```
process AIRPLANE
     call TOWER giving GATE yielding RUNWAY
     work TAXI.TIME (GATE, RUNWAY) minutes
     request 1 RUNWAY
     work TAKEOFF.TIME (AIRPLANE) minutes
     relinquish 1 RUNWAY
end " process AIRPLANE

process AIRPLANE
     call TOWER giving GATE yielding RUNWAY
     work TAXI.TIME (GATE, RUNWAY) minutes
     request 1 RUNWAY
     work TAKEOFF.TI
     relinquish 1
end " process AI
```

Operating System Interface User's Manual

CACI

SIMSCRIPT II.5 Operating System Interface (SOSI)

For product information or technical support contact:

CACI Products Company
1011 Camino Del Rio South, Suite 230
San Diego, CA 92108
Phone: (619) 542-5228
Fax: (619) 692-1013

TABLE OF CONTENTS

INTRODUCTION

SIMSCRIPT II.5 now gives you an **Operating System Interface** functions and routines. These functions allow easy access to typical operating system functions such as checking if a file exists, reading directories, asking for the current directory, changing directories, and a lot more.

The OS interface functions are intended to allow you to *write portable applications that use some OS functions*. They are *not* intended to provide access to *all* functions of your underlying operating system. Generally, all OSI functions are implemented on all platforms. However, some functions may not be meaningful on a given platform and then will just 'do nothing'. The '**Availability**' section in each routine's description will contain any remarks specific to a platform/ operating system (e.g. when a function is not available on a certain operating system).

The OS interface functions are not defined in the standard SIMSCRIPT II.5 runtime library and thereby available at all times. They need to be defined (as functions with return values) in your Preamble. All necessary OSI definitions are given in the APPENDIX.

Most functions in the OS interface return a value that indicates whether the requested operation was completed successfully. In the documentation we use the following constants that should be defined in your PREAMBLE when you use OS interface routines. They are defined as follows:

```
define .TRUE    to mean 1
define .FALSE   to mean 0
define .OSOK    to mean 0
define .OSERROR to mean -1
```

The OS interface routines are grouped into 3 sections. For each section we will give an overview that lists the available functions in that section. The individual functions will be described in the reference section in alphabetical order. All OS interface functions begin with "OS.".

Overview Pages:

1. Miscellaneous OS Queries
2. Files and Directories
3. Process Management

Reference:

4. Alphabetical List of OS Interface functions

Appendix:

OSI definition statements

Chapter 1 Miscellaneous OS Queries

For each function we list the name, the result mode, and a brief description. **RM**=Result modes: **T**=text, **I**=integer, **A**=Alpha, **P**=pointer. Two integer modes are designated specially to indicate the possible return values: **OI**=OS return code (=integer: .OSOK or .OSERROR), **BI**=boolean integer (=integer: .TRUE or .FALSE).

Name	RM	Description
OS.GET.OSTYPE.F	T	What Operating System are we on? (text name)
OS.GET.WINDOWSYSTYPE.F	T	What Window System are we on? (text name)
OS.GET.COMPUTERTYPE.F	T	What Computer Type are we on? (text name)
OS.GET.HOSTNAME.F	T	Returns host name (machine name on network)
OS.GET.ENVVAR.F	T	Returns value of environment variable
OS.SYSTEMTIME.F	I	Returns system time in seconds (integer)
OS.TIME2STRING.F	T	Converts time integer to string in fixed format
OS.VIEWHELP.F	OI	Call OS help viewer with given DB and context ID.

Chapter 2 Files and Directories

For each function we list the name, the result mode, and a brief description. **RM**=Result modes: **T**=text, **I**=integer, **A**=Alpha, **P**=pointer. Two integer modes are designated specially to indicate the possible return values: **OI**=OS return code (=integer: `.OSOK` or `.OSERROR`), **BI**=boolean integer (=integer: `.TRUE` or `.FALSE`).

