

StrongForth 3.1 Glossary: msvcr

-1?error (single -- 1st)

If `single` is equal to -1, throw an exception with the error code returned by `errno`. `1st` is equal to `single`.

. (zero-terminated-string --)

strongforth.sf

Send zero-terminated-string to the default output stream.

0?error (single -- 1st)

If `single` is equal to zero, throw an exception with the error code returned by `errno`. `1st` is equal to `single`.

?error (single --)

If `single` is not equal to zero, throw an exception with the error code returned by `errno`.

abort (--)

Abort StrongForth and return control to the operating system.

`abort` calls the *MSVCRT* library function `abort`.

argc (-- unsigned)

`unsigned` is the number of command-line arguments passed to StrongForth.

argv (-- address -> zero-terminated-string)

`address -> zero-terminated-string` is the address of a list of zero-terminated strings that contain the command-line arguments passed to StrongForth. The first item of the list is the name of the StrongForth executable.

asctime (address -> unsigned -- zero-terminated-string)

Convert a time represented by an array of 9 unsigned single-cell values into zero-terminated-string. zero-terminated-string has the fixed format

DOW MMM DD hh:mm:ss YYYY\n\0

with DOW as the day of week in three letters, MMM as the month in three letters, DD as the day of month, hh as hours, mm as minutes, ss as seconds and YYYY as the year. `address ->` `unsigned` is the address of the first item of the array, whose structure is as follows:

item #	Description
0	seconds (0 - 59)
1	minutes (0 - 59)

2	hours (0 - 23)
3	day of month (1 - 31)
4	month (0 - 11; January = 0)
5	year minus 1900
6	day of week (0 - 6; Sunday = 0)
7	day of year (0 - 365; January 1 = 0)
8	0

`asctime` calls the *MSVCRT* library function `asctime` with address `-> unsigned as timeptr`.

`atexit (address -- integer)`

Add the function with code address `address` to the list of functions that are to be called when StrongForth exits normally. The function may not have parameters. `integer` is zero if and only if `atexit` was executed successfully.

`atexit` calls the *MSVCRT* library function `atexit` with `address` as `func`.

`bsearch (address unsigned unsigned address address -- 4 th)`

Perform a binary search of a sorted array. The first `address` is a pointer to a callback function with two array arguments as parameters and a signed integer as return value. The first `unsigned` is the width of each element of the array in bytes. The second `unsigned` is the number of elements of the array. The second `address` is the address of the first element of the array. The third `address` is pointer to the key element to search for.

The return value of the callback function is less than zero if the first argument is less than the second one, greater than zero if the first argument is greater than the second one, or equal to zero if both arguments are equal.

`4 th` is the address of an array element that matches the one the third `address` points to, or null if there is no match.

`bsearch` calls the *MSVCRT* library function `bsearch` with the first `address` as `compare`, the first `unsigned` as `width`, the second `unsigned` as `num`, the second `address` as `base` and the third `address` as `key`.

`buffer-mode (stack-diagram -- 1st)`

When used in a stack diagram, specifies an input or output parameter with data type `buffer-mode`.

`calloc (unsigned unsigned -- address)`

Allocate a consecutive chunk of memory for a number of items with a given size. The first `unsigned` is the size of each item in bytes. The second `unsigned` is the total number of items. Initialize each item with zero. `address` is the address of the first allocated item.

`calloc` calls the *MSVCRT* library function `calloc` with the first `unsigned` as `size` and the second `unsigned` as `number`.

`clearerr (file --)`

Reset the error indicator and the end-of-file indicator for *file*.

`clearerr` calls the *MSVCRT* library function `clearerr` with *file* as *stream*.

clock (-- unsigned)

unsigned is the number of milliseconds elapsed since initialization of the *MSVCRT* library, or -1 if an error occurred.

`clock` calls the *MSVCRT* library function `clock`.

ctime (address -> signed -- zero-terminated-string)

Convert a time represented as the number of seconds elapsed since midnight (00:00:00), January 1, 1970, of the local time zone, into zero-terminated-string. *address -> signed* is a pointer to where the number of seconds is stored. *zero-terminated-string* has the fixed format

DOW MMM DD hh:mm:ss YYYY\n\0

with DOW as the day of week in three letters, MMM as the month in three letters, DD as the day of month, hh as hours, mm as minutes, ss as seconds and YYYY as the year.

`ctime` calls the *MSVCRT* library function `ctime` with *address -> signed* as *sourceTime*.

cwait-action (stack-diagram -- 1st)

When used in a stack diagram, specifies an input or output parameter with data type *cwait-action*.

doserrno (-- address -> signed)

address -> signed is the address of a cell that contains the error code of the most recent *MSVCRT* I/O operation that failed. The error code might differ from the one returned by `errno`.

environ (-- address -> zero-terminated-string)

address -> zero-terminated-string is the address of a list of zero-terminated strings that constitute the process environment of StrongForth.

EOF (-- character)

msvcrt.sf

character has the value -1, which is used by some functions of the *MSVCRT* library as end-of-file indicator.

errno (-- address -> signed)

address -> signed is the address of a cell that contains the error code of the most recent *MSVCRT* operation that failed.

exit (integer --)

Call, in reverse order, the functions registered by `atexit` and `_onexit`. Flush all I/O buffers and close all open streams. Terminate StrongForth and return to the caller of StrongForth. `integer` is made available to the caller as a status value.

`exit` calls the *MSVCRT* library function `exit` with `integer` as status.

`fclose (file -- flag)`

Flush and close the stream `file`. `flag` is false if and only if the operation succeeds.

`fclose` calls the *MSVCRT* library function `fclose` with `file` as stream.

`feof (file -- integer)`

`integer` is nonzero if and only if an attempt was made to read past the end of `file`.

`feof` calls the *MSVCRT* library function `feof` with `file` as stream.

`ferror (file -- integer)`

`integer` is nonzero if and only if an error occurred reading from or writing to `file`.

`ferror` calls the *MSVCRT* library function `ferror` with `file` as stream.

`fflush (file -- flag)`

Flush `file`. If `file` was opened in write mode, the contents of the stream buffer are written to the file or device and the buffer is discarded. If `file` was opened in read mode, or if the stream has no buffer, `fflush` has no effect, and any buffer is retained.

`flag` is false if and only if the operation succeeded.

`fflush` calls the *MSVCRT* library function `fflush` with `file` as stream.

`fgetc (file -- character)`

Read a single character `character` from the current position of `file`. Increment the associated file pointer. `character` is equal to EOF if the operation fails.

`fgetc` calls the *MSVCRT* library function `fgetc` with `file` as stream.

`fgetpos (address file -- integer)`

Store the current position of `file` at `address`. `fsetpos` can later use the information stored at `address` to restore this position. `integer` is zero if and only if the operation was successful.

`fgetpos` calls the *MSVCRT* library function `fgetpos` with `address` as `pos` and `file` as stream.

`fgets (file unsigned address -> character -- zero-terminated-string)`

Read up to `unsigned` minus 1 characters from the current position of `file` and store them at `address -> character`, appending a terminating null character. Characters are read up to the first newline character or the end of the file. The newline character, if read, is included in the

string. Advance the current position of `file` by the number of characters read.
`zero-terminated-string` is the zero-terminated string starting at `caddress` ->
character or null if the operation fails.

`fgets` calls the *MSVCRT* library function `fgets` with `file` as stream, unsigned as
`numChars` and `caddress` -> character as `str`.

`file-control (stack-diagram -- 1st)`

When used in a stack diagram, specifies an input or output parameter with data type
`file-control`.

`file-descriptor (stack-diagram -- 1st)`

When used in a stack diagram, specifies an input or output parameter with data type
`file-descriptor`.

`file-status (stack-diagram -- 1st)`

When used in a stack diagram, specifies an input or output parameter with data type
`file-status`.

