UOS User's Guide

Table of contents

UOS User's Manual

UOS User's Guide

October 2023

Contents

Contents

Preface

Preface

Intended Audience

This manual is intended for all users of the UOS operating system.

Within this document, certain key combinations are indicated by delimiting the keys with slashes. For example:
Ctrl/Y
indicates to hold down the Ctrl (control) key and then press the Y key.

Using UOS

Using UOS

To use UOS, you must have a user name and you must log in with that name. Usually a password is also associated with the user name. Sometimes multiple passwords or other authentication methods are required. Your password should be kept secret so that no one else can log in to your account. Also passwords should be sufficiently long (at least 8 characters) and not be an actual English word or other easily-guessed password.

To log in, press ENTER on a terminal connected to the UOS system. You will then be prompted for a user name and a password. If you enter valid values for both, you will be logged in and can access UOS.

When you finish using UOS, you should logout to prevent someone else from using your logged-in account to access UOS.

Files and Directories

Files and Directories

Data is stored on disks and tapes as files, which are named places reserved for the file contents. Files exist within a directory, which is a collection of zero or more files. "Folder" is an alternate name for a directory. Directories can contain children directories, which can contain files and other directories. The backslash (\) character is used to indicate the end of a directory name. Thus, the specification:

\uos\users\fred\myfile.txt

indicates a file named "myfile.txt" in a directory called "fred". The "fred" directory is a child directory of the "users" directory, which is a child directory of the "uos" directory. The initial backslash indicates the root folder of a disk or tape. The root directory has no name, which is why it is denoted by a single backslash.

The list of directories that precede the file name is called the "path" to the file. Each file has a unique name in a given path, but the same file name can be used in different paths. For instance, the following two files are different files because they exist in different paths:

\fred\temp\temp.tmp
\fred\temp.tmp

Wildcards

Wildcards

Wildcards are special characters that can be used in a file or directory name, for purposes of matching patterns in the names. They are supported by system programs to allow operations on multiple files. For instance, the file name:

a*.txt

indicates any file with a name starting with "a" (case insensitive) followed by zero or more characters of any kind, and with a file extension of ".txt".

There are four types of wildcards that can be used in a file specification - two of them are only valid in the path portion. The other two can be used anywhere.

Wildcard	Description
*	Matches 0 or more characters.
?	Matches a single character.
**	Only in paths. Match in all subdirectories. Any further pathing information after this wildcard is ignored.
**^	Only in paths. Refers to the parent directory.

The parent folder is indicated by the "**^" specifier. "**^" must be the entirety of the characters between delimiting backslashes or it will be treated as the wildcard "*^". To illustrate, "**^\" indicates a parent directory. But "\B**^A\" would be treated as the wildcard specification "\B*^A\" - that is, any directory starting with "B" and ending with "^A". Likewise the "**" wildcard must be the only characters between the delimiting backslashes or it will be reduced to a single "*".

For example, "a**\myfile.txt" indicates "myfile.txt" in any folder under the "a" folder. This would include, for instance, "a\mydir\myfile.txt" and "a\backup\new\myfile.txt", as well as "a\myfile.txt", and so forth.

Using UCL to Interact with the System

Using UCL to Interact with the System

The UOS Command Language (UCL) is a means of telling the operating system what specific operations to do on behalf of the user. UCL provides numerous commands and functions which can be used to control computer operation and/or to write scripts for later execution. Commands entered manually are called "immediate mode" operations.

The default UCL prompt is a dollar sign ($). This indicates that UCL is ready to accept a command.

Input Line Editing

Deleting Parts of the Command Line

Pressing the the Delete key (or backspace on a PC keyboard) will move the cursor backwards and erase the character in that space. If line editing is enabled, you can use Ctrl/U to delete characters from the beginning of the line to the current cursor position.

Key Sequences

Certain keys and key sequences are provided to allow users to edit a line of input before you are ready to submit it.

Key(s)	Function
Ctrl/Z	Signals the end of the file for data entered from the terminal.
Enter or Return	Submits the current line for processing.
Ctrl/T	Displays a line of statistical information about the current process. This display includes your node and user name, the time, the name of the image you are running, and information about system resources used during your terminal session.
Backspace or Delete	Deletes the character to the left of the cursor.
Ctrl/A	Switches between overstrike and insert mode. It is reset to insert mode at the beginning of each line.
Ctrl/B	Command recall. See the following section on Command recall.
Ctrl/D	Moves the cursor one character to the left.
Ctrl/E	Moves the cursor to the end of the line.
Ctrl/F	Moves the cursor one character to the right.
Ctrl/I or Tab	Moves the cursor to the next tab stop on the terminal. By default, tab stops are at every eighth character
Ctrl/J or Line feed	Submits the current line for processing.
Ctrl/L	Moves cursor to the beginning of the new page.

Ctrl/R	Redraws the current command line and leaves the cursor positioned where it was when you pressed Ctrl/R.
Ctrl/U	Deletes all text in the current line that is to the left of the cursor.
Ctrl/X	Cancels the current line and deletes data in the type-ahead buffer.
Ctrl/O	Alternately suspends and continues display of output to the terminal.
Ctrl.S	Suspends terminal output until Ctrl/Q is pressed.
Ctrl/Q	Resumes terminal output suspended by Ctrl/S
Ctrl/C	Interrupts command processing.
Ctrl/Y	Interrupts command processing.

Command Recall

Command Recall

As commands are entered in UCL, they are stored in a buffer, called the "recall buffer". There are two ways to interact with saved commands in the recall buffer.

Control-B will recall a previously-entered command. Each time it is pressed, it displays the previously entered command, or does nothing if there is no previous command. Pressing Control-B multiple times will step back through the entered commands starting with the most recent and moving back toward the first with each Control-B. A recalled command is shown on the input line and it can be executed by pressing ENTER. The line is in the input buffer and can be edited with line editing keystrokes as if you had typed the command from the keyboard.

The RECALL command can be used to view the recall buffer or select a specific previously-entered command.

Date and Time

Date and Time

UOS Dates can be specified in three forms: absolute, delta, and combination.

Absolute Date/Time Format
An absolute date/time indicates a specific date or time of day (or both). The default format is: {dd-mmm-yyyy}{:}{hh:mm:ss.cc}

The fields are:

Field	Meaning
dd	Day of month. An integer from 1 to 31.
mmm	Month of year. One of the following: JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC.
yyyy	Year. An integer value from 0 to 8000.
hh	Hour of day. An integer value from 0 to 23.
mm	Minute of hour. An integer value from 0 to 59.
ss	Second of minute. An integer value from 0 to 59
cc	Hundredths of seconds. An integer value from 0 to 99

Note that although UOS keeps track of time to the nanosecond, there is no way to specify a time interval of less than .01 seconds using this format. Also note that even if the hardware that UOS is running on is not able to keep time to this level of resolution, the timestamp value still reserves the space to represent it.

An absolute time can also be specified as one of the following special values.

Value	Definition
TODAY	The current date at 00:00:00.0 o'clock
TOMORROW	Tomorrow at 00:00:00.0 o'clock
YESTERDAY	Yesterday at 00:00:00.0 o'clock

The date/time can be abbreviated as follows.

- The date or time can be truncated to the right. For instance "28-FEB" or "11:".
- If you specify both a date and time, they must be separated by a colon.
- Any field can be null if the delimiters are included.
- Unspecified date fields default to the appropriate value for the current date.
- Unspecified time fields default to zero (0).

Examples of abbreviated date/times

Specification	Result
14-OCT	October 14 of this year at 00:00:00.0 o'clock.
11:	11:00 today.
10-:30	12:30 AM on the 10th day of the current month.
9-	00:00:00.0 (12 AM) on the 9th day of the current month.

Delta Date/Time Format

Delta date/time format is an interval relative to the current date and time to a date and time in the past or future. In terms of stand-alone delta times, UOS only supports future delta times. The reason is that delta times are stored as negative values. The format is:
+{dddd-}{hh:mm:ss.cc}

The fields are:

Field	Meaning
+	This is a plus sign (+) to refer to a future date/time. Include an additional minus sign (-) to refer to a previous date/time.
dddd	Days. An integer from 1 to 9999.
hh	Hours. An integer value from 0 to 23.
mm	Minutes. An integer value from 0 to 59.
ss	Seconds. An integer value from 0 to 59
cc	Hundredths of seconds. An integer value from 0 to 99

Delta date/time specifications must start with a plus sign.

You can abbreviate the delta date/time as follows.

- The time can be truncated to the right. For instance "11:".

- If you specify both a date and time, they must be separated by a dash.
- Any field can be null if the delimiters are included.
- Unspecified time fields default to zero (0).

Combination Date/Time Format
You can combine an absolute date/time with a delta date/time. The default format is:
{"}{absolute date/time}+delta{"}
{"}{absolute date/time}-delta{"} {"}{absolute date/time}+-delta{"}

The absolute and delta time/date specifications use the same fields and defaults as described above. The one exception is that the delta time/date can start with a minus sign instead of a plus-and-a-minus as long as that is not ambiguous (it can always start with +-). Thus, in combination dates, a delta can be past or future. But you cannot mix plus and negative deltas (such as a negative day with a positive hour) - the entire delta portion of the combination date is either future or past.

The following rules apply:
- The combination can be enclosed in quotes, but they are not required.
- The absolute date/time can be omitted to offset the delta from the current date/time. However, if no absolute date/time is specified, past delta date/times must start with the "+-" prefix.

Examples of Combination date/times

Date Specification	Result
+2	Two days from now.
11:-0:2	9:00 AM today.
31-DEC +::5	12:05 AM on December 31 of the current year.
1-JAN+-0:2:30	9:30 PM on December 31 of last year. Note that the +- construct must be used to distinguish between a dash used to delimit month and year from a negative offset.
1-JAN:-0:2:30	9:30 PM on December 31 of last year. Use of a colon prevents the dash from being interpreted as a delimiter.
+3:0:15	Three days and fifteen minutes from now.

Privileges

Privileges

Privileges allow users to bypass various restrictions that protect the integrity of UOS resources and other users. Authorized privileges are defined for a user at the time of account creation. Upon logging in, these privileges are enabled for that session. The privileges are:

Privilege	Description
ACNT	Allows the process to use the RUN command and the Create Process system service to create processes for which accounting is disabled. When accounting is disabled the resource usage for the process is not logged in the accounting file.
ALLSPOOL	The user's process can allocate spooled devices.
ALTPRI	Allows process to alter the priority of itself or other processes, and change the priority of bath or print jobs. WORLD privilege is required to change the priority of processes of other user accounts.

AUDIT	Allows the process to write security audit log file messages.
BUGCHK	Allows the process to write records to the error logger.
BYPASS	Allows process to bypass UIC-based and ACL protections for files and other objects. It also allows access to files marked for deletion but still open, convert directories to normal files, ignore file locks, and modify file ownership.
CMEXEC	Allows calls to Change Mode to Supervisor system service.
CMKRNL	Allows call to Change Mode to Kernel/Executive system service.
DETACH	Alternative name for IMPERSONATE privilege.
DIAGNOSE	Allows process to run diagnostics and intercept error log messages.
DOWNGRADE	Allows process to write to objects of lower secrecy.
EXQUOTA	Allows process to exceed usage quotas for stores.
GROUP	Allows process to affect processes belonging to a group that the process user belongs to.
GRPNAM	Allows process to bypass access controls on symbol tables to add or remove names from tables belonging to any group that the process user belongs to.
GRPPRV	Allows the process to access files and objects per the Group protection field for groups that the user belongs to.
IMPERSONATE	Allows detached processes to be created with a different UIC, so long as MAXJOBS and MAXDETACH quotas are not exceeded.
IMPORT	Allows process to manipulate mandatory access controls.
LOG_IO	Allows process to execute the QIO system service to perform operations on I/O devices beyond the normal read and write.
MOUNT	Allows process to mount volumes.
NETMBX	Allow network control operations.
OPER	Allows process to use OPCOM.
PFNMAP	Allows unrestricted access to physical memory.
PHY_IO	Allows process to bypass normal read/write operations to write directly to I/O devices.
PRMCEB	Allows creation/deletion of permanent common event flag clusters.
PRMGBL	Allows creation/deletion of permanent global sections.
PRMMBX	Allows creation/deletion of permanent mailboxes.
PSWAPM	Allows control of swapping operations.
READALL	Allows process to bypass existing restrictions and allow reading any file or object.
SECURITY	Allows process to perform security-related functions such as changing the system password and starting/stopping audit servers.
SETPRV	Allows process to create processes that have privileges greater than the user.
SHARE	Allows process to open assigned devices or to assign non-shared devices.
SHMEM	Allows process to create global sections and mailboxes in memory shared by multiple processors.
SYSGBL	Allows process to create/delete system global sections.
SYSLCK	Allows process to obtain lock information and lock system-wide resources.
SYSNAM	Allows process to bypass access controls on symbol tables to add or remove names from system tables.
SYSPRV	Allows process to access protected objects via the system field in protection codes, modify the owner, protection code, and/or ACL of any object. The privilege also allows modifications to the SYSUAF file.
TMPMBX	Allows process to create temporary mailbox.
UPGRADE	Allows process to write to objects of higher integrity.
VOLPRO	Allows process to initialize volumes with different UICs, override UIC protection of a

	volume, override volume expiration dates, and mount volumes as foreign.
WORLD	Allows process to control any/all other processes.

Many of these privileges are designed to be assigned to images so that they can perform operations on behalf of users.

Commonly Used System Programs

Commonly Used System Programs

UOS provides several common useful programs that can be executed directly from UCL, either interactively or in scripting.

COPY

COPY

COPY can copy one or more input files to one or more output files, or concatenate multiple input files into a single output file.

Format

COPY input-filespec{,...} output-filespec

Parameters

input-filespec{,...}

Specifies the name of an existing file to copy. Wildcards may be specified and multiple file specifications can be specified, delimited by commas (,) or plus signs (+). Any filename containing a comma or plus must be enclosed in quotes. If no device or directory is specified, COPY uses the current default device and directory.

output-filespec

Specifies the name of the output file into which the input is copied. If no device or directory is specified, COPY uses the current default device and directory. COPY replaces any other missing fields (file name, file type, version number) with the corresponding field of the input filespec.
Wildcards can be used, which will be subsituted from the current input file.

Description

If a single input file is specified, that file is copied to the specified destination. If multiple input files are specified, the operation depends upon the output specification. If the output specification contains any wildcards, each input file is copied to the output specification with the wildcard fields substituted from the input file. If the output specification doesn't include any wildcards, the input files are concatenated into the specified output file. Note that missing output fields are treated as "*" wildcards.

If the output file contains a wildcard or the name or extension isn't explicitly specified, the output file will have the same creation date. Otherwise, the output file will be treated as a new file and given the current creation date. In all cases, the Last Modified date is set to the current date, and the Last Backup, Last Access, and Expiration dates are cleared.

The default protection is that of the existing output file. If no output file exists, the current default protection is applied. However, ACLs on the target directory may affect the availability of the new file(s).

The owner of the output file(s) will be the user copying the file(s). However, if the user has SYSPRV or BYPASS privilege, the output file(s) will have the same owner as any previous version of the file or - if there are no previous version - the file is assigned the same owner as the parent directory.

If a directory is copied, an empty output directory is created - the files in the directory are not copied.

Priveleges assigned to a file are limited to the user's current privileges unless the user has the SETPRV privilege.

ACLs on input files are not copied, but all other meta data for the file is copied, except as described above. In the case of conflicting meta data during a concatenation operation, the first instance of meta data from the input files takes precedence.

Switches

/ALLOCATION{=size}

Specifies initial size of each output file, in bytes. If not specified, or the size is null, the initial size comes from the input file size. In a concatenation operation, the output file size is taken from the first input file, if the switch isn't specified.

/BACKUP

Indicates that the /SINCE or /BEFORE switch is to use the file's backup time. This switch is incompatible with the /CREATED, /EXPIRED, and /MODIFIED switches. If none of these are specified, the default is /CREATED.

/BEFORE{=time}

Selects only those files dated prior to the specified time. The time can be an absolute time, a combination of absolute and delta times, or one of the following keywords: BOOT, LOGIN, TODAY (the default), TOMORROW, or YESTERDAY. One of the following can be used with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (the default), /EXPIRED, or /MODIFIED.

/BLOCK_SIZE=n

Overrides the default block size (128) for RMS blocked files.

/BY_OWNER{=user}

Selects only those files whose owner is the specified user. This may be the user name or the user identification code (UIC). A numeric value is checked against valid user names first and, if not found, is considered to be a UIC.

/CONCATENATE (default)
/NOCONCATENATE

Creates one output file from multiple input files when no wildcard characters are used in the output file specification. The /NOCONCATENATE switch generates multiple output files. A wildcard character in an input file specification results in a single output file consisting of the concatenation of all input files matching the file specification. Input files matching the wildcard specification are concatenated in random order. If /CONCATENATE is used with an output file specification that has no wildcards, the output files are assigned different version numbers.

/CONFIRM
/NOCONFIRM (default)

Controls whether the user is queried before each copy operation to confirm that the operation should be performed on that file. The following responses are valid:

Affirm	**Decline**	**Other**
YES	NO	QUIT
TRUE	FALSE	Ctrl/Z
1	0	ALL

Any combination of uppercase and lowercase letters can be used for word responses. The responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), Entering "QUIT" or

pressing Ctrl/Z stops the copy operation at that point. A response of "ALL" causes COPY to continue with no further prompting. If the response is null (such as by simply pressing the ENTER key), it is considered the same as typing "NO". Any other response results in an error message being displayed and the user is prompted again.

/CONTIGUOUS
/NOCONTIGUOUS
Specifies that the output file must occupy contiguous physical disk clusters. By default, COPY creates a contiguous output file only if the input file is contiguous. Also, by default, if not enough space exists for a contiguous allocation, the output file will not be contiguous and COPY does not report an error. During concatenation, if some input files are contiguous and some are not, the output file may or may not be contiguous. /CONTIGUOUS can be used to ensure that the output files are contiguous.

The /CONTIGUOUS switch has no effect when files are copied to sequential devices (such as tapes) because sequential devices are already inherently contiguous. The switch also has no effect when copying files from a serial device since the size of the file cannot be determined until after it is copied to the disk. In this case, the file must first be copied to the disk and then copied in-place on the disk with the /CONTIGUOUS switch.

/CREATED (default)
Indicates which file timestamp to use in conjunction with /BEFORE or /SINCE. /CREATED selects files based on their dates of creation. This is incompatible with /BACKUP, /EXPIRED, and /MODIFIED, which also allow for selecting files according to time attributes. If none of switches are used, the default is /CREATED.

/EXCLUDE=(filespec{,...})
Excludes the specified files from the copy operation. Device specifications in the file specifications are ignored. Wildcard characters are allowed in the file specification; however, you cannot use relative version numbers to exclude a specific version. If only one file is specified, the parentheses can be omitted.

/EXPIRED
Indicates that the expiration date is used with the /BEFORE or /SINCE switches. /EXPIRED is incompatible with /BACKUP, /CREATED, and /MODIFIED. If none of these is specified, the default is /CREATED.

/EXTENSION=n
Specifies the number of clusters to be added to the output file each time the file is extended. If /EXTENSION is not specified, the extension attribute of the corresponding input file determines the default extension attribute of the output file. If the input file has no extension attribute, the default clustersize of the output device is used.

/LOG
/NOLOG (default)
Controls whether COPY displays the file specifications of each file copied. When /LOG is used, COPY displays the following for each file copied:
The file specifications of the input and output files.
The number of blocks or records copied.
The total number of new files created.

/MODIFIED
Indicates that the modification timestamp is to be used with /BEFORE or /SINCE. This is incompatible with /BACKUP, /CREATED, and /EXPIRED, which also allow the selection of files according to time attributes. If none of these are specified the default is /CREATED.

/OVERLAY
/NOOVERLAY (default)
Indicates that data in the input file is to be copied into the existing output file, overlaying the existing data, rather than allocating new space for the file. The physical location of the file on disk does not change; however, for RMS indexed and relative files, if the output file has fewer blocks allocated than the

input file, the copy fails giving an RMS-E-EOF error. /OVERLAY is ignored if the output file is written to a non-filestructured store.

