Лабораторная работа 09

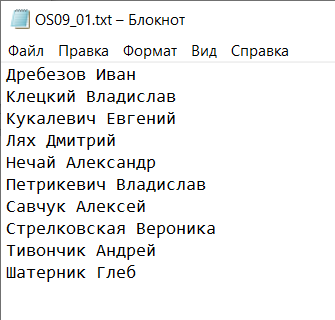
Файловая система

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\_EXISTING,  FILE\_ATTRIBUTE\_NORMAL,  NULL);  if (file == INVALID\_HANDLE\_VALUE) {  wcerr << L"Не удалось открыть файл: " << path << endl;  return false;  }  SYSTEMTIME sysTime;  BY\_HANDLE\_FILE\_INFORMATION fi;  BOOL fResult = GetFileInformationByHandle(file, &fi);  if (fResult)  {  wcout << L"File name:\t" << getFileName(path);  wcout << L"\nFile type:\t " << (fileData.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY ? L"Directory" : L"File");  wcout << L"\nFile size:\t" << fi.nFileSizeLow << L" bytes";  FileTimeToSystemTime(&fi.ftCreationTime, &sysTime);  wcout << L"\nCreate time:\t" << setfill(L'0') << setw(2) << sysTime.wHour + 3 << L':' << setw(2) << sysTime.wMinute << L':' << setw(2) << sysTime.wSecond;  wcout << L" " << setw(4) << sysTime.wYear << L'-' << setw(2) << sysTime.wMonth << L'-' << setw(2) << sysTime.wDay;  FileTimeToSystemTime(&fi.ftLastWriteTime, &sysTime);  wcout << L"\nUpdate time:\t" << setfill(L'0') << setw(2) << sysTime.wHour + 3 << L':' << setw(2) << sysTime.wMinute << L':' << setw(2) << sysTime.wSecond;  wcout << L" " << setw(4) << sysTime.wYear << L'-' << setw(2) << sysTime.wMonth << L'-' << setw(2) << sysTime.wDay;  }  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.

|  |
| --- |
| BOOL printFileTxt(LPWSTR path)  {  wifstream file(path, ios::binary);  file.imbue(locale(locale::empty(), new codecvt\_utf8<wchar\_t>));  if (!file.is\_open()) {  wcerr << L"Не удалось открыть файл для чтения: " << path << endl;  return false;  }  wcout << L"\n\n\tPrint file:\n";  wstring line;  while (getline(file, line)) {  wcout << line << endl;  }  file.close();  return true;  } |

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

|  |
| --- |
| #include <Windows.h>  #include <iostream>  #include <string>  #include <iomanip>  #include <fstream>  #include <codecvt>  #include <locale>  using namespace std;  #define FILE\_PATH L"C:/Лабы/Операционные системы/Лабы/Лаба9/lab9/OS09\_01.txt"  WIN32\_FIND\_DATA fileData;  wstring getFileName(wchar\_t\* filePath) //извлекает имя фацле из пути к нему  {  wstring ws(filePath); //Преобразует filePath в строку wstring  const size\_t last\_slash\_idx = ws.find\_last\_of(L"\\/"); //Находит последний символ / или \, чтобы отделить имя файла от пути  if (wstring::npos != last\_slash\_idx)  ws.erase(0, last\_slash\_idx + 1); //Удаляет все символы до и включая последний слэш и возвращает только имя файла  return ws;  }  LPCWSTR getFileType(HANDLE file) //определение типа файла  {  switch (GetFileType(file))  {  case FILE\_TYPE\_UNKNOWN:  return L"FILE\_TYPE\_UNKNOWN";  case FILE\_TYPE\_DISK:  return L"FILE\_TYPE\_DISK";  case FILE\_TYPE\_CHAR:  return L"FILE\_TYPE\_CHAR";  case FILE\_TYPE\_PIPE:  return L"FILE\_TYPE\_PIPE";  case FILE\_TYPE\_REMOTE:  return L"FILE\_TYPE\_REMOTE";  default:  return L"[ERROR]: WRITE FILE TYPE";  }  }  BOOL printFileInfo(LPWSTR path) //открывает файли выводит инфу  {  HANDLE file = CreateFile( //открытие или создание файла для чтения  path,  GENERIC\_READ,  NULL,  NULL,  OPEN\_EXISTING,  FILE\_ATTRIBUTE\_NORMAL,  NULL);  if (file == INVALID\_HANDLE\_VALUE) { //если файл не открыт  wcerr << L"Не удалось открыть файл: " << path << endl;  return false;  }  SYSTEMTIME sysTime; //получение информации о файле  BY\_HANDLE\_FILE\_INFORMATION fi; //переменная для хранения  BOOL fResult = GetFileInformationByHandle(file, &fi); //GetFileInformationByHandle заполняет структуру fi информацией о файле  if (fResult) //если информация получена  {  wcout << L"File name:\t" << getFileName(path);  wcout << L"\nFile type:\t " << (fileData.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY ? L"Directory" : L"File"); //является ли файл директорией или обычным файлом  wcout << L"\nFile size:\t" << fi.nFileSizeLow << L" bytes";  //Конвертирует время создания файла в читаемый формат и выводит его  FileTimeToSystemTime(&fi.ftCreationTime, &sysTime);  wcout << L"\nCreate time:\t" << setfill(L'0') << setw(2) << sysTime.wHour + 3 << L':' << setw(2) << sysTime.wMinute << L':' << setw(2) << sysTime.wSecond;  wcout << L" " << setw(4) << sysTime.wYear << L'-' << setw(2) << sysTime.wMonth << L'-' << setw(2) << sysTime.wDay;  //Выводит время последнего изменения файла. В конце закрывает дескриптор файла.  FileTimeToSystemTime(&fi.ftLastWriteTime, &sysTime);  wcout << L"\nUpdate time:\t" << setfill(L'0') << setw(2) << sysTime.wHour + 3 << L':' << setw(2) << sysTime.wMinute << L':' << setw(2) << sysTime.wSecond;  wcout << L" " << setw(4) << sysTime.wYear << L'-' << setw(2) << sysTime.wMonth << L'-' << setw(2) << sysTime.wDay;  }  CloseHandle(file);  return true;  }  BOOL printFileTxt(LPWSTR path)  {  //Открывает текстовый файл и выводит его содержимое  wifstream file(path, ios::binary); //wifstream для работы с широкими строками  file.imbue(locale(locale::empty(), new codecvt\_utf8<wchar\_t>)); //устанавливает кодировку UTF-8  if (!file.is\_open()) { //открыт ли файл  wcerr << L"Не удалось открыть файл для чтения: " << path << endl;  return false;  }  //читает построчно и выводит инфу  wcout << L"\n\n\tPrint file:\n";  wstring line;  while (getline(file, line)) {  wcout << line << endl;  }  file.close();  return true;  }  int main()  {  //кодировка для русского языка  SetConsoleOutputCP(1251);  SetConsoleCP(1251);  setlocale(LC\_ALL, "ru");  LPWSTR path = (LPWSTR)FILE\_PATH; //риводит FILE\_PATH к типу LPWSTR  printFileInfo(path); //вывод инфы о файле  printFileTxt(path); //содержимое  return 0;  } |
|  |

