Кроссплатформенный компилятор языка программирования Oberon-07/11
для i386 Windows/Linux/KolibriOS.

Данная реализация является демонстрационной: предоставляет разработчику лишь минимальные удобства; имеет сильно ограниченную стандартную библиотеку; оптимизация отсутствует.

Состав программы

1. Compiler.exe – транслятор исходного кода (*.ob07 кодировка UTF-8 с BOM-сигнатурой, или ANSI) в машинные коды i386, форматы исполняемых файлов – PE, ELF и MENUET01/MS COFF. Параметры:
   1) имя главного модуля
   2) тип приложения и платформа
      “con” – Windows console
      “gui” – Windows GUI
      “dll” – Windows DLL
      “elf” – Linux
      “kos” – KolibriOS
      “obj” – KolibriOS DLL
   3) размер стека в мегабайтах (необязательный параметр – по умолчанию 1 Мб, для ELF игнорируется). Если 2-й параметр = “obj” (KolibriOS DLL), то этот параметр задается шестнадцатиричным числом (0x00000001..0xffffffff) и определяет версию программы, по умолчанию - 1.

   Например:

   ```
   "C:\oberon-07\example.ob07" con 1
   "C:\oberon-07\example.ob07" obj 0x00020005
   ```

   В случае успешной компиляции, компилятор передает код завершения 0, иначе 1. При работе компилятора в KolibriOS, код завершения не передается. Сообщения компилятора выводятся на консоль (Windows), в терминал (Linux), на доску отладки (KolibriOS).
2. Editor.exe – многостраничный текстовый редактор для редактирования файлов модулей (*.ob07) с подсветкой синтаксиса, нумерацией строк и автоматическим преобразованием ключевых слов и стандартных идентификаторов к верхнему регистру. Использует кодировку UTF-8 с BOM-сигнатурой.
3. Compile.cmd – командный файл для запуска компилятора из редактора

4. Elf – шаблон исполняемого файла формата ELF
5. Errors.txt – список ошибок компиляции
6. папка Lib – стандартные модули
7. папка Source\Compiler – исходный код компилятора
8. папка Source\Editor – исходный код редактора

Отличия от оригинала.

1. Расширен псевдомодуль SYSTEM.
2. В идентификаторах может использоваться символ подчеркивания "_".
3. Добавлены системные флаги.
4. Оператор CASE реализован в соответствии с синтаксисом и семантикой данного оператора в языке Oberon (Revision 1.10.90).
5. Расширен набор стандартных процедур.
6. Семантика охраны/проверки типа уточнена для нулевого указателя.
7. Семантика DIV и MOD уточнена определением результата деления на отрицательные числа.
8. Добавлены однострочные комментарии (начинаются с пары символов "//")

Особенности реализации.

1. Основные типы.

Тип	Диапазон значений	Размер, байт
INTEGER	-2147483648 .. 2147483647	4
REAL	1.40E-45 .. 3.34E+38	4
LONGREAL	4.94E-324 .. 1.70E+308	8
CHAR	символ ASCII (0X .. 0FFX)	1
BOOLEAN	FALSE, TRUE	1
SET	множество из целых чисел {0 .. 31}	4

2. Максимальная длина идентификаторов – 255 символов
3. Максимальная длина строковых констант – 255 символов
4. Максимальная длина строк исходного кода – 511 символов
5. Максимальная размерность открытых массивов – 5

6. Максимальное количество объявленных типов-записей – 2047
7. Стандартная процедура NEW при выделении блока памяти, заполняет его нулями
8. Глобальные и локальные переменные инициализируются нулями
9. В отличие от многих Oberon-реализаций, данная реализация в настоящее время не имеет сборщика мусора

Псевдомодуль SYSTEM

Псевдомодуль SYSTEM содержит низкоуровневые и небезопасные процедуры, ошибки при использовании процедур псевдомодуля SYSTEM могут привести к повреждению данных времени выполнения и аварийному завершению программы.

