

Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему
«Системный монитор доступа к файлам через список
управления доступом»
БГУИР КР 1-40 02 01 305 ПЗ

Студентка:

Гайдина П.А.

Руководитель:

Глоба А.А.

Минск 2022

СОДЕРЖАНИЕ

1	Обзор литературы	4
1.1	Обзор существующих аналогов	4
1.1.1	FileAudit (File and Folder Access Monitoring)	4
1.1.2	FileMon	4
1.2	Постановка задачи	5
2	Системное проектирование	7
2.1	Блок интерфейса	8
2.2	Блок работы приложения	8
3	Функциональное проектирование	9
4.	Разработка программных модулей	12
4.1	проход по заданной директории	12
5.	Руководство пользователя	13
6.	Тестирование	16
	Заключение	19
	Список литературы:	20

ВВЕДЕНИЕ

Системный монитор — это аппаратный или программный компонент, используемый для мониторинга системных ресурсов и производительности в компьютерной системе.

Среди проблем управления, связанных с использованием инструментов системного мониторинга, — использование ресурсов, конфиденциальность и доступ к файлам.

Программные мониторы встречаются чаще, иногда как часть механизма виджетов. Эти системы мониторинга часто используются для отслеживания системных ресурсов, таких как использование и частота ЦП или объем свободной оперативной памяти. Они также используются для отображения таких элементов, как свободное место на одном или нескольких жестких дисках, температура процессора и других важных компонентов, а также сетевой информации, включая IP-адрес системы и текущую скорость загрузки и выгрузки. Другие возможные отображения могут включать дату и время, время безотказной работы системы, имя компьютера, имя пользователя, жесткий диск S.M.A.R.T. данные, скорость вращения вентилятора и напряжение, обеспечиваемое блоком питания.

Реже встречаются аппаратные системы, отслеживающие подобную информацию. Обычно они занимают один или несколько отсеков для дисков на передней панели корпуса компьютера и либо напрямую взаимодействуют с аппаратным обеспечением системы, либо подключаются к программной системе сбора данных через USB. При любом подходе к сбору данных система мониторинга отображает информацию на небольшой ЖК-панели или на серии небольших аналоговых или цифровых светодиодных дисплеев. Некоторые аппаратные системные мониторы также позволяют напрямую управлять скоростью вращения вентилятора, что позволяет пользователю быстро настроить охлаждение в системе.

Несколько очень дорогих моделей аппаратных системных мониторов разработаны для взаимодействия только с определенной моделью материнской платы. Эти системы напрямую используют датчики, встроенные в систему, предоставляя более подробную и точную информацию, чем обычно предоставляют менее дорогие системы мониторинга.

Системный монитор доступа к файлам через список управления доступом представляет собой программный компонент, содержащий в себе информацию о параметрах доступа к различным файлам.

1 Обзор литературы

1.1 Обзор существующих аналогов

Прежде, чем начать выполнение курсового проекта, были проанализированы уже существующие утилиты для поиска одинаковых файлов на диске.

1.1.1 FileAudit (File and Folder Access Monitoring)

FileAudit позволяет анализировать доступ к файлам, хранящимся как локально, так и в облаке, и использовать их. С помощью данной утилиты можно планировать централизованные отчеты в соответствии с несколькими критериями.

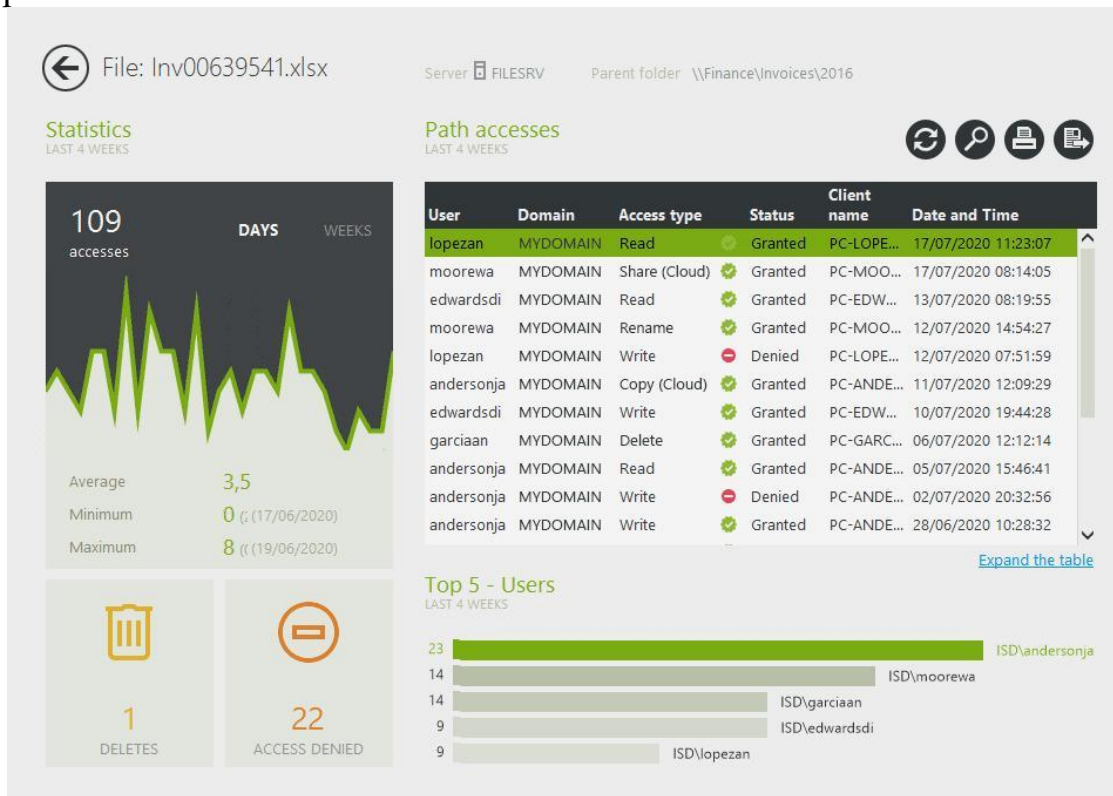


Рисунок 1.1 – Скриншот работы программы FileAudit

1.1.2 FileMon

Filemon следит за работой и контролирует активность всей файловой системы Microsoft Windows, а также отображает все происходящие процессы в реальном времени.

