Мартин Рейзер

Краткий экскурс в систему Oberon

Martin Reiser (1991) The Oberon System: User Guide and Programmer's Manual // ACM Press.

1995, А. Л. Китаев, перевод с англ.

1. Немного истории

Профессор Никлаус Вирт уже давно проявляет интерес к компиляторам, языкам программирования и персональным рабочим станциям. Его первое крупное достижение связано с языком программирования Паскаль, появившимся на свет вскоре после назначения Вирта в ETH в 1968 г. (Eidgenoessische Technische Hochschule, или Swiss Federal Institute of Technology, или Швейцарский Федеральный технологический институт). Инженер-электрик по образованию, он избрал уникальный подход к проектированию языков — параллельно с разработкой аппаратного обеспечения.

Первым плодом этой работы стал персональный компьютер Lilith, спроектированный на основе набора микропроцессоров AMD2901 фирмы Advanced MicroDevices. Он был создан для того, чтобы изучить возможности использования написанных на Паскале программ на машинах со стековой архитектурой. В ходе работы над проектом стали очевидны ограничения Паскаля как языка системного программирования, что привело к появлению языка программирования Modula, а позднее и Modula-2, непосредственно предназначенных для поддержки программного обеспечения рабочей станции Lilith.

Преемником Lilith стал компьютер Ceres-1 (спроектированный Х.Эберле — H.Eberle — и Н. Виртом.), конструктивно решенный в более традиционном духе. Эта машина была создана на основе микропроцессора NS32032, который лучше других имевшихся на рынке чипов подходил для поддержки стекового языка типа Modula, обеспечивающего раздельную компиляцию. Ceres оснащен дисплеем с высокой разрешающей способностью, а также устройствами ввода в виде клавиатуры и мыши. При желании можно использовать цветной монитор. Данные, которые должны быть сохранены при отключении электропитания, записываются на жестком диске; для создания резервных копий применяется накопитель на гибких магнитных дисках. Первой операционной системой, работавшей на компьютере Ceres-1, была система Medos, реализованная на языке Modula-2. За моделью Ceres-1 вскоре последовала Ceres-2, где был использован более быстродействующий микропроцессор NS32532. Позднее была закончена и модель Ceres-3 — бездисковый вариант с процессором NS32GX32.

В конце 1985 г. Вирт и Гуткнехт начали работу над принципиально новой системой, главными свойствами которой должны были стать расширяемость и гибкость. Вирт, которого восхищала точность и надежность космического аппарата Voyager, предложил для проекта несколько неожиданное название Oberon по имени спутника планеты Уран (в то время, когда новый проект еще только обсуждался, американский космический аппарат как раз проходил вблизи этого небесного тела).

Цель предпринятых серьезных усилий по проектированию новой системы состояла в том, чтобы понять, как проектировать языки и системы с помощью концепций объектно-ориентированного программирования. В связи с этим термин "Oberon" имеет три значения.

(1) Название проекта

(2) Новый язык программирования

(3) Операционная система для персональной рабочей станции

"Решения должны быть настолько простыми,
насколько это возможно, но не излишне простыми".

А.Эйнштейн

Эту цитату Вирт предпослал своей статье "Язык программирования Oberon" [1]. Нам представляется, что она идеально отражает его подход к проектированию как языка, так и операционной системы. Вирт и Гуткнехт свидетельствуют: "Проектируя аппаратуру и программное обеспечение для системы Oberon, мы руководствовались следующим основополагающим принципом: стремиться к ясности и простоте. Это не только разумно в свете тех обстоятельств, что наш коллектив был крайне малочисленным и мы хотели получить работающую систему в сроки, диктуемые человеческим терпением. Это просто необходимо при создании любой системы, претендующей на надежность. Лучший путь для достижения ясности и простоты — это создание логичной и ориентированной на получение определенных результатов структуры. А это, в свою очередь, становится возможным, когда модель, положенная в основу проекта, хорошо понимается, когда она достаточно проста и непротиворечива.

