Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

—

Институт компьютерных наук и кибербезопасности

**Высшая школа кибербезопасности**

**ЛАБОРАТОРНАЯ РАБОТА № 12**

**Система контроля доступа**

по дисциплине «Основы информационной безопасности»

Выполнили

студенты гр. 5131001/30002 Мишенев Н. С.

Квашенникова В. М.

<*подпись*>

Руководитель

программист Вагисаров В. Б.

<*подпись*>

Санкт-Петербург

2024г.

СОДЕРЖАНИЕ

[Цель работы 3](#_Toc341260576)

[Постановка задачи 3](#_Toc1946340292)

[Описание структур ACL 3](#_Toc1969454822)

[Расчет действующих разрешений 4](#_Toc2109103708)

[Произведенные модификации в структуре ACL 6](#_Toc328201574)

[ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ 6](#_Toc63419545)

[ВЫВОД 7](#_Toc764410415)

[ПРИЛОЖЕНИЕ А 8](#_Toc843065386)

**Цель работы**

Освоение средств контроля и управления доступом пользователей к ресурсам операционной системы, приобретение навыков распределения прав на примере файловой системы NTFS в среде Windows.

**Постановка задачи**

* В разделе файловой системы NTFS создать каталог;
* Продемонстрировать, что групповые права распространяются на всех членов группы;
* Продемонстрировать приоритет запрещающих прав над разрешающими;
* Разработать четыре тестовые утилиты, которые выполняют операции создания, чтения, записи и изменения атрибутов безопасности.

# **Описание структур ACL**

Структура ACL (Access Control List) для объектов представляется в виде списка элементов, где каждый элемент представлен в формате ACE (Access Control Entry) вида "субъект - тип ACE - права доступа".

1. Субъект - это идентификатор пользователя или группы, для которых устанавливаются права доступа.
2. Тип ACE может быть:

* Allow (Разрешение): позволяет указанному субъекту выполнять определенные действия с объектом.
* Deny (Запрещение): запрещает указанному субъекту выполнять определенные действия с объектом.

1. Права доступа - это специфические разрешения или действия, которые предоставляются или ограничиваются для указанного субъекта.

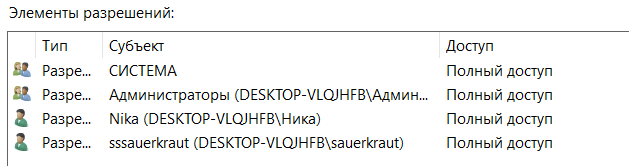


Рисунок 1 — Описание ACL для файла

Основные права доступа в NTFS:

* Чтение: разрешает только чтение файла.
* Запись: разрешает чтение и запись.
* Изменение: разрешает чтение, запись, переименование, удаление и редактирование атрибутов.
* Полный доступ: даёт пользователю все права, включай возможность редактирования ACL.

# **Расчет действующих разрешений**

Добавим группу “Гости” и пользователя “Гость” в элементы разрешений во вкладке дополнительных параметров безопасности для файла и при добавлении укажем полный доступ к файлу. Как можно заметить, и у группы и пользователя имеется весь спектр прав доступа (рис. 2), (рис. 3).

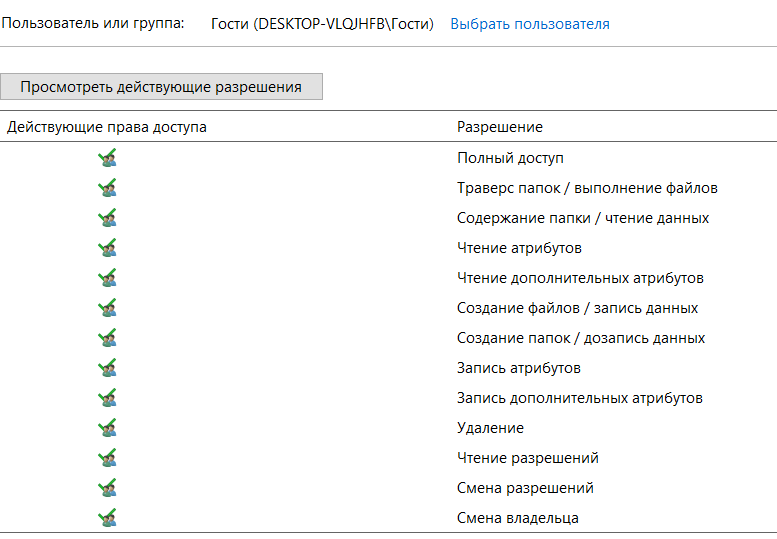


Рисунок 2 — Разрешения для группы "Гости"