/PROTECTION=(ownership{:access}{,...})
Specifies protection for the output file. The ownership parameter is: system (S), owner (O), group (G), or world (W). The access parameter is: read (R), write (W), execute (E), or delete (D). The default protection, including any protection attributes not specified, is that of the existing output file. If no output file exists, the current default protection applies.

/READ_CHECK
/NOREAD_CHECK (default)
Instructs copy to verify the read operation by performing multiple read operations. This does not apply to serial devices

/REPLACE
/NOREPLACE (default)
If a file exists with the same file specification (including version number) as that entered for the output file, the existing file is to be deleted if /REPLACE is specified. By default, COPY creates a new version of a file if a file with that specification exists, incrementing the version number. /NOREPLACE signals an error when a conflict in version numbers occurs.

/SINCE{=time}
Selects only those files dated on or after the specified time. The time can be specified as absolute time, a combination of absolute and delta times, or as one of the following keywords: BOOT, JOB_LOGIN, LOGIN, TODAY (default), TOMORROW, or YESTERDAY. /SINCE indicates the time attribute to be used as the basis for selection in conjunction with /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/STYLE=keyword
Specifies the file name format for display purposes. The valid keywords for this switch are CONDENSED and EXPANDED. Descriptions are as follows:

Keyword	Explanation
CONDENSED (default)	Displays the file name representation of what is generated to fit into a 255-length character string.
EXPANDED	Displays the file name representation of what is stored on disk.

The keywords CONDENSED and EXPANDED are mutually exclusive. This specifies which file name format is displayed in the output message, along with the confirmation if requested.

/SYMLINK
/NOSYMLINK (default)
/NOSYMLINK indicates that if an input file is a symbolic link, the file to which the symbolic link refers is the file that is copied. Otherwise, the link itself is copied.

/TRUNCATE (default)
/NOTRUNCATE
Controls whether the COPY operation truncates an output file at the end-of-file (EOF) when copying it. When copying an RMS file, this operation can only be used with sequential files. By default, the actual size of the input file determines the size of the output file. If /NOTRUNCATE is used, the allocation of the input file determines the size of the output file. Note that any data following the EOF may or may not be copied to the output file.

/VERSION
/NOVERSION (default)
Indicates whether or not filenames with trailing semicolons and numbers should be treated as having version numbers. /NOVERSION indicates that any version numbers should be ignored and simply treated as part of the filename.

/WRITE_CHECK
/NOWRITE_CHECK (default)
Reads the output file data after it is written to verify that the data copied successfully. This only needs to be used for copy operations to devices that do not have inherent data integrity protection. It does not apply to serial devices.

Examples

COPY MYFILE.TXT COPY.TXT
Copies the file named MYFILE.TXT from the current device/directory to a new file named COPY.TXT in the same device/directory.

COPY *.EXE temp*.EXEBACKUP
Copies all files with the extension .EXE from the current device/directory to new files with the same name, but the .EXEBACKUP extension in the temp folder in the current device/directory.

COPY FILE1.TXT+FILE2.TXT+FILE3.TXT FILES.TXT
Copies the contents of FILE1.TXT into FILES.TXT, then the contents of FILE2.TXT into FILES.TXT at the end, then the contents of FILE3.TXT into FILES.TXT at the end. When compete, FILES.TXT will contain the contents of all three input files.

COPY MYAPP.EXE MYAPP.EXE/CONTIGUOUS
Copies MYAPP.EXE over itself, making it contiguous.

HELP

HELP

The HELP program displays information about using the system, including formatting and descriptions of utilities. In response to the Topic? prompt, you can do any of the following:

- Type the name of a utility or topic for which you want information.
- Type INSTRUCTIONS for detailed instructions on using HELP.
- Type HINTS if you are not sure of the utility or topic for which you need help.
- Type a question mark (?) to redisplay the most recently displayed text.
- Press ENTER to back up to a previous topic or exit the HELP utility.
- Enter @filespec to choose a different help file from the default.
- Enter * to display help for all topics at the current level.

Format

HELP {topic{ subtopic...}}

Command Parameters

topic{ subtopic...}
Specifies the topic(s) for which you want information. Wildcards (* and ?) can be used to select matching topics at a given level. For example, "COPY *" should show all subtopics for the COPY CUSP.

Qualifiers

/EXACT
Used with /SEARCH to specify a search that must match the search string exactly and must be delimited with quotes (").

/HIGHLIGHT{=keyword}
Used with /SEARCH to specify the type of highlighting for search results. When a match is found, the entire line is highlighted. The following keywords can be used: BOLD, BLINK, REVERSE, and

UNDERLINE. BOLD is the default if no keyword is specified.

/INSTRUCTIONS (default)
/NOINSTRUCTIONS
Displays an explanation of the HELP utility along with the list of topics. If /NOINSTRUCTIONS are specified, only the list of topics is displayed.

/LIBLIST (default)
/NOLIBLIST
Displays any auxiliary help libraries.

/LIBRARY=filespec
/NOLIBRARY
Uses an alternate help library instead of the default (sys$help:helplib.hlb). The default location used is sys$help, unless otherwise specified. The default file type is .HLB.
/NOLIBRARY excludes the default help library from the search order.

/OUTPUT=filespec
/NOOUTPUT
Defines where the output from HELP is directed. By default, the output is to sys$output. If the file specification is partial, the output file name is HELP and the default file type is .LIS. Wildcard characters are not allowed in the file specification.
/NOOUTPUT suppresses output.

/PAGE{=keyword}
/NOPAGE (default)
Controls the output from HELP. The following keywords can be used:
CLEAR_SCREEN Clear screen before each page is displayed
SCROLL Display information one line at a time.
SAVE{=n} Save n pages of text that can be scrolled through. If not specified, n is set to 5.

When using /PAGE=SAVE, the user is allowed to scroll backwards and forwards through help text. The following keys can be used:
Ctrl-B Scroll up one line
Enter Get next page of information
Space Get next page of information
Ctrl-Z Exit

/PAGE is not compatible with /OUTPUT.

/PROMPT (default)
/NOPROMPT
Permits interactive use of HELP. The use of /NOPROMPT tells HELP to exit as soon as it outputs the requested information. When used interactively, HELP will prompt the user after the requested article is displayed.

If the user enters a topic from the current level, the information for that topic is displayed. A different library can be specified, prefixed by an at-sign (@) in order to switch to a different help library. If the user presses ENTER, HELP exits one level. If at the top level, ENTER will exit the HELP CUSP. Control-Z will immediately exit HELP.

/SEARCH="string"
Searches the displayed text for the specified string. Quotes are only required if the search string contains spaces.

/USERLIBRARY=(level{,level...})
/NOUSERLIBRARY (default)
Names the libraries to be used to resolve topic references. The valid levels are:
PROCESS Libraries defined at process level.
GROUP Libraries defined at group level.

SYSTEM Libraries defined at system level.
ALL All libraries (default).
NONE No libraries (same as /NOUSERLIBRARY)

Auxiliary help libraries are defined with the logical names HLP$LIBRARY, HLP$LIBRARY_1, and so forth. Libraries are searched in the following order: current root library, main root library (if not the same as the current root), process-level libraries, group-level libraries, and system-level libraries.

/WRAP
/NOWRAP (default)
Controls whether HELP wraps the text within the margins of the current output device. /NOWRAP simply outputs the text as-is and lets the output device do its own wrapping.

Examples

$ HELP
This command, entered without any swtiches or parameters causes a display of help topics from the default root help library, sys$help:helplib.hlb. It then prompts the user for commands.

$ HELP COPY
This command displays the help text for the COPY CUSP. It then prompts the user for commands.

$ HELP/NOPROMPT COPY *
This command displays the help text for the COPY CUSP, and the text of all subtopics, and then exits.

LOGIN

LOGIN

Initiates an interactive terminal session.

Format

Ctrl/C
Ctrl/Y
Enter

Description

When Ctrl/C, Ctrl/Y, or Enter is typed on a terminal not associated with a process, the LOGIN program is run. The system prompts for the user name, and authentication (usually a password).

LOGIN performs the following functions:
Prompt for user name
Prompt for authentication
Validates user name and authentication
Sets process characteristics as per the UAF settings for the user
Start default shell, or the shell specified in the UAF, or the shell specified with /CLI
Execute the command file sys$system:login.com

Qualifiers can be specified immediately after the username.

Qualifiers

/CLI=shell name
Specifies the name of an alternate shell to override the default shell specified in the UAF for the user. The shell must exist in sys$system. The default shell is UCL.

/COMMAND=filespec
/NOCOMMAND
Specifies a command file to execute upon login. If none is specified, the default login command file is

used. If /NOCOMMAND is specified, no login command file is used.

/DISK=device
The device specification to be associated with the SYS$DISK logical name for this session.

/NEW_PASSWORD
The user is prompted for a new password, as if the password had expired, before login can complete.

LOGOUT

LOGOUT

Terminates the process.

Format
LOGOUT

Description
The LOGOUT command must be used to end an interactive session, whether remote or local. Under most circumstances, you can turn the power off at the terminal, hang up a phone/modem, or terminate a network connection without using the LOGOUT command, but the process remains logged in and detached.

Qualifiers
/BRIEF
Displays a brief logout message (process name, date, and time). This is the default.

/FULL
Displays a long logout message including accounting information for the process. This is the default for batch jobs.

/{NO}HANGUP
The default is /NOHANGUP. /HANGUP will disconnect from a dial-up or remote connection.

Example
```
$ LOGOUT
  GONZALES     logged out at 03-Dec-2021 15:30:26.22
```


Scripting

Scripting

UCL commands can be placed into script files that can be executed multiple times.

Generally, lines in UCL command files must begin with a dollar-sign ($) to indicate that the line is a UCL command and not data intended for a program to consume. This is not always absolutely required - a line without a dollar-sign *might* work - but it is a good practice.

Line Continuation
Sometimes it is useful to continue a very long command onto multiple lines. When being typed at a terminal, a command is usually considered complete at the end of the line. To allow continuing a command onto the next input line from the terminal, a dash (-) can be included at the end of the line. The dash must be immediately preceded by a space or tab. Although not required in command files, the line continuation may be used there as well. Note, however, that any line with a comment on it cannot be continued since the dash is considered to be part of the comment itself. Multiple lines can be continued. An example of line

continuation:
dir \users\george\c*.* -

/date/size -

/page

The above lines would be interpreted as a single line that looked like this:
dir \users\george\c*.*/date/size/page

Interactive Mode
Finally, let's consider the difference between running a command file and typing commands manually from a terminal (called "interactive mode"). In the latter case, some operations make no sense and will result in an error. For instance, a GOTO command is pointless in interactive mode, because there is no target label to which to move execution. Likewise, using a label in interactive mode makes no sense because once the line has been entered, it is executed and forgotten. Any future (or prior) reference to that label can not go to it. We will discuss these situations as they come up.

UOS Errors and UCL Error Handling
It is possible that syntactically or contextually incorrect commands may be encountered in UCL - people are only human. Sometimes, situations beyond the script writer's control may cause errors to happen. Thus, UCL has error handling capabilities.

UOS errors can be warnings, errors, or fatal errors. Error handling can deal with each of these separately, if desired. When an error is displayed to the user, it consists of a string following this format:
%xxx-y-zzzzzz, error text

where "xxx" is the source of the error (usually a three-letter code, such as "UCL"), "y" is the level of error (W=warning, E=error, F=fatal), and "zzzzzz" is an error abbreviation code (usually six characters) which is unique for each error from a given source. Sometimes there are additional lines of information provided by UCL which indicate the items in the command line that caused the error. Example of an error:

%UCL-W-NOLBLS, label ignored - use only within command procedures

Command processing
When UCL executes a command, there are several phases that make up the processing. First is symbol substitution (described in the next section). Second, the command is parsed. During the parse phase (sometimes called the lexical phase), the command is checked for validity and lexical functions are evaluated. Finally, the command is executed.

Note that it is possible for some lexical functions to execute in the parsing phase even if the command generates an error. Consider the following command:

$ A = F$EDIT(A, F$PID(CTX))

During the parsing phase, F$PID is evaluated so that it can be used as a parameter to F$EDIT. This will change the CTX context, but because F$PID returns an integer value, F$EDIT parsing will return generate an error.

Substitution

Substitution

UOS symbols serve as UCL variables. UCL uses UOS symbols so that the symbols are available to any

other (different) shells running under UOS. UCL allows for two types of symbols/variables: local and global. Local symbols are stored in the LNM_PROCESS table, while global symbols are stored in the LNM_JOB table. Unless otherwise specified, when a symbol is references in UCL, if there are symbols in both the local and global tables with the same name, the local symbol is used.

Symbol substitution means that the actual values of symbols are substituted for the symbol names in the command line prior to the line being executed.

Phase One Symbol Substitution
The apostrophe (') is used to indicate a symbol substitution. For instance:
dir A

would return a directory listing of any files named "A". But
dir 'A'

would return a directory listing of any files with the name of the value of symbol A. For instance, if symbol A contained the value "XYZ", then the above command would look like this after substitution:
dir XYZ

A couple of things to note: first, the apostrophes must enclose the symbol. Second, the apostrophes are removed when the symbol value is substituted. Because the symbol name is delimited by apostrophes, we can embed the symbol name anywhere in the line. For instance:
dir X'A'1

would do a directory listing of any filename starting with "X", ending with "1" and with symbol A's value in-between. In the case of A containing "XYZ", after substitution the command would be:
dir XXYZ1

In the case of string literals (those delimited by double quotes), two apostrophes must be used to specify the start of a symbol name for substitution. Thus, single apostrophes can be freely used within literals without having to worry about inadvertant symbol substitution. Example:
A = "Hello, "NAME"'

Note that the symbol is still terminated by only a single apostrophe.

Symbol substitution is iterative. That is, if the substituted value contains apostrophe-delimited symbol name, then that is also substituted. For example, assume we have two symbols with the following values:
A = "'B"'

B = "UOS"

Now, if UCL does a substitution of the following:
dir 'A'

The first iteration of substitution results in:
dir 'B'

The next iteration results in:
dir UOS

Note that iterative substitution does not occur inside string literals. Thus, a full iterative substitution of:
dir ""A"'

will result in a final result of:
dir "'B"'

Phase Two Symbol Substitution
The apostrophe-driven symbol substitution is only the first of two phases of symbol substitution. After symbols have been substituted as described above, UCL next checks for any symbol name preceded by an

ampersand (&). Unlike phase one substitution, the ampersand substitution cannot be used in the middle of other text. In other words, the ampersand must be preceded by white space or a special character (not A-Z, 0-9, $, or _), and likewise for the character following the symbol name. In general, ampersand substitution should be avoided - use it only when it is the only way to accomplish what you need to do. To understand how both phases of symbol substitution work together, consider the following - assume P is empty (equals "") and B is "5":
dir 'P"B'

the substitution would result in:
dir 5

This is because both P and B are substituted, as expected. But consider:
dir &P'B'

In this case, after phase 1 substitution, the command would look like this:
dir &P5

Then phase two of symbol substitution occurs. In this case, symbol P5 is substituted. Assuming P5 contains "ABC", the new command will be:
dir ABC

which is quite different from
dir 5

Symbol values are also used in expressions. In such a case, the symbol's value is used without apostrophes or ampersands because of context. However symbol substitution may, in most cases, accomplish the same end result in expressions. But since it is unnecessary, it is best not to do so in command files, both for performance sake and clarity of script code.

Expressions

Expressions

Expressions are used in assigning values to symbols, in IF statements, and in function parameters.

Values
Expressions consist of operators and values. Values can be literals, symbols, or functions. A numeric literal is an integer numeric value and would be something like:
32767
whereas a string literal is delimited by quotes, containing any characters, and would look something like:
"This is text"

In expressions, the value of the symbol is used as if it were a literal value. Note that no substitution is required to use symbols in expressions unless you wish to have a level of indirection. Unless you want indirection, you probably do not want to use substitution. Consider this scenario: You want to compare text stored in symbol A with another value. If you use the code:
A .EQS. "Text"
then the contents of A will be compared to "Text". The quotes are not included in the comparison since they are simply syntax to indicate to the UCL parser that the value is text instead of a symbol called Text. If symbol A contains "Text", then the comparison will be true since they are equal.

Consider this substitution:
'A .EQS. "Text"
In this case, A is substituted with it's value before the expression is evaluated. Thus, the expression parser would see the following:
Text .EQS. "Text"

Because the first "Text" isn't in quotes, it is assumed to be a symbol named "Text". The value of that symbol would be used for the comparison or there is no symbol named Text and the expression will result in an undefined symbol error.

Numeric literals can also be specified in hexadecimal (base 16), by prefixing the value with "%X". For instance:
-%0FF
would be equal to -255.

Functions
Another type of value is the function. UCL functions are called "lexical functions" and are always prefixed with "F$". We will discuss these functions in a later article, but let us used the F$EDIT function as an example. Functions take parameters, specified within parentheses and delimited by commas. Different functions take different numbers of parameters, with different meanings. F$EDIT, for instance, takes two parameters and a reference to it would look something like:
F$EDIT(P1,P2)
where P1 and P2 are values that the function uses to do some sort of calculation and then return the result of that calculation. That result is a value which the expression parser can use as if it were a symbol reference or literal.

For example:
A .EQS. F$EDIT(B,"UPCASE")
Each parameter is an expression in itself.

Operators
An operator is something that takes two values and does something with them. There are two types of operators: arithmetic and logical. Arithmetic operators are things like addition and multiplication. Logical operators perform comparisons between values, such as Equal or Greater Than. Because an expression can contain multiple values and operators, some means of determining which order the operations occur must be defined. In Algebra, for instance, the following two equations result in different values:
1+2*3
3*1+2

But the following two result in the same value:
1+2*3
2*3+1

The reason is because, the rules of algebra define that multiplication has a higher precedence than addition, so the multiplication is always done first. Thus, we cannot simply process operators and operands from left to right and have the result be algebraically correct. Here are the arithmetical operators and their precedences:

Operator	Precedence	Description
-	7	Unary minus
+	7	Unary plus
*	6	Multiplication
/	6	Division
+	5	Addition
-	5	Subtraction

The higher precedence operators are evaluated before lower precedence operators. If the operators have the same precedence, they are evaluated left to right. Of special note are the unary operators. A unary minus is used to negate a numeric value. For instance:
-A

If symbol A has a value of "5", the above evaluates to -5 (negative five). Unary plus essentially does nothing since it leaves the value undefined. For instance, +5 is the same as 5 and +-5 is the same as -5. You may

have noticed that unary operators do not work like the other operators - they affect a single value instead of two values. The other operators could be called binary operators since they operate on two values. The unary operators must occur on the left side of the value to which they apply.

Here are the logical operators and their precedence:

Operator	Precedence	Description
.EQ.	4	Numeric values equal
.EQS.	4	String values equal
.NE.	4	Numeric values not equal
.NES.	4	String values not equal
.GT.	4	Numeric value greater than
.GTS.	4	String value greater than
.GE.	4	Numeric value greater than or equal to
.GES.	4	String value greater than or equal to
.LT.	4	Numeric value less than
.LTS.	4	String value less than
.LE.	4	Numeric value less than or equal to
.LES.	4	String value less than or equal to
.NOT.	3	Logical NOT
.AND.	2	Logical AND
.OR.	1	Logical OR

Logical operations work on boolean values: true and false. The comparison operators (all with precedence 4) are used to compare two values. There are two types of each comparison: numeric and string. Consider the following expression:
A .EQS. B
The result of this operation is true if the contents of symbols A and B exactly match, and false if they do not.

Likewise:
A .LT. B
compares A and B, as numbers, and returns true if A is less than B.

The last three operators perform operations on boolean values. The following tables express the results of these operators.

.AND.

Operation	Result
false .AND. false	false
false .AND. true	false
true .AND. false	false
true .AND. true	true

.OR.