Обсуждение языка программирования Oberon не входит в задачу этой книги (поэтому мы отсылаем читателя к монографии "Oberon Language: Steps beyond Pascal and Modula" [2]). Два приведенных ниже высказывания Вирта показывают, что сам он считает этот язык программирования развитием языка Modula-2.

> "Сначала мы планировали выразить систему на языке Modula, поскольку этот язык подерживает идею модульного проектирования вполне эффективно и с помощью тщательно отобранных средств взаимодействия. Ведь операционная система должна быть не более, чем набором базовых модулей, а разработку приложения следует рассматривать как расширение этого набора для достижения определенной цели: программирование — это всегда расширение данной системы.
>
> Если в процедурном царстве идея расширяемости поддерживается современными языками, такими, как Modula, то в сфере типов данных эта идея укоренилась несколько слабее. Так, Modula не позволяет адекватно описывать новые типы данных как расширения других, определенных программистами типов. Нам потребовались новые средства, что и привело к появлению расширения языка Modula".
>
> "Вскоре стало ясно, что правило "сосредоточиваться на существенном и игнорировать несущественное" следует применять не только при проектировании новой системы, но и (не менее жестко) при работе с языком, на котором формулируется система. Поэтому применение этого принципа привело к тому, что на смену языку Modula пришел новый язык. Впрочем, прилагательное "новый" следует понимать правильно: Oberon вырос из языка Modula, в структуру которого при этом было очень мало что добавлено, и даже кое-что изъято. Избрав эволюционный, а не революционный путь, мы сохранили верность той давней традиции, которая идет от Алгола к Паскалю, затем — к Modula-2 и, наконец, к Oberon."

Применительно к механизму расширения типов Oberon позволяет программировать в объектно-ориентированном стиле. В соответствии с идеологией проектировщика это достигается при минимуме конструкций. Здесь вообще не существует явных конструкций, классов, методов и сообщений. Это заложено уже в проекте, что подтверждается следующей цитатой из Вирта:

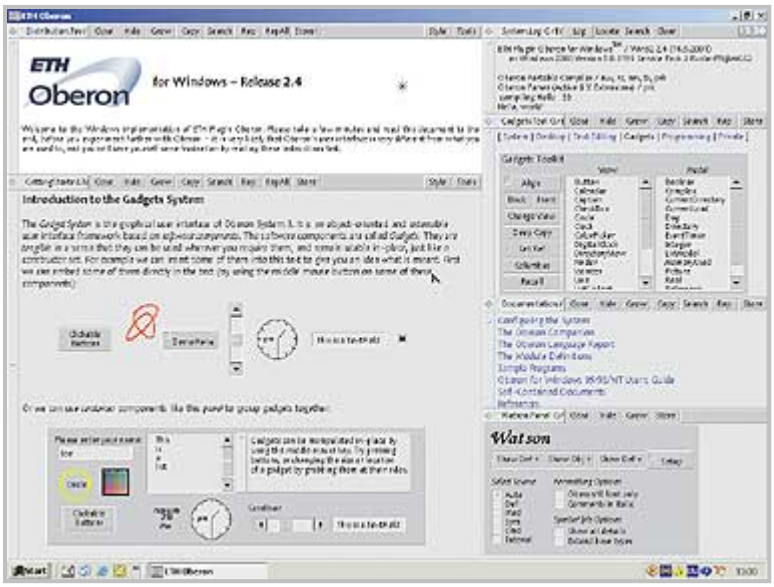
> "Перечислить все идеи, на основе которых сформировалось то, что сегодня называется Oberon'ом, просто нет возможности. Большинство этих идей родилось в ходе применения или изучения существующих языков (таких, как Modula-2, Ada, Smalltalk, Cedar), которые часто показывали нам, каких решений следует избегать" [3].

Ниже мы опишем особенности Oberon с точки зрения интерфейса пользователя, языка и системной архитектуры.

2. Пользовательский интерфейс системы Oberon

Глядя на экран типичного терминала ЭВМ или персонального компьютера, пользователь чаще всего видит перед собой строчки текста. Он уже знает, что такое курсор: точка, в которой можно вводить или удалять текст.

