Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра интеллектуальных информационных технологий

**Отчет по лабораторной работе №7**

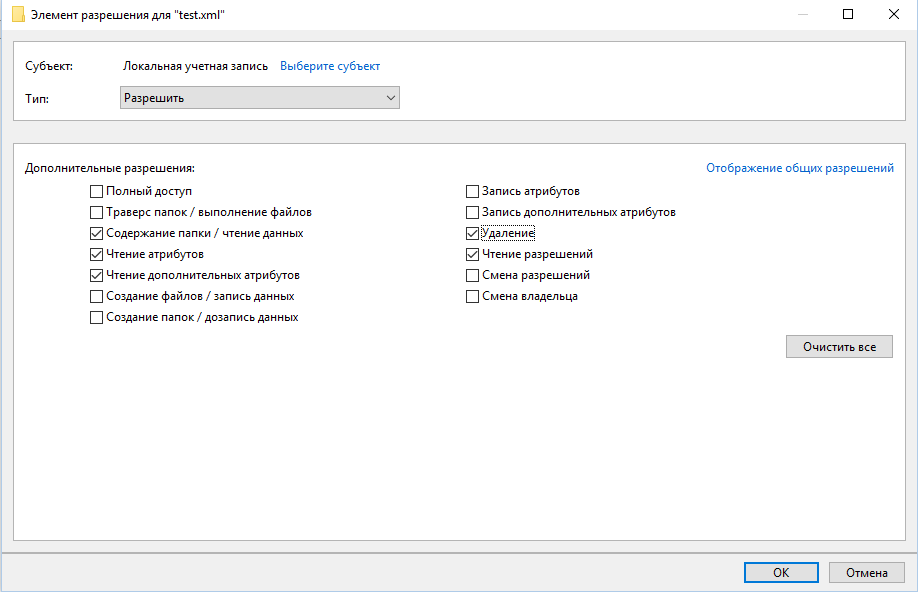
**по курсу «СиМОИБ»**

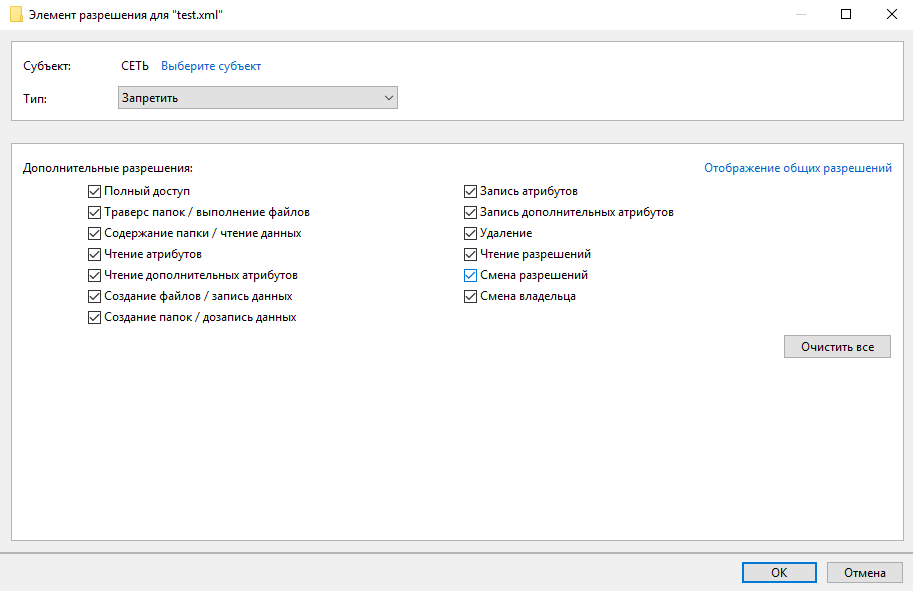
**на тему: «Средства управления доступом в OS Windows»**

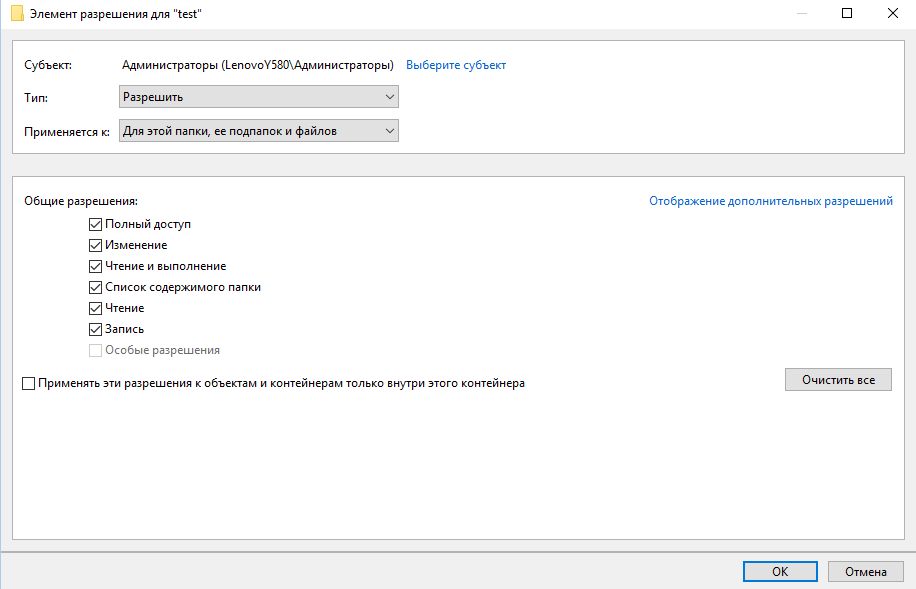
|  |  |
| --- | --- |
| Выполнил студент группы 021702: | Локтев К.А. |
| Проверил: | Колячко В. В. |