Name	RM	Description
OS.MAX.FNAMELEN.F	I	Max. number of characters in a file name (not extension)
OS.GET.DIRSEPCHAR.F	A	Return directory separator character for this OS ("/" or "\")
OS.GET.CWD.F	T	Get current working directory
OS.GET.CURDRIVE.F	T	Get current drive (with colon) on PC's
OS.GET.PROGDIR.F	T	Find file in PATH
OS.MAKE.DIR.F	OI	Make new directory
OS.CHANGE.DIR.F	OI	Change to given directory
OS.REMOVE.DIR.F	OI	Remove existing directory (must be empty)
OS.FILE.EXISTS.F	BI	Check if given file exists
OS.DELETE.FILE.F	OI	Delete existing file
OS.RENAME.FILE.F	OI	Rename file
OS.OPEN.DIR.F	P	Open a directory for for listing files. Returns 'handle'.
OS.NEXT.DIRENTRY.F	T	Returns next directory entry for directory handle.
OS.CLOSE.DIR.F	OI	Closes directory with given handle.
OS.FILE.TYPE.F	I	Returns file type of given file
OS.TEST.ACCESS.F	BI	Check if we can access given file
OS.FILE.MODTIME.F	I	Returns time when file was last modified
OS.FILE.ACCESSTIME.F	I	Returns time when file was last accessed
OS.FILE.SIZE.F	I	Returns file size
OS.MAKE.EMPTY.FILE.F	OI	Makes empty file with given name
OS.MAKE.TMP.FILE.F	T	Creates unique temporary file in given directory
OS.COPY.FILE.F	OI	Copies file
OS.MATCHES.FILEPATTERN.F	BI	Does given file name match given pattern?
OS.IS.LEGAL.FNAMECHAR.F	BI	Is this a legal file name character
OS.INCR.FNAME.IFEX.F	T	'Increment file name' if given file already exists
OS.MAP2LEGALFNAME.F	T	Map string to legal file name on current OS
OS.APPEND.SLASH.F	T	Append slash to given path

OS.REMOVE.FINAL.SLASH.F	T	Remove final slash from given path
OS.DIRNAME.F	T	Get directory component of path (all but last)
OS.BASENAME.F	T	Get file component of path (last)

Chapter 3 Process Management

For each function we list the name, the result mode, and a brief description. **RM**=Result modes: **T**=text, **I**=integer, **A**=Alpha, **P**=pointer. Two integer modes are designated specially to indicate the possible return values: **OI**=OS return code (=integer: `.OSOK` or `.OSERROR`), **BI**=boolean integer (=integer: `.TRUE` or `.FALSE`).

Name	RM	Description
`OS.SYSTEMCALL.F`	`I`	Performs system call (OS cmd) and returns the return code
`OS.START.BGTASK.F`	`I`	Starts background task of given command, returns the PID
`OS.CHECK.BGTASK.R`	`-`	Check if any background task has finished
`OS.KILL.BGTASK.F`	`OI`	Kill background process with given PID
`OS.GET.PID.F`	`I`	Get PID of this program's process
`OS.GET.PPID.F`	`I`	Get parent's PID of this program's process

Chapter 4 Alphabetical List of OS Interface Functions

OS.APPEND.SLASH.F

Call: OS.APPEND.SLASH.F(PATH)

Arguments:

PATH Text: A path / directory name.

Result: Text: Path with appended slash.

Description: Appends a directory separator to the given path if it doesn't already have a trailing slash. Useful for creating full paths from a directory and a file name.

See also: OS.REMOVE.FINAL.SLASH.F, OS.DIRNAME.F, OS.BASENAME.F, OS.GET.DIRSEPCHAR.F

Example:

```
FULLPATH = concat.f( os.append.slash.f(PATH), FILENAME)
```

OS.BASENAME.F

Call: OS.BASENAME.F(PATH)

Arguments:

PATH Text: A path.

Result: Text: Directory component of PATH

Description: Returns the file name component of the given path, i.e. the last component in the path (separated by a directory separator). When the given PATH has a trailing slash, it is ignored. This means that this function returns the last 'real component' of the path, and not an empty string in this case.

See also: OS.DIRNAME.F, OS.APPEND.SLASH.F, OS.REMOVE.FINAL.SLASH.F, OS.GET.DIRSEPCHAR.F

OS.CHANGE.DIR.F

Call: OS.CHANGE.DIR.F(DIR)

Arguments:

DIR Text: Directory to switch to.

Result: Integer: .OSOK (0) for ok, .OSERROR (-1) for error (e.g. directory didn't exist).

Description: Tries to set the 'current working directory' to the given directory. When DIR is not a full path, DIR must exist as a directory in the current working directory.

See also: OS.MAKE.DIR.F, OS.REMOVE.DIR.F

OS.CHECK.BGTASK.R

Call: call OS.CHECK.BGTASK.R yielding PID, EXITCODE, STATUS

Arguments:

PID Integer: Process ID (PID) of a terminated process. Special values:
PID = -1 means 'no child processes present', PID = 0 means 'no background processes have terminated yet'

EXITCODE Integer: Exit code of terminated background process (passed out with EXIT.R).

STATUS Integer: Status of terminated process: 0 for normal termination. When <> 0, it is the number of the signal that terminated the process.