`fopen (zero-terminated-string zero-terminated-string -- file)`

Open the file whose path is specified by the second `zero-terminated-string` in the mode
specified by the first `zero-terminated-string`, and return its file handle `file`. `file` is
null if the operation fails.

`fopen` calls the *MSVCRT* library function `fopen` with the first `zero-terminated-string`
as mode and the second `zero-terminated-string` as filename.

`fputc (file character -- 2nd)`

Write `character` to `file` at the file position and increment the file position indicator if
required. `2nd` is character.

`fputc` calls the *MSVCRT* library function `fputc` with `file` as stream and character as `c`.

`fputs (file zero-terminated-string -- integer)`

Write `zero-terminated-string` excluding the terminating null character to the current
position of `file`.

`fputs` calls the *MSVCRT* library function `fputs` with `file` as stream and
`zero-terminated-string` as `str`.

`fread (file unsigned unsigned address -- 2nd)`

Read multiple items from `file` and store them in a buffer starting at `address`. The first
`unsigned` is the maximum number items read. The second `unsigned` is the size in bytes of
each item. Advance the file pointer associated with `file` by the number of bytes actually read.
`2nd` is the number of items actually read.

`fread` calls the *MSVCRT* library function `fread` with `file` as stream, the first unsigned as count, the second unsigned as size and address as buffer.

`free (address --)`

Return the contiguous memory space starting at `address` to the system for later allocation. An ambiguous condition exists if `address` does not indicate a memory space that was previously allocated.

`free` calls the *MSVCRT* library function `free` with `address` as memblock.

`free (zero-terminated-string --)`

strongforth.sf

Return the memory space that has been allocated for `zero-terminated-string` to the system for later allocation. An ambiguous condition exists if `zero-terminated-string` has not been previously allocated.

`freopen (file zero-terminated-string zero-terminated-string -- file)`

Close `file` and reassign it to the file whose path is specified by the second `zero-terminated-string` in the mode specified by the first `zero-terminated-string`. `file` is equal to `file` if the operation succeeds, and null if the operation fails.

`freopen` calls the *MSVCRT* library function `freopen` with `file` as stream, the first `zero-terminated-string` as mode and the second `zero-terminated-string` as filename.

`fseek (seek-origin signed file -- integer)`

Move the file pointer associated with `file` to a new location that is signed bytes relative to the position specified by `seek-origin`. The position of the origin can be either the beginning of the file, the current file position or the end of the file. `integer` is zero if and only if the operation was successful.

`fseek` calls the *MSVCRT* library function `fseek` with `seek-origin` as origin, signed as offset and `file` as stream.

`fsetpos (address file -- integer)`

Restore the previous position of `file` from the information stored at `address`. The information has been stored at `address` by a previous use of `fgetpos`. `integer` is zero if and only if the operation was successful.

`fsetpos` calls the *MSVCRT* library function `fsetpos` with `address` as pos and `file` as stream.

`ftell (file -- unsigned)`

`unsigned` is the current position of `file`.

`ftell` calls the *MSVCRT* library function `ftell` with `file` as stream.

fwrite (file unsigned unsigned address -- 2nd)

Write multiple items stored at `address` to `file`. The first unsigned is the number items to be written. The second unsigned is the size in bytes of each item. Advance the file pointer associated with `file` by the number of bytes actually written. `2nd` is the number of items actually written, which may be less than the first unsigned if an error occurs.

`fwrite` calls the *MSVCRT* library function `fwrite` with `file` as `stream`, the first unsigned as `count`, the second unsigned as `size` and `address` as `buffer`.

getchar (-- character)

Read a single character from `stdin` and return it as `character`. Increment the associated file pointer.

`getchar` calls the *MSVCRT* library function `getchar`.

getenv (zero-terminated-string -- zero-terminated-string)

Search the list of local environment variables for the variable name in `zero-terminated-string`. The search is case insensitive. The returned `zero-terminated-string` is a copy of the variable entry in the environment, or null if the environment variable does not exist.

`getenv` calls the *MSVCRT* library function `getenv` with `zero-terminated-string` as `varname`.

gmtime (address -> signed -- address -> unsigned)

Convert a time represented as seconds elapsed since midnight (00:00:00), January 1, 1970, coordinated universal time (UTC), to an array of 9 unsigned single-cell values. `address -> signed` is a pointer to where the number of seconds is stored. `address -> unsigned` is the static address of the first array item, or null if `address -> signed` points to an invalid time. The structure of the array is as follows:

item #	description
0	seconds (0 - 59)
1	minutes (0 - 59)
2	hours (0 - 23)
3	day of month (1 - 31)
4	month (0 - 11; January = 0)
5	year minus 1900
6	day of week (0 - 6; Sunday = 0)
7	day of year (0 - 365; January 1 = 0)
8	0

`gmtime` calls the *MSVCRT* library function `gmtime` with `address -> signed` as `sourceTime`.

handle (stack-diagram -- 1st)

When used in a stack diagram, specifies an input or output parameter with data type `handle`.

heap-constant (stack-diagram -- 1st)

When used in a stack diagram, specifies an input or output parameter with data type heap-constant.

LC_ALL (-- locale-category)

msvcrt.sf

locale-category is the LC_ALL manifest constant of the *MSVCRT* library to be used as an input parameter to set-locale.

LC_COLLATE (-- locale-category)

msvcrt.sf

locale-category is the LC_COLLATE manifest constant of the *MSVCRT* library to be used as an input parameter to set-locale.

LC_CTYPE (-- locale-category)

msvcrt.sf

locale-category is the LC_CTYPE manifest constant of the *MSVCRT* library to be used as an input parameter to set-locale.

LC_MONETARY (-- locale-category)

msvcrt.sf

locale-category is the LC_MONETARY manifest constant of the *MSVCRT* library to be used as an input parameter to set-locale.

LC_NUMERIC (-- locale-category)

msvcrt.sf

locale-category is the LC_NUMERIC manifest constant of the *MSVCRT* library to be used as an input parameter to set-locale.

LC_TIME (-- locale-category)

msvcrt.sf

locale-category is the LC_TIME manifest constant of the *MSVCRT* library to be used as an input parameter to set-locale.

LK_LOCK (-- locking-mode)

msvcrt.sf

locking-mode is the LK_LOCK manifest constant of the *MSVCRT* library to be used as an input parameter to _locking.

LK_NBLCK (-- locking-mode)

msvcrt.sf

locking-mode is the LK_NBLCK manifest constant of the *MSVCRT* library to be used as an input parameter to _locking.

LK_NBRLOCK (-- locking-mode)

msvcrt.sf

locking-mode is the LK_NBRLOCK manifest constant of the *MSVCRT* library to be used as an input parameter to _locking.

LK_RLCK (-- locking-mode)

msvcrt.sf

locking-mode is the LK_RLCK manifest constant of the *MSVCRT* library to be used as an input parameter to `_locking`.

LK_UNLCK (-- locking-mode)

msvcrt.sf

locking-mode is the LK_UNLCK manifest constant of the *MSVCRT* library to be used as an input parameter to `_locking`.

locale-category (stack-diagram -- 1st)

When used in a stack diagram, specifies an input or output parameter with data type `locale-category`.

localeconv (-- address)

address is a pointer to a structure that contains information on locale settings.

`localeconv` calls the *MSVCRT* library function `localeconv`.

localtime (address -> signed -- address -> unsigned)

Convert a time represented as the number of seconds elapsed since midnight (00:00:00), January 1, 1970, of the local time zone, to an array of 9 unsigned single-cell values. `address -> signed` is a pointer to where the number of seconds is stored. `address -> unsigned` is the static address of the first array item, or null if `address -> signed` points to an invalid time. The structure of the array is as follows:

item #	Description
0	seconds (0 - 59)
1	minutes (0 - 59)
2	hours (0 - 23)
3	day of month (1 - 31)
4	month (0 - 11; January = 0)
5	year minus 1900
6	day of week (0 - 6; Sunday = 0)
7	day of year (0 - 365; January 1 = 0)
8	0

`localtime` calls the *MSVCRT* library function `localtime` with `address -> signed` as `sourceTime`.

locking-mode (stack-diagram -- 1st)

When used in a stack diagram, specifies an input or output parameter with data type `locking-mode`.

longjmp (integer address --)

Restores the stack environment and execution locale set by `setjmp`. `integer` is the value to be returned by `setjmp`. `address` is the address where `setjmp` stored the environment.