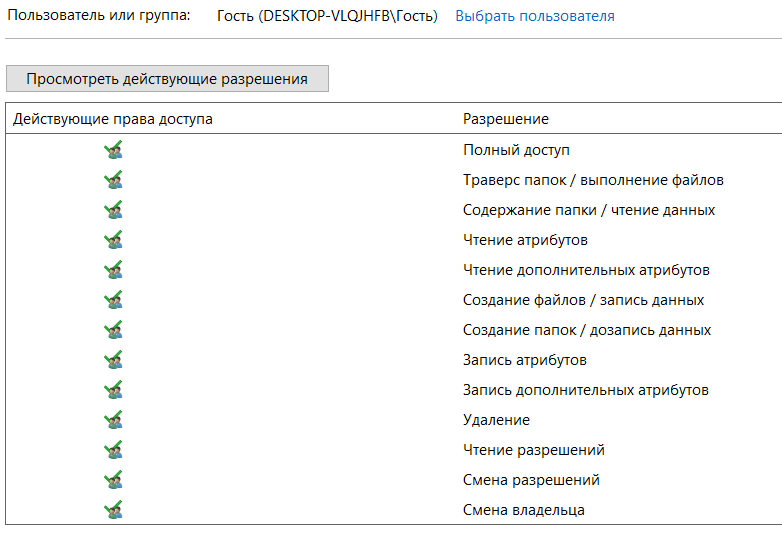


Рисунок 3 — Разрешения для пользователя "Гость"

Теперь запретим группе запись, у пользователя пропали те же права, что и были запрещены группе.

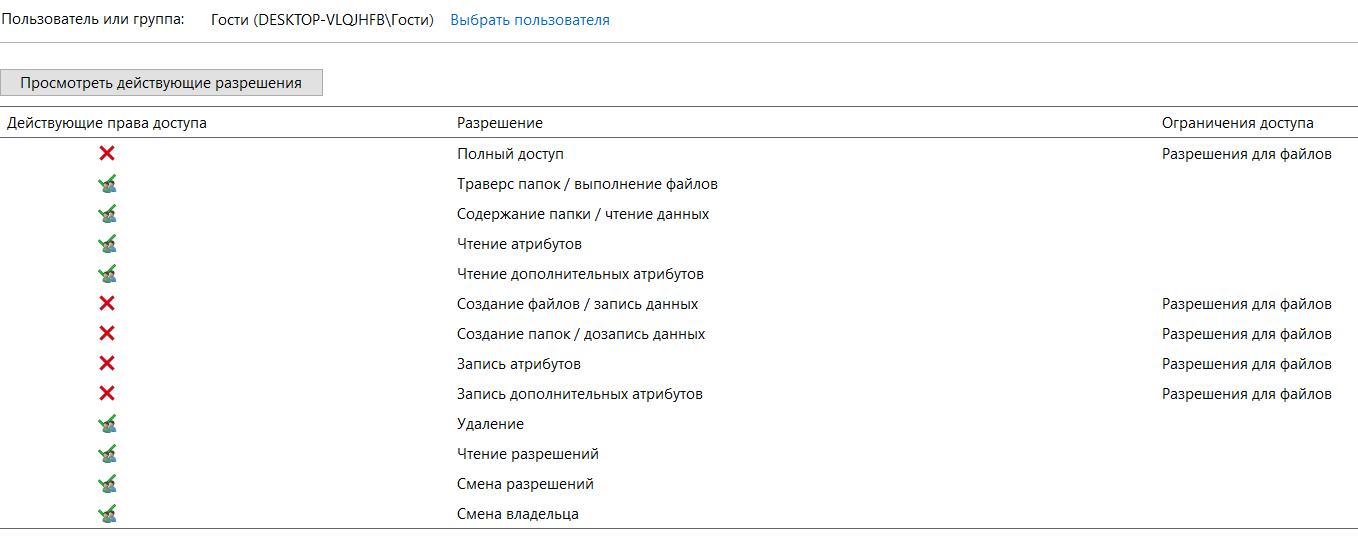


Рисунок 4 — Разрешения для группы "Гости"

