

U2 Family Application Development - Test 713

Handout for SAPUG October 16th 2003 IBM U2 Meeting

Ross Morrissey <http://uv.rossmorrissey.com>

This is a set of study notes put together from sparse information on the Internet and cut and pasted from online UV documentation. Note that these notes were assembled before I took the test. YMMV.

The Exam

To attain the IBM Certified Solutions Expert - U2 Family Application Development certification, candidates must pass 1 test. "There are a total of 79 questions. To pass this exam you must score 62% or greater. You will have 90 minutes to take this test."

Recommended Prerequisites

Attend L1-UD 100 UniData Fundamentals or L1-UV 900 UniVerse Fundamentals and L1-UD 950 UniData ODBC or L1-UV 911 UniVerse ODBC.

Attend L1-UD310 Structured UniBasic: The Programming Commands or L1-UV903 UniVerse Database Programming training course.

Six months experience, reinforcing the skills learned by designing and writing application programs. Additional experience and skills related to database functions such as account and file creation and maintenance is helpful.

Skill in implementing application software across multiple platforms.

Knowledge of fundamental database functions and facilities.

A working knowledge of the design implications of the U2 APIs.

Section 1 - Application Design (13%) - 10 Questions

Create hashed files

CREATE.FILE DATA name type mod sep

Define file dictionaries

D, I, PH

Define file indices

CREATE.INDEX name itype NO.NULLS

Integrate application tools (PROCS/Paragraphs, UniQuery/Retrieve, SQL)

PA - DATA statement

PQ

- ✓ PROCREAD Assigns the contents of the primary input buffer of the proc to a variable.
- ✓ PROCWRITE Writes the specified string to the primary input buffer of the proc that called your BASIC program.

EVAL "itype expression"

Define API

All accessed through **UniRPC**

UCI (Uni Call Interface) is the C-language API. It lets developers write UNIX and Windows client programs that use SQL statements to access and manipulate data in U2 databases. UCI is modelled on ODBC [RETICODE SQLConnect (hdbc, szDSN, cbDSN, szSchema, cbSchema)]

UniOLEDB is the U2 OLEDB provider.

InterCall is an open API that lets client application programs developed on UNIX or Windows systems access data on UniVerse or UniData servers. [ic_readnext (select_list_num, record_id, max_id_size, id_len, code)]

UniObjects is an API to UniVerse from Visual Basic, or from any other program development environment that uses the Microsoft ActiveX interface. [String = SessObj.GetAtVariable(@variableID)]

UniObjects for Java is an API that lets developers create Java-based applications that access U2 databases. [public UniString (UniSession aSession)]

Use external subroutines

The **GCI** (General Calling Interface) acts as a gateway from UniVerse BASIC to an external subroutine.

The GCI passes data to and from subroutines via arguments and arrays.

The GCI allows BASIC programs to make:

- ✓ Calls to external subroutines written in FORTRAN 77, C, and C++.1
- ✓ System calls to operating system commands.

BCI - BASIC SQL Client Interface is an (API) that makes UniVerse a client in a client/server environment. [STATUS = SQLExecDirect(STMTENV, DTBL1)]

Section 2 - Logic Constructs (15%) - 12 Questions

Control program flow

ABORT Terminates all programs and returns to the UniVerse command level.

BEGIN CASE Indicates the beginning of a set of **CASE** statements.

CALL Executes an external subroutine.

CASE Alters program flow based on the results returned by expressions.

CHAIN Terminates a BASIC program and executes a UniVerse command.

CONTINUE Transfers control to the next logical iteration of a loop.

END Indicates the end of a program or a block of statements.

END CASE Indicates the end of a set of CASE statements.

ENTER Executes an external subroutine.

EXIT Quits execution of a **LOOP...REPEAT** loop and branches to the statement following the **REPEAT** statement.

FOR Allows a series of instructions to be performed in a loop a given number of times.

GOSUB Branches to and returns from an internal subroutine.

GOTO Branches unconditionally to a specified statement within the program or subroutine.

IF Determines program flow based on the evaluation of an expression.