Operation	Result
false .OR. false	false
false .OR. true	true
true .OR. false	true
true .OR. true	true

The .NOT. operator is a unary operator that simply inverts the boolean value, like so:

Operation	Result
.NOT. false	true
.NOT. true	false

Data types
UCL doesn't have strict data typing, so it is important to understand how values are interpreted when certain operators are used with them. For boolean values, any number that is odd (1,3,5, etc) is considered "true" and any other number is considered "false". Any string that begins with an uppercase or lowercase T or Y is considered "true" and any other case is considered "false". Boolean results are always "1" for true and "0" for false.

If a symbol contains a numeric value, that value is used as-is for numeric operators. But if the symbol contains something that is not numeric, it is treated as the number 0 for numeric operations. Note that UCL only supports whole (integer) values. Signs (plus or minus) are allowed, but trying to include a fractional point (a decimal point) means it will be interpreted as a string (thus, 0) for numeric operations. Note that any string that begins with an uppercase or lowercase T or Y is considered to have a numeric value of 1.

String operations work with any type of value. Booleans are treated as "0" or "1", and numeric values are nothing more than a string of decimal digits. However, it is important to note that the type of operator used with a numeric value can affect the result of the operation. Consider the following two cases:
"0100" .LTS. "10"
"0100" .LT. "10"

Because the first operation is a string comparison, the values are handled as a string of characters and the result is true because a string beginning with "0" sorts before a string beginning with "1" and therefore "0100" is less than "10". However, in the second example the result is false because the values are interpreted as numbers instead of strings of characters, and 100 is not less than 10. Also consider the following comparison:
"A" .LT. "10"

Although "A" sorts after "1" as a string, because .LT. operates on numeric values, and "A" is not numeric, the "A" is interpreted as a 0, which is less than 10 so the result of the expression is true.

Coercing precedence
Parentheses can be used to alter the order of evaluation to be different than that which results from precedence. Consider the following two expressions:
1+2*3
(1+2)*3

Because of order or precedence, the first expression evaluates to 7. In the second expression, the result is 9 because the contents between the parentheses are evaluated as if it had the highest precedence.

Further comments
Internally, integers are 64-bits in UCL.

UCL has no typing. Values are treated as integers when integers are required and as strings otherwise. Function parameters are also typeless, although errors can occur if the function expects a valid number but doesn't receive one.

Syntax specification
Here is the Backus-Naur form (BNF) definition of UCL expression syntax:

```
expression ::= subexpression | subexpression operator expression

subexpression ::= function | value | unary expression | value operator
expression | ( expression )

value ::= symbol | literal

unary ::= - | + | .NOT.

symbol ::= letter | letter alphanum

letter ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q |
R | S | T | U | V | W | X | Y | Z

alphanum ::= letter alphanum | digit alphanum

digit ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0

operator ::= + | - | / | * | logical | comparison

logical ::= .AND. | .OR.

comparison ::= numeric-comparison | string-comparison

numeric-comparison ::= .GE. | .LE. | .GT. | .LT. | .NE. | .EQ.

string-comparison ::= .GES. | .LES. | .GTS. | .LTS. | .NES. | .EQS.

literal ::= " anychar " | number

number ::= decimal | hexadecimal

decimal ::= digit | digit decimal

hexadecimal ::= %X hexdigits

hexdigits ::= hexdigit | hexdigit hexdigits

hexdigit ::= digit | A | B | C | D | E | F

function ::= F$ letter alphanum ( parameters )

parameters ::= expression | expression , parameters
```

Labels

Labels

Labels are used to mark target locations in command files for the use of certain commands. A label is a symbol starting with a letter and consisting of alphanumeric values, and terminated with a colon (:).

Although not recommended, the same label name can be used multiple times. If duplicate labels exist in a command file, and the label name is referenced, UCL will transfer control to the label most recently processed. If the label has not been encountered at the time it is referenced, the entire command file is searched, starting at the beginning of the file.

Any symbol name followed by a colon is treated as a label even if there is a symbol of the same name. Labels are ignored when encountered, other than having UCL note where the label was encountered.

Example

```
$ Retry:
```


= (Assignment)

= (Assignment)

This assigns a value to a symbol.

Format

symbol{[position,size]} {:}={=} value

Description

UCL has two symbol scopes: local and global. The local symbols are stored in the process symbol table and the global symbols are stored in the job symbol table. For the root process in a process tree (the job process), these symbols are the same. For subprocesses, the local symbols are different than the global symbols When one is referenced, the local symbol will be used if it exists. If not found, the global symbol is used. The symbol need not be defined when a value is assigned to it. If it doesn't exist, it will be created. The use of the = (single equal) will assign a value to a local symbol, whereas the == (double equal) will assign the value to a global symbol.

If the equals is prefixed with a colon (:), the value being assigned to the symbol is assumed to be a string - otherwise if it can be interpreted as an integer value, it is converted to normalized decimal form before the symbol is set. That is, any leading "0"s are removed and any hex values (specified with "%X") are converted to decimal.

When using the string assignment (:= or :==), you can also specify a substring to set within the existing symbol. The format is:

symbol[offset,length] := value

where "symbol" is the symbol's name, "offset" is the starting offset (0 = first character), and "length" is the number of characters to replace in the symbol's value. Negative values are not allowed for the offset or length. A length of 0 will result in no change to the symbol. In fact, if the symbol doesn't exist, assigning a substring with a length of 0 will not result in the symbol being created - it is a null operation. If the offset plus the length is greater than the length of the symbol's existing value, that value will be extended with space characters to be the necessary width. Likewise, if the specified length is less than

the length of the value being assigned, the assigned value is space-filled at the end to be the appropriate length. If the length is less than the length of the assigned value, only the specified number of bytes are copied to the symbol.

The following table illustrates the different assignment operators.

Operator	Local/Global	Example	Resulting symbol value
=	Local	A="01234"	1234
==	Global	A=="01234"	1234
:=	Local	A:="01234"	01234
:==	Global	A:=="01234"	01234

Examples

```
X = "Text"
```
Assigns the local symbol X.

```
X == "Global text"
```
Assigns the global symbol X.

```
$ X = "3"
```
The local symbol X is set to "3".

```
$ X[3,1] := "0"
```
The existing local symbol X is set to "3 0".

```
$ X[2,3] := "ABCD"
```
The local symbol X is now "3 ABC". Note that only "ABC" is copied into X because the specified length is 3. The second space and "0" previously existing in X are overwritten with "AB", and the length of X is expanded to include "C".

! (Comment)

! (Comment)

Indicates that the remainder of the line is a comment and should be ignored by UCL. You cannot use command continuation on a line that has a comment since the dash is interpreted as part of the comment itself.

Comments
Comments help to document scripts. They begin with an exclamation point (!) and extend to the end of the line. They can be on a line by themselves or after some other script code.

Format
! comment text

Example
$ WRITE SYS$OUTPUT FILENAME ! Show the user the next filename

Lexical Functions

Lexical Functions

UCL lexical functions are constructs which return information and provide access to some system services. They are called lexical functions because the command interpreter evaluates them during the command input scanning (lexical parsing) phase of command execution.

Lexical functions can be used in any context in which symbols or expressions can be used.

The general format of a lexical function is:

F$function-name(arguments)

F$	Indicates that a lexical function name follows.
function-name	The name of the function. This may be any unique abbreviation.
()	Parentheses are required for all functions, even if they take no arguments.
arguments	One or more arguments, depending upon the function. If there are multiple arguments, they must be delimited by commas.

A detailed description of each function, including examples, is given in the following sections.

F$CONTEXT

F$CONTEXT

Each call to F$CONTEXT will define a filter to be applied to a context to be used with the F$PID function. F$CONTEXT can be called as many times as needed to produce the criteria needed. Lists of item values are allowed.

Format

F$CONTEXT(type, symbol, criteria, value, qualifier)

Return Value

The function returns a null string ("").

Arguments

type

Specifies the context type. At present, the only context type available is "PROCESS", which is used for constructing selection criteria for F$PID.

symbol

Specifies a symbol that UCL will use to refer to the context object that it constructs to hold the contexts. After the context is set up, this symbol should be used when calling F$PID. Multiple contexts can exist simultaneously, each one specified with a different symbol.

If the symbol is not yet defined or doesn't refer to a valid UCL context, a new context is created and the address is assigned to the symbol. If the symbol already refers to a valid UCL context, the F$CONTEXT call will add another filter to the existing context. Once F$PID is used with this context, the context is "frozen" and further attempts to add more filters will result in an error, although the context will remain valid and can continue to be used. To cancel a context, use the CANCEL "criteria".

criteria

This is a keyword that specifies the criteria for the filter to use.

Criteria	Value	Comments

	pe	
ACCOUNT	String	Valid account name or list of names.
AUTHPRI	Integer	The authorized base priority.
CANCEL		Cancels the selection criteria for this context.
CURPRIV	Keyword	Valid privilege name keyword or list of keywords.
HW_MODEL	Integer	Valid hardware model number.
HW_NAME	String	Valid hardware name.
JOBPRCCNT	Integer	Subprocess count for entire job.
JOBTYPE	Keyword	Valid job-type keyword. Valid keywords are DETACHED, NETWORK, BATCH, LOCAL, DIALUP, and REMOTE.
MASTER_PID	String	PID of master process.
MODE	Keyword	Valid process mode keyword. Valid keywords are OTHER, NETWORK, BATCH, and INTERACTIVE.
NODE_CSID	Integer	Node's cluster ID number.
NODENAME	String	Node name or list of node names. The default is the local node. To request all nodes, use the value "*".
OWNER	String	PID of immediate parent process.
PRCCNT	Integer	Subprocess count of process.
PRCNAM	String	Process name.
PRI	Integ	Process priority level number.

	er	
PRIB	Integer	Base process priority level number.
STATE	Keyword	Valid process state keyword.
STS	Keyword	Valid process status keyword.
TERMINAL	String	Terminal name or list of names.
USERNAME	String	User name or list of user names.

value

Specifies the value of the filter's criteria. For example, to iterate through all of the processes running under the username "SYSTEM", specify "USERNAME" with "SYSTEM", like so:
$ X = F$CONTEXT("PROCESS",context,"USERNAME","SYSTEM","EQL")

Multiple values can be used for a given criteria. For instance, if you wanted to iterate over all processes with ther username "SYSTEM", or "ALEX", or "MARKETING", you could use the following:
$ X = F$CONTEXT("PROCESS",context,"USERNAME","SYSTEM,ALEX,MARKETING","EQL")
In such a case, a match on any of the values is considered a match (as if an OR operand was used). Each different filter in the context is considered an AND operation - meaning that all filters must match, but each filter can be multiple values treated as an OR.

Further, wildcards can be used with any string comparison. Both the asterisk (*) and question mark (?) wildcards are allowed. For example, to iterate through all processes with a username starting with "A", you could use the following:
$ X = F$CONTEXT("PROCESS",context,"USERNAME","A*","EQL")

Or if you wanted all processes whose username was "SYSTEM" or which started with A:
$ X = F$CONTEXT("PROCESS",context,"USERNAME","A*,SYSTEM","EQL")

qualifier

Specifies the comparison to be made for the criteria. Note that although any qualifier can be used with any criteria, some of them may not be useful. For instance, ALL and ANY are used to compare bit values - primarily intended for privilege mask comparisons.

Qualifier	**Description**
LSS	Less than the value specified in the call to F$PID
LEQ	Less than or equal to the value specified in the call to F$PID
GTR	Greater than the value specified in the call to F$PID

GEQ	Greater than or equal to the value specified in the call to F$PID
EQL	Equal to the value specified in the call to F$PID
NEQ	Not equal to the value specified in the call to F$PID
ALL	Requires that all bits in the value be set for a process
ANY	Requests that any bits in the value be set for a process

Example

$ X = F$CONTEXT("PROCESS",context,"USERNAME","A*,SYSTEM","EQL")

F$CUNITS

F$CUNITS

F$CUNITS converts a number from one specified unit of measure to another.

Format

F$CUNITS(number {,from, to})

Return Value

The converted value. If the value is greater than 1,000, the value is adjusted and suffixed with "KB", "MB, "GB", or "TB".

Arguments

number

Specifies a 64-bit (or smaller) integer value to convert.

from

Optional unit of measure from which to convert. The only supported option for this field is "BLOCKS".

to

Optional unit of measure to which to convert. The only supported option for this field is "BYTES".

Example

$ A = F$CUNITS(100, "BLOCKS", "BYTES")

This example converts 100 blocks to the equivalent number of bytes. The result is "51.2KB".

F$CSVI

F$CVSI

F$CVSI converts the specified bits in a character string to a signed integer value.

Format

F$CVSI(start, count, string)

Return Value
The selected bits from the string converted into a signed integer value.

Arguments
start
The offset of the first bit to be extracted. The least significant (rightmost) bit of a string is position 0. The starting bit must be an integer expression.
If the start value is negative or is a value that exceeds the number of bits in the string, an INVRANGE error is generated.

count
The number of bits from the string to be extracted, starting at the specified start bit.
If the value is negative or if this value plus the start exceeds 64 or the number of bits in the string, an INVRANGE error is generated.

string
Specifies the string from which the bits are taken.

Examples

Example 1:
$ X = F$CSVI(0,4,"+")

In this example, the string contains a plus sign, which has a hex value of 2B. Since we are requesting 4 bits starting with bit 0, we are addressing the low four bits (or hexadecimal B). Because the highest of the 4 bits is set, the sign is extended and the result is -5.

$ X = F$CSVI(0,8," ")

In this example, the string contains a space, which has a value of 32. We grab all 8 bits of the value. Since the highest bit is not set, there is no sign that is set. Thus, the result is 32.

Example 2:
$ X = F$CSVI(0,32,"ABC")

In this example, the string contains three characters, which means there are 24 bits. Since the bit count is larger than the number of available bits, this generates the INVRANGE error.

F$CVUI

F$CVUI

F$CVUI converts the specified bits in a character string to an unsigned integer value.

Format
F$CVUI(start, count, string)

Return Value
The selected bits from the string converted into an unsigned integer value.

Arguments
start
The offset of the first bit to be extracted. The least significant (rightmost) bit of a string is position 0. The starting bit must be an integer expression.

If the start value is negative or is a value that exceeds the number of bits in the string, an INVRANGE error is generated.

count
The number of bits from the string to be extracted, starting at the specified start bit.

If the value is negative or if this value plus the start exceeds 63 or the number of bits in the string, an INVRANGE error is generated.

string
Specifies the string from which the bits are taken.

Examples
Example 1:
$ X = F$CSUI(0,4,"+")

In this example, the string contains a plus sign, which has a hex value of 2B. Since we are requesting 4 bits starting with bit 0, we are addressing the low four bits (or hexadecimal B), which is 11.

Example 2:
$ X = F$CSUI(0,32,"ABC")

In this example, the string contains three characters, which means there are 24 bits. Since the bit count is larger than the number of available bits, this generates the INVRANGE error.

F$CVTIME

F$CVTIME

F$CVTIME converts an absolute or combination time string to an absolute, delta, or a string of the form yyyy-mm-dd hh:mm:ss.cc. All arguments are optional,

Format
F$CVTIME({input_time},{output_format},{output_time})

Return Value
A string containing the requested information.

Arguments
input_time
Specifies a string containing an absolute, delta, or combination time. "TODAY", "TOMORROW", and "YESTERDAY" are also allowed.
If this argument is omitted or a null string (""), the current system date and time is used in its place. Any parts of the date field which are omitted are defaulted to the current date. Any parts of the time field which are omitted are defaulted to 0. Arguments can be omitted to the right of the last argument specified, but commas (,) must be used as placeholders if arguments are omitted to the left of the last argument specified.
Note: if this argument is a delta time, the output_format argument must be "DELTA".

output_format
Specifies the format to be returned. It must be one of the following values:

ABSOLUTE	The date/time is returned as per the default date and/or time format. Single digit days are returned with no leading space or zero.
COMPARISON (default)	The date and time is returned in the form yyyy-mm-dd hh:mm:ss.cc. This format can be used for comparing two date/times. If the output_format is omitted or null (""), it is assumed to be a request for a comparison date/time result.
DELTA	The request is returned in delta format, which is dddd-hh:mm:ss.cc. If this option is used, the input_time argument must be a delta time specification. This must be used if the input_time argument is a delta time.

output_time
Specifies a string containing one of the following (which may not be abbreviated): DATE, MONTH, DATEFIME, SECOND, DAY, TIME, HOUR, WEEKDAY, HUNDREDTH, YEAR, MINUTE, DAUPF YEAR, HOUROFYEAR, MINUTEOFYEAR, SECONDOF YEAR.
If this argument is omitted or null (""), the default is DATETIME.

If the input_time argument is a delta time and the output_format argument is "DELTA", then this argument cannot be MONTH, WEEKDAY, YEAR, DAYOFYEAR, HOUROFYEAR, MINUTEOFYEAR, or SECONDOFYEAR.

Examples
$ X = F$CVTIME()

This is equivalent to:
$ X = F$CVTIME("","COMPARISON","DATETIME") In this example, the current date/time is returned in the form yyyy-mm-dd hh:mm:ss.cc, for instance, "21-DEC-2019 11:55:12.20".

The resulting string can be compared using UCL operators (for instnace, .LTS. and .GTS.).

Example
$ X = F$CVTIME("YESTERDAY",,"WEEKDAY")

In this example, F$CVTIME returns the weekday that corresponds to the input time of "YESTERDAY". Thus, if the current day is tuesday, the function will return "Monday".

F$DELTA_TIME

F$DELTA_TIME

F$DELTA_TIME returns the time difference between to dates.

Format
F$DELTA_TIME(start_time,end_time)

Return Value
A string containing a delta time specification which indicates the difference between the start and end times. The string has the following format:
+dddd hh:mm:ss.cc

Arguments
start_time
Specifies a string containing an absolute or combination time indicating the starting date/time. "TODAY", "TOMORROW", and "YESTERDAY" are also allowed.

end_time
Specifies a string containing an absolute or combination time indicating the ending date/time. "TODAY", "TOMORROW", and "YESTERDAY" are also allowed.

Example
$ X = F$DELTA_TIME("1-JAN-2019 10:10:00","1-JAN-2019 10:30:01")

This would result in X containing "+0:0:20:01".

F$DELETE

F$DEVICE

F$DEVICE iterates through system devices.

Format

F$DEVICE({devnam}, {devclass}, {devtype}, {context})

Return Value

The name of the next device matching the criteria. After the last device name in the device list is returned, the function returns a null string ("").

Arguments

devnam

Specifies a string containing the name of the device to search for. The asterisk (*) and question mark (?) wildcards are allowed. If omitted, all devices are searched.

devclass

Specifies a string containing the device class to match. This must be one of the following values:

Value	Description
any	Match any device
disk	Disks
tape	Tapes
card	Cards
term	Terminals
lp	Printers
realtime	Real-time devices
audio	Audio devices
video	Non-terminal video devices
mailbox	Mailboxes
remcsl_storage	Remote storage
misc	All other devices

devtype

Specifies a string containing the device type to match. This is included for compatiblity with VMS, but is ignored by UOS. The value of this argument has no effect on the operation of the function.

context

An integer value representing the device scan context. This is used to maintain separate search contexts when F$DEVICE is used more than once in a command procedure. If omitted, a default wildcard context is used.

If the same context value is specified but different criteria are specified, then that context is reset and the first matching device is returned.

Description

F$DEVICE allows you to search or devices matching certain criteria.

F$DEVICE can be used in a loop to return all device names that match the selection criteria. Note that the order the devices is returned cannot be assumed to follow any particular order. After the last device name is returnd, the next F$DEVICE call returns a null string.

You must maintain the context of the search explicitly by specifying the context and using the same selection criteria in each F$DEVICE call.

Example

$ X = F$DEVICE("","DISK",,) This would return the first/next disk device. Because no context is specified, F$DEVICE uses an implicit context.

F$DIRECTORY

F$DIRECTORY

Returns the current default directory name. F$DIRECTORY has no arguments, but the parentheses must follow the function name.

Format

F$DIRECTORY()

Return Value

The name of the current default directory name.

Arguments

None.

