0Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Операционные системы

Студент: Лопатнюк П.В.

ФИТ 3 курс 1 группа

Преподаватель: Бернацкий П.В.

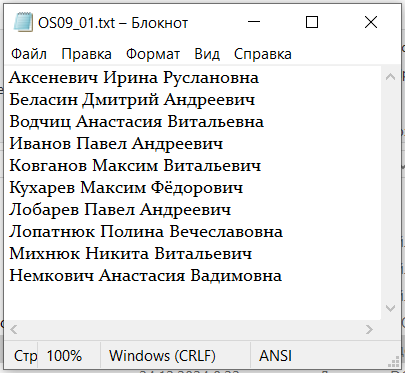
Минск 2024

**Лабораторная работа 09. Файловая система**

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;

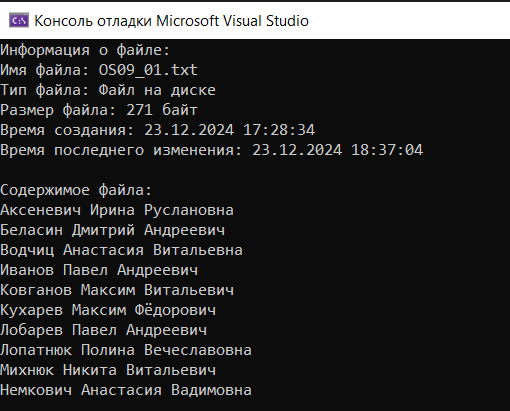
- дата и время последнего обновления.

6. 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.
3. Продемонстрируйте работоспособность приложения **OS09\_01**.

|  |
| --- |
| #include <windows.h>  #include <iostream>  #include <iomanip>  using namespace std;  #define FILE\_PATH "D:\\Уник\\1sem\\ОС\\лаба9-ос\\OS09\_01.txt"  string getFileName(const char\* filePath) {  string filename(filePath);  const size\_t last\_slash\_idx = filename.find\_last\_of("\\/");  if (string::npos != last\_slash\_idx)  filename.erase(0, last\_slash\_idx + 1);  return filename;  }  LPCSTR getFileType(HANDLE file) {  switch (GetFileType(file)) {  case FILE\_TYPE\_UNKNOWN: //0  return "Неизвестный тип файла";  case FILE\_TYPE\_DISK: //1  return "Файл на диске";  case FILE\_TYPE\_CHAR: //2  return "Символьный файл";  case FILE\_TYPE\_PIPE: //3  return "Файл канала";  case FILE\_TYPE\_REMOTE: //4  return "Удалённый файл";  default:  return "[ОШИБКА]: Неизвестный тип файла";  }  }  BOOL printFileInfo(const char\* path) {  HANDLE file = CreateFileA(  path,  GENERIC\_READ, // Операции с файлом  FILE\_SHARE\_READ, // Разрешение на совместный доступ  NULL, // Указатель на атрибуты безопасности  OPEN\_EXISTING, // Режим открытия файла  FILE\_ATTRIBUTE\_NORMAL, // Атрибуты файла  NULL // Указатель на шаблон файла  );  if (file == INVALID\_HANDLE\_VALUE) {  cerr << "Ошибка при открытии файла: " << path << endl;  return FALSE;  }  BY\_HANDLE\_FILE\_INFORMATION fi;  if (!GetFileInformationByHandle(file, &fi)) {  cerr << "Не удалось получить информацию о файле." << endl;  CloseHandle(file);  return FALSE;  }  SYSTEMTIME sysTime;  string filename = getFileName(path);  cout << "Имя файла: " << filename << endl;  cout << "Тип файла: " << getFileType(file) << endl;  DWORD fileSize = fi.nFileSizeLow;  cout << "Размер файла: " << fileSize << " байт" << endl;  FileTimeToSystemTime(&fi.ftCreationTime, &sysTime);  cout << "Время создания: " << setfill('0') << setw(2) << sysTime.wDay << "."  << setw(2) << sysTime.wMonth << "." << setw(4) << sysTime.wYear << " "  << setw(2) << sysTime.wHour << ":" << setw(2) << sysTime.wMinute << ":"  << setw(2) << sysTime.wSecond << endl;  FileTimeToSystemTime(&fi.ftLastWriteTime, &sysTime);  cout << "Время последнего изменения: " << setfill('0') << setw(2) << sysTime.wDay << "."  << setw(2) << sysTime.wMonth << "." << setw(4) << sysTime.wYear << " "  << setw(2) << sysTime.wHour << ":" << setw(2) << sysTime.wMinute << ":"  << setw(2) << sysTime.wSecond << endl;  CloseHandle(file);  return TRUE;  }  BOOL printFileTxt(const char\* path) {  HANDLE file = CreateFileA(  path,  GENERIC\_READ, // Операции с файлом  FILE\_SHARE\_READ, // Разрешение на совместный доступ  NULL, // Указатель на атрибуты безопасности  OPEN\_EXISTING, // Режим открытия файла  FILE\_ATTRIBUTE\_NORMAL, // Атрибуты файла  NULL // Указатель на шаблон файла  );  if (file == INVALID\_HANDLE\_VALUE) {  cerr << "Ошибка при открытии файла: " << path << endl;  return FALSE;  }  DWORD bytesRead;  const DWORD bufferSize = 1024;  char buffer[bufferSize];  cout << "\nСодержимое файла:" << endl;  while (ReadFile(file, buffer, bufferSize, &bytesRead, NULL) && bytesRead > 0) {  cout.write(buffer, bytesRead);  }  CloseHandle(file);  return TRUE;  }  int main() {  setlocale(LC\_ALL, "RUS");  const char\* fileName = FILE\_PATH;  cout << "Информация о файле:" << endl;  if (!printFileInfo(fileName)) {  cerr << "Не удалось получить информацию о файле." << endl;  }  if (!printFileTxt(fileName)) {  cerr << "Не удалось прочитать содержимое файла." << endl;  }  return 0;  } |



