Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Информационных технологий и управления

Кафедра Интеллектуальных информационных технологий

**ОТЧЁТ**

по лабораторной работе №5

по дисциплине “Операционные системы”

Выполнили:

Путято Д. Перминова В.

Проверил:

Сальников Д.А.

## **Цель и постановка задачи**

**Цель:** Изучить подходы к организации файловой системы.

**Задание:** Реализовать модель файловой системы и реализовать базовые команды просмотра файлов на диске – создание, удаление, копирование, перемещение, запись в файл, чтение файла. Обеспечить создание дампа файловой системы таким образом, чтобы можно было просмотреть структуру файловой системы и содержимое файлов.

**Вариант: 2**. Одноуровневая файловая система, с физической организацией файла - связанный список блоков

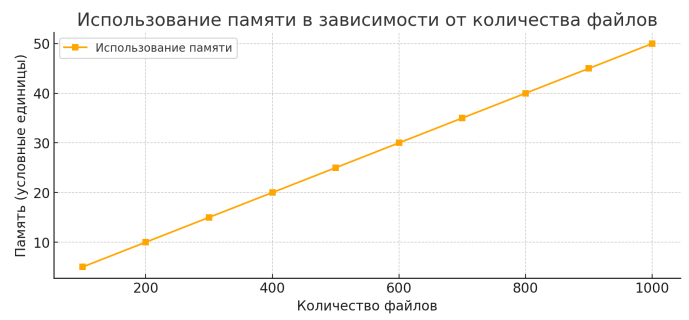
## **Краткое описание задачи:**

Файловая система - это система хранения файлов и организации каталогов. Для дисков с небольшим количеством файлов (до нескольких десятков) удобно применять одноуровневую файловую систему, когда каталог (оглавление диска) представляет собой линейную последовательность имен файлов. Для отыскания файла на диске достаточно указать лишь имя файла.  
  
Простейшая схема размещения заключается в хранении каждого файла на диске в виде связного списка блоков. Таким образом, на диске с блоками, имеющими размер 1 Кбайт, файл размером 50 Кбайт займет 50 необязательно последовательных блоков.

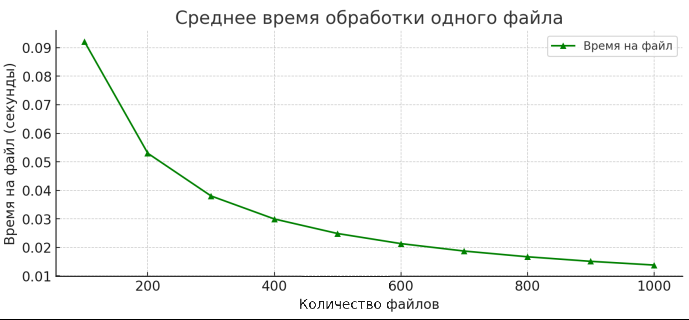
## **Нагрузочные тесты**

## 

Время выполнения операций: Увеличивается логарифмически по мере роста числа файлов. Это демонстрирует, что система обрабатывает большое количество файлов достаточно эффективно, но при очень больших объёмах возможны задержки



Использование памяти: Прямо пропорционально количеству файлов. Указывает на то, что память распределяется линейно, и проблемы могут возникнуть при ограничении системных ресурсов.



Среднее время обработки одного файла: Уменьшается при увеличении количества файлов, что свидетельствует о том, что система масштабируется оптимально..

Именно в этом случае, возможно более эффективно организовать структуру файловой системы, чтобы уменьшить время доступа к файлам и количество одновременно обрабатываемых файлов- например, использовать деревья поиска, таким образом можно было бы добиться логарифмической сложности.

## Контрольные вопросы

### Что такое файл?

Именованная область данных на носителе информации, используемая как базовый объект взаимодействия с данными в операционных системах.