Description

The F$DIRECTORY lexical function can be used to save the name of the current default directory so that it can be used to restore the default directory later.

Example

$ X = F$DIRECTORY() This would return the process' current default directory.

F$EDIT

F$EDIT

F$EDIT returns the specified string with the specified edits.

Format

F$EDIT(string, edit-values)

Return Value

A character string containing the specified edits.

Arguments

string

The value to be edited. Quoted sections of the string are not modified.

edit-values

Specifies a string containing one or more of the following keywords:

Value	Meaning
COLL	Remove all spaces and tabs.

APS E	
COM PRE SS	Replace multiple spaces or tabs with a single space
LOW ERC ASE	Change all uppercase characters to lowercase
TRIM	Remove leading and trailing spaces and tabs
UNC OMM ENT	Remove comments
UPC ASE	Change all lowercase characters to uppercase

If more than one edit value is specified, separate them with commas (,). The values may not be abbreviated. Any characters within quotation marks (") are unmodified. If both LOWERCASE and UPCASE are specified, UPCASE overrides LOWERCASE. Comments begin with any exclamation (!) that is not within quotes.

Example

```
$ X = F$EDIT(" Hello world ! This is a comment","LOWERCASE,TRIM,UNCOMMENT")
```


F$ELEMENT

F$ELEMENT

F$ELEMENT returns one element from a string of elements.

Format

```
F$ELEMENT(index, delimiter, string)
```

Return Value

A character string containing the specified element.

Arguments

index

Specifies the zero-based index of the element to extract. This is an integer expression. If this value exceeds the number of elements in the string, the function returns the delimiter.

delimiter

This indicates the character used to delimit the elements. Note that this must be a single Unicode character of less than 128.

string

This specifies the string of delimited elements.

Example

```
$ X = F$ELEMENT(1,",","A,B,C,D,E,F,G,H,I,J")
```

This would return a value equal to "B".

F$ENVIRONMENT

F$ENVIRONMENT

F$ENVIRONMENT returns information about the current UCL/process environment.

Format

F$ENVIRONMENT(item)

Return Value

The information corresponding to the specified item.

Arguments

item

A keyword indicating the type of information to return. The following are the valid keywords:

Value	Meaning
CAPTIVE	TRUE if the process is logged into a captive account.
CONTROL	The control characters currently enabled. Multiple characters are delimited with commas. Null is returned if no control characters are enabled.
DEFAULT	The current default disk and directory.
DISIMAGE	TRUE if the logged-in account does not allow direct image invocation (eg the RUN command).
INTERACTIVE	TRUE if the process is interactive.
KEY_STATE	Current locked keypad state.
MAX_DEPTH	Maximum allowable command procedure depth.
MESSAGE	Current setting of SET MESSAGE qualifiers.
NOCONTROL	The control characters currently disabled. Multiple characters are delimited with commas. Null is returned if no control characters are disabled.
ON_CONTROL_Y	Within a command procedure, this returns TRUE if ON_CONTROL_Y is set and FALSE otherwise.
ON_SEVERITY	Within a command procedure, this returns the severity level at which the action specified with the ON command was specified.
OUTPUT_RATE	Delta time of the batch job default output rate. Returns null if used interactively.
PROCEDURE	Specification of current command file. If interactive, the terminal device name is returned.
PROMPT	Current DCL prompt.
PROMPT_CONTROL	TRUE if prompt is preceded by a CRLF.
PROTECTION	Current default file protection.
RESTRICTED	TRUE if in a restricted account, FALSE otherwise.
SYMBOL_SCOPE	Indicates the current symbol scoping state: [NO]LOCAL or [NO]GLOBAL.
VERB_SCOPE	Indicates the current verb scoping state: [NO]LOCAL or [NO]GLOBAL.
VERIFY_IMAGE	TRUE if image verification is set. FALSE otherwise.
VERIFY_PREFIX	Returns prefix control string.
VERIFY_PROCEDURE	True if SET_VERIFY=PROCEDURE is set.

Example

```
$ X = F$ENVIRONMENT("PROMPT")
```

This would return the current UCL prompt.

F$EXTRACT

F$EXTRACT

F$EXTRACT extracts the specified characters from the specified string.

Format

F$EXTRACT(start,length,value)

Return Value

A string containing the characters delimited by the start and length arguments. The source value is not modified.

Arguments

start

Specifies the offset of the starting character of the value that you want to extract. The offset of the first character is 0.

length

Specifies the number of characters from the value that you want to extract. The length is an integer expression that is 0 or larger. If the length exceeds the number of characters from the offset to the end of the string, F$EXTRACT returns the characters from the start to the end of the string.

value

Specifies the character string to extract characters from.

Example

```
$ X = F$EXTRACT(0,4,"ABCDEFG")
```

This would set X to "ABCD".

F$FAO

F$FAO

F$FAO converts character and numeric input to character strings. Note that while DCL generates ASCII strings, in UCL this function generates UTF-8 strings.

Format

F$FAO(control,argument{,...})

Return Value

A string containing formatted output created from the control argument and the optional arguments.

Arguments

control

Specifies the fixed text of the output string and formatting directives. The directives are described below.

argument{,...}

Specifies arguments that correspond to FAO directives in the control string. The arguments may be integer or string. The order of the arguments must correspond exactly with the order of the directives in the control string, even if some directives require multiple arguments.

Directives

The following directives are supported by F$FAO. String directives:

Directive	Description
!AS	String

Zero-filled numeric directives:

Directive	Description
!BB	Convert a byte value to the ASCII representation of that value in base 2.
!BW	Convert a word value to the ASCII representation of that value in base 2.
!BL	Convert a longword value to the ASCII representation of that value in base 2.
!BQ	Convert a quadword value to the ASCII representation of that value in base 2.
!OB	Convert a byte value to the ASCII representation of that value in base 8.
!OW	Convert a word value to the ASCII representation of that value in base 8.
!OL	Convert a longword value to the ASCII representation of that value in base 8.
!OQ	Convert a quadword value to the ASCII representation of that value in base 8.
!XB	Convert a byte value to the ASCII representation of that value in base 16.
!XW	Convert a word value to the ASCII representation of that value in base 16.
!XL	Convert a longword value to the ASCII representation of that value in base 16.
!XQ	Convert a quadword value to the ASCII representation of that value in base 16.
!ZB	Convert a byte value to the ASCII representation of that value in base 10.
!ZW	Convert a word value to the ASCII representation of that value in base 10.
!ZL	Convert a longword value to the ASCII representation of that value in base 10.
!ZQ	Convert a quadword value to the ASCII representation of that value in base 10.

Blank-filled numeric directives:

Direct	Description

Directive	Description
!UB	Convert an unsigned byte value to the ASCII representation of that value in base 10.
!UW	Convert an unsigned word value to the ASCII representation of that value in base 10.
!UL	Convert an unsigned longword value to the ASCII representation of that value in base 10.
!UQ	Convert an unsigned quadword value to the ASCII representation of that value in base 10.
!SB	Convert a signed byte value to the ASCII representation of that value in base 10.
!SW	Convert a signed word value to the ASCII representation of that value in base 10.
!SL	Convert a signed longword value to the ASCII representation of that value in base 10.
!SQ	Convert a signed quadword value to the ASCII representation of that value in base 10.

Other Directives:

Directive	Description
!/	Inserts a new line (carriage return and linefeed). It takes no parameters.
!_	Inserts a horizontal tab (ASCII 9). It takes no parameters.
!^	Inserts a form feed. It takes no parameters.
!!	Inserts an exclamation point. It takes no parameters
!%S	Inserts the letter S if the most recently converted numeric value is not 1. If the character before the directive is upper case, an upper case S is inserted, otherwise a lowercase s is inserted.
!%T	Inserts the system time. The parameter is the datetime stamp. If the parameter is 0, the current time is inserted.
!%U	Same as !UQ.
!%I	Converts a UIC to the account name. If an invalid UIC is specified, the directive is treated as !UQ.
!%D	Inserts the system date and time. The parameter is the timestamp. If the parameter is 0, the current date/time is inserted.
!n%C	Conditional. See discussion of conditionals below.
!%E	Else portion of conditional. See discussion of conditionals below.
!%F	End of conditional. See discussion of conditionals below.
!n<	See next directive.

!>	The preceding directive and this one are used together to define an output field that has a width of n. Within this field are displayed all directives between the !n< and !> directives. The field is blank-filled on the right to make it n characters wide if necessary. All directives within this field are left-justified and blank-filled. Note that these can be nested.
!n *c	Repeats the character c in the output n times.
!-	Reuse the most recently used parameter value.
!+	Skip the next parameter value.

Conditionals

!%nC, !%E, and !%F are used together to insert values depending upon parameter values. This is primarily for use with plurals. The general format is:
!%nCa!%Eb!%F

If n matches the last parameter value, then a is inserted, otherwise b is inserted. Example:
!ZB !%1Cchild!%Echildren!%F

In this example, if the first parameter is 1, the output would be:
1 child

But if the first parameter is not 1, the output would be:
n children
where "n" is the value of the first parameter.

Width and Filling

The following table illustrates how the directives interact with width and filling.

Directive Type	Default output width	When explicit width is greater than default	When explicit width is less than default
!BB	8	Right justify and blank fill	Result truncated on left
!BW	16	Right justify and blank fill	Result truncated on left
!BL	32	Right justify and blank fill	Result truncated on left
!BQ	64	Right justify and blank fill	Result truncated on left
!OB	3	Right justify and blank fill	Result truncated on left
!OW	6	Right justify and blank fill	Result truncated on left
!OL	11	Right justify and blank fill	Result truncated on left
!OQ	22	Right justify and blank fill	Result truncated on left
!HB	2	Right justify and blank fill	Result truncated on left
!HW	4	Right justify and blank fill	Result truncated on left
!HL	8	Right justify and blank fill	Result truncated on left
!HQ	16	Right justify and blank fill	Result truncated on left
Unsigned zero-filled decimal	As many charact	Right justify and blank fill	Field completely filled with asterisks (*)

	ers as are necessary		
Signed or unsigned decimal	As many characters as are necessary	Right justify and zero-filled	
Strings	As many characters as in the string	Left justify and blank fill to specified length	Truncate on right

Example

$ X = F$FAO("NUMBER OF FILES: !SL",COUNT)

This would insert the numeric value of COUNT into the string at the point where "!SL" occurs. For instance, if COUNT was equal to 105, the resulting string would be "NUMBER OF FILES: 105".

F$FILE_ATTRIBUTES

F$FILE_ATTRIBUTES

F$FILE_ATTRIBUTES returns attribute information for a specified file.

Format

F$FILE_ATTRIBUTES(filespec,item)

Return Value

Either an integer or a string, depending upon the item requested. The table below indicates the return values.

Arguments

filespec

Specifies the name of the file for which you are requesting information.

item

Specifies the type of information to return. The table below indicates the item values.

Item	Information Returned
AI	TRUE if after-image journaling is is enabled, FALSE otherwise.
ALQ	RMS allocation quantity.
BDT	Backup date/time.
BI	TRUE if before-image journaling is enabled, FALSE otherwise.
BKS	RMS bucket size.
BLS	RMS block size.
CBT	TRUE if contiguous-best-try, FALSE otherwise.
CDT	Creation date/time.
CTG	TRUE if contiguous, FALSE otherwise.
DEQ	RMS default extension quantity.
DID	This doesn't have any meaning in UOS. It returns null.

DIRECTORY	TRUE if file is a directory, FALSE otherwise.
DVI	Name of the device the file resides on.
EDT	Expiration date/time.
EOF	Logical end of file - logical file length in bytes.
ERASE	TRUE if file is erased when deleted, FALSE otherwise.
FFB	RMS First Free Byte.
FID	Always returns 0.
FILE_LENGTH_HINT	RMS file length hint.
FSZ	RMS fixed control area size.
GBC	Global buffer count.
GBC32	Enhanced global buffer count.
GBCFLAGS	Global buffer cache flags. Returns PERCENT, DEFAULT, or NONE.
GRP	Group name for file.
JOURNAL_FILE	TRUE if this is a journal file, FALSE otherwise.
KNOWN	TRUE if this file was installed with the Install utility, FALSE otherwise.
LOCKED	TRUE if the file is deaccessed-locked, FALSE otherwise.
LRL	RMS longest record length.
MBM	Always 0.
MOVE	TRUE if movefile operations are enabled, FALSE otherwise.
MRN	RMS maximum record number.
MRS	Maximum record size (record size for fixed-format RMS and non-RMS files).
NOA	RMS number of areas.
NOBACKUP	FALSE if the file is marked for backup, TRUE otherwise.
NOK	RMS number of keys.
ORG	RMS file organization: SEQ, REL, IDX. Returns null if not an RMS file.
PRESHELVED	TRUE if file is pre-shelved, FALSE otherwise.
PRO	File protection string.
PVN	RMS prolog version number.
RAT	RMS record attributes. Returns null if not an RMS file.
RCK	TRUE if file is marked Read-check, FALSE otherwise.
RDT	Revision date/time.
RFM	RMS record format string: VAR, FIX, VFC, UDF, STM, STMLF, or STMCR. Returns null if not an RMS file.
RU	TRUE if recovery unit journaling is enabled, FALSE otherwise.
RVN	Revision number.
SHELVABLE	TRUE if file is shelvable, FALSE otherwise.
SHELVED	TRUE if file is shelved, FALSE otherwise.
STORED_SEMANTICS	RMS stored semantics.
UIC	File owner UIC.
VERLIMIT	Version limit number. 0 indicates no version limit.

WCKTRUE	TRUE if file is write-checked, FALSE otherwise.

Example:

$ FILE_EOF = F$FILE_ATTRIBUTES("CALC.EXE","EOF")

This would assign the logical file size, in bytes, to the FILE_EOF symbol.

F$GETDVI

F$GETDVI returns a specific piece of information about the specified device.

Format

F$GETDVI(devicename,item{,pathname})

Return Value

Either an integer or a string, depending upon the item requested. The table below indicates the valid items and the corresponding return values.

Arguments

devicename

Specifies the name of the device for which you are requesting information.

item

Specifies the type of device information to return. See the table below.

pathname

Specifies the path name for a multipath-capable device. If specified, it is validated against the existing paths for the specified device. If such path does not exist, an error is returned - even if the requested item doesn't make use of the path name.

Description

F$GETDVI invokes the GETDVI system service to return information on the specified device. Use F$DEVICE to iterate through available devices. In addition to the items returned by the GETDVI system service, "EXISTS" is also a valid item for F$GETDVI. The following table lists all valid items.

Item	Information returned
ACCESSTIME_RECORDED	TRUE if volume supports recording of access times. FALSE otherwise.
ACPPID	Ancillary control process (ACP) identification.
ACPTYPE	ACP type code: F11V1, F11V12, F11V3, F11V4, F11V5, F64, HBS, JNL, MTA, NET, REM, UCX, or ILLEGAL. ILLEGAL is returned if the device is not mounted, mounted using /FOREIGN, or the ACPTYPE is not defined.
ALL	TRUE if the device is allocated. FALSE otherwise.
ALLDEVNAM	Allocation class device name
ALLOCLASS	Allocation class of the host.
ALT_HOST_AVAIL	TRUE if host serving the alternate path is available. FALSE otherwise.
ALT_HOST_NAME	Name of the host serving the alternate path
ALT_HOST_TYPE	Hardware type of the host serving the alternate path.
AVAILABLE_P	Number of available working paths for a multipath-capable device.

ATH_COUNT	
AVL	TRUE if device is available for use. FALSE otherwise.
CCL	TRUE if the device is a carriage control device. FALSE otherwise.
CLUSTER	Volume cluster size.
CONCEALED	TRUE if the logical device name translates to a concealed device. FALSE otherwise.
CYLINDERS	Number of cylinders on the volume (local disks only).
DEVBUFSIZ	Device buffer size.
DEVCHAR	Device characteristics.
DEVCHAR2	Additional device characteristics.
DEVCLASS	Device class.
DEVDEPEND	Device dependent information.
DEVDEPEND2	Additional device dependent information.
DEVICE_MAX_IO_SIZE	The maximum contiguous transfer size supported by the device driver. Note that other software layers, such as RMS, may impose a smaller size.
DEVICE_TYPE_NAME	Device type name. For some hardware, such as SCSI, the device type is retrieved from the hardware device itself.
DEVLOCKNAM	A unique lock name for the device.
DEVNAM	Device name.
DEVSTS	Device status.
DEVTYPE	Device type.
DFS_ACCESS	TRUE to indicate a remote virtual disk. FALSE otherwise.
DIR	TRUE if the device is file structured. FALSE otherwise.
DMT	TRUE if device is marked for dismount. FALSE otherwise.
DUA	TRUE if device is generic. FALSE otherwise.
ELG	TRUE if device has error logging enabled.
ERASE_ON_DELETE	TRUE if files are zeroed upon deletion on the volume. FALSE otherwise.
ERRCNT	Error count on device. If pathname is specified, the count is only for that pathname, otherwise it is the aggregate for all paths.
ERROR_RESET_TIME	Time at which the error count was reset
EXISTS	TRUE if the device exists on the system. FALSE otherwise.
EXPSIZE	Current expansion limit on the volume.
FC_HBA_FIRMWARE_REV	Same as FIRMWARE_REV.
FC_NODE_NAME	Fibre Channel host bus adapter node name.
FC_PORT_NAME	Fibre Channel host bus adapter port name.
FIRMWARE_REV	Hardware firmware revision information. If hardware doesn't support this, a null string is returned.
FOD	TRUE if the device is a files-oriented device. FALSE otherwise.
FOR	TRUE if the device is mounted using /FOREIGN. FALSE otherwise.
FREEBLOCKS	Number of free space on the volume (disks only).
FULLDEVNAM	Fully qualified device name.
GEN	TRUE if device is generic. FALSE otherwise.
HARDLINKS_SUPPORTED	TRUE if volume's file system supports POSIX hardlinks. FALSE otherwise.
HOST_AVAIL	TRUE if host serving the primary path is available. FALSE otherwise.
HOST_COUNT	Number of hosts that make the device available to a UOS Cluster.

HOST_NAME	Name of the host serving the primary path.
HOST_TYPE	Hardware type of the host serving the primary path.
IDV	TRUE if the device is capable of producing input. FALSE otherwise.
LAN_ALL_MULTICAST_MODE	TRUE if the device is enabled to receive all multicast packets rather than only packets addressed to the enabled multicast addresses. FALSE otherwise.
LAN_AUTONEG_ENABLED	TRUE if device is set to autonegotiate the speed and duplex settings. FALSE otherwise.
LAN_DEFAULT_MAC_ADDRESS	The default MAC address of the device. Null is returned for non-network devices.
LAN_FULL_DUPLEX	TRUE if the device is operating in full-duplex mode. FALSE otherwise.
LAN_JUMBO_FRAMES_ENABLED	TRUE if jumbo frames are enabled. FALSE otherwise.
LAN_LINK_STATE_VALID	TRUE if the device driver correctly maintains the link status. FALSE otherwise.
LAN_LINK_UP	TRUE if the link is up. FALSE otherwise.
LAN_MAC_ADDRESS	Current MAC address of the device. Null is returned for non-network devices.
LAN_PROMISCUOUS_MODE	TRUE if device is enabled to receive all packets rather than those matching its MAC. FALSE otherwise.
LAN_PROTOCOL_NAME	Name of the LAN protocol running on the device. Null is returned for non-network devices.
LAN_PROTOCOL_TYPE	Type of LAN protocol running on the device. Null is returned for non-network devices.
LAN_SPEED	Speed of the LAN device in megabits per second. Null is returned for non-network devices.
LOCKID	Clusterwide lock identification.
LOGVOLNAM	Logical volume name.
MAILBOX_BUFFER_QUOTA	Current mailbox quota.
MAILBOX_INITIAL_QUOTA	Initial mailbox quota.
MAXBLOCK	Size of volume in bytes.
MAXFILES	Maximum number of files on volume (disks only). 0 if no limit.
MBX	TRUE if device is a mailbox. FALSE otherwise.
MEDIA_ID	Non-decoded media ID. Null if no media on device.
MEDIA_NAME	Name of media type.
MEDIA_TYPE	Device name prefix.
MNT	TRUE if the device is mounted. FALSE otherwise.
MOUNT_TIME	Time at which the device was mounted.
MOUNTCNT	Number of times the device has been mounted on the local system.
MOUNTVER_ELIGIBLE	TRUE if the volume is eligible to undergo mount verification. FALSE otherwise.
MPDEV_AUTO_PATH_SW_CNT	Number of times a multipath device has automatically switched paths due to I/O errors or due to falling back to a local path from a remote path.
MPDEV_CURRENT_PATH	Current path name for multipath devices. If the device is not part of a multipath set, this returns the name of the device path. Returns null string if device doesn't support multipathing.

