

Глава 4. Файлы и каталоги

Файл	2
Файл.....	2
Создать	3
Открыть – открытие файла	4
Закреть – закрытие файла.....	6
Кодировка – для файла	7
Позиция	7
УстановитьПозицию	8
Обрезать	9
ЭтоКонец.....	10
ЗаписатьСтроку	10
ПрочитатьСтроку	11
ОчиститьФайл	12
РазмерФайла	13
ЗаписатьДанные.....	14
ПрочитатьДанные	15
ПрочитатьXML	15
СкопироватьВФайл	16
ЗаголовкиНТТР.....	17
Добавить-заголовкиНТТР	19
Прочитать-заголовкиНТТР.....	19
УстановитьCookie-заголовкиНТТР.....	20
ПрочитатьCookie-заголовкиНТТР.....	21
УдалитьCookie-заголовкиНТТР.....	21
СкопироватьИзИнтернет.....	22
СкопироватьВПоле	23
СкопироватьИзПоля	24
ПоказатьОтчет – для файла	24
СформироватьОтчет	25
ВывестиНаПринтер.....	27
СброситьДанные	28
DBFТаблица.....	28
ЗакодироватьBASE64	30
РаскодироватьBASE64	32
Имена файлов и каталогов	33
НайтиФайл	33
НайтиКаталог	34
СписокФайлов	35
СписокКаталогов	36
ПереименоватьФайл.....	37
СкопироватьФайл	38
УдалитьФайл	39
МожноОткрытьФайл	40
ПросмотрФайла	41
РедактироватьТекстовыйФайл.....	42
ИмяКаталога.....	43
КаталогВременныхФайлов.....	44
КаталогМодулей	45
ЕстьМодуль	45



ИмяФайла.....	46
ИмяИсполняемогоФайла.....	47
УстановитьИмяФайла	48
РасширениеФайла	49
КороткийПуть	50
СоздатьКаталог	50

Файл

Файл

Создаёт новый файл с заданным именем и указанным режимом доступа к нему.

Синтаксис

`файл ([ИмяФайла [, Режим]])`

Параметры

ИмяФайла (текст) — имя существующего файла, к которому открывается доступ. Если параметр не указан, то создается временный файл. Необходимо указывать полный путь к каталогу, в котором нужно создать файл, иначе функция создаст новый файл в каталоге программы СБиС++.

Режим (текст) — режимы доступа к файлу, которые могут представлять комбинации следующих букв:

Режимы доступа	Описание
Ч	чтение
З	запись
С	создание файла, если его нет
О	очищение файла, если его нет, то создание
Р	разделяемый режим (совместный доступ к файлу)

Если режим доступа не указан, то создаётся файл с доступом на чтение и запись.

Возвращает

Объект - новый файл. Если параметр не указывается, то создается временный файл.

Примеры

```
файл("test.txt"); # создается файл «test.txt».
```

См. также:

- [Создать](#)
- [Открыть – открытие файла](#)
- [Закреть – закрытие файла](#)
- [Кодировка – для файла](#)

Создать

Создает файл в указанном каталоге.

Синтаксис

```
файл.Создать (Имяфайла)
```

Параметры

ИмяФайла (текст) – имя файла с расширением. Необходимо указывать полный путь к каталогу, в котором необходимо создать указанный файл, иначе функция создаст файл в каталоге программы СБИС++. В отличие от функции «Файл», если указанный файл уже существует при выполнении функции «Создать» будет выдано соответствующее сообщение и предложение перезаписать этот файл.

Возвращает

Да (логическое значение) – если файл создан.

Нет (логическое значение) – если файл не создан.

Примеры

Необходимо создать два файла и записать в них информацию, используя один временный файл:

```
перем оФ = файл ( );  
оФ.создать ( "с:\\1.txt" );  
оФ.Записатьстроку ( "первый файл" );  
оФ.создать ( "с:\\2.txt" );  
оФ.Записатьстроку ( "второй файл" );  
оФ.Закреть ( );
```

См. также:

- [Файл](#)

Открыть – открытие файла

Открывает доступ к указанному файлу.

Синтаксис

```
файл.Открыть (Имяфайла [ , Режим ] )
```

Параметры

ИмяФайла (текст) – имя файла с расширением, к которому необходимо открыть доступ. Необходимо указывать полный путь доступа к файлу, иначе функция будет искать файл в каталоге программы СБИС++.

Режим (текст) — режимы доступа к файлу, которые могут представлять комбинации следующих букв:

Режимы доступа	Описание
Ч	чтение
З	запись
С	создание файла, если его нет
О	очищение файла, если его нет, то создание
Р	разделяемый режим (совместный доступ к файлу)

Если режим доступа не указан, то функция открывает файл с доступом на чтение и запись.

Возвращает

Да (логическое значение) – если доступ к файлу открыт.

Нет (логическое значение) – если невозможно открыть доступ к файлу (например, если файл не существует).

Примеры

Откроем доступ к файлу и прочитаем часть строки из файла:

```
офайл = Файл ();  
Если (офайл.Открыть ("пример.txt", "ч") )  
{  
    офайл.ПрочитатьСтроку (пИмя) ;  
    офайл.Закреть () ;  
}  
Иначе  
    Сообщить ("Невозможно открыть файл") ;
```

См. также:

- [Файл](#)
- [Закреть – закрытие файла](#)



- [Создать](#)

Закреть – закрытие файла

Закрывает доступ к открытому файлу.

Синтаксис

`Файл.Закреть ()`

Параметры

Не указываются.

Комментарии

Функция используется, в основном, при совместном доступе к файлу. Если требуется, например, просмотреть файл, который в это время открыт и выполняются какие-то действия с этим файлом (например, запись данных), то ничего не выйдет, пока мы не закроем этот файл. При этом все внесенные изменения в файле будут сохранены.

Возвращает

Да (логическое значение) – если доступ к файлу закрыт.

Нет (логическое значение) – если невозможно закрыть доступ к файлу.

Примеры

Допустим, мы открыли какой-то файл и записываем в него новые данные. При этом, если мы захотим просмотреть этот файл, программа нам его не покажет:

```
оФайл = Файл ( ) ;  
пНазв = "пример.txt" ;  
оФайл.Открыть (пНазв , "з") ;  
оФайл.ЗаписатьСтроку ("текст , который будет записан в  
файл" ) ;  
оФайл.ПросмотрФайла (пНазв) ; # окно просмотра файла  
не откроется .
```

Но если мы перед просмотром закроем этот файл,

```
...  
оФайл.Закреть ( ) ;
```



`оФайл . ПросмотрФайла (пНазв) ;`
то мы сможем просмотреть содержимое файла.

См. также:

- [Файл](#)
- [Открыть – открытие файла](#)
- [Создать](#)

Кодировка – для файла

Устанавливает режимы кодировки текста в файле.

Синтаксис

`Файл . Кодировка ("dos" | "win")`

Параметры

В зависимости от желаемой кодировки «dos» или «win».

Возвращает

Да (логическое значение) – если кодировка установлена;

Нет (логическое значение) – если кодировка не установлена.

Пример

```
оФайл = Файл("test.txt");  
оФайл . Кодировка("dos"); # установлена dos -  
кодировка .
```

См. также:

- [Файл](#)

Позиция

Возвращает текущую позицию курсора в файле.

Синтаксис

`Файл . Позиция ()`



Параметры

Не указываются.