Description: Checks if any background tasks have finished. If so, the PID, exit code and the status of the terminated process are returned in the yielded arguments. This call always returns, whether a background process did terminate or not. You should always call this routine within a loop since there can always be multiple background processes that have terminated.

See also: OS.STARTBGTASK.F, OS.KILLBGTASK.F, OS.GET.PID.F

Example and description of process model: see OS.STARTBGTASK.F

OS.CLOSE.DIR.F

Call: OS.CLOSE.DIR.F(DIRHNDL)

Arguments:

DIRHNDL Pointer: Handle to directory that was returned from OS.OPEN.DIR.F

Result: Integer: .OSOK (0) for ok, .OSERROR (-1) for error.

Description: Closes a directory that was opened for reading with OS.OPEN.DIR.F. The open call returns the directory handle that is used to refer to that directory for reading and for closing the directory.

See also: OS.OPENDIR.F, OS.NEXT.DIRENTRY.F

Example: see OS.OPENDIR.F

OS.COPY.FILE.F

Call: OS.COPY.FILE.F(FROM, TO)

Arguments:

FROM Text: Name of file to copy from.

TO Text: Name of file to copy to.

Result:

Description: Copies a whole file.

OS.DELETE.FILE.F

Call: OS.DELETE.FILE.F(FNAME)

Arguments:

FNAME Text: Name of file to be deleted.

Result: Integer: .OSOK (0) for ok, .OSERROR (-1) for error.

Description: Deletes file.

OS.DIRNAME.F

Call: OS.DIRNAME.F(PATH)

Arguments:

PATH Text: A full path with ior without trailing slash.

Result: Text: The directory component of the full path specified in PATH.

Description: Splits the directory and file name component in the given PATH.

See also: OS.BASENAME.F, OS.APPENDSLASH.F, OS.REMOVE.FINAL.SLASH.F, OS.GET.DIRSEPCHAR.F

OS.FILE.EXISTS.F

Call: OS.FILE.EXISTS.F(FNAME)

Arguments:

FNAME Text: Name of file to be checked for existance. Can be local file name (in current directory) or fully specified path.

Result: Integer: .TRUE (1) for file exists, .FALSE (0) for error.

Description: Checks if a given file exists.

OS.FILE.TYPE.F

Call: OS.FILE.TYPE.F(FNAME)

Arguments:

FNAME Text: Name of file whose file type is requested. Can be full path or local file name.

Result: Integer: A code is returned with the following meanings (not all may apply to every platform):
0 : unknown file type (should never happen)
1 : directory
2 : ordinary file
3 : character special (Unix only)
4 : block special (Unix only)
5 : FIFO/ pipe (Unix only)

Description: Returns the file type of the given file.

See also: OS.FILE.ACCESSTIME.F, OS.FILE.MODTIME.F, OS.FILE.SIZE.F

OS.FILE.ACCESSTIME.F

Call: OS.FILE.ACCESSTIME.F(FNAME)

Arguments:

FNAME Text: File name (local of full path).

Result: Integer: Time of last access to the given file.

Description: Checks time of last access to the given file. This time integer represents the local time on the computer in seconds since a fixed point in time in the past (depends on machine). This integer in itself can be used to compare two files to see which one is 'newer'. Use OS.TIME2STRING.F to convert a time integer to a fixed format human readable string.
On platforms that don't keep separate times for access and modification of a file (e.g. DOS), the modification time will be returned.

See also: OS.FILE.MODTIME.F

OS.FILE.MODTIME.F

Call: OS.FILE.MODTIME.F(FNAME)

Arguments:

FNAME Text: File name (local or full path)

Result: Integer: Time of last modification of this file.

Description: Checks time of last modification of the given file. This time integer represents the local time on the computer in seconds since a fixed point in time in the past (depends on machine). This integer in itself can be used to compare two files to see which one is 'newer'. Use OS.TIME2STRING.F to convert a time integer to a fixed format human readable string.

See also: OS.FILE.ACCESSTIME.F

OS.FILE.SIZE.F

Call: OS.FILE.SIZE.F(FNAME)

Arguments:

FNAME Text: File name (local or full path)

Result: Integer: Size of given file in bytes.

OS.GET.COMPUTERTYPE.F

Call: OS.GET.COMPUTERTYPE.F

Arguments: none

Result: Text: A text that identifies the type of computer the program is running on. Currently supported are:

"PC x86"	PC's with x86 CPU (386 and up)
"SPARC"	Sun Sparc's
"VAX"	VAXes
"DECSTATION"	DECstation (MIPS processor)
"SGI"	Silicon Graphics Indigo et. al
"RS6000"	IBM RS/6000
"NEXT"	The NeXT Computer Hardware (black box)
"HP700"	HP 700 family
"MOTOROLA 88"	Motorola based on 88100
"ALPHA"	DEC's Alpha Processor

Description: Returns the computer type on which the program is running. Note that this concerns the *hardware* of the platform. The same computer may be able to run several operating systems and window systems.

