/\* this is MainMenu.c \*/

#include "MainMenu.h"

#include "Error.h"

#include "/home/stud17/16101361/.TeamProject/menu/security.h"

// security 관련 헤더 파일

static pid\_t pid;

// 자기 자신의 pid

//child가 있으면 child ID

int main(int argc, char\* argv[])

{

pid = getpid();

/\* security 체크 \*/

char\* const rkey = argv[0];

char\* uid;

int is\_root = 0; // 슈퍼 유저용 지역변수

if (check\_logfile(rkey) == -1) {

usrErr\_Exit(2, "check\_logfile\n");

}

uid = getenv("USRID");

if (!strcmp(uid, ROOTID))

{

is\_root = 1;

}

/\*----------------------------------------\*/

/\* 종료 관련 처리 \*/

if (atexit(clear\_Screen) != 0)

{

perr\_Exit(1, "atexit error in MainMenu\n");

}

if (signal(SIGINT, sig\_int) == SIG\_ERR)

{

perr\_Exit(1, "signal error in MainMenu\n");

}

if (signal(SIGCHLD, sig\_chld) == SIG\_ERR)

{

perr\_Exit(1, "sginal error in MainMenu\n");

}

/\* MainMenu 프로그램 \*/

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

init\_Screen();

while (true)

{

scan\_Input(input\_buf); // input\_buf에 scan

switch (get\_menu(input\_buf, is\_root))

{

/\* 명령어 여기에 넣고,

enum menu\_t랑 ( 헤더 파일 )

get\_menu 함수랑

print\_Help 함수에 각각 추가해주세요@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@필독\*/

case \_chumask:

ch\_umask(input\_buf);

break;

case \_umask:

show\_umask();

break;

case \_quit:

return 0;

break;

case \_help:

print\_Help(is\_root);

break;

case \_clear:

init\_Screen();;

break;

case \_date:

whatDateIsToday();

break;

case \_time:

whatTimeIsIt();

break;

case \_chpasswd:

change\_password(uid);

break;

/\* 기본 기능 외 추가 기능. 여기부터 추가해 주세요\*/

/\* 나중에 추가한 함수가 아래로 가도록 해주세요 \*/

case \_filelist:

pid = filelist(rkey); // child ID

if (waitpid(pid, NULL, 0) < 0) {

perr\_Exit(1, "wait filelist\n");

}

pid = getpid(); // current ID

init\_Screen();

break;

case \_anitool:

pid = anitool(rkey); // child ID

if (waitpid(pid, NULL, 0) < 0) {

perr\_Exit(1, "wait anitool\n");

}

pid = getpid(); // current ID

init\_Screen();

break;

case \_chat:

pid = chatting(rkey); // child ID

if (waitpid(pid, NULL, 0) < 0) {

perr\_Exit(1, "wait chat\n");

}

pid = getpid(); // current ID

init\_Screen();

break;

case \_encrypt:

pid = file\_encryption(rkey); // child ID

if (waitpid(pid, NULL, 0) < 0) {

perr\_Exit(1, "wait chat\n");

}

pid = getpid(); // current ID

init\_Screen();

break;

/\* 여기서부터는 관리자 전용 명령어 입니다 \*/

// 비관리자 사용시 경고 출력도 넣어줘

case \_addusr:

add\_user();

break;

case \_delusr:

delete\_user();

break;

/\*-------------------------------------------------\*/

case -1:

default:

warn();

}

memset(input\_buf, '\0', S\_BUFSIZE);

// buf 초기화

}

/\* ------------------------------------------------ \*/

}

menu\_t get\_menu(const char\* input, const int is\_root)

{ /\* 여기에 명령어랑 반환 값 넣어주세요. \*/

/\* strcmp 앞에 꼭 ! 넣어주세요(동일하면 0 반환함) \*/

/\* 나중에 추가한 if문이 맨 위로 오게 넣어주세요 \*/

if (!strcmp(input, "filelist")) {

return \_filelist;

}

/\*-------------------------------\*/

if (!strcmp(input, "delusr") && is\_root) {

return \_delusr;

}

/\*-------------------------------\*/

if (!strcmp(input, "addusr") && is\_root) {

return \_addusr;