При помощи утилиты пользователь может легко увидеть, какие программы или динамические библиотеки работают, исправить некоторые

критические ошибки в системных файлах, к тому же Filemon не только показывает журнал событий и изменений, которые происходят, но и сообщает об их удалении, открытии, чтении, записи файла или каталога. Таким образом, может обнаружить шпионов и проследить за их деятельностью.

Среди прочей базовой функциональности, в технические возможности включены поиск, сохранение полученных результатов в файл, установка фильтров для исследования операционной системы.

The screenshot shows the File Monitor application window with the title bar 'File Monitor - Sysinternals: www.sysinternals.com'. The menu bar includes 'File', 'Edit', 'Options', 'Volumes', and 'Help'. The toolbar contains icons for file operations and monitoring. The main window displays a table of system events.

#	Time	Process	Request	Path	Result	Other
479	23:23:56	SVCHOST...	OPEN	C:\	SUCCESS	Options: Open Direct
480	23:23:56	SVCHOST...	DIRECTORY	C:\	SUCCESS	FileBothDirectoryInfo
481	23:23:56	SVCHOST...	CLOSE	C:\	SUCCESS	
482	23:23:56	SVCHOST...	OPEN	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Options: Open Acce
483	23:23:56	SVCHOST...	QUERY INFORMATION	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	FileBasicInformation
484	23:23:56	SVCHOST...	SET INFORMATION	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	FileBasicInformation
485	23:23:56	SVCHOST...	QUERY INFORMATION	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Length: 15742
486	23:23:56	SVCHOST...	READ	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Offset: 0 Length: 256
487	23:23:56	SVCHOST...	READ	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Offset: 15230 Length
488	23:23:56	SVCHOST...	READ	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Offset: 14768 Length
489	23:23:56	SVCHOST...	READ	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Offset: 1099 Length:
490	23:23:56	SVCHOST...	READ	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Offset: 7803 Length:
491	23:23:56	SVCHOST...	READ	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Offset: 6742 Length:
492	23:23:56	SVCHOST...	READ	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Offset: 9598 Length:
493	23:23:56	SVCHOST...	READ	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Offset: 14931 Length
494	23:23:56	SVCHOST...	READ	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Offset: 14912 Length
495	23:23:56	SVCHOST...	READ	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Offset: 9153 Length:
496	23:23:56	SVCHOST...	READ	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Offset: 7172 Length:
497	23:23:56	SVCHOST...	READ	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Offset: 0 Length: 512
498	23:23:56	SVCHOST...	QUERY INFORMATION	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Length: 15742
499	23:23:56	SVCHOST...	READ	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Offset: 512 Length: 5
500	23:23:56	SVCHOST...	READ	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Offset: 0 Length: 2
501	23:23:56	SVCHOST...	QUERY INFORMATION	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Length: 15742
502	23:23:56	SVCHOST...	QUERY INFORMATION	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Length: 15742
503	23:23:56	SVCHOST...	QUERY INFORMATION	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Length: 15742
504	23:23:56	SVCHOST...	READ	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Offset: 0 Length: 409
505	23:23:56	SVCHOST...	READ	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Offset: 11646 Length
506	23:23:56	SVCHOST...	READ	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Offset: 0 Length: 409
507	23:23:56	SVCHOST...	QUERY INFORMATION	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Length: 15742
508	23:23:56	SVCHOST...	CLOSE	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	
509	23:23:56	SVCHOST...	OPEN	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	Options: Open Acce
510	23:23:56	SVCHOST...	QUERY INFORMATION	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	FileBasicInformation
511	23:23:56	SVCHOST...	CLOSE	C:\WINDOWS\Prefetch\FILEMON.EX...	SUCCESS	
512	23:23:56	navapvc...	DIRECTORY	C:\Programme\Gemeinsame Dateien\...		Change Notify

Рисунок 1.2 – Скриншот работы программы FileMon

Из минусов данной программы можно назвать устаревший внешний вид и не поддерживаемость на современных операционных системах.

1.2 Постановка задачи

После рассмотрения аналогов можно сказать, что все они обладают большим количеством функций, которые невозможно реализовать в курсовом проекте за данный период времени. Поэтому были выбраны несколько ключевых возможностей, которые будут выполнены в рамках одного семестра:

- Программа должна иметь удобный пользовательский интерфейс.
- В программе должно быть реализовано сканирование файлов.
- В программе должен быть вывод параметров доступа к файлам.

В качестве языка программирования выбран C++, по причине быстрой производительности, требуемой для обработки большого количества объектов, поддержки объектно-ориентированного подхода к программированию и наличия опыта в использования данного языка. Для создания удобного пользовательского интерфейса был выбран Фреймворк Qt

2 Системное проектирование

В GNU/Linux как и других Unix-подобных операционных системах понятие типа файла не связано с расширением файла (несколькими буквами после точки в конце имени), как это обстоит в Windows.