**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[260];  wcstombs(filepath, fileName, 260);  wcout << L"\n====== Удаление строки: " << row << L"\n\n";  try  {  //открытие файла для чтения  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";  }  //чтение в буфер  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";  }  //Преобразует данные в wstring  wstring\_convert<codecvt\_utf8<wchar\_t>> conv;  string str(buf, n);  wstring wstr = conv.from\_bytes(str);  //вывод и закрытие файла  wcout << L"\t\tДО:\n" << wstr << endl;  CloseHandle(hf);  HANDLE hAppend = CreateFile(fileName, GENERIC\_WRITE, NULL, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);  if (hAppend == INVALID\_HANDLE\_VALUE)  throw "[ERROR] Unable to open file for writing.";  //Переводит wstring в wstringstream для построчного чтения  wstringstream stream(wstr);  //переменные для обработки строк  wstring line;  wstring editedWstr;  DWORD currentRow = 1;  //построчное чтение файла  while (getline(stream, line))  {  if (currentRow != row)  {  editedWstr += line + L"\n"; //запись в editedWstr, если номера не совпадают с row  }  currentRow++;  }  //существует ли строка для удаления  if (currentRow <= row)  {  CloseHandle(hAppend);  throw "[ERROR] Can't find this row.\n";  }  string editedStr = conv.to\_bytes(editedWstr); //преобразование строки в string  DWORD written;  b = WriteFile(hAppend, editedStr.c\_str(), (DWORD)editedStr.size(), &written, NULL); //записывает её в файл  if (!b)  {  CloseHandle(hAppend);  throw "[ERROR] Write file unsuccessful";  }  //результат после удаления  wcout << L"\n\t\tПОСЛЕ:\n" << editedWstr << endl;  CloseHandle(hAppend);  wcout << L"\n==========================================\n";  return true;  }  catch (const char\* em)  {  wcout << em << L" \n";  wcout << L"==========================================\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**.