MPDEV_MAN_PATH_SW_CNT	Number of times a multipath device has manually switched paths.
MT3_DENSITY	Current density of the tape.
MT3_SUPPORTED	TRUE if device supports densities. FALSE otherwise.
MULTIPATH	TRUE if the device is a member of a multipath set. FALSE otherwise.
MVSUPMSG	TRUE if mount verification OPCOM messages are currently suppressed on this device. FALSE otherwise.
NET	TRUE if device is a network device. FALSE otherwise.
NEXTDEVNAM	Device name of the next volume in a volume set (disks only).
NOCACHE_ON_VOLUME	TRUE if device is mounted with no caching. FALSE otherwise.
NOHIGHWATER	TRUE if high-water marking is disabled on the device. FALSE otherwise.
NOSHARE_MOUNTED	TRUE if the volume is mounted with /NOSHARE. FALSE otherwise.
ODS2_SUBSET0	TRUE if volume mounted on the device only supports a subset of the ODS-2 file structure.
ODS5	TRUE if volume is mounted OS5. FALSE otherwise.
ODV	TRUE if device is capable of providing output.
OPCNT	Operation count of the device. If the pathname is specified, only the operation count for that path is returned. Otherwise, the aggregate count for all paths is returned.
OPR	TRUE if the device is an operator. FALSE otherwise.
OWNUIC	Owner ID of the device owner.
PATH_AVAILABLE	TRUE if the specified path is available. FALSE otherwise. If pathname is omitted, information about the current path of the device is returned.
PATH_NOT_RESPONDING	TRUE if the specified path is marked as not responding. FALSE otherwise. If pathname is omitted, information about the current path of the device is returned.
PATH_POLL_ENABLED	TRUE if the specified path is enabled for multipath polling. FALSE otherwise. If pathname is omitted, information on the current path of the device is returned.
PATH_SWITCH_FROM_TIME	Time from which this path was switched.
PATH_SWITCH_TO_TIME	Time to which this path was switched.
PATH_USER_DISABLED	TRUE if specified path has been disabled. FALSE otherwise. If pathname is omitted, information on the current path of the device is returned.
PID	Process ID of the device owner.
PREFERRED_CPU	A 64-bit integer where the corresponding CPU index that is the preferred CPU for this device is set to 1.
PREFERRED_CPU_BITMAP	Equivalent to PREFERRED_CPU except that the result is a string of 0s and 1s corresponding to the bits returned by PREFERRED_CPU.
PROT_SUBSYSTEM_ENABLED	TRUE if the device is mounted with the protection subsystems enabled. FALSE otherwise.
QLEN	Current queue length of the device. This is the number of I/O requests in the device's driver.
RCK	TRUE if the device has reach checking enabled. FALSE otherwise.
RCT	TRUE if the disk contains RCT. FALSE otherwise.
REC	TRUE if the device is record oriented. FALSE otherwise.
RECSIZ	Current maximum blocked record size of device, in bytes.

REFCNT	Reference count of processes using the device.
REMOTE_DEVICE	TRUE if the device is remote. FALSE otherwise.
RND	TRUE if device allows random access. FALSE otherwise.
ROOTDEVNAME	Device name of the root volume in a volume set (disks only).
RTM	TRUE if the device is real-time. FALSE otherwise.
SCSI_DEVICE_FIRMWARE_REV	Firmware revision number of a SCSI device. Null returned for other devices.
SDI	TRUE if the device is single-directory structured. FALSE otherwise.
SECTORS	Number of sectors per track (disks only). 0 is returned for disks without those metrics available (such as virtual disks).
SERIALNUM	Volume serial number (disks only). For non-disks, if the device reports a serial number, it is returned here. Otherwise null.
SERVED_DEVICE	TRUE if this is a served device. FALSE otherwise.
SET_HOST_TERMINAL	TRUE if the device is a remote terminal from a remote node (via SET HOST). FALSE otherwise.
SHDW_CATCHUP_COPYING	TRUE if the device is a member that is the target of a full copy operation. FALSE otherwise.
SHDW_COPIER_NODE	The name of the node that is actively performing the copy or merge operation.
SHDW_DEVICE_COUNT	The total number of devices in the shadow set, including devices being added as copy targets.
SHDW_GENERATION	Current internal revision number for the shadow set.
SHDW_MASTER	TRUE if this device is a shadow set.
SHDW_MASTER_MBR	Name of master member unit used for merge, copy repair, and recovery operations.
SHDW_MASTER_NAME	Device name representing the name of the shadow set that this device belongs to.
SHDW_MBR_COPY_DONE	The percent of the copy operation completed on this shadow set member unit.
SHDW_MBR_COUNT	The total number of devices in the shadow set, not including devices being added as copy targets.
SHDW_MBR_MERGE	The percent of the merge operation complete on this member unit.
SHDW_MBR_READ_COST	Read cost value for this unit in the shadow set.
SHDW_MEMBER	TRUE if the device is a member of a shadow set. FALSE otherwise.
SHDW_MERGE_COPYING	TRUE if the device is a merge member of a shadow set.
SHDW_MINIMERGE_ENABLE	TRUE if the device will undergo a min-merge instead of a full merge, if a system in the cluster crashes.
SHDW_NEXT_MBR_NAME	Device name of the next member in the shadow set. If the virtual shadow set device name is passed, the first device in the shadow set is returned. If there are no more members, null is returned.
SHDW_READ_SOURCE	The name of the member unit that will be used for reads at this time. This will be the unit with the lowest sum of queue length and read cost.

SHDW_READ_SITE	The site value for the device. This is set by SET DEVICE or SET SHADOW.
SHDW_TIMEOUT	The user-specified timeout value st for the device. If not set, the system default value is used.
SHR	TRUE if the device is shareable. FALSE otherwise.
SPL	TRUE if the device is being spooled. FALSE otherwise.
SPLDEVNAME	Name of the device being spooled.
SQD	TRUE if the device is sequential block-oriented.
STS	Status information.
SWL	TRUE if the device is software write-locked. FALSE otherwise.
TOTAL_PATH_COUNT	Number of paths for a multipath device.
TRACKS	Number of tracks per cylinder (disks only). Will return 0 if the disk has no metrics (such as virtual disks).
TRANSCNT	Volume transaction count.
TRM	TRUE if the device is a terminal. FALSE otherwise.
TT_ACCPORNAM	The terminal server name and port name.
TT_ALTYPEAHD	TRUE if the terminal has an alternate type-ahead buffer.
TT_ANSICRT	TRUE if the terminal is an ANSI video terminal.
TT_APP_KEYPAD	TRUE if the terminal is in keypad applications mode.
TT_AUTOBAUD	TRUE if the terminal has automatic baud rate detection.
TT_AVO	TRUE if the terminal has AVO (VT100-family).
TT_BLOCK	TRUE if the terminal has block mode capability.
TT_BRDCSTMBX	TRUE if terminal uses mailbox broadcast messages.
TT_CHARSET	A bitmap indicating the coded character set supported by the terminal.
TT_CRFILL	TRUE if the terminal requires fill after CR.
TT_CS_HANGUL	TRUE if terminal supports the Hangul character set.
TT_CS_HANYU	TRUE if terminal supports the Hanyu character set.
TT_CS_HANZI	TRUE if terminal supports the Hanzi character set.
TT_CS_KANA	TRUE if terminal supports the Kana character set.
TT_CS_KANJI	TRUE if terminal supports the Kanji character set.
TT_CS_THAI	TRUE if terminal supports the Thai character set.
TT_DECCRT	TRUE if terminal is a DEC CRT terminal.
TT_DECCRT2	TRUE if terminal is a DEC CRT2 terminal.
TT_DECCRT3	TRUE if terminal is a DEC CRT3 terminal.
TT_DECCRT4	TRUE if terminal is a DEC CRT4 terminal.
TT_DIALUP	TRUE if terminal is connected to dial-up.
TT_DISCONNECT	TRUE if the terminal can be disconnected.
TT_DMA	TRUE if terminal has direct memory access mode.
TT_DRCS	TRUE if the terminal supports loadable character fonts.
TT_EDIT	TRUE if the terminal edit flag is set.
TT_EDITING	TRUE if advanced editing is enabled.
TT_EIGHTBIT	TRUE if the terminal uses 8-bit ASCII character set.

TT_ESCAPE	TRUE if terminal generates escape sequences.
TT_FALLBACK	TRUE if the terminal uses the multinational fallback option.
TT_HALFDUP	TRUE if terminal is in half-duplex mode.
TT_HANGUP	TRUE if the hangup flag is set.
TT_HOSTSYNC	TRUE if the terminal uses host/terminal sync communication.
TT_INSERT	TRUE if insert mode is the default line editing mode.
TT_LFFILL	TRUE if terminal requires fill after LF.
TT_LOCALECHO	TRUE if terminal has local echo set.
TT_LOWER	TRUE if the terminal supports lowercase letters.
TT_MBXDSABL	TRUE if mailboxes associated with the terminal will receive unsolicited input or input notification.
TT_MECHFORM	TRUE if terminal has mechanical FF.
TT_MECHTAB	TRUE if terminal has mechanical tabs (HT).
TT_MODEM	TRUE if terminal is connected to a modem.
TT_MODHANGUP	TRUE if the modify hangup characteristic is set.
TT_NOBRDCST	TRUE if terminal will not receive broadcast messages.
TT_NOECHO	TRUE if input characters are not echoed.
TT_NOTYPEAHD	TRUE if data must be solicited by a read operation.
TT_OPER	TRUE if terminal is an operations terminal.
TT_PAGE	Terminal page/screen height.
TT_PASTHRU	TRUE if PASALL mode with flow control is available.
TT_PHYDEVNAM	Physical device name associated with a handle or a virtual terminal.
TT_PRINTER	TRUE if the terminal has a printer port.
TT_READSYNC	TRUE if the terminal has read synchronization.
TT_REGIS	TRUE if the terminal has ReGID graphics.
TT_REMOTE	TRUE if terminal has established modem control.
TT_SCOPE	TRUE if the terminal is a video terminal.
TT_SECURE	TRUE if the terminal can recognize a secure server.
TT_SETSPEED	TRUE if the terminal line speed can be set.
TT_SIXEL	TRUE if sixel is supported.
TT_SYSPWD	TRUE if the system password is enabled for this terminal.
TT_TTSYNC	TRUE if there is terminal/host synchronization.
TT_WRAP	TRUE if CRLF should be inserted if the cursor moves beyond the right margin.
UNIT	Unit number of device.
VOLCHAR	128-bit string indicating the volume characteristics/capabilities. If a bit is set, the feature is supported.
VOLCOUNT	Count of volumes in a volume set.
VOLNAM	Volume name the device is part of.
VOLNUMBER	Number of the current volume in a volume set.
VOLSETMEM	TRUE if the device belongs to a volume set.
VOLSIZE	The current logical volume size.
VOLUME_EXTEND_QUANTIT	Number of bytes that files on the volume are extended by.

Y	
VOLUME_MOUNT_GROUP	TRUE if the volume is mounted /GROUP.
VOLUME_MOUNT_SYS	TRUE if volume is mounted /SYSTEM.
VOLUME_PENDING_WRITE_ERR	Number of pending write errors on device.
VOLUME_RETAIN_MAX	Maximum retention time for the volume (via SET VOLUME/RETENTION).
VOLUME_RETAIN_MIN	Minimum retention time for the volume (via SET VOLUME/RETENTION).
VOLUME_SPOOLED_DEV_CBT	The number of devices spooled to the volume.
VOLUME_WINDOW	Default window size for the volume.
VPROT	Volume protection mask.
WCK	TRUE if the device has write-checking enabled.
WRITETHRU_CACHE_ENABLED	TRUE if volume is mounted with write-through caching enabled.
WWID	Worldwide identifier for a Fibre Channel device.

Example

$ X = F$GETDVI("_DISKA0:","ERRCNT")
This would set X to the error count of DISKA0:.

$ X = F$GETDVI("_TERMA0:","DEVCLASS")
This would set X to "TERM".

F$GETJPI

F$GETJPI

F$GETJPI returns a specific piece of information about the specified process.

Format

F$GETJPI(pid,item)

Return Value

Either an integer or a string, depending upon the item requested. The table below indicates the valid items and the corresponding return values.

Arguments

pid

Specifies the process ID of the process for which you are requesting information. If this parameter is null, the current process is used. Use F$PID function to get a list of process IDs.

item

Specifies the type of device information to return. See the table below.

Description

F$GETJPI invokes the GETJPI system service to return information on the specified device. The following table lists all valid items.

Item	Information returned
ACCOUNT	The account name.
APTCNT	Active page table count.
ASTACT	Access modes with active ASTs.
ASTCNT	Remaining AST quota.
ASTEN	Access modes with ASTs enabled.
ASTLM	AST limit quota.
AUTHPRI	Maximum authorized priority.
AUTHPRIV	Authorized privileges.
BIOCNT	Remaining buffered I/O quota.
BIOLM	Buffered I/O limit quota.
BUFIO	Count of buffered I/O operations.
BYTCNT	Remaining buffered I/O byte count quota.
BYTLM	Buffered I/O byte count quota.
CASE_LOOKUP_IMAGE	Information about the file name lookup case sensitivity for the life of the currently running image: BLIND or SENSITIVE.
CASE_LOOKUP_PERM	Information about the file name lookup case sensitivity for the life of the process: BLIND or SENSITIVE.
CLASSIFICATION	Current MAC classification
CLINAME	Current command language interpreter (shell). Always returns "UCL".
CPULIM	CPU time limit.
CPUTIM	CPU time used, in nanoseconds.
CREPRC_FLAGS	Flags specified by the stsflg argument in the $CREPRC call that created the process.
CURPRIV	Current process privileges.
CURRENT_CAP_MASK	Current capabilities mask.
DFPFC	Default page fault cluster size.
DFWSCNT	Default working set size.
DIOCNT	Remaining direct I/O count.
DIOLM	Direct I/O limit.
DIRIO	Count of direct I/O operations.
EFCS	Local event flags 0-31.
EFCU	Local event flags 32-63.
EFWM	Event flag wait mask.
ENQCNT	Lock request quota remaining.
ENQLM	Lock request quota limit.
EXCVEC	Address of a list of exception vectors.
FAST_VP_SWITCH	Number of times process has issued vector processor request that enabled an inactive vector processor without the expense of a vector context switch.
FILCNT	Remaining open file quota.
FILLM	Open file quota.
FINALEXC	Address of a list of final exception vectors.
FREP0VA	First free page at end of executable/data address space.
FREP1VA	Last free page before start of stack space.
FREPTECNT	Number of pages available for virtual memory expansion.

GPGCNT	Global page count in working set.
GRP	Group(s) the process user belongs to.
HOME_RAD	Home resource affinity domain (RAD).
IMAGECOUNT	Number of image rundowns.
IMAGE_AUTHPRIV	Authorized privilege mask for running image.
IMAGE_PERMPRIV	Default privilege mask for running image.
IMAGE_WORKPRIV	Current (working) privilege mask for running image.
IMAGNAME	File name of current image.
IMAGPRIV	Privileges current image was installed with.
INSTALL_RIGHTS	Binary content of the install rights list, comma delimited.
INSTALL_RIGHTS_SIZE	Number of bytes needed to store the install rights.
JOBPRCCNT	Number of subprocesses owned by job.
JOBTYPE	Execution mode of the process at the root of the job tree.
LAST_LOGIN_I	Time of last interactive login.
LAST_LOGIN_N	Time of last noninteractive login.
LOGIN_FAILURES	Number of login failures prior to the start of the current session.
LOGIN_FLAGS	Bitmask containing additional information relating to the login sequence.
LOGINTIM	Process creation time.
MASTER_PID	PID of the top of the job's process tree.
MAXDETACH	Maximum number of detached processes allowed to the user who owns the process.
MAXJOBS	Maximum number of active processes allowed for user who owns the process.
MODE	Current process mode (BATCH, INTERACTIVE, NETWORK, or OTHER).
MSGMASK	Current message mask.
MULTITHREAD	Current multuthread limit.
NODENAME	Name of the cluster node on which the process is running.
NODE_CSID	Cluster ID of the cluster node on which the process is running.
NODE_VERSION	UOS version number of the cluster node on which the process is running.
OWNER	PID of process' owner.
PAGEFLTS	Page fault count.
PAGFILCNT	Remaining page file quota.
PAGFILLOC	Location of the page file.

PARSE_STYLE_PERM	Values set by the $SET_PROCESS_PROPERTIESW.
PARSE_STYLE_IMAGE	Values set by the $SET_PROCESS_PROPERTIESW.
PERMANENT_CAP_MASK	Permanent capabilities mask.
PERSONA_AUTHPRIV	Authorized privilege mask of the persona.
PERSONA_ID	The ID of the persona.
PERSONA_PERMPRIV	Default privilege mask of the persona.
PERSONA_RIGHTS	Binary content of the persona rights list, comma delimited.
PERSONA_RIGHTS_SIZE	Number of bytes needed to store the persona rights.
PERSONA_WORKPRIV	Current privilege mask of the active persona.
PGFLQUOTA	Page file quota.
PHDFLAGS	Flags word.
PID	Process ID.
PPGCNT	Process page count.
PRCCNT	Number of subprocesses owned by process.
PRCLM	Subprocess quota.
PRCNAM	Process name.
PRI	Current priority.
PRIB	Base priority.
PROC_INDEX	Process' index number.
PROCESS_RIGHTS	Contents of the process local rights list, delimited by commas.
PROCPRIV	Default privileges.
RIGHTSLIST	Contents of all process rights lists, delimited by commas.
RIGHTS_SIZE	Number of bytes required for the rights list.
SCHED_CLASS_NAME	Name of the scheduling class if process is class scheduled. Null otherwise.
SHRFILLM	Maximum number of open shared files for the job to which the process belongs.
SITESPEC	Per-process site-specific integer.
SLOW_VP_SWITCH	Number of times process has issued a vector instruction that enabled a inactive vector processor with a fill vector context switch.
STATE	Process state.
STS	First 32-bits of process status flags.

STS2	Second 32-bits of process status flags.
SUBSYSTEM_RIGHTS	Binary content of the subsystem rights lists, delimited by commas.
SUBSYSTEM_RIGHTS_SIZE	Number of bytes needed to store the system rights.
SWPFILLOC	Swap file location.
SYSTEM_RIGHTS	Contents of system rights list for the process, including identifier names, delimited by commas.
SYSTEM_RIGHTS_SIZE	Number of bytes needed to store the system rights.
TABLENAME	File specification of process CLI table.
TERMINAL	Login terminal name for interactive users.
TMBU	Termination mailbox unit number.
TOKEN	Token size (TRADITIONAL or EXPANDED).
TQCNT	Remaining timer queue entry quota.
TQLM	Timer queue quota.
TT_ACCPORNAM	Access port name for terminal associated with process.
TT_PHYDEVNAM	Physical device name of the terminal associated with process.
UAF_FLAGS	User authorization file (UAF) flags for user who owns process.
UIC	User ID code.
USERNAME	User name of process.
VIRTPEAK	Peak virtual address size.
VOLUMES	Count of currently mounted volumes.
VP_CONSUMER	Flag indicating if the process is a vector consumer.
VP_CPUTIM	Total amount of time process has accumulated as a vector consumer.
WSAUTH	Maximum authorized working set size.
WSAUTHEXT	Maximum authorized working set extent.
WSEXTENT	Current working set extent.
WSPEAK	Working set peak.
WSQUOTA	Working set size quota.
WSSIZE	Current working set limit.