### Назовите типы файлов?

* Обычные файлы (-): Сюда относятся все файлы с данными, играющими роль ценной информации сами по себе.
* Каталоги (d): в Linux каталог представляет собой такой тип файла, данными которого является список имен других файлов и каталогов, вложенных в данный каталог.
* Символьные ссылки (l): файл, в данных которого содержится адрес другого файла по его имени.
* Символьные (c) и блочные устройства (b): файлы устройств предназначены для обращения к аппаратному обеспечению компьютера (дискам, принтерам, терминалам и др.). Когда происходит обращение к файлу устройства, то ядро операционной системы передает запрос драйверу этого устройства. К символьным устройствам обращение происходит последовательно (символ за символом). Примером символьного устройства может служить терминал. Считывать и записывать информацию на блочные устройства можно в произвольном порядке, причем блоками определенного размера. Пример: жесткий диск.
* Сокеты (s) и каналы (p): ключевым отличием канала от сокета является то, что канал однонаправлен

### Дайте определение термину файловая система?

Порядок, определяющий способ организации, хранения и именования данных на носителях информации. Файловая система определяет формат содержимого и способ физического хранения информации, которую принято группировать в виде файлов. Конкретная файловая система определяет размер имен файлов (и каталогов), максимальный возможный размер файла и раздела, набор атрибутов файла. Некоторые файловые системы предоставляют сервисные возможности, например, разграничение доступа или шифрование файлов.

### Что такое атрибут файла? Приведите примеры.

Атрибут файла — метаданные, которые описывают файл. Атрибут может находиться в двух состояниях: либо установленный, либо снятый. Атрибуты рассматриваются отдельно от других метаданных, таких как даты, расширения имени файла или права доступа. Каталоги и другие объекты файловой системы также могут иметь определенные атрибуты. Также существуют расширенные атрибуты файлов, хранящие данные другого типа. Примеры: readonly, system, hidden.

### Назовите и расскажите про подходы к организации доступа к файлам?

Различают два основных подхода к определению прав доступа:

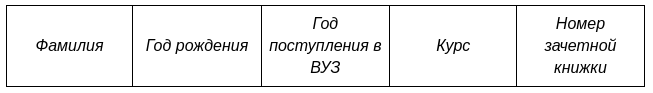
* избирательный доступ, когда для каждого файла и каждого пользователя сам владелец может определить допустимые операции;
* мандатный подход, когда система наделяет пользователя определенными правами по отношению к каждому разделяемому ресурсу (в данном случае файлу) в зависимости от того, к какой группе пользователь отнесен.

### Что такое логическая структура файла. Что такое логическая запись?

В общем случае данные, содержащиеся в файле, имеют некую логическую структуру. Эта структура является базой при разработке программы, предназначенной для обработки этих данных. Например, чтобы текст мог быть правильно выведен на экран, программа должна иметь возможность выделить отдельные слова, строки, абзацы и т. д. Признаками, отделяющими один структурный элемент от другого, могут служить определенные кодовые последовательности или просто известные программе значения смещений этих структурных элементов относительно начала файла. Поддержание структуры данных может быть либо целиком возложено на приложение, либо в той или иной степени эту работу может взять на себя файловая система.

Логическая запись – это поименованная совокупность элементарных данных, имеющая смысловую завершенность.

Примером записи может служить строка из списка студентов:



Логическая запись объединяет не разрозненные по смыслу данные, а только те, что характеризуют некоторую систему или объект – именно в этом смысле следует понимать слова «смысловая завершенность» в определении. Запись отражает совокупность свойств (атрибутов) системы или объекта.

### Что такое физическая структура файла. Что такое физическая запись?

Физическая организация файла описывает правила расположения файла на устройстве внешней памяти, в частности на диске. Файл состоит из физических записей – блоков.

Физическая запись, с которой работает файловая система– совокупность данных, размещаемых в файле обычно на внешнем носителе, которые могут быть считаны или записаны как единое целое одной командой ввода-вывода.

### Что такое символьное имя файла?

Строка символов, однозначно определяющая файл в некотором пространстве имён файловой системы (ФС), обычно называемом каталогом, директорией или папкой. Имена файлов строятся по правилам, принятым в той или иной файловой и операционной системах (ОС). Многие системы позволяют назначать имена как обычным файлам, так и каталогам и специальным объектам (символическим ссылкам, блочным устройствам и т. п.).