PROCEDURE ADR(v: любой тип): INTEGER

v – переменная, процедура или строковая константа; результат - адрес v

PROCEDURE BIT(a, n: INTEGER): BOOLEAN

n-й бит Память[a] (0 <= n <= 7)

PROCEDURE SIZE(T): INTEGER

размер типа T

PROCEDURE TYPEID(T): INTEGER

номер типа в таблице типов-записей

T – тип-запись или тип-указатель

PROCEDURE GET(a: INTEGER; VAR v: любой основной тип)

v := Память[a]

PROCEDURE PUT(a: INTEGER; x: любой основной тип)

Память[a] := x

PROCEDURE MOVE(Source, Dest, n: INTEGER)

Копирует n байт памяти из Source в Dest,

области Source и Dest не должны перекрываться

PROCEDURE CODE(s: ARRAY OF CHAR)

Вставка машинного кода

s – строковая константа шестнадцатиричных цифр

количество цифр должно быть четным

например: SYSTEM.CODE("B801000000") (* mov eax, 1 *)

Также в модуле SYSTEM определен тип CARD16 (2 байта). Для типа CARD16 не допускаются никакие явные операции, за исключением присваивания. Преобразования CARD16 → INTEGER и INTEGER → CARD16 могут быть реализованы так:

```
PROCEDURE Card16ToInt(w: SYSTEM.CARD16): INTEGER;
VAR i: INTEGER;
BEGIN
  SYSTEM.PUT(SYSTEM.ADR(i), w)
  RETURN i
END Card16ToInt;

PROCEDURE IntToCard16(i: INTEGER): SYSTEM.CARD16;
VAR w: SYSTEM.CARD16;
BEGIN
  SYSTEM.GET(SYSTEM.ADR(i), w)
  RETURN w
END IntToCard16;
```

Системные флаги

При объявлении процедурных типов и глобальных процедур, после ключевого слова PROCEDURE может быть указан системный флаг соглашения вызова.
Его формат: "[" ("stdcall" | "cdecl" | "winapi") "]", то есть [stdcall], [cdecl] или [winapi]. Например:

```
PROCEDURE [cdecl] MyProc(x, y, z: INTEGER): INTEGER;
```

Если указан флаг [winapi], то принимается соглашение stdcall и процедуру-функцию можно вызвать как собственно процедуру, вне выражения. Флаг [winapi] доступен только для платформы Windows.

При объявлении типов-записей, после ключевого слова RECORD может быть указан системный флаг [noalign] или [union]. Флаг [noalign] означает отсутствие выравнивания полей

записи, а флаг [union] означает, что смещения всех полей записи равны нулю, при этом размер записи равен размеру наибольшего поля. Записи RECORD [union] … END соответствуют объединениям (union) в языке C. Записи с системными флагами не могут иметь базового типа и не могут быть базовыми типами для других записей.

Использование любого системного флага требует импорта модуля SYSTEM.

Оператор CASE.

Синтаксис оператора CASE:

```
CaseStatement = CASE Expression OF Case {"|" Case} [ELSE StatementSequence] END.
Case = [CaseLabelList ":" StatementSequence].
CaseLabelList = CaseLabels {"," CaseLabels}.
CaseLabels = ConstExpression [".." ConstExpression].
```

Например:

```
CASE x OF
|-1:    DoSomething1
| 1:    DoSomething2
| 0:    DoSomething3
ELSE
      DoSomething4
END
```

От оригинального синтаксиса отличается возможностью использования константных выражений в метках вариантов, а не только собственно констант, а также наличием необязательной ветки ELSE. Если не выполнен ни один вариант оператора CASE и ELSE отсутствует, то программа прерывается с ошибкой времени выполнения.

Проверка/охрана типа нулевого указателя.

Оригинальное сообщение о языке не определяет поведение программы при выполнение охраны p(T) и проверки типа p IS T при p=NIL. Во многих Oberon-реализациях, выполнение такой операции приводит к ошибке времени выполнения. В данной реализации охрана типа нулевого указателя не приводит к ошибке, а проверка типа дает результат FALSE. В ряде случаев это позволяет значительно сократить частоту применения охраны типа.

Дополнительные стандартные процедуры.

DISPOSE(VAR v: любой_указатель)
Освобождает память, выделенную процедурой NEW для динамической переменной v^, и присваивает переменной v значение NIL.

LSR(x, n: INTEGER): INTEGER
Логический сдвиг x на n бит вправо.

BITS(x: INTEGER): SET
Интерпретирует x как значение типа SET. Не производит машинного кода, выполняется на этапе компиляции.

LENGTH(s: ARRAY OF CHAR): INTEGER
Длина 0X-завершенной строки s, без учета символа 0X. Если символ 0X отсутствует, функция возвращает длину массива s.

