Лабораторная работа 09

Файловая система

OC, ПОИТ-3

1. ***Внимание! Для работы с файловой системой использовать только OS API.***

**Задание 01.Windows**

1. С помощью Notepad (Notepad+) создайте на дисковом устройстве текстовый файл **OS09\_01.txt.** Заполните его 10 строками из списка студентов вашей подгруппы. Буквы кириллические.



1. Разработайте приложение **OS09\_01**.
2. Приложение **OS09\_01** вызывает функцию **printFileInfo**, имеющую следующий прототип.



1. Функция **printFileInfo** выводит в стандартный поток вывода следующую информацию:

- имя файла;

- тип файла;

- размер файла;

- дата и время создания файла;

- дата и время последнего обновления.

|  |
| --- |
| BOOL printFileInfo(LPWSTR path)  {  HANDLE file = CreateFile(  path,  GENERIC\_READ,  NULL,  NULL,  OPEN\_ALWAYS,  FILE\_ATTRIBUTE\_NORMAL,  NULL);  SYSTEMTIME sysTime;  FILETIME localTime;  BY\_HANDLE\_FILE\_INFORMATION fi;  BOOL fResult = GetFileInformationByHandle(file, &fi);  if (fResult)  {  cout << "File name:\t" << getFileName((wchar\_t\*)FILE\_PATH);  wcout << "\nFile type:\t" << getFileType(file);  cout << "\nFile size:\t" << fi.nFileSizeLow << " bytes";  FileTimeToLocalFileTime(&fi.ftCreationTime, &localTime);  FileTimeToSystemTime(&localTime, &sysTime);  printf("\nCreate time:\t%02d.%02d.%04d %02dh %02dm %02ds", sysTime.wDay, sysTime.wMonth, sysTime.wYear, sysTime.wHour % 24, sysTime.wMinute, sysTime.wSecond);  FileTimeToLocalFileTime(&fi.ftLastWriteTime, &localTime);  FileTimeToSystemTime(&localTime, &sysTime);  printf("\nUpdate time:\t%02d.%02d.%04d %02dh %02dm %02ds", sysTime.wDay, sysTime.wMonth, sysTime.wYear, sysTime.wHour % 24, sysTime.wMinute, sysTime.wSecond);  }  CloseHandle(file);  return true;  } |

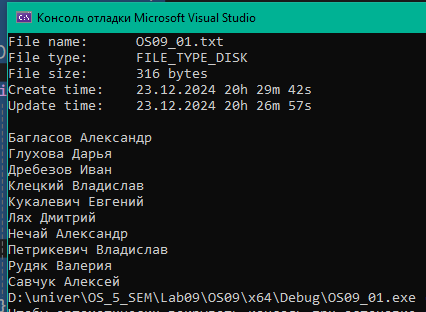
1. Приложение **OS09\_01** вызывает функцию **printFileTxt**, имеющую следующий прототип.



1. Функция **printFileTxt** выводит в стандартный поток вывода содержимое файла.
2. При вызове функции укажите в качестве параметра имя текстового файла, созданного в п.2.

|  |
| --- |
| 1. BOOL printFileTxt(LPCWSTR path) 2. { 3. HANDLE file = CreateFile( 4. path, 5. GENERIC\_READ, 6. 0, 7. NULL, 8. OPEN\_EXISTING, 9. FILE\_ATTRIBUTE\_NORMAL, 10. NULL 11. ); 12. if (file != INVALID\_HANDLE\_VALUE) { 14. char buffer[1024]; 15. DWORD bytesRead; 16. while (ReadFile(file, buffer, sizeof(buffer) - 1, &bytesRead, NULL) && bytesRead > 0) { 17. buffer[bytesRead] = '\0'; 18. cout << buffer; 19. } 20. CloseHandle(file); 21. return TRUE; 22. } 23. else { 24. cerr << "[ERROR]: open file" << endl; 25. return FALSE; 26. } 27. } |

1. Продемонстрируйте работоспособность приложения **OS09\_01**.