`longjmp` calls the *MSVCRT* library function `longjmp` with `integer` as value and `address` as `env`.

`malloc (unsigned -- address)`

Allocate a memory block of at least `unsigned` bytes. `address` is the address of the first byte of the allocated memory block, or null if the operation failed.

`malloc` calls the *MSVCRT* library function `malloc` with `unsigned` as `size`.

`mblen (unsigned address -> character -- signed)`

`signed` is the length in bytes (+1, +2 or +3) of the multibyte character `address -> character` points to, or zero if `address -> character` is null. If `address -> character` does not point to a valid multibyte character within the first `unsigned` characters, `signed` is equal to -1.

`mblen` calls the *MSVCRT* library function `mblen` with `unsigned` as `count` and `address -> character` as `mbstr`.

`mktime (address -> unsigned -- signed)`

Convert a time represented by an array of 9 unsigned single-cell values, into `signed`, the number of seconds elapsed since midnight (00:00:00), January 1, 1970, Coordinated Universal Time (UTC). `address -> unsigned` is the address of the first item of the array, whose structure is as follows:

item #	Description
0	seconds (0 - 59)
1	minutes (0 - 59)
2	hours (0 - 23)
3	day of month (1 - 31)
4	month (0 - 11; January = 0)
5	year minus 1900
6	day of week (0 - 6; Sunday = 0)
7	day of year (0 - 365; January 1 = 0)
8	0

`mktime` calls the *MSVCRT* library function `mktime` with `address -> unsigned` as `timeptr`.

`multibyte-codepage (stack-diagram -- 1st)`

When used in a stack diagram, specifies an input or output parameter with data type `multibyte-codepage`.

`multibyte-type (stack-diagram -- 1st)`

When used in a stack diagram, specifies an input or output parameter with data type `multibyte-type`.

O_APPEND (-- file-control) msvcrt.sf

file_control is the O_APPEND manifest constant of the *MSVCRT* library to be used as an input parameter to _open_osfhandle, _open and _sopen.

O_BINARY (-- file-control) msvcrt.sf

file_control is the O_BINARY manifest constant of the *MSVCRT* library to be used as an input parameter to _setmode, _open, _sopen and _pipe.

O_CREAT (-- file-control) msvcrt.sf

file_control is the O_CREAT manifest constant of the *MSVCRT* library to be used as an input parameter to _open and _sopen.

O_EXCL (-- file-control) msvcrt.sf

file_control is the O_EXCL manifest constant of the *MSVCRT* library to be used as an input parameter to _open and _sopen.

O_NOINHERIT (-- file-control) msvcrt.sf

file_control is the O_NOINHERIT manifest constant of the *MSVCRT* library to be used as an input parameter to _open, _sopen and _pipe.

O_RAW (-- file-control) msvcrt.sf

file_control is the O_RAW manifest constant of the *MSVCRT* library to be used as an input parameter to _setmode, _open and _sopen.

O_RDONLY (-- file-control) msvcrt.sf

file_control is the O_RDONLY manifest constant of the *MSVCRT* library to be used as an input parameter to _open_osfhandle, _open and _sopen.

O_RDWR (-- file-control) msvcrt.sf

file_control is the O_RDWR manifest constant of the *MSVCRT* library to be used as an input parameter to _open and _sopen.

O_TEXT (-- file-control) msvcrt.sf

file_control is the O_TEXT manifest constant of the *MSVCRT* library to be used as an input parameter to _open_osfhandle, _setmode, _open, _sopen and _pipe.

O_TRUNC (-- file-control) msvcrt.sf

file_control is the O_TRUNC manifest constant of the *MSVCRT* library to be used as an input parameter to _open and _sopen.

O_WRONLY (-- file-control) msvcrt.sf

`file_control` is the `O_WRONLY` manifest constant of the *MSVCRT* library to be used as an input parameter to `_open` and `_sopen`.

P_DETACH (-- process-mode) msvcrt.sf

`process_mode` is the `P_DETACH` manifest constant of the *MSVCRT* library to be used as an input parameter to `_spawnv`, `_spawnve`, `_spawnvp` and `_spawnvpe`.

P_NOWAIT (-- process-mode) msvcrt.sf

`process_mode` is the `P_NOWAIT` manifest constant of the *MSVCRT* library to be used as an input parameter to `_spawnv`, `_spawnve`, `_spawnvp` and `_spawnvpe`.

P_NOWAITO (-- process-mode) msvcrt.sf

`process_mode` is the `P_NOWAITO` manifest constant of the *MSVCRT* library to be used as an input parameter to `_spawnv`, `_spawnve`, `_spawnvp` and `_spawnvpe`.

P_OVERLAY (-- process-mode) msvcrt.sf

`process_mode` is the `P_OVERLAY` manifest constant of the *MSVCRT* library to be used as an input parameter to `_spawnv`, `_spawnve`, `_spawnvp` and `_spawnvpe`.

P_WAIT (-- process-mode) msvcrt.sf

`process_mode` is the `P_WAIT` manifest constant of the *MSVCRT* library to be used as an input parameter to `_spawnv`, `_spawnve`, `_spawnvp` and `_spawnvpe`.

perror (zero-terminated-string --)

Print error message `zero-terminated-string` to `stderr`.

`perror` calls the *MSVCRT* library function `perror` with `zero-terminated-string` as message.

process-mode (stack-diagram -- 1st)

When used in a stack diagram, specifies an input or output parameter with data type `process-mode`.

putchar (character -- 1st)

Write a single character `character` to `stdout` and increment the current position. If the operation is successful, `1st` is equal to `character`, otherwise `1st` is equal to `EOF`.

`putchar` calls the *MSVCRT* library function `putchar` with `character` as `c`.

puts (zero-terminated-string -- integer)

Type zero-terminated-string on the console and advance to the next line of text. If puts is successful, integer has a non-negative value, otherwise integer is equal to EOF.

puts calls the *MSVCRT* library function puts with zero-terminated-string as str.

qsort (address unsigned unsigned address --)

Sort an array using the quick-sort algorithm. The first address is a pointer to a callback function with two array arguments as parameters and a signed integer as return value. The first unsigned is the width of each element of the array in bytes. The second unsigned is the number of elements of the array. The second address is the address of the first element of the array. The array is overwritten.

The return value of the callback function is less than zero if the first argument is less than the second one, greater than zero if the first argument is greater than the second one, or equal to zero if both arguments are equal.

qsort calls the *MSVCRT* library function qsort with the first address as compare, the first unsigned as width, the second unsigned as num and the second address as base.

rand (-- unsigned)

unsigned is a pseudorandom number between 0 and 32767.

rand calls the *MSVCRT* library function rand.

realloc (unsigned address -- 2nd)

Change the size of a memory block that has been previously allocated at address. unsigned is the new size of the memory block. If address is null, realloc performs the semantics of malloc. If unsigned is zero, realloc performs the semantics of free. 2nd is the address of the first byte of the reallocated memory block. Note that 2nd may differ from address. An ambiguous condition exists if address is not the address of the first byte of a memory block allocated with calloc, malloc, or realloc.

realloc calls the *MSVCRT* library function realloc with unsigned as size and address as memblock.

remove (zero-terminated-string -- flag)

Delete the file with the path given by zero-terminated-string. flag is false if and only if the file was successfully deleted.

remove calls the *MSVCRT* library function remove with zero-terminated-string as path.

rename (zero-terminated-string zero-terminated-string -- integer)

Rename the file with the path given by the second zero-terminated-string to the name given by the first zero-terminated-string. integer is zero if and only if the file was successfully renamed.

rename calls the *MSVCRT* library function rename with the first zero-terminated-string as newname and the second zero-terminated-string as oldname.

rewind (file --)

Reposition the file pointer to the beginning of file.

rewind calls the *MSVCRT* library function rewind with file as stream.