See also: OS.GET.OSTYPE.F, OS.GET.WINDOWSYSTYPE.F

OS.GET.CURDRIVE.F

Call: OS.GET.CURDRIVE.F

Arguments: none

Result: Text: For Windows and OS/2: Current drive letter with trailing colon. For Unix: ""

Description: Returns the current drive. On some operating systems, the drive is part of the a full path.

See also: OS.GET.CWD.F, OS.GET.DIRSEPCHAR.F

OS.GET.CWD.F

Call: OS.GET.CWD.F

Arguments: none

Result: Text: Full path to current working directory.

See also: OS.CHANGE.DIR.F

OS.GET.DIRSEPCHAR.F

Call: OS.GET.DIRSEPCHAR.F

Arguments: none

Result: Alpha: The character that is used as a directory as a directory separator in paths. This is an operating system specific constant.

OS.GET.ENVVAR.F

Call: OS.ENVVAR.F(EVNAME)

Arguments:

EVNAME Text: Name of environment variable.

Result: Text: The value of the given environment variable EVNAME

Availability: All platforms

OS.GET.HOSTNAME.F

Call: OS.GET.HOSTNAME.F

Arguments: none

Result: Text: Name of the host computer that is used on the network. Currently supported only for Unix. For other platforms, it returns "".

Description: Returns the host name of the current machine (used for all network communications to identify the machine).

Availability: Unix only.

OS.GET.OSTYPE.F

Call: OS.GET.OSTYPE.F

Arguments: none

Result: Text: A text is returned that represents the operating system running the SIMSCRIPT II.5 program. Currently supported:

"OS2"	OS/2 2.0 and up
"WIN32"	Windows NT
"WIN16"	Windows 3.1 (or later) on DOS (16 bit)
"SUNOS"	SunOS (not Solaris!)
"SOLARIS"	Sun Solaris
"VMS"	VMS (DEC)
"ULTRIX:	Ultrix (DEC)
"IRIX"	Irix (Silicon Graphics)
"HPUX"	HP-UX (HP)
"AIX"	AIX (IBM RS/6000)
"NEXTSTEP"	NextSTEP (NeXT)
"M88"	Motorola Unix (M88)
"SCOUNIX"	SCO Unix

OS.GET.PID.F

Call: OS.GET.PID.F

Arguments: none

Result: Integer: The process ID for the process running the SIMSCRIPT program.

See also: OS.GET.PPID.F, OS.START.BGTASK.F, OS.KILL.BGTASK.F

Example: see OS.START.BGTASK.F

OS.GET.PPID.F

Call: OS.GET.PPID.F

Arguments: none

Result: Integer: The process ID **of the parent** of the process running the SIMSCRIPT program (i.e. the program start started this program as a background task).

Availability: Not available on Windows.

See also: OS.GET.PID.F, OS.START.BGTASK.F, OS.KILL.BGTASK.F

Example: see OS.START.BGTASK.F

OS.GET.PROGDIR.F

Call: OS.GET.PROGDIR.F(FNAME)

Arguments:

FNAME Text: The file name of the program to be searched for in the PATH.

Result: Text:

Description: Allows to find programs. When FNAME is a full path (contains directory separator characters), just directory part is returned. In that case it is equivalent to OS.DIRNAME.F. When FNAME is just a file name, this function looks through all directories specified in the PATH environment variable and searches if it can find the given FNAME in any of the directories. If found, the full path is returned, otherwise the empty string is returned.
The directories are searched in the order in which they are mentioned in the PATH environment variable. The first found file is returned.

OS.GET.WINDOWSYSTYPE.F

Call: OS.GET.WINDOWSYSTYPE.F

Arguments: none

Result: Integer: Code representing the window system the SIMSCRIPT II.5 program is running under. The currently available codes are:

"PM"	OS/2 2.0 Presentation Manager
"WIN32"	Windows NT Windows WIN32 API
"WINDOWS"	Windows 3.1 (or later) on DOS (16 bit)
"OPENLOOK"	Sun's Openlook
"MOTIF"	Motif
"DECWINDOWS"	DEC's DecWindows
"NEXTSTEP"	NeXTStep

Description: This call queries for the window system. You should normally never have to worry about the window system you are running under since SIMGRAPHICS (I or II) will take care of the details. On occasion, you might want to do OS calls e.g. to start up a special window. In that case the commands are window system specific and you must know which window system you are on, in order to write portable code.