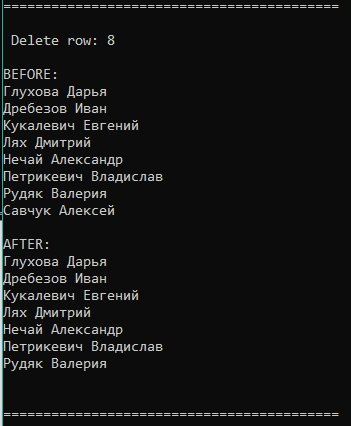
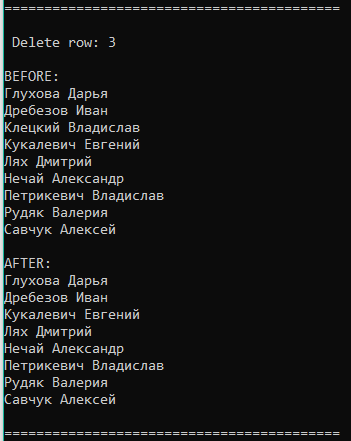
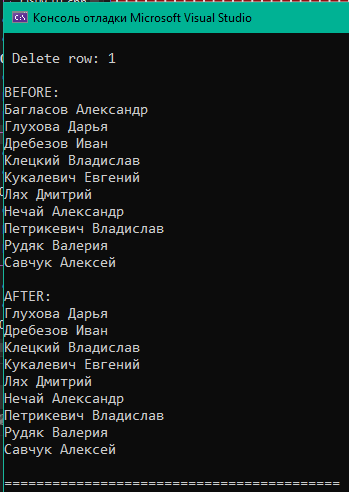
**Задание 02.Windows**

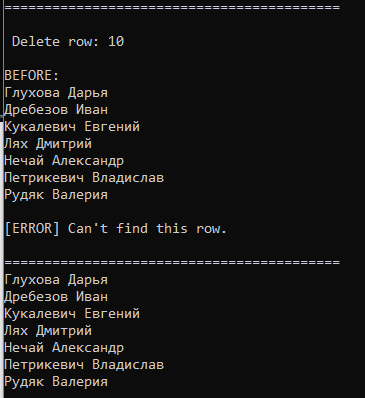
1. Разработайте приложение **OS09\_02.**
2. Приложение **OS09\_02** вызывает функцию **delRowFileTxt**, имеющую следующий прототип.



|  |
| --- |
| BOOL delRowFileTxt(LPWSTR fileName, DWORD row)  {  char filepath[20];  filepath[19] = '\0';  wcstombs(filepath, fileName, 20);  cout << "\n Delete row: " << row << "\n";  try  {  HANDLE hf = CreateFile(fileName, GENERIC\_READ, NULL, NULL, OPEN\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);  if (hf == INVALID\_HANDLE\_VALUE)  {  CloseHandle(hf);  throw "[ERROR] Create or open file failed";  }  DWORD n = NULL;  char buf[1024];  BOOL b;  ZeroMemory(buf, sizeof(buf));  b = ReadFile(hf, &buf, sizeof(buf), &n, NULL);  if (!b)  {  CloseHandle(hf);  throw ("[ERROR] Read file unsuccessful");  }  cout << "\nBEFORE:\n";  cout << buf << endl;  CloseHandle(hf);  HANDLE hAppend = CreateFile(fileName, GENERIC\_WRITE, NULL, NULL, OPEN\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);  char editedBuf[1024];  ZeroMemory(editedBuf, sizeof(editedBuf));  int line = 1;  int j = 0;  bool rowFound = false;  for (int i = 0; i < n; i++)  {  if (line == row)  rowFound = true;  else  {  editedBuf[j] = buf[i];  j++;  }  if (buf[i] == '\n')  line++;  }  if (!rowFound)  {  CloseHandle(hAppend);  throw ("[ERROR] Can't find this row.\n");  }  b = WriteFile(hAppend, editedBuf, n, &n, NULL);  if (!b)  {  CloseHandle(hAppend);  throw ("[ERROR] Write file unsuccessful\n");  }  cout << "\nAFTER:\n";  cout << editedBuf << endl;  CloseHandle(hAppend);  cout << "\n==========================================\n";  return true;  }  catch (const char\* em)  {  cout << em << " \n";  cout << "==========================================\n";  return false;  }  } |

1. Функция применяется к файлу **OS09\_01.txt** (п.2) и вызывается последовательно 4 раза, с row = 1,3,8,10. Результат выполнения продемонстрируйте с помощью функции **printFileTxt** (п.6).
2. Продемонстрируйте работоспособность приложения **OS09\_02**.