S_IEXEC (-- file-status)

msvcrt.sf

file_status is the S_IEXEC manifest constant of the *MSVCRT* library.

S_IFCHR (-- file-status)

msvcrt.sf

file_status is the S_IFCHR manifest constant of the *MSVCRT* library.

S_IFDIR (-- file-status)

msvcrt.sf

file_status is the S_IFDIR manifest constant of the *MSVCRT* library.

S_IFMT (-- file-status)

msvcrt.sf

file_status is the S_IFMT manifest constant of the *MSVCRT* library.

S_IFREG (-- file-status)

msvcrt.sf

file_status is the S_IFREG manifest constant of the *MSVCRT* library.

S_IREAD (-- file-status)

msvcrt.sf

file_status is the S_IREAD manifest constant of the *MSVCRT* library to be used as an input parameter to _chmod, _umask, _open and _sopen.

S_IWRITE (-- file-status)

msvcrt.sf

file_status is the S_IWRITE manifest constant of the *MSVCRT* library to be used as an input parameter to _chmod, _umask, _open and _sopen.

SEEK_CUR (-- seek-origin)

msvcrt.sf

seek-origin is the SEEK_CUR manifest constant of the *MSVCRT* library to be used as an input parameter to _lseek, _lseeki64 and fseek.

SEEK_END (-- seek-origin)

msvcrt.sf

seek-origin is the SEEK_END manifest constant of the *MSVCRT* library to be used as an input parameter to _lseek, _lseeki64 and fseek.

SEEK_SET (-- seek-origin)

msvcrt.sf

seek-origin is the SEEK_SET manifest constant of the *MSVCRT* library to be used as an input parameter to `_lseek`, `_lseeki64` and `fseek`.

seek-origin (stack-diagram -- 1st)

When used in a stack diagram, specifies an input or output parameter with data type seek-origin.

setjmp (address -- integer)

Save the stack environment at address. The stack environment can later be restored by `longjmp`, returning to the location immediately following the call to `setjmp`. integer is zero after saving the stack environment. When returning from `longjmp`, integer has the value of the argument passed to `longjmp`, or 1 if this value is 0.

`setjmp` calls the *MSVCRT* library function `setjmp` with address as env.

setlocale (zero-terminated-string locale-category -- zero-terminated-string)

Set, change, or query some or all of the current programs locale information specified by zero-terminated-string and locale-category. zero-terminated-string contains the changed or unchanged locale information.

`setlocale` calls the *MSVCRT* library function `setlocale` with zero-terminated-string as locale and locale-category as category.

setvbuf (unsigned buffer-mode address -> character file -- flag)

Control buffering and buffer size for file. file must be open and must not have undergone an I/O operation since it was opened. address -> character is used as the buffer with unsigned bytes, unless it is null. If address -> character is null, `setvbuf` allocates a buffer. buffer-mode specifies whether a full buffer, a line buffer or no buffer at all shall be used. flag is false if and only if the operation was successful.

`setvbuf` calls the *MSVCRT* library function `setvbuf` with unsigned as size, buffer-mode as mode, address -> character as buffer and file as stream.

SH_COMPAT (-- sharing-mode)

msvcrt.sf

sharing-mode is the SH_COMPAT manifest constant of the *MSVCRT* library to be used as an input parameter to `_fsopen`.

SH_DENYNO (-- sharing-mode)

msvcrt.sf

sharing-mode is the SH_DENYNO manifest constant of the *MSVCRT* library to be used as an input parameter to `_sopen` and `_fsopen`.

SH_DENYRD (-- sharing-mode)

msvcrt.sf

sharing-mode is the SH_DENYRD manifest constant of the *MSVCRT* library to be used as an input parameter to `_sopen` and `_fsopen`.

SH_DENYRW (-- sharing-mode)

msvcrt.sf

sharing-mode is the SH_DENYRW manifest constant of the *MSVCRT* library to be used as an input parameter to `_sopen` and `_fsopen`.

SH_DENYWR (-- sharing-mode)

msvcrt.sf

sharing-mode is the SH_DENYWR manifest constant of the *MSVCRT* library to be used as an input parameter to `_sopen` and `_fsopen`.

sharing-mode (stack-diagram -- 1st)

When used in a stack diagram, specifies an input or output parameter with data type sharing-mode.

srand (unsigned --)

Make unsigned the seed value for the pseudorandom number generator used by `rand`.

`srand` calls the *MSVCRT* library function `srand` with unsigned as seed.

strerror (integer -- zero-terminated-string)

zero-terminated-string is a system error message associated with the error code integer.

`strerror` calls the *MSVCRT* library function `strerror` with integer as `errno`.

**strftime (address -> unsigned zero-terminated-string unsigned
caddress -> character -- unsigned)**

Format a time value specified by the 9 cells array `address -> unsigned` according to the format specified by `zero-terminated-string`. Store at most unsigned characters of the formatted time string at `caddress -> character`. Output parameter unsigned is the character count of the formatted time string, or zero if input parameter unsigned is less than the length of the formatted time string.

`address -> unsigned` is the address of the first item of the array, whose structure is as follows:

item #	Description
0	seconds (0 - 59)
1	minutes (0 - 59)
2	hours (0 - 23)
3	day of month (1 - 31)
4	month (0 - 11; January = 0)
5	year minus 1900
6	day of week (0 - 6; Sunday = 0)
7	day of year (0 - 365; January 1 = 0)

`strftime` calls the *MSVCRT* library function `strftime` with `address -> unsigned` as `timeptr`, `zero-terminated-string` as `format`, `unsigned` as `maxsize` and `caddress -> character` as `strDest`.

`string (zero-terminated-string -- caddress -> character unsigned)`

strongforth.sf

`caddress -> character` is the address of the first character of `zero-terminated-string`. `unsigned` is the length of `zero-terminated-string`.

`strlen (zero-terminated-string -- unsigned)`

`unsigned` is the length of `zero-terminated-string`.

`strlen` calls the *MSVCRT* library function `strlen` with `zero-terminated-string` as `str`.

`system (zero-terminated-string -- integer)`

Pass `zero-terminated-string` to the command interpreter, which executes the string as an operating-system command. `system` uses the `COMSPEC` and `PATH` environment variables to locate the command-interpreter file `CMD.exe`. `integer` is the status code returned by the command.

`system` calls the *MSVCRT* library function `system` with `zero-terminated-string` as `command`.

`time (address -> signed -- 2nd)`

`2nd` is the number of seconds elapsed since midnight (00:00:00), January 1, 1970, Coordinated Universal Time (UTC), according to the system clock. Additionally, store this value at `address -> signed`.

`time` calls the *MSVCRT* library function `time` with `address -> signed` as `destTime`.

`tmpfile (-- file)`

`file` is a temporary file created in the current working directory, or null if an error occurred. The temporary file is automatically deleted when `file` is closed, when StrongForth terminates normally, or when executing `_rmtmp`, assuming that the current working directory does not change. The temporary file is opened in binary read/write mode.

`tmpfile` calls the *MSVCRT* library function `tmpfile`.

`tmpnam (caddress -> character -- zero-terminated-string)`

Store the zero-terminated name of a file that does not currently exist at `caddress -> character`, or in an internal static buffer if `caddress -> character` is null. `zero-terminated-string` is the generated filename, or null if an error occurred.

`tmpnam` calls the *MSVCRT* library function `tmpnam` with `caddress -> character` as `str`.

ungetc (file character -- 2nd)

Pushes character back to file, assuming that character was the most recent character read from file. Clear the end-of-file indicator. If ungetc is successful, 2nd is character, otherwise 2nd is EOF. An attempt to push EOF onto file is ignored.

ungetc calls the *MSVCRT* library function ungetc with file as stream and character as c.

