UOS Programmer's Manual

Table of contents

Title Page

UOS Programmer's Manual

October 2023

Preface

Preface

This document is intended for people writing programs to run on UOS.

Overview

Overview

UOS is a flexible, general-purpose, multi-user system that supports high integrity and dependability along with the benefits of being open source.

UOS is designed to provide software compatibility across all the processors on which it runs. Different hardware architectures may make binary compatibility impossible, but compatible compilers should compile code on all platforms without logic changes.

UOS is backward compatible with earlier versions of UOS so that programs that run on one version will run on all future versions, including UCL scripts.

UOS is programming language-agnostic; any compiler/interpreter should be able to use the UOS services without difficulty.

Run-Time Libraries

UOS provides several language-independent procedures and services for programs. These procedures adhere to the UOS calling standard and include the following sets of Run-Time Library routines:

- LIB. General library routines, including common I/O procedures.
- DMG. Device-independent display management.procedures
- FS. File System File support procedures
- Hash. Hashing and encryption procedures.
- LBR. Librarian procedures.

UOS system services are provided by the UOS executive and perform operations such as file access, process management, symbol management, and device I/O. At run time, the calling program passes control of the process to the system service, which performs the requested operation and then returns control to the program.

User Interface

User Interface

There are two possible user interfaces for UOS: the console interface and the Graphical User Interface. The differences between these can be somewhat mitigated by using the Universal User Interface (UUI).

UUI

UUI

The UUI is a facility used by most UOS system programs which provides a consistent way to easily create a command line/graphical user interface. UUI is controlled by a text definition that the program passes to UUI. In a graphical environment, the definition is used to create the graphical elements of the UI.

TUUI Class

TUUI Class

The UUI is implemented via the TUUI class, which has the following methods:

Clear
procedure Clear

This procedure clears the user interface.

Execute
function Execute(Name : PChar = nil) : boolean
bool Execute(char* Name)

This procedure stars the user interface, using the command line to fill the values of the components. It exits upon completion of parsing the command line. If no command line is provided, it will exit when the user requests an exit or if an error occurs. If it exits due to an error, the result will be false, otherwise it will be true.

If there is only one input component, the method returns as soon as that item is provided by either the user or the command line. If there is no command line, the user will be prompted. The way the prompt is derived is as follows.

If the input component is contained within a parent component, a label component is searched for within the parent. Otherwise, all components are searched. The prompt is derived from the only label component found in the search. If no label component (or multiple label components) is defined, the prompt will be taken from the $prompt variable, if defined. The prompt is prefixed with an underscore (_) when displayed to the user.

If a non-null name is passed to the routine, only the component with that name is executed.

Get_Component_By_Name
function Get_Component_By_Name(X : PChar) : TUUI_Component TUUI_Component Get_Component_By_Name(char* X)

Returns an instance of a UUI component whose name matches the passed name. Nil is returned if the component wasn't found.

Definition
property Definition : PChar

Returns/sets the UI definition. Error is set if there was an error compiling the definition. Setting the definition clears any previous UI definition.

Error
function Error : PChar
char* Error()

Returns the text of the last error. It returns null if there is no error.

Error_Line
function Error_Line : PChar
char* Error_Line()

Returns the text of the source line that causes the last error. The return value is undefined if there was no error.

Get_Variable
function Get_Variable(N : PChar) : PChar
char* Get_Variable(char* N)

Returns the value of the passed variable. Null is returned if the variable isn't defined. The only default variable is $margin, which defaults to 8.

Set_Variable
procedure Set_Variable(N, V : string)
void Set_Variable(char* N, char* V)

Sets the value of the variable named by N to the value specified by V. If the variable isn't defined, it is created.

TUUI_Component Class

TUUI_Component class

TUUI_Component encapsulates a UUI component. These are created as a result of setting the UI Definition in a TUUI instance. Components can be broadly categorized as either input components or display components.

Input components
Input components are those that are directly changed by interaction of the user or via the command line.

TUUI_Action	represents an operation that can be performed.
TUUI_Boolean	represents a boolean value.
TUUI_Integer	represents an integer value provided by the user.
TUUI_List	represents an text value that is one of a list of valid values.
TUUI_String	represents a single line of text provided by the user.

Display components
Display components don't usually have direct user interaction and are used for layout and/or informational purposes.

TUUI_Label represents static text. This usually doesn't show in a command line interface, unless a prompt is needed.

TUUI_Rectangle represents a rectangular area. This is ignored in a command line interface.

All TUUI_Components have the following properties:

Property	Value	Description
False_Hint	PChar	Hint for a boolean component when it is not selected
Flags	Cardinal	Flags
Left	Integer	The left pixel of the component
Height	Integer	The height of the component
Hint	PChar	Hint for the component
Inverse	PChar	Name of the inverse component
List	TStringList	Valid items for the component
Maximum	Integer	Maximum valid value
Minimum	Integer	Minimum valid value
Name	PChar	Name of the component
Parent	TUUI_Component	The parent (containing) object of this component
Selected	Boolean	True if boolean item is selected
Text	PChar	Text value of the component
Top	Integer	The top pixel of the component
True_Hint	PChar	Hint for a boolean component when it is selected
UUI	Boolean	True if the current value/state of the component was set by UUI from the command line when it was executed.
Width	Integer	The width of the component

UUI Definitions

The UUI UI definition is a text source consisting of one item per line. Blank lines are ignored and leading and trailing spaces are ignored. A line ends with a combination of one or more CR (ASCII 13) and/or LF (ASCII 10) characters.

The definition source defines one or more components, each of which contain various attributes. Undefined attributes are assigned a default value. A component can be nested within any other component.

