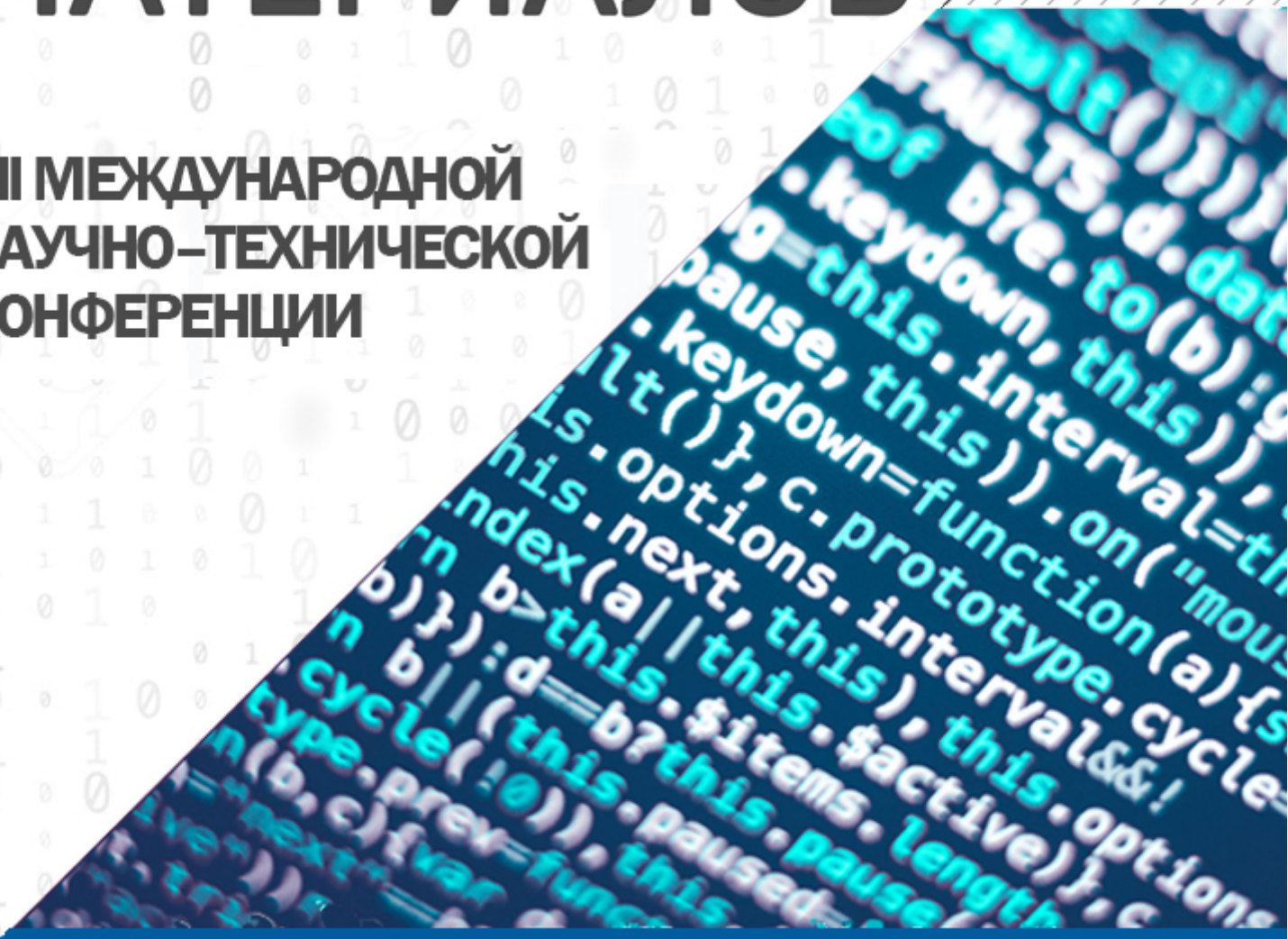
Донецкий Национальный Технический университет | Факультет Компьютерных Наук и Технологий

100 лет ДОННТУ

СБОРНИК МАТЕРИАЛОВ

XII МЕЖДУНАРОДНОЙ НАУЧНО-ТЕХНИЧЕСКОЙ КОНФЕРЕНЦИИ



ИУС МКМ

26–27 мая 2021

г. Донецк

В РАМКАХ VII МЕЖДУНАРОДНОГО НАУЧНОГО ФОРУМА ДОНЕЦКОЙ НАРОДНОЙ РЕСПУБЛИКИ

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДОНЕЦКОЙ НАРОДНОЙ РЕСПУБЛИКИ

ГОУ ВПО «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ НАУК И ТЕХНОЛОГИЙ

ИНФОРМАТИКА, УПРАВЛЯЮЩИЕ СИСТЕМЫ, МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ (ИУСМКМ-2021)

Материалы XII Международной научно-технической конференции в рамках VII Международного Научного форума Донецкой Народной Республики к 100-летию ДонНТУ

26–27 мая 2021 г.