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS //отключает предупреждение о небезопасных функциях  #include <Windows.h>  #include <iostream>  #include <locale>  #include <codecvt>  #include <sstream>  using namespace std;  #define FILE\_PATH L"C:/Лабы/Операционные системы/Лабы/Лаба9/lab9/OS09\_01.txt"  #define READ\_BYTES 500 //максимальное количество байт, которое будет прочитано из файла  BOOL printFileText(LPWSTR fileName) //открывает файл и выводит содержимое  {  try  {  wcout << L"\n\n\t------[RESULT]------\n";  HANDLE hf = CreateFile(fileName, GENERIC\_READ, NULL, NULL, OPEN\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL); //открытие файла  if (hf == INVALID\_HANDLE\_VALUE) throw "[ERROR] Create or open file failed."; //если не открылся  DWORD n = NULL;  char buf[1024];  ZeroMemory(buf, sizeof(buf));  BOOL b = ReadFile(hf, &buf, READ\_BYTES, &n, NULL); //читает данные из файла  if (!b) throw "[ERROR] Read file failed"; //если не проситалось  string str(buf, n); //преобразует прочитанные байты в строку  wstring\_convert<codecvt\_utf8<wchar\_t>> conv; //преобразует в wstring ????зачем  wstring wstr = conv.from\_bytes(str);  //вывод содержимого и закрытие дескриптора  wcout << wstr << endl;  CloseHandle(hf);  return true;  }  catch (const char\* em)  {  wcout << em << endl;  return false;  }  }  //удаление строк  BOOL delRowFileTxt(LPWSTR fileName, DWORD row)  {  char filepath[260]; //преобразование filepath в char массив???? что это и зачем  wcstombs(filepath, fileName, 260);  wcout << L"\n---------Удаление строки: " << row << L"\n\n";  try  {  //открытие файла для чтения  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";  }  //чтение в буфер  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";  }  //Преобразует данные в wstring  wstring\_convert<codecvt\_utf8<wchar\_t>> conv;  string str(buf, n);  wstring wstr = conv.from\_bytes(str);  //вывод и закрытие файла  wcout << L"\t\tДО:\n" << wstr << endl;  CloseHandle(hf);  //Открывает файл для записи, предварительно удаляя его содержимое.  HANDLE hAppend = CreateFile(fileName, GENERIC\_WRITE, NULL, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);  if (hAppend == INVALID\_HANDLE\_VALUE)  throw "[ERROR] Unable to open file for writing.";  //Переводит wstring в wstringstream для построчного чтения  wstringstream stream(wstr);  //переменные для обработки строк  wstring line;  wstring editedWstr;  DWORD currentRow = 1;  //построчное чтение файла  while (getline(stream, line))  {  if (currentRow != row)  {  editedWstr += line + L"\n"; //запись в editedWstr, если номера не совпадают с row  }  currentRow++;  }  //существует ли строка для удаления  if (currentRow <= row)  {  CloseHandle(hAppend);  throw "[ERROR] Can't find this row.\n";  }  string editedStr = conv.to\_bytes(editedWstr);//преобразование строки в string  DWORD written;  b = WriteFile(hAppend, editedStr.c\_str(), (DWORD)editedStr.size(), &written, NULL); //записывает её в файл  if (!b)  {  CloseHandle(hAppend);  throw "[ERROR] Write file unsuccessful";  }  //результат после удаления  wcout << L"\n\t\tПОСЛЕ:\n" << editedWstr << endl;  CloseHandle(hAppend);  wcout << L"\n==========================================\n";  return true;  }  catch (const char\* em)  {  wcout << em << L" \n";  wcout << L"==========================================\n";  return false;  }  }  int main()  {  //установка кодировки  SetConsoleOutputCP(1251);  SetConsoleCP(1251);  setlocale(LC\_ALL, "ru");  LPWSTR file = (LPWSTR)FILE\_PATH; //риводит FILE\_PATH к типу LPWSTR??? зачем  //указание какие строки удалить  delRowFileTxt(file, 1);  delRowFileTxt(file, 3);  delRowFileTxt(file, 8);  delRowFileTxt(file, 10);  printFileText(file);//вывод содержимого  return 0;  } |
|  |

**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  {  wcout << L"\n----------Insert row: " << row << L"\n\n";  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";  }  //чтение данных в файл  DWORD n = NULL;  char buf[1024];  BOOL b;  ZeroMemory(buf, sizeof(buf));  b = ReadFile(hf, &buf, sizeof(buf) - 1, &n, NULL); // Читаем с ограничением  if (!b)  {  CloseHandle(hf);  throw "[ERROR] Read file unsuccessful";  }  buf[n] = '\0'; // Завершаем строку  wstring\_convert<codecvt\_utf8<wchar\_t>> conv; //преобразовние данных в wstring  string strBuf(buf);  wstring wstrBuf = conv.from\_bytes(strBuf);  //данные до изменения  wcout << L"\t\tBEFORE:\n" << wstrBuf << endl;  CloseHandle(hf);  //открытие для записи  HANDLE hAppend = CreateFile(fileName, GENERIC\_WRITE, NULL, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);  if (hAppend == INVALID\_HANDLE\_VALUE)  throw "[ERROR] Unable to open file for writing.";  wstringstream stream(wstrBuf); //Переводит wstring в wstringstream для построчного чтения.  wstring line;  wstring editedBuf;  DWORD currentRow = 1;  //построчное чтение  while (getline(stream, line))  {  if (currentRow == row) //если номер текущей строки совпадает с row, вставляет строку  {  editedBuf += str;  editedBuf += L"\r\n";  }  editedBuf += line + L"\r\n";  currentRow++;  }  if (currentRow <= row) //Если указанная строка больше, чем общее количество строк, добавляет новую строку в конец  {  editedBuf += str;  editedBuf += L"\r\n";  }  string editedStr = conv.to\_bytes(editedBuf); //преобразование строки в string  DWORD written;  //запись строки в файл  b = WriteFile(hAppend, editedStr.c\_str(), (DWORD)editedStr.size(), &written, NULL);  if (!b)  {  CloseHandle(hAppend);  throw "[ERROR] Write file unsuccessful";  }  //данные после изменения  wcout << L"\t\tAFTER:\n" << editedBuf << endl;  CloseHandle(hAppend);  wcout << L"\n==========================================\n";  return true;  }  catch (const char\* em)  {  wcout << em << L" \n";  wcout << L"==========================================\n";  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**.