Однако, поработав с дисплеем, пользователь быстро понимает, что текст — это не текст. Список файлов в нашем примере создан самой системой. Это изменяемый текст, в том смысле, что его нельзя сохранять, распечатывать или редактировать. Текст можно вводить только в самой нижней строке, и тогда он представляет собой команду. Наш пользователь обнаружил, что текст модален: это или сообщение системы, или редактируемый текст, то есть команда.



В более поздних программных продуктах появились меню: команды изображаются в виде списков, и, чтобы одна из них была выполнена, на нее следует навести курсор. Однако меню отличаются от редактируемых текстов. Если для выполнения команды меню нужно ввести параметры, открывается так называемое диалоговое окно (dialog box). Вводится новый режим. Пользователь не может продолжать работу, пока он не завершит все действия в диалоговом окне.

Интерфейс системы Oberon совсем не похож на традиционные средства взаимодействия. В нем отсутствуют понятие командной строки и меню. Вместо всего этого здесь имеется текст, причем текст, обладающий такими свойствами, какие и должны у него быть в понимании всякого разумного человека, еще не испорченного так называемой "компьютерной грамотностью": этот текст допускает редактирование, вывод на печать и сохранение в памяти.

Текст есть текст: не больше и не меньше. Оставшуюся часть этой книги мы посвятим рассмотрению следствий, вытекающих из такого подхода.

2.1. Дисплей Oberon

Oberon предназначен для поддержки пользователя, работающего с дисплеем и применяющего два устройства ввода: мышь и клавиатуру. При это был сделан осознанный выбор в пользу довольно большого монитора (на станции Ceres разрешение: 1024*800 пикселов). Вот как выглядит типичный дисплей Oberon. Экран разделяется на не перекрывающие друг друга окна, которые называются визуализаторами (viewers). Визуализаторы располагаются друг над другом в два ряда, и они носят название треков (tracks). Каждый визуализатор содержит документ, обрабатываемый пользователем. В качестве документов могут выступать текст, графика, рисунки.

Изображение на дисплее может показаться обычным, но его отличает имеющая важное значение принципиальная особенность: содержащиеся в нем тексты не являются модальными. Похожие на меню надписи в заголовках окон — это тоже тексты, причем точно такие же, как и редактируемый текст в главном поле соответствующего визуализатора.

2.2. Выполнение команд

Команда просто набирается в текстовом визуализаторе. А для ее выполнения достаточно подвести к ней курсор и щелкнуть одной из клавиш мыши. Команда может быть размещена в любом месте текста. Командная строка была естественной для систем 60-х годов, в которых использовались телетайпы. Сегодня это анахронизм, который уже не облегчает работу пользователя.

Если команда выводит на экран данные (как показанная выше команда вывода каталога), то открывается новый текстовый визуализатор с теми данными, которые и порождаются командой. И здесь текст тоже можно редактировать, записывать и распечатывать. Команды языка Oberon выводят неизменяемые данные.

Сервисные программы: мостик между схемами "запомни и набери" и "наведи и щелкни". Отказ от модальности выводимого на экран текста и от командной строки позволил сделать большой шаг вперед — свести воедино два подхода к обработке команд, которые можно обозначить как "запомни и набери" и "наведи и щелкни".

Команды вводятся в визуализатор текста (с помощью стандартного редактора системы Oberon) и затем выполняются нажатием кнопки мыши. Поэтому вполне логично собрать часто используемые команды в единый текст и записать его на диск. Такой текст называется сервисной программой (tool). Сервисная программа на экране очень похожа на меню. Набор команд выводится в виде списка, и пользователь просто выполняет их с помощью мыши.

Но если при этом необходимо ввести параметры, то можно обойтись без сложного (модального) ввода или диалоговых окон. Ведь мы имеем дело с редактируемым текстом, и параметры вводятся без всяких дополнительных ухищрений, т.е. вновь обретается та гибкость, которую давала среда "запомни и набери". В сущности говоря, сервисные программы системы Oberon дают почти идеальный синтез обоих подходов, а ведь каждый из них до сих пор имеет как своих критиков, так и своих сторонников. В науке нередко случается так, что новый взгляд на вещи, способный свести воедино разные сферы человеческой мысли, открывает перед нами новые горизонты. Похоже, что именно так обстоят дела и в случае с Oberon.