LOOP Repeatedly executes a sequence of statements under specified conditions.

NEXT Defines the end of a **FOR...NEXT** loop.

ON Transfers program control to a specified internal subroutine or to a specified statement, under specified conditions.

REPEAT Repeatedly executes a sequence of statements under specified conditions.

RETURN Transfers program control from an internal or external subroutine back to the calling program.

RETURN (value) Returns a value from a user-written function.

STOP Terminates the current program.

SUBR() Returns the value of an external subroutine.

WHILE/UNTIL Provides conditions under which the **LOOP...REPEAT** statement or **FOR...NEXT** statement terminates.

Control file access

Process Lock: LOCK semaphore THEN statement ELSE statement

AUTHORIZATION "username" - specify or change the effective run-time user of a program. After an AUTHORIZATION statement is executed, any SQL security checking acts as if username is running the program.

Employ logical operators

AND (or the equivalent &)

OR (or the equivalent !)

NOT

Employ arithmetic operators

-	Negation	-X
^	Exponentiation	X ^ Y
**	Exponentiation	X ** Y
*	Multiplication	X * Y
/	Division	X / Y
+	Addition	X + Y
-	Subtraction	X - Y

Use internal subroutines and user-defined functions

GOSUB label

DEFFUN functionname (a)

FUNCTION Identifies a user-written function.

Section 3 - User Interface (10%) - 8 Questions

Define character based screen layout

@() Returns an escape sequence used for terminal control.

@(column [,row]) @(-code [,arg])

Code	Action
------	--------

-1	Screen clear and home
----	-----------------------

-2	Cursor home
----	-------------

-3	Clear to end of screen
----	------------------------

-4	Clear to end of line
----	----------------------

CRT Outputs data to the screen.

Use the CRT statement to print data on the screen, regardless of whether a PRINTER ON statement has been executed. The syntax for print.list is the same as for PRINT statements.

DATA Stores values to be used in subsequent requests for data input.

DISPLAY Outputs data to the screen.

FOOTING Specifies text to be printed at the bottom of each page.

HEADING Specifies text to be printed at the top of each page.

HUSH Suppresses all text normally sent to a terminal during processing.

PRINT Outputs data to the terminal screen or to a printer.

PRINTERR Prints a formatted error message from the ERRMSG file on the bottom line of the terminal.

TPRINT Sends data with delays to the screen, a line printer, or another specified print file (that is, a logical printer).

TPRINT [ON print.channel] [print.list]

print.list can also contain time delays of the form \$ <time>. time is specified in milliseconds to the tenth of a millisecond. As the print list is processed, each time delay is executed as it is encountered.

Accept and format user input

BREAK Enables or disables the Break key on the keyboard.
CLEARDATA Clears all data previously stored by the DATA statement.
ECHO Controls the display of input characters on the terminal screen.
INPUT Allows data input from the keyboard during program execution.
INPUT @ Positions the cursor at a specified location and defines the length of the input field.
INPUTCLEAR Clears the type-ahead buffer.
INPUTDISP @ Positions the cursor at a specified location and defines a format for the variable to print.
INPUTERR Prints a formatted error message from the ERRMSG file on the bottom line of the terminal.
INPUTNULL Defines a single character to be recognized as the empty string in an INPUT @ statement.
INPUTTRAP Branches to a program label or subroutine on a TRAP key.
KEYEDIT Assigns specific editing functions to the keys on the keyboard to be used with the INPUT statement.
KEYEXIT Specifies exit traps for the keys assigned editing functions by the KEYEDIT statement.
KEYIN() Reads a single character from the input buffer and returns it.
KEYTRAP Specifies traps for the keys assigned specific functions by the KEYEDIT statement.
PROMPT Defines the prompt character for user input.