**Задание 02.Windows**

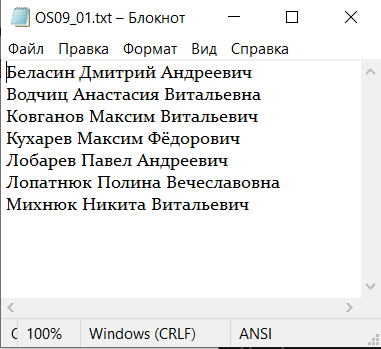
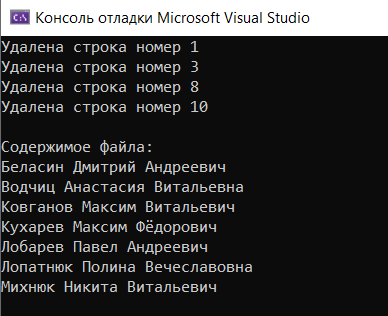
1. Разработайте приложение **OS09\_02**.
2. Приложение **OS09\_02** вызывает функцию **delRowFileTxt**, имеющую следующий прототип.



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).

|  |
| --- |
| #include <windows.h>  #include <iostream>  #include <iomanip>  using namespace std;  #define FILE\_PATH L"D:\\Уник\\1sem\\ОС\\лаба9-ос\\OS09\_01.txt"  BOOL delRowFileTxt(LPWSTR FileName, DWORD row) {  HANDLE file = CreateFileW(  FileName,  GENERIC\_READ | GENERIC\_WRITE,  FILE\_SHARE\_READ,  NULL,  OPEN\_EXISTING,  FILE\_ATTRIBUTE\_NORMAL,  NULL  );  if (file == INVALID\_HANDLE\_VALUE) {  cerr << "Ошибка при открытии файла: " << FileName << endl;  return FALSE;  }  DWORD fileSize = GetFileSize(file, NULL);  if (fileSize == INVALID\_FILE\_SIZE) {  cerr << "Не удалось получить размер файла." << endl;  CloseHandle(file);  return FALSE;  }  char\* buffer = new char[fileSize + 1];  DWORD bytesRead;  if (!ReadFile(file, buffer, fileSize, &bytesRead, NULL) || bytesRead != fileSize) {  cerr << "Не удалось прочитать файл." << endl;  delete[] buffer;  CloseHandle(file);  return FALSE;  }  CloseHandle(file);  char\* newBuffer = new char[fileSize + 1]; //содержимое файла после удаления строки.  DWORD newBufferSize = 0;  DWORD currentRow = 1;  DWORD i = 0;  bool rowDeleted = false;  while (i < fileSize) {  if (buffer[i] == '\n') {  if (currentRow == row) {  rowDeleted = true;  }  else {  newBuffer[newBufferSize++] = buffer[i];  }  currentRow++;  }  else {  if (currentRow != row || rowDeleted) {  newBuffer[newBufferSize++] = buffer[i];  }  }  i++;  }  file = CreateFileW(  FileName,  GENERIC\_WRITE,  0,  NULL,  CREATE\_ALWAYS,  FILE\_ATTRIBUTE\_NORMAL,  NULL  );  if (file == INVALID\_HANDLE\_VALUE) {  cerr << "Ошибка при открытии файла для записи." << endl;  delete[] buffer;  delete[] newBuffer;  return FALSE;  }  DWORD bytesWritten;  WriteFile(file, newBuffer, newBufferSize, &bytesWritten, NULL);  CloseHandle(file);  delete[] buffer;  delete[] newBuffer;  return TRUE;  }  BOOL printFileTxt(LPCWSTR path) {  HANDLE file = CreateFileW(  path,  GENERIC\_READ,  FILE\_SHARE\_READ,  NULL,  OPEN\_EXISTING,  FILE\_ATTRIBUTE\_NORMAL,  NULL  );  if (file == INVALID\_HANDLE\_VALUE) {  cerr << "Ошибка при открытии файла: " << path << endl;  return FALSE;  }  DWORD bytesRead;  const DWORD bufferSize = 1024;  char buffer[bufferSize];  cout << "\nСодержимое файла:" << endl;  while (ReadFile(file, buffer, bufferSize, &bytesRead, NULL) && bytesRead > 0) {  cout.write(buffer, bytesRead);  }  CloseHandle(file);  return TRUE;  }  int main() {  setlocale(LC\_ALL, "RUS");  const wchar\_t\* fileName = FILE\_PATH;  for (DWORD row : {1, 3, 8, 10}) {  if (!delRowFileTxt(const\_cast<LPWSTR>(fileName), row)) {  cerr << "Не удалось удалить строку номер " << row << endl;  }  else {  cout << "Удалена строка номер " << row << endl;  }  }  printFileTxt(fileName);  return 0;  } |

1. Продемонстрируйте работоспособность приложения **OS09\_02**.



**Задание 03.Windows**

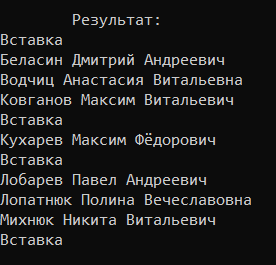
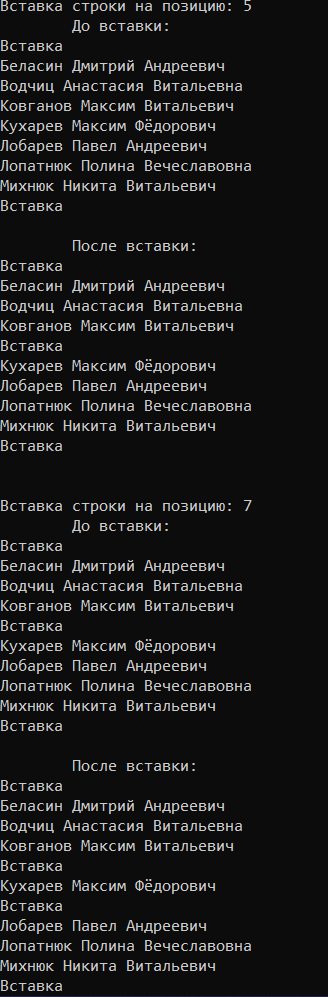
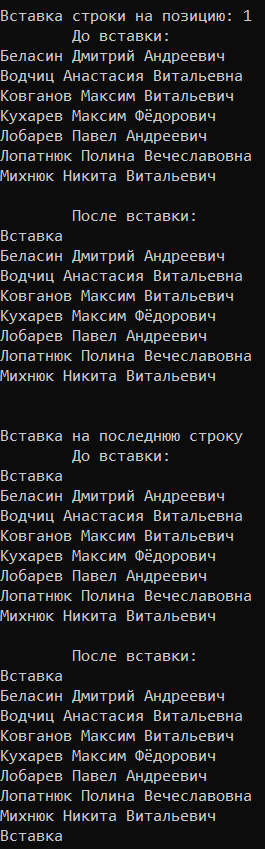
1. Разработайте приложение **OS09\_03.**
2. Приложение **OS09\_03** вызывает функцию **insRowFileTxt**, имеющую следующий прототип.



