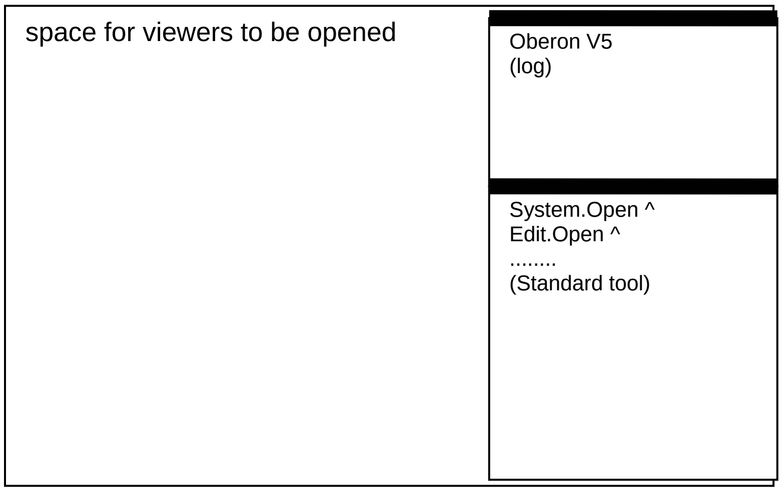
How to use the Oberon System

Niklaus Wirth, November 2015

The Oberon System was designed and implemented in 1990 as an example of a modern, interactive operating system with windows and mouse. In contrast to commercial systems it was to be simple and well-structured in order to be published in its entirety and available for teaching the essentials. It was programmed in the concise language Oberon, and it requires only a small fraction of the resources which commercial systems demand..

How to begin

After start-up (power on) the following layout appears on the display.



On the right side, there are two windows (called *viewers*) vertically arranged. The top viewer shows a log, in which the system records the user's actions. (It displays the text *Oberon.Log*). The lower viewer is the standard command tool, containing various general commands. (We call texts containing commands *tools*). Commands in Oberon have the form *M.P*, where *M* is the name of the module containing the procedure *P*. Commands may occur in any text, and they are activated by pointing at them and clicking the middle mouse button (the *command button*).

On the left side appears an empty space where new viewers may be placed. If, for example, we wish to display a text (file) named *Sample.Mod* . We select the name by pointing at it and clicking the right mouse button (*the select button*) and then activate the command *Edit.Open* in the standard tool viewer. If the name does not appear somewhere on the screen already, type the name (see editing texts below) to enter it. Now a new viewer containing the text is opened in the area to the left. If a text (file) with the specified name does not exist, an empty viewer appears.

The viewers now visible are *text viewers*. They consist of two parts, called *frames*. The narrow one on top is called the *title bar*. It contains the name of the displayed text and several commands for viewer handling and text editing, among them

System.Close	close the viewer
Edit.Store	store the text as a file
Edit.Search	search for the selected word from the position of the caret

Editing texts

The most frequent task is editing a text. For this case, some commands can be issued by simple mouse clicks alone rather than clicking the command name.

When a text is to be entered, it is always *inserted* at the position of the *caret* displayed as a hook. The caret is positioned by pointing (with the cursor) at the desired position and clicking the left button (pointer button). Then the text is entered by simply typing it.

A piece of a text can be *selecte*d in order to be subjected to a subsequent command. This is done by moving the cursor over the piece of text while holding the right button (select button) pressed. Often it is only necessary to select the first character of a parameter.

Selection can be turned into *deletion* by briefly clicking also the left button while holding the right button pressed. This is called *inter-clicking*. Similarly, a piece of text being selected can be copied-over to the caret position by inter-clicking the middle button.

The functions of the mouse buttons are summarized by the following table:

inter-click:		left	middle	right
primary click:				
left	**set caret**	-	copy looks	copy
middle	**command**	-	-	-
right	**select**	delete	copy	-

The keyboard features two special keys, so-called *control keys*. The *esc* key undoes selections. The *backspace* key deletes the character left of the caret.