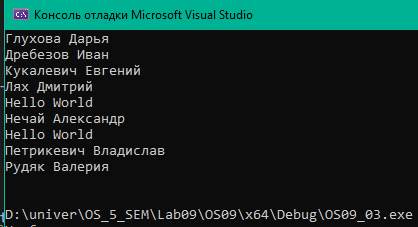
**Задание 03.Windows**

1. Разработайте приложение **OS09\_03.**
2. Приложение **OS09\_03** вызывает функцию **insRowFileTxt**, имеющую следующий прототип.



|  |
| --- |
| BOOL insRowFileTxt(LPWSTR fileName, LPWSTR str, DWORD row) {  try {    HANDLE hFile = CreateFile(  fileName,  GENERIC\_READ,  FILE\_SHARE\_READ,  NULL,  OPEN\_EXISTING,  FILE\_ATTRIBUTE\_NORMAL,  NULL);  if (hFile == INVALID\_HANDLE\_VALUE) {  throw "Cannot open file for reading";  }  //BOM  unsigned char bom[3];  DWORD bytesRead;  ReadFile(hFile, bom, 3, &bytesRead, NULL);  bool isUtf8 = (bytesRead == 3 && bom[0] == 0xEF && bom[1] == 0xBB && bom[2] == 0xBF);  SetFilePointer(hFile, 0, NULL, FILE\_BEGIN); // в начало  // чтение файла в буфер  DWORD fileSize = GetFileSize(hFile, NULL);  if (fileSize == INVALID\_FILE\_SIZE) {  CloseHandle(hFile);  throw "Cannot get file size";  }  char\* buffer = new char[fileSize + 1];  if (!ReadFile(hFile, buffer, fileSize, &bytesRead, NULL)) {  CloseHandle(hFile);  delete[] buffer;  throw "Cannot read file";  }  buffer[bytesRead] = '\0';  CloseHandle(hFile);  // разбиваем на строки  vector<string> lines;  string currentLine;  for (DWORD i = 0; i < bytesRead; i++) {  if (buffer[i] == '\n') {  if (!currentLine.empty() && currentLine.back() == '\r') {  currentLine.pop\_back();  }  lines.push\_back(currentLine);  currentLine.clear();  }  else {  currentLine += buffer[i];  }  }  if (!currentLine.empty()) {  lines.push\_back(currentLine);  }  delete[] buffer;  // LPWSTR str в UTF-8 или ANSI  string newLine;  int sizeNeeded;  if (isUtf8) {  sizeNeeded = WideCharToMultiByte(CP\_UTF8, 0, str, -1, NULL, 0, NULL, NULL);  newLine.resize(sizeNeeded - 1);  WideCharToMultiByte(CP\_UTF8, 0, str, -1, &newLine[0], sizeNeeded - 1, NULL, NULL);  }  else {  sizeNeeded = WideCharToMultiByte(CP\_ACP, 0, str, -1, NULL, 0, NULL, NULL);  newLine.resize(sizeNeeded - 1);  WideCharToMultiByte(CP\_ACP, 0, str, -1, &newLine[0], sizeNeeded - 1, NULL, NULL);  }  // вставляем  if (row == -1) {  lines.push\_back(newLine);  }  else if (row == 0) {  lines.insert(lines.begin(), newLine);  }  else if (row >= lines.size()) {  lines.push\_back(newLine);  }  else {  lines.insert(lines.begin() + row, newLine);  }  //запись  hFile = CreateFile(  fileName,  GENERIC\_WRITE,  0,  NULL,  CREATE\_ALWAYS,  FILE\_ATTRIBUTE\_NORMAL,  NULL);  if (hFile == INVALID\_HANDLE\_VALUE) {  throw "Cannot open file for writing";  }    DWORD bytesWritten;  if (isUtf8) {  const char\* utf8BOM = "\xEF\xBB\xBF";  WriteFile(hFile, utf8BOM, 3, &bytesWritten, NULL);  }  for (const auto& line : lines) {  string lineWithNewline = line + "\r\n";  if (!WriteFile(hFile, lineWithNewline.c\_str(), lineWithNewline.length(), &bytesWritten, NULL)) {  CloseHandle(hFile);  throw "Cannot write to file";  }  }  CloseHandle(hFile);  return TRUE;  }  catch (const char\* err) {  cout << "Error: " << err << endl;  return FALSE;  }  } |

1. Функция применяется к файлу **OS09\_01.txt** (п.2) и вызывается последовательно 4 раза, с row = 0,-1,5,7. Результат выполнения продемонстрируйте с помощью функции **printFileTxt** (п.6).
2. Продемонстрируйте работоспособность приложения **OS09\_03**.