Unix-подобная ОС не следит за расширениями файлов. Задача связать расширения файла с конкретным пользовательским приложением, в котором этот файл будет открываться, видимо лежит на какой-либо дополнительной программе. В свою очередь пользовательское приложение анализирует структуру данный файла, расширение ему также безразлично.

Таким образом, среди файловых атрибутов, хранящихся в операционной системе на базе ядра Linux, нет информации о типе данных в файле. Там есть информация о более существенном разделении, связанном с тем, что в Unix-подобных системах все объекты – это файлы. Все объекты весьма разнообразны. Поэтому тип файла в Linux – это скорее тип объекта, но не тип данных как в Windows.

В операционной системе GNU/Linux существуют следующие типы файлов: обычные файлы, каталоги, символьные ссылки, блочные устройства, символьные устройства, сокеты, каналы. Каждый тип имеет собственное обозначение одним символом.

После определения требований к функционалу разрабатываемого приложения его следует разбить на функциональные блоки. Такой подход упростит понимание проекта, позволит устранить проблемы в архитектуре, обеспечит гибкость и масштабируемость программного продукта в будущем путем добавления новых блоков.

Для начала следует понять, какие функции должен выполнять наш системный монитор. В первую очередь, конечно же, должен выводиться весь список файлов, и папок, содержащихся в корневой директории. Более того, исходя из названия темы следует, должны выводиться права доступа для каждого файла, а именно доступ к чтению, записи в файл, запуска данной программы. Также требуется функционал, позволяющий переходить по всем папкам. Для более удобного пользования программой, требуется реализовать возможность перехода в предыдущую папку и возврат в корневую папку.

В операционной системе Linux есть много отличных функций безопасности, но она из самых важных — это система прав доступа к файлам. Каждый файл имеет три категории доступа: владелец, группа и остальные. Владелец — набор прав для владельца файла, пользователя, который его создал или сейчас установлен его владельцем. Обычно владелец имеет все права, чтение, запись и выполнение. Группа — любая группа пользователей, существующая в системе и привязанная к файлу. Но это может быть только одна группа и обычно это группа владельца, хотя для файла можно назначить и другую группу. Остальные — все пользователи, кроме владельца и пользователей, входящих в группу файла.

Каждый файл имеет три параметра доступа. Чтение — разрешает получать содержимое файла, но на запись нет. Для каталога позволяет получить список файлов и каталогов, расположенных в нем. Запись — разрешение на запись новых данных в файл или изменение существующих, а также возможность создавать и изменять файлы и каталоги; Выполнение — нельзя выполнить программу, если у нее нет флага выполнения. Этот атрибут устанавливается для всех программ и скриптов, именно с помощью него система может понять, что этот файл нужно запускать как программу.

2.1 Блок интерфейса

Блок пользовательского интерфейса предназначен для взаимодействия пользователя с приложением, основываясь на представлении и визуализации данных.

Для разрабатываемого проекта создается простейший пользовательский интерфейс с помощью фреймворка Qt. Данный пользовательский интерфейс представляет собой набор пунктов меню, с которыми пользователь может взаимодействовать с помощью клавиатуры.

2.2 Блок работы приложения

Блок работы приложения необходим для взаимодействия пользователя с приложением, а также обработки сигналов других блоков и является неотъемлемой его частью. Тут осуществляется весь необходимый функционал и возможности:

- Выбор начальной директории;
- Просмотр обработанных файлов в режиме реального времени;
- Вывод параметров доступа к файлам для текущего пользователя;

Блок сканирования файлов

Блок сканирования файлов предназначен для прохода по всем, начиная с заданной пользователем, директориям и поддиректориям с целью занесения файлов в массив и дальнейшего его использования.

3 Функциональное проектирование

В данном разделе будет рассмотрен функционал разрабатываемого ПО.

3.1 main.cpp

directoryBrowsing. Функция является ключевой в данном курсовом проекте. Она представляет собой набор команд для прохождения по заданной в параметре директории с сопутствующим занесением файлов и папок в определенные массивы. Массивы char** directoriesArray и char** filesArray нужны для последующей итерации и перемещению по файлам системы.

Данная функция формирует путь для каждого файла, требуемый будущим проходом по вложенным директориям вглубь корневой системы Linux.

```
unsigned    directoryBrowsing(string    introducedDir,    string*
&directoriesArray, string* &filesArray, string* &permissionArray,
            int    &numberOfDirectories,    int    &numberOfFiles,    int
&numberOfPermission) {
    int accessParameters;
    DIR *dir = NULL;
    string* temp;
    string resultPermission;
    struct dirent *entry = NULL;
    string pathName;
    mode_t status;
    dir = opendir(introducedDir.c_str());
    if( dir == NULL ) {
        cout << "Error opening " << introducedDir << ":" <<strerror(errno);
        return 0;
    }
    entry = readdir(dir);
    while(entry != NULL) {
        struct stat entryInfo;
        if((strncmp(entry->d_name, ".", PATH_MAX) == 0) ||
            (strncmp(entry->d_name, "..", PATH_MAX) == 0)) {
            entry = readdir(dir);
            continue;
        }
        pathName = introducedDir;
        pathName += "/";
        pathName += entry->d_name;
        if(lstat(pathName.c_str(), &entryInfo) == 0) {
            if(S_ISDIR(entryInfo.st_mode)) {
                directoriesArray[numberOfDirectories] = pathName;
                numberOfDirectories++;
            }
        }
    }
}
```

```

        temp = new string[numberOfDirectories + 1];
        for (int i = 0; i < numberOfDirectories; i++) {
            temp[i] = directoriesArray[i];
        }
        directoriesArray = temp;
        resultPermission = string(displayPermission(entryInfo.st_mode));
        permissionArray[numberOfPermission] = resultPermission;
        numberOfPermission++;
        temp = new string[numberOfPermission + 1];
        for (int i = 0; i < numberOfPermission; i++){
            temp[i] = permissionArray[i];
        }
        permissionArray = temp;
    }
    else if(S_ISREG(entryInfo.st_mode)) {
        filesArray[numberOfFiles] = pathName;
        numberOfFiles++;
        temp = new string[numberOfFiles + 1];
        for (int i = 0; i < numberOfFiles; i++) {
            temp[i] = filesArray[i];
        }
        filesArray = temp;
        resultPermission = string(displayPermission(entryInfo.st_mode));
        permissionArray[numberOfPermission] = resultPermission;
        numberOfPermission++;
        temp = new string[numberOfPermission + 1];
        for (int i = 0; i < numberOfPermission; i++){
            temp[i] = permissionArray[i];
        }
        permissionArray = temp;
        cout <<"\t" <<pathName << "\t\t\t\n";
    }
    else if(S_ISLNK(entryInfo.st_mode)) {
        char targetName[PATH_MAX + 1];
        if(readlink(pathName.c_str(), targetName, PATH_MAX) != -1) {
            filesArray[numberOfFiles] = pathName;
            numberOfFiles++;
            temp = new string[numberOfFiles + 1];
            for (int i = 0; i < numberOfFiles; i++) {
                temp[i] = filesArray[i];
            }
            filesArray = temp;
        }
    }
}