Validate user input

ICONV(string,conversion)

Output reports

FOOTING Specifies text to be printed at the bottom of each page.
HEADING Specifies text to be printed at the top of each page.
OPENDEV Opens a device for input or output.
PAGE Prints a footing at the bottom of the page, advances to the next page, and prints a heading at the top.
PRINT Outputs data to the terminal screen or to a printer.
PRINTER CLOSE Indicates the completion of a print file and readiness for the data stored in the system buffer to be printed on the line printer.
PRINTER ON | OFF Indicates whether print file 0 is to output to the terminal screen or to the line printer.
PRINTER RESET Resets the printing options.
TABSTOP Sets the current tab stop width for PRINT statements.
TPRINT Sends data with delays to the screen, a line printer, or another specified print file (that is, a logical printer).

Section 4 - File Handling (18%) - 14 Questions

Open, close, clear, delete files

CLEARFILE Erases all records from a file.
CLOSE Writes data written to the file physically on the disk and releases any file or update locks.
INDICES() Returns information about the secondary key indexes in a file.
OPEN Opens a UniVerse file to be used in a BASIC program.
OPENPATH Opens a file to be used in a BASIC program.
Sequential File I/O
CLOSESEQ Writes an end-of-file mark at the current location in the record and then makes the record available to other users.
CREATE Creates a record in a UniVerse type 1 or type 19 file or establishes a path.
OPENSEQ Prepares a UniVerse file for sequential use by the BASIC program.
STATUS Determines the status of a UniVerse file open for sequential processing.

Select data from files

CLEARSELECT Sets a select list to empty.

DELETELIST Deletes a select list saved in the &SAVEDLISTS& file.
GETLIST Activates a saved select list so it can be used by a READNEXT statement.
READLIST Assigns an active select list to a variable.
READNEXT Assigns the next record ID from an active select list to a variable.
SELECT Creates a list of all record IDs in a UniVerse file for use by a subsequent READNEXT statement.
SELECTE Assigns the contents of select list 0 to a variable.
SELECTINDEX Creates select lists from secondary key indexes.
SELECTINFO() Returns the activity status of a select list.
SSELECT Creates a sorted list of all record IDs from a UniVerse file.
WRITELIST Saves a list as a record in the &SAVEDLISTS& file.

Read data from files

BSCAN Scans the leaf-nodes of a B-tree file (type 25) or a secondary index.
READ Assigns the contents of a record to a dynamic array variable.
READL Sets a shared read lock on a record, then assigns the contents of the record to a dynamic array variable.
READU Sets an exclusive update lock on a record, then assigns the contents of the record to a dynamic array variable.
READV Assigns the contents of a field of a record to a dynamic array variable.
READVL Sets a shared read lock on a record, then assigns the contents of a field of a record to a dynamic array variable.
READVU Sets an exclusive update lock on a record, then assigns the contents of a field of the record to a dynamic array variable.
TRANS() Returns the contents of a field in a record of a UniVerse file.
XLATE() Returns the contents of a field in a record of a UniVerse file.

Sequential File I/O

READBLK Reads a block of data from a UniVerse file open for sequential processing and assigns it to a variable.
READSEQ Reads a line of data from a UniVerse file opened for sequential processing and assigns it to a variable.
TIMEOUT Terminates READSEQ or READBLK if no data is read in the specified time.

Write data to files

DELETE Deletes a record from a UniVerse file.
DELETEU Deletes a record from a previously opened file.
WRITE Replaces the contents of a record in a UniVerse file.
WRITEU Replaces the contents of the record in a UniVerse file without releasing the record lock.
WRITEV Replaces the contents of a field of a record in a UniVerse file.
WRITEVU Replaces the contents of a field in the record without releasing the record lock.

Sequential File I/O

FLUSH Immediately writes all buffers.
NOBUF Turns off buffering for a sequential file.
SEND Writes a block of data to a device that has been opened for I/O using OPENDEV or OPENSEQ.
WEOFSEQ Writes an end-of-file mark to a UniVerse file open for sequential processing at the current position.
WRITEBLK Writes a block of data to a record in a sequential file.
WRITESEQ Writes new values to the specified record of a UniVerse file sequentially.
WRITESEQF Writes new values to the specified record of a UniVerse file sequentially and ensures that the data is written to disk.