}

/\*-------------------------------\*/

if (!strcmp(input, "encrypt")) {

return \_encrypt;

}

/\*-------------------------------\*/

if (!strcmp(input, "chat")) {

return \_chat;

}

/\*-------------------------------\*/

if (!strcmp(input, "chpasswd")) {

return \_chpasswd;

}

/\*-------------------------------\*/

if (!strcmp(input, "anitool")) {

return \_anitool;

}

/\*-------------------------------\*/

if (!strcmp(input, "time")) {

return \_time;

}

/\*-------------------------------\*/

if (!strcmp(input, "date")) {

return \_date;

}

/\*-------------------------------\*/

if (!strcmp(input, "umask")) {

return \_umask;

}

else if (!strncmp(input, "umask", 5))

{

return \_chumask;

}

/\*-------------------------------\*/

if (!strcmp(input, "quit")) {

return \_quit;

}

else if (!strcmp(input, "q")) {

return \_quit;

}

/\*-------------------------------\*/

if (!strcmp(input, "help")) {

return \_help;

}

else if (!strcmp(input, "?")) {

return \_help;

}

/\*-------------------------------\*/

if (!strcmp(input, "clear")) {

return \_clear;

}

return -1;

}

void print\_Help(const int is\_root)

{

/\* 여기에 도움말 추가해주세요 \*/

/\* 같은 카테고리 안에서 글자순입니다 \*/

printf(" - COMMAND USAGE -\n");

printf("\n");

/\* 기본 기능 \*/

printf(" -------------- basic function --------------\n");

printf("chpasswd : change your password\n");

printf("clear : clear monitor\n");

printf("date : show what the date today is\n");

printf("help or ? : help for command\n");

printf("time : show what time it is\n");

printf("umask : current umask value\n");

printf("umask [octal] : set umask\n");

printf("quit or q : exit program\n");

printf("\n");

/\* 확장 기능 \*/

printf(" ------------- advanced function -------------\n");

printf("anitool : use dot animation tool\n");

printf("chat : use chatting tool\n");

printf("encrypt : use file encryption tool\n");

printf("filelist : use filelist program\n");

printf("\n");

/\* 슈퍼유저 기능 \*/

if (is\_root)

{

printf(" ------------ management function ------------\n");

printf("addusr : add member to the project\n");

printf("delusr : delete member of the project\n");

printf("\n");

}

}

/\*\*\*\*\*\*\*\* 여기 이후에는 fork -> exec 하는 함수만 정의해주세요. pid\_t anitool() 참고 \*\*\*\*\*\*\*\*/

// + 계정 관련 함수

pid\_t filelist(char\* const rkey)

{

pid\_t pid; // 반환할 pid

pid = fork(); // 0 in child, child ID in parent

if (pid < 0)

{

perr\_Exit(1, "fork error in filelist()\n");

}

if (pid == 0) // child 라면

{

if (execl(FILELIST, rkey, (char\*)0) < 0) // filelist 실행

{

perr\_Exit(1, "execl error in filelist()\n");

}

}

return pid; // child ID 반환

}

pid\_t anitool(char\* const rkey)

{

pid\_t pid; // 반환할 pid

pid = fork(); // 0 in child, child ID in parent

if (pid < 0)

{

perr\_Exit(1, "fork error in anitool()\n");

}

if (pid == 0) // child 라면

{

if (execl(ANITOOL, rkey, (char\*)0) < 0) // anitool 실행

{

perr\_Exit(1, "execl error in anitool()\n");

}

}

return pid; // child ID 반환

}

pid\_t chatting(char\* const rkey)

{

pid\_t pid; // 반환할 pid

pid = fork(); // 0 in child, child ID in parent

if (pid < 0)

{

perr\_Exit(1, "fork error in chatting()\n");

}

if (pid == 0) // child 라면

{

if (execl(CHAT, rkey, (char\*)0) < 0) // chat 실행

{

perr\_Exit(1, "execl error in chatting()\n");

}

}

return pid; // child ID 반환

}

pid\_t file\_encryption(char\* const rkey)

{

pid\_t pid; // 반환할 pid

pid = fork(); // 0 in child, child ID in parent

if (pid < 0)