```

```

    }
}
entry = readdir(dir);
}
(void)closedir(dir);
return 0;
}

```

displayPermission.

Не менее важная функция в данном проекте, которая отвечает непосредственно за считывание и вывод прав доступа файла для данного пользователя, группы и всех остальных. В данной функции используется поле структуры stat. Структура stat полезна для работы с файлами, в частности для получения информации о самом файле. К примеру поле st_mode, которое используется в функции displayPermission, содержит в себе информацию о правах доступа к файлу.

```

char* display_permission ( int st_mode ) {
    static const char xtbl[10] = "rwxrwxrwx";
    char amode[10];
    int i, j;

    for ( i = 0, j = ( 1 << 8 ); i < 9; i++, j >>= 1 )
        amode[i] = ( st_mode&j ) ? xtbl[i]: '-';
    if ( st_mode & S_ISUID ) amode[2]= 's';
    if ( st_mode & S_ISGID ) amode[5]= 's';
    if ( st_mode & S_ISVTX ) amode[8]= 't';
    amode[9]='\0';
    return amode;
}

```

4. Разработка программных модулей

В данном разделе будет рассмотрен алгоритм, используемый в данном системном ПО.

4.1 проход по заданной директории

Структурная схема представлена в Приложении А.

Шаг 1. Начало алгоритма.

Шаг 2. Объявление и инициализация необходимых переменных.

Шаг 3. Открытие потока каталога.

Шаг 4. Проверка на существовании директории, переданной в параметре функции.

Шаг 5. Если выражение (`dir == NULL`) истинно, вывести сообщение об ошибке, завершить программу.

Шаг 6. Если выражение (`dir == NULL`) ложно, продолжить выполнение программы.

Шаг 7. Чтение оглавление каталога `dir`.

Шаг 8. Начало цикла.

Шаг 9. Объявление структуры `stat`.

Шаг 10. Проверка на чтение из головной директории.

Шаг 11. Если выражение истинно, тогда чтение следующего оглавления каталога `dir`.

Шаг 12. Если выражение ложно, продолжить выполнение программы.

Шаг 13. Формирование полной директории к файлу.

Шаг 14. Проверка на получение информации о файле.

Шаг 15. Если выражение ложно, выводится ошибка, чтение следующего оглавления каталога.

Шаг 16. Если выражение истинно, проверяется тип файла.

Шаг 17. Если это папка, занести директорию в массив `directoriesArray`.

Шаг 18. Занести права доступа к папке в массив `permissionArray`.

Шаг 19. Если это обычный файл, занести директорию в массив `filesArray`.

Шаг 20. Занести права доступа к файлу в массив `permissionArray`.

Шаг 21. Если это ссылка, занести директорию в массив `filesArray`.

Шаг 22. Занести права доступа к ссылке в массив `permissionArray`.

Шаг 23. Чтение оглавления каталога.

Шаг 24. Проверка условия цикла.

Шаг 25. Если директория прочитана не полностью перейти на Шаг 9.

Шаг 26. Если директория прочитана полностью, выйти из цикла.

Шаг 28. Закрытие потока каталога.

Шаг 29. Конец алгоритма.

5. Руководство пользователя

Требования для пользования программой: ОС — Linux, наличие QT на компьютере.

Функционал программы: данное приложение создано для вывода доступа к файлам всей файловой системы. В приложении создан удобный и интуитивный интерфейс. В программе можно перемещаться по всем папкам, выбирать интересующую категорию. На следующем скриншоте представлен общий вид приложения.

