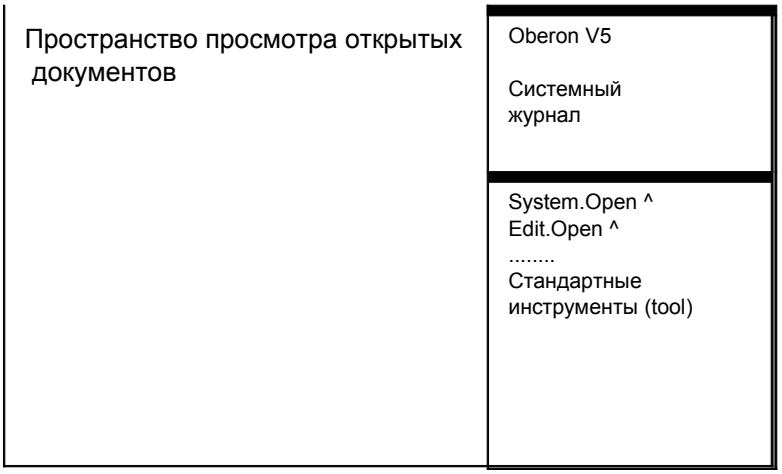
Как использовать Oberon SystemV5

Никлаус Вирт, Ноябрь 2015

Система Oberon разработана в 1990 году как пример надежной, компактной, но мощной, расширяемой операционной системы, которую может полностью контролировать и конструировать один пользователь. В отличие от систем коммерческих, она проста, обозрима, хорошо структурирована, легко доступна, в том числе и для преподавательских целей. Она была написана на лаконичном типобезопасном языке Oberon и требует ничтожной доли ресурсов, которые используют коммерческие системы.

Как начать?

После запуска, на дисплее появится следующий вид.



Справа находится системный трэк с двумя окнами просмотра (вьюверами). Верхнее отображает журнал (Oberon.Log) , в котором система описывает свое поведение и действия пользователя. Нижнее окно - это окно стандартных инструментов, содержащее различные общие команды (Мы называем «инструменты» - тексты содержащие список команд для определенной цели).

Команды в Oberon имеют вид M.P, где M - имя модуля а P имя процедуры модуля M. Команды могут встречаться в любом месте текста и системы, и они активируются при наведении на них курсора и нажатии средней кнопки мыши (СКМ) или колеса мыши. Пустое место слева, ожидает новые вьюверы, это трек пользователя. Например, мы хотим отобразить файл Sample.Mod, то выделем его имя удерживая правую клавишу мыши (ПКМ), а затем активируем команду Edit.Open с помощью СКМ, в нижнем правом окне инструментов. Если имя файла не отображается где-либо на экране, введите его (см. редактирование текстов ниже). Теперь слева откроется новый вьювер, содержащий текст модуля Sample.Mod. Если файла с указанным именем не существует - появится пустой вьювер.

Видимые сейчас вьюверы являются средствами просмотра текста.
Они состоят из двух частей, называемых фреймами.

Узкая верхняя часть называется строкой заголовка. Она содержит название отображаемого текста и несколько команд для работы с вьювером и редактирования текста, среди них:

System.Close закрывает программу просмотра

Edit.Store сохраняет текст в файл

Edit.Search поиск выделенного слова по положению каретки

Редактирование текстов

Наиболее частой задачей является редактирование текста. Для этого случая некоторые команды можно вводить простым щелчком мыши, а не щелчком по названию команды. Когда нужно ввести текст, он всегда вставляется в позицию каретки отображаемого в виде уголка. Каретка позиционируется путем наведения курсора на нужную позицию и нажатием левой клавиши мыши (ЛКМ). Затем вводится текст простым набором. Часть текста может быть выделена для входных параметров команды. Для этого необходимо выделить курсором фрагмент текста, удерживая нажатой ПКМ. Выделение можно удалить, кратковременно нажав ЛКМ удерживая нажатой ПКМ. Это называется *интерклик*. Аналогично, выделенный фрагмент текста можно скопировать на место каретки, нажав СКМ. Функции клавиш мыши представлены в следующей таблице (жирным шрифтом основная команда клавиши):

	интерклик ЛКМ	интерклик СКМ	интерклик ПКМ
ЛКМ **установить каретку**	---	вставить копию	копировать
СКМ **выполнить команду**	---	---	---
ПКМ **выбрать**	удалить	копировать	---

На клавиатуре есть две специальные клавиши управления:

Esc - отменяет выбор, курсор и метку «звездочка».

Backspace - удаляет символ слева от каретки.

В программах просмотра текста по левому краю расположена полоса прокрутки. Когда курсор находится в полосе прокрутки, он принимает форму указателя вверх/вниз. Нажатие клавиши мыши вызывает прокрутку текста вверх или вниз.

ЛКМ - текст перемещается вверх или к концу.

СКМ - перемещает бегунок в позицию курсора.

ПКМ - текст перемещается вниз или к началу.

