0;

gotoxy(1, 4 + OPENING\_MAX);

while (!(ch = kbhit()))

{

} // 값이 없을땐 대기

switch (ch)

{

case UP:

up\_Opening(opening\_cur);

break;

case DOWN:

down\_Opening(opening\_cur);

break;

case SPACE:

case ENTER:

space\_Opening(\*opening\_cur, nthread);

break;

}

erase\_Line(4 + OPENING\_MAX);

}

void up\_Opening(short int\* opening\_cur)

{

if (\*opening\_cur > OPENING\_MIN) {

(\*opening\_cur)--;

}

}

void down\_Opening(short int\* opening\_cur)

{

if (\*opening\_cur < OPENING\_MAX) {

(\*opening\_cur)++;

}

}

void space\_Opening(const short int opening\_cur, pthread\_t\* nthread)

{

switch (opening\_cur)

{

case 1:

if (pthread\_create(nthread, NULL, thr\_NewFile, NULL) != 0) {

perr\_Exit(1, "thread error in space\_Opening");

}

break;

case 2:

if (pthread\_create(nthread, NULL, thr\_ExcuteFile, NULL) != 0) {

perr\_Exit(1, "thread error in space\_Opening");

}

break;

case 3:

exit(0); // 프로그램 종료

break;

}

}

bool isAbleToSaveLoad(const char\* filename)

{

struct stat info;

int permi\_R, permi\_W, permi\_X;

if (access(filename, F\_OK) != 0) { // 파일이 없다면

return true; // 입력할 수 있다. (출력은 어차피 예외체크에서 막힘)

}

if (lstat(filename, &info) == -1) { // 파일이 있는데 lstat 실패하면

return false;

}

if (!S\_ISREG(info.st\_mode))

{ // regular 파일이 아니면

return false;

}

permi\_R = access(filename, R\_OK);

permi\_W = access(filename, W\_OK);

permi\_X = access(filename, X\_OK);

if (((permi\_R == 0) && (permi\_W == 0)) && (permi\_X != 0))

{ // 읽고 쓸 수 있고 실행할 수 없다면

return true;

}

else { // 읽고 쓸 수 없거나 실행할 수 있다면

return false;

}

}