Examples

```
$ N = F$GETJPI("","USERNAME")
```

This example would obtain the username for the current process.

F$GETSYI

F$GETSYI returns information about the local system or about a node within a cluster.

Format

F$GETSYI(item {,name {,id}})

Return Value

Either an integer or a string, depending upon the item requested. The table below indicates the valid items and the corresponding return values.

Arguments

item

Indicates the type of information to be reported about the local, or specified, node. The valid codes are listed in the table below.

name

Optional name of the node for which to obtain information. For the local system, this should be null.

id

The cluster ID of the cluster node to return information for. If 0 is specified, the local system is assumed.

Description

F$GETSYI invokes the GETSYI system service to return status and identification on the local system or another node in a cluster. The table below lists the valid item codes and describes what information is returned. If a specific node is provided, it can be specified by name or cluster ID. By using F$CSID to obtain each cluster node, information can be returned for all nodes in the cluster.

The following are the valid item codes:

Item	Information returned
ACTIVE_CPU_MASK	A number representing a bitmask indexed by CPU number. If a given bit is set, that CPU is in the active set.
ACTIVECPU_CNT	The count of active CPUs.
ARCHFLAG	Architecture flags for the system.
ARCH_NAME	Name of the CPU architecture.
ARCH_TYPE	Type of CPU architecture.
AVAIL_CPU_MASK	A number representing a bitmask indexed by CPU number. If a given bit is set, that CPU is in the active set and participating in scheduling activities.
AVAILCPU_CNT	The count of CPUs recognized by the system.
BOOT_DEVICE	Name of the device that UOS was booted from.
BOOTTIME	Date/time that UOS was booted.
CHARACTER_EMULATED	TRUE if this is a VAX with character instruction set emulation.
CLUSTER_EVOTES	Total number of votes in the cluster.
CLUSTER_FSYSID	System ID for the first node to boot in the cluster (the founding node). This is a character string containing a hexadecimal value.
CLUSTER	Time when the first node in the cluster was booted.

_FTIME	
CLUSTER_MEMBER	TRUE or FALSE if the node is the member of a cluster.
CLUSTER_NODES	Total number of nodes in the cluster.
CLUSTER_QUORUM	Total quorum for the cluster.
CLUSTER_VOTES	Total number of votes in the cluster.
COMMUNITY_ID	AlphaServer system hardware community ID.
CONSOLE_VERSION	Console firmware version.
CONTIG_GBLPAGES	Total number of free, contiguous global pages.
CPU	Processor type.
CPU_AUTOSTART	A list of CPUs that will brought into the active set if it transitions into the current instance from outside or is powered up while owned. A list of zeros and ones, delimited by commas, indexed by CPU. Any entry with a 1 indicates the CPU will be brought into the active set.
CPU_FAILOVER	Destinations for crashed Alpha CPUs. A list delimited by commas, indexed by CPU.
CPUCAP_MASK	List of hexadecimal values, delimited by commas, indexed by CPU. Each value is a bitmask indicating CPU capabilities.
CPUTYPE	The processor type.
CWLOGICALS	Flag indicating that the clusterwide logical name database has been initialized on the system.
DECIMAL_EMULATED	TRUE if this is a VAX CPU with decimal instruction set emulation.
DECNET_FULLNAME	Node name.
DECNET_VERSION	Network version.
D_FLOAT_EMULATED	TRUE if this is a VAX with D Float instruction emulation.
ERLBUFFERPAG_S2	Number of system pages used for each S2 errorlog buffer.
ERRORLOGBUF_S2	Number of S2 errorlog buffers.
ERLBUFFERPAGES	Number of system pages used for each S0 errorlog buffer.
ERRORLOGBUFFERS	Number of S0 errorlog buffers.

F_FLOAT_EMULATED	TRUE if this is a VAX with F Float instruction emulation.
FREE_GBLPAGES	Current count of free global pages.
FREE_GBLSECTS	Current count of free global section table entries.
FREE_PAGES	Total number of free pages.
G_FLOAT_EMULATED	TRUE if this is a VAX with G Float instruction emulation.
GALAXY_ID	128-bit Galaxy ID for AlphaServer GS systems.
GALAXY_MEMBER	1 if member of a Galaxy ID for AlphaServer GS systems, 0 if not.
GALAXY_PLATFORM	1 if running on a Galaxy platform for AlphaServer GS systems, 0 if not.
GALAXY_SHMEMSIZE	Number of shared memory pages for AlphaServer GS systems.
GH_RSRVPGCNT	Number of pages covered by granularity hints for AlphaServer GS systems.
GLX_FORMATION	Time when galaxy configuration was created.
GLX_MAX_MEMBERS	Maximum count of instances that may join the galaxy configuration for AlphaServer GS systems.
GLX_MBR_MEMBER	A 64-byte integer. Each 8 bytes represents a galaxy instance from 7 to 9. Value is 1 if instance is a member.
GLX_MBR_NAME	A string indicating the names which are known in the Galaxy membership.
GLX_TERMINATION	The time when the galaxy configuration was terminated for AlphaServer GS systems.
H_FLOAT_EMULATED	TRUE if this is a VAX with H Float instruction emulation.
HP_ACTIVE_CPU_CNT	The count of CPUs in the hard partition that are not in firmware console mode.
HP_ACTIVE_SP_CNT	The count of active UOS instances currently executing within the hard partition.
HP_CONFIG_SBB_CNT	The count of existing system building blocks within the current hard partition.
HP_CONFIG_SP_CNT	The maximum count of soft partitions within the current hard partition.
HW_MODEL	System model type.
HW_NAM	System model name.

E	
ITB_ENTRIES	On alpha, number of I-stream translation buffer entries that support granularity hints.
MAX_CPUS	The maximum number of CPUs that can be recognized by the system.
MEMSIZE	Number of pages of memory available to UOS.
MODIFIED_PAGES	Number of modified pages.
MULTITHREAD	Value of the MULTITHREAD system parameter.
NODE_AREA	Network area for the node.
NODE_CSID	Cluster ID of the node in the form of a string containing a hexadecimal value.
NODE_EVOTES	Number of votes alloted to the node.
NODE_HWVERS	Hardware version of the specified node.
NODE_NUMBER	Network number for the specified node.
NODE_QUORUM	Node's quorum.
NODE_SWINCARN	Software incarnation number for the node in the form of a string containing a hexadecimal value.
NODE_SWTYPE	Type of UOS software for the node.
NODE_SWVERS	Software version of the specified node.
NODE_SYSTEMID	System ID of the node as a hexadecimal string.
NODE_VOTES	Number of votes alloted to the node.
NODENAME	Node name (not including double colon).
NPAGED_FREE	Number of free bytes in the non-paged pool.
NPAGED_LARGEST	Size of largest contiguous area of free memory in the non-paged pool.
NPAGED_TOTAL	Total size (in bytes) of non-paged pool.
NPAGED_INUSE	Total number of bytes currently used in the non-paged pool.
PAGED_FREE	Number of free bytes in the paged pool.
PAGED_INUSE	Total number of bytes currently used in the paged pool.
PAGED_LARGEST	Size of largest contiguous area of free memory in the paged pool.
PAGED_TOTAL	Total size (in bytes) of non-paged pool.
PAGEFIL	Number of free pages in the currently installed paging files.

E_FREE	
PAGEFILE_PAGE	Total number of pages in the currently installed paging files.
PAGE_SIZE	Number of bytes in a physical page of memory.
PALCODE_VERSION	Version of PALCODE on an Alpha system.
PARTITION_ID	Soft partition ID for AlphaServer systems that support partitioning.
POTENTIAL_CPU_MASK	A number representing a bitmask indexed by CPU number. If a given bit is set, that CPU is in the potential set.
POTENTIALCPU_CNT	The count of the CPUs in the hard partition that are in the potential set.
POWERED_CPU_MASK	A number representing a bitmask indexed by CPU number. If a given bit is set, that CPU is powered up.
POWEREDCPU_CNT	The count of CPUs in the hard partition that are physically powered up.
PRESENT_CPU_MASK	A number representing a bitmask indexed by CPU number. If a given bit is set, that CPU is in the present set.
PRESENTCPU_CNT	The count of CPUs in the hard partition that physically reside in a hardware slot.
PRIMARY_CPUID	The ID of the primary processor for the node.
QUANTUM	Maximum amount of processor time a process can receive while other processes are waiting.
RAD_CPUS	List of RAD,CPU pairs, delimited by commas on AlphaServer GS systems.
RAD_MAX_RAD	The maximum number of RADs possible on this platform on AlphaServer GS systems.
RAD_MEMSIZE	List of RAD,PAGES pairs, delimited by commas on AlphaServer GS systems.
RAD_SHMEMSIZE	List of shared RAD,PAGES pairs, delimited by commas on AlphaServer GS systems.
REAL_CPUTYPE	Actual CPU type of the primary CPU on the node.
SCSNODE	Galaxy instance name on AlphaServer GS systems that support partitioning.
SCS_EXISTS	TRUE or FALSE to indicate whether the system communication subsystem (SCS) is currently loaded on the node.
SID	System ID.
SWAPFILE_FREE	Number of free pages in currently installed swap files.
SWAPFILE_PAGE	Number of pages in the currently installed swap files.
SYSTEM_RIGHTS	Contents of system rights list on the local node. Always null for a remote node.
SYSTEM_UUID	The 128-bit Universal Unique Identifier for the node.

SYSTYPE	The family or system hardware platform.
TOTAL_PAGES	Total number of physical memory pages.
USED_GBLPAGCNT	Number of pages currently in use in the global page table.
USED_GBLPAGMAX	Maximum number of pages ever in use in the global page table.
USED_PAGES	Total number of used pages.
VERSION	UOS version.
VECTOR_EMULATOR	Flag indicating presence of vector instruction emulator facility (VVIEF) on the node.
VP_MASK	Mask indicating which processors have vector coprocessors.
VP_NUMBER	Number of vector processors in the system.

The items that return rights lists return an array of int64 values.

Example
$ X = F$GETSYI('VERSION')

This sets the symbol X to the current version of UOS.

F$IDENTIFIER

F$IDENTIFIER

F$IDENTIFIER converts a UIC to a username or a username to a UIC.

Format
F$IDENTIFIER(value,type)

Return Value
A string containing the username for the passed UIC or an integer containing the UIC for the passed username. If the value is invalid or represents a user or UIC that doesn't exist, the function returns a null string ("") if you are converting from UIC to name, or 0 if you are converting from name to UIC.

Arguments
value
Specifies the offset of the starting character of the value that you want to extract. The offset of the first character is 0.

type
Specifies the type of conversion to perform. If this is "NAME_TO_NUMBER", the value parameter is treated as a username and the function returns the UIC for that user. If this is "NUMBER_TO_NAME", the value parameter is treated as a UIC and the function returns the username for that UIC.

F$INTEGER

F$INTEGER

F$INTEGER returns the integer equivalent of the result of the passed expression.

Format

F$INTEGER(expression)

Return Value

Returns the result of the passed expression as an integer.

Arguments

expression

Specifies the expression or value to be evaluated. Integer values are returned as an integer.
If the expression evaluates to a string, the result is converted to an integer. If the string does not contain a valid integer, the function returns 1 if the string begins with T, t, Y, or y. Otherwise it returns 0.

Example

```
$ A = F$INTEGER("12")

The symbol A would be set to 12 by this script.
```


F$LENGTH

F$LENGTH

F$LENGTH returns the length of the specified string. If the parameter is an integer, this returns the length of that integer as a string.

Format

F$LENGTH(value)

Return Value

Returns the length of the passed value.

Arguments

value

Specifies the value whose length to return.

Example

```
$ A = F$LENGTH("Hello")

The symbol A would be set to 5 by this script.
```


F$LOCATE

F$LOCATE

F$LOCATE locates the specified substring in another string and returns the offset of the substring within the other string. Positions are zero-based, thus the first character is at position 0. If the substring is not found, the length of the string is returned.

Format

F$LOCATE(substring, string)

Return Value
An integer value relative to position 0 from the beginning (leftmost) character of the string. If the substring is not found in the string, the string length is returned (the offset immediately after the last offset in the string).

Arguments
substring
Specifies the string to search for within the next parameter value.

string
Specifies the string to search for the substring.

Example
$ A = F$LOCATE("Wo","Hello World")

This script locates the position of "Wo" within "Hello World". The symbol A would be set to 6 in this case.

F$MATCH_WILD

F$MATCH_WILD

F$MATCH_WILD performs a wildcard match between one string and a pattern string. TRUE is returned if the strings match.

Format
F$MATCH_WILD(string, pattern)

Return Value
Returns the result of wildcard comparison as TRUE or FALSE.

Arguments
string
Specifies the string to match to the wildcard pattern.

pattern
Specifies the wildcard pattern to match with the string.

Example
$ A = F$MATCH_WILD("Hello World","Hello*")

The symbol A would be set to TRUE in this case.

F$MESSAGE

F$MESSAGE

F$MESSAGE returns a string containing the facility, severity, identification, and text associated with a given system status code.

Format
F$MESSAGE(code{,options})

Return Value
A character string containing the system message with the elements specified in the second argument

(or all elements if the second argument is omitted).

Even though each message in the system message file has a numeric code associated with it, not every possible numeric value corresponds to a system message. If a code is specified that has no corresponding message, the function returns a string containing the NOMSG error message.

Arguments

code

A integer code specifying the error to return text for.

options

Specifies one or more message fields to include in the returned string. If no options are specified or the argument is omitted, all fields are included. More than one field can be specified by delimiting them by commas. The following field names can be specified:

Keyword	**Field included**
FACILITY	Facility name
SEVERITY	Severity level indicator
IDENT	Message identification code
TEXT	Explanatory text of message

If the facility, severity, or ident is returned, it (or they) will be prefixed with the percent sign (%). If more than one are specified, they are delimited with dashes (-).
If only the text is included, it is not prefixed with any character. But if it is included with the facility, severity, and/or ident, the text is separated from the rest of the fields by a comma and a blank (,).

Example

```
$ A = F$MESSAGE(%X0F)
```

This call will set the symbol A to the error message corresponding to error message 15 (0F hexadecimal). The text will include all four fields of the message.

F$MODE

F$MODE

F$MODE returns a character string showing the mode in which the executing process is executing. It takes no parameters, but the parentheses are required.

Format

```
F$MODE()
```

Arguments

None.

Description

The function returns the string INTERACTIVE for interactive processes. Otherwise is returns the character string BATCH, NETWORK, or OTHER, as appropriate. Note that the string returned is always uppercase.

This function is useful for command procedures that must operate differently when executed interactively and noninteractively.

Example

```
$ IF F$MODE().EQS."INTERACTIVE" THEN GOTO INTERACTIVE
```


F$PARSE

F$PARSE

F$PARSE parses a file specification and returns either the expanded file specification or the requested field requested.

Format

F$PARSE(filespec, {default}, {related}, {field}, {type})

Return Value

A character string containing the expanded file specification or the specified field. If a complete file specification is not provided for the filespec argument, F$PARSE will supply defaults from the default argument, and if the corresponding item isn't in either the filespec or default, it is provided from the related argument.

If an error is detected during the parse, a null string is returned by F$PARSE. This can occur if the node, device, or directory do not exist. However, if a field name or a type argument of "SYNTAX_ONLY" is passed, F$PARSE returns the requested item without checking for the existence of the node, device, or directory.

Arguments

filespec

Specifies a character string containing the file specification to be parsed. This specification may include asterisk (*) and question mark (?) wildcards. Wildcards will be returned for the corresponding items.

default

Specifies a character string containing the default file specification. The fields in this specification will be substituted if the field is missing from the specification passed in filespec.

related

Specifies a character string containing the default file specification. The fields in this specification will be substituted if the field is missing from both of the specifications passed in filespec and default.

field

Specifies which field of the file specification is to be returned. If a null string is passed to this argument, the entire file specification is returned. The following are the valid field names that can be specified (do not abbreviate):

NODE	Node name
DEVICE	Device name
DIR	Directory/path

EC TO RY	
NA ME	File name
TY PE	File type
VE RSI ON	File version number

type
Specifies a character string containing the type of parsing to be performed. By default (if this argument is omitted), this function will verify the existence of the node, device, and path specified. However, this verification does not happen if a value is passed to the field argument. F$PARSE will also translate any logical names provided in the arguments. If not null, the argument must be one of the following keywords:

NO_ CON CEA L	Translates concealed logicals
SYN TAX_ ONL Y	Parse the file specification without verifying that the node/device/directory path exist.

Description
All but the first argument may be omitted, but commas must be provided in all cases. The SYS$PARSE system call is used to process the file specifications.

If the device and/or path are not provided in any of the arguments, F$PARSE substitutes the current default device and directory. Any other fields not provided in any of the arguments is returned as null.

F$PARSE can validate that a node, device, and path exist, but it will not validate the presence of a specific file.

F$PARSE will ignore any text that occurs after any character that is not valid for use in a file specification unless the specification is included in quotes. For the case of access information for a node, which is enclosed within quotes, a double-quote can be used within quoted node names to indicate the access information, which will be returned as part of the node field.

Examples
$ A = F$PARSE("INFO.DAT","_DISKA1:\uos",,,,"SYNTAX_ONLY")

The symbol A would be set to "_DISKA1:\uos\INFO.DAT".

$ A = F$PARSE("_DISKB0:\UOS\Users\Fred\INFO.DAT",,,"DIRECTORY",)

The symbol A would be set to "\UOS\Users\".

F$PID

F$PID

Each call to F$PID will return the next process ID that matches the context set up by F$CONTEXT. The

context is updated to point to the next process ID.

Format
F$PID(symbol)

Return Value
Returns the ID of the next process matching the criteria, or a null string if no more processes match.

Arguments
symbol
Specifies a symbol that UCL will use to refer to the context object that holds the UOS process context. Multiple contexts can exist simultaneously, each one specified with a different symbol.

The first time you use F$PID in a script, you should use a symbol that is either undefined or equated to null, or that has been created by the F$CONTEXT function. If the symbol was used with F$CONTEXT then F$PID function returns the first process that fits the criteria specified in the F$CONTEXT function and are accessible to your current privileges.

If a null string is passed to F$PID, it is considered to be a wildcard context and F$PID will iterate through all processes on the system that the caller has the privileges to access. There can only be one wildcard process context at a time for a given calling process.

After the last PID in the process list is returned, F$PID returns a null string.

Example:
$ PID = F$PID(CONTEXT)

F$PRIVILEGE

F$PRIVILEGE

F$PRIVILEGE returns a string value of either TRUE or FALSE, depending upon whether or not the current process has the specified privileges.

Format
F$PRIVILEGE(privileges)

Arguments
privileges
a string containing a single privilege name or comma-delimited list of multiple privilege names.

Description
The function will return TRUE if the passed privileges match the process' current privileges, and FALSE otherwise. Privilege names can be negated by prefixed with "NO" - for instance, "NOBYPASS". In order for the function to return TRUE, all specified privileges must be current for the process and no negated privileges can be.

Example
$ X = F$PRIVILEGE("OPER,NOGROUP")

If the process has the OPER privilege and does not have the GROUP privilege, X will be set to TRUE. If the process has the GROUP privilege or doesn't have the OPER privilege, X is set to FALSE.

F$PROCESS

F$PROCESS

F$PROCESS returns the current process name string. It has no arguments, but must be followed by parentheses.

Format
F$PROCESS()

Arguments
None.

Description
The function returns the current process name string.

Example
$ X = F$PROCESS()

F$SEARCH

F$SEARCH

F$SEARCH searches a directory for a passed file specification and returns the first/next fully qualified file name matching that specification.

Format
F$SEARCH(filespec{,context})

Return Value
A string containing the fully qualified file name matching the specification. If no file is found, a null string ("") is returned.