г. Донецк, ДОННТУ – 2021

УДК 004

ИУСМКМ-2021 : материалы XII Международной научно-технической конференции «Информатика, управляющие системы, математическое и компьютерное моделирование» / Донецкий нац. техн. ун-т ; сост.: А. И. Воронова, Т. А. Васяева ; под ред. Р. В. Мальчевой. – Донецк : ДОННТУ, 2021. – 149 с.

Сборник подготовлен по результатам XII Международной научно-технической конференции «Информатика, управляющие системы, математическое и компьютерное моделирование», проведенной в рамках VII Международного Научного форума Донецкой Народной Республики.

Организаторами конференции выступили Министерство образования и науки ДНР; ГОУ ВПО «Донецкий национальный технический университет» (ДонНТУ), факультет компьютерных наук и технологий (ФКНТ), кафедра автоматизированных систем управления (АСУ); ФГАОУ ВО «Национальный исследовательский университет “Московский институт электронной техники”» (НИУ «МИЭТ»); ФГБОУ ВО «Кубанский государственный университет» (КубГУ); ФГАОУ ВО «Севастопольский государственный университет» (СевГУ).

Материалы, вошедшие в сборник, представлены научно-педагогическими сотрудниками, аспирантами, магистрантами и студентами высших учебных заведений из России, Беларуси, ДНР и ЛНР.

Рекомендовано к публикации на заседании Ученого совета ФКНТ ДонНТУ.
Протокол № 5 от «18» июня 2021 г.

Организационный комитет:
Аноприенко А. Я., к. т. н., проф., ректор ДОННТУ; **Николаенко Д. В.**, к. т. н., доц., декан ФКНТ ДОННТУ; **Секирин А. И.**, к. т. н., доц., зав. каф. АСУ ДОННТУ; **Кожухов И. Б.**, д. ф-м. н., проф., проф. каф. «Высшая математика № 1» НИУ «МИЭТ»; **Шевченко В. И.**, к. т. н., доц., зав. каф. «Корпоративные информационные системы» СевГУ; **Алексеев Е. Р.**, к. т. н., доц., доц. каф. «Информационные образовательные технологии» КубГУ; **Лапицкая Н. В.**, к. т. н., доц., зав. каф. «Программное обеспечение информационных технологий» учреждения образования «Белорусский государственный университет информатики и радиоэлектроники»; **Пацей Н. В.**, к. т. н., доц., зав. каф. «Программная инженерия» учреждения образования «Белорусский государственный технологический университет»; **Мальчева Р. В.**, к. т. н., доц., проф. каф. «Компьютерная инженерия» ДОННТУ; **Хмелевой С. В.**, к. т. н., доц., доц. каф. АСУ ДОННТУ; **Васяева Т. А.**, к. т. н., доц., доц. каф. АСУ, зам. декана ФКНТ ДОННТУ; **Воронова А. И.**, асс. каф. АСУ ДОННТУ; **Андриевская Н. К.**, ст. преп. каф. АСУ ДОННТУ; **Соломченко Н. Н.**, нач. отдела ТСО ДОННТУ.

Адрес оргкомитета:
283001, г. Донецк, просп. 25-летия РККА, 1, Донецкий национальный технический университет, 8 учебный корпус, ФКНТ, кафедра АСУ, ком. 8.601.
E-mail: iuskm@donntu.org

© ГОУ ВПО «Донецкий национальный технический университет», 2021

УДК 004.45

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ UNIFIED EXTENSIBLE FIRMWARE INTERFACE ДЛЯ РАБОТЫ С ФАЙЛОВЫМИ СИСТЕМАМИ, ФАЙЛАМИ И КАТАЛОГАМИ

Якубов Я. А.[1], Чередникова О. Ю.[1], Янковский И. А.[2]
[1]Донецкий национальный технический университет,
кафедра компьютерной инженерии;
[2]Полесский государственный университет,
кафедра информационных технологий и интеллектуальных систем
E-mail: alexgdi@outlook.com