See also: OS.GET.OSTYPE.F, OS.GET.COMPUTERTYPE.F

OS.INCR.FNAME.IFEX.F

Call: OS.INCR.FNAME.IFEX.F(FNAME)

Arguments:

FNAME Text: Name of file you want to use. It can be a local file (in the current directory), or a full path. The format of the FNAME argument is:

{path}filename{idx}{extension}

Result: Text: A unique and non-existant file name (in the current directory).

Description: 'Increment file name if file already exists': This function supports file name mapping on systems that have limited length for file names (eg. DOS). When you need a new file and this file already exists, you may want to use a file name that is 'close' to the original, just differing by a running index at the end of the name part.
Algorithm: When FNAME doesn't exist, it is immediately returned. If it exists, and *idx* (an integer number part at the end of the file name) is not present, *idx* is set to 0. Otherwise *idx* is incremented until the resulting file (or path) doesn't yet exist.

See also: OS.IS.LEGAL.FNAME.CHAR, OS.MAP2LEGALFNAME.F

Example: assume the DOS operating system (8 characters for file name)

```
  "-- user enters free form text for file name
write as "Please enter free-form name for ...", +
read FREETEXT.NAME
  "-- user entered: "Network Simulation Model"
  "-- map to a legal file name
LEGAL.FNAME = OS.MAP2LEGALFNAME.F(FREETEXT.NAME)
  "-- resulted in "NETWORKS"
  "-- append extension
LEGAL.FNAME = concat.f(LEGAL.FNAME, ".db")
  "-- make sure the file name is unique (no overwrite)
FNAME = OS.INCR.FNAME.IFEX.F(LEGAL.FNAME)
  "-- "NETWORKS.DB" existed, returned "NETWORK0.DB"

  "-- now use the new unique file name ...
...
```

OS.IS.LEGAL.FNAMECHAR.F

Call: OS.IS.LEGAL.FNAMECHAR.F(CHAR)

Arguments:

CHAR Alpha: Character to be tested.

Result: Integer: .TRUE (1) if CHAR is a legal character to be used in a file name on the current operating system, .FALSE (0) otherwise.

Description: Checks if a character is legal in a file name. Note that this function checks only for one character at a time. It doesn't notice of course, that a file name under DOS cannot have two extension dots etc. To map a general file name or general string to a legal file name under the current operating system, use OS.MAP2LEGALFNAME.F.

See also: OS.MAP2LEGALFNAME.F

Example: see OS.INCR.FNAME.IFEX.F

OS.KILL.BGTASK.F

Call: OS.KILL.BGTASK.F(PID)

Arguments:

PID Integer: The Process ID (PID) of the process to be killed.

Result: Integer: .OSOK (0) for ok, .OSERROR (-1) for error. You must have 'permission' to kill the process.

Description: Aborts a background process previously started by OS.START.BGTASK.F. The kill is performed by a 'kill signal' so that OS.CHECK.BGTASK.R for the process PID will return a non-zero PROC.STATUS indicating that that process was killed by an external signal.

Availability: Not available on Windows.

See also: OS.CHECK.BGTASK.R, OS.START.BGTASK.F, OS.SYSTEMCALL.F, OS.GET.PID.F, OS.GET.PPID.F

Example: see OS.START.BGTASK.F

OS.MAKE.DIR.F

Call: OS.MAKE.DIR.F(DIRNAME)

Arguments:

DIRNAME Text: Name of directory to be created. Can be just a name or a full path.

Result: Integer: .OSOK (0) for ok, .OSERROR (-1) for error.

Description: Creates a new subdirectory. On many operating system, when a full path is given, the parent directory of DIRNAME must exist (i.e. OS.MAKE.DIR.F can create only the last level of subdirectory, but not missing intermediate subdirectories).

See also: OS.CHANGE.DIR.F, OS.REMOVE.DIR.F

OS.MAKE.EMPTY.FILE.F

Call: OS.MAKE.EMPTY.FILE.F(PATH)

Arguments:

PATH Text: Name (or full path) of file to be created.

Result: Integer: .OSOK (0) for ok, .OSERROR (-1) for error.

Description: Creates an empty file with the given name. If the file already exists, it is overwritten.

OS.MAKE.TMP.FILE.F

Call: OS.MAKE.TMP.FILE.F(DIR)

Arguments:

DIR Text: Directory where the temporary file is to be created. "" or "." means 'current directory'.

Result: Text: Name of the temporary file that was created.