Возвращает

Число – текущая позиция.

Примеры

Необходимо узнать позицию курсора до и после записи строки в указанный файл:

```
офайл = файл («Proba2.txt»);  
Сообщить (офайл.Позиция ()); # при открытии файла  
позиция курсора 0.  
офайл.ЗаписатьСтроку (Win2Dos («Записать строку в  
файл»));  
Сообщить (офайл.Позиция ()); # после записи строки в  
файл позиция курсора 24.
```

См. также:

- [УстановитьПозицию](#)
- [Файл](#)
- [ЭтоКонец](#)

УстановитьПозицию

Устанавливает в файле курсор на указанную позицию.

Синтаксис

```
файл . УстановитьПозицию (НомерПоз)
```

Параметры

НомерПоз (число) – номер позиции, в которую нужно установить курсор в файле.

Возвращает

Число – номер текущей позиции курсора в файле.

Примеры

Необходимо прочитать информацию из указанного файла, начиная с 20 позиции курсора:

```
офайл = файл("history.txt");
офайл.УстановитьПозицию(20);
пока (!офайл.ЭтоКонец())
{
    офайл.ПрочитатьСтроку(пЧастьСтроки);
    Сообщить(пЧастьСтроки);
}
офайл.Закреть();
```

См. также:

- [Позиция](#)
- [Файл](#)
- [Обрезать](#)

Обрезать

Удаляет данные из файла, начиная с указанной позиции и до конца файла.

Синтаксис

```
файл.Обрезать()
```

Параметры

Не указываются.

Возвращает

Число – номер позиции курсора в файле, начиная с которого будет обрестись информация в файле.

Примеры

Необходимо удалить из файла информацию, начиная с 300 – ой позиции курсора:

```
офайл = файл("Proba2.txt");
офайл.УстановитьПозицию(300);
офайл.Обрезать();
офайл.Закреть();
```

**См. также:**

- [Файл](#)
- [УстановитьПозицию](#)

ЭтоКонец

Определяет по текущей позиции курсора достигнут ли конец файла.

Синтаксис

```
файл . ЭтоКонец ( )
```

Параметры

Не указываются.

Возвращает

Да (логическое значение) — если конец файла достигнут.

Нет (логическое значение) — если ещё не достигнут конец файла.

Примеры

Перепишем информацию построчно из одного файла в другой:

```
офайл1 = файл ("Proba2.txt");  
офайл2 = файл ("пример.txt");  
офайл1 . УстановитьПозицию (0);  
Пока (!офайл1 . ЭтоКонец ( )  
{  
    Перем пСтрока;  
    офайл1 . ПрочитатьСтроку (пСтрока);  
    офайл2 . ЗаписатьСтроку (пСтрока);  
}
```

См. также:

- [Файл](#)
- [Позиция](#)

ЗаписатьСтроку

Записывает указанную строку в файл.

Синтаксис

`файл.ЗаписатьСтроку (Строка)`

Параметры

Строка (текст) - текст, который будет записан в файл.

Возвращает

Ничего не возвращает.

Примеры

Необходимо в файл записать строку, содержащую путь к каталогу программы СБиС++, в котором находится файл «sbis.exe»:

```
офайл = файл ("Proba2.txt");  
офайл.ЗаписатьСтроку ("СБиС=" +  
ИмяКаталога (ИмяИсполняемогоФайла ( ) ) );  
офайл.Закрыть ( );
```

Требуется дозаписать в файл строку:

```
файл ("c:\\log.txt", "С").ЗаписатьСтроку ( ТекВремя+  
"+"Событие" );
```

См. также:

- [Файл](#)
- [УстановитьПозицию](#)
- [ПрочитатьСтроку](#)
- [ОчиститьФайл](#)

ПрочитатьСтроку

Считывает и записывает в указанную переменную часть прочитанной строки, начиная с текущей позиции.

Синтаксис

`файл.ПрочитатьСтроку (ИмяПеременной)`

Параметры



ИмяПеременной (текст) - переменная, в которую будет записана часть строки, не превышающая 255 символов.

Возвращает

Да (логическое значение) – если строка прочитана до конца строки или конца файла.

Нет (логическое значение) – если строка не прочитана до конца. В этом случае, в переменную считываются 255 символов строки.

Примеры

В указанном файле создадим первую строку, содержащую больше 255 символов. В этом случае, при первом выполнении цикла функция возвратит «Нет» и в переменную `пЧастьСтроки` будут записаны первые 255 символов:

```
офайл = файл ("Proba2.txt");
перем пЧастьСтроки;
перем пПолностьюПрочитали;
пока (!офайл.ЭтоКонец())
{
    пПолностьюПрочитали =
офайл.ПрочитатьСтроку(пЧастьСтроки);
    сообщить(пЧастьСтроки, пПолностьюПрочитали);
}
офайл.Закрыть.
```

См. также:

- [Файл](#)
- [УстановитьПозицию](#)
- [ЗаписатьСтроку](#)
- [ОчиститьФайл](#)

ОчиститьФайл

Удаляет содержимое файла.

Синтаксис



`файл.ОчиститьФайл()`

Параметры

Не указываются.

Возвращает

Ничего не возвращает.

Примеры

Необходимо из файла удалить всю имеющуюся в нем информацию:

```
офайл = файл("Proba2.txt");  
офайл.ОчиститьФайл();  
офайл.Закрыть();
```

См. также:

- [Файл](#)
- [Открыть – открытие файла](#)
- [Закрыть – закрытие файла](#)

РазмерФайла

Возвращает размер указанного файла.

Синтаксис

`файл.РазмерФайла()`

Параметры

Не указываются.

Возвращает

Число - размер указанного файла в байтах.

Примеры

Необходимо узнать размер указанного файла

```
офайл = файл("history.txt");  
Сообщить(офайл.РазмерФайла());  
офайл.Закрыть();
```

**См. также:**

- [Файл](#)
- [Открыть – открытие файла](#)
- [Закреть – закрытие файла](#)

ЗаписатьДанные

Записывает данные любого типа в файл.

Синтаксис

`файл.ЗаписатьДанные (Значение)`

Параметры

Значение – данные любого типа, которые необходимо добавить в файл.

Возвращает

Число – количество записанной информации в байтах.

Примеры

Создадим правило операции для документа, при вызове которого (например, при закрытии документа) в указанный файл будут записаны значения параметров, указанные для этого документа:

```
функция НаДокументЗакреть ()
{
    переменная файл = Файл ("Параметры.txt");
    файл.ЗаписатьДанные (Документ.Параметры);
    файл.Закреть ();
}
```

См. также:

- [Файл](#)
- [Открыть – открытие файла](#)
- [Закреть – закрытие файла](#)
- [ПрочитатьДанные](#)

ПрочитатьДанные

Считывает и записывает в указанную переменную данные из файла.

Синтаксис

`Файл.ПрочитатьДанные (ИмяПеременной [, Кол-во])`

Параметры

ИмяПеременной (переменная) - переменная, в которую будут записаны данные из файла.

Кол-во (число) – количество считываемой информации из файла в байтах. Если параметр не указан, считывает всю информацию из файла. Если указано количество, превышающее размер файла, будет выдано сообщение о попытке чтения за пределами файла/области данных.

Возвращает

Число - количество считываемой информации в байтах.