Use locking techniques

Shared Record Lock: READL filevar, key LOCKED statement THEN statement ELSE statement ON ERROR statement

Exclusive Record Lock: READU filevar, key LOCKED statement THEN statement ELSE statement ON ERROR statement

Shared File Lock: FILELOCK filevar, FS ON ERROR statement LOCKED statement

Intent File Lock: FILELOCK filevar, IX ON ERROR statement LOCKED statement

Exclusive File Lock: FILELOCK filevar, FX ON ERROR statement LOCKED statement

Process Lock: LOCK semaphore THEN statement ELSE statement

RECORDLOCKED() Establishes whether or not a record is locked by a user.

RECORDLOCKL Sets a shared read-only lock on a record in a file.

RECORDLOCKU Locks the specified record to prevent other users from accessing it.

RELEASE Unlocks records locked by READL, READU, READVL, READVU, MATREADL, MATREADU, MATWRITEV, WRITEV, or WRITEVU statements.

Use SQL triggers

Developers use triggers to enforce certain business rules when users or programs make changes to a database.

CREATE TRIGGER triggername BEFORE INSERT OR UPDATE ON tablename FOR EACH ROW CALLING 'triggerprog';

BASIC subroutine *triggerprog* must define 14 arguments

Section 5 - Data Formatting and Manipulation (19%) - 15 Questions

Format numeric data

FMT (string, [width][fill]justification [edit][mask])

edit: n[m]Used with L, R, or T justification, n is the number of digits to display to the right of the decimal point, and m descales the value by m minus the current precision.

DTX() Converts a decimal integer into its hexadecimal equivalent.

FIX() Rounds an expression to a decimal number having the accuracy specified by the PRECISION statement.

FMT() Converts data from its internal representation to a specified format for output.

✓ FMT (string, [width][fill]justification [edit][mask])

✓ edit: n[m]Used with L, R, or T justification, n is the number of digits to display to the right of the decimal point, and m descales the value by m minus the current precision.

FMTS() Converts elements of a dynamic array from their internal representation to a specified format for output.

OCONV() Converts data from its internal representation to an external output format.

PRECISION Sets the maximum number of decimal places allowed in the conversion from the internal binary format **XTD()** Converts a hexadecimal string into its decimal equivalent.

Format date/time data

DATE=OCONV('9166','D2") 3 Feb 93

DATE=OCONV(9166,'D/E') 3/2/1993

DATE=OCONV('9166','D2-") 2-3-93

DATE=OCONV(0,'D') 31 Dec 1967

TIME=OCONV(10000,'MT") 02:46

TIME=OCONV("10000","MTHS") 02:46:40am

TIME=OCONV(10000,'MTH") 02:46am

TIME=OCONV(10000,'MT.") 02.46

TIME=OCONV(10000,'MTS") 02:46:40

Format text data

FMT() Converts data from its internal representation to a specified format for output.

- ✓ FMT (string, [width][fill]justification [edit][mask])
- ✓ edit: n[m]Used with L, R, or T justification, n is the number of digits to display to the right of the decimal point, and m descales the value by m minus the current precision.

Format string/substring data

ASCII() Converts EBCDIC representation of character string data to the equivalent ASCII character code values.

CHAR() Converts a numeric value to its ASCII character string equivalent.

CHARS() Converts numeric elements of a dynamic array to their ASCII character string equivalents.

EBCDIC() Converts data from its ASCII representation to the equivalent code value in EBCDIC.

FMT() Converts data from its internal representation to a specified format for output.

FMTS() Converts elements of a dynamic array from their internal representation to a specified format for output.

SEQ() Converts an ASCII character code value to its corresponding numeric value.

SEQS() Converts each element of a dynamic array from an ASCII character code to a corresponding numeric value.

Format output data

FMT() Converts data from its internal representation to a specified format for output.

FMTS() Converts elements of a dynamic array from their internal representation to a specified format for output.

CONV() Converts data from its internal representation to an external output format.

CONVS() Converts elements of a dynamic array from their internal representation to an external output format.

Search strings/substrings

COL1() Returns the column position immediately preceding the selected substring after a BASIC FIELD function is executed.

COL2() Returns the column position immediately following the selected substring after a BASIC FIELD function is executed.