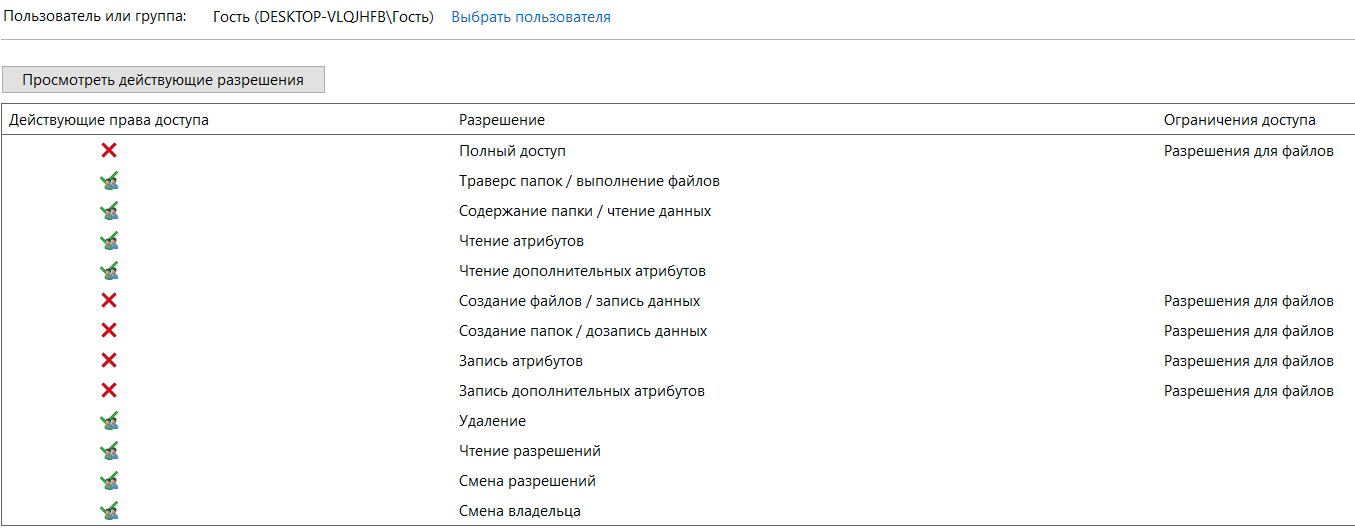


Рисунок 5 — Разрешения для пользователя "Гость"

# **Произведенные модификации в структуре ACL**

Был создан файл “test.txt”, владельцем которого является пользователь “сlowi. Все пользователи имеют полный доступ к данному файлу.

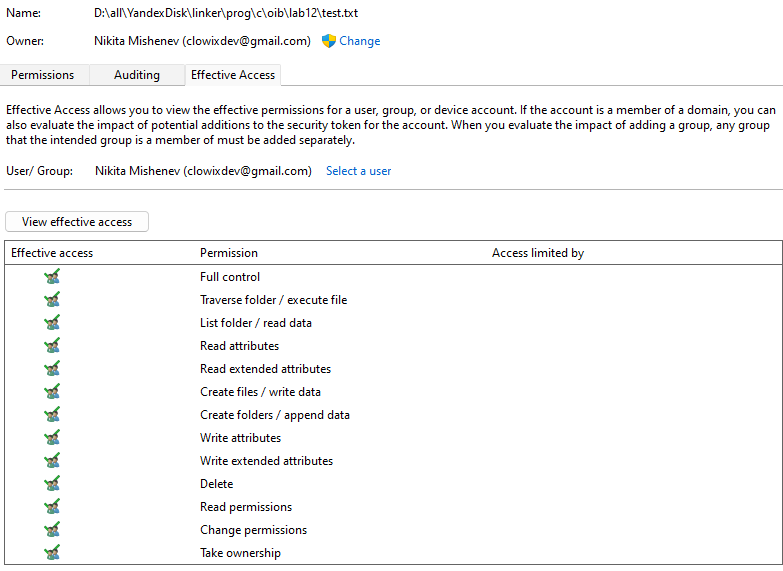


Рисунок 6 — Разрешения для пользователя "clowi"

С помощью разработанной утилиты изменим права пользователя – запретим запись в файл. Права изменятся и в параметрах безопасности.