A component definition starts with a line of the following format:

object name:type

where "name" is the component name and "type" is the component type. Component names must contain only alphanumeric and dollar signs ($) or underscores (_). Component names are case-insensitive and must be unique within the UI definition. The component type can be one of the following:

Type	Category	Class	Description
boolean	Input	TUUI_Boolean	A true/false item (such as a checkbox).
integer	Input	TUUI_Integer	An integer value.
label	Display	TUUI_Label	Static text.
list	Input	TUUI_List	A list of possible values.
operation	Input	TUUI_Action	An operation (such as a button or menu item).
rectangle	Display	TUUI_Rectangle	Rectangular area.

string Input TUUI_String single line of text.
A component definition ends with a line consisting of:
end

Between the object and end lines, other objects can be defined and/or attributes set. Although examples are shown with indentation, the indentation is entirely to make the definition easier to read - it is ignored by UUI.

Attribute lines have the format:

attributename = value

where "attributename" is the name of one of the valid component attributes and "value" is the value for that attribute. If the value is text, it must be delimited with single or double quotes. If the value is numeric, it can be a number or an expression. Note that case is ignored except within quotes. The following list of attributes are common to all components. Attributes with the "GUI only" note indicate that the attribute only has an effect in a GUI interface.

Attribute	**Value type**	**Applies to**	**Description**
allownull	boolean	strings	True to allow the text to be null. Defaults to true. Only applies to text/string input components - ignored on all others.
falsehint	text	booleans	Popup hint to show when a boolean component is not selected. GUI only.
hint	text	all	Popup hint to show. Defaults to null. GUI only.
inverse	component name	booleans	Name of boolean component that represents the opposite of this component. Only applies to boolean components.
left	integer	all	Pixel position of left side of component relative to its parent's left side. Defaults to 0. GUI only.`
list	list	lists	A list of items, delimited by commas.
maximum	integer	integers	Maximum value for an integer component. Has no meaning for other types of components.
minimum	integer	integers	Minimum value for an integer component. Has no meaning for other types of components.
selected	boolean	booleans	True if item is selected/checked/active/set. Only has meaning for boolean components.
text	text	all	Text of component. Defaults to null.
top	integer	all	Pixel position of top side of component relative to its parent's top. Defaults to 0 GUI only.
truehint	text	booleans	Popup hint to show when a boolean component is selected. GUI only.
type	text	lists	Indicates options for the component. For list components, this can be "constrained" to mean that the component's value must be one of the items in the list.
visible	boolean	all	True for component to be visible. Default is true. GUI only.
width	integer	all	Width of component, in pixels. Defaults to the actual width of the text or image. GUI only.

Boolean values
Boolean values are either "True" or "False".

Integer values
Integer values must be numeric. If they contain fractional parts, they will be truncated to an integer value. The value may be an expression that evaluates to a number. See the appendix on Numeric Expressions for details.

List values

List values are a series of values, separated by commas. Any string values must be included within quotes.

Text values
Text values must be delimited by either single (') or double (") quotes.

Here is an example UUI defintion:

```
object Main:rectangle
  top = 2
  object label:label
    text = "Name:"
    top = $margin
    left = $margin
  end
  object name:string
    hint = "Name"
    top = $margin
    left = label.width + $margin
    allownull = false
  end
end
```

Note: the spaces around the equal signs are not required.

Numeric Expressions

The basic format of a numeric expression is:

term operator term

where "term" is a number, a variable, a function, a constant, or two terms separated by an operator. "operator" is a arithmetic operator. The following arithmetic operators are available:

Operator	Description
!	Factorial (do a factorial of the previous value)
%	Percentage (divide the previous value by 100)
*	Multiply
/	Divide
^	Exponentiation
+	Addition (or unary plus)
-	Subtraction (or unary minus)
#	Round to the specified number of digits. If rounded to a negative number, it is rounded to the right of the decimal point.
D	Random Gaussian distribution. For instance, 2D10 will return a random integer between 1 and 10, plus a random integer between 1 and 10 (ie a value between 2 and 20).
MOD	Division remainder only
MIN	Return left side value if greater than the right side value, otherwise, return the right side value
MAX	Return left side value if less than the right side value, otherwise, return the right side value

Note that all operators operate on two values except for unary plus, unary minus, percent, and factorial.

The following integer bit operators are available:

Operator	Description

NOT	1's-complement inversion.
AND	Bitwise AND
OR	Bitwise OR
XOR	Bitwise exclusive-OR
NAND	Bitwise NOT AND
NOR	Bitwise NOT OR
XNOR	Bitwise NOT XNOR

Constants
PI is a predefined constant equal to 3.14159265...

Functions
Functions have the form:

FUNCTION(expression)

where expression is the value to perform the function on. Available trigonometric functions are:

Function	Description
TAN	Tangent
SIN	Sine
COS	Cosine
SEC	Secant
COT	Cotangent
CSC	Co-secant

ARC and Hyperbolic functions are available by prefixing the trigonometric function name with "ARC", "HYP", or "HYPARC". For instance, "HYPARCSIN".

Other functions available:

Function	Description
ABS	Absolute value
DEG	Radian to degree conversion
EXP	Constant E raised to the specified power
FIB	Fibonacci sequence
INT	Integer
LOG	Natural logarithm
LOG10	Common logarithm
RAD	Degree to radian conversion
SGN	Sign (returns -1 if negative, 1 if positive, or 0 if 0)
SQR	Square root

Variables
A variable is either a defined variable or a component and attribute pair of the form:

name.attribute

where "name" is the name of a component, and "attribute" is the attribute of that component. Note that this

attribute must be numeric or the expression is invalid. For example:

Main.top

Variables are defined via the TUUI.Set_Variable method. The only pre-defined variable is "$margin", which indicates the current margin (default is 8).