Примеры

Необходимо считать 925 байт информации из указанного файла:

```
оФайл = Файл("history.txt");  
перем пСтрока;  
оФайл.ПрочитатьДанные(пСтрока, 925);  
Сообщить(пСтрока);  
оФайл.Закрыть();
```

См. также:

- [Файл](#)
- [Открыть – открытие файла](#)
- [Закрыть – закрытие файла](#)
- [ЗаписатьДанные](#)

ПрочитатьXML

Возвращает структурированный объект по XML-файлу, который содержит название, атрибуты и теги.



Синтаксис

`файл . ПрочитатьXML ()`

Параметры

Не указывается.

Комментарии

Функция широко используется в ЭО.

Возвращает

Объект, содержащий переменные: «Атрибуты» (типа «Объект»), «Название» (типа «Текст»), «Теги» (типа «Массив»). Каждый элемент массива представляет собой объект, который в свою очередь также раскладывается на составляющие: атрибут, название и теги, и т.д.

Примеры

Необходимо разобрать на составляющие следующий XML-файл:

```
оФайл=файл ( ) ;  
оФайл . Открыть ( "NO_EDUPR_4623_7605015260760501001_200  
80415_V7AD196A-F0E2-47D3-BEAD-1DEB9815E757.xml" ) ;  
оСоставляющие=оФайл . ПрочитатьXML ( ) ;  
отладить ( оСоставляющие ) ;
```

См. также:

- [Файл](#)
- [Открыть – открытие файла](#)
- [Закрыть – закрытие файла](#)
- [ЗаписатьДанные](#)

СкопироватьВФайл

Копирует данные из одного файла в другой.

Синтаксис

`файл . СкопироватьВФайл (НовФайл)`



Параметры

НовФайл (объект) – файл, в который будут скопированы данные из исходного файла.

Возвращает

Число - количество скопированной информации в байтах.

Примеры

Необходимо скопировать данные из файла «history.txt» в файл «Proba.txt»:

```
офайл = файл("history.txt");  
офайл1 = файл("Proba.txt");  
офайл.СкопироватьВФайл(офайл1);
```

См. также:

- [Файл](#)
- [Открыть – открытие файла](#)
- [Закрыть – закрытие файла](#)
- [ЗаписатьДанные](#)

ЗаголовкиHTTP

Возвращает объект для передачи дополнительных параметров Post-запроса.

Синтаксис

ЗаголовкиHTTP ()

Параметры

Без параметров.

Комментарии

Функция тесно связана с функцией **СкопироватьИзИнтернет**: стало возможным не только копировать страницы, но и отсылать данные на сервер (заполненные формы, файлы и т.п.).

Возвращает



Объект, с помощью которого можно передавать дополнительные параметры POST-запроса.

Примеры

```
оЗапрос = файл( "C:\\temp\\post.txt" );
оРезультат = файл( "C:\\temp\\1.html" );
оЗаголовки = ЗаголовкиHTTP();
оЗаголовки.Добавить( "Content-Type", "application/x-
www-form-urlencoded" );
оРезультат.СкопироватьИзИнтернет(
"http://some.server.ru/login.php", оЗаголовки,
оЗапрос);
оЗапрос = Нет;
оРезультат = Нет;
Сообщить(оЗаголовки.Прочитать("Content-Type"));
Сообщить(оЗаголовки.Прочитать("X-Powered-By"));
Сообщить(оЗаголовки.Прочитать("X-Powered-By",0));
Сообщить(оЗаголовки.Прочитать("X-Powered-By",1));
Сообщить(оЗаголовки.Прочитать("X-Powered-By",2));
оЗаголовки.УстановитьCookie("test_name1",
"test_value1");
оЗаголовки.УстановитьCookie("test_name2",
"test_value2");
Сообщить(оЗаголовки.ПрочитатьCookie("test_name1"));
оЗаголовки.УдалитьCookie("test_name1");
Сообщить(оЗаголовки.ПрочитатьCookie("test_name1"));
Сообщить(оЗаголовки.ПрочитатьCookie("test_name2"));
оЗаголовки.УдалитьCookie();
Сообщить(оЗаголовки.ПрочитатьCookie("test_name2"));
оЗаголовки = Нет;
```

См. также:

- [Файл](#)
- [СкопироватьИзИнтернет](#)
- [Добавить](#)
- [Прочитать](#)
- [УстановитьCookie](#)
- [ПрочитатьCookie](#)

- [УдалитьCookie](#)

Добавить-заголовкиНТТР

Добавляет в объект «ЗаголовкиНТТР» указанный НТТР-заголовок и значение.

Синтаксис

ЗаголовкиНТТР.Добавить (Название , Значение)

Параметры

Название (текст) – название НТТР-заголовка.

Значение (текст) – значение НТТР-заголовка.

Возвращает

Да (логическое значение) – если заголовок добавлен.

Нет (логическое значение) – если не удалось добавить заголовок.

Примеры

Подробный пример использования метода приводится у функции [ЗаголовкиНТТР](#).

См. также:

- [ЗаголовкиНТТР](#)

Прочитать-заголовкиНТТР

Возвращает значение НТТР-заголовка с указанным названием и индексом.

Синтаксис

ЗаголовкиНТТР.Прочитать (Название [, Индекс])

Параметры

Название (текст) – название НТТР-заголовка.

Индекс (целое) – значение НТТР-заголовка. Если параметр не указан, то функция возвратит значение первого НТТР-заголовка.



Возвращает

Любой – значение HTTP-заголовка. Если такого заголовка нет, то возвращает значение «Нет».

Примеры

Подробный пример использования метода приводится у функции `ЗаголовкиHTTP`.

См. также:

- [ЗаголовкиHTTP](#)

УстановитьCookie-заголовкиHTTP

Добавляет в объект «ЗаголовкиHTTP» Cookie (тоже HTTP-заголовок) с заданным именем и значением.

Синтаксис

`ЗаголовкиHTTP.УстановитьCookie(Название, Значение)`

Параметры

Название (текст) – название Cookie.

Значение (текст) – значение Cookie.

Возвращает

Да (логическое значение) – если Cookie добавлен.

Нет (логическое значение) – если не удалось добавить Cookie.

Примеры

Подробный пример использования метода приводится у функции `ЗаголовкиHTTP`.

См. также:

- [ЗаголовкиHTTP](#)

ПрочитатьCookie-заголовкиHTTP

Возвращает значение Cookie с указанным названием.

Синтаксис

ЗаголовокHTTP . Прочитать (Название)

Параметры

Название (текст) – название Cookie.

Возвращает

Любой – значение Cookie. Если такого нет, то возвращает значение «Нет».

Примеры

Подробный пример использования метода приводится у функции ЗаголовкиHTTP.

См. также:

- [ЗаголовкиHTTP](#)

УдалитьCookie-заголовкиHTTP

Удаляет Cookie с указанным названием.

Синтаксис

ЗаголовокHTTP . УдалитьCookie ([Название])

Параметры

Название (текст) – название Cookie. Если параметр не указан, то функция удаляет все Cookie.

Возвращает

Да (логическое значение) – если удалось удалить Cookie.

Нет (логическое значение) – если не удалось удалить Cookie.

Примеры

Подробный пример использования метода приводится у функции `ЗаголовкиНТТР`.

См. также:

- [ЗаголовкиНТТР](#)

СкопироватьИзИнтернет

Копирует данные интернет-страницы в файл.