Arguments
filespec
The string containing the file specification to be search for. If node/device/path is omitted, the current node/device/path is used. The filename is not defaulted. If the file specification includes wildcards, each time F$SEARCH is called, the next matching file is returned. When no more matches are found, a null string ("") is returned.

context
Specifies a positive integer value (between 0 and 2147483647, inclusive) representing the search context. This allows multiple multiple wildcard searches to be done simultaneously. If omitted, a default context is used.

If a different file specification is provided for a previously-used context (or for the default context), a new lookup is started.

Description
F$SEARCH can be used to iterate through files in a directory, if a wildcard is provided. Note: any loop should exit when a null string is returned. Any of the wildcards valid for a UOS file specification can be used here: *, ?, and **. Only files which exist during the call are returned. Files that are created or deleted during the iteration may or may not be returned.

Example
$ NAME = F$SEARCH("*.*")

This will set the NAME symbol to the first filename in the current path.

F$SETPRV

F$SETPRV

F$SETPRV enables or disabled one or more user privileges. It returns a list of the current user privileges. A privilege can only be enabled if the user is authorized to have that privilege.

Format

F$SETPRV(privileges)

Return Value

A string containing the privileges for the current process before they were changed by the F$SETPRV function.

Arguments

privileges

A string containing a one or more privilege names. If multiple privileges are provided, they must be delimited by commas (,).

Description

The F$SETPRV lexical function invokes the SETPRV system service to enable or disable specific user privileges. It returns a list of user privileges that represent the state of the specified privileges that were held by the process before F$SETPRV was executed. That list can be passed to F$SETPRV to reset the privileges to their original state. Note that attempting to set privileges not authorized to the user will fail.

Example

```
$ OLDPRIVS = F$SETPRV("BYPASS,GRPPRV")
```

In this example, OLDPRIVS is set to the current state of the BYPASS and GRPPRV privileges, and the BYPASS and GRPPRV privileges are set.

F$STRING

F$STRING

F$STRING is provided for compatibility with VMS. UCL returns the parameter (evaluating any integer or string expressions).

Format

F$STRING(expression)

Return Value

Returns the evaluated expression.

Arguments

expression

A string or integer symbol, literal, value, or expression.

Description

The F$STRING lexical function evaluates the passed expression and returns the result.

Example

```
$ A = F$STRING(A - 2)
```

In this example, A is set to the evaluated value of A minus 2. This is equivalent to the following:

```
$ A = A - 2
```

F$TIME

F$TIME

F$TIME returns the current date and time in absolute time format. The function takes no parameters, but must be followed by parentheses.

Format

F$TIME()

Return Value

Returns a character string containing the current date and time in the following format:
dd-mmm-yyyy hh:mm:ss.cc

Arguments

none

Description

The day of month is space-padded if the day is less than 10. Thus, the time portion always begins at the 13th character. Note that if assigning to a symbol using string assignment (:=), the leading space (if any) is trimmed.

Example

$ A = F$TIME()

F$TRNLNM

F$TRNLNM

F$TRNLNM translates a symbol name and returns the equivalence name string or the requested attributes of the specified symbol name.

Format

F$TRNLNM(name {, table {, index {, mode {, case {, item}}}}})

Return Value

A character string containing the equivalence name or requested attribute of the symbol name. If no match is found, null is returned.

Arguments

name

a string containing the name of the symbol/logical to translate.

table

A string containing the name of the symbol table that F$TRNLNM should search to translate the symbol name. If no table is specified, F$TRNLNM searches the process, job, group, and system symbol name tables, in that order.

index

The index of the equivalence value to return. The first equivalence value is index 0, which is the default used if this option is omitted or null.

mode

A string containing one of the following access modes:

USER	User mode (ring 3). This is the default.
SUPERVISOR	Supervisor mode (ring 2).
DEVICE	Device mode (ring 1).
EXECUTIVE	An alternate value for DEVICE, provided for compatibility with VMS.
KERNEL	Kernel mode (ring 0).

When a table is being searched for a symbol, a symbol with the specified mode is looked for first. If not found, the next lower level access mode is looked for, and so forth down to Kernel mode.

case

A string containing any of the following items, delimited by commas. If omitted or a null string is specified, the defaults are used.

CASE_BLIND	Default. The first symbol matching the name is used, regardless of the case of the symbol name.
CASE_SENSITIVE	The symbol must match the name exactly, including case, to be returned.
INTERLOCKED	Wait to complete lookup until any current clusterwide symbol changes are finished.
NONINTERLOCKED	Default. Symbol lookup occurs without waiting for any clusterwide symbol changes to finish.

item

A string containing one of the following items. If omitted or a null string is specified, the item defaults to "VALUE".

ACCESS_MODE	The access mode associated with the symbol. One of the following: USER, SUPERVISOR, DEVICE, KERNEL.
CLUSTERWIDE	TRUE or FALSE to indicate if the symbol is in a clusterwide symbol table.
CONCEALED	TRUE or FALSE to indicate if the CONCEALED attribute is set for the symbol.
CONFINE	TRUE or FALSE to indicate if the symbol is confined (not copied to subprocesses).
CRELOG	TRUE or FALSE to indicate if the symbol was defined with the CRELOG system service.
LENGTH	Length of the value (equivalence name) of the specified symbol.
MAX_INDEX	Largest index for the symbol's equivalence values.

NAME	The case-sensitive name of the found symbol.
NO_ALIAS	TRUE or FALSE to indicate if the NO_ALIAS flag is set for the symbol.
TABLE	TRUE or FALSE to indicate whether the symbol name is the name of a symbol table.
TABLE_NAME	The name of the table in which the symbol was found.
TERMINAL	TRUE or FALSE to indicate if the TERMINAL flag is set for the symbol.
VALUE	The symbol's value. If the symbol has multiple equivalence values, the equivalence value associated with the specified index is returned.

Description

F$TRNLNM uses the TRNLNM system service to translate a symbol and return the equivalence name value, or other requested data. The translation is not iterative.

You can omit optional arguments to the right of the last specified argument. Any arguments to the left can be omitted, but the comma delimiter must be included.

Examples

$ A = F$TRNLNM("a","LNM$PROCESS",,,,"NAME")

This would return the case-sensitive name of the symbol whose normalized (upper case) name is A. If A is not defined, a null string is returned.

F$TYPE

F$TYPE

The F$TYPE lexical function returns the data type of a symbol.

Format

F$TYPE(symbol)

Return Value

F$TYPE returns the data type of a symbol. If the symbol is a process context from the F$PID function, "PROCESS_CONTEXT" is returned. If the symbol is a cluster context from F$CSID, "CLUSTER_SYSTEM_CONTEXT" is returned. Otherwise, "INTEGER" is returned if the contents of the symbol represents a valid integer or "STRING" if not. If the symbol is undefined, a null string is returned.

Arguments

symbol

The symbol name whose type is returned.

Description

The F$TYPE lexical function returns the data type of a symbol. An expression can be passed as well, and the result of that expression is used to determine the type returned.

Example

$ A = F$TYPE(B)

If the symbol B contains "3" or some other number, A would be set to "INTEGER".

F$UNIQUE

F$UNIQUE

F$UNIQUE generates a string that is suitable to be a file name and is guaranteed to be unique across the cluster. Unique file names can be used when creating temporary files. Note that this function guarantees a unique name for each time it is called, no matter which process or node is making the call. However, it does not guarantee that the name is not already in use on the system.

Format
F$UNIQUE()

Return Value
A unique character string.

Arguments
none

Example
$ A = F$UNIQUE()

This call will set the symbol A to a unique string which can be used as a filename.

F$USER

F$USER

F$USER returns the current user name as a string. The function takes no arguments, but must be followed by parentheses.

Format
F$USER()

Return Value
A character string containing the current user's username.

Arguments
none

Example
$ A = F$USER()

This call will set the symbol A to the user name of the user running the script.

F$VERIFY

F$VERIFY

F$VERIFY returns an integer value indicating whether the procedure verification flag is currently on or off. If arguments are provided, the procedure and/or image verification can be changed. The parentheses must follow the function name even if no arguments are provided.

Format
F$VERIFY({proc} {,image})

Return Value
The function returns 0 if procedure verification is off, or 1 if it is on.

Arguments
proc
Specifies 1 to turn procedure verification on, or 0 to turn it off. If omitted, the current procedure verification setting is left unchanged. Note that the function returns the procedure verification state prior to making any change to it.

image
Specifies 1 to turn image verification on, or 0 to turn it off. If omitted, the current image verification setting is left unchanged.

Examples
$ A = F$VERIFY()
This code will set the symbol A to the current procedure verification flag. No other processing occurs.

$ A = F$VERIFY(,1)
This code will set the symbol A to the current procedure verification flag. The procedure verification is left unaltered, and the image verification is enabled.

$ A = F$VERIFY(0)
This code will set the symbol A to the current procedure verification flag and then turns procedure verification off.

$ A = F$VERIFY(1,0)
This call will set the symbol A to the current procedure verification flag and then turns procedure verification on and image verification off.

Commands

Commands

UCL commands are used to perform actions on behalf of UCL scripts. Some commands are used to control flow of the script, such as loops. Some are used to conditionally execute other commands. Still others are used to perform such actions as input and output.

Some commands support qualifiers which are delimited by slashes. The general format of commands is:

command {qualifiers} {parameter {parameter...} }

@

@

This executes a UCL script.

Format
@ filespec {/OUTPUT=outfilespec} {parameter{ parameter...}}

Parameters
filespec

File specification of the script to execute. Wildcards (* and ?) are not permitted in the file name. The default file type is ".com".

parameter
Specifies one or more optional parameters. The symbols (P1, P2, etc.) are assigned these values during the execution of the specified script. Each parameter is delimited by one or more spaces. A null parameter can be provided by two consecutive quotes (""). Each parameter can contain any characters desired, consistent with the following rules.

- If the first parameter starts with a slash, the entire parameter must be surrounded by quotes.
- If the parameter includes embedded spaces, the parameter must be enclosed in quotes.
- All alphabetic characters are converted to uppercase. If you wish to preserve lowercase characters, the parameter must be enclosed in quotes.
- You can enclose quotes within the parameter by using double quotes for each quote.

For instance, the following parameters:
```
Hello there
```
will be converted to two parameters as if the following were executed:
```
P1="HELLO"
P2="THERE"
```

The following:
```
"Hello there"
```
will be converted to a single parameter as if:
```
P1="Hello there"
```

The following:
```
"Hello ""there"""
```
the value assigned to P1 in this case would be:
```
Hello "there"
```

Description
The @ command is used to execute a UCL script. It can be used within a UCL script to execute another script. Each use of @ creates a new scope with local symbols. The maximum number of levels allowed depends upon system quotas.

Qualifiers
/OUTPUT=outfilespec
Specifies the name of the file to which the script output is written. By default, the output goes to the current SYS$OUTPUT specification. The /OUTPUT qualifier must immediately follow the file specification or else it will be interpreted as a parameter.

Example
```
$ @MAKE MYAPP
```


CALL

CALL

Transfers control to a labeled subroutine in the current script file without creating a new scope.

Format
CALL label {/OUTPUT=filespec} {parameter {...}}

Parameters
label
Specifies a label that must appear within the command file. The label may follow or precede the CALL

command. If there are duplicate labels, the last one encountered is used. If it hasn't been encountered yet, the entire script is searched from the beginning and the first instance found is used. If still not found, an error occurs.

parameter

Specifies one or more optional parameters. The symbols (P1, P2, etc.) are assigned these values during the execution of the specified script. Each parameter is delimited by one or more spaces. A null parameter can be provided by two consecutive quotes (""). Each parameter can contain any characters desired, consistent with the following rules.

- If the first parameter starts with a slash, the entire parameter must be surrounded by quotes.
- If the parameter includes embedded spaces, the parameter must be enclosed in quotes.
- All alphabetic characters are converted to uppercase. If you wish to preserve lowercase characters, the parameter must be enclosed in quotes.
- You can enclose quotes within the parameter by using double quotes for each quote.

For instance, the following parameters:

```
Hello there
```

will be converted to two parameters as if the following were executed:

```
P1="HELLO"
P2="THERE"
```

The following:

```
"Hello there"
```

will be converted to a single parameter as if:

```
P1="Hello there"
```

The following:

```
"Hello ""there"""
```

the value assigned to P1 in this case would be:

```
Hello "there"
```

Description

If the command procedure is not coming from a random-access device, CALL will generate an error. Otherwise, control of the script is transferred to the first command immediately following the SUBROUTINE at the specified label. The CALL command operates similarly to the @ command, except that the overhead of opening a new command file and creating a new symbol table is avoided. Execution continues until ENDSUBROUTINE, RETURN, or EXIT is encountered, at which point control returns to the line following the CALL command.

CALL creates a new procedure level, but not a new scope. Parameters are assigned to symbols P1, P2, etc., which are local to the calling level. Local symbols defined in the current procedure level are available to the subroutine called with CALL.

A subroutine has only one entry point - code cannot jump into the middle of a subroutine. And each SUBROUTINE command must have a matching ENDSUBROUTINE command. If SUBROUTINE is encountered during normal execution of the command file (i.e. not as the target of a CALL), all lines up to, and including, the matching ENDSUBROUTINE are skipped.

Qualifier

/OUTPUT=filespec

Using this switch will direct all output to the specified file or device. If not specified, the current output setting is used. Wildcards are not allowed in the filespec. When the subroutine exits, the output is restored to what it was immediately prior to the CALL.

Example

```
$ A = 1
```

```
$Test1:
$ CALL Test2
$ IF A.LE.10 THEN GOTO Test1
$ EXIT
$Test2:
$ SUBROUTINE
$ WRITE SYS$OUTPUT "This is Test2"
$ CALL Test3
$ A = A + 1
$Test3:
$  SUBROUTINE
$  WRITE SYS$OUTPUT "This is Test3"
$  ENDSUBROUTINE
$ ENDSUBROUTINE
```

This sample script shows how to use CALL. A is set to 1 and the lines between Test1 and EXIT will loop 10 times. Each time through the loop, control is transferred to the line after Test2, which writes "This is Test2" and then uses CALL to transfer control to the line after Test3, which write "This is Test3". When ENDSUBROUTINE is executed, control is returned to the line after the CALL Test3, which adds 1 to A and then the nested SUBROUTINE is skipped until the ENDSUBROUTINE matching the SUBROUTINE at Test2, which returns to the line after CALL Test2. The output from this will be:

```
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
```


CLOSE

Closes a file previously opened with OPEN.

Format

CLOSE symbol{:}

Parameters

symbol

The name of the symbol used to open the file.

Description

This command closes an open file. Files opened with the OPEN command remain open until closed with CLOSE. If a command procedure terminates without closing open files, the files remains open.

Qualifiers

/DISPOSITION=option

Specifies actions to take after the file is closed. The options are:

Option	Meaning
DELETE	Delete the file.
KEEP	(default) Keep the file.
PRINT	Print the file.
SUBMIT	Submit the command file to the batch queue.

/ERROR=label

Specify a label to jump to if there was an error when closing the file.

/{NO}LOG

/LOG is the default and indicates that if an error occurs, it is reported to the user, even if /ERROR is specified. /NOLOG means that no error message is displayed on error.

Example

```
$ OPEN/READ/ERROR=NoFile IN Filename
$ READ IN Line
$ CLOSE IN
```


ENDIF

ENDIF

ENDIF ends a multi-line IF...THEN command. See the description of the IF command for more information.

Format

ENDIF

ENDSUBROUTINE

ENDSUBROUTINE

Indicates the end of a local subroutine in a command file. ENDSUBROUTINE must be the last executable command of a subroutine. See the CALL command for more details.

Format

ENDSUBROUTINE

Parameters

None.

Example

```
$Test3:
$ SUBROUTINE
$ WRITE SYS$OUTPUT "This is Test3"
$ ENDSUBROUTINE
```

EOJ

EOJ

Marks the end of a batch job.

Format
EOJ

Parameters
None

Description
The EOJ (End Of Job) command marks the end of a batch job submitted via card reader. It is not required. If used in any other context, EOJ logs the process out.

Example
$ EOJ

EXIT

EXIT

Terminates processing of a UCL script and returns control to the calling command level.

Format
EXIT {code}

Parameters
code
An optional numeric code that the reserved global symbol $STATUS is set to. The lower 3 bits of the value are assigned to the $SEVERITY symbol. If no code is specified, the current value of $STATUS is returned.

Description
The EXIT command is used to terminate the execution of a procedure.

Example
```
$ EXIT
```


GOSUB

GOSUB

Transfers control to a labeled subroutine in the current script file without creating a new procedure nesting level.

Format
GOSUB label

Parameter
label

Specifies a label that must appear within the command file. The label may follow or precede the GOSUB command. If there are duplicate labels, the last one encountered is used. If it hasn't been encountered yet, the entire script is searched from the beginning and the first instance found is used. If still not found, an error occurs.

Description

If the command procedure is not coming from a random-access device, GOSUB does nothing. Otherwise, control of the script is transferred to the command immediately following the specified label. The RETURN command will end the subroutine and return control to the command immediately following the GOSUB.
GOSUB does not create a new script scope and so all labels and local symbols defined in the current scope are available to the subroutine called with GOSUB.

Example

```
$ A = 1
$Test1:
$ GOSUB Test2
$ IF A.LE.10 THEN GOTO Test1
$ EXIT
$Test2:
$ WRITE SYS$OUTPUT "This is Test2"
$ GOSUB Test3
$ A = A + 1
$ RETURN
$Test3:
$ WRITE SYS$OUTPUT "This is Test3"
$ RETURN
```

This sample script shows how to use GOSUB and RETURN. A is set to 1 and the lines between Test1 and EXIT will loop 10 times. Each time through the loop, control is transferred to the line after Test2, which writes "This is Test2" and then uses GOSUB to transfer control to the line after Test3, which write "This is Test3". When RETURN is executed, control is returned to the line after the GOSUB Test3, which adds 1 to A and then executes RETURN, which returns to the line after GOSUB Test2. The output from this will be:

```
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
This is Test2
This is Test3
```

GOTO

GOTO

Transfers control to a label in a script.

Format

GOTO label

Parameter

label

Specifies a label in the script. When the GOTO command is executed, control passes to the command following the specified label.

Description

GOTO is used to transfer control to a line that is not the next line in the script file. If the script file is not on a random access device (such as a disk file), the GOTO command performs no operation.

If there are duplicate labels of the same name, control is always transferred to the label most recently encountered during script execution. If not previously encountered, the entire script is searched for the label, in which case the first matching label is used. If no matching label is found, an error occurs.

Example

```
$ A=0
$LOOP:
.
.
.
$ A=A+1
$ IF A.LT.10 THEN GOTO LOOP
```


IF

IF

IF tests the value of an expression and, depending upon the form uses, executes one of the following:
If THEN follows IF on the same line: if the expression is true, the command after THEN is executed; if the expression is false, execution continues with the next line.

If THEN follows IF on the next line: if the expression is true, all of the commands after that point are executed until ELSE or ENDIF is encountered; if the expression is false, execution begins after the next ELSE (or ENDIF if ELSE is not specified).

Note that the expression evaluates to false if it has an even numeric value or is a string that starts with any letter other than Y, y, T, or t. The expression evaluates to true if it has an odd numeric value or is a string that starts with Y, y, T, or t.

Format

Form 1:

$ IF expression THEN {$} command

Form 2:

```
$ IF expression
$ THEN {{$} command}
command
.
.
.
$ ENDIF
```

Form 3:

```
$ IF expression
$ THEN {{$} command}
command
.
.
.
$ ELSE {command}
command
.
.
.
$ ENDIF
```

Parameters

expression
: Defines the comparison to be made. The result of the comparison is either 0 (false) or non-zero (true). Comparisons can be numeric or string literals, symbol names, or lexical functions separated by operators. Any string not enclosed within quotes (") are assumed to be symbol names. If an unknown symbol name is specified, an error will result and the next command is executed.

command
: Specifies command(s) to be executed under the appropriate circumstances based on the true or false value of the expression.