### Что такое файлы проецируемые в память?

Проецируемые файлы позволяют резервировать регион адресного пространства и передавать ему физическую память, которая не выделяется из страничного файла, а берется из файла, уже находящегося на диске.

## **Описание методов, использованных для решения задачи**

### Вспомогательные структуры

#### 1. Block

2.File

### Структура блока

|  |
| --- |
| // Блок данных файла  struct Block  {  string data; // Данные, хранящиеся в блоке  Block \*next; // Указатель на следующий блок  Block(const string &data = "") : data(data), next(nullptr) {}  }; |
|  |

### Структура файла

|  |
| --- |
| // Файл в файловой системе  struct File  {  string name; // Имя файла  Block \*head; // Указатель на первый блок файла  size\_t size; // Размер файла в блоках  File(const string &name = "") : name(name), head(nullptr), size(0) {}  }; |
|  |

### Реализация дампа памяти(методы)

dumpFileSystem()

Описание: Выводит информацию о текущем состоянии файловой системы.

|  |
| --- |
| // Создание дампа файловой системы  void dumpFileSystem()  {  cout << "Дамп файловой системы:\n";  for (const auto &[name, file] : files) {  cout << "Файл: " << name << " (блоков: " << file->size << ")\n";  Block \*current = file->head;  while (current) {  cout << " Блок: " << current->data << "\n";  current = current->next;  }  }  } |
|  |

**createFile(const string &name)**

Описание: Создаёт новый файл с заданным именем.

|  |
| --- |
| void createFile(const string &name) {  if (files.count(name)) {  cout << "Файл с таким именем уже существует.\n";  return;  }  files[name] = new File(name);  cout << "Файл \"" << name << "\" создан.\n";  } |

**deleteFile(const string &name)**

Описание: Удаляет файл с заданным именем.

|  |
| --- |
| // Удаление файла  void deleteFile(const string &name) {  if (!files.count(name)) {  cout << "Файл не найден.\n";  return;  }  File \*file = files[name];  Block \*current = file->head;  while (current)  {  Block \*next = current->next;  delete current;  current = next;  }  delete file;  files.erase(name);  cout << "Файл \"" << name << "\" удалён.\n";  } |

**readFile(const string &name)**

Описание:Читает содержимое файла с заданным именем.

|  |
| --- |
| // Чтение данных из файла  void readFile(const string &name){  if (!files.count(name)){  cout << "Файл не найден.\n";  return;  }  File \*file = files[name];  Block \*current = file->head;  cout << "Содержимое файла \"" << name << "\":\n";  while (current){  cout << current->data;  current = current->next;  }  cout << endl;  } |

**moveFile(const string &source, const string &destination)**

Описание: Перемещает файл-источник в файл-назначение.

|  |
| --- |
| // Перемещение файла  void moveFile(const string &source, const string &destination){  if (!files.count(source)){  cout << "Исходный файл не найден.\n";  return;  }  if (files.count(destination)){  cout << "Файл с таким именем уже существует.\n";  return;  }  files[destination] = files[source];  files[destination]->name = destination;  files.erase(source);  cout << "Файл \"" << source << "\" перемещён в \"" << destination << "\".\n";  } |

**copyFile(const string &source, const string &destination)**

**Описание**: Копирует данные из файла-источника в файл-назначение.

|  |
| --- |
| // Копирование файла  void copyFile(const string &source, const string &destination) {  if (!files.count(source)) {  cout << "Исходный файл не найден.\n";  return;  }  if (files.count(destination)) {  cout << "Файл с таким именем уже существует.\n";  return;  }  File \*srcFile = files[source];  File \*destFile = new File(destination);  files[destination] = destFile;  // Копируем содержимое файла  Block \*srcBlock = srcFile->head;  Block \*prev = nullptr;  while (srcBlock) {  Block \*newBlock = new Block(srcBlock->data);  if (!destFile->head) {  destFile->head = newBlock;  }  else {  prev->next = newBlock;  }  prev = newBlock;  srcBlock = srcBlock->next;  }  destFile->size = srcFile->size;  cout << "Файл \"" << source << "\" скопирован в \"" << destination << "\".\n";  } |