Синтаксис

`файл.СкопироватьИзИнтернет (Адрес [, НТТР , файл])`

Параметры

Адрес (текст) – адрес интернет-страницы, данные которой будут скопированы в файл.

НТТР (объект) – объект, созданный функцией «`ЗаголовкиНТТР`». Используется для формирования НТТР-заголовков при передаче запроса и загрузки в него же заголовков ответа. Параметр используется только совместно с параметром «Файл».

Файл (объект) – файл содержит непосредственно **Post-запрос**. Параметр используется только совместно с параметром «НТТР».

Комментарии

Если при вызове функции указаны параметры **НТТР** и **Файл**, то выполняем **Post-запрос** (для передачи на сервер большого объема данных, более 256 символов). Обязательное условие – сервер должен поддерживать такой запрос. Если не указаны параметры, то выполняем **Get-запрос** (передаем только ссылку на сайт в обозревателе).

Возвращает

Число - количество скопированной информации в байтах.

Примеры

Необходимо скопировать в указанный файл данные интернет – страницы:

```
оФайл = Файл ("Сайт.html" );  
оФайл.СкопироватьИзИнтернет ("http://inside.tensor.ru  
/");  
оФайл.Закреть ();
```

См. также:

- [Файл](#)
- [ЗаголовкиHTTP](#)
- [СкопироватьВФайл](#)
- [ЗаписатьДанные](#)

СкопироватьВПоле

Копирует двоичные данные из файла в указанное поле записи из базы данных.

Синтаксис

```
Файл.СкопироватьВПоле (ИмяПоля)
```

Параметры

ИмяПоля (текст) – имя двоичного поля записи, в которое будут скопированы данные из файла.

Возвращает

Число - количество скопированной информации в байтах.

Примеры

Необходимо скопировать информацию из файла «history.txt» в поле «Данные» записи из таблицы форм отчетности:

```
оФайл = Файл ("history.txt" );  
оФормы = Таблица ("Формы отчетности" );  
оФормы.Добавить ();  
оФайл.СкопироватьВПоле (оФормы.Данные) ;  
оФайл.Закреть ();
```

См. также:



- [Файл](#)
- [ПрочитатьДанные](#)
- [СкопироватьИзПоля](#)

СкопироватьИзПоля

Копирует двоичные данные из поля записи из базы данных в указанный файл.

Синтаксис

`файл . СкопироватьИзПоля (ИмяПоля)`

Параметры

ИмяПоля (текст) – имя двоичного поля, из которого будут скопированы данные в файл.

Возвращает

Число - количество скопированной информации в байтах.

Примеры

Необходимо скопировать данные из двоичного поля записи из таблицы «Внешние документы» во временный файл.

```
офайл = файл ( ) ;  
оформы = Таблица ("Внешние документы") ;  
Следующий (оформы)  
офайл . СкопироватьИзПоля (оформы . Документ) ;  
офайл . Закрыть ( ) ;
```

См. также:

- [Файл](#)
- [ПрочитатьДанные](#)
- [СкопироватьВПоле](#)

ПоказатьОтчет – для файла



Показывает текущее содержимое файла, содержащего отчет.

Синтаксис

`файл.ПоказатьОтчет ([КонтОбъект [, Режим]])`

Параметры

КонтОбъект (объект) - контекстный объект. Если контекстного объекта нет, то в качестве параметра необходимо указать «Нет».

Режим (текст) — режимы доступа к отчёту, которые могут представлять комбинации следующих букв:

Режимы доступа	Описание
М	модальный
Р	запускается окно предварительного просмотра перед выводом на принтер

Если режим доступа не указан, то отчёт показывается в обычном режиме.

Возвращает

Да (логическое значение) – если отчет показывается.

Нет (логическое значение) – если отчет не показывается.

Примеры

Покажем отчет в предварительном просмотре:

```
офайл = файл ("ОтчетФрм N37 от 31-12-04.html");  
офайл.ПоказатьОтчет (нет, "Р");  
офайл.Закрыть ();
```

См. также:

- [СформироватьОтчет](#)

СформироватьОтчет

Записывает сформированный отчет в указанный файл.



Синтаксис

Файл.СформироватьОтчет (ИмяОтчета [, КонтОбъект [, Режим]])

Параметры

ИмяОтчета (текст) – имя отчета, предназначенный для записи в файл.

КонтОбъект (объект) - контекстный объект. Если контекстного объекта нет, то в качестве параметра необходимо указать «Нет».

Режим (текст) — режимы формирования отчёта, которые могут представлять комбинации следующих букв:

Режимы доступа	Описание
П	при формировании отчёта сохраняются только те теги, которые видны на принтере (используется в комбинации с режимом «в»)
В	при формировании отчёта сохраняются только видимые теги
Т	при формировании отчёта сохраняется таблица стилей
С	при формировании отчёта сохраняются скрипты
О	при формировании отчёта сохраняется только содержимое тега <body> (сам тег при этом не сохраняется)
А	возможность сохранения отчёта в формате «.mhtml»

Если режим доступа не указан, то отчёт показывается в обычном режиме.

Комментарий

Функция используется, например, при формировании отчетов в электронной отчетности. При этом, сформированный отчет должен находиться в каталоге модулей электронной отчетности.

Возвращает

Да (логическое значение) – если отчет записан в файл.

Нет (логическое значение) – если не записан в файл.

Примеры

Запишем отчет "ОтчетФрм N37 от 31-12-04" в файл "Отчет.html":

```
офайл = файл ("Отчет.html") ;
офайл.СформироватьОтчет ("ОтчетФрм N37 от 31-12-04") ;
офайл.Закреть () ;
```

См. также:

- [ПоказатьОтчет – для файла](#)

ВывестиНаПринтер

Выводит содержимое файла на указанный принтер.

Синтаксис

```
файл.ВывестиНаПринтер (ИмяПринтера [ , ТипДанных ] )
```

Параметры

ИмяПринтера (текст) — имя принтера, на который выводится содержимое файла.

ТипДанных (текст) – тип данных. Для каждого принтера существует свой набор типов данных. Например, если указать тип данных «ТЕХТ», то содержимое файла выведется на печать в текстовом режиме. По - умолчанию, указывается тип "RAW".

Комментарии

Функция предназначена для прямой печати на принтер и применяется, например, для печати на чековом принтере.

Возвращает

Число – размер напечатанной информации в байтах.

0 – если не удалось распечатать данные файла.

Примеры

Необходимо вывести содержимое файла на указанный принтер в текстовом режиме:

```
офайл = файл("whats_new.txt");  
офайл.ВывестиНаПринтер("CanonLBP", "ТЕХТ");  
офайл.Закреть();
```

См. также:

- [ПоказатьОтчет – для файла](#)
- [СформироватьОтчет](#)

СброситьДанные

Сбрасывает текущее содержимое файла на жесткий диск. При этом доступ к файлу остаётся открытым.

Синтаксис

```
файл.СброситьДанные()
```

Параметры

Без параметров.

Возвращает

Да (логическое значение) – данные сброшены на диск.

Нет (логическое значение) – данные не были сброшены на диск.

DBFТаблица

Создаёт файл с расширением «.dbf» заданной структуры.