WAIT_CHILD (-- cwait-action)

msvcrt.sf

cwait_action is the WAIT_CHILD manifest constant of the *MSVCRT* library to be used as an input parameter to _cwait.

WAIT_GRANDCHILD (-- cwait-action)

msvcrt.sf

cwait_action is the WAIT_GRANDCHILD manifest constant of the *MSVCRT* library to be used as an input parameter to _cwait.

**zero-terminated-string (caddress -> character unsigned --
zero-terminated-string)**

Allocate unsigned plus 1 characters of dynamic memory space. Copy unsigned characters starting at caddress -> character and a terminating null character to the allocated memory. zero-terminated-string is the address of the first character in dynamic memory.

zero-terminated-string (stack-diagram -- 1st)

When used in a stack diagram, specifies an input or output parameter with data type zero-terminated-string.

_access (logical zero-terminated-string -- flag)

flag is false if and only if the file with the path zero-terminated-string exists and has the access mode specified by logical. If bit 1 is set in logical, _access checks for write access. If bit 2 is set in logical, _access checks for read access. If both bit 1 and bit 2 are set in logical, _access checks for read and write access. If none of these bits is set in logical, _access checks only if the file exists.

_access calls the *MSVCRT* library function _access with logical as mode and zero-terminated-string as path.

_beginthread (address unsigned address -- handle)

Create a thread that begins execution of a code definition at the second address. The code definition may have no output parameters. The thread is terminated automatically when the code definition returns. The first address points to the input parameters of the code definition. It should be null if the code definition has no input parameters. unsigned is the stack size of the new thread. handle is the operating system handle of the new thread, or -1 if the operation fails.

`_beginthread` calls the *MSVCRT* library function `_beginthread` with the first address as `arglist`, unsigned as `stack_size` and the second address as `start_address`.

**`_beginthreadex (address -> unsigned unsigned address address
unsigned address -- handle)`**

Create a thread that begins execution of a code definition at the second address. The code definition may have no output parameters. The thread is terminated automatically when the code definition returns. The first address points to the input parameters of the code definition. It should be null if the code definition has no input parameters. The first unsigned controls the initial state of the new thread. The second unsigned is the stack size of the new thread. `address -> unsigned` is the address of a cell that receives the thread identifier. `handle` is the operating system handle of the new thread, or zero if the operation fails. The third address is a pointer to a structure that determines whether `handle` can be inherited by child processes.

`_beginthreadex` calls the *MSVCRT* library function `_beginthreadex` with `address ->` unsigned as `thrddadr`, the first unsigned as `initflag`, the first address as `arglist`, the second address as `start_address`, the second unsigned as `stack_size` and the third address as `security`.

`_c_exit (--)`

Close all open streams. Return to the caller of StrongForth without terminating the StrongForth process.

`_c_exit` calls the *MSVCRT* library function `_c_exit`.

`_cexit (--)`

Call, in reverse order, the functions registered by `atexit` and `_onexit`. Flush all I/O buffers and close all open streams. Return to the caller of StrongForth without terminating the StrongForth process.

`_cexit` calls the *MSVCRT* library function `_cexit`.

`_chdir (zero-terminated-string -- flag)`

Changes the current working directory to the absolute or relative path `zero-terminated-string`. `flag` is false if the operation succeeds, otherwise `flag` is true.

`_chdir` calls the *MSVCRT* library function `_chdir` with `zero-terminated-string` as `dirname`.

`_chdrive (unsigned -- flag)`

Changes the current disk drive to the drive selected by unsigned. 1 selects drive A:, 2 selects drive B:, ... 26 selects drive Z:. `flag` is false if the operation succeeds, otherwise `flag` is true.

`_chdrive` calls the *MSVCRT* library function `_chdrive` with unsigned as `drive`.

`_chmod (file-status zero-terminated-string -- flag)`

Changes the read/write permission setting of the file with the name zero-terminated-string to file-status.

`_chmod` calls the *MSVCRT* library function `_chmod` with file-status as pmode and zero-terminated-string as filename.

`_chsize (unsigned file-descriptor -- flag)`

Changes the size of the file associated with file-descriptor to unsigned. flag is false if and only if the operation succeeded.

`_chsize` calls the *MSVCRT* library function `_chsize` with unsigned as size and file-descriptor as fd.

`_close (file-descriptor -- flag)`

Close the file associated with file-descriptor. flag is false if and only if the operation is successful.

`_close` calls the *MSVCRT* library function `_close` with file-descriptor as fd.

`_commit (file-descriptor -- flag)`

Flush the file associated with file-descriptor to the mass storage medium. flag is false if and only if the operation succeeds.

`_commit` calls the *MSVCRT* library function `_commit` with file-descriptor as fd.

`_cputs (zero-terminated-string -- integer)`

Type zero-terminated-string on the console. integer is zero if and only if the operation is successful.

`_cputs` calls the *MSVCRT* library function `_cputs` with zero-terminated-string as str.

`_ctime64 (address -> signed-double -- zero-terminated-string)`

Convert a time represented as seconds elapsed since midnight (00:00:00), January 1, 1970, of the local time zone, into zero-terminated-string. address -> signed-double is a pointer to where the number of seconds is stored. zero-terminated-string has the fixed format

DOW MMM DD hh:mm:ss YYYY\n\0

with DOW as the day of week in three letters, MMM as the month in three letters, DD as the day of month, hh as hours, mm as minutes, ss as seconds and YYYY as the year.

`_ctime64` calls the *MSVCRT* library function `_ctime64` with address -> signed-double as sourceTime.

`_cwait (cwait-action handle address -> integer -- 2nd)`

Wait for the termination of the process specified by handle. handle should be the value returned by `_spawn`. address -> integer points to a buffer where the return structure of

the process specified by `handle` is to be stored. If `address -> integer` is null, the return structure will not be stored. `cwait-action` is ignored by the operating system.

2nd is the actual handle of the terminated process, or -1 if the operation failed. 2nd should normally not be used, because the process associated with it has been terminated.

`_cwait` calls the *MSVCRT* library function `_cwait` with `cwait-action` as `action`, `handle` as `procHandle` and `address -> integer` as `termstat`.

`_dup (file-descriptor -- 1st)`

Creates a second file descriptor `1st` for the open file associated with `file-descriptor` to the mass storage medium. `1st` is equal to -1 if the operation fails.

`_dup` calls the *MSVCRT* library function `_dup` with `file-descriptor` as `fd`.

`_dup2 (file-descriptor 1st -- flag)`

Forces `file-descriptor` to refer to the same file as `1st`. If `file-descriptor` is associated with an open file at the time of the call, that file is closed. `flag` is false if and only if the operation succeeds.

`_dup2` calls the *MSVCRT* library function `_dup2` with `file-descriptor` as `fd2` and `1st` as `fd1`.

`_endthread (--)`

Terminate a thread that has been created by `_beginthread` and close the thread handle.

`_endthread` calls the *MSVCRT* library function `_endthread`.

`_endthreadex (unsigned --)`

Terminate a thread that has been created by `_beginthreadex`. `unsigned` is the thread exit code.

`_endthreadex` calls the *MSVCRT* library function `_endthreadex`.

`_eof (file-descriptor -- signed)`

`signed` is 1 if the current position of the file associated with `file-descriptor` is the end of the file, -1 if an error occurred during execution, and 0 in all other cases.

`_eof` calls the *MSVCRT* library function `_eof` with `file-descriptor` as `fd`.

**`_execv (address -> zero-terminated-string zero-terminated-string
-- handle)`**

Load and execute a new child process. `zero-terminated-string` is the filename of the executable. `address -> zero-terminated-string` points to the command-line arguments. `handle` is equal to -1 if the input parameters are invalid. Otherwise, `_execv` does not return.