|  |
| --- |
| 1. #define \_CRT\_SECURE\_NO\_WARNINGS 2. #include <Windows.h> 3. #include <iostream> 4. #include <locale> 5. #include <codecvt> 6. #include <sstream> 7. #define FILE\_PATH L"C:/Лабы/Операционные системы/Лабы/Лаба9/lab9/OS09\_01.txt" 8. #define READ\_BYTES 1000 9. using namespace std; 10. BOOL printFileText(LPWSTR fileName) 11. { 12. try 13. { 14. wcout << L"\n\n\t---------RESULT----------\n"; 15. HANDLE hf = CreateFile(fileName, GENERIC\_READ, NULL, NULL, OPEN\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL); 16. if (hf == INVALID\_HANDLE\_VALUE) throw "[ERROR] Create or open file failed."; 17. DWORD n = NULL; 18. char buf[1024]; 19. ZeroMemory(buf, sizeof(buf)); 20. BOOL b = ReadFile(hf, &buf, READ\_BYTES, &n, NULL); //чтение данных из файла 21. if (!b) throw "[ERROR] Read file failed"; 22. string str(buf, n); //преобразует байты в строку 23. wstring\_convert<codecvt\_utf8<wchar\_t>> conv; //преобразует в wstring 24. wstring wstr = conv.from\_bytes(str); 25. //вывод содержимого 26. wcout << wstr << endl; 27. CloseHandle(hf); 28. return true; 29. } 30. catch (const char\* em) 31. { 32. wcout << L"[ERROR] " << em << endl; 33. return false; 34. } 35. } 36. //вставка строки в указанный ряд 37. BOOL insRowFileTxt(LPWSTR fileName, LPWSTR str, DWORD row) 38. { 39. try 40. { 41. wcout << L"\n----------Insert row: " << row << L"\n\n"; 42. HANDLE hf = CreateFile(fileName, GENERIC\_READ, NULL, NULL, OPEN\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL); 43. if (hf == INVALID\_HANDLE\_VALUE) 44. { 45. CloseHandle(hf); 46. throw "[ERROR] Create or open file failed"; 47. } 48. //чтение данных в файл 49. DWORD n = NULL; 50. char buf[1024]; 51. BOOL b; 52. ZeroMemory(buf, sizeof(buf)); 53. b = ReadFile(hf, &buf, sizeof(buf) - 1, &n, NULL); // Читаем с ограничением 54. if (!b) 55. { 56. CloseHandle(hf); 57. throw "[ERROR] Read file unsuccessful"; 58. } 59. buf[n] = '\0'; // Завершаем строку 60. wstring\_convert<codecvt\_utf8<wchar\_t>> conv; //преобразовние данных в wstring 61. string strBuf(buf); 62. wstring wstrBuf = conv.from\_bytes(strBuf); 63. //данные до изменения 64. wcout << L"\t\tBEFORE:\n" << wstrBuf << endl; 65. CloseHandle(hf); 66. //открытие для записи 67. HANDLE hAppend = CreateFile(fileName, GENERIC\_WRITE, NULL, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL); 68. if (hAppend == INVALID\_HANDLE\_VALUE) 69. throw "[ERROR] Unable to open file for writing."; 70. wstringstream stream(wstrBuf); //Переводит wstring в wstringstream для построчного чтения. 71. wstring line; 72. wstring editedBuf; 73. DWORD currentRow = 1; 74. //построчное чтение 75. while (getline(stream, line)) 76. { 77. if (currentRow == row) //если номер текущей строки совпадает с row, вставляет строку 78. { 79. editedBuf += str; 80. editedBuf += L"\r\n"; 81. } 82. editedBuf += line + L"\r\n"; 83. currentRow++; 84. } 85. if (currentRow <= row) //Если указанная строка больше, чем общее количество строк, добавляет новую строку в конец 86. { 87. editedBuf += str; 88. editedBuf += L"\r\n"; 89. } 90. string editedStr = conv.to\_bytes(editedBuf); //преобразование строки в string 91. DWORD written; 92. //запись строки в файл 93. b = WriteFile(hAppend, editedStr.c\_str(), (DWORD)editedStr.size(), &written, NULL); 94. if (!b) 95. { 96. CloseHandle(hAppend); 97. throw "[ERROR] Write file unsuccessful"; 98. } 99. //данные после изменения 100. wcout << L"\t\tAFTER:\n" << editedBuf << endl; 101. CloseHandle(hAppend); 102. wcout << L"\n==========================================\n"; 103. return true; 104. } 105. catch (const char\* em) 106. { 107. wcout << em << L" \n"; 108. wcout << L"==========================================\n"; 109. return false; 110. } 111. } 112. int main() 113. { 114. SetConsoleOutputCP(1251); 115. SetConsoleCP(1251); 116. setlocale(LC\_ALL, "ru"); 117. LPWSTR file = (LPWSTR)FILE\_PATH; 118. wchar\_t strToIns[] = L"1. Новая строка"; 119. insRowFileTxt(file, strToIns, 1); 120. insRowFileTxt(file, strToIns, -1); 121. insRowFileTxt(file, strToIns, 5); 122. insRowFileTxt(file, strToIns, 7); 123. printFileText(file); 124. } |
|  |