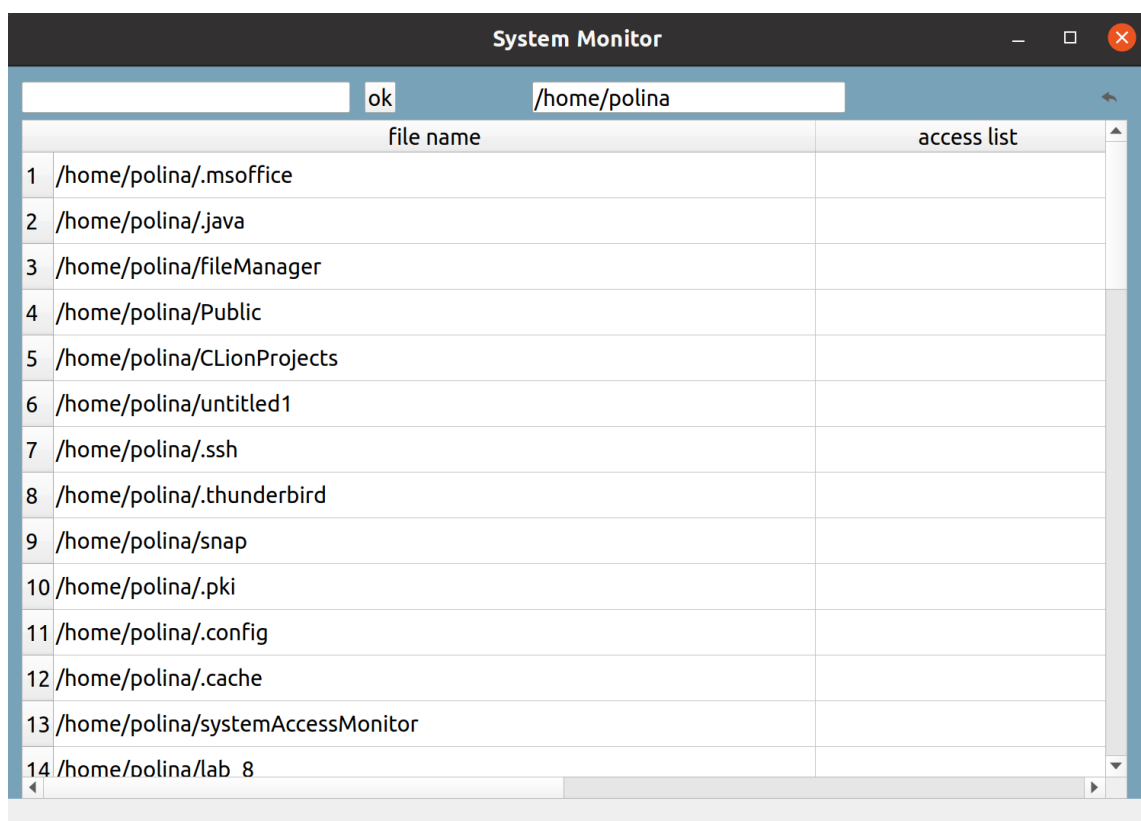


Рисунок 5.1 – Общий вид приложения

В правом столбце отображаются права доступа для разных пользователей, в левом – список файлов в домашней папке (начальная директория).

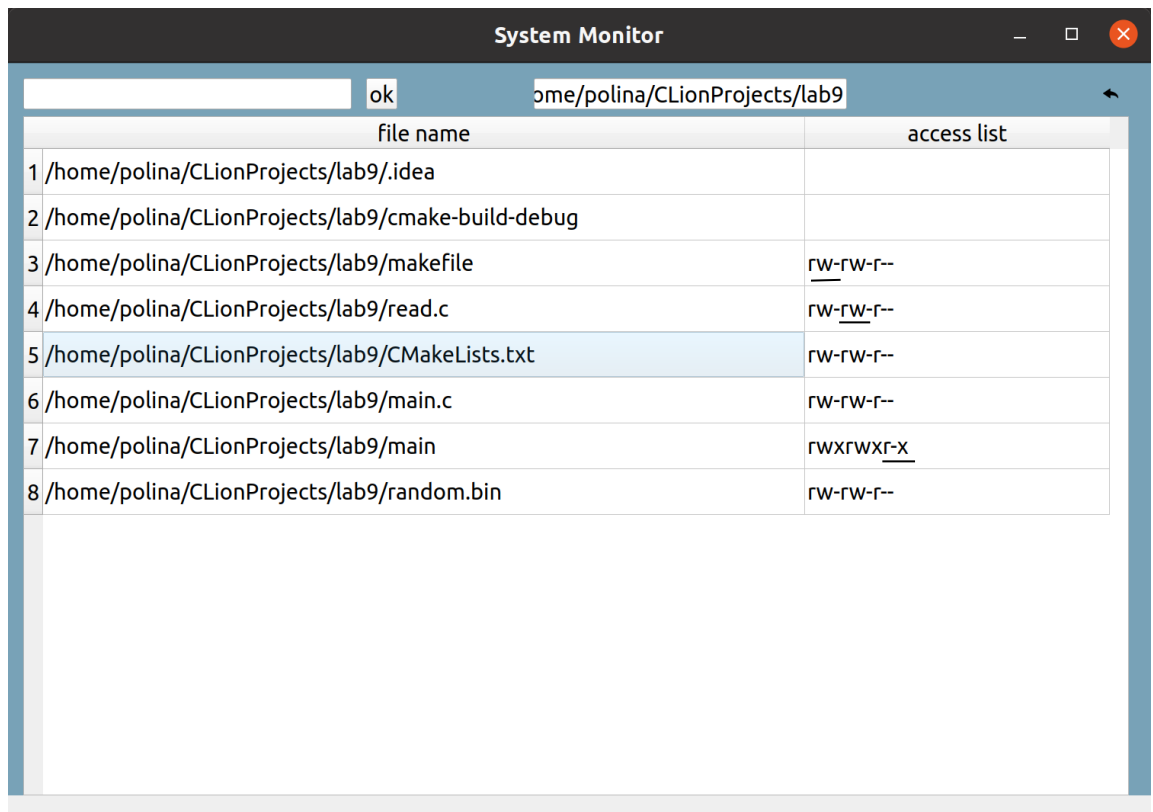


Рисунок 5.2 – Переход по заданной директории через поле поиска

Первые 3 символа обозначают права доступа для владельца данного файла. Обозначения являются стандартными для операционной системы Linux. Символ «r» показывает, имеет ли пользователь доступ для чтения данного файла, символ «w» отвечает за возможность изменения файла и наконец символ «x» показывает, имеет ли пользователь возможность запускать данный файл. Следующие три символа показывают те же самые права доступа, однако не для владельца, а для группы. И последние 3 символа показывают доступ к файлу для всех остальных пользователей.