DIV и MOD.

x	y	x DIV y	x MOD y
5	3	1	2
-5	3	-2	1
5	-3	-2	-1
-5	-3	1	-2

Скрытые параметры процедур.

Некоторые процедуры могут иметь скрытые параметры, они отсутствуют в списке формальных параметров, но используются компилятором. Это возможно в следующих случаях:

1. Процедура имеет формальный параметр открытый массив:
   PROCEDURE Proc(x: ARRAY OF ARRAY OF LONGREAL);
   Вызов транслируется так:
   Proc(SYSTEM.ADR(x), LEN(x), LEN(x[0])
2. Процедура имеет формальный параметр-переменную типа RECORD:
   PROCEDURE Proc(VAR x: Rec);

Вызов транслируется так:

Proc(SYSTEM.TYPEID(Rec), SYSTEM.ADR(x))

3. Процедура является вложенной, глубина вложения k, для глобальных процедур k = 0

   PROCEDURE Proc(p1, p2, … pn);

   Вызов транслируется так:

   Proc(base(k - 1), base(k - 2), … base(0), p1, p2, … pn), где base(m) – адрес базы кадра стека охватывающей процедуры глубины вложения m (используется для доступа к локальным переменным охватывающей процедуры)

Модуль RTL

Все программы неявно используют модуль RTL. Компилятор транслирует некоторые операции (проверка/охрана типа, сравнение строк, сообщения об ошибках времени выполнения и др.) как вызовы процедур этого модуля. Пользователь не должен вызывать явно процедуры модуля RTL за исключением процедуры SetClose:

PROCEDURE SetClose(proc: PROC), где TYPE PROC = PROCEDURE

SetClose назначает процедуру proc (без параметров) вызываемой при выгрузке dll-библиотеки (Windows), если приложение компилируется как Windows DLL. Для прочих типов приложений и платформ вызов процедуры SetClose ни на что не влияет. Сообщения об ошибках времени выполнения выводятся в диалоговых окнах (Windows), в терминал (Linux), на доску отладки (KolibriOS).

Модуль API

Существуют три реализации модуля API: для Windows, Linux и KolibriOS. Как и модуль RTL, модуль API не предназначен для прямого использования. Он обеспечивает кроссплатформенность компилятора.

Генерация исполняемых файлов DLL

Разрешается экспортировать только процедуры. Для этого, процедура должна находиться в главном модуле программы, и ее имя должно быть отмечено символом экспорта (“*”). KolibriOS DLL всегда экспортируют идентификаторы version (версия программы) и START – адрес процедуры инициализации DLL:

PROCEDURE [stdcall] START (state: INTEGER): INTEGER

Эта процедура должна быть вызвана перед использованием DLL. Параметр state зарезервирован и может быть любым целым числом. Процедура всегда возвращает 1.

В настоящее время генерация DLL для Linux не реализована.

Стандартные модули (Windows)

MODULE Out – консольный вывод

PROCEDURE Open

открывает консольный вывод

PROCEDURE Int(x, width: INTEGER)

вывод целого числа x;

width - количество знакомест, используемых для вывода

PROCEDURE Real(x: LONGREAL; width: INTEGER)

вывод вещественного числа x в плавающем формате;

width - количество знакомест, используемых для вывода

PROCEDURE Char(x: CHAR)

вывод символа x

PROCEDURE FixReal(x: LONGREAL; width, p: INTEGER)

вывод вещественного числа x в фиксированном формате;

width - количество знакомест, используемых для вывода;

p - количество знаков после десятичной точки

PROCEDURE Ln

переход на следующую строку

PROCEDURE String(s: ARRAY OF CHAR)

вывод строки s

PROCEDURE Utf8(s: ARRAY OF CHAR)

вывод строки s, представленной в кодировке UTF-8

MODULE In – консольный ввод

VAR Done: BOOLEAN

принимает значение TRUE в случае успешного выполнения операции ввода и FALSE в противном случае

PROCEDURE Open

открывает консольный ввод,
также присваивает переменной Done значение TRUE

PROCEDURE Int(VAR x: INTEGER)

ввод числа типа INTEGER

PROCEDURE Char(VAR x: CHAR)

ввод символа

PROCEDURE Real(VAR x: REAL)

ввод числа типа REAL

PROCEDURE LongReal(VAR x: LONGREAL)

ввод числа типа LONGREAL

PROCEDURE String(VAR s: ARRAY OF CHAR)