**Задание 04.Windows**

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



|  |
| --- |
| BOOL printWatchRowFileTxt(LPWSTR FileName, DWORD mlsec, DWORD maxDuration)  {  LARGE\_INTEGER fileSize = { 0 }; //размер файла  int rowCount = 0; //счётчик строк  try {  //создание уведомлений  HANDLE notif = FindFirstChangeNotification(FOLDER\_PATH, FALSE, FILE\_NOTIFY\_CHANGE\_LAST\_WRITE);  if (notif == INVALID\_HANDLE\_VALUE) {  cout << "Ошибка при инициализации уведомления о изменениях." << endl;  return FALSE;  }  DWORD dwWaitStatus; //дя статуса ожидания  DWORD startTime = GetTickCount(); //текущее время с момента запуска системы  while (true) {  if ((GetTickCount() - startTime) >= maxDuration) {  cout << "\nВремя слежения истекло." << endl;  break;  }  dwWaitStatus = WaitForSingleObject(notif, mlsec); //ожидает изменения в папке  switch (dwWaitStatus) {  case WAIT\_OBJECT\_0: {  if (FindNextChangeNotification(notif) == FALSE) { //если не найдено next изменения, выходим из цикла  break;  }  WaitForSingleObject(mutex, INFINITE); //ожидает пока мьютекс булет доступен  HANDLE of = CreateFile(  FileName,  GENERIC\_READ,  FILE\_SHARE\_READ,  NULL,  OPEN\_EXISTING,  FILE\_ATTRIBUTE\_NORMAL,  NULL  );  if (of == INVALID\_HANDLE\_VALUE) {  DWORD dwError = GetLastError();  //cout << "Ошибка: не удалось открыть файл. Код ошибки: " << dwError << endl;  }  else if (GetFileSizeEx(of, &fileSize)) { //получает размер файла  //для буфера, куда будут считаваться данные  char\* buf = new char[(fileSize.QuadPart + 1) \* sizeof(char)];  ZeroMemory(buf, (fileSize.QuadPart + 1) \* sizeof(char));  DWORD n = 0;  if (ReadFile(of, buf, fileSize.QuadPart, &n, NULL)) {  int position = 0; //текущая позиция в буфере  rowCount = 0; //кол-во строк  bool lastLineIsEmpty = false; //флаг была ли последняя строка пустой  while (buf[position] != '\0') {  if (buf[position] == '\n') { //если встречается данный символ, то увеличиваем кол-во строк  rowCount++;  lastLineIsEmpty = false;  }  else {  lastLineIsEmpty = true; //иначе флаг в состояние, когда строка не пустая  }  position++;  }  //Если последняя строка не пустая и не заканчивается символом новой строки, увеличиваем счетчик строк  if (lastLineIsEmpty && position > 0 && buf[position - 1] != '\n') {  rowCount++;  }  }  delete[] buf; //освобождение памяти  CloseHandle(of);  }  else {  cout << "Ошибка при чтении файла." << endl;  }  ReleaseMutex(mutex);  if (rowC != rowCount) { //если кол-во строк изменилось  cout << "\nКоличество строк: " << rowCount;  if (rowCount > rowC) {  cout << " (Строки добавлены)" << endl;  }  else if (rowCount < rowC) {  cout << " (Строки удалены)" << endl;  }  rowC = rowCount;  }  break;  }  default:  break;  }  }  CloseHandle(notif);  }  catch (const char\* err) {  cout << "Ошибка: " << err << "\n";  return false;  }  return true;  } |

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**.