`_execv` calls the *MSVCRT* library function `_execv` with `address -> zero-terminated-string` as `argv` and `zero-terminated-string` as `cmdname`.

**`_execve (address -> zero-terminated-string address ->
zero-terminated-string zero-terminated-string -- handle)`**

Load and execute a new child process. zero-terminated-string is the filename of the executable. The first address -> zero-terminated-string points to the environment settings. The second address -> zero-terminated-string points to the command-line arguments. handle is equal to -1 if the input parameters are invalid. Otherwise, _execve does not return.

_execve calls the *MSVCRT* library function _execve with the first address -> zero-terminated-string as envp, the second address -> zero-terminated-string as argv and zero-terminated-string as cmdname.

**`_execvp (address -> zero-terminated-string zero-terminated-string
-- handle)`**

Load and execute a new child process. zero-terminated-string is the path of the executable. address -> zero-terminated-string points to the command-line arguments. handle is equal to -1 if the input parameters are invalid. Otherwise, _execvp does not return.

_execvp calls the *MSVCRT* library function _execvp with address -> zero-terminated-string as argv and zero-terminated-string as cmdname.

**`_execvpe (address -> zero-terminated-string address ->
zero-terminated-string zero-terminated-string -- handle)`**

Load and execute a new child process. zero-terminated-string is the path of the executable. The first address -> zero-terminated-string points to the environment settings. The second address -> zero-terminated-string points to the command-line arguments. handle is equal to -1 if the input parameters are invalid. Otherwise, _execvpe does not return.

_execvpe calls the *MSVCRT* library function _execvpe with the first address -> zero-terminated-string as envp, the second address -> zero-terminated-string as argv and zero-terminated-string as cmdname.

`_exit (integer --)`

Terminate StrongForth and return to the caller of StrongForth. integer is made available to the caller as a status value.

_exit calls the *MSVCRT* library function _exit with integer as status.

`_expand (unsigned address -- 2nd)`

Changes the size of a previously allocated memory block starting at address to unsigned bytes by trying to expand or shrink the block without moving it. The contents of the block remain unchanged. 2nd is equal to address if the operation succeeds, or null if it fails.

_expand calls the *MSVCRT* library function _expand with unsigned as size and address as memblock.

`_fcloseall (-- unsigned)`

Flush and close all open streams except stdin, stdout and stderr. unsigned is the total number of streams closed, or -1 if the operation fails.

`_fcloseall` calls the *MSVCRT* library function `_fcloseall`.

`_fdopen (zero-terminated-string file-descriptor -- file)`

Associate file with file-descriptor, allowing a file that is opened for low-level I/O to be buffered and formatted. zero-terminated-string specifies the access mode. file is null if the operation fails.

`_fdopen` calls the *MSVCRT* library function `_fdopen` with zero-terminated-string as mode and file-descriptor as fd.

`_filelength (file-descriptor -- unsigned)`

unsigned is the length of the file associated with file-descriptor. An ambiguous condition exists if the file length is greater than max-unsigned.

`_filelength` calls the *MSVCRT* library function `_filelength` with file-descriptor as fd.

`_filelengthi64 (file-descriptor -- unsigned-double)`

unsigned-double is the length of the file associated with file-descriptor.

`_filelengthi64` calls the *MSVCRT* library function `_filelengthi64` with file-descriptor as fd.

`_fileno (file -- file-descriptor)`

file-descriptor is the file descriptor associated with the stream file.

`_fileno` calls the *MSVCRT* library function `_fileno` with file as stream.

`_findclose (handle -- flag)`

Closes the search handle handle and releases associated resources. handle was previously provided by `_findfirst`, `_findfirst64` or `_findfirsti64`. flag is false if and only if the operation succeeded.

`_findclose` calls the *MSVCRT* library function `_findclose` with handle as handle.

`_findfirst (address zero-terminated-string -- handle)`

handle is a unique search handle identifying the file or group of files that match the file specification in zero-terminated-string, or -1 if no matching files were found or an error occurs. The file specification is a path that may contain wildcard characters. handle can be used in a subsequent call to `_findnext` or to `_findclose`. address is a pointer to a structure which `_findfirst` fills with information about the first instance of a file name that matches the file specification.

`_findfirst` calls the *MSVCRT* library function `_findfirst` with `address` as `fileinfo` and zero-terminated-string as `filespec`.

`_findfirst64 (address zero-terminated-string -- handle)`

`handle` is a unique search handle identifying the file or group of files that match the file specification in `zero-terminated-string`, or -1 if no matching files were found or an error occurs. The file specification is a path that may contain wildcard characters. `handle` can be used in a subsequent call to `_findnext64` or to `_findclose`. `address` is a pointer to a structure which `_findfirst64` fills with information about the first instance of a file name that matches the file specification.

`_findfirst64` calls the *MSVCRT* library function `_findfirst64` with `address` as `fileinfo` and zero-terminated-string as `filespec`.

`_findfirsti64 (address zero-terminated-string -- handle)`

`handle` is a unique search handle identifying the file or group of files that match the file specification in `zero-terminated-string`, or -1 if no matching files were found or an error occurs. The file specification is a path that may contain wildcard characters. `handle` can be used in a subsequent call to `_findnexti64` or to `_findclose`. `address` is a pointer to a structure which `_findfirsti64` fills with information about the first instance of a file name that matches the file specification.

`_findfirsti64` calls the *MSVCRT* library function `_findfirsti64` with `address` as `fileinfo` and zero-terminated-string as `filespec`.

`_findnext (address handle -- flag)`

Fill the structure pointed to by `address` with information about the next instance of a file name that matches the unique search handle `handle`. `handle` was previously provided by `_findfirst`. `flag` is false if and only if a matching file was found and no error occurred.

`_findnext` calls the *MSVCRT* library function `_findnext` with `address` as `fileinfo` and `handle` as `handle`.

`_findnext64 (address handle -- flag)`

Fill the structure pointed to by `address` with information about the next instance of a file name that matches the unique search handle `handle`. `handle` was previously provided by `_findfirst64`. `flag` is false if and only if a matching file was found and no error occurred.

`_findnext64` calls the *MSVCRT* library function `_findnext64` with `address` as `fileinfo` and `handle` as `handle`.

`_findnexti64 (address handle -- flag)`

Fill the structure pointed to by `address` with information about the next instance of a file name that matches the unique search handle `handle`. `handle` was previously provided by `_findfirsti64`. `flag` is false if and only if a matching file was found and no error occurred.

`_findnexti64` calls the *MSVCRT* library function `_findnexti64` with address as `fileinfo` and handle as `handle`.

`_flushall (-- unsigned)`

Writes the contents of all buffers associated with open output streams to mass storage. Clear the contents of all buffers associated with open input streams. `unsigned` is the total number of open streams.

`_flushall` calls the *MSVCRT* library function `_flushall`.

`_FREEENTRY (-- heap-constant)`

msvcrt.sf

`heap-constant` is the `_FREEENTRY` manifest constant of the *MSVCRT* library.

**`_fsopen (sharing-mode zero-terminated-string
zero-terminated-string -- file)`**

Open the file whose path is specified by the second `zero-terminated-string` in the mode specified by the first `zero-terminated-string`, and return its file handle `file`. Prepare the file for subsequent shared reading or writing as defined by `sharing-mode`. `file` is null if the operation fails.

`_fsopen` calls the *MSVCRT* library function `_fsopen` with `sharing-mode` as `shflag`, the first `zero-terminated-string` as `mode` and the second `zero-terminated-string` as `filename`.

`_fstat (address file-descriptor -- flag)`

Stores information about the file associated with `file-descriptor` in a structure located at `address`.

`_fstat` calls the *MSVCRT* library function `_fstat` with `address` as `buffer` and `file-descriptor` as `fd`.

`_fstat64 (address file-descriptor -- flag)`

Stores information about the file associated with `file-descriptor` in a structure located at `address`.