Аннотация:
Якубов Я. А., Чередникова О. Ю., Янковский И. А. Исследование возможностей Unified Extensible Firmware Interface для работы с файловыми системами, файлами и каталогами. *В статье исследованы возможности UEFI для работы с файловыми системами, файлами и каталогами. Рассмотрены возможности UEFI для разработки системных утилит типа файловый менеджер.*

Annotation:
Yakubov Y. A., Cherednikova O. Y., Yankovsky I. A. Research capabilities of Human Interface Infrastructure for developing UEFI GUI applications. *The article explores the capabilities of UEFI for working with file systems, files and directories. Considered the possibilities of UEFI for the development of system utilities such as a file manager.*

Общая постановка проблемы

На сегодняшний день системные утилиты выполняют различные специализированные задачи, связанные с работой оборудования или операционной системы. Решение многих задач невозможно или ограничено окружением операционной системы, в которой выполняется системная утилита [1]. Задачи, которые стоят перед большинством системных утилит, связаны с операциями над файлами. Использование возможностей UEFI для работы с файловыми системами, файлами и каталогами и разработка системной утилиты в виде UEFI приложения позволяет решать задачи, стоящие перед системными утилитами, без ограничений накладываемых операционной системой. Для работы с накопителями, файловыми системами, каталогами и файлами UEFI предоставляет множество расширяемых программных интерфейсов. Для работы с файловыми системами UEFI предоставляет протокол (программный интерфейс) Simple File System Protocol, а для работы с файлами и каталогами используется протокол File Protocol.

Возможности UEFI для работы с файловыми системами

Протокол Simple File System Protocol предоставляет пофайловый доступ к устройству, которое имеет поддерживаемую файловую систему. Данный протокол используется для открытия тома и получения экземпляра протокола File Protocol, который указывает на корневой каталог и предоставляет интерфейс для доступа к файлам на устройстве.

Встроенное программное обеспечение системной платы автоматически создает дескрипторы типа Simple File System Protocol для любого блочного устройства, которое содержит поддерживаемую файловую систему. По умолчанию такими файловыми системами являются файловые системы семейства FAT, такие как FAT12, FAT16 и FAT32. Для использования других файловых систем необходима реализация их поддержки во встроенном программном обеспечении или загрузка драйвера соответствующей файловой

системы используя UEFI Driver Model. Для получения экземпляра протокола Simple File System Protocol используется фрагмент кода, представленный на рис. 1.

```
EFI_SIMPLE_FILE_SYSTEM_PROTOCOL *gSfsp;

Status = gBS->LocateProtocol(&gEfiSimpleFileSystemProtocolGuid, NULL, (VOID **)&gSfsp);

if (EFI_ERROR(Status))
{
        Print(L"Error: Could not find a GOP instance!\n");
        Status = EFI_NOT_FOUND;
        return EFI_ERROR(-1);
}
```

Рисунок 1. Фрагмент кода, реализующий получения экземпляра протокола Simple File System Protocol

Для получения дескриптора корневого каталога по дескриптору типа Simple File System Protocol используется функция OpenVolume. В качестве параметров функции OpenVolume передаются указатель на экземпляр протокола Simple File System Protocol и указатель на указатель переменной типа EFI_FILE_PROTOCOL в которую по выходу из функции будет возвращён дескриптор корневого каталога. После получения дескриптора корневого каталога весь доступ к тому и его файлам осуществляется через дескриптор с использованием протокола File Protocol. Том считается закрытым после закрытия всех дескрипторов файлов на этом томе. Фрагмент кода, реализующий алгоритм открытия тома и получения дескриптора корневого каталога представлен на рис. 2.

```
EFI_FILE_PROTOCOL *gRoot;

Status = gSfsp->OpenVolume(gSfsp, &gRoot);

if (EFI_ERROR(Status))
{
        Print(L"Error: OpenVolume error!\n");
        Status = EFI_NOT_FOUND;
        return EFI_ERROR(-1);
}
```

Рисунок 2. Фрагмент кода, выполняющий получение корневого каталога

Возможности UEFI для работы с файлами и каталогами

Доступ к файловому вводу-выводу обеспечивается при помощи дескриптора корневого каталога, который является экземпляром протокола File Protocol. Дескриптор обеспечивает доступ к содержимому файла или каталога, а также является ссылкой на место в дереве каталогов файловой системы, в которой находится файл [2]. С любым указанным дескриптором файла, другие файлы могут быть открыты относительно местоположения этого файла, создавая новые дескрипторы. Пока на устройстве есть открытые файлы, следует избегать использования базовых протоколов доступа к устройству, например, таких как протоколы Disk IO Protocol или Block IO Protocol.