Управление вьюверами

Вьюверы можно расширять, перемещать или копировать. Они увеличиваются или уменьшаются нажатием ЛКМ, когда курсор находится в заголовке, а затем перетаскиванием его вверх или вниз. Вьювер перемещается в другое место, также щелкая ЛКМ+СКМ. Дубликат вьювера можно создать, активировав команду System.Copy в строке заголовка. Обратите внимание, что в этом случае старый и новый вьювер отображают один и тот же текст, а не копию текста. Эта возможность может пригодиться, когда фрагмент текста необходимо переместить или скопировать в позицию, не видимую в том же вьювере (перемещение на большое расстояние).

Команда System.Grow в строке заголовка создает копию на на весь экран (левый трек). При закрытии вьювера (команда System.Close в строке заголовка), исходный вьювер снова появляется.

Команды

Следующие команды появляются в окне стандартных инструментов System.Tool.

Символ ^ указывает на то, что имя должно быть выбрано ранее. Затем команда применяется к этому выделенному тексту (как в примере выше).

Символ ~ является завершающим символом команды. Он отделяет команду от любого текста идущего далее по строке, дабы избежать ошибок.

System.Open ^	открыть вьювер в системном треке справа
System.Recall	закрыть последний открытый вьювер
Edit.Open ^	открыть вьювер в пользовательском треке слева
Edit.Recall	отменить последнюю операцию удаления
Edit.ChangeFont	заменить шрифт выделенного текста
Edit.SetFont	использовать указанный шрифт для последующего ввода текста
System.Directory ^	поиск каталога для выделенного имени
System.Free ~	выгрузить указанные модули
System.CopyFiles => ~	копировать, например, file1 => file2 file3 => file4 ~
System.RenameFiles => ~	переименовать, например, file1 => file2 file3 => file4 ~
System.DeleteFiles ~	удалить, например, file1 file2 file3 ~ (из каталога)
System.ShowModules ~	показать модули
System.ShowCommands ^	показать команды (например, процедуры выделенного модуля)
ORP.Compile @	компилировать выделенный текст
Hilbert. Draw	нарисовать кривую Гильберта, (используется как пример).

При нажатии на команду M.P, модуль M ищется в хранилище и если он найден то загружается в основное хранилище. Затем ищется его процедура P и выполняется. Список загруженных модулей может быть сформирован командой System.ShowModules, а список его команд можно получить командой System.ShowCommands. Любая процедура без параметров в любом (скомпилированном) модуле является доступной как команда. Доступ к ее параметрам осуществляется через сканер. В качестве примера, рассмотрим следующий модуль:

```
MODULE M0;
        IMPORT Texts, Oberon;
         VAR W: Texts.Writer;

    PROCEDURE P0*;
        VAR sum: INTEGER; S: Texts.Scanner;
    BEGIN sum := 0;
        Texts.OpenScanner(S, Oberon.Par.text, Oberon.Par.pos); Texts.Scan(S);
        WHILE S.class = Texts.Int DO
          Texts.WriteInt(W, S.i, 6); sum := sum + S.i; Texts.Scan(S)
        END ;
        Texts.Write(W, sum, 8); Texts.WriteLn(W); Texts.Append(Oberon.Log, W.buf)
    END P0;
BEGIN Texts.OpenWriter(W)


END M0.
```

После успешной компиляции (ORP.Compile @), команда: M0.P0 2 3 5 7 11 ~ приводит к выводу этих чисел и их суммы в стандартный системный журнал System.Log.

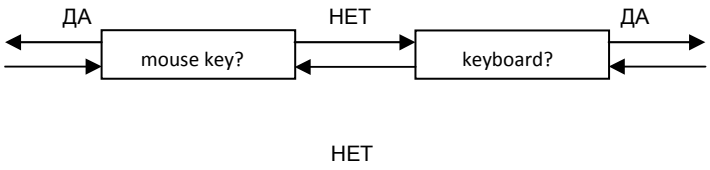
Ядро системы Oberon

Ядро системы состоит из цикла, который непрерывно отслеживает появление команды. Команда идентифицируется, управление передается, и команда выполняется. Команда может исходить от явного нажатия СКМ на текст M.P, или это может быть ЛКМ или ПКМ (см. редактирование команды).

Еще одним источником ввода является клавиатура. Если нажата какая-либо клавиша, это интерпретируется как команда на чтение этого символа. Исключение составляют клавиши Esc, Ctrl-z (или F1). Esc интерпретируется

как команда отменить все выделения, backspace - для удаления символа слева от каретки, а Ctrl-z - для установки глобального маркера (звездочка).

Первоначально загруженная система содержит, кроме модуля Oberon, модуль System, текстовую систему (модули TextFrames, MenuViewers, Texts, Fonts, Input), систему просмотра (модули Viewers, Display), загрузчик и компоновщик (модуль Modules), файловая система (модули Files, FileDir), менеджер дискового пространства и сборщик мусора (модуль Kernel).



Компилятор загружается по требованию, как и другие прикладные программы.

https://www.inf.ethz.ch/personal/wirth/ProjectOberon/index.html

http://www.projectoberon.com/

5