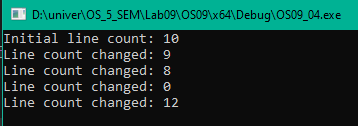
**Задание 04.Windows**

1. Разработайте приложение **OS09\_04.**
2. Приложение **OS09\_04** вызывает функцию **printWathRowFileTxt**, имеющую следующий прототип.



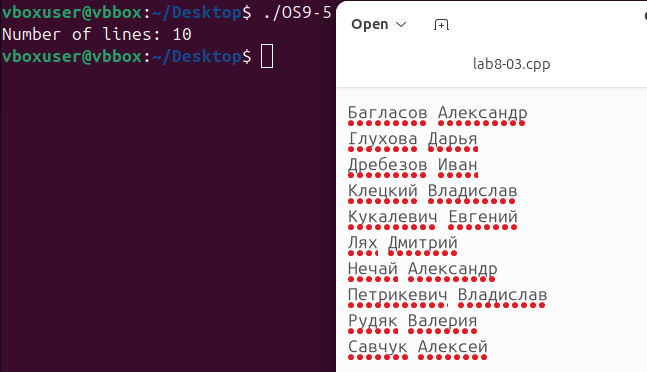
|  |
| --- |
| void printWathRowFileTxt(LPWSTR fileName, DWORD mlsec) {  HANDLE hDir = CreateFile(  DIRECTORY\_PATH,  FILE\_LIST\_DIRECTORY,  FILE\_SHARE\_READ | FILE\_SHARE\_WRITE | FILE\_SHARE\_DELETE,  NULL,  OPEN\_EXISTING,  FILE\_FLAG\_BACKUP\_SEMANTICS,  NULL  );  if (hDir == INVALID\_HANDLE\_VALUE) {  cerr << "Error opening directory handle." << endl;  return;  }  DWORD startTime = GetTickCount();  DWORD initialLineCount = getLineCount(fileName);  cout << "Initial line count: " << initialLineCount << endl;  char buffer[1024];  DWORD bytesReturned;  while (true) {  if (GetTickCount() - startTime >= mlsec) {  cout << "Timer expired." << endl;  break;  }  if (ReadDirectoryChangesW(  hDir,  &buffer,  sizeof(buffer),  FALSE,  FILE\_NOTIFY\_CHANGE\_SIZE | FILE\_NOTIFY\_CHANGE\_LAST\_WRITE,  &bytesReturned,  NULL,  NULL  )) {  FILE\_NOTIFY\_INFORMATION\* pNotify = (FILE\_NOTIFY\_INFORMATION\*)buffer;  do {  if (pNotify->Action == FILE\_ACTION\_MODIFIED) {  DWORD newLineCount = getLineCount(fileName);  if (newLineCount != initialLineCount) {  cout << "Line count changed: " << newLineCount << endl;  initialLineCount = newLineCount;  }  }  pNotify = pNotify->NextEntryOffset ? (FILE\_NOTIFY\_INFORMATION\*)((char\*)pNotify + pNotify->NextEntryOffset) : NULL;  } while (pNotify);  }  Sleep(100);  }  CloseHandle(hDir);  } |

1. Функция применяется к файлу **OS09\_01.txt** (п.2), следит (***используйте функцию наблюдения за файлами в каталоге***) за изменением количества строк в файле в течении mlsec и выводит информацию об изменениях в стандартный поток вывода.
2. Продемонстрируйте работоспособность приложения **OS09\_04** совместно с приложениями **OS09\_03** и **OS09\_04**.