ввод строки

PROCEDURE Ln

ожидание нажатия ENTER

MODULE Console – дополнительные процедуры консольного вывода

CONST

Следующие константы определяют цвет консольного вывода

Black = 0 Blue = 1 Green = 2 Cyan = 3 Red = 4
Magenta = 5 Brown = 6 LightGray = 7 DarkGray = 8
LightBlue = 9 LightGreen = 10 LightCyan = 11
LightRed = 12 LightMagenta = 13 Yellow = 14
White = 15

PROCEDURE Cls
    очистка окна консоли

PROCEDURE SetColor(FColor, BColor: INTEGER)
    установка цвета консольного вывода: FColor – цвет текста,
    BColor – цвет фона, возможные значения – вышеперечисленные константы

PROCEDURE SetCursor(x, y: INTEGER)
    установка курсора консоли в позицию (x, y)

PROCEDURE GetCursor(VAR x, y: INTEGER)
    записывает в параметры текущие координаты курсора консоли

PROCEDURE GetCursorX(): INTEGER
    возвращает текущую x-координату курсора консоли

PROCEDURE GetCursorY(): INTEGER
    возвращает текущую y-координату курсора консоли

MODULE Math – математические функции

CONST

pi* = 3.141592653589793D+00

e* = 2.718281828459045D+00

VAR INF, negINF: LONGREAL
положительная и отрицательная бесконечность

PROCEDURE IsNan(x: LONGREAL): BOOLEAN
возвращает TRUE, если x – не число

PROCEDURE IsInf(x: LONGREAL): BOOLEAN
возвращает TRUE, если x – бесконечность

PROCEDURE sqrt(x: LONGREAL): LONGREAL
квадратный корень x

PROCEDURE exp(x: LONGREAL): LONGREAL
экспонента x

PROCEDURE ln(x: LONGREAL): LONGREAL
натуральный логарифм x

PROCEDURE sin(x: LONGREAL): LONGREAL
синус x

PROCEDURE cos(x: LONGREAL): LONGREAL
косинус x

PROCEDURE tan(x: LONGREAL): LONGREAL
тангенс x

PROCEDURE arcsin(x: LONGREAL): LONGREAL
арксинус x

PROCEDURE arccos(x: LONGREAL): LONGREAL
арккосинус x

PROCEDURE arctan(x: LONGREAL): LONGREAL

арктангенс x

PROCEDURE arctan2(y, x: LONGREAL): LONGREAL

арктангенс y/x

PROCEDURE power(base, exponent: LONGREAL): LONGREAL

возведение числа base в степень exponent

PROCEDURE log(base, x: LONGREAL): LONGREAL

логарифм x по основанию base

PROCEDURE sinh(x: LONGREAL): LONGREAL

гиперболический синус x

PROCEDURE cosh(x: LONGREAL): LONGREAL

гиперболический косинус x

PROCEDURE tanh(x: LONGREAL): LONGREAL

гиперболический тангенс x

PROCEDURE arcsinh(x: LONGREAL): LONGREAL

обратный гиперболический синус x

PROCEDURE arccosh(x: LONGREAL): LONGREAL

обратный гиперболический косинус x

PROCEDURE arctanh(x: LONGREAL): LONGREAL

обратный гиперболический тангенс x

PROCEDURE round(x: LONGREAL): LONGREAL

округление x до ближайшего целого

PROCEDURE frac(x: LONGREAL): LONGREAL;

дробная часть числа x

PROCEDURE floor(x: LONGREAL): LONGREAL

наибольшее целое число (представление как LONGREAL),

не больше x: floor(1.2) = 1.0

PROCEDURE ceil(x: LONGREAL): LONGREAL

наименьшее целое число (представление как LONGREAL),

не меньше x: ceil(1.2) = 2.0

PROCEDURE sgn(x: LONGREAL): INTEGER

если x > 0 возвращает 1

если x < 0 возвращает -1

если x = 0 возвращает 0

MODULE File – работа с файлами

CONST

OPEN_R = 0

OPEN_W = 1

OPEN_RW = 2

SEEK_BEG = 0

SEEK_CUR = 1

SEEK_END = 2

PROCEDURE Create(FName: ARRAY OF CHAR): INTEGER

создает новый файл с именем FName (полное имя с путем), открывет файл для записи и возвращает идентификатор файла (целое число), в случае ошибки, возвращает -1

PROCEDURE Open(FName: ARRAY OF CHAR;