Description: Creates a unique temporary file in the given DIRectory. This function can be used for intermediate files. Note that these temporary files are not automatically removed when the program terminates.

See also: OS.MAKE.EMPTY.FILE, OS.INCR.FNAME.IFEX.F

Example:

```
TEMPDIR = OS.GET.ENVVAR.F("TEMP")
        TEMPFILE = OS.MAKE.TMP.FILE.F(TEMPDIR)
```

OS.MAP2LEGALFNAME.F

Call: OS.MAP2LEGALFNAME.F(FTEXT)

Arguments:

FTEXT Text: An arbitrary free text that will be mapped to a file name.

Result: Text: A legal file name on the current operating system.

Description: All characters that are illegal in file names (OS.IS.LEGAL.FNAMECHAR.F) are deleted from FTEXT and the resulting string is truncated to the maximum allowable file name length (OS.MAX.FNAMELEN.F).

See also: OS.IS.LEGAL.FNAMECHAR.F, OS.MAX.FNAMELEN.F, OS.INCR.FNAME.IFEX.F

Example: see OS.INCR.FNAME.IFEX.F

OS.MATCHES.FILEPATTERN.F

Call: OS.MATCHES.FILEPATTERN.F(STR, PAT)

Arguments:

STR	Text: String that is to be matched (generally a file name)
PAT	Text: Pattern (contains "*" and "?" wild cards)

Result: Integer: .TRUE in case of match, .FALSE otherwise

Description: This function implements the pattern matching that is typically used for file names. It is intended to be used when filtering files in reading directories. The wild card characters "*" and "?" have their usual meaning ("*"=sequence of any_char, "?"=any_char).
Algorithm: The matching of PAT against STR is done as a FORWARD comparison: A "?" matches exactly one any_char in STR, a "*" matches a sequence (0 or more) any_chars in STR, *except* the one character that follows the "*" in PAT. For example, "*." matches "abc.", but not "abc.d.".

See also: OS.OPEN.DIR.F, OS.NEXT.DIRENTRY.F

Example:

```
"-- open current directory and print all SIM files
DIRHNDL = OS.OPEN.DIR.F("")
FNAME = OS.NEXT.DIRENTRY.F(DIRHNDL)
while FNAME <> ""
do
  if OS.MATCHES.FILEPATTERN.F(FNAME, "*.sim")
    write FNAME as "SIM File = ", T *, /
  endif
  FNAME = OS.NEXT.DIRENTRY.F(DIRHNDL)
loop
RC = OS.CLOSE.DIR.F(DIRHNDL)
```

OS.MAX.FNAMELEN.F

Call: OS.MAX.FNAMELEN.F

Arguments: none

Result: Integer: The maximum number of characters safely allowed in a file name on the current operating system. This *does not include the file extension!* On DOS, this number is 8. On Unix, where there is no fixed separation between file name and extension part, the full length is returned.

Note: This number is not necessarily the maximum file name length on any given file system (numbers may differ between file systems), but the length that can safely be assumed for *all* file systems.

OS.NEXT.DIRENTRY.F

Call: OS.NEXT.DIRENTRY.F(DIRHNDL)

Arguments:

DIRHNDL Pointer: Directory handle that was returned when the directory was opened.

Result: Text: Next file or directory name in the directory, or "" when no more entries are found.

Description: Returns the next file or directory name in the directory associated with DIRHNDL. Note that you can open and read multiple different directories at the same time by using several OS.OPEN.DIR.F with different DIRHNDL pointers.

See also: OS.OPEN.DIR.F, OS.CLOSE.DIR.F, OS.MATCHES.FILEPATTERN.F

Example: see OS.OPEN.DIR.F

OS.OPEN.DIR.F

Call: OS.OPEN.DIR.F(DIR)

Arguments:

DIR Text: The directory to be opened. "" and "." means the current directory. You can specify a local subdirectory and a full path to any directory in the file system.

Result: Pointer: A *Directory Handle* that is associated with this directory.

Description: Opens the given directory for reading. A directory handle is returned that is associated with this directory. Using different OS.OPEN.DIR.F and different directory handles (in OS.NET.DIRENTRY.F) allows you to read several directories and the same time. A directory opened with OS.OPEN.DIR.F should be closed with OS.CLOSE.DIR.F after reading is complete to free up internal storage associated with the directory handle.

See also: OS.CLOSE.DIR.F, OS.NEXT.DIRENTRY.F

Example:

```
"-- read and print the current directory
DIRHNDL = OS.OPEN.DIR.F("")
FNAME = OS.NEXT.DIRENTRY.F(DIRHNDL)
while FNAME <> ""
do
  FNAME = OS.NEXT.DIRENTRY.F(DIRHNDL)
loop
RC = OS.CLOSE.DIR.F(DIRHNDL)
```

OS.REMOVE.DIR.F

Call: OS.REMOVE.DIR.F(DIRNAME)

Arguments:

DIRNAME Text: Name of directory to be removed. Local name or full path.