Синтаксис

DBFТаблица (ИмяФайла , Структура [, Кодировка])

Параметры

ИмяФайла (текст) – имя файла с расширением «.dbf», который будет создан. Если указанный файл уже существует, то он будет переписан без дополнительного предупреждения. Если файл помещается в несуществующий каталог, то он будет автоматически создан программой СБиС++.

Структура (текст или объект) – описание структуры. В качестве структуры можно передать объект, например, выборку, таблицу и т.п., или название формата записи, который описан в файле ресурсов программы. Пользователь может воспользоваться уже готовым форматом (с уже готовым набором полей), или создать свой новый элемент в ресурсе с типом «Формат записи». В этом формате необходимо задать свой набор полей, который функция определит для указанного dbf-файла. Например:

Поля записи	
Имя поля	Тип поля
Имя строки	Текст фиксированной длины
Название строки	Строка переменной длины
Новые данные	Двоичные данные

Необходимо учесть следующее:

- двоичные поля, флаги, перечисляемые, время - превращаются в текстовые поля;
- поля связи преобразовываются в числовые поля;
- даты остаются датами;
- длина имени поля должна быть не более 11 символов, включая пробелы (это ограничение dbf-формата).

Кодировка (текст) – кодировка, в которой будет сохранен созданный файл. Если данный параметр не указан, то по умолчанию файл сохраняет-



ся в формате Windows. В случае необходимости сохранения в формате Dos, нужно явно задать параметр кодировка равный «ДОС» или «DOS».

Возвращает

Объект, содержащий запись созданного файла. Просмотреть такую таблицу можно средствами программы (<Ctrl+O>) или воспользоваться утилитой «Джинн».

Примеры

Необходимо создать файл «table.dbf» в Dos-кодировке с такими же полями, как у выборки «Входящие платежи»:

```
oТабл = Выборка ("Входящие платежи") ;
oDBFТаблица=DBFТаблица ("C:\\Таблицы\\table.dbf" ,
oТабл, "ДОС") ;
```

Необходимо создать файл «table.dbf» в Dos-кодировке с такими же полями, как у выборки «Входящие платежи»:

```
oТабл = Выборка ("Входящие платежи") ;
oDBFТаблица=DBFТаблица ("C:\\Таблицы\\table.dbf" ,
oТабл, "ДОС") ;
```

Необходимо создать файл «table.dbf» с теми полями, которые указаны в формате «Новый формат записи» (созданный пользователем):

```
oDBFТаблица=DBFТаблица ("C:\\Таблицы\\table.dbf" , "Новый формат записи") ;
```

ЗакодироватьBASE64

Кодирует содержимое файла с помощью схемы BASE64.

Синтаксис

ЗакодироватьBASE64 (файл)

Параметры

Файл (объект) – объект типа «файл» с данными для кодирования, который должен быть предварительно открыт.

Возвращает

Да (логическое значение) – операция кодирования прошла успешно.

Нет (логическое значение) - операция закончилась неудачно.

Комментарии

В формате электронной почты MIME, BASE64 это схема, по которой произвольная последовательность байт преобразуется в последовательность печатных ASCII символов. Это определяет MIME как транспортное кодирование содержимого для использования в электронной почте. Используются только символы латинского алфавита в верхнем и нижнем регистре — символы (A—Z, a—z), цифры (0—9), и символы «+» and «/», с символом «=» в качестве специального кода суффикса.

Полная спецификация этой формы BASE64 содержится в RFC 1421 и RFC 2045. Эта схема используется для кодирования последовательности октетов (байт). Это соответствует определению файлов почты во всех системах. Результирующие закодированные по BASE64 данные имеют длину, большую оригинальной в соотношении 4:3, и напоминают по виду случайные символы.

Для того, чтобы преобразовать данные в BASE64, первый байт помещается в самые старшие восемь бит 24-битного буфера, следующие в средние восемь и третий в младшие значащие восемь бит. Если кодируется менее чем три байта, то соответствующие биты буфера устанавливаются в ноль. Далее каждые шесть бит буфера, начиная с самых старших, используются как индексы строки «ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/>» и её символы, на которые указывают индексы, помещаются в выходную строку. Если кодируются только один или два байта, используются только первые два или три символа строки и выходная строка дополняется двумя или одним символами «=». Это предотвращает добавление дополнительных битов к восстановленным данным. Процесс повторяется над оставшимися входными данными.

В основном сканнеры спама, которые не декодируют сообщения в BASE64, часто пропускают сообщения в BASE64, так как они кажутся достаточно случайными, или не содержат ключевые слова в тексте BASE64, чтобы не быть принятыми за спам.

Пример

```
оФайл = Файл ( ) ;
оФайл.ЗаписатьСтроку ( "Данные" ) ;
ЗакодироватьBASE64 ( оФайл ) ;
```



```
офайл.УстановитьПозицию( 0 );  
перем пСтрока ;  
офайл.ПрочитатьСтроку( пСтрока ) ;  
Сообщить ( пСтрока ) ; # Выдаст результат :  
"xODt7fv1DQo="
```

См. также:

- [РаскодироватьBASE64](#)

РаскодироватьBASE64

Декодирует содержимое файла с помощью схемы BASE64.

Синтаксис

РаскодироватьBASE64 (файл)

Параметры

Файл (объект) – объект типа «файл» с закодированными данными, который должен быть предварительно открыт.

Возвращает

Да (логическое значение) – операция кодирования прошла успешно.

Нет (логическое значение) - операция закончилась неудачно.

Комментарии

В формате электронной почты MIME, BASE64 это схема, по которой произвольная последовательность байт преобразуется в последовательность печатных ASCII символов. Это определяет MIME как транспортное кодирование содержимого для использования в электронной почте. Используются только символы латинского алфавита в верхнем и нижнем регистре — символы (A—Z, a—z), цифры (0—9), и символы «+» and «/», с символом «=» в качестве специального кода суффикса.

Полная спецификация этой формы BASE64 содержится в RFC 1421 и RFC 2045. Эта схема используется для кодирования последовательности октетов (байт). Это соответствует определению файлов почти во всех системах. Результирующие закодированные по BASE64 данные имеют длину,



большую оригинальной в соотношении 4:3, и напоминают по виду случайные символы.

Для того, чтобы преобразовать данные в BASE64, первый байт помещается в самые старшие восемь бит 24-битного буфера, следующие в средние восемь и третий в младшие значащие восемь бит. Если кодируется менее чем три байта, то соответствующие биты буфера устанавливаются в ноль. Далее каждые шесть бит буфера, начиная с самых старших, используются как

индексы	строки
---------	--------

 «ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/>» и её символы, на которые указывают индексы, помещаются в выходную строку. Если кодируются только один или два байта, используются только первые два или три символа строки и выходная строка дополняется двумя или одним символами «=». Это предотвращает добавление дополнительных битов к восстановленным данным. Процесс повторяется над оставшимися входными данными.

В основном сканнеры спама, которые не декодируют сообщения в BASE64 часто пропускают сообщения в BASE64, так как они кажутся достаточно случайными, или не содержат ключевые слова в тексте BASE64, чтобы не быть принятыми за спам.

Пример

```
оФайл = Файл ( ) ;
оФайл .ЗаписатьСтроку ( "xODt7fv1DQo=" ) ;
РаскодироватьBASE64 ( оФайл ) ;
оФайл .УстановитьПозицию ( 0 ) ;
перем пСтрока ;
оФайл .ПрочитатьСтроку ( пСтрока ) ;
Сообщить ( пСтрока ) ; # Сообщит результат : "Данные"
```

См. также:

- [ЗакодироватьBASE64](#)

Имена файлов и каталогов

НайтиФайл

Проверяет существование файла в указанном каталоге.