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).

|  |
| --- |
| #include <Windows.h>  #include <iostream>  #define FILE\_PATH L"D:\\Уник\\1sem\\ОС\\лаба9-ос\\OS09\_01.txt"  #define READ\_BYTES 1000  using namespace std;  BOOL printFileText(LPWSTR fileName)  {  try  {  cout << "\tРезультат:\n";  HANDLE hf = CreateFile(fileName, GENERIC\_READ, NULL, NULL, OPEN\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);  if (hf == INVALID\_HANDLE\_VALUE) throw "Не удалось создать или открыть файл.";  DWORD n = NULL;  char buf[1024];  ZeroMemory(buf, sizeof(buf));  BOOL b = ReadFile(hf, &buf, READ\_BYTES, &n, NULL);  if (!b) throw "Не удалось прочитать файл";  cout << buf << endl;  CloseHandle(hf);  return true;  }  catch (const char\* em)  {  cout << "Ошибка:" << em << endl;  return false;  }  }  BOOL insRowFileTxt(LPWSTR fileName, LPWSTR str, DWORD row)  {  char stringToInsert[50];  wcstombs(stringToInsert, str, 50);  if (row == -1) {  cout << "\nВставка на последнюю строку\n";  }  else {  cout << "\nВставка строки на позицию: " << 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 "Не удалось создать или открыть файл";  }  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 "Не удалось прочитать файл";  }  cout << "\tДо вставки:\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;  for (int i = 0; i < n; i++)  {  if (line == row)  {  for (int k = 0; k < sizeof(stringToInsert); k++)  {  editedBuf[j] = stringToInsert[k];  j++;  if (stringToInsert[k + 1] == '\0')  {  editedBuf[j] = '\r';  j++;  editedBuf[j] = '\n';  j++;  row = 0;  break;  }  }  i--;  }  else  {  editedBuf[j] = buf[i];  j++;  }  if (buf[i] == '\n')  line++;  if (buf[i + 1] == '\0' && row == -1)  {  for (int k = 0; k < sizeof(stringToInsert); k++)  {  editedBuf[j] = stringToInsert[k];  j++;  if (stringToInsert[k + 1] == '\0')  {  editedBuf[j] = '\r';  j++;  editedBuf[j] = '\n';  j++;  row = 0;  break;  }  }  }  }  b = WriteFile(hAppend, editedBuf, j, &n, NULL);  if (!b)  {  CloseHandle(hAppend);  throw "Не удалось записать в файл\n";  }  cout << "\tПосле вставки:\n";  cout << editedBuf << endl;  CloseHandle(hAppend);  return true;  }  catch (const char\* em)  {  cout << em << " \n";  return false;  }  }  int main()  {  setlocale(0, "ru");  LPWSTR file = (LPWSTR)FILE\_PATH;  char str[] = "Вставка";  wchar\_t wStr[50];  mbstowcs(wStr, str, strlen(str) + 1);  LPWSTR strToIns = wStr;  insRowFileTxt(file, strToIns, 1); // Вставка на 1-ю строку  insRowFileTxt(file, strToIns, -1); // Вставка в конец  insRowFileTxt(file, strToIns, 5); // Вставка на 5-ю строку  insRowFileTxt(file, strToIns, 7); // Вставка на 7-ю строку  printFileText(file);  } |

1. Продемонстрируйте работоспособность приложения **OS09\_03**.



**Задание 04.Windows**

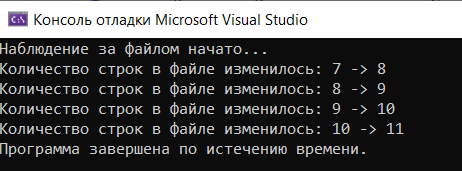
1. Разработайте приложение **OS09\_04.**
2. Приложение **OS09\_04** вызывает функцию **printWathRowFileTxt**, имеющую следующий прототип.



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.