FIELD() Examines a string expression for any occurrence of a specified delimiter and returns a substring that is marked by that delimiter.

GROUP() The GROUP function works identically to the FIELD function.

FINDSTR Locates a given occurrence of a substring.

INDEX() Returns the starting column position of a specified occurrence of a particular substring within a string expression.

MATCHFIELD() Returns the contents of a substring that matches a specified pattern or part of a pattern.

Manipulate strings/substrings

ALPHA() Determines whether the expression is an alphabetic or non-alphabetic string.

CHECKSUM() Returns a cyclical redundancy code (a checksum value).

COMPARE() Compares two strings for sorting.

CONVERT Converts specified characters in a string to designated replacement characters.

CONVERT() Replaces every occurrence of specified characters in a variable with other specified characters.

COUNT() Evaluates the number of times a substring is repeated in a string.

COUNTS() Evaluates the number of times a substring is repeated in each element of a dynamic array.

DOWNCASE() Converts all uppercase letters in an expression to lowercase.

DQUOTE() Encloses an expression in double quotation marks.

EREPLACE() Substitutes an element of a string with a replacement element.

EXCHANGE() Replaces one character with another or deletes all occurrences of a specific character.

FIELDSTORE() Replaces, deletes, or inserts substrings in a specified character string.

FOLD() Divides a string into a number of shorter sections.

GROUPSTORE Modifies existing character strings by inserting, deleting, or replacing substrings that are separated by a delimiter character.

LEFT() Specifies a substring consisting of the first n characters of a string.

LEN() Calculates the length of a string.

LOWER() Converts system delimiters that appear in expressions to the next lower-level delimiter.

QUOTE() Encloses an expression in double quotation marks.

RAISE() Converts system delimiters that appear in expressions to the next higher-level delimiter.

RIGHT() Specifies a substring consisting of the last n characters of a string.

SOUNDEX() Returns the soundex code for a string.

SPACE() Generates a string consisting of a specified number of blank spaces.

SQUOTE() Encloses an expression in single quotation marks.

STR() Generates a particular character string a specified number of times.

TRIM() Deletes extra blank spaces and tabs from a character string.

TRIMB() Deletes all blank spaces and tabs after the last nonblank character in an expression.

TRIMF() Deletes all blank spaces and tabs up to the first nonblank character in an expression.

UPCASE() Converts all lowercase letters in an expression to uppercase.

Section 6 - Array, Variable, and Statement Label Handling (11%) - 9 Questions

Define and manipulate dimensioned arrays

DIMENSION Declares the name, dimensionality, and size constraints of an array variable.

MAT Assigns a new value to every element of an array with one statement.

MATREAD Assigns the data stored in successive fields of a record from a UniVerse file to the consecutive elements of an array.

MATREADL Sets a shared read lock on a record, then assigns the data stored in successive fields of the record to the consecutive elements of an array.

MATREADU Sets an exclusive update lock on a record, then assigns the data stored in successive fields of the record to the consecutive elements of an array.

MATWRITE Assigns the data stored in consecutive elements of an array to the successive fields of a record in a UniVerse file.

MATWRITEU Assigns the data stored in consecutive elements of an array to the successive fields of a record in a UniVerse file, retaining any update locks set on the record.

Define and manipulate dynamic arrays

CATS() Concatenates elements of two dynamic arrays.

CHANGE() Substitutes an element of a string with a replacement element.

DCOUNT() Evaluates the number of delimited fields contained in a string.

DEL Deletes the specified field, value, or subvalue from a dynamic array.

DELETE() Deletes a field, value, or subvalue from a dynamic array.

EXTRACT() Extracts the contents of a specified field, value, or subvalue from a dynamic array.

FIELDS() Examines each element of a dynamic array for any occurrence of a specified delimiter and returns substrings that are marked by that delimiter.

FIND Locates a given occurrence of an element within a dynamic array.

GETREM() Returns the numeric value for the position of the REMOVE pointer associated with a dynamic array.

INDEXS() Returns the starting column position of a specified occurrence of a particular substring within each element of a dynamic array.