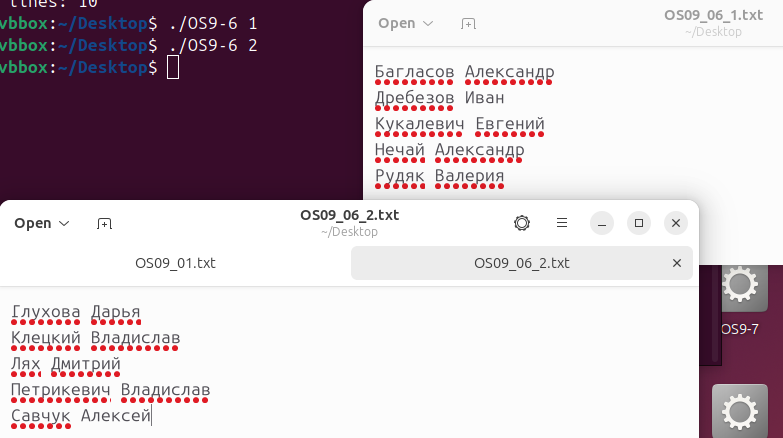
**Задание 05.Linux**

1. Создайте текстовый файл **OS09\_05.txt,** аналогичный файлу **OS09\_01.txt** (п.2).
2. Разработайте приложение **OS09\_05,** подсчитывающее количество строк и выводящее это значение в стандартный поток.
3. Продемонстрируйте работоспособность приложения **OS09\_05**.



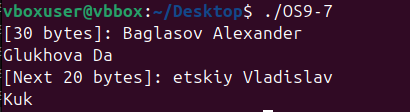
**Задание 06.Linux**

1. Разработайте приложение **OS09\_06,** принимающее 1 параметр, принимающее целочисленное числовое значение.
2. Если числовое значение принимает нечетное значение, то приложение создает новый файл **OS09\_06\_1.txt**, содержащий только нечетные строки из файла **OS09\_05.txt**.
3. Если числовое значение принимает четное значение, то приложение создает новый файл **OS09\_06\_2.txt**, содержащий только четные строки из файла **OS09\_05.txt**.
4. Продемонстрируйте работоспособность приложения **OS09\_06**.



**Задание 07.Linux**

1. Разработайте приложение **OS09\_07,** демонстрирующее возможности функции **lseek.**
2. Поясните назначение и принцип работы функции **lseek**.



**Задание 08.** Ответьте на следующие вопросы

**31. Что такое файл?**

Файл — это логическая структура данных, используемая для хранения информации на носителе (диск, флешка и т.д.). Файлы могут содержать текст, изображения, аудио, видео или любые другие данные.

**32. Перечислите основные характеристики (атрибуты) файла.**

* **Имя файла**: уникальное название файла в пределах каталога.
* **Размер файла**: объем данных, хранящихся в файле.
* **Тип файла**: определяет формат или содержание файла (например, текст, изображение).
* **Дата и время создания**: время, когда файл был создан.
* **Дата и время последнего изменения**: время, когда файл был последний раз изменен.
* **Атрибуты**: свойства файла, такие как только для чтения, скрытый, системный и т.д.

**33. Что такое файловая система?**

Файловая система — это способ организации, хранения и управления файлами на носителе данных. Она определяет, как данные записываются, читаются и организуются на диске.

**34. Перечислите основные функции файловой системы.**

* Создание и удаление файлов и каталогов.
* Чтение и запись данных в файлы.
* Управление атрибутами файлов.
* Организация доступа к данным.
* Поддержка иерархической структуры каталогов.

**35. Перечислите 3 названия файловой системы.**

* NTFS (New Technology File System)
* FAT32 (File Allocation Table 32)
* ext4 (Fourth Extended Filesystem)

**36. Какая файловая система установлена на вашем компьютере под Windows? под Linux?**

* **Windows**: Обычно NTFS.
* **Linux**: Обычно ext4, но могут использоваться и другие (например, XFS, Btrfs).

**37. Что такое каталог файловой системы? Перечислите наименования специальных каталогов.**

Каталог — это структура, которая содержит ссылки на файлы и другие подкаталоги. Специальные каталоги:

* **Корневой каталог**: основной каталог файловой системы.
* **Текущий каталог**: каталог, в котором в данный момент находится программа.
* **Домашний каталог**: каталог, связанный с конкретным пользователем.

**38. Поясните понятие «текущий каталог приложения».**

Текущий каталог приложения — это каталог, в котором выполняется программа или приложение. Он является базой для относительных путей к файлам.

**39. Что такое специальные имена файлов? Перечислите их, для чего они нужны.**

Специальные имена файлов — это зарезервированные имена, которые имеют особое значение. Примеры:

* . (текущий каталог)
* .. (родительский каталог)
* CON, PRN, AUX, NUL в Windows (особые устройства).  
  Они нужны для упрощения навигации и доступа к системным ресурсам.