Работа с файлами выполняется при помощи различных функций. Такие функции как Open и Close выполняют открытие и закрытие файла. Функции Read и Write обеспечивают операции побайтового чтения и записи. Также протокол имеет различные вспомогательные функции, которые упрощают работу и взаимодействие с файлами.

Для выполнения операции открытия файлов и каталогов предоставляется функция Open. Функция Open открывает новый файл относительно дескриптора исходного файла или дескриптора корневого каталога.

Для выполнения операции закрытия файла используется функция Close. Функция Close закрывает переданный ей дескриптор файла. Во время закрытия файла, все данные

кэшированные файловой системой сбрасываются на устройство и файл закрывается. Программа должна закрывать дескриптор файла корневого каталога и любых других открытых файловых дескрипторов перед завершением работы [3].

Для выполнения операции удаления файла используется функция Delete. Функция Delete закрывает и удаляет файл. Если файл не может быть удален, возвращается код предупреждения со статусом EFI_WARN_DELETE_FAILURE, но дескриптор будет закрыт.

Для выполнения операции чтения файла или каталога используется функция Read. Функция Read выполняет побайтовое чтение данных из файла.

Если читаемый объект является файлом, функция считывает запрошенное количество байтов, задаваемое переменной BufferSize из файла в текущей позиции файла, и возвращает их в буфер, заданный указателем Buffer. Если чтение выходит за пределы конца файла, количество прочитанных данных усекается до конца файла. Текущая позиция файла увеличивается на количество возвращенных байтов.

Если это каталог, функция считывает запись каталога в текущей позиции файла и возвращает запись в буфер, заданный указателем Buffer. Если размер буфера недостаточен для хранения текущей записи каталога, возвращается код EFI_BUFFER_TOO_SMALL, а текущая позиция файла не обновляется. Переменная BufferSize устанавливается равным размеру буфера, необходимому для чтения записи каталога. В случае успешного выполнения операции текущая позиция файла обновляется на следующую запись каталога. Если записей в каталоге больше нет, при чтении возвращается буфер нулевой длины. Запись каталога является структурой типа EFI_FILE_INFO. Фрагмент кода, реализующий алгоритм чтения записей каталога представлен на рис. 3.

```
BufferSize = sizeof(EFI_FILE_INFO) + sizeof(CHAR16) * 512;
Status = gBS->AllocatePool(EfiBootServicesCode, BufferSize, (VOID **)&FileInfo);
while (1)
{
        ReadSize = BufferSize;
        // Выполняем чтение записи каталога
        Status = Directory->Read(Directory, &ReadSize, FileInfo);
        // Если размера передаваемого буфера не хватило, выделить буфер необходимого размера.
        if (Status == EFI_BUFFER_TOO_SMALL)
        {
                BufferSize = ReadSize;
                BREAK_ERR(Status = gBS->FreePool(FileInfo));
                BREAK_ERR(Status = gBS->AllocatePool(EfiBootServicesCode, BufferSize,
                                (VOID **)&FileInfo));
                BREAK_ERR(Status = Directory->Read(Directory, &ReadSize, FileInfo));
        }
        // Больше нет записей в каталоге, выход.
        if (ReadSize == 0) break;
        BREAK_ERR(Status);
        // Вывод информации о записи каталога
        Print(L"%s Size: %d FileSize: %d Physical Size: %d\n", FileInfo->FileName, FileInfo->Size,
                FileInfo->FileSize, FileInfo->PhysicalSize);
        Print(L"Attr: %s%s%s%s%s%s\n", FileInfo->Attribute & EFI_FILE_READ_ONLY ? L"RO " : L"",
                FileInfo->Attribute & EFI_FILE_HIDDEN ? L"H " : L"", FileInfo->Attribute &
EFI_FILE_SYSTEM ? L"S " : "", FileInfo->Attribute & EFI_FILE_RESERVED ? L"R " : L"", FileInfo-
>Attribute & EFI_FILE_DIRECTORY ? L"D " : L"", FileInfo->Attribute & EFI_FILE_ARCHIVE ? L"A " :
L"");
}
```

Рисунок 3. Фрагмент кода, реализующий чтение записей каталога

Для выполнения операции записи данных в файл используется функция Write. Функция Write выполняет побайтовую запись данных, заданных указателем на переменную Buffer размером, заданным переменной BufferSize в текущую позицию файла. Текущая позиция файла увеличивается на фактическое количество записанных байтов, которое возвращается в переменную BufferSize. Частичная запись происходит только в случае, если отсутствует свободное пространство для записи. Размер файла автоматически увеличивается при необходимости. Прямая запись в открытые каталоги не поддерживается.