Синтаксис

НайтиФайл (ИмяФайла)

Параметры

ИмяФайла (текст) — имя файла, существование которого необходимо проверить в каталоге. Поиск файла будет производиться только в указанном каталоге, в подкаталогах функция файл не ищет. Например, если мы хотим найти файл «sbis.exe», мы должны написать НайтиФайл("C:\\СБис++\\sbis.exe"), а не НайтиФайл("C:\\sbis.exe"). Необходимо указывать полный путь доступа к файлу. Если указано только имя файла, то функция ищет указанный файл только в каталоге программы СБис++.

Возвращает

Да (логическое значение) — если указанный файл существует в каталоге программы СБис++.

Нет (логическое значение) — если указанный файл не существует в каталоге программы СБис++.

Примеры

Необходимо проверить существование в каталоге программы файла «history.txt» и узнать размер этого файла, в случае его нахождения. Иначе будет выдано соответствующее сообщение:

```
НазваниеФайла = "history.txt";
Если (НайтиФайл (НазваниеФайла) )
{
    офайл = файл (НазваниеФайла) ;
    Сообщить (офайл . РазмерФайла ( ) ) ;
    офайл . Закрыть ( ) ;
}
Иначе
    Сообщить ("Невозможно найти файл
\""+НазваниеФайла+"\"") ;
```

См. также:

- [НайтиКаталог](#)

НайтиКаталог



Проверяет существование указанного каталога.

Синтаксис

НайтиКаталог (*ИмяКаталога*)

Параметры

ИмяКаталога (текст) — имя каталога, который необходимо найти. Необходимо указывать полный путь к каталогу, в котором будет происходить поиск нужного каталога. Если указано только имя каталога, то будет функция будет искать каталог в каталоге программы СБиС++.

Возвращает

Да (логическое значение) — если указанный каталог найден.

Нет (логическое значение) — если указанный каталог не найден.

Примеры

Необходимо проверить существование данного каталога и если его не существует, то создать в указанном каталоге:

```
Если (!НайтиКаталог("С:\СБиС++\files"))  
СоздатьКаталог("С:\СБиС++\files")
```

См. также:

- [НайтиФайл](#)
- [СоздатьКаталог](#)

СписокФайлов

Ищет указанный файл в каталоге и всех его подкаталогах.

Синтаксис

СписокФайлов (*ИмяФайла*)

Параметры

ИмяФайла (текст) — имя файла с расширением, который необходимо найти в указанном каталоге. Необходимо указывать полный путь к каталогу, в котором будет производиться поиск нужного файла. Если указано



только имя файла с расширением, то функция будет искать файл в каталоге программы СБиС++.

Если в имени или в расширении файла указать символ «*» или «?», то поиск файлов будет производиться по некоторому шаблону: в первом случае – произвольное количество символов в имени или расширении файла; во втором – один символ в имени или расширении файла.

Возвращает

Объект - массив, содержащий все найденные пути к указанному файлу.

Примеры

Необходимо подсчитать количество HTML – отчетов(файлов с расширением «.html») в каталоге модулей с основными средствами:

```
перем мОтчеты = СписокФайлов(КаталогМодулей() +  
"!Основные средства\*.html");  
Сообщить(Размер(мОтчеты));
```

См. также:

- [СписокКаталогов](#)

СписокКаталогов

Ищет указанный каталог во всех каталогах и подкаталогах.

Синтаксис

СписокКаталогов (ИмяКаталога)

Параметры

ИмяКаталога (текст) — имя каталога, который необходимо найти. Если указано только имя, то поиск будет производиться в каталогах программы СБиС++. Поэтому необходимо указывать полный путь доступа к этому каталогу.

Если в качестве параметра функции указать символ «*» или «?», то поиск каталогов будет производиться по некоторому шаблону: в первом случае – произвольное количество символов в имени каталога; во втором – один символ в имени каталога.

Возвращает

Объект - массив, содержащий все найденные пути к указанному каталогу.

Примеры

Необходимо найти полный путь к программам СБиС++, установленным на компьютере пользователя. Поиск будет производится по каталогу «Модули», который обязательно присутствует в каталоге установленной программы СБиС++:

```
перемен мСписок = СписокКаталогов ("С:\Модули");  
перемен пНом = 0;  
Если (Размер (мСписок))  
    Пока (пНом++ < Размер (мСписок))  
        {  
            Если (Найти (мСписок [пНом], "СБиС"))  
                Сообщить (ИмяКаталога (мСписок [пНом])) ;  
        }  
Иначе  
    Сообщить ("Данного каталога нет") ;
```

См. также:

- [СписокФайлов](#)

ПереименоватьФайл

Изменяет имя указанного файла на новое.

Синтаксис

ПереименоватьФайл (**ИмяФайлаСтарое**, **ИмяФайлаНовое**)

Параметры

ИмяФайлаСтарое (текст) – старое имя файла с расширением, который необходимо переименовать. Необходимо указывать полный путь доступа к файлу, иначе функция будет искать файл в каталоге программы СБиС++.

ИмяФайлаНовое (текст) – новое имя файла с расширением. Необходимо указывать полный путь доступа к переименованному файлу, иначе функция после переименования переместит файл в каталог программы СБиС++.

Возвращает



Да (логическое значение) - если файл переименован.

Нет (логическое значение) - если файл переименовать не удалось. При этом процесс не прерывается, а продолжают выполняться следующие действия в программе.

Примеры

Необходимо переименовать файл с именем, содержащим пробелы, в файл с именем, содержащим вместо пробелов символ «_»:

```
перем пПолныйПутьКФайлу ;
пИмяФайла = ИмяФайла (пПолныйПутьКФайлу) ;
Если (Найти (пИмяФайла , " "))
    пЗамИмяФайла = Заменить (пИмяФайла , " " , "_" ) ;
ПереименоватьФайл (пПолныйПутьКФайлу ,
ИмяКаталога (пПолныйПутьКФайлу) + пЗамИмяФайла +
РасширениеФайла (пПолныйПутьКФайлу) ) ;
```

См. также:

- [ИмяФайла](#)
- [РасширениеФайла](#)
- [ИмяКаталога](#)

СкопироватьФайл

Копирует файл или группу файлов в указанный каталог.

Синтаксис

СкопироватьФайл (ИмяФайла , ИмяКаталога)

Параметры

ИмяФайла (текст) — имя файла, который необходимо скопировать в указанный каталог. Необходимо указывать полный путь к файлу, иначе программа будет искать файл в каталоге программы СБИС++. При указании имени файла и/или расширения допускается использование символа «*». При этом копируется не один, а серия файлов. Так, например, если мы хотим скопировать все файлы с расширением «.txt» в каталог «ТЕХТ», нужно написать так: СкопироватьФайл(“ *.txt“, “ТЕХТ”).



ИмяКаталога (текст) – имя каталога, в который копируется исходный файл. Данный каталог обязательно должен существовать. Необходимо указывать полный путь к каталогу, иначе программа будет искать указанный каталог в каталоге программы СБиС++.

Возвращает

Число – количество скопированных файлов.