Mode: INTEGER): INTEGER

открывает существующий файл с именем FName (полное имя с путем) в режиме Mode = (OPEN_R (только чтение), OPEN_W (только запись), OPEN_RW (чтение и запись)), возвращает идентификатор файла (целое число), в случае ошибки,

возвращает -1

PROCEDURE Read(F, Buffer, Count: INTEGER): INTEGER

Читает данные из файла в память. F - числовой идентификатор файла, Buffer – адрес области памяти, Count – количество байт, которое требуется прочитать из файла; возвращает количество байт, которое было прочитано из файла

PROCEDURE Write(F, Buffer, Count: INTEGER): INTEGER

Записывает данные из памяти в файл. F - числовой идентификатор файла, Buffer – адрес области памяти, Count – количество байт, которое требуется записать в файл; возвращает количество байт, которое было записано в файл

PROCEDURE Seek(F, Offset, Origin: INTEGER): INTEGER

устанавливает позицию чтения-записи файла с идентификатором F на Offset, относительно Origin = (SEEK_BEG – начало файла,
SEEK_CUR – текущая позиция, SEEK_END – конец файла), возвращает позицию относительно начала файла, например: Seek(F, 0, 2) - устанавливает позицию на конец файла и возвращает длину файла; при ошибке возвращает -1

PROCEDURE Close(F: INTEGER)

закрывает ранее открытый файл с идентификатором F

PROCEDURE Delete(FName: ARRAY OF CHAR): BOOLEAN

удаляет файл с именем FName (полное имя с путем), возвращает TRUE, если файл успешно удален

PROCEDURE Exists(FName: ARRAY OF CHAR): BOOLEAN

возвращает TRUE, если файл с именем FName (полное имя) существует

MODULE Read – чтение основных типов данных из файла F

Процедуры возвращают TRUE в случае успешной операции чтения

PROCEDURE Char(F: INTEGER; VAR x: CHAR): BOOLEAN

PROCEDURE Int(F: INTEGER; VAR x: INTEGER): BOOLEAN

PROCEDURE Real(F: INTEGER; VAR x: REAL): BOOLEAN

PROCEDURE LongReal(F: INTEGER; VAR x: LONGREAL): BOOLEAN

PROCEDURE Boolean(F: INTEGER; VAR x: BOOLEAN): BOOLEAN

PROCEDURE Set(F: INTEGER; VAR x: SET): BOOLEAN

PROCEDURE Card16(F: INTEGER; VAR x: SYSTEM.CARD16): BOOLEAN

MODULE Write – запись основных типов данных в файл F

Процедуры возвращают TRUE в случае успешной операции записи

PROCEDURE Char(F: INTEGER; x: CHAR): BOOLEAN

PROCEDURE Int(F: INTEGER; x: INTEGER): BOOLEAN

PROCEDURE Real(F: INTEGER; x: REAL): BOOLEAN

PROCEDURE LongReal(F: INTEGER; x: LONGREAL): BOOLEAN

PROCEDURE Boolean(F: INTEGER; x: BOOLEAN): BOOLEAN

PROCEDURE Set(F: INTEGER; x: SET): BOOLEAN

PROCEDURE Card16(F: INTEGER; x: SYSTEM.CARD16): BOOLEAN

MODULE Dir – работа с папками

PROCEDURE Create(DirName: ARRAY OF CHAR): BOOLEAN

создает папку с именем DirName, все промежуточные папки должны существовать

PROCEDURE Remove(DirName: ARRAY OF CHAR): BOOLEAN

удаляет пустую папку с именем DirName

PROCEDURE Exists(DirName: ARRAY OF CHAR): BOOLEAN

возвращает TRUE, если папка с именем DirName существует

MODULE DateTime – дата, время

CONST ERR = -7.0D5

PROCEDURE Now(VAR Year, Month, Day, Hour, Min, Sec, MSec: INTEGER)

возвращает в параметрах компоненты текущей системной даты и времени

PROCEDURE Encode(Year, Month, Day, Hour, Min, Sec, MSec: INTEGER): LONGREAL

возвращает дату, полученную из компонентов
Year, Month, Day, Hour, Min, Sec, MSec;
при ошибке возвращает константу ERR = -7.0D5

PROCEDURE Decode(Date: LONGREAL; VAR Year, Month, Day,
Hour, Min, Sec, MSec: INTEGER): BOOLEAN