Result: Integer: .OSOK for success, .OSERROR for error

Description: Removes the given directory. Error conditions: Directory didn't exist or wasn't empty, insufficient permissions etc.

See also: OS.MAKE.DIR.F

OS.REMOVE.FINAL.SLASH.F

Call: OS.REMOVE.FINAL.SLASH.F(PATH)

Arguments:

PATH Text: A path name that may or may not have a trailing slash (directory separator).

Result: Text: PATH without a trailing slash

Description: Removes a possibly trailing slash (directory separator for this operating system, query with OS.GET.DIRSEPCHAR.F) contained in PATH. This is the inverse operation to OS.APPEND.SLASH.F

See also: OS.APPEND.SLASH.F

OS.RENAME.FILE.F

Call: OS.RENAME.FILE.F(OLD, NEW)

Arguments:

OLD Text: Name of existing file
NEW Text: New name of file

Result: Integer: .OSOK for ok, .OSERROR for error.

Description: Renames a file. On most operating system, you can also use this function to rename directories.

OS.START.BGTASK.F

Call: OS.START.BGTASK.F(OSCMD, MINIMIZED)

Arguments:

OSCMD Text: Operating system command to be executed (shell command, just as typed on the command line). See Note for Windows below!

MINIMIZED Integer: When <> 0, the application started by OSCMD will run minimized (just an icon but no window is shown).

Result: Integer: The PID (process ID, integer > 0) of the started background process, or -2 for "command empty" or -1 for "process start failed".

Description: This function allows to start background tasks. Each background task (as well the current program) is a process that has a *process ID* (PID). Using this PID, you can later call OS.CHECK.BGTASK.R to see if the process is finished and get the result code, and/or OS.KILL.BGTASK.F to kill (abort) a process.
Note that this is an *asynchronous* exection of the given OSCMD. To execute a command synchronously, i.e. wait until the command is finished, use OS.SYSTEMCALL.F.

Note for Windows: In Windows 3.1 you can only start Windows programs this way, but not DOS programs. DOS programs must be started using "PIF" files that describe which DOS program to run, in what 'working directory', whether to run minimized or not, etc. To start a DOS program from your SIMSCRIPT program, you must create a "PIF" file, e.g. "MYPROG.PIF" using the Windows PIF editor. In the "Program Filename" field you must enter the name of the DOS program, in the "Optional Parameters" you can enter command line arguments, and in the "Start-up Directory" you can specify the working directory for this DOS program. To start the DOS program, you can then call

```
RC = OS.START.BGTASK.F("MYPROG.PIF", 0)
```

from your SIMSCRIPT program. The PIF file must be in your path!

See also: OS.CHECK.BGTASK.R, OS.KILL.BGTASK.F, OS.SYSTEMCALL.F

OS.SYSTEMCALL.F

Call: OS.SYSTEMCALL.F(OSCMD, MINIMIZED)

Arguments:

OSCMD Text: Operating system command to be executed. See Note for Windows in the OS.START.BGTASK.F documentation!

MINIMIZED Integer: When <> 0, the application started by OSCMD will run minimized (just an icon but no window is shown).

Result: Integer: Return code from the executed command.

Description: Executes the given command and returns its return code. This call executes synchronously, i.e. waits until the OSCMD is completed. To execute a program or issue an OS command in the background, i.e. asynchronously, use OS.START.BGTASK.F.

See also: OS.START.BGTASK.F, OS.CHECK.BGTASK.R

Example:

```
"-- Unix: list directory and print it using lpr
RC = OS.SYSTEMCALL.F("ls *.sim | lpr", 0)
```

OS.SYSTEMTIME.F

Call: OS.SYSTEMTIME.F

Arguments: none

Result: Integer representing the current system time in seconds since a fixed point in the past.

Description: Returns the number of seconds since a fixed point in the past (the reference point may differ on different platforms). This essentially is the 'system clock'. To get a human readable form of the current system time, use OS.TIME2STRING.F.

See also: OS.TIME2STRING.F

OS.TEST.ACCESS.F

Call: OS.TEST.ACCESS.F(FNAME, AMODE)

Arguments:

FNAME Text: Name of file (or directory) to be tested. Local file or full path.
AMODE Integer: Code of the access mode to be checked.
The values are:
0: read
1: write
2: ReadandWrite
3: Execute
4: File exists

Result: Integer: .TRUE when access permitted, .FALSE otherwise

Description: Checks if a given file can be accessed by the current program in the given access mode AMODE. On operating systems that do not support access mode control, this function always returns .TRUE.