Драйвер файловой системы может кэшировать данные, относящиеся к открытому файлу, поэтому предусмотрена функция Flush, которая сбрасывает все незаписанные данные этого файла в файловой системе на физический носитель. Если нижележащее устройство

может кэшировать данные, файловая система также должна сообщить устройству о необходимости сброса данных на накопитель.

Для получения текущей позиции в файле используется функция GetPosition. Функция GetPosition возвращает значение текущей позиции указателя в файле в параметр Position.

Для установки позиции указателя в файле используется функция SetPosition. Значение позиции для установки передаётся в параметре Position. Значение позиции, заданное как 0xFFFFFFFFFFFFFFFF устанавливает указатель в конец файла. Если операция была применена к каталогу, то единственная позиция, которая может быть установлена – нулевая. Установка такой позиции указателя приводит к повторному запуску процесса чтения записей каталога.

Для получения информации о файле или каталоге используется функция GetInfo. Функция GetInfo принимает параметр InformationType типа EFI_GUID, параметры BufferSize и Buffer и возвращает информацию типа InformationType для указанного файла. Если файл не поддерживает запрошенный тип информации, возвращается код EFI_UNSUPPORTED. Если размер буфера недостаточен для запрошенной структуры, возвращается код EFI_BUFFER_TOO_SMALL, а переданный параметр BufferSize устанавливается равным размеру буфера, необходимому для выполнения запроса. Все типы информации, определенные спецификацией UEFI, являются обязательными типами информации, которые должны поддерживать все файловые системы.

Для изменения информации о файле используется функция SetInfo. Функция SetInfo устанавливает информацию типа InformationType для запрашиваемого файла. Спецификацией UEFI версии 2.8 определены три значения параметра InformationType типа EFI_GUID, которые определяют тип запрашиваемой или передаваемой информации. Такими значениями являются EFI_FILE_INFO_ID, EFI_FILE_SYSTEM_INFO_ID и EFI_FILE_SYSTEM_LABEL_ID, которые запрашивают информацию о файле, файловой системе и метке файловой системы соответственно.

Структура EFI_FILE_INFO используется в функциях SetInfo и GetInfo для установки или получения общей информации о файле или каталоге. Структура содержит такие поля и структуры как размер, физический размер, время создания, время модификации, время последнего доступа и атрибуты файла или каталога.

Поскольку файл доступный только для чтения можно открыть только в режиме только для чтения, параметр InformationType имеющий значение EFI_FILE_INFO_ID можно использовать с файлом только для чтения, потому что этот метод является единственным, который можно использовать для изменения атрибута только для чтения в таком файле. В этом случае может быть изменён только атрибут EFI_FILE_READONLY поля Attribute структуры EFI_FILE_INFO. UEFI поддерживает различные атрибуты для файлов и каталогов, такие как только для чтения, скрытый и т.д. Установленные атрибуты файла при помощи функции SetInfo будут действительны при следующем открытии файла с помощью функции Open. На носителе доступным только для чтения функция SetInfo не может использоваться для изменения информации о файле.

При использовании функции SetInfo на объекте, которым является каталог действуют следующие особенности и ограничения:

- для каталога размер файла определяется содержимым каталога и не может быть изменен путем установки FileSize, а поле PhysicalSize определяется полем FileSize и не может быть изменено. Попытки изменения этих полей у каталога будут проигнорированы;
- для каталога бит атрибута EFI_FILE_DIRECTORY не может быть изменен. Он должен соответствовать фактическому типу файла;
- нулевое значение в CreateTime, LastAccess или ModificationTime приводит к тому, что поля игнорируются (и не обновляются).