Минск 2022

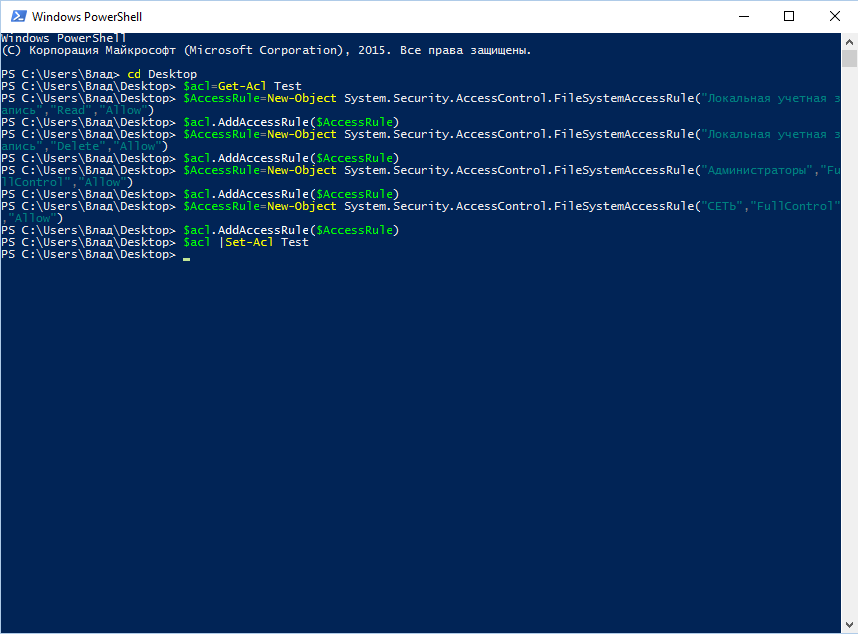
**Задание 1:** Создать файл с различными привилегиями с помощью интерфейса Windows.







**Задание 2:** Создать файл с различными привилегиями с помощью коммандной строки.



Выполняемые действия для того, чтобы дать пользователю какие-либо права на папку(файл).

1. Считываем текущий список ACL из папки Test.

|  |
| --- |
| 1. Создаём переменную с указанием пользователя, прав доступа и типа разрешения. 2. Передаем переменную в класс FileSystemAccessRule для создания объекта. 3. Применяем разрешения к папке. |
|  |

**Задание 3**: Создать файл с различными привилегиями с помощью C++;

void AddFileSecurity(String^ fileName, String^ account,

Security::AccessControl::FileSystemRights rights, Security::AccessControl::AccessControlType controlType)

{

// Получаем текущий FileSecurity объект, который представляет

// текущий список ACL

Security::AccessControl::FileSecurity^ fSecurity = IO::File::GetAccessControl(fileName);

// Добавляем правило с указанием пользователя,

//прав доступа и типа разрешения.

fSecurity->AddAccessRule(gcnew Security::AccessControl::FileSystemAccessRule

(account, rights, controlType));

// Применяем разрешения к папке

IO::File::SetAccessControl(fileName, fSecurity);

}

void RemoveFileSecurity(String^ fileName, String^ account,

Security::AccessControl::FileSystemRights rights, Security::AccessControl::AccessControlType controlType)

{

// Получаем текущий FileSecurity объект, который представляет

// текущий список ACL

Security::AccessControl::FileSecurity^ fSecurity = IO::File::GetAccessControl(fileName);

// Удаляем правило с указанием пользователя,

//прав доступа и типа разрешения.

fSecurity->RemoveAccessRule(gcnew Security::AccessControl::FileSystemAccessRule

(account, rights, controlType));

// Применяем разрешения к папке

IO::File::SetAccessControl(fileName, fSecurity);

}

int main()

{

try

{

String^ fileName = "test.xml";

AddFileSecurity(fileName, "Локальная учетная запись",

Security::AccessControl::FileSystemRights::Read, Security::AccessControl::AccessControlType::Allow);

AddFileSecurity(fileName, "Локальная учетная запись",

Security::AccessControl::FileSystemRights::Delete, Security::AccessControl::AccessControlType::Allow);

AddFileSecurity(fileName, "Администраторы",

Security::AccessControl::FileSystemRights::FullControl, Security::AccessControl::AccessControlType::Allow);

AddFileSecurity(fileName, "СЕТЬ",

Security::AccessControl::FileSystemRights::FullControl, Security::AccessControl::AccessControlType::Deny);

Console::WriteLine("Done.");

}

catch (Exception^ ex)

{ Console::WriteLine(ex->Message); } }

**Выводы:**

Управление доступом в ОС Windows предусмотрено четырьмя разными способами – графическим интерфейсом (GUI), консольным интерфейсом (CLI), с помощью исполняемого файла кода и с помощью редактирования реестров (последнее в лабораторной работе не рассмотрено). Первый способ удобен своей интуитивностью для пользователя и наглядностью, однако не универсален и не всегда содержит полный список разрешений. Написание исполняемого файла кода – вариант получше, однако всё ещё отличается опосредованностью: могут возникнуть проблемы с разрешениями для исполняемого файла, компиляцией кода и т.д. Консольный интерфейс же отвечает большинству требований хорошего интерфейса управления доступом – разрешения для консоли настраиваются прямо в командах с помощью флагов, а язык программирования, используемый в ней, «родной» для Windows, что минимизирует шансы проблем и несостыковок.