|  |
| --- |
| 1. #define \_CRT\_NON\_CONFORMING\_WCSTOK 2. #define \_CRT\_SECURE\_NO\_WARNINGS 3. #include <iostream> 4. #include <cstdlib> 5. #include <windows.h> 6. #include <ctime> 7. using namespace std; 8. #define FOLDER\_PATH L"C:/Лабы/Операционные системы/Лабы/Лаба9/lab9" 9. #define FILE\_PATH L"C:/Лабы/Операционные системы/Лабы/Лаба9/lab9/OS09\_01.txt" 10. int rowC = 0; //кол-во строк в файле 11. HANDLE mutex; //Дескриптор мьютекса для управления доступом к файлу 12. BOOL printWatchRowFileTxt(LPWSTR FileName, DWORD mlsec, DWORD maxDuration) 13. { 14. LARGE\_INTEGER fileSize = { 0 }; //размер файла 15. int rowCount = 0; //счётчик строк 16. try { 17. //создание уведомлений 18. HANDLE notif = FindFirstChangeNotification(FOLDER\_PATH, FALSE, FILE\_NOTIFY\_CHANGE\_LAST\_WRITE); 19. if (notif == INVALID\_HANDLE\_VALUE) { 20. cout << "Ошибка при инициализации уведомления о изменениях." << endl; 21. return FALSE; 22. } 23. DWORD dwWaitStatus; //дя статуса ожидания 24. DWORD startTime = GetTickCount(); //текущее время с момента запуска системы 25. while (true) { 26. if ((GetTickCount() - startTime) >= maxDuration) { 27. cout << "\nВремя слежения истекло." << endl; 28. break; 29. } 30. dwWaitStatus = WaitForSingleObject(notif, mlsec); //ожидает изменения в папке 31. switch (dwWaitStatus) { 32. case WAIT\_OBJECT\_0: { 33. if (FindNextChangeNotification(notif) == FALSE) { //если не найдено next изменения, выходим из цикла 34. break; 35. } 36. WaitForSingleObject(mutex, INFINITE); //ожидает пока мьютекс булет доступен 37. HANDLE of = CreateFile( 38. FileName, 39. GENERIC\_READ, 40. FILE\_SHARE\_READ, 41. NULL, 42. OPEN\_EXISTING, 43. FILE\_ATTRIBUTE\_NORMAL, 44. NULL 45. ); 46. if (of == INVALID\_HANDLE\_VALUE) { 47. DWORD dwError = GetLastError(); 48. //cout << "Ошибка: не удалось открыть файл. Код ошибки: " << dwError << endl; 49. } 50. else if (GetFileSizeEx(of, &fileSize)) { //получает размер файла 51. //для буфера, куда будут считаваться данные 52. char\* buf = new char[(fileSize.QuadPart + 1) \* sizeof(char)]; 53. ZeroMemory(buf, (fileSize.QuadPart + 1) \* sizeof(char)); 54. DWORD n = 0; 55. if (ReadFile(of, buf, fileSize.QuadPart, &n, NULL)) { 56. int position = 0; //текущая позиция в буфере 57. rowCount = 0; //кол-во строк 58. bool lastLineIsEmpty = false; //флаг была ли последняя строка пустой 59. while (buf[position] != '\0') { 60. if (buf[position] == '\n') { //если встречается данный символ, то увеличиваем кол-во строк 61. rowCount++; 62. lastLineIsEmpty = false; 63. } 64. else { 65. lastLineIsEmpty = true; //иначе флаг в состояние, когда строка не пустая 66. } 67. position++; 68. } 69. //Если последняя строка не пустая и не заканчивается символом новой строки, увеличиваем счетчик строк 70. if (lastLineIsEmpty && position > 0 && buf[position - 1] != '\n') { 71. rowCount++; 72. } 73. } 74. delete[] buf; //освобождение памяти 75. CloseHandle(of); 76. } 77. else { 78. cout << "Ошибка при чтении файла." << endl; 79. } 80. ReleaseMutex(mutex); 81. if (rowC != rowCount) { //если кол-во строк изменилось 82. cout << "\nКоличество строк: " << rowCount; 83. if (rowCount > rowC) { 84. cout << " (Строки добавлены)" << endl; 85. } 86. else if (rowCount < rowC) { 87. cout << " (Строки удалены)" << endl; 88. } 89. rowC = rowCount; 90. } 91. break; 92. } 93. default: 94. break; 95. } 96. } 97. CloseHandle(notif); 98. } 99. catch (const char\* err) { 100. cout << "Ошибка: " << err << "\n"; 101. return false; 102. } 103. return true; 104. } 105. int main() 106. { 107. setlocale(LC\_ALL, "ru"); 108. SetConsoleCP(1251); 109. SetConsoleOutputCP(1251); 110. mutex = CreateMutex(NULL, FALSE, L"FileAccessMutex"); //создание мьютекса для управления доступом к файлу 111. if (mutex == NULL) { 112. cout << "Ошибка создания мьютекса." << endl; 113. return 1; 114. } 115. LPWSTR fileName = (LPWSTR)FILE\_PATH; //сохраняет путь к файлу 116. DWORD waitTime = 1000; //время ожидания 117. DWORD maxDuration = 10000; //max время слежения 118. if (!printWatchRowFileTxt(fileName, waitTime, maxDuration)) { 119. cout << "Ошибка при отслеживании изменений в файле." << endl; 120. } 121. CloseHandle(mutex); 122. return 0; 123. } |
|  |