Часто возникает проблема распространения релизов системы. Здесь требуются две вещи: среда распространения и последовательность команд для установки нового релиза. Надо сказать, что выполнение второй задачи — дело весьма нудное, к тому же тут трудно избежать ошибок. Так вот, в системе Oberon становится возможным запускать простой меморандум, в котором эта последовательность команд отображена во всех деталях. Для того чтобы установить релиз, нужно просто по очереди подводить курсор к каждой из команд и нажимать клавишу мыши. Ниже приводится сообщение электронной почты, которое выполняется как последовательность команд, и в результате мы получаем систему с файлового сервера.

3. Архитектура системы Oberon

В этом обзоре мы можем коснуться лишь малой части богатого набора нововведений.

3.1. Объектно-ориентированное проектирование

Идеи иерархической структуризации и абстракции данных в наши дни хорошо известны. Элемент новизны в архитектуру Oberon'а привносит механизм расширения типов.

Объекты (а говоря точнее, активные объекты) представляют собой экземпляры абстрактных типов данных, реализованных в виде записей с процедурной переменной, именуемой обработчиком объекта (handler). Параметры обработчика — это поля переменной записи, именуемой сообщением (message). Заполняя поля сообщения и запуская обработчик, мы тем самым посылаем сообщение объекту. В отличие от традиционных систем объектно-ориентированного программирования, Oberon не эмулирует концепцию класса и метода.

В системе Oberon проходит "обкатку" объектно-ориентированное конструктивное решение, которое, возможно, было бы правильнее назвать субъективным подходом (instance-centered). Связывание процедур и объектов производится не во время трансляции, а в ходе выполнения (программы). Мы говорим о встраивании обработчика внутрь объекта. В соответствии с парадигмой системы Oberon, сообщение (или блоки параметров для обработчика) описываются пользователем, а не модулем, в котором задается данный объект. Более подробное рассмотрение субъективной объектно-ориентированной системы выходит за рамки данного обзора, и мы отсылаем читателя ко второй части книги.

3.2. Модули, команды и абстрактные типы данных

Традиционные системы оснащаются средствами для выполнения программ. Когда программа загружена и запущена, она берет на себя контроль над системой. Затем программа выполняется вплоть до своего завершения (обычно это занимает долгое время), и операционная система запускает новую программу. С прекращением выполнения программы высвобождаются использовавшиеся ею ресурсы, в частности, память, которую она занимала.

В системе Oberon традиционное понятие программы отсутствует. Единица кода, которая выполняется с помощью пользовательского интерфейса, именуется командой (command). Команда — это процедура без параметров, экспортируемая модулем, который написан на языке программирования Oberon. Поскольку решающее значение при активации команд имеет эффективность, командные модули должны быть резидентными в памяти. Однако загружать все модули при загрузке системы неудобно. Поэтому подули загружаются динамически, по мере надобности. Когда модуль загружен, он остается в памяти.

В одном модуле часто реализуются один или несколько абстрактных типов данных. И тот факт, что после загрузки подули остаются резидентными в памяти, имеет одно важное следствие — экземпляры абстрактных типов данных могут теперь существовать в ходе всего сеанса работы: команды могут воздействовать на экземпляры абстрактных типов данных и взаимодействовать через них. Важным примером этого может служить абстрактный тип Text.

3.3. Интерпретатор команд, мультипрограммирование

Любая система должна располагать интерпретатором команд. Если традиционная система работает в режиме ожидания, то управление передается циклу интерпретатора системных команд, который ожидает ввода данных. Можно ввести, например, команду для загрузки интерактивного приложения. После этого управление перейдет к интерпретатору команд данного приложения, который имеет свой цикл опроса. При этом пользователь может выбирать: подавать ли системные команды или команды, которые понимает данная прикладная программа.