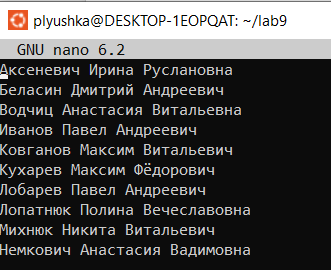
|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <Windows.h>  #include <iostream>  #define FILE\_PATH L"D:\\Уник\\1sem\\ОС\\лаба9-ос\\OS09\_01.txt"  #define READ\_BYTES 1000  using namespace std;  DWORD countLinesInFile(LPWSTR fileName)  {  DWORD lineCount = 0;  try  {  HANDLE hf = CreateFile(fileName, GENERIC\_READ, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);  if (hf == INVALID\_HANDLE\_VALUE)  throw "Не удалось открыть файл.";  char buf[1024];  DWORD n = 0;  ZeroMemory(buf, sizeof(buf));  BOOL b = ReadFile(hf, &buf, sizeof(buf), &n, NULL);  if (!b)  throw "Не удалось прочитать файл.";  for (DWORD i = 0; i < n; i++)  {  if (buf[i] == '\n')  lineCount++;  }  CloseHandle(hf);  }  catch (const char\* em)  {  cout << "Ошибка: " << em << endl;  return -1;  }  return lineCount;  }  BOOL printWatchRowFileTxt(LPWSTR fileName, DWORD mIsec)  {  try  {  HANDLE hDir = FindFirstChangeNotification(L"D:\\Уник\\1sem\\ОС\\лаба9-ос", FALSE, FILE\_NOTIFY\_CHANGE\_LAST\_WRITE);  if (hDir == INVALID\_HANDLE\_VALUE)  throw "Не удалось открыть каталог для наблюдения.";  DWORD prevLineCount = countLinesInFile(fileName);  if (prevLineCount == -1)  throw "Не удалось подсчитать количество строк.";  cout << "Наблюдение за файлом начато..." << endl;  DWORD dwWaitStatus;  DWORD startTime = GetTickCount64();  while (true)  {  DWORD elapsedTime = GetTickCount64() - startTime;  if (elapsedTime >= mIsec)  {  cout << "Программа завершена по истечению времени." << endl;  break;  }  dwWaitStatus = WaitForSingleObject(hDir, mIsec);  if (dwWaitStatus == WAIT\_OBJECT\_0)  {  DWORD currentLineCount = countLinesInFile(fileName);  if (currentLineCount != prevLineCount)  {  if (currentLineCount == -1)  {  cout << "Ошибка: Не удалось прочитать файл." << endl;  break;  }  cout << "Количество строк в файле изменилось: " << prevLineCount << " -> " << currentLineCount << endl;  prevLineCount = currentLineCount;  }  if (!FindNextChangeNotification(hDir))  {  cout << "Ошибка при повторном ожидании изменений." << endl;  break;  }  }  else if (dwWaitStatus == WAIT\_TIMEOUT)  {  continue;  }  else  {  cout << "Ошибка при ожидании изменений." << endl;  break;  }  }  CloseHandle(hDir);  return TRUE;  }  catch (const char\* em)  {  cout << em << endl;  return FALSE;  }  }  int main()  {  setlocale(LC\_ALL, "ru");  LPWSTR file = (LPWSTR)FILE\_PATH;  DWORD mIsec = 10000;  if (!printWatchRowFileTxt(file, mIsec))  {  cout << "Программа завершена с ошибкой." << endl;  }  return 0;  } |

1. Продемонстрируйте работоспособность приложения **OS09\_04** совместно с приложениями **OS09\_03** и **OS09\_04**.



**Задание 05.Linux**

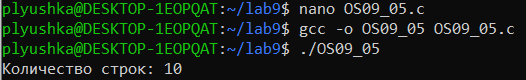
1. Создайте текстовый файл **OS09\_05.txt,** аналогичный файлу **OS09\_01.txt** (п.2).



1. Разработайте приложение **OS09\_05,** подсчитывающее количество строк и выводящее это значение в стандартный поток.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  int main() {  FILE \*file = fopen("./OS09\_05.txt", "r");  if (file == NULL) {  perror("Ошибка при открытии файла");  return 1;  }  int line\_count = 0;  int ch;  while ((ch = fgetc(file)) != EOF) {  if (ch == '\n') {  line\_count++;  }  }  if (ch != EOF) {  line\_count++;  }  fclose(file);  printf("Количество строк: %d\n", line\_count);  return 0;  } |

1. Продемонстрируйте работоспособность приложения **OS09\_05**.