**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**.

|  |
| --- |
| 1. #include <stdio.h> 2. #include <stdlib.h> 3. #include <sys/types.h> 4. #include <time.h> 5. #include <unistd.h> 6. #include <sys/stat.h> 7. #include <fcntl.h> 8. #include <locale.h> 9. int main() 10. { 11. char \*locale = setlocale(LC\_ALL, "ru"); 12. char c; //текущий читаемый символ 13. int rowNumber = 0; 14. int in = open("./OS09\_05.txt", O\_RDONLY); //только для чтения. Если успешно, возвращается 1 16. while (read(in,&c,1) == 1) //читает один байт из файла и записывает в переменную с 17. if(c == '\n') rowNumber++; 18. printf("Number of lines: %d \n",rowNumber); 19. return 0; 20. } |
|  |

**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**.

|  |
| --- |
| 1. #include <stdio.h> 2. #include <stdlib.h> 3. #include <sys/types.h> 4. #include <time.h> 5. #include <unistd.h> 6. #include <sys/stat.h> 7. #include <fcntl.h> 8. #include <locale.h> 9. #include <stdbool.h> 10. int main(int argc, char \*argv[]) 11. { 12. char \*locale = setlocale(LC\_ALL, ""); 13. char c; 14. int rowNumber = 0; 15. int in = open("./OS09\_05.txt", O\_RDONLY); 16. int out; //дескриптор выходного файла 17. int num; //для параметра, переданного из командной строки 18. bool yaBool = 0; //чётность строк 20. if (argc != 2) 21. { 22. printf("Please, provide an integer parameter.\n"); 23. exit(1); 24. } 25. num = atoi(argv[1]); //преобразование в целое число переданной строки 27. if (num % 2 != 0) 28. out = open("./OS09\_06\_1.txt", O\_CREAT | O\_WRONLY, 0777); 29. else 30. { 31. out = open("./OS09\_06\_2.txt", O\_CREAT | O\_WRONLY, 0777); 32. yaBool = 1; 33. } 35. while (read(in, &c, 1) == 1) 36. { 37. if (!yaBool) 38. write(out, &c, 1); //запись в выходной файл 39. if (c == '\n') 40. yaBool = !yaBool; 41. } 42. close(out); 43. close(in); 44. return 0; 45. } |
|  |

**Задание 07.Linux**

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

|  |
| --- |
| 1. #include <stdio.h> 2. #include <stdlib.h> 3. #include <sys/types.h> 4. #include <time.h> 5. #include <unistd.h> 6. #include <sys/stat.h> 7. #include <fcntl.h> 8. #include <locale.h> 9. #include <stdbool.h> 10. #include <stddef.h> 11. int main(int argc, char \*argv[]) 12. { 13. int fd = open("OS09\_05.txt",O\_RDONLY); 14. if(fd == 1){ 15. perror("Error opening file"); 16. return 1; 17. } 18. char buffer[128]; //для данных, прочитанных из файла 19. ssize\_t bytesRead; //кол-во прочитанных байтов 20. // чтение первых 10 байт 21. lseek(fd,0,SEEK\_SET); // ставим позицию чтения в начало 22. bytesRead = read(fd,buffer,10); 23. buffer[bytesRead] = '\0'; 24. printf("[10 bytes]: %s\n",buffer); 25. // Чтение следующих 10 байт 26. lseek(fd,10,SEEK\_CUR); // смещение на 10 от текущей позиции; 27. bytesRead = read(fd,buffer,10); 28. buffer[bytesRead] = '\0'; 29. printf("[Next 10 bytes]: %s\n",buffer); 31. // чтение последних 30 байт от конца файла 32. lseek(fd,-22,SEEK\_END); 33. bytesRead = read(fd,buffer,30); 34. buffer[bytesRead] = '\0'; 35. printf("[End 30 bytes]: %s\n",buffer); 36. close(fd); 37. return 0; 38. } |
|  |

**Задание 08.** Ответьте на следующие вопросы

1. Что такое файл?

**Файл** – набор логических записей; абстракция для унифицированного доступа к данным.

1. Перечислите основные характеристики (атрибуты) файла.

Имя, размер, тип, дата и время создания, дата и время изменения, права доступа

1. Что такое файловая система?

Система управления файлами, часть ОС, обеспечивающая доступ к фалам. Устанавливает связь между логическим представлением и физическим расположением данных.

1. Перечислите основные функции файловой системы.

создание/удаление каталогов, включение/исключение подкаталогов, включение/исключение файла в каталог, создание/удаление файла, открытие/закрытие доступа к файлу, чтение/запись логических записей файла, установка (поддержка) указателя файла.

1. Перечислите 3 названия файловой системы.

FAT32, NTFS, ext,ext2,ext3, ext4,BtrFS(Oracle), ISO 9660(CD,DVD).

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

Правой кнопкой мыши на Пуск->Управление дисками. NTFS – Windows.

1. Что такое каталог файловой системы? перечислите наименования специальных каталогов.

**Каталог** – файл, содержащий информацию о месте расположения других файлов; специальные каталоги: . (точка), .. (две точки).