Понятно, что, работая с системой, где открыто несколько окон (а в каждом окне выполняется своя прикладная программа), не обойтись без использования мультипрограммирования. В известных системах для этого нужно прерывать выполнение программ и записывать на диск информацию о состоянии.

В системе Oberon этого делать не нужно. Здесь все команды принадлежат одному и тому же уровню, и каждая из них может быть выполнена по первому требованию. Для того чтобы все это стало возможным, требуется новая архитектура, обладающая следующими основными особенностями.

- Имеется только один цикл — цикл событий (event loop), который инкапсулируется в системном модуле.
- Неделимой единицей операции является вызов процедуры.

На дисплее Oberon мы видим множество соседствующих друг с другом визуализаторов (viewer), в каждом из которых выполняется та или иная задача (редактирование текста, рисование, черчение и т.д.). Визуализаторы реализованы как активные объекты.

При работе в режиме ожидания контроль над системой осуществляется циклом событий, который постоянно опрашивает драйверы устройств. Когда же поступает сигнал о событии, обработчику соответствующего визуализатора направляется сообщение, идентифицирующее данное событие. Обработчик определяет, что следует предпринять по обращении мыши или клавиатуры. Он является также менеджером дисплея, ведающим выводом данных на экран. По завершении вызова обработчика управление вновь передается циклу событий.

Важное следствие такого конструктивного решения состоит в том, что цепочка обычных вызовов процедур никогда не прерывается. Информации о состоянии, которую нужно записывать на диск при мультипрограммированиии, здесь просто нет. Не существует и скрытых состояний, и единственный процесс цикла может обслуживать одновременно несколько пользовательских задач, не вызывая осложнений, характерных для ситуации истинного мультипрограммирования.

3.4. Управление памятью, сборка мусора

В системе Oberon для отображения в памяти модулей (сегментов программы) используются диспетчеры памяти современных микропроцессоров. Модули загружаются по мере необходимости и, будучи загруженными, остаются резидентными в памяти.

Память компьютера состоит из стека для локальных переменных процедур и динамически распределяемой области (кучи). Сборка мусора (garbage collection) проводится с целью ограничить размеры этой области. Данная операция осуществляется не только из соображений удобства, но и для повышения надежности системы. Вообще говоря, если вы хотите, чтобы свободное пространство было распределено строго рационально и корректно, не стоит поручать это дело программисту — ведь он всего лишь человек.

3.5. Абстрактные документы на примере текстов

Как правило, в визуализаторах обрабатываются документы, которые отображаются внутри периметра соответствующего визуализатора. Такими документами могут быть тексты, графические изображения, рисунки и т.д.

Мы уже отмечали тот факт, что в системе Oberon текстам отводится особая роль. Текст — это абстрактный тип данных, который экспортируется модулем Texts. Он представляет собой последовательность байтов, обладающих соответствующими свойствами.

К этому следует добавить, что тексты — это активные объекты. Когда в ходе выполнения какой-либо процедуры в данных текста происходят изменения, всем видимым визуализаторам направляется сообщение с информацией об этом. Если какой-либо визуализатор содержит измененный текст, его изображение впоследствии будет откорректировано. Таким образом, операции по внесению изменений в документ и по корректировке изображения строго разделяются. Можно сказать, что текст изображает себя лишь после того, как он был изменен.

3.6. Расширяемость

Одна из важный целей, преследуемых разработчиками Oberon, — достижение расширяемости системы. Это понятие может включать в себя:

- создание команд, выполняемых из текстов и обрабатывающих соответствующие абстрактные документы (например, тексты);
- создание новых классов визуализаторов, состоящих из:
    a) абстрактных документов,
    b) визуализатора с обслуживающим его обработчиком,
    c) набора команд, выполняемых из текста, включая команду Open, которая создает экземпляр визуализатора.