OS.TIME2STRING.F

Call: OS.TIME2STRING.F(ITIME)

Arguments:

ITIME Integer: Integer representing a time.

Result: Text: Fixed text format of the time, 24 characters long. The format is:

"Sat Apr 13 15:33:00 1991"

Description: Transforms an integer representation of time ITIME to a text representation. The argument ITIME usually comes from a call to OS.SYSTEMTIME.F or OS.FILE.MODTIME.F / OS.FILE.ACCESSTIME.F.

See also: OS.SYSTEMTIME.F

OS.VIEWHELP.F

Call: OS.VIEWHELP.F(HLPDB, CONTEXTID)

Arguments:

HLPDB Text: Name (file name) of the help data base, e.g. "SIMDEBUG.HLP"
CONTEXTID Integer: Context ID for context sensitive help. A 0 (zero) causes the contents page to be displayed.

Result: Text:

Description: This function calls the system specific help viewer to display help from a help data base. A context ID can be given for context sensitive help, i.e. to immediately jump to a topic page. A context ID of 0 causes the contents page (top page of help) to be displayed.

Availability: On PC Windows platform only.

APPENDIX A OSI Definition Statements

```
'' Preamble definitions for Operating System Interface routines
'' have to be included in user's  preamble

  normally mode is undefined

  define .TRUE       to mean 1
  define .FALSE
    to mean 0
  define .OSOK       to mean 0
  define .OSERROR    to mean -1

  ''Miscelaneous OS Queries

  define OS.GET.OSTYPE.F          as text function given 0
  define OS.GET.WINDOWSYSTYPE.F   as text function given 0
  define OS.GET.COMPUTERTYPE.F    as text function given 0
  define OS.GET.HOSTNAME.F        as text function given 0
  define OS.GET.ENVVAR.F          as text function given 1
  define OS.GET.PROGDIR.F         as text function given 1
  define OS.SYSTEMTIME.F          as integer function given 0
  define OS.TIME2STRING.F         as text function given 1
  define OS.VIEWHELP.F            as integer function given 2

  ''Files and Directories

  define OS.MAX.FNAMELEN.F        as integer function given 0
  define OS.GET.DIRSEPCHAR.F      as alpha function given 0
  define OS.GET.CWD.F             as text function given 0
  define OS.GET.CURDRIVE.F        as text function given 0
  define OS.MAKE.DIR.F            as integer function given 1
  define OS.CHANGE.DIR.F          as integer function given 1
  define OS.REMOVE.DIR.F          as integer function given 1
  define OS.FILE.EXISTS.F         as integer function given 1
  define OS.DELETE.FILE.F         as integer function given 1
  define OS.RENAME.FILE.F         as integer function given 1
  define OS.OPEN.DIR.F            as pointer function given 1
  define OS.NEXT.DIRENTRY.F       as text function given 1
  define OS.CLOSE.DIR.F           as integer function given 1
  define OS.FILE.TYPE.F           as integer function given 1
  define OS.TEST.ACCESS.F         as integer function given 2
  define OS.FILE.MODTIME.F        as integer function given 1
  define OS.FILE.ACCESSTIME.F     as integer function given 1
  define OS.FILE.SIZE.F           as integer function given 1
  define OS.MAKE.EMPTY.FILE.F     as integer function given 1
  define OS.MAKE.TMP.FILE.F       as text function given 1
  define OS.COPY.FILE.F           as integer function given 2
  define OS.MATCHES.FILEPATTERN.F as integer function given 2
  define OS.IS.LEGAL.FNAMECHAR.F  as integer function given 1
  define OS.INCR.FNAME.IFEX.F     as text function given 1
  define OS.MAP2LEGALFNAME.F      as text function given 1
  define OS.APPEND.SLASH.F        as text function given 1
  define OS.REMOVE.FINAL.SLASH.F  as text function given 1
  define OS.DIRNAME.F             as text function given 1
  define OS.BASENAME.F            as text function given 1

  ''Process Management

  define OS.SYSTEMCALL.F          as integer function given 2
  define OS.START.BGTASK.F        as integer function given 2
```

```
        define OS.CHECK.BGTASK.R        as routine yielding 3
        define OS.KILL.BGTASK.F         as integer function given 1
        define OS.GET.PID.F             as integer function given 0
        define OS.GET.PPID.F            as integer function given 0
  ''end OSI definitions
```