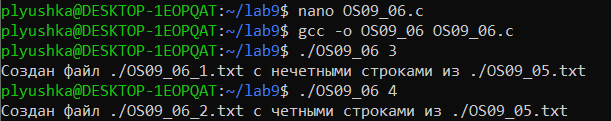


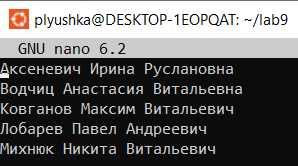
**Задание 06. Linux**

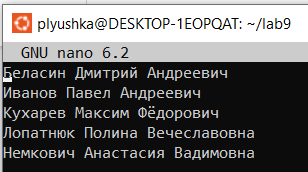
1. Разработайте приложение **OS09\_06,** принимающее 1 параметр, принимающее целочисленное числовое значение.
2. Если числовое значение принимает нечетное значение, то приложение создает новый файл **OS09\_06\_1.txt**, содержащий только нечетные строки из файла **OS09\_05.txt**.
3. Если числовое значение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

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  void process\_file(const char \*input\_file, const char \*output\_file, int is\_odd) {  FILE \*in = fopen(input\_file, "r");  if (in == NULL) {  perror("Ошибка при открытии входного файла");  exit(1);  }  FILE \*out = fopen(output\_file, "w");  if (out == NULL) {  perror("Ошибка при создании выходного файла");  fclose(in);  exit(1);  }  char buffer[1024];  int line\_number = 1;  while (fgets(buffer, sizeof(buffer), in) != NULL) {  if ((line\_number % 2 == 1 && is\_odd) || (line\_number % 2 == 0 && !is\_odd)) {  fputs(buffer, out);  }  line\_number++;  }  fclose(in);  fclose(out);  }  int main(int argc, char \*argv[]) {  if (argc != 2) {  fprintf(stderr, "Использование: %s <целое число>\n", argv[0]);  return 1;  }  int value = atoi(argv[1]);  if (value == 0 && strcmp(argv[1], "0") != 0) {  fprintf(stderr, "Ошибка: Неверный ввод целого числа\n");  return 1;  }  const char \*input\_file = "./OS09\_05.txt";  const char \*output\_file = (value % 2 == 1) ? "./OS09\_06\_1.txt" : "./OS09\_06\_2.txt";  process\_file(input\_file, output\_file, value % 2 == 1);  printf("Создан файл %s с %s строками из %s\n", output\_file, (value % 2 == 1) ? "нечетными" : "четными", input\_file);  return 0;  } |

1. Продемонстрируйте работоспособность приложения **OS09\_06**.



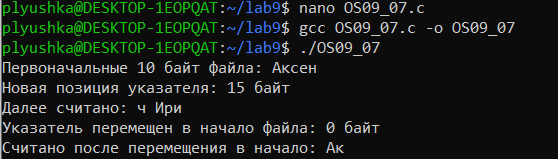




**Задание 07. Linux**

1. Разработайте приложение **OS09\_07,** демонстрирующее возможности функции **lseek.**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <fcntl.h>  int main() {  const char \*file\_name = "./OS09\_05.txt";  int fd = open(file\_name, O\_RDWR);  if (fd == -1) {  perror("Ошибка при открытии файла");  return 1;  }  char buffer[11];  ssize\_t bytes\_read = read(fd, buffer, 10);  if (bytes\_read == -1) {  perror("Ошибка при чтении из файла");  close(fd);  return 1;  }  buffer[bytes\_read] = '\0';  printf("Первоначальные 10 байт файла: %s\n", buffer);  off\_t new\_position = lseek(fd, 5, SEEK\_CUR);  if (new\_position == -1) {  perror("Ошибка при перемещении указателя с помощью lseek");  close(fd);  return 1;  }  printf("Новая позиция указателя: %ld байт\n", new\_position);  bytes\_read = read(fd, buffer, 10);  if (bytes\_read == -1) {  perror("Ошибка при чтении из файла");  close(fd);  return 1;  }  buffer[bytes\_read] = '\0';  printf("Далее считано: %s\n", buffer);  new\_position = lseek(fd, 0, SEEK\_SET);  if (new\_position == -1) {  perror("Ошибка при перемещении указателя в начало файла");  close(fd);  return 1;  }  printf("Указатель перемещен в начало файла: %ld байт\n", new\_position);  bytes\_read = read(fd, buffer, 5);  if (bytes\_read == -1) {  perror("Ошибка при чтении из файла");  close(fd);  return 1;  }  buffer[bytes\_read] = '\0';  printf("Считано после перемещения в начало: %s\n", buffer);  close(fd);  return 0;  } |



1. Поясните назначение и принцип работы функции **lseek**.