**Специальные имена**: con, lpt1, prn, aux, com… (не могут быть именами файлов)

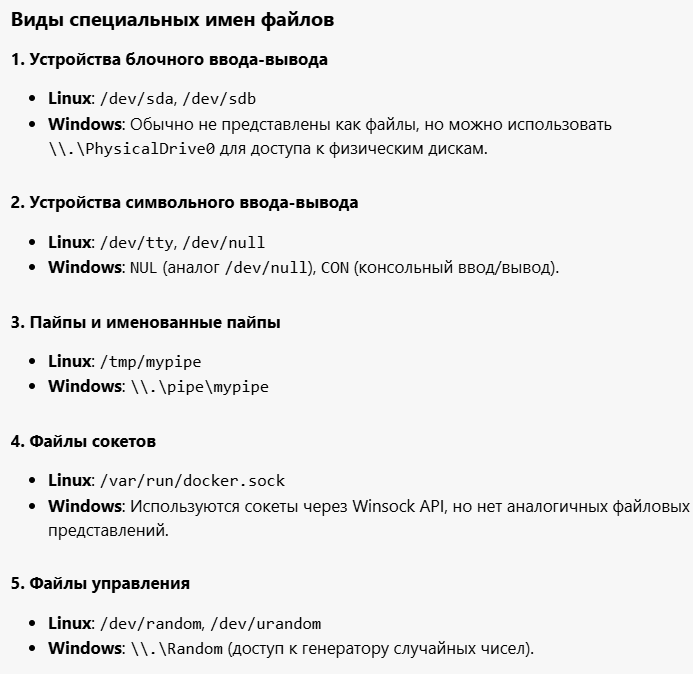
1. Поясните понятие «текущий каталог приложения».

**Текущий каталог приложения** — это каталог, из которого приложение запускается и в котором оно может искать файлы по относительным путям.

1. Что такое специальные имена файлов? перечислите их, для чего они нужны.

**Специальные имена** — это зарезервированные идентификаторы, которые имеют особое значение в операционной системе.

Они используются для обозначения устройств и потоков ввода/вывода



1. Для чего используются буферы ввода-вывода?

Области памяти для хранения физически считанных данных; необходимы для устранения несоответствия между физическими и логическими чтением/записью.

1. Поясните понятие «кэширование».

Перемещение в быстродействующую память наиболее часто используемых данных

1. Поясните понятие «указатель позиции файла».

**Указатель файла** – объект файловой системы, позиционирующий логическую запись.

1. Поясните понятие «маркер конца файла».

**Маркер конца файла** — это специальный символ, который указывает на конец данных в файле. Он используется для определения, когда следует остановить чтение

1. Поясните понятие «блокировка файла».

**Блокировка** **файла** — это механизм, позволяющий предотвратить доступ к файлу нескольким процессам одновременно, тем самым избегая конфликтов при чтении или записи

1. Windows. Функция OS API для создания файла.

createFile

1. Windows. Функция OS API для открытия файла.

createFile

1. Windows. Функция OS API для удаления файла.

deleteFile

1. Windows. Функция OS API для записи в файл.

writeFile

1. Windows. Функция OS API для чтения файла.

readFile

1. Windows. Назначение и отличие функций OS API: CopyFile, MoveFile, ReplaseFile.
   * **CopyFile**: Копирует файл из одного места в другое.
   * **MoveFile**: Перемещает файл из одного места в другое (удаляет оригинал).
   * **ReplaceFile**: Заменяет файл другим файлом, при этом сохраняет информацию о безопасности
2. Windows. Перечислите функции OS API, которые изменяют текущее значение указателя позиции файла.

SetFilePointer , SetFilePointerEx

1. Windows. Перечислите функции OS API для блокировки и разблокировки файлов.

**LockFile**: Блокирует файл, **UnlockFile**: Разблокирует файл

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

Механизм позволяет отслеживать изменения в каталоге. Для его реализации используются функции:

* FindFirstChangeNotification
* FindNextChangeNotification
* FindCloseChangeNotification

1. 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.
2. Linux. Что такое FHS?

**FHS** – стандарт иерархии файловой системы

1. Linux. Перечислите типы файловых систем.

Ext2, ext3, ext4, BtrFS, XFS, JFS

1. Linux. Что такое inode?

**inode** – структура данных, которая содержит информацию о файле или каталоге на файловой системе

1. Linux. Поясните назначение функций open, read, write, close, ioctl, stat, flush**,** lseek, lstat, fstat.
   * **open**: Открывает файл и возвращает дескриптор.
   * **read**: Читает данные из файла.
   * **write**: Записывает данные в файл.
   * **close**: Закрывает файл, освобождая ресурсы.
   * **ioctl**: Выполняет операции ввода-вывода, специфичные для устройства.
   * **stat**: Получает информацию о файле.
   * **flush**: Сбрасывает кэшированные данные на диск.
   * **lseek**: Устанавливает позицию в файле.
   * **lstat**: Получает информацию о файле, включая символические ссылки.
   * **fstat**: Получает информацию о файле по дескриптору.