извлекает компоненты
Year, Month, Day, Hour, Min, Sec, MSec из даты Date;
при ошибке возвращает FALSE

MODULE Utils – разное

VAR ParamCount: INTEGER - количество параметров программы

PROCEDURE Utf8To16(source: ARRAY OF CHAR;
VAR dest: ARRAY OF CHAR): INTEGER;

преобразует символы строки source из кодировки UTF-8 в кодировку UTF-16, результат записывает в строку dest, возвращает количество 16-битных символов, записанных в dest

PROCEDURE ParamStr(VAR str: ARRAY OF CHAR; n: INTEGER)

записывает в строку str n-й параметр программы

PROCEDURE PutSeed(seed: INTEGER)

Инициализация генератора случайных чисел целым числом seed

PROCEDURE Rnd(range: INTEGER): INTEGER

Целые случайные числа в диапазоне 0 <= x < range

MODULE WINAPI – привязки к некоторым API-функциям Windows

Стандартные модули (KolibriOS)

MODULE Out – консольный вывод

Интерфейс – как модуль Out для Windows, за исключением отсутствующей процедуры Utf8.

MODULE In – консольный ввод

Интерфейс – как модуль In для Windows

MODULE Console – дополнительные процедуры консольного вывода.

Интерфейс – как модуль Console для Windows

MODULE ConsoleLib – интерфейс библиотеки console.obj

MODULE Math – математические функции

Интерфейс – как модуль Math для Windows

MODULE Dir – работа с папками

Интерфейс – как модуль Dir для Windows

MODULE FSys – типы данных для работы с файловой системой

```
TYPE

FNAME = ARRAY 520 OF CHAR

FS = POINTER TO rFS

rFS = RECORD (* информационная структура файла *)
        subfunc, pos, hpos, bytes, buffer: INTEGER;
        name: FNAME
END

FD = POINTER TO rFD

rFD = RECORD (* структура блока данных входа каталога *)
        attr: INTEGER;
        ntyp: CHAR;
        reserved: ARRAY 3 OF CHAR;
        time_create, date_create,
        time_access, date_access,
        time_modif,   date_modif,
        size, hsize: INTEGER;
        name: FNAME
END

PROCEDURE GetFileInfo*(FName: ARRAY OF CHAR; VAR Info: rFD): BOOLEAN
        Записывает структуру блока данных входа каталога для файла или папки с именем
        FName в параметр Info. В случае ошибки, возвращает FALSE.


MODULE File – работа с файлами

CONST

SEEK_BEG = 0
SEEK_CUR = 1
```

SEEK_END = 2

PROCEDURE Exists(FName: ARRAY OF CHAR): BOOLEAN

возвращает TRUE, если файл с именем FName (полное имя) существует

PROCEDURE Close(VAR F: FSys.FS)

освобождает память, выделенную для информационной структуры файла F и присваивает F значение NIL

PROCEDURE Open(FName: ARRAY OF CHAR): FSys.FS

возвращает указатель на информационную структуру файла с именем FName (полное имя с путем), в случае ошибки, возвращает NIL

PROCEDURE Delete(FName: ARRAY OF CHAR): BOOLEAN

удаляет файл с именем FName (полное имя с путем), возвращает TRUE, если файл успешно удален

PROCEDURE Seek(F: FSys.FS; Offset, Origin: INTEGER): INTEGER

устанавливает позицию чтения-записи файла с указателем на информационную структуру F на Offset, относительно Origin = (SEEK_BEG – начало файла, SEEK_CUR – текущая позиция, SEEK_END – конец файла), возвращает позицию относительно начала файла, например: Seek(F, 0, 2) - устанавливает позицию на конец файла и возвращает длину файла; при ошибке возвращает -1

PROCEDURE Read(F: FSys.FS; Buffer, Count: INTEGER): INTEGER

Читает данные из файла в память. F - указатель на информационную структуру файла, Buffer – адрес области памяти, Count – количество байт, которое требуется прочитать из файла; возвращает количество байт, которое было прочитано из файла и соответствующим образом изменяет позицию чтения/записи в информационной структуре F.

PROCEDURE Write(F: FSys.FS; Buffer, Count: INTEGER): INTEGER

Записывает данные из памяти в файл. F - указатель на информационную структуру файла, Buffer – адрес области памяти, Count – количество байт, которое требуется записать в файл; возвращает количество байт, которое было записано в файл и соответствующим образом изменяет позицию чтения/записи в информационной

структуре F.