0 – если файл скопировать не удалось. При этом процесс не прерывается, а продолжают выполняться следующие действия в программе.

Примеры

Необходимо скопировать все файлы с расширением «txt», лежащие в каталоге программы СБиС++, в каталог «ТЕХТ» на диске С. Можно воспользоваться следующим примером:

СкопироватьФайл ("*.txt", "C:\\ТЕХТ"); # в результате в указанный каталог будет скопировано 20 файлов.

См. также:

- [ИмяФайла](#)
- [ИмяКаталога](#)
- [УдалитьФайл](#)
- [ПросмотрФайла](#)

УдалитьФайл

Удаляет указанный файл или группу файлов.

Синтаксис

УдалитьФайл (ИмяФайла)

Параметры

ИмяФайла (текст) — имя файла, который необходимо удалить. Необходимо указывать полный путь доступа к файлу, иначе функция будет искать указанный файл для удаления в каталоге программы СБиС++. При указании имени файла и/или расширения допускается использование символа «*». При этом будет удален не один, а серия файлов. Так, например, если



мы хотим удалить все файлы с расширением «txt», нужно написать так:
УдалитьФайл(“ *.txt”).

Возвращает

0 – если указанный файл не удален (например, такой файл не найден). При этом процесс не прерывается, а продолжают выполняться следующие действия в программе.

Число – отличное от нуля, соответствующее количеству удаленных файлов.

Примеры

Необходимо из каталога «DB» программы СБиС++ удалить все временные файлы (файлы с расширением «.tmp»), которые создаются при проверке базы данных:

```
УдалитьФайл ("C : \\СБиС++ \\db \\* . tmp" ) .
```

См. также:

- [СкопироватьФайл](#)
- [ПереименоватьФайл](#)
- [ПросмотрФайла](#)

МожноОткрытьФайл

Проверяет возможность открытия доступа к файлу с указанным режимом доступа.

Синтаксис

```
МожноОткрытьФайл (ИмяФайла , Режим)
```

Параметры

ИмяФайла (текст) — имя файла с расширением, к которому нужно открыть доступ. Необходимо указывать полный путь доступа к файлу, иначе функция будет искать указанный файл в каталоге программы СБиС++.

Режим (текст) — режимы доступа к файлу, которые могут представлять комбинации следующих букв:



Режимы доступа	Описание
Ч	чтение
З	запись
С	создание файла, если его нет
О	очищение файла, если его нет, то создание
Р	разделяемый режим (совместный доступ к файлу)

Если режим доступа не указан, то создаётся файл с доступом на чтение и запись.

Возвращает

Да (логическое значение) — если к файлу можно открыть доступ.

Нет (логическое значение) — если к файлу нельзя открыть доступ (например, указан несуществующий файл).

Примеры

Необходимо запустить программу СБиС++, если можно открыть доступ к файлу «sbis.exe»:

```
Если (МожноОткрытьФайл ("D:\\_WORK\БАЗЫ\1Б\sbis.exe", "З"))
    Запустить ("D:\\_WORK\БАЗЫ\1Б\sbis.exe");
Иначе
    Ошибка ("На компьютере уже запущена программа СБиС.");
```

См. также:

- [Открыть – открытие файла](#)

ПросмотрФайла

Открывает файл в программе СБиС++ для просмотра данных.



Синтаксис

ПросмотрФайла (ИмяФайла)

Параметры

ИмяФайла (текст) – имя просматриваемого файла с расширением «.txt». Кроме текстовых файлов, возможен просмотр файлов с картинками, т.е. файлов с расширением «.gif», «.jpg» т.д. Необходимо указывать полный путь доступа к этому файлу. Если указано только имя файла с расширением, то функция ищет указанный файл в каталоге программы СБиС++.

Возвращает

Ничего не возвращает.

Примеры

Необходимо найти файл с рисунком и просмотреть его в программе:

```
Если (НайтиФайл ("C:\\Documents and Settings\\All  
Users.WINXP\\Документы\\Мои рисунки\\Образцы  
рисунков\\Водяные лилии.jpg"))  
    ПросмотрФайла ("C:\\Documents and Settings\\All  
Users.WINXP\\Документы\\Мои рисунки\\Образцы  
рисунков\\Водяные лилии.jpg")
```

См. также:

- [МожноОткрытьФайл](#)

РедактироватьТекстовыйФайл

Открывает текстовый файл в программе для редактирования.

Синтаксис

РедактироватьТекстовыйФайл (ИмяФайла)

Параметры

ИмяФайла (текст) – имя редактируемого файла, указанного с расширением. При указании имени файла, функция ищет этот файл в корневом каталоге программы и затем открывает его для редактирования. Если файл на-



ходится в любом другом каталоге, для дальнейшего его редактирования, вам необходимо указать полный путь доступа к этому файлу.

Возвращает

Ничего не возвращает.

Примеры

Необходимо внести изменения в текстовый файл «whats_new.txt»:

```
РедактироватьТекстовыйФайл ("whats_new.txt") .
```

См. также:

- [МожноОткрытьФайл](#)

ИмяКаталога

Возвращает путь к каталогу, в котором находится указанный файл.

Синтаксис

ИмяКаталога (Путь)

Параметры

Путь (текст) — полный путь доступа к указанному файлу. Если указано только имя файла с расширением, то функция будет искать указанный файл в каталоге программы СБиС++.

Возвращает

Текст, путь к каталогу.

Примеры

Необходимо в файл записать строку, содержащую путь к каталогу программы СБиС++, в котором находится файл «sbs.exe»:

```
офайл = файл ("Proba2.txt") ;  
офайл.ЗаписатьСтроку ("СБиС=" +  
ИмяКаталога (ИмяИсполняемогоФайла ( ) ) ) ;  
офайл.Закрыть ( ) ;
```

См. также:



- [ИмяФайла](#)
- [РасширениеФайла](#)
- [КаталогВременныхФайлов](#)
- [КаталогМодулей](#)
- [ИмяИсполняемогоФайла](#)

КаталогВременныхФайлов

Возвращает путь к каталогу, где находятся временные файлы программы.

Синтаксис

`КаталогВременныхФайлов ()`

Параметры

Не указываются.

Возвращает

Текст, путь к каталогу временных файлов, прописанный у параметра «Временные» в файле «sbis.ini». Если путь в файле не прописан, то функция возвратит путь к каталогу «tmp», указанный по умолчанию.

Пример

Необходимо скопировать временные файлы из каталога «db» в каталог временных файлов:

```
пКаталог = КаталогВременныхФайлов ( ) ;  
СкопироватьФайл ("db\\*.tmp" , пКаталог) ;
```

См. также:

- [ИмяКаталога](#)
- [ИмяФайла](#)
- [РасширениеФайла](#)
- [КаталогМодулей](#)
- [ИмяИсполняемогоФайла](#)



КаталогМодулей

Возвращает путь к каталогу модулей текущей загруженной конфигурации.

Синтаксис

`КаталогМодулей ()`

Параметры

Не указываются.

Возвращает

Текст - путь к каталогу, в котором находятся модули текущей загруженной конфигурации.