Более того, в программе реализована возможность перехода по любой директории из текущей директории.

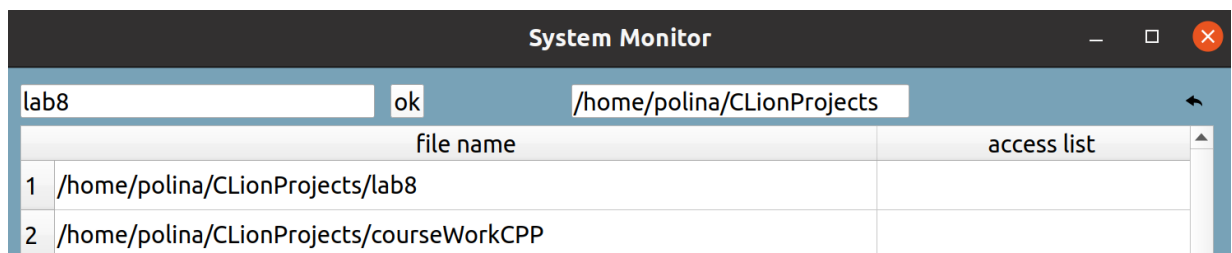


Рисунок 5.3 – Выбор директории, по которой можно перейти

Так же можно увидеть текущую директорию, путь к ней и непосредственно само содержимое директории.

Так же реализована возможность выбора папки из списка простым нажатием левой клавиши компьютерной мыши.

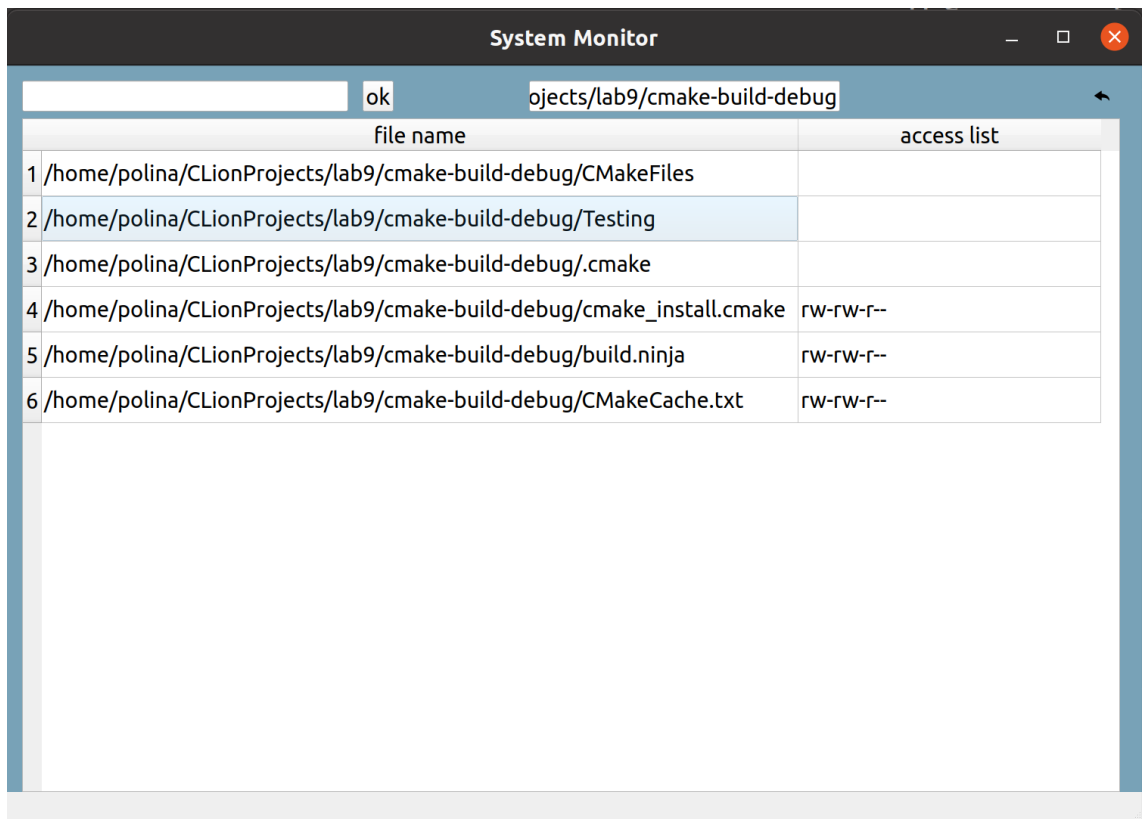


Рисунок 5.4 – Выбор папки из списка при помощи мыши

6. Тестирование

Для начала, следует протестировать правильность работы программы.

В первую очередь попробуем перейти по выбранной папке из общего списка директорий.