Команды, обрабатывающие документы, — экземпляры абстрактного типа данных — могут легко создаваться в любое время. Запрет скрытых состояний исключает создание такими командами помех визуализатору, в котором обрабатывается данный документ. Программисту остается лишь создать объектный модуль, содержащий эту команду. Система динамической загрузки позволяет использовать данный модуль без предварительного запуска компоновщика.

Наиболее важным аспектом расширения системы Oberon является введение понятия класса визуализаторов. Архитектура системы позволяет создавать как абстрактные документы, так и новые типы визуализаторов без выполнения процедур установки или перекомпиляции системного кода. Тот факт, что все это стало возможным в среде с сильным контролем типов, представляет собой значительное достижение в области объектно-ориентированного программирования.

4. Краткие выводы

Oberon — плод исследовательского проекта. В данном обзоре мы рассматривали его наиболее важные характеристики. Далее в книге эти концепции будут обсуждены более подробно — уже на уровне системы в целом. А пока обратимся к выводам, сделанным самими создателями Oberon [4].

> "Система Oberon отличается от традиционных операционных систем в нескольких отношениях.
>
> (1) В ней отсутствует понятие программы; вместо активации программы "единицей действия", определяемой оператором компьютера, становится вызов процедуры.
>
> (2) Каждый вызов процедуры (команды) — атомарная реплика в диалоге оператора с компьютером: переключение с одной задачи на другую происходит не между двумя произвольными командами машины, а между двумя командами пользователя.
>
> (3) Данные, составляющие команду, вводятся не с клавиатуры, а из текстов и документов других видов. Вместо того чтобы выводить данные на экран, команды генерируют выходные данные в виде (изображаемых) структур данных.
>
> (4) Интерфейс между двумя последовательными действиями состоит из абстрактных структур данных (текстов, графики), которые хранятся не в файлах на диске, а в основной памяти. Когда эти структуры изображаются в визуализаторах, они допускают редактирование.
>
> (5) Oberon обеспечивает распределенную интерпретацию команд. Каждый визуализатор рассматривается как участок экрана прямоугольной формы, способный индивидуально интерпретировать команды. Для достижения этого используется парадигма объектно-ориентированного программирования. Всякий раз когда на ввод поступает обращение к визуализатору, последнему направляется сообщение.
>
> (6) Oberon оснащен простой и исключительно эффективной файловой системой. Дисковый каталог организован в виде B-дерева. Проводится четкое различие между файлом и агрегатами, с помощью которых осуществляется доступ к этому файлу; такие агрегаты называются райдерами (riders).
>
> (7) Модули в системе Oberon загружаются лишь в тех случаях, когда они фактически используются. Такая отложенная загрузка (delayed loading) играет важную роль, поскольку пакеты могут статически состоять из десятков модулей, лишь малая часть которых используется в данной конкретной прикладной программе. Отложенная загрузка управляется с помощью обращений к отсутствующей странице, для чего применяется механизм виртуальной адресации.

(8) В ядро системы Oberon встроен сборщик мусора, который, однако, не запускается как отдельный процесс. Вместо этого он явно активируется между командами при условии, что стек в данный момент пуст. Это условие значительно упрощает алгоритм и ускоряет его выполнение.

(9) Сама система и пользовательские пакеты реализуются на таком языке, который допускает расширение типов данных и полиморфные операции с гарантированной проверкой типов (type safety). Для систем, основанных на автоматическом восстановлении памяти, полная проверка типов является обязательной.

(10) Реализация системы Oberon может быть расширена (возможно, на это уйдут годы) путем описания новых типов данных, которые представляют собой расширения существующих импортированных типов данных. Объекты расширенных типов совместимы с объектами их базового типа и, следовательно, могут быть интегрированы в существующие структуры данных.

(11) В системе Oberon не существует реальных различий между пользователями и программистами. Имея в своем распоряжении мощную основу в виде множества модулей, пользователи могут расширять систему или адаптировать ее к своим потребностям путем создания новых сервисных программ".