Examples

1. Form 1

```
$ IF P1 .EQS. "" THEN EXIT
$ IF P1 .EQS. "X" THEN WRITE SYS$OUTPUT "Invalid parameter value"
```

2. Form 2

```
$ IF F$ENVIRONMENT("INTERACTIVE")
$ THEN
$ WRITE SYS$OUTPUT "UOS version: ", F$GETSYI("VERSION")
$ ENDIF
```

3. Nested IFs

```
$ IF AVAILABLE .EQ. 0
$ THEN
$   IF AMOUNT .GE. 0
$   THEN WRITE SYS$OUTPUT "It doesn't fit"
$   ELSE WRITE SYS$OUTPUT "Copying next item..."
$   ENDIF
$ ENDIF
```

Note: the extra indentation is not required, but using it makes the script more easily understood.

INQUIRE

Reads a value from SYS$COMMAND (usually the terminal or the next line in the current command procedure) and assigns that value to a symbol.

Format

INQUIRE {qualifiers} symbolname {prompt}

Parameters

symbolname

Specifies the name of the symbol to which the value is assigned. If the symbol doesn't exist, it is created.

prompt

Specifies optional prompt text to be displayed to the user just prior to accepting input. The prompt is converted to upper case unless enclosed in quotes ("). Quotes must be used if the prompt contains punctuation or whitespace (tabs and/or spaces). To include quotes in the prompt, use doubled quotes. If a prompt is not specified, UCL uses the symbol name as the prompt. The prompt is displayed with a colon (:) and space at the end of the prompt string. This behavior can be modified with the /PUNCTUATION qualifier.

Description

INQUIRE displays the prompt and reads the response from the process' command stream. That is: the sys$command device originally assigned to the process. That means that if the INQUIRE command is encountered inside a command procedure file (however deeply nested), the prompt will be sent to the terminal that started the command file, and the input value will be read from the same.

If the command file is being run as a batch job, the input value is read from the next line of the topmost nesting level. If that line begins with a dollar sign ($), it is considered to be a command instead of input. In this case, the symbol is assigned a null value.

When the value is assigned to the symbol, the value is first converted to upper case, leading and trailing whitespace is trimmed, and multiple spaces/tabs between characters are reduced to a single space. This conversion is not done within quotes (") in the input.

Single quotes (') not occurring within double quotes (") will trigger the symbol substitution feature of UCL.

Qualifiers

/GLOBAL

This indicates that the symbol is a global symbol rather than a local symbol.

/LOCAL (default)

This indicates that the symbol is a local symbol rather than a global symbol. This is the default behavior.

/{NO}PUNCTUATION

The default is /PUNCTUATION, which displays a colon and space after the prompt. Using /NOPUNCTUATION suppresses the colon and space.

Example

```
$ INQUIRE/NOPUNCTUATION CONFIRM "Are you sure you want to proceed? "
$ IF .NOT. CONFIRM THEN RETURN
```

This code would result in the following prompt being shown on the terminal (no colon is shown due to

the /NOPUNCTUATION):

```
Are you sure you want to proceed?
```

If the user enters an odd numeric value or a string that begins with T, t, Y, or y, the command file will continue. Otherwise it will return to the caller.

ON

ON

Performs an action when a command or program executed by a command procedure encounters an error condition or is interrupted.

Format

ON condition THEN {$} command

Parameters

condition

Specifies a condition. It must be one of the following keywords, which may be abbreviated to as few as one character:

WARNING	Execute command if a warning occurs.
ERROR	Execute command if an error occurs.
SEVERE_ERROR	Execute command if a severe error occurs.
CONTROL_Y	Control-Y was typed on SYS$INPUT.

The default condition is: ON ERROR THEN EXIT.

command

Specifies the UCL command to execute. The dollar sign ($) is optional. When an error level is specified, the command executes if an error occurs which is greater than or equal to the specified level of errors.

Description

After each command is executed, UCL checks the current error status and if an error occurs that is equal to or greater than the specified error level, the command specified with ON is executed. Unless the command is GOTO or EXIT, control returns to the next command in the command file after the one that resulted in an error.

For instance if ON WARNING is used, then the specified command will be executed when a warning, error, or severe error occurs. If ON SEVERE_ERROR is used, then the specified command is executed only if a severe error occurs. In that case, the script will continue if a warning or error occurs.

The global symbol $SEVERITY indicates the error level.

An active ON condition only applies to the command procedure in which it executes. Therefore, if ON is used in a script which then calls another script with @, it will not apply to the called file. Likewise, if ON is used in the called script file, that ON will not apply once the script file has exited to the calling script. Separate from catching errors is the ability to catch control-Y.

It is highly recommended that ON CONTROL_Y not be used in command scripts since this disables the normal processing of control-Y. Any infinite loops cannot be interrupted in this case.

Example

```
$ ON ERROR THEN GOTO Next
$ WRITE FILE "This is Test2"
$ EXIT
$Next:
```

```
    $ ON ERROR THEN EXIT
.
.
.
```

The code first requests that any error will result in the code after the "Next" label be executed. If no error occurs on the WRITE, the script exits after the operation. If an error occurs, UCL begins execution at the "Next" label, which requests that any error will result in exiting the script, and then proceeds to execute any following commands.

OPEN

OPEN

Opens or creates a file for reading and/or writing.

Format

OPEN symbol{:} filespec

Parameters

symbol

The symbol name to assign to the open file. This may be followed by an optional colon.

filespec

Specifies the name of the file being opened or created. The file type defaults to ".dat". Wildcards are not allowed.

Description

Files can be opened for reading and/or writing, and they can be created if they don't exist. Once open, the file can be used in the READ and WRITE commands. The CLOSE command is used to close an open file. The files will remain open until explicitly closed or the process logs out. Note that if a command file exits with an error, any files it opened and did not explicitly close will remain open.

The logical devices SYS$INPUT, SYS$OUTPUT, SYS$COMMAND, and SYS$ERROR do not need to be opened explicitly. All other files must be opened with the OPEN command. All symbols associated with OPENed files are local to the current process. Deleting a symbol associated with a file, without first closing the file, will result in the file remaining open but not being accessible by the process.

Attempting to open a new file with an existing symbol association will fail without warning and subsequent attempts to read or write the file will apply to the file that was originally assigned. Thus, if you wish to reuse a symbol for a new file, you should first CLOSE the file before opening a new one.

Qualifiers

/APPEND

Opens an existing file and positions the file pointer at the end of the file. New writes will be added to the end of the file. The /APPEND and /WRITE qualifiers are mutually exclusive.

/ERROR=label

Transfers control to the location specified by "label". This operation overrides any current ON condition. If this qualifier is not specified and an error occurs, the current ON condition action is taken.

/READ (default)

Opens file for reading. If you specify /READ without /WRITE, the file must already exist.

/{NO}SHARE{=option}

Opens the file as sharable.

/WRITE

Opens the file for writing. This creates a new file. /WRITE and /APPEND are mutually exclusive. If /

READ is used with /WRITE, the file must already exist.

Example

```
$ OPEN/READ/ERROR=NoFile IN Filename
$ READ IN Line
```


READ

READ

Reads a single record from the specified file and places the data in the specified symbol. If the input file is sequential or untyped, a single line is read.

Format

READ file{:} symbol

Parameters

file

This is the symbol name used when the file was opened. The logical names SYS$INPUT, SYS$OUTPUT, SYS$ERROR, and SYS$COMMAND can also be used. The colon suffix is optional.

symbol

This is the symbol to write the data to.

Description

To read a file, it must first be opened with the OPEN command (except for the process-permanent logicals SYS$INPUT, SYS$OUTPUT, SYS$COMMAND, and SYS$ERROR).

Unlike the INQUIRE command, the READ command doesn't trim extra whitespace, remove quotation marks, or change case. Nor does READ do symbol substitution.

After reading from a file, the file position is moved to the next location such that the next READ will return the next line or record in the file.

Qualifiers

/DELETE

Deletes the record from an indexed file after it has been read.

/END_OF_FILE=label

If the file position is at the end of the file, and this qualifier is specified, control is transferred to the specified label.

/ERROR=label

If the file read causes an error, control is transferred to the specified label.

/INDEX=n

Specifies the index (n) to look up when reading an indexed file. This has no effect on non-indexed files. If /INDEX is not specified for an indexed file, the last index read is used (if this is the first read after opening an indexed file, index 0 is used).

/KEY=string

Reads the first record with the key that matches the specified string. Binary and integer keys are not allowed. This applies only to indexed files.

/MATCH=comparison

Specifies the algorithm for matching keys. This must be one of the following values:

Comparison Value	Meaning
EQ	Selects keys that equal the match value (default)
GE	Selects keys greater than or equal to the match value
GT	Selects keys greater than the match value
LE	Selects keys less than or equal to the match value
LT	Selects keys less than the match value

/NOLOCK
Indicates that the read record is not to be locked.

/PROMPT=string
Indicates a prompt to display when reading from a terminal. The default prompt is "DATA:".

/TIME_OUT=n
Indicates the number of seconds until the READ operation is terminated if no data is received.

/NOTIME_OUT (default)
Indicates that there is no timeout when waiting for data.

/WAIT
Indicates to wait for a record to become available.

Example

```
$ OPEN/READ IN Filename
$Loop:
$ READ/END_OF_FILE=Done IN Text
$ WRITE SYS$OUTPUT Text
$ GOTO Loop
$Done:
$ CLOSE IN
```


RECALL

RECALL

Displays previously entered commands for re-execution.

Format
RECALL {qualifiers} {specifier}

Parameters
specifier
Indicates the first character(s) of the command to recall. If more than one previous command begins with the characters specified here, the most recent matching command is shown.

If the specified value is a number, the number is used as an index into the recall buffer, and that command is the one shown.

Note that the command cannot be specified along with certain qualifiers.

Description
When you enter commands, they are stored in a recall buffer for later reuse with the RECALL command or control-B. The RECALL command, itself, is not stored in the recall buffer. By default, up to 254 commands are stored in the buffer. If this limit is exceeded, the oldest entered command is removed to

make use for the new command.

When RECALL is used, the command is shown but not executed. To execute the command, you must press ENTER. The command line editing feature can be used to modify the recalled command before you press ENTER to execute it.

Qualifiers

/ALL

Displays all of the commands (and their numbers) in the recall buffer. If a specifier is provided, all commands whose prefix matches that specifier are shown.

/ERASE

Erases the contents of the recall buffer.

/INPUT=filespec

Reads the contents of the specified file into the recall buffer. Each line of the file is saved as a command in the buffer. If not specified, the file extension defaults to ".LIS".

/OUTPUT=filespec

Writes the contents of the recall buffer to the specified file. If not specified, the file extension defaults to ".LIS". This can be combined with a specifier or /SEARCH to direct a subset of commands to the file.

/{NO}PAGE

/NOPAGE is the default. When used with /ALL, /PAGE will pause after each screenful of commands. /PAGE is ignored in cases where multiple lines are not output to the terminal. The list can be terminated at this point by control-Z, control-C, or control-Y.

/SEARCH

Searches the recall buffer for any/all command(s) containing the specifier and displays them.

Examples

```
$ RECALL S
```

This command recalls the most recent command starting with "S".

```
$ RECALL/SEARCH/OUTPUT=commands 3
```

This command writes each command containing the character "3" to "commands.lis".

RETURN

RETURN

Terminates a subroutine and returns control to the command following the calling GOSUB command.

Format

RETURN {code}

Parameter

code

An optional code that can be used to set the global $STATUS and $SEVERITY symbols. If not provided, $STATUS and $SEVERITY are unchanged by the RETURN command.

Description

RETURN terminates a subroutine called via the GOSUB command and returns control to the line immediately following the GOSUB that initiated the subroutine. If RETURN is encountered without a previous GOSUB command execution, an error occurs. RETURN is ignored if encountered in a

command stream that is not on a random access device.

Example

```
$ GOSUB Test
$ EXIT
$Test:
$ WRITE SYS$OUTPUT "This is Test"
$ RETURN
```


RUN

RUN

Executes a program in the context of the current process.

Format

RUN {qualifiers} filespec {parameters...}

Parameters

filespec

Indicates the name of the program file to execute. Programs can be run without prefixing them with RUN if the program name isn't a UCL command, unless the program name is prefixed with a path. For instance the following program must be executed by RUN: if.exe.
The following rules determine which program is executed if the path and/or extension are omitted:
If a path isn't specified, the program must exist in the user's execution path.
The first instance of the program found in the execution path is the one that is executed, so if programs with the specified name exist in multiple directories in the execution path, you must specify the directory to execute one that is later in the path.
If a file extension is not provided and multiple matching programs are found in a given directory, the one executed depends upon the current order of execution for file extensions.

parameters

Indicates zero, one, or more parameters to pass to the program when execution begins.

Description

This command executes the specified program in the current process. The current process privileges are used when running the program, unless the program has privileges assigned to it. You must have Read or Execute access to the program to execute it.
If any qualifier other than /{NO}DEBUG, /USER, or /DETACH is specified, the process is run as a subprocess. /DEBUG is incompatible with programs run in subprocesses.

Qualifiers

/ACCOUNTING (default)
/NOACCOUNTING

/NOACCOUNTING requires the ACNT privilege. /NOACCOUNTING turns off resource tracking for the program.

/AST_LIMIT=quota

Specifies the AST quota for the subprocess. This cannot exceed the current process quota unless the user has the EXQUOTA privilege.

/AUTHORIZE
/NOAUTHORIZE (default)

The IMPERSONATE privilege is required use /NOAUTHORIZE. Programs run with /NOAUTHORIZE use the system logged-out quotas rather than the current user.

/BUFFER_LIMIT=quota

Specifies the BIOLM quota for the subprocess. If the quota exceeds that of the user, the EXQUOTA privilege is required.

/DEBUG
/NODEBUG (default)
Runs the program in the debugger, if debugging is enabled for the program.

/DELAY=delta-time
The subprocess will be hibernated for the specified time before it begins execution.

/DETACHED
Specifies that the program will be run in a subprocess.

/DUMP
/NODUMP (default)
If /DUMP is specified, a dump file will be created if the program ends abnormally.

/ENQUEUE_LIMIT=quota
Specifies the maximum number of locks the subprocess can have. If this exceeds the ENQLM quota of the current process, the EXQUOTA privilege is required.

/ERROR=filespec
Specifies the subprocess' SYS$ERROR value.

/EXTENT=quota
Specifies the maximum WSEXTENT value of the subprocess. If the quota exceeds that of the process owner, the EXQUOTA privilege is required.

/FILE_LIMIT=quota
Specifies the FILLM value of the subprocess. If the quota exceeds that of the process owner, the EXQUOTA privilege is required.

/INPUT=filespec
Specifies the subprocess' SYS$INPUT value.

/INTERVAL=delta-time
Indicates the interval between the subprocess being regularly hibernated and awakened.

/IO_BUFFERED=quota
Indicates the BIOLM quota of the subprocess. If the quota exceeds that of the process owner, the EXQUOTA privilege is required.

/IO_DIRECT=quota
Indicates the DIOLM quota of the subprocess. If the quota exceeds that of the process owner, the EXQUOTA privilege is required.

/JOB_TABLE_QUOTA=quota
Indicates the quota for a detached process' job symbol table. This is ignored for subprocesses. A value of 0 means an unlimited table size.

/MAILBOX=unit
Indicates the mailbox to be notified when the program completes. If not specified, no process is notified when the subprocess or detached program completes.

/MAXIMUM_WORKING_SET=quota
Indicates the WSLIMIT quota of the subprocess. If the quota exceeds that of the process owner, the EXQUOTA privilege is required.

/OUTPUT=filespec
Specifies the subprocess' SYS$OUTPUT value.

/PAGE_FILE=quota
Indicates the PGFLQUOTA quota of the subprocess. If the quota exceeds that of the process owner, the EXQUOTA privilege is required.

/PRIORITY=priority
Specifies the subprocess' priority. If this is greater than the priority of the current process, the ALTPRI privilege is required.

/PRIVILEGES=(privilege{,...})
Specifies the subprocess' privilege(s). To specify privileges you do not have, you must have the SETPRV privilege. If not specified, the subprocess is given the same set of privileges as the current privileges of the current process. If the privilege specified is "NOSAME", the subprocess will have no privileges.

/PROCESS_NAME=name
Specifies the subprocess' name. If not specified, the subprocess will have a null name.

/QUEUE_LIMIT=quota
Indicates the TQELM quota of the subprocess. If the quota exceeds that of the process owner, the EXQUOTA privilege is required.

/RESOURCEWAIT (default)
/NORESOURCEWAIT
If /RESOURCEWAIT is specified, the subprocess will hibernate until any requested resource becomes available. Otherwise an error is caused if a resource is unavailable.

/SCHEDULE=absolute-time
If specified, the subprocess will hibernate until the specified time.

/SERVICE_FAILURE
/NOSERVICE_FAILURE (default)
If /SERVICE_FAILURE is not specified, the subprocess will receive an error code when any system service call fails. Otherwise, all system services calls return a success code.

/SUBPROCESS_LIMIT=quota
Indicates the MAXJOBS quota of the subprocess. If the quota exceeds that of the process owner, the EXQUOTA privilege is required.

/SWAPPING (default)
/NOSWAPPING
/NOSWAPPING indicates that the subprocess cannot be swapped out of memory. Specifying this requires the PSWAPM privilege.

/USER=username
Indicates that the program is to execute in the context of the specified user. If the user is a different user, the IMPERSONATE privilege is required.

/WORKING_SET=quota
Indicates the WSQUOTA quota for the subprocess. If the quota exceeds that of the process owner, the EXQUOTA privilege is required.

/TIME_LIMIT=quota
Indicates the CPULIMIT quota of the subprocess. If the quota exceeds that of the process owner, the EXQUOTA privilege is required.

Examples

```
$ RUN SYS$SYSTEM:LOGOUT.EXE
$ RUN MYPROG/DETACH
```

SET {NO}ON

SET {NO}ON

Turns the error handling on and off.

Format
SET {NO}ON

Parameters
None.

Description
Use SET NOON to enable normal error handling, which is to report the error and exit the script (if not interactive). Use SET ON to disable normal error handling and allow the use of customer error handling set via the ON command.

Example
```
$ SET NOON
$ @Custom.com
$ SET ON
```


SUBROUTINE

SUBROUTINE

Indicates the start of a local subroutine in a command file. SUBROUTINE must be the first executable command after the label associated with the subroutine. See the CALL command for more details.

Format
SUBROUTINE

Parameters
None.

Example
```
$Test3:
$ SUBROUTINE
$ WRITE SYS$OUTPUT "This is Test3"
$ ENDSUBROUTINE
```


THEN

THEN

THEN is used as part of the IF command. See the description of the IF command for more information.

Format
THEN

WRITE

This command writes the specified data as one record to the specified open file.

Format

WRITE file expression{,...}

Parameters

file

This specifies a file - either a file handle returned by the OPEN command or one of the process permanent files identified by the SYS$INPUT, SYS$OUTPUT, SYS$ERROR, and SYS$COMMAND logicals.

expression

Specifies data to be written to the file. This can be a literal, or an expression consisting of variables, lexical functions, literals and operators. A list of expressions delimited by commas (,) can be specified. In that case, the expressions are concatenated and then written to the file.

Description

The WRITE command updates the file position when it completes. The file must have been opened with either the /WRITE or /APPEND qualifier on the OPEN command or the WRITE command will fail except for SYS$INPUT, SYS$OUTPUT, SYS$ERROR, and SYS$COMMAND files, which don't have to be explicitly opened in order to be written to.

Qualifiers

/ERROR=label
If an I/O error occurs during the operation, control transfers to the specified label. If this qualifier isn't defined, the current ON condition action is taken if an error occurs. The $STATUS global symbol contains the error code.

/SYMBOL
This qualifier is provided for compatibility with DCL, but has no effect in UCL. There is effectively no UCL limit on the size of data written with the WRITE command.

/UPDATE
Replaces the last record read with this data. This only applies to RMS files. The file must have been read prior to this command. If replacing a sequential file record, the data must be the same size.

/WAIT
/NOWAIT
If /NOWAIT is specified, the operation will complete immediately instead of synchronizing with another reader of the mailbox. /WAIT is the default.

Example

```
$ WRITE SYS$OUTPUT "Beginning update..."
```