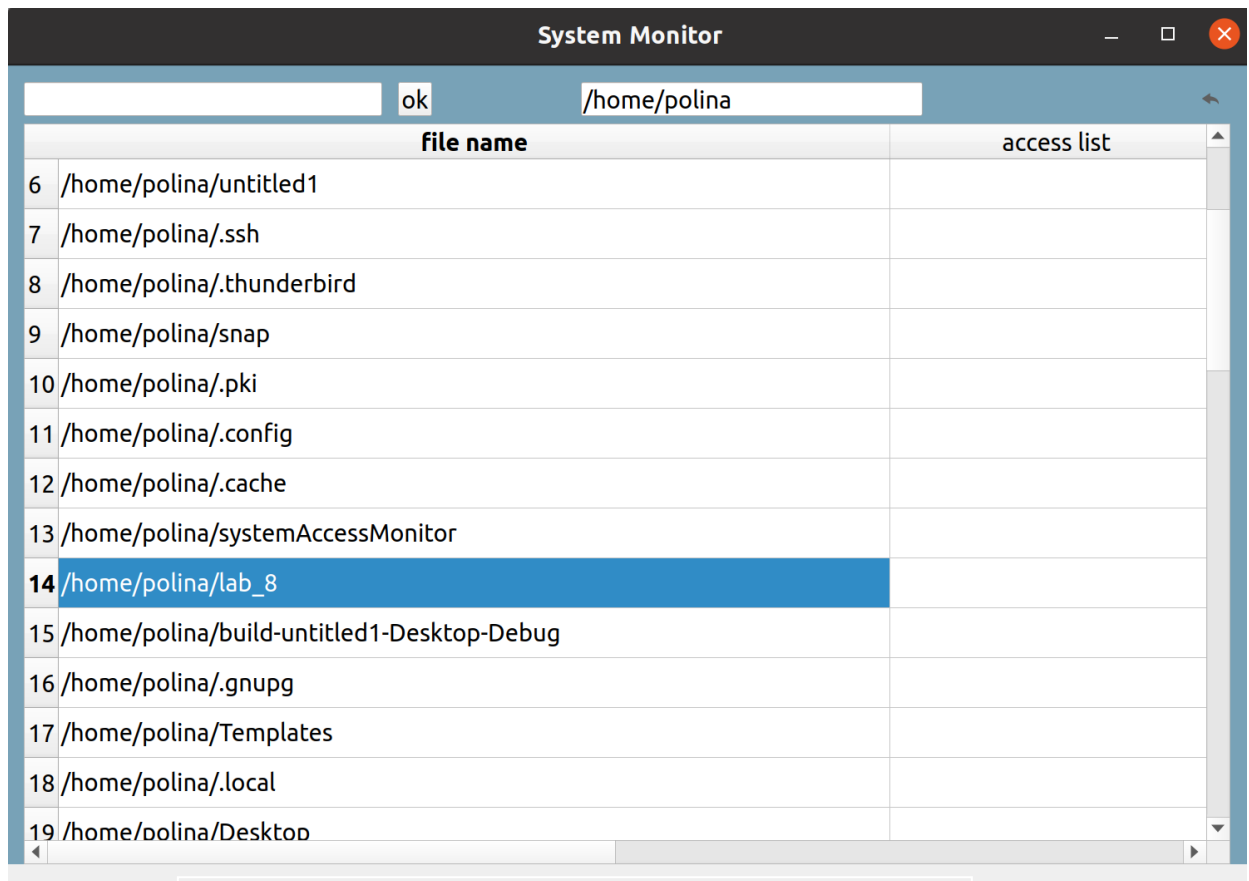
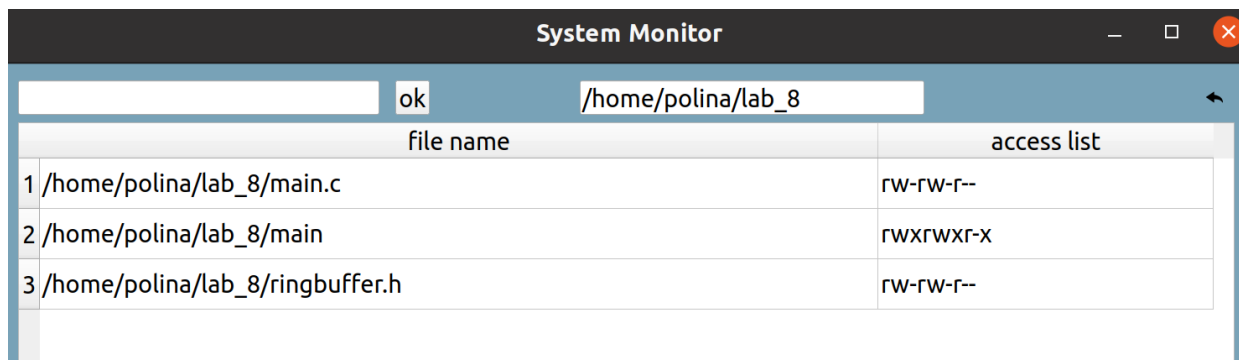