* Открытие файла: Открываем файл для чтения и записи с использованием функции open.
* Чтение первых 10 байт: Читаем первые 10 байт с помощью read и выводим их.
* Перемещение указателя: Используем lseek для перемещения указателя на 5 байт от текущей позиции с флагом SEEK\_CUR. Далее, снова считываем данные.
* Перемещение указателя в начало файла: С помощью lseek с флагом SEEK\_SET перемещаем указатель в начало файла и снова считываем данные.
* Закрытие файла: Закрываем файл с помощью close.

Функция **lseek** используется для перемещения указателя в файле, который связан с дескриптором, в указанную позицию. Это позволяет контролировать, с какой позиции в файле будет происходить дальнейшее чтение или запись данных.

**off\_t lseek(int fd, off\_t offset, int whence);**

**Параметры:**

**fd** — дескриптор файла, который был открыт с помощью функции open(). Это идентификатор, который система использует для работы с файлом.

**offset** — смещение, на которое нужно переместить указатель в файле. Для положительных значений смещение идет вперед (в сторону конца файла). Для отрицательных значений — в сторону начала файла.

**whence** — базовая позиция для смещения, которая определяет, откуда будет производиться смещение:

* **SEEK\_SET** — отсчитывает смещение от начала файла.
* **SEEK\_CUR** — отсчитывает смещение от текущей позиции указателя.
* **SEEK\_END** — отсчитывает смещение от конца файла.

**Возвращаемое значение:**

* Если операция успешна, возвращается новая позиция указателя в файле (в байтах).
* Если операция не удалась, возвращается -1 и устанавливается ошибка, которая может быть проверена с помощью errno.

**Принцип работы:**

Функция lseek изменяет текущую позицию указателя в файле, что позволяет указать, с какого места в файле нужно читать или записывать данные. В отличие от операций чтения и записи, которые обычно изменяют данные, lseek только изменяет позицию, не изменяя содержимое файла.

**Задание 08.** Ответьте на следующие вопросы

1. Что такое файл?
2. Перечислите основные характеристики (атрибуты) файла.
3. Что такое файловая система?
4. Перечислите основные функции файловой системы.
5. Перечислите 3 названия файловой системы.
6. Какая файловая система установлена на вашем компьютере под Windows? под Linux?
7. Что такое каталог файловой системы? перечислите наименования специальных каталогов.
8. Поясните понятие «текущий каталог приложения».
9. Что такое специальные имена файлов? перечислите их, для чего они нужны.
10. Для чего используются буферы ввода-вывода?
11. Поясните понятие «кэширование».
12. Поясните понятие «указатель позиции файла».
13. Поясните понятие «маркер конца файла».
14. Поясните понятие «блокировка файла».
15. Windows. Функция OS API для создания файла.
16. Windows. Функция OS API для открытия файла.
17. Windows. Функция OS API для удаления файла.
18. Windows. Функция OS API для записи в файл.
19. Windows. Функция OS API для чтения файла.
20. Windows. Назначение и отличие функций OS API: CopyFile, MoveFile, ReplaseFile.
21. Windows. Перечислите функции OS API, которые изменяют текущее значение указателя позиции файла.
22. Windows. Перечислите функции OS API для блокировки и разблокировки файлов.
23. Windows. Поясните механизм «наблюдение за каталогом», перечислите набор функций OS API, позволяющий реализовать этот механизм.
24. Windows. Перечислите функции OS API для работы с каталогами, поясните их назанчения.
25. Linux. Что такое FHS?
26. Linux. Перечислите типы файловых систем.
27. Linux. Что такое inode?
28. Linux. Поясните назначение функций open, read, write, close, ioctl, stat, flush**,** lseek, lstat, fstat.