dL4 Driver "POSIX.TTY Window

(14k) 5pgs : dL4 Driver "POSIX.TTY Window"
Abstract
This document contains implementation notes and details for the dL4 driver "POSIX TTY Window". For background, readers should refer to dL4 Window Class Document for a general discussion on Windows in dL4.

IMPLEMENTATION DETAILS
1.1. Two Different Kinds Of Windows
The POSIX TTY Window driver makes a distinction between two different kinds of windows.
1. The "screen" window.
2. All other windows.
The screen window must be the first window opened by the driver, and can only be opened using a special calling sequence. This open is in fact performed automatically by both SCOPE and RUN as a consequence of opening the "Default Window" driver (described below).
The screen window serves as the root window and ultimate parent window to all other windows. When a BASIC program requests an "independent" window, the screen window actually ends up being assigned as the parent.
In the remainder of this document, we will refer to all other non-screen windows simply as "windows".
1.2. The Screen Window
The screen window is intended to represent the actual terminal.
Its capabilities and behavior as a "window" are thus necessarily different.
The user is granted much more direct or "raw" access to the screen window than is allowed or defined by the generic "Window" class. In fact, the only channel operations that are allowed on the screen window are:
READ
WRITE
ERASE
Other window functions like SIZE, MOVE, HIDEWINDOW, etc. are not allowed.

When performing I/O with a screen window, the window driver does not impose any pre-defined limits upon which characters are valid, and makes few assumptions about the actual effect of I/O with the screen. So, for example, a program may manually output something like "\33\`1a" to the screen and the driver will make no assumptions about what actually happened at the terminal.
This allows code which e.g. programs a terminal's status line to continue to work, although only when conversing directly with the screen window.
1.3. The "Default Window" driver
Both SCOPE and RUN, upon startup, effectively perform the following statement:
Open #C,{"dL4","TITL,WRAP,SCRL",80,25} As "Default Window"
The channel #C is later provided to the BASIC interpreter to be used as the default input and output channel, as well as used by SCOPE to facilitate his command-line conversation with the user.
The overall intent of the Default Window driver is to setup the default INPUT and PRINT connection expected by all existing IRIS and dL4 software; namely, the connection to something that resembles a "terminal".
The Default Window driver as implemented on POSIX simply acts as a switch and makes a choice between one of two drivers:
1. "POSIX TTY Window"
2. "Phantom Window"

If the process' standard in and out are open (file descriptors 0 and 1) then the POSIX TTY Window driver is selected. It does not matter the type of file to which these file descriptors refer; even if directed to a regular file or pipe, the TTY window driver will be used. When the POSIX TTY driver detects the initial open coming from the Default Window driver, it establishes that channel as the screen window.
If 0 and 1 are both closed, the process is assumed to be a phantom port and the Phantom Window driver is selected. In fact, dL4 creates a phantom port by three simple steps; fork() a new process, close files 0 and 1, and exec() SCOPE.

1.4 Tracking the Screen Window

The canvas contents of the screen window are not actually recorded by the window driver. Therefore, portions of the screen window obscured by a child window will not be redrawn when the window is closed. The driver clears the screen upon creation of the first window in an attempt to address this problem. The screen window remains accessible, even when other windows are present. Any I/O that occurs with the screen window when other windows are visible may produce unexpected results. The assumption is that programmers will either use the screen window or actual windows, but not both simultaneously.

1.5 The Terminal Description File

In order to function, the POSIX TTY Window driver must locate a "terminal description file". This file describes the capabilities of the physical terminal, represented primarily in terms of how to input and output all the Unicode characters which the terminal is supposed to "support" within dL4.
The layout of and syntax used within a terminal description file constitutes a language of its own, and a full description is beyond the scope of this document. There is currently a document available from DCI simply entitled "Terminal Description Files" which fully explains the format of these files.
The name of the terminal description file is taken directly from the TERM environment variable, and the following directories are searched, in order:
Directory given in TERMDIR variable, if present
Current working directory
Directory given in HOME variable, if present
The use of TERMDIR is the most practical for long-term setup, by establishing a directory to serve as a central "library" of dL4 terminal descriptions.

2. MISCELLANEOUS IMPLEMENTATION NOTES

2.1. Required Characters

The following characters must be defined in the terminal description file in order for the window driver to create windows:
'CS' Clear screen
'x,yMOVETO' Absolute cursor position
'GH' Horizontal line
'GV' Vertical line
'G1' Upper-left corner
'G2' Upper-right corner
'G3' Lower-left corner
'G4' Lower-right corner
" " Space

2.2. Supported Text Drawing Modes

The following text drawing modes are supported in windows, subject to their support by the terminal itself:

Blink
Bold
Dim
Italic
Reverse video
Strike-out
Underline
Protected
Foreground color ('BL','CY','GN','MA','RE','WH','YE', or '#FONTCOLOR')
Background color ('#BACKCOLOR')

2.3. Supported I/O Modes

The following I/O modes are NOT currently supported by the POSIX TTY window driver:

Echo "\" on escape mode

2.4. I/O Special Output Characters In the Screen Window

The following output characters are supported differently when output to the screen window. Normally, characters output to the screen window are simply processed according to the [OutputMacros] section of the terminal description file. The characters documented below are handled internally by the driver.

The following characters are processed separately by the driver:

'IOBE' Begin input echo mode.
'IOEE' End input echo mode.
'IOTE' Toggle input echo mode.
'IOBD' Enable destructive backspace mode.
'IOED' Disable destructive backspace mode.
'IOBC' Enable activate-on-control-character mode.
'IOEC' Disable activate-on-control-character mode.
'IOBI' Enable binary input mode.

'IOEI' Disable binary input mode.
'IOBO' Enable binary output mode.
'IOBX' Enable XON/XOFF input flow control mode.
'IOEX' Disable XON/XOFF input flow control mode.
'IOCI' Clear the input type-ahead buffer.
'IORS' Reset all I/O modes to default state.

Revision date: 12/03/97 DCI Engineering