Примеры

Необходимо рядом с каталогом модулей создать каталог «files»:

```
пКаталог = КаталогМодулей ( ) ;
пПоз = НайтиСКонца ( пКаталог , "\\\" ); # Ищем
последний слэш;
Если ( пПоз == Размер ( пКаталог ) )
    пКаталог = Подстрока ( пКаталог , 1 , пПоз - 1 ) ;
пПоз = НайтиСКонца ( пКаталог , "\\\" ); # Ищем каталог
СБиС++;
Если ( пПоз )
    пКаталог = Подстрока ( пКаталог , 1 , пПоз - 1 ) ;
пКаталог = пКаталог + "\\files\\" ; # К каталогу
СБиС++ добавляем files ;
СоздатьКаталог ( пКаталог ) ;
```

См. также:

- [ИмяКаталога](#)
- [ИмяФайла](#)
- [РасширениеФайла](#)
- [КаталогВременныхФайлов](#)
- [ИмяИсполняемогоФайла](#)

ЕстьМодуль



Проверяет, установлен ли указанный модуль в программе. Все установленные модули находятся в каталоге «Модули».

Синтаксис

ЕстьМодуль (НазваниеМодуля)

Параметры

НазваниеМодуля (текст) – название модуля, существование которого необходимо проверить.

Возвращает

Да (логическое значение) - если такой модуль установлен в программе.

Нет (логическое значение) - если модуль не установлен в программе.

Примеры

Функция может быть использована, например, при построении регламентированных форм отчётности:

```
Если (!ЕстьМодуль ("Путевые листы"))  
    Ошибка ("Для заполнения декларации необходимо  
установить модуль Путевые листы") ;
```

См. также:

- [ИмяКаталога](#)
- [ИмяФайла](#)
- [РасширениеФайла](#)
- [КаталогВременныхФайлов](#)
- [ИмяИсполняемогоФайла](#)

ИмяФайла

Возвращает имя файла без расширения.

Синтаксис

Имяфайла (Имя)



Параметры

Имя (текст) — имя файла с расширением. Необходимо указывать полный путь доступа к файлу, иначе функция будет искать файл в каталоге программы СБиС++.

Возвращает

Текст - имя файла без расширения.

Примеры

Необходимо заменить в имени файла пробелы, если таковые имеются, на символ «_»:

```
перем пПолныйПутьКФайлу ;
пИмяфайла = Имяфайла (пПолныйПутьКФайлу) ;
Если (Найти (пИмяфайла , " "))
    пЗамИмяфайла = Заменить (пИмяфайла , " ", "_") ;
```

См. также:

- [УстановитьИмяФайла](#)
- [РасширениеФайла](#)
- [ИмяИсполняемогоФайла](#)

ИмяИсполняемогоФайла

Возвращает путь к исполняемому файлу (файлу запуска программы с расширением «.exe»).

Синтаксис

```
ИмяИсполняемогоФайла ( )
```

Параметры

Не указываются.

Возвращает

Текст - путь к исполняемому файлу.

Примеры

Необходимо в файл записать строку, содержащую путь к каталогу программы СБиС++, в котором находится файл «sbis.exe»:

```
оФайл = Файл ("Proba2.txt");  
оФайл.ЗаписатьСтроку ("СБиС=" +  
ИмяКаталога (ИмяИсполняемогоФайла ( ) ) );  
оФайл.Закрыть ( ) ;
```

См. также:

- [ИмяКаталога](#)
- [ИмяФайла](#)
- [РасширениеФайла](#)
- [КаталогВременныхФайлов](#)
- [КаталогМодулей](#)

УстановитьИмяФайла

Устанавливает имя файла.

Синтаксис

```
Файл.УстановитьИмяФайла (Имя)
```

Параметры

Имя (текст) — имя файла, которое необходимо установить для объекта типа «Файл».

Комментарий

Функция используется в том случае, если объект типа «Файл» создан без привязки к реальному файлу.

Возвращает

Ничего не возвращает.

Примеры

```
оФайл = Файл ( ) ;
```



```
оФайл. УстановитьИмяФайла ("Пример" );  
Сообщить (оФайл.ИмяФайла ( ) )
```

См. также:

- [ИмяФайла](#)
- [РасширениеФайла](#)
- [ИмяИсполняемогоФайла](#)

РасширениеФайла

Возвращает расширение указанного файла.

Синтаксис

```
РасширениеФайла (ИмяФайла)
```

Параметры

ИмяФайла (текст) – имя файла с расширением. Необходимо указывать полный путь доступа к файлу, иначе функция будет искать указанный файл в каталоге программы СБиС++.

Возвращает

Строка (текст) – расширение файла.

Примеры

Необходимо заменить файлы с расширением «.tmp» из каталога «db» программы СБиС++ на файлы с расширением «.TMP»:

```
мСписок = СписокФайлов ("db\\*.tmp" );  
перем мНовСписок [ ] ;  
перем пНом = 0 ;  
Пока (пНом++<Размер (мСписок) )  
{  
    мНовСписок [пНом] = Заменить (мСписок [пНом] ,  
    РасширениеФайла (мСписок [пНом] ) ,  
    Вверх (РасширениеФайла (мСписок [пНом] ) ) ) ;  
    ПереименоватьФайл (мСписок [пНом] ,  
    мНовСписок [пНом] ) ;  
}
```

См. также:



- [ИмяКаталога](#)
- [ИмяФайла](#)

КороткийПуть

Возвращает путь доступа к указанному файлу в формате «8.3» (8 символов имени, 3 символа расширения файла).

Синтаксис

КороткийПуть (ИмяФайла)

Параметры

ИмяФайла (текст) – имя файла с расширением. Этот файл обязательно должен существовать. Необходимо указывать полный путь доступа к файлу, иначе функция будет искать файл в каталоге программы СБис++.

Возвращает

Строка (текст) – путь доступа к файлу в формате«8.3» (8 символов имени, 3 символа расширения файла).

Примеры

Необходимо указать короткий путь доступа к указанному файлу:

```
Сообщить (КороткийПуть ("Описание функций.txt")); # в  
пПуть будет «ОПИСАН~1.TXT».
```

См. также:

- [ИмяФайла](#)
- [РасширениеФайла](#)
- [ИмяИсполняемогоФайла](#)

СоздатьКаталог

Создает каталог с указанным именем.

Синтаксис



СоздатьКаталог (ИмяКаталога)

Параметры

ИмяКаталога (текст) — произвольное имя каталога, создается в каталоге с программой. Можно указать полный путь, где данный каталог должен быть создан.

Возвращает

Да (логическое значение) - если каталог создан.

Нет (логическое значение) - если каталог не создан. При этом процесс не прерывается, а продолжают выполняться следующие действия в программе.

Примеры

Необходимо рядом с каталогом модулей создать каталог «files»:

```
пКаталог = КаталогМодулей ( );
пПоз = НайтиСКонца ( пКаталог, "\\\" ); # Ищем
последний слэш;
Если ( пПоз == Размер ( пКаталог ) )
    пКаталог = Подстрока ( пКаталог, 1, пПоз - 1 );
пПоз = НайтиСКонца ( пКаталог, "\\\" ); # Ищем каталог
СБиС++;
Если ( пПоз )
    пКаталог = Подстрока ( пКаталог, 1, пПоз - 1 );
пКаталог = пКаталог + "\\files\\"; # К каталогу
СБиС++ добавляем files;
СоздатьКаталог ( пКаталог );
```

См. также:

- [ИмяКаталога](#)
- [КаталогВременныхФайлов](#)
- [КаталогМодулей](#)
- [НайтиКаталог](#)