PROCEDURE Create(FName: ARRAY OF CHAR): FSys.FS

создает новый файл с именем FName (полное имя с путем), возвращает указатель на информационную структуру файла, в случае ошибки, возвращает NIL

MODULE Read – чтение основных типов данных из файла F

Процедуры возвращают TRUE в случае успешной операции чтения

PROCEDURE Char(F: FSys.FS; VAR x: CHAR): BOOLEAN

PROCEDURE Int(F: FSys.FS; VAR x: INTEGER): BOOLEAN

PROCEDURE Real(F: FSys.FS; VAR x: REAL): BOOLEAN

PROCEDURE LongReal(F: FSys.FS; VAR x: LONGREAL): BOOLEAN

PROCEDURE Boolean(F: FSys.FS; VAR x: BOOLEAN): BOOLEAN

PROCEDURE Set(F: FSys.FS; VAR x: SET): BOOLEAN

PROCEDURE Card16(F: FSys.FS; VAR x: SYSTEM.CARD16): BOOLEAN

MODULE Write – запись основных типов данных в файл F

Процедуры возвращают TRUE в случае успешной операции записи

PROCEDURE Char(F: FSys.FS; x: CHAR): BOOLEAN

PROCEDURE Int(F: FSys.FS; x: INTEGER): BOOLEAN

PROCEDURE Real(F: FSys.FS; x: REAL): BOOLEAN

PROCEDURE LongReal(F: FSys.FS; x: LONGREAL): BOOLEAN

PROCEDURE Boolean(F: FSys.FS; x: BOOLEAN): BOOLEAN

PROCEDURE Set(F: FSys.FS; x: SET): BOOLEAN

PROCEDURE Card16(F: FSys.FS; x: SYSTEM.CARD16): BOOLEAN

MODULE DateTime – дата, время

CONST ERR = -7.0D5

PROCEDURE Now(VAR Year, Month, Day, Hour, Min, Sec: INTEGER)
возвращает в параметрах компоненты текущей системной даты и времени

PROCEDURE Encode(Year, Month, Day, Hour, Min, Sec: INTEGER): LONGREAL
возвращает дату, полученную из компонентов
Year, Month, Day, Hour, Min, Sec;
при ошибке возвращает константу ERR = -7.0D5

PROCEDURE Decode(Date: LONGREAL; VAR Year, Month, Day,
Hour, Min, Sec: INTEGER): BOOLEAN
извлекает компоненты
Year, Month, Day, Hour, Min, Sec из даты Date;
при ошибке возвращает FALSE

MODULE Utils – разное

VAR ParamCount: INTEGER - количество параметров программы

PROCEDURE ParamStr(VAR str: ARRAY OF CHAR; n: INTEGER)
записывает в строку str n-й параметр программы

MODULE KOSAPI

PROCEDURE sysfunc1(arg1: INTEGER): INTEGER
PROCEDURE sysfunc2(arg1, arg2: INTEGER): INTEGER
…
PROCEDURE sysfunc7(arg1, arg2, arg3, arg4, arg5, arg6, arg7: INTEGER): INTEGER

Обертки для функций API ядра KolibriOS. arg1..arg7 соответствуют регистрам eax, ebx, ecx, edx, esi, edi, ebp перед выполнением системного вызова; результат – значение регистра eax после системного вызова.

PROCEDURE sysfunc22(arg1, arg2: INTEGER; VAR res2: INTEGER): INTEGER

Обертка для функций API ядра KolibriOS. arg1 – регистр eax, arg2 – регистр ebx, res2 – значение регистра ebx после системного вызова; результат – значение регистра eax после системного вызова.

PROCEDURE LoadLib(name: ARRAY OF CHAR): INTEGER

Загружает DLL с полным именем name. Возвращает адрес таблицы экспорта. В случае ошибки, возвращает 0.

PROCEDURE GetProcAdr(name: ARRAY OF CHAR; lib: INTEGER): INTEGER

name – имя процедуры

lib – адрес таблицы экспорта DLL

Возвращает адрес процедуры. В случае ошибки, возвращает 0.

PROCEDURE GetCommandLine(): INTEGER

Возвращает адрес строки параметров

PROCEDURE GetName(): INTEGER

Возвращает адрес строки с именем программы

Стандартные модули (Linux)

В настоящее время не реализованы.