**40. Для чего используются буферы ввода-вывода?**

Буферы ввода-вывода используются для временного хранения данных, передаваемых между устройствами ввода-вывода и приложениями. Это позволяет оптимизировать производительность, уменьшая количество операций чтения и записи.

**41. Поясните понятие «кэширование».**

Кэширование — это процесс временного хранения данных для ускорения доступа к ним. Когда данные запрашиваются, система сначала проверяет кэш; если данные там находятся, они считываются быстрее, чем если бы их загружали из основного хранилища.

**42. Поясните понятие «указатель позиции файла».**

Указатель позиции файла — это смещение, которое указывает, где в файле происходит чтение или запись. Он определяет текущее положение в открытом файле.

**43. Поясните понятие «маркер конца файла».**

Маркер конца файла — это специальный индикатор, который указывает на конец содержимого файла. Он используется для обозначения места, где чтение или запись данных должны прекращаться.

**44. Поясните понятие «блокировка файла».**

Блокировка файла — это механизм, который предотвращает одновременный доступ к файлу несколькими процессами. Это необходимо для защиты данных от повреждений или неконсистентности.

**45. Windows. Функция OS API для создания файла.**

Функция CreateFile используется для создания нового файла или открытия существующего.

**46. Windows. Функция OS API для открытия файла.**

Функция CreateFile также используется для открытия файла с различными параметрами доступа.

**47. Windows. Функция OS API для удаления файла.**

Функция DeleteFile используется для удаления указанного файла из файловой системы.

**48. Windows. Функция OS API для записи в файл.**

Функция WriteFile используется для записи данных в открытый файл.

**49. Windows. Функция OS API для чтения файла.**

Функция ReadFile используется для чтения данных из открытого файла.

**50. Windows. Назначение и отличие функций OS API: CopyFile, MoveFile, ReplaceFile.**

* **CopyFile**: копирует файл из одного места в другое.
* **MoveFile**: перемещает файл, удаляя его из исходного местоположения.
* **ReplaceFile**: заменяет файл другим, с возможностью резервного копирования.

**51. Windows. Перечислите функции OS API, которые изменяют текущее значение указателя позиции файла.**

* SetFilePointer
* SetFilePointerEx
* ReadFile
* WriteFile

**52. Windows. Перечислите функции OS API для блокировки и разблокировки файлов.**

* LockFile
* UnlockFile
* LockFileEx
* UnlockFileEx

**53. Windows. Поясните механизм «наблюдение за каталогом», перечислите набор функций OS API, позволяющий реализовать этот механизм.**

Механизм «наблюдение за каталогом» позволяет отслеживать изменения в файловой системе (например, создание, изменение или удаление файлов). Функции для реализации:

* FindFirstChangeNotification
* FindNextChangeNotification
* FindCloseChangeNotification

**54. Windows. Перечислите функции OS API для работы с каталогами, поясните их назначения.**

* CreateDirectory: создает новый каталог.
* RemoveDirectory: удаляет пустой каталог.
* GetFileAttributes: получает атрибуты файла или каталога.
* SetFileAttributes: устанавливает атрибуты файла или каталога.

**55. Linux. Что такое FHS?**

FHS (Filesystem Hierarchy Standard) — это стандарт, описывающий структуру каталогов в операционных системах на базе Linux. Он определяет, какие каталоги должны существовать и что в них должно находиться.

**56. Linux. Перечислите типы файловых систем.**

* ext4
* XFS
* Btrfs
* JFS
* FAT32
* NTFS

**57. Linux. Что такое inode?**

inode — это структура данных в файловой системе, содержащая информацию о файле (например, его размер, права доступа, временные метки и расположение данных на диске). Каждый файл и каталог имеют уникальный inode.

**58. Linux. Поясните назначение функций open, read, write, close, ioctl, stat, flush, lseek, lstat, fstat.**

* **open**: открывает файл и возвращает дескриптор.
* **read**: считывает данные из файла.
* **write**: записывает данные в файл.
* **close**: закрывает открытый файл.
* **ioctl**: управляет устройствами через файловые дескрипторы.
* **stat**: получает информацию о файле (например, размер и атрибуты).
* **flush**: очищает буфер вывода, записывая данные в файл.
* **lseek**: изменяет текущее положение указателя в файле.
* **lstat**: получает информацию о файле, включая символические ссылки.
* **fstat**: получает информацию о файле по дескриптору.