{

perr\_Exit(1, "fork error in file\_encryption()\n");

}

if (pid == 0) // child 라면

{

if (execl(ENCRYPT, rkey, (char\*)0) < 0) // encryt 실행

{

perr\_Exit(1, "execl error in file\_encryption()\n");

}

}

return pid; // child ID 반환

}

void add\_user()

{

char id[30] = { '\0', }, pass[30] = { '\0', }, buf[1024] = { '\0', };

int fd; // FIFO 저장

printf("Input User ID\n");

scan\_Input\_plus(buf, "ID");

if (strlen(buf) > 30)

{

printf("You should input under 30 characters!\n");

return;

}

strcpy(id, buf);

buf[0] = '\0';

printf("Input User Password\n");

scan\_Input\_plus(buf, "Password");

if (strlen(buf) > 30)

{

printf("You should input under 30 characters!\n");

return;

}

strcpy(pass, buf);

buf[0] = '\0';

printf("Input User Password Again\n");

scan\_Input\_plus(buf, "Password");

if (strlen(buf) > 30)

{

printf("You should input under under 30 characters!\n");

return;

}

if (strcmp(buf, pass))

{

printf("Not Match!\n");

return;

}

strcpy(buf, "addusr:");

strcat(buf, id);

strcat(buf, ":");

strcat(buf, pass);

strcat(buf, ";");

// addusr: "id": "pass";

if ((fd = open(LOGMSG, O\_WRONLY, 0644)) < 0)

{

printf("Server is closed.\n");

return;

}

if (write(fd, buf, strlen(buf)) < 0)

{ // fd = FIFO

perr\_Exit(1, "write error in addusr\n");

}

close(fd);

}

void delete\_user()

{

char id[30] = { '\0', }, buf[1024] = { '\0', };

int fd; // FIFO 저장

printf("Input User ID\n");

scan\_Input\_plus(buf, "ID");

if (strlen(buf) > 30)

{

printf("You should input under 30 characters!\n");

return;

}

strcpy(id, buf);

buf[0] = 0;

printf("Input User ID again\n");

scan\_Input\_plus(buf, "ID");

if (strlen(buf) > 30)

{

printf("You should input under 30 characters!\n");

return;

}

if (strcmp(id, buf))

{

printf("Not match!!");

return;

}

strcpy(buf, "delusr:");

strcat(buf, id);

strcat(buf, ":");

// delusr:"id":

if ((fd = open(LOGMSG, O\_WRONLY, 0644)) < 0)

{ // fd = FIFO

printf("Server is closed.\n");

return;

}

if (write(fd, buf, strlen(buf)) < 0)

{

perr\_Exit(1, "write error in delusr\n");

}

close(fd);

}

void change\_password(const char \*id)

{

char pass[30] = { '\0', }, buf[1024] = { '\0', };

int fd; // FIFO 저장

printf("Input User Password\n");

scan\_Input\_plus(buf, "Password");

if (strlen(buf) > 30)

{

printf("You should input under 30 characters!\n");

return;

}

strcpy(pass, buf);

buf[0] = '\0';

printf("Input User Password Again\n");

scan\_Input\_plus(buf, "Password");

if (strlen(buf) > 30)

{

printf("You should input under 30 characters!\n");

return;

}

if (strcmp(buf, pass))

{

printf("Not Match!\n");

return;

}

strcpy(buf, "chpw:");

strcat(buf, id);

strcat(buf, ":");

strcat(buf, pass);

strcat(buf, ";");

if ((fd = open(LOGMSG, O\_WRONLY, 0644)) < 0)

{ // fd = FIFO

printf("Server is closed.\n");

return;

}

if (write(fd, buf, strlen(buf)) < 0)

{

perr\_Exit(1, "write error in chpasswd\n");

}

close(fd);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 여기 이후는 sginal 처리 함수 입니다. 안 보셔도 됩니다 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void sig\_int(int signo)

{

if (pid == getpid())

{ // 자식 프로세스가 없으면

usrErr\_Exit(1, "program exited by sigint\n");

}

}

void sig\_chld(int stgno)

{

/\* 의도적으로 비어있음 \*/

}