/\* MainMenu\_F.c \*/

#include "MainMenu.h"

#include "Error.h"

void scan\_Input\_plus(char\* string, const char\* output)

{

/\* 사실상 scan\_Input\_plus에서 printf 부분만 바뀌었습니다 \*/

/\* 지역 변수 정의 \*/

int len;

int index;

bool isVoid = true; // 공백일 시 일어나는 버그 처리용 변수

char new[S\_BUFSIZE] = { '\0', };

char\* ptr;

char vac; // vacuum cleaner(진공 청소기)의 약어

/\* 출력 \*/

printf("%s << ", output);

fflush(stdout);

/\* 입력 \*/

len = read(STDIN\_FILENO, string, S\_BUFSIZE);

if (len < 0) {

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

}

else if (len >= S\_BUFSIZE) { /\* 길이가 버퍼크기보다 클때 에러처리 \*/

printf("too long");

while ((vac = getchar()) != '\n' && vac != '\0')

{

} // 입력스트림 비우기

//////while( read( STDIN\_FILENO, string, S\_BUFSIZE) > 0 )

//////{} // 입력 스트림 비우기 \*/ // 왜 작동 안 할까

////// -> 입력스트림은 EOF가 없어서?

memset(string, '\0', S\_BUFSIZE);

return;

}

string[len - 1] = '\0';

// 입력 마무리

/\* 공백일 시 일어나는 버그 처리 \*/

for (index = 0; index < len - 1; index++)

{ /\* ' ' 외에 다른 값이 있는지 확인, 있으면 false 없으면 true \*/

if (string[index] != ' ')

{

isVoid = false;

break;

}

}

if (isVoid == true) { // 정상적인 경우면 isVoid -> false, 아니면 true

return;

}

/\* 단어 사이의 중복된 space 제거 \*/

ptr = strtok(string, " ");

strcpy(new, ptr);

while (ptr = strtok(NULL, " "))

{ // ptr != NULL

strcat(new, " ");

strcat(new, ptr);

}

/\* 분할된 값을 string에 복사 \*/

memset(string, '\0', S\_BUFSIZE);

strcpy(string, new);

}

void scan\_Input(char string[])

{

/\* 지역 변수 정의 \*/

int len;

int index;

bool isVoid = true; // 공백일 시 일어나는 버그 처리용 변수

char new[S\_BUFSIZE] = { '\0', };

char\* ptr;

char vac; // vacuum cleaner(진공 청소기)의 약어

/\* 출력 \*/

printf("command >> ");

fflush(stdout);

/\* 입력 \*/

len = read(STDIN\_FILENO, string, S\_BUFSIZE);

if (len < 0) {

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

}

else if (len >= S\_BUFSIZE) { /\* 길이가 버퍼크기보다 클때 에러처리 \*/

printf("too long");

while ((vac = getchar()) != '\n' && vac != '\0')

{

} // 입력스트림 비우기

//////while( read( STDIN\_FILENO, string, S\_BUFSIZE) > 0 )

//////{} // 입력 스트림 비우기 \*/ // 왜 작동 안 할까

////// -> 입력스트림은 EOF가 없어서?

memset(string, '\0', S\_BUFSIZE);

return;

}

string[len - 1] = '\0';

// 입력 마무리

/\* 공백일 시 일어나는 버그 처리 \*/

for (index = 0; index < len - 1; index++)

{ /\* ' ' 외에 다른 값이 있는지 확인, 있으면 false 없으면 true \*/

if (string[index] != ' ')

{

isVoid = false;

break;

}

}

if (isVoid == true) { // 정상적인 경우면 isVoid -> false, 아니면 true

return;

}

/\* 단어 사이의 중복된 space 제거 \*/

ptr = strtok(string, " ");

strcpy(new, ptr);

while (ptr = strtok(NULL, " "))

{ // ptr != NULL

strcat(new, " ");

strcat(new, ptr);

}

/\* 분할된 값을 string에 복사 \*/

memset(string, '\0', S\_BUFSIZE);

strcpy(string, new);

}

void clear\_Screen()

{

printf("\033[2J");

printf("\033[1;1f");

fflush(stdout);

}

void init\_Screen()

{

clear\_Screen();

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf(" UNIX PROGMRAMMING TEAM-PROJECT ® 2017 \n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("/ ? or \"help\" for help /\n");

}

void warn()

{

printf(":incorrect command\n");

}

void show\_umask()

{

int show;

show = umask(0777/\* meanless value \*/); // 기존 값을 show에 준다

printf("umask: %04o\n", show);

umask(show); // umask값 복구

}

void ch\_umask(const char string[])

{

int whenSpaceEnd = 0;

int index = 0, arglen = 0;

int mode = 0;

for (index = strlen("umask"); string[index] == ' '; index++)

{

} // index <- Mode Number의 시작점

whenSpaceEnd = index;

if (whenSpaceEnd == strlen("umask") /\* 5 \*/)

{ // 띄어쓰기가 없으면 X

warn();

return;

}

for (index = whenSpaceEnd; string[index] != '\0'; index++)

{

} // index -> '\0'의 위치

arglen = index - whenSpaceEnd;

if (arglen == 3) // arglen이 3이랑 4일때만 통과

{

if (!isNum(&string[whenSpaceEnd], 3, 8)) {

warn(); // 수가 아니면 X

return;

}

}

else if (arglen == 4)

{

if (!isNum(&string[whenSpaceEnd], 4, 8)) {

warn();

return;

}

}

else // arglen이 3과 4가 아니면 종료

{

warn();

return;

}

mode = getIntStr(&string[whenSpaceEnd], 8);

if (mode < 0 || mode > 0777) {

printf(":escape octal scope\n");

return;

}

umask(mode);

show\_umask();

}

int getIntStr(const char str[], const int format /\* 진수 \*/)

{

int index, value = 0;

for (index = 0; (str[index] >= '0') &&

(str[index] <= ('0' + format - 1)); index++)

{

value \*= format;

value += (str[index] - '0');

}

return value;

}

bool isNum(const char string[], const int digits /\* 자릿수 \*/,

const int format /\* 진수 \*/)

{

int index = 0;

for (index = 0; index < digits; index++)

{

if (!((string[index] >= '0') &&

(string[index] <= ('0' + format - 1))))

{ // 만약 8진수라면 '7'까지 확인

return false; // 숫자가 아니면 false

}

}

return true;

}

void whatDateIsToday()

{

FILE\* p\_read; // p\_read는 pipe queue의 나가는 부분의 포인터

char date[S\_BUFSIZE];

p\_read = popen("date", "r");

if (p\_read == NULL) {

usrErr\_Exit(1, "popen failed in whatDateIsToday\n");

}

if (fgets(date, sizeof(date), p\_read) == NULL) {

usrErr\_Exit(1, "No data from date in whatDataIsToday\n");

}

printf("%s", date);

pclose(p\_read);

}

void whatTimeIsIt()

{

time\_t current;

time(&current); // 일련의 숫자로 기록됨

printf("%s", ctime(&current)); // 사람이 읽기 쉽게 변경해서 출력

}