Как быстродействие, так и объем памяти компьютерной техники неуклонно растут. А имеющиеся в нашем распоряжении программные продукты по большей части созданы уже давно и эволюционируют медленно, причем это делается так, что один уровень кода накладывается на другой, и в результате получаются системы огромных размеров. Часто возникает такая ситуация: операционная система рабочей станции занимает 1 Мбайт памяти, а текстовый процессор и программа по созданию электронных таблиц требуют 4 Мбайта, так что и на нынешнем поколении быстродействующих рабочих станций эти программы работают очень медленно.

В этой ситуации Oberon уже самим фактом своего существования доказывает, что ситуация не фатальна и что не существует некоего порога, который программное обеспечение не в состоянии преодолеть. Вся система состоит из 15 000 строк исходного текста, а сгенерированный код занимает лишь 150 Кбайт! Как бы то ни было, курс на экономию ресурсов оправдывает себя. Ведь Oberon — это сложная система, и в сравнении с хорошо известными коммерческими операционными системами она демонстрирует не меньшую, если не большую функциональность.

5. Версия системы, реализации Oberon и приложения

Работа над Oberon велась на основе той посылки, что это — открытая система, и попытки пользователя изменить ее функциональность и создавать новые функции будут только приветствоваться. Более того, можно надеяться, что с течением времени различие между пользователем и программистом будет все менее заметным. А из этого следует, что будет трудно найти две абсолютно идентичные системы.

В Институте компьютерных систем (Institut fuer Computersysteme), входящего в состав Швейцарского Федерального технологического института, идет работа над несколькими проектами, повышающими функциональность системы с помощью создания новых классов визуализаторов.

- Ряд редакторов программ, которые представляют собой расширения базового редактора Oberon, описанного в настоящем пособии.
- Два графических редактора Graph и Oil, позволяющие рисовать линии с помощью мыши.
- Редактор документов Leda, с помощью которого пользователь может вести сложную обработку текстов и форматировать страницы в режиме WYSIWYG.
- Программа рисования Paint, обрабатывающая растровые образы.

Object Oberon — это небольшое расширение языка Oberon, где вводятся заимствованные из объектно-ориентированного программирования концепции класса и сообщения. Он был разработан Ханспетером Мессенбоком (H.Moessenboeck) и Йозефом Темплом (J.Templ). Эти

авторы изменили также функциональность классов визуализаторов, в результате чего было в сущности создано новое семейство систем Oberon; со стандартным Oberon'ом его объединяют модули Oberon, а также низкоуровневые модули Oberon'a.

Несколько лет система Oberon работала на машине Ceres, экспериментальной рабочей станции, созданной в Институте компьютерных систем. В продаже имеются реализации для компьютеров SUN Sparcstation и Apple Macintosh. В большей или меньшей степени близки к завершению работы по подготовке реализаций для других машин — DECsystem 3100, IBM PS/2 и IBM RISC System/6000.

Литература

[1] Wirth N. (1988) The Programming Language Oberon // Software: Practice and Experience, Vol.18, No.7, p.671-690.

[2] Reiser M., Wirth N. (1992) Programming in Oberon: Steps Beyond Pascal and Modula-2 // ACM Press.

[3] Wirth N. (1988) From Modula to Oberon // Software: Practice and Experience, Vol.18, No.7, p.661-670.

[4] Wirth N., Gutknecht J. (1989) The Oberon System // Software: Practice and Experience, Vol.19, No.9, p.857-893.

Об авторе. Мартин Рейзер (Martin Reiser) — ведущий специалист IBM Research Division и одновременно руководитель IBM Zurich Research Laboratory. Он является лауреатом многих премий и пользуется международным признанием, прежде всего, за свои исследования в области оценки производительности компьютерных систем. Профессор Рейзер был удостоен нескольких премий (Outstanding Innovation awards) от фирмы IBM. Он член IEEE, лауреат IFIP Silver Core и премии IEEE Koji Kobayashi Computers and Communications Award. Начиная с 1989 года тесно сотрудничал с Виртом и Гуткнехтом в ETH Zurich в рамках проекта Oberon. Результатом работы послужила книга "The Oberon System: User Guide and Programmer's Manual", которая является замечательным руководством для программистов, проектировщиков, студентов и исследователей.