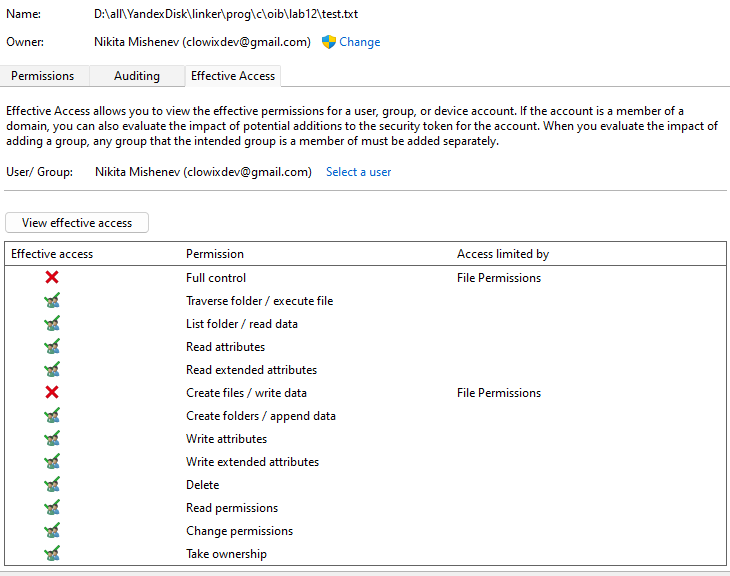


Рисунок 6 — Разрешения для пользователя "clowi" после внесения изменений

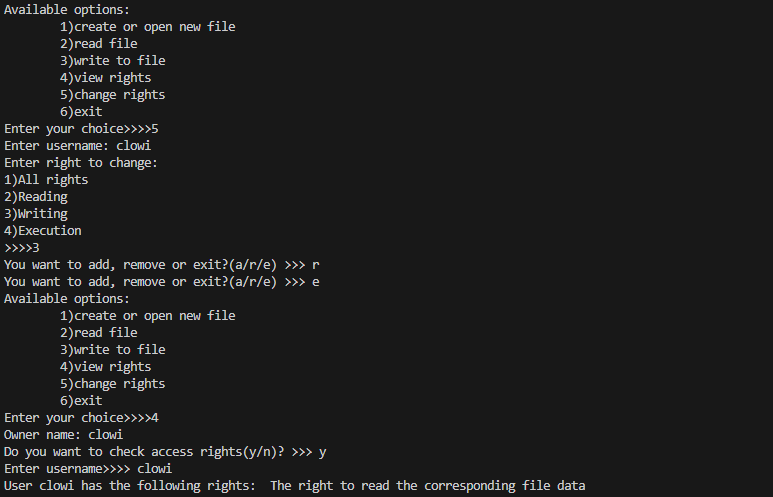


Рисунок 7 — Изменение прав с помощью разработанной утилиты

К редактированию утилитой доступны права записи, чтения, выполнения. Программа сохраняет старый ACL и действует посредством создания нового ACL и присоединения его, как нового объекта DACL.

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое множество действующих разрешений?

Это множество разрешенных пользователю действий, вычисляющихся путем вычитания множества запретов из запрещающих ACE файла или каталога из множества разрешений из разрешающих ACE.

2.Почему проверка прав доступа к файлу осуществляется только при открытии файла, а не при обращениях к нему?

Процесс в течение сеанса работы может осуществлять доступ ко многим объектам, а количество активных процессов и проверяемых ACE в каждый момент времени довольно большое, поэтому монитор безопасности проверяет возможность доступа процесса к объекту только при его открытии(получении идентификатора), а не при каждом обращении.

3. Можно ли запретить администратору системы доступ к какому-либо файлу? Может ли он обойти это ограничение?

Администратору можно запретить доступ к файлу, однако он может с легкостью обойти запреты, сделав себя владельцем файла, что позволит ему дать себе все необходимые права.