Структура EFI_FILE_SYSTEM_INFO используется в функции GetInfo для получения информации о системном томе и в функции SetInfo для установки метки тома. Данная структура может быть получена по дескриптору файла корневого каталога, который был получен при первоначальном открытии файловой системы при помощи функции OpenVolume. Все поля структуры доступны только для чтения, кроме VolumeLabel. Поле VolumeLabel содержит значение метки тома, которое можно создать или изменить, вызвав SetInfo с обновленным значением поля VolumeLabel. Структура EFI_FILE_SYSTEM_INFO содержит такие поля как размер структуры включая размер VolumeLabel, ReadOnly, размер тома, свободное пространство, номинальный размер блока, метку тома в виде нуль терминированной строки. Структура EFI_FILE_SYSTEM_VOLUME_LABEL используется в функциях GetInfo и SetInfo для получения или изменения значения метки тома в поле VolumeLabel данной структуры.

Анализ результатов работы

Для анализа результатов и демонстрации возможностей UEFI по работе с файловыми системами, файлами и каталогами, было разработано приложение, демонстрирующее основные функции и возможности. Результат работы программы, приведенный на рис. 4. демонстрирует содержимое корневого каталога: файлы и каталоги, их атрибуты и даты создания, модификации и доступа к этим объектам.

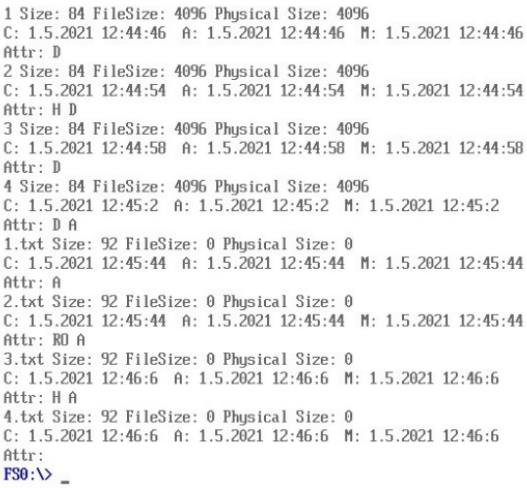


Рисунок 4. Результат работы демонстрационной программы

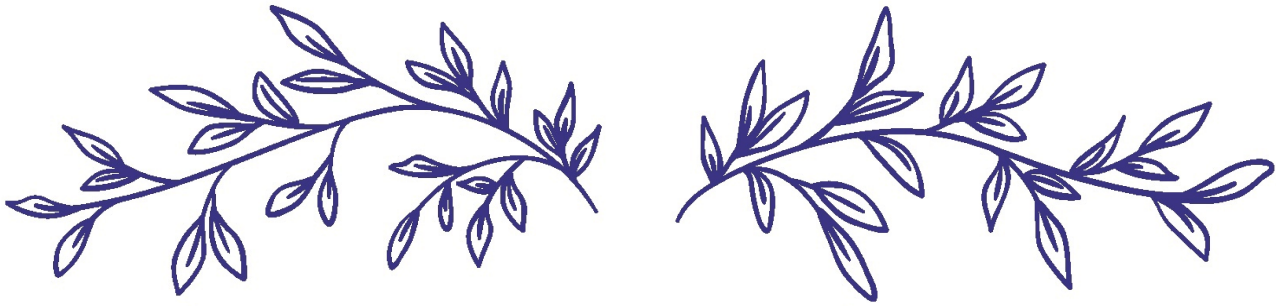
Выводы

В работе были рассмотрены, исследованы и продемонстрированы возможности UEFI по работе с дисковыми устройствами, файловыми системами, файлами и каталогами для разработки системных утилит типа файловый менеджер. Возможности таких программных интерфейсов UEFI как Simple File System Protocol и File Protocol позволяют реализовывать операции по работе с дисковыми устройствами, файловыми системами, файлами и каталогами для разработки системной утилиты типа файловый менеджер.

Литература

1. Embedded Firmware Solutions: Development Best Practices for the Internet of Things // V. Zimmer, J. Sun, M. Jones, S. Reinauer. – 1st edition. – Apress Open, 2015. – 263 p.
2. Зиммер, В. По ту сторону BIOS. Разработка с Unified Extensible Firmware Interface // Зиммер, В, Ротман М., Марисетти С. – 3-е изд., 2017. – 325 с.
3. Unified Extensible Firmware Interface (UEFI) Specification. Version 2.7 Errata A – Unified EFI Forum, 2017. – 2575 с.

СОДЕРЖАНИЕ



ДонНТУ: 100-летний путь успеха
Поздравляем с юбилеем!