`_fstat64` calls the *MSVCRT* library function `_fstat64` with `address` as `buffer` and `file-descriptor` as `fd`.

`_fstati64 (address file-descriptor -- flag)`

Stores information about the file associated with `file-descriptor` in a structure located at `address`.

`_fstati64` calls the *MSVCRT* library function `_fstati64` with `address` as `buffer` and `file-descriptor` as `fd`.

`_ftime (address --)`

Store the current local time in the structure starting at address.

`_ftime` calls the *MSVCRT* library function `_ftime` with address as `timeptr`.

`_ftime64 (address --)`

Store the current local time in the structure starting at address.

`_ftime64` calls the *MSVCRT* library function `_ftime64` with address as `timeptr`.

**`_fullpath (unsigned zero-terminated-string address -> character
-- zero-terminated-string)`**

Fill the buffer starting at `caddress -> character` with up to unsigned characters of the absolute path name belonging to the relative path name provided in `zero-terminated-string`. `zero-terminated-string` is the absolute path name starting at `address -> character`. If `address -> character` is null, a buffer is allocated using `malloc`. If the relative pathname is invalid or if the absolute pathname is longer than unsigned characters, `zero-terminated-string` is null.

`_fullpath` calls the *MSVCRT* library function `_fullpath` with unsigned as `maxLength`, `zero-terminated-string` as `relpath` and `address -> character` as `abspath`.

`_futime (address -> signed file-descriptor -- flag)`

Sets the access time and the modification date on the open file associated with `file-descriptor`. The two time values are expected in the given order at `address -> signed` as the number of seconds elapsed since midnight (00:00:00), January 1, 1970, Coordinated Universal Time (UTC). `flag` is false if and only if the operation succeeds.

`_futime` calls the *MSVCRT* library function `_futime` with `address -> signed` as `filetime` and `file-descriptor` as `fd`.

`_futime64 (address -> signed-double file-descriptor -- flag)`

Sets the access time and the modification date on the open file associated with `file-descriptor`. The two time values are expected in the given order at `address -> signed-double` as the number of seconds elapsed since midnight (00:00:00), January 1, 1970, Coordinated Universal Time (UTC). `flag` is false if and only if the operation succeeds.

`_futime64` calls the *MSVCRT* library function `_futime64` with `address -> signed-double` as `filetime` and `file-descriptor` as `fd`.

`_get_heap_handle (-- handle)`

`handle` the handle to the Win32 heap used by the *MSVCRT* library.

`_get_heap_handle` calls the *MSVCRT* library function `_get_heap_handle`.

`_get_osfhandle (file-descriptor -- file)`

`file` is the operating system file handle associated with the *MSVCRT* file descriptor `file-descriptor`.

`_get_osfhandle` calls the *MSVCRT* library function `_get_osfhandle` with file-descriptor as `fd`.

`_getch (-- character)`

Accept a single character from the console and return it as `character`. Do not type an echo.

`_getch` calls the *MSVCRT* library function `_getch`.

`_getche (-- character)`

Accept a single character from the console and return it as `character`. Type an echo of the character on the console.

`_getche` calls the *MSVCRT* library function `_getche`.

`_getcwd (unsigned caddress -> character -- zero-terminated-string)`

Copy up to unsigned minus 1 characters of the current working directory path to the memory space starting at `caddress -> character` and append a terminating null character. Return the resulting string as `zero-terminated-string`. If `caddress -> character` is null, a buffer with unsigned characters is allocated using `malloc`.

`zero-terminated-string` is null if a buffer cannot be allocated, if the path is longer than unsigned characters, or if unsigned is zero or greater than `max-signed`.

`_getcwd` calls the *MSVCRT* library function `_getcwd` with unsigned as `maxlen` and `caddress -> character` as `buffer`.

`_getdcwd (unsigned caddress -> character unsigned -- zero-terminated-string)`

Copy up to unsigned minus 1 characters (the first unsigned) of the current working directory path on the drive specified by the second unsigned to the memory space starting at `caddress -> character` and append a terminating null character. The second unsigned indicates the current disk drive. 0 selects the default drive, 1 selects drive A:, 2 selects drive B:, ... 26 selects drive Z:. Return the resulting string as `zero-terminated-string`. If `caddress -> character` is null, a buffer with unsigned characters is allocated using `malloc`.

`zero-terminated-string` is null if a buffer cannot be allocated, if the path is longer than unsigned characters (the first unsigned), or if the first unsigned is zero or greater than `max-signed`.

`_getdcwd` calls the *MSVCRT* library function `_getdcwd` with the first unsigned as `maxlen`, `caddress -> character` as `buffer` and the second unsigned as `drive`.

`_getdiskfree (address -> unsigned unsigned -- unsigned)`

Fills four cells starting at `address -> unsigned` with information about the disk drive with the index `unsigned`. 0 as `unsigned` selects the default drive, 1 selects drive A:, 2 selects drive B:, ... 26 selects drive Z:.

The first cell contains the total number of clusters on the disk drive. The second cell contains the number of unused clusters on the disk drive. The third cell contains the number of sectors in each cluster. The fourth cell contains the size of each sector in bytes.

`_getdiskfree` calls the *MSVCRT* library function `_getdiskfree` with `address -> unsigned` as `driveinfo` and with `unsigned` as `drive`.

`_getdrive (-- unsigned)`

`unsigned` indicates the current disk drive. 1 indicates drive A:, 2 indicates drive B:, ... 26 indicates drive Z:.

`_getdrive` calls the *MSVCRT* library function `_getdrive`.

`_getdrives (-- logical)`

`logical` is a bit mask in which each bit represents the availability of a disk drive. Bit 0 is set if and only if drive A: is available, bit 1 is set if and only if drive B: is available, and so on. Bit 26 is set if and only if drive Z: is available. If `_getdrives` fails, `logical` is zero.

`_getdrives` calls the *MSVCRT* library function `_getdrives`.

`_getmaxstdio (-- unsigned)`

`unsigned` is the number of simultaneously open files permitted at the stream I/O level.

`_getmaxstdio` calls the *MSVCRT* library function `_getmaxstdio`.

`_getmbcp (-- multibyte-codepage)`

`multibyte-codepage` is the current multibyte code page, or zero if a single byte code page is in use.

`_getmbcp` calls the *MSVCRT* library function `_getmbcp`.

`_getpid (-- integer)`

`integer` is the process ID obtained from the system.

`_getpid` calls the *MSVCRT* library function `_getpid`.

`_getw (file -- single)`

Read the binary value of a single cell from `file` and return it as `single`. Advance the current position of `file`. `single` is equal to -1 if the operation fails.

`_getw` calls the *MSVCRT* library function `_getw` with `file` as `stream`.

`_gmtime64 (address -> signed-double -- address -> unsigned)`

Convert a time represented as the number of seconds elapsed since midnight (00:00:00), January 1, 1970, coordinated universal time (UTC), to an array of 9 unsigned single-cell values. `address -> signed-double` is a pointer to where the number of seconds is stored. `address ->`

unsigned is the static address of the first array item, or null if address ->
signed-double points to an invalid time. The structure of the array is as follows:

item #	Description
0	seconds (0 - 59)
1	minutes (0 - 59)
2	hours (0 - 23)
3	day of month (1 - 31)
4	month (0 - 11; January = 0)
5	year minus 1900
6	day of week (0 - 6; Sunday = 0)
7	day of year (0 - 365; January 1 = 0)
8	0

_gmtime64 calls the *MSVCRT* library function _gmtime64 with address ->
signed-double as sourceTime.

_HEAPBADBEGIN (-- heap-constant)

msvcrt.sf

heap-constant is the _HEAPBADBEGIN manifest constant of the *MSVCRT* library to be returned by _heapchk and _heapwalk.

_HEAPBADNODE (-- heap-constant)

msvcrt.sf

heap-constant is the _HEAPBADNODE manifest constant of the *MSVCRT* library to be returned by _heapchk and _heapwalk.