4. Как изменить владельца объекта в среде Windows?

Да, для этого необходимо открыть окно дополнительных параметров безопасности файла.

5. Имеет ли владелец какие-ибо права к файлу, если существует ACE, запрещающая полный доступ к этому файлу на име владельца?

Владелец всегда будет иметь права просмотра и изменения ACE.

ВЫВОД

Во время выполнения лабораторной работы мы освоили инструменты контроля и управления доступом пользователей к ресурсам операционной системы. Мы приобрели навыки управления правами файловой системы NTFS и разработали программу для чтения и изменения ACE (Access Control Entries). Мы изучили приоритет запрещающих прав перед разрешающими и провели расчет действующих разрешений для объектов на основе индивидуальных и групповых настроек доступа.

ПРИЛОЖЕНИЕ А

Листинг 1 - исходный код программы на языке С

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <windows.h>

#include <tchar.h>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <aclapi.h>

#define PATH\_LENGTH 260

void clear(char\* arr, int len) {

for (int i = 0; i < len; i++) {

arr[i] = '\0';

}

}

void opening(LPCSTR file\_name)

{

HANDLE file = NULL;

file = CreateFileA(file\_name, FILE\_READ\_ATTRIBUTES | FILE\_WRITE\_ATTRIBUTES, 0, NULL, OPEN\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (file == INVALID\_HANDLE\_VALUE)

{

DWORD error = GetLastError();

printf("Error %d occurred while creating or opening file\n", error);

return;

}

}

void print(LPCSTR file\_name)

{

DWORD error = 0;

PACL pacl = NULL;

PSECURITY\_DESCRIPTOR psd = NULL;

PSID p\_user\_sid;

SID\_NAME\_USE siduse;

ACCESS\_MASK AccessRights = 0;

DWORD dwsize = 260;

char answer = 0;

LPTSTR domain = (LPTSTR)malloc(512), owner = (LPTSTR)malloc(512);

LPTSTR user = (LPTSTR)malloc(260);

PTRUSTEE pTrustee = (PTRUSTEE)malloc(sizeof(TRUSTEE));

if (pTrustee) {

pTrustee->TrusteeForm = TRUSTEE\_IS\_NAME;

pTrustee->TrusteeType = TRUSTEE\_IS\_WELL\_KNOWN\_GROUP;

}

error = GetNamedSecurityInfoA(file\_name, SE\_FILE\_OBJECT, DACL\_SECURITY\_INFORMATION |

OWNER\_SECURITY\_INFORMATION | GROUP\_SECURITY\_INFORMATION, &p\_user\_sid, NULL, &pacl, NULL, &psd);

if (error != ERROR\_SUCCESS)

{

printf("Something went wrong with getting PACL.GetNamedSecurityInfo : %u \n",error);

return;

}

if (!LookupAccountSid(NULL,p\_user\_sid, owner,&dwsize, domain, &dwsize,&siduse))

{

error = GetLastError();

printf("Search by SID was failed. Error code: %d\n", error);

return;

}

answer = 0;

printf("Owner name: %s\nDo you want to check access rights(y/n)? >>> ", owner);

fflush(stdin);

scanf("%c", &answer);

if (answer == 'y')

{

printf("Enter username>>>> ");

getchar();

scanf("%s", user);

if (pTrustee) pTrustee->ptstrName = user;

if (pTrustee)

GetEffectiveRightsFromAcl(pacl,pTrustee,&AccessRights);

printf("User %s has the following rights: ", user);

if (((AccessRights & KEY\_ALL\_ACCESS) == KEY\_ALL\_ACCESS))

{

printf("All possible access rights to the file\n");

}

else if (((AccessRights & KEY\_READ) == KEY\_READ))

printf("The right to read the corresponding file data\n");

else if (((AccessRights & KEY\_WRITE) == KEY\_WRITE))

printf("The right to write data to the file.\n");

else if (((AccessRights & KEY\_EVENT) == KEY\_EVENT))

printf("Right to execute file\n");

else printf("-\n");

}

else if (answer == 'n')

{

printf("OKK :((\n");

return;

}

else

{

printf("error input :((\nTry again.");

return;