Text viewers have a *scroll bar* at their left edge. While the cursor is in the scroll bar, it assumes the form of an up/down pointer in order to show its difference in functionality. Clicking a button then causes the text to scroll up or down.

left button	the current line moves up to the top of the viewer
middle button	show the section at the relative position given by the cursor
right button	the top line moves down to the current cursor position

Viewer handling

Viewers can be extended, moved, or copied. A viewer is enlarged or shrunk by clicking the left button, while the cursor is in the title bar, and then dragging the bar up or down. A viewer is moved to another location by also inter-clicking with the middle button.

A duplicate of a viewer may be generated by activating the command *System.Copy* in the title bar. Note that in this case the old and the new viewer show the same text, and not a copy of the text. This facility comes in handy, when a piece of text needs to be moved or copied into a position not visible in the same viewer (long distance move, or long-distance copy).

The command *System*.*grow* in the title bar generates a copy extending over the entire column (or over the entire display). By closing the viewer (command *System.close* in the title bar), the original viewer reappears. We may imagine that *grow* lifts a viewer to an overlay in the third dimension.

Commands

The following commands appear in the standard tool *System.Tool*. The character ^ indicates, that a name must be selected previously. The command then applies to that named text (file, as in the example above). The character ~ indicates that a list of names must be inserted before the ~ character, and the command will apply to all named objects.

System.Open ^	open viewer in the system track to the right
System.Recall	close the last viewer opened
Edit.Open ^	open viewer in the user track to the left
Edit.Recall	undo last editing operation
Edit.ChangeFont	applies to selected piece of text
Edit.SetFont	use specified font for subsequent input
System. Directory ^	search directory for specified file names
System.Free ~	remove specified modules from store
System.CopyFiles => ~	e.g. file1 => file2 file3 => file4~
System.RenameFiles => ~	e.g. file1 => file2 file3 => file4~
System.DeleteFiles ~	e.g. file1 file2 file3~ (from file directory)
System.ShowModules	
System.ShowCommands ^	
ORP.Compile @	compile selected text
Hilbert. Draw	draw Hilbert curve, as an example

When clicking a command *M.P*, module *M* is searched in the file store and, if found and not already present, loaded into main store. Then its command *P* is searched and executed (if found). A list of loaded modules can be generated by the command *System.ShowModules*, and a list of its commands is obtained by the command *System.ShowCommands*. Any parameter-less procedure in any (compiled) module is accessible as a command. Its parameters are accessed via a *scanner*. As an example, consider the following short program:

```
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.
```

After its successful compilation (ORP.Compile @), the command

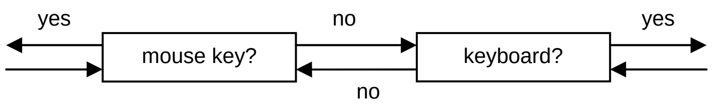
```
M0.P0   2 3 5 7 11~
```

causes these numbers and their sum to be output to the standard viewer *System.Log*.

The core of the Oberon System

The system's core consists of a loop which continuously senses for a command. to appear. The command is identified, control is dispatched, and the command is executed. A command may stem from an explicit click (middle button) on a text of the form *M.P*, or it may be a click of the left or right mouse buttons (see editing commands).

A further source of input is the keyboard. If any key is pressed, this is interpreted as a command to read that character. Exceptions are the esc, ctrl-z (or F1), and backspace keys. *esc* is interpreted as a command to undo all selections, *backspace* to remove the character left of the caret and ctrl-z to set the global marker.



The initially loaded system contains, apart from module Oberon, the command module System,a text system (modules TextFrames, MenuViewers, Texts, Fonts, Input), a viewer system (modules Viewers, Display), the loader and linker (module Modules), a file system (modules Files, FileDir), and the disk space manager and the garbage collector (module Kernel). The compiler is loaded on demand, like other application programs.

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

http://www.projectoberon.com/