Рисунок 6.1 – Тестовый переход по выбранной папке

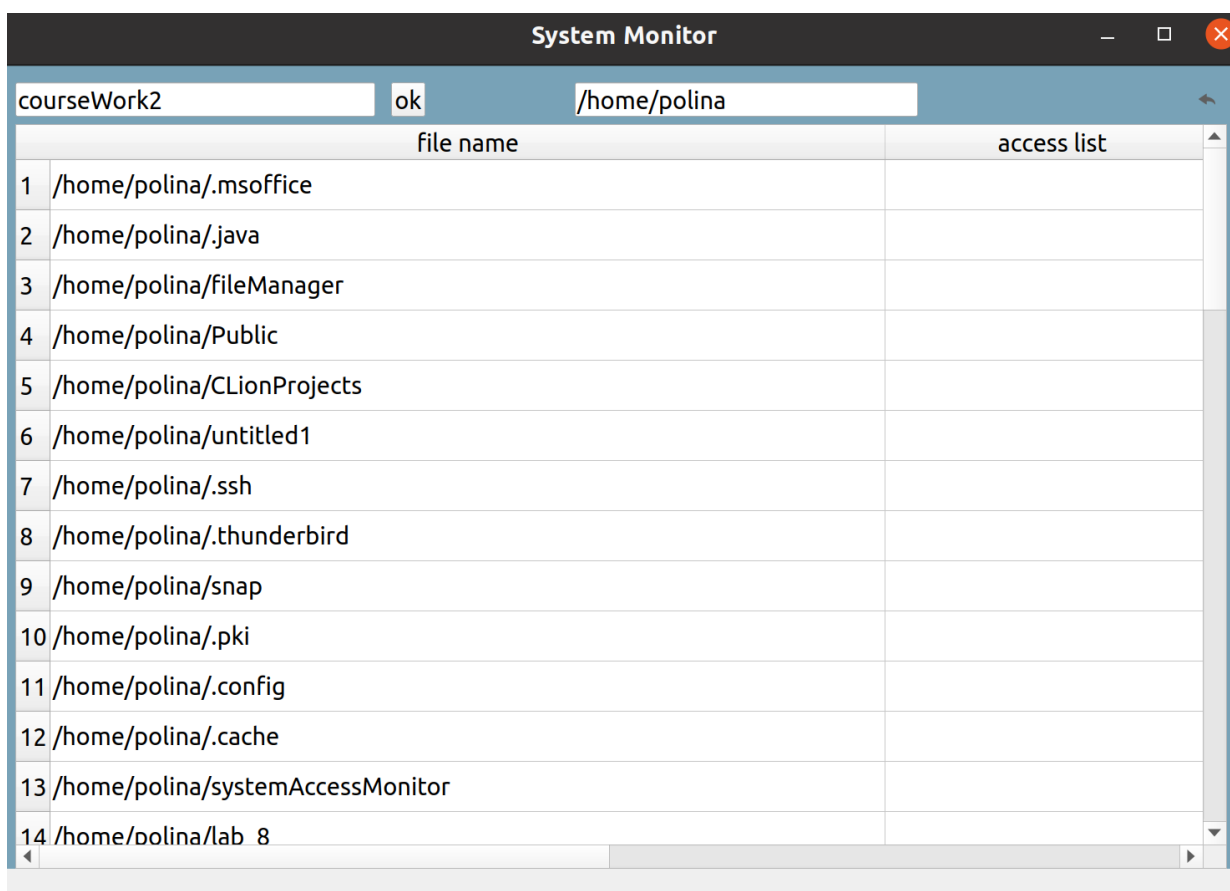
После выбора папки lab8 на экране видим следующее:



	file name	access list
1	/home/polina/lab_8/main.c	rw-rw-r--
2	/home/polina/lab_8/main	rw-rw-r-x
3	/home/polina/lab_8/ringbuffer.h	rw-rw-r--

Рисунок 6.2 – Содержимое папки lab8

Следовательно, данный функционал является тоже полностью рабочим. Далее, протестируем функцию поиска папки по заданному имени.



	file name	access list
1	/home/polina/.msoffice	
2	/home/polina/.java	
3	/home/polina/fileManager	
4	/home/polina/Public	
5	/home/polina/CLionProjects	
6	/home/polina/untitled1	
7	/home/polina/.ssh	
8	/home/polina/.thunderbird	
9	/home/polina/snap	
10	/home/polina/.pki	
11	/home/polina/.config	
12	/home/polina/.cache	
13	/home/polina/systemAccessMonitor	
14	/home/polina/lab 8	

Рисунок 6.3 – Тестовый переход в папку через поле поиска

После перехода на экране увидим следующее:

System Monitor	
<div> <input type="text"/> <input type="button" value="ok"/> <input type="text" value="/home/polina/courseWork2"/> </div>	
file name	access list
1 /home/polina/courseWork2/.git	
2 /home/polina/courseWork2/functions.cpp	rw-rw-r--
3 /home/polina/courseWork2/functions.h	rw-rw-r--
4 /home/polina/courseWork2/mainwindow.cpp	rw-rw-r--
5 /home/polina/courseWork2/courseWork2.pro	rw-rw-r--
6 /home/polina/courseWork2/courseWork2.pro.user	rw-rw-r--
7 /home/polina/courseWork2/mainwindow.ui	rw-rw-r--
8 /home/polina/courseWork2/main.cpp	rw-rw-r--
9 /home/polina/courseWork2/mainwindow.h	rw-rw-r--

Рисунок 6.4 – Результат перехода по папке

Заключение

В ходе работы над курсовым проектом был разработан системный монитор доступа к файлам через список управления доступом для операционной системы Linux. Процесс работы описан в данной пояснительной записке.

Системный монитор отображает список файлов и права доступа к ним. Так же возможен проход по различным папкам для дальнейшего прохода по директориям.

В дальнейшем планируется расширить функционал программы и добавить возможность изменять права доступа к файлам по типу команды `chmod` в терминале Linux.

В ходе работы автор изучил много статей как на русском языке, так и на английском в поисках полезной информации для реализации программы, а также для понятного пользовательского интерфейса был глубже изучен фреймворк QT.

Список литературы:

- [1] Описание системного монитора. – [Электронный ресурс]. – Адрес ресурса: https://en.wikipedia.org/wiki/System_monitor - Дата доступа 14.03.2022
- [2] Э. Таненбаум Т. Остин «Архитектура компьютера» 6-е издание, 2013 год
- [3] Роберт Лав «Linux Системное программирование» 2-е издание, 2014 год