INS Inserts a specified field, value, or subvalue into a dynamic array.

INSERT() Inserts a field, value, or subvalue into a dynamic array.

LENS() Calculates the length of each element of a dynamic array.

LOCATE Searches a dynamic array for a particular value or string, and returns the index of its position.

MATBUILD Builds a string by concatenating the elements of an array.

MATPARSE Assigns the elements of an array from the elements of a dynamic array.

REMOVE Removes substrings from a dynamic array. REMOVE x FROM darray SETTING delim
REMOVE() Successively removes elements from a dynamic array. Extracts successive fields, values, etc., for dynamic array processing.
REVREMOVE Successively removes elements from a dynamic array, starting from the last element and moving right to left. Extracts successive fields, values, etc., for dynamic array processing.
REPLACE() Replaces all or part of the contents of a dynamic array.
REUSE() Reuses the last value in the shorter of two multivalued lists in a dynamic array operation.
SETREM Sets the position of the REMOVE pointer associated with a dynamic array.
SPACES() Generates a dynamic array consisting of a specified number of blank spaces for each element.
SPLICE() Inserts a string between the concatenated values of corresponding elements of two dynamic arrays.
STRS() Generates a dynamic array whose elements consist of a character string repeated a specified number of times.
SUBSTRINGS() Creates a dynamic array consisting of substrings of the elements of another dynamic array.
TRIMBS() Deletes all trailing blank spaces and tabs from each element of a dynamic array.
TRIMFS() Deletes all leading blank spaces and tabs from each element of a dynamic array.
TRIMS() Deletes extra blank spaces and tabs from the elements of a dynamic array.

Define statement labels

A statement label is a unique identifier for a program line. A statement label consists of a string of characters followed by a colon. The colon is optional when the statement label is completely numeric. Like variable names, alphanumeric statement labels begin with an alphabetic character and can include periods (.), dollar signs (\$), and percent signs (%). Upper- and lowercase letters are interpreted as different; that is, ABC and Abc are different labels. Statement labels, like variable names, can be as long as the length of the physical line, but only the first 64 characters are significant. A statement label can be put either in front of a BASIC statement or on its own line. The label must be first on the line—that is, the label cannot begin with a space.

Define variable names

A variable name must begin with an alphabetic character. It can also include one or more digits, letters, periods, dollar signs, underscores, or percent signs. Spaces and tabs are not allowed. A variable name can be any length up to the length of the physical line, but only the first 64 characters are significant. A variable name cannot be any of the reserved words listed in Appendix D. In UniVerse, upper and lowercase characters in a variable name are interpreted differently.

Use EQUATE, COMMON and INCLUDE

EQU[ATE] symbol TO expression [,symbol TO expression ...]

EQU[ATE] symbol LIT[ERALLY] string [,symbol LIT string ...]

COM[MON] [/name/] variable [,variable ...]

INCLUDE [filename] program

INCLUDE program FROM filename

You can nest INCLUDE statements.

The INCLUDE statement is a synonym for the \$INCLUDE and #INCLUDE statements.

Section 7 - System Variables and Functions (10%) - 8 Questions

Use COUNT and DCOUNT

DCOUNT(A,@AM) is equivalent to COUNT(A,@AM) + (A NE "")

Get and use information from FILEINFO

FILEINFO (file.variable , key)

file.variable is the file variable of an open file.
key is a number that indicates the particular information required.

Use PERFORM and EXECUTE

PERFORM Executes UniVerse sentences and paragraphs from within the BASIC program in the existing environment

EXECUTE Executes UniVerse sentences and paragraphs from within the BASIC program in a new environment

Get and use information from SYSTEM

SYSTEM (expression)

Use the SYSTEM function to check on the status of a system function.

Use system DATE and TIME

DATE()

TIME()

Section 8 - Debugging Application Programs (4%) - 3 Questions

Understand basic debugger capabilities

Temp variables \$R0 - \$Rn, \$MATRIX

123 B

VAL/

VAL!3

VAL M

S

Invoke program debugger

RAID BP TEST

Compile programs for debugging

A = B + 1

DEBUG