}

}

void change(LPTSTR file\_name)

{

DWORD error;

PSECURITY\_DESCRIPTOR presecurity\_descriptor;

TCHAR username[260];

DWORD AccesPermissions[] = { GENERIC\_ALL, FILE\_READ\_DATA, FILE\_WRITE\_DATA, GENERIC\_EXECUTE };

PACL oldacl = NULL, newacl = NULL;

EXPLICIT\_ACCESS explicit\_access;

char answer1=0,answer2=0;

printf("Enter username: ");

scanf("%s", username);

error = GetNamedSecurityInfoA(file\_name, SE\_FILE\_OBJECT, DACL\_SECURITY\_INFORMATION |

OWNER\_SECURITY\_INFORMATION | GROUP\_SECURITY\_INFORMATION, NULL, NULL, &oldacl, NULL, &presecurity\_descriptor);

if (error != ERROR\_SUCCESS)

{

printf("Something went wrong with getting PACL.GetNamedSecurityInfo : %u \n", error);

return;

}

ZeroMemory(&explicit\_access, sizeof(EXPLICIT\_ACCESS));

while (1)

{

printf("Enter right to change:\n1)All rights\n2)Reading\n3)Writing\n4)Execution\n>>>>");

getchar();

scanf("%d", &answer1);

if ((answer1 <= 4 && answer1 >= 1))

{

explicit\_access.grfAccessPermissions = AccesPermissions[answer1 - 1];

while (1)

{

printf("You want to add, remove or exit?(a/r/e) >>> ");

fflush(stdin);

scanf("%c", &answer2);

if (answer2 == 'e') break;

else if (answer2 == 'a') explicit\_access.grfAccessMode = SET\_ACCESS;

else if (answer2 == 'r') explicit\_access.grfAccessMode = DENY\_ACCESS;

else { printf("Wrong input :(\n"); continue; }

explicit\_access.grfInheritance = NO\_INHERITANCE;

explicit\_access.Trustee.TrusteeForm = TRUSTEE\_IS\_NAME;

explicit\_access.Trustee.ptstrName = username;

}

error = SetEntriesInAcl(1, &explicit\_access, oldacl, &newacl);

if (ERROR\_SUCCESS != error)

{

printf("SetEntriesInAcl Error %u\n", error);

return;

}

error = SetNamedSecurityInfo(file\_name, SE\_FILE\_OBJECT,

DACL\_SECURITY\_INFORMATION | PROTECTED\_DACL\_SECURITY\_INFORMATION, NULL, NULL, newacl, NULL);

if (ERROR\_SUCCESS != error) {

printf("SetNamedSecurityInfo Error %u\n", error);

return;

}

return;

}

else

{

printf("Wrong input :(\n");

continue;

}

}

}

void read(char\* file\_name)

{

FILE\* file = fopen(file\_name, "r");

if (file == NULL) printf("The file cannot be read.\n");

else {

char data[200] = { 0 };

fgets(data, 200, file);

printf("You have enough rights to read this file. First 200 symbols of data:\n %s\n", data);

}

}

void write(char\* file\_name)

{

FILE\* file = fopen(file\_name, "w");

if (file == NULL) printf("You have no rights to write in file.\n");

else {

char data[200] = { 0 };

printf("You have enough rights to write in this file. Enter new data:");

fflush(stdin);

fgets(data, 200, stdin);

fputs(data, file);

}

}

int main(void)

{

char file\_path[PATH\_LENGTH] = { 0 };

int option=123456;

printf("Enter file path>>>");

gets(file\_path);

while (1)

{

printf("Available options:\n 1)create or open new file\n 2)read file\n 3)write to file\n 4)view rights\n 5)change rights\n 6)exit\nEnter your choice>>>>");

scanf\_s("%d", &option);

switch (option)

{

case 1:

clear(file\_path, PATH\_LENGTH);

printf("Enter file path>>>");

fflush(stdin);

gets(file\_path);

opening(file\_path);

break;

case 2:

read(file\_path);

break;

case 3:

write(file\_path);

break;

case 4:

print(file\_path);

break;

case 5:

change(file\_path);

break;

case 6:

printf("Shutting down...");

return 0;

default: printf("Error input\n Try again\n");

}

}

}