_HEAPBADPTR (-- heap-constant)

msvcrt.sf

heap-constant is the _HEAPBADPTR manifest constant of the *MSVCRT* library to be returned by _heapchk and _heapwalk.

_heapchk (-- heap-constant)

heap-constant indicates whether the heap is consistent.

_heapchk calls the *MSVCRT* library function _heapchk.

_HEAPEMPTY (-- heap-constant)

msvcrt.sf

heap-constant is the _HEAPEMPTY manifest constant of the *MSVCRT* library to be returned by _heapchk and _heapwalk.

_HEAPEND (-- heap-constant)

msvcrt.sf

heap-constant is the _HEAPEND manifest constant of the *MSVCRT* library to be returned by _heapwalk.

_heapmin (-- flag)

Minimize the heap by releasing unused heap memory to the operating system. `flag` is false if and only if the operation succeeds.

`_heapmin` calls the *MSVCRT* library function `_heapmin`.

`_HEAPOK (-- heap-constant)`

msvcrt.sf

`heap-constant` is the `_HEAPOK` manifest constant of the *MSVCRT* library to be returned by `_heapchk` and `_heapwalk`.

`_heapwalk (address -> single - heap-constant)`

Walks through the heap from one entry to the next, updating the heap information structure at `address -> single` with each call. The heap information structure consists of three cells. If the first cell is null before `_heapinfo` is executed, `_heapinfo` stores the heap information structure of the first heap entry at `address -> single`. `heap-constant` indicates whether the heap is consistent and whether the end of the heap has been exceeded.

`_heapwalk` calls the *MSVCRT* library function `_heapwalk` with `address -> single` as `entryinfo`.

`_iob (-- file)`

`file` is the first element in an array of 20 stdio control structures.

`_IOFBF (-- buffer-mode)`

msvcrt.sf

`buffer-mode` is the `_IOFBF` manifest constant of the *MSVCRT* library to be used as an input parameter to `setvbuf`.

`_IOLBF (-- buffer-mode)`

msvcrt.sf

`buffer-mode` is the `_IOLBF` manifest constant of the *MSVCRT* library to be used as an input parameter to `setvbuf`.

`_IONBF (-- buffer-mode)`

msvcrt.sf

`buffer-mode` is the `_IONBF` manifest constant of the *MSVCRT* library to be used as an input parameter to `setvbuf`.

`_isatty (file-descriptor -- integer)`

`integer` is nonzero if `file-descriptor` is a character device. Otherwise, `integer` is zero.

`_isatty` calls the *MSVCRT* library function `_isatty` with `file-descriptor` as `fd`.

`_kbhit (-- integer)`

`integer` is non-zero if a key has been hit on the console. Otherwise, `integer` is zero.

`_kbhit` calls the *MSVCRT* library function `_kbhit`.

```
_lfind ( address unsigned address -> unsigned address address -- 5  
th )
```

Perform a linear search of an array. The first address is a pointer to a callback function with two array arguments as parameters and an integer as return value. unsigned is the width of each element of the array in bytes. address -> unsigned is the address of a variable containing the number of elements of the array. The second address is the address of the first element of the array. The second address is pointer to the key element to search for.

The return value of the callback function is equal to zero if both arguments are equal, and non-zero otherwise.

5 th is the address of an array element that matches the one the third address points to, or null if there is no match.

_lfind calls the *MSVCRT* library function `_lfind` with the first address as compare, unsigned as width, address -> unsigned as num, the second address as base and the third address as key.

```
_localtime64 ( address -> signed-double -- address -> unsigned )
```

Convert a time represented as seconds elapsed since midnight (00:00:00), January 1, 1970, of the local time zone, to an array of 9 unsigned single-cell values. address -> signed-double is a pointer to where the number of seconds is stored. address -> unsigned is the static address of the first array item, or null if address -> signed-double points to an invalid time. The structure of the array is as follows:

item #	Description
0	seconds (0 - 59)
1	minutes (0 - 59)
2	hours (0 - 23)
3	day of month (1 - 31)
4	month (0 - 11; January = 0)
5	year minus 1900
6	day of week (0 - 6; Sunday = 0)
7	day of year (0 - 365; January 1 = 0)
8	0

_localtime64 calls the *MSVCRT* library function `_localtime64` with address -> signed-double as sourceTime.

```
_locking ( unsigned locking-mode file-descriptor -- flag )
```

Locks or unlocks unsigned bytes of the file associated with file-descriptor, beginning at the current position of the file. locking-mode determines whether the sequence of bytes shall be locked or unlocked for read and/or write access.

_locking calls the *MSVCRT* library function `_locking` with unsigned as nbytes, locking-mode as mode and file-descriptor as fd.

```
_lsearch ( address unsigned address -> unsigned address address --  
5 th )
```

Perform a linear search of an array. The first address is a pointer to a callback function with two array arguments as parameters and an integer as return value. unsigned is the width of each

element of the array in bytes. `address -> unsigned` is the address of a variable containing the number of elements of the array. The second address is the address of the first element of the array. The third address is pointer to the key element to search for.

The return value of the callback function is equal to zero if both arguments are equal, and non-zero otherwise.

If an array element that matches the one the third address points to is found, 5th is the address of this array element. If there is no match, append the element the third address points to to the array and increment the number stored at `address -> unsigned`. 5th is the address of the new element.

`_lsearch` calls the *MSVCRT* library function `_lsearch` with the first address as `compare`, `unsigned` as `width`, `address -> unsigned` as `num`, the second address as `base` and the third address as `key`.

`_lseek (seek-origin signed file-descriptor -- unsigned)`

Move the file pointer of the file associated with `file-descriptor` to a new location `signed` bytes relative to the origin. `seek-origin` specifies whether the origin is the beginning of the file, the current position of the file pointer or the end of the file. `unsigned` is the new position of the file pointer, or -1 if the operation fails.

`_lseek` calls the *MSVCRT* library function `_lseek` with `seek-origin` as `origin`, `signed` as `offset` and `file-descriptor` as `fd`.

`_lseeki64 (seek-origin signed-double file-descriptor -- unsigned-double)`

Move the file pointer of the file associated with `file-descriptor` to a new location `signed-double` bytes relative to the origin. `seek-origin` specifies whether the origin is the beginning of the file, the current position of the file pointer or the end of the file. `unsigned-double` is the new position of the file pointer, or -1 if the operation fails.

`_lseeki64` calls the *MSVCRT* library function `_lseeki64` with `seek-origin` as `origin`, `signed-double` as `offset` and `file-descriptor` as `fd`.

`_makepath (zero-terminated-string zero-terminated-string zero-terminated-string address -> character --)`

Compose a path from separately provided filename extension, filename, directory path and drive letter and store it as a zero-terminated string starting at `address -> character`.

`_makepath` calls the *MSVCRT* library function `_makepath` with the first zero-terminated-string as `ext`, the second zero-terminated-string as `fname`, the third zero-terminated-string as `dir`, the fourth zero-terminated-string as `drive` and `address -> character` as `path`.

`_MB_CP_ANSI (-- multibyte-codepage)`

`msvcrt.sf`

`multibyte-codepage` is the `_MB_CP_ANSI` manifest constant of the *MSVCRT* library to be used as an input parameter to `_setmbcp`.

`_MB_CP_LOCALE (-- multibyte-codepage)` msvcrt.sf

multibyte-codepage is the `_MB_CP_LOCALE` manifest constant of the *MSVCRT* library to be used as an input parameter to `_setmbcp`.

`_MB_CP_OEM (-- multibyte-codepage)` msvcrt.sf

multibyte-codepage is the `_MB_CP_OEM` manifest constant of the *MSVCRT* library to be used as an input parameter to `_setmbcp`.

`_MB_CP_SBCS (-- multibyte-codepage)` msvcrt.sf

multibyte-codepage is the `_MB_CP_SBCS` manifest constant of the *MSVCRT* library to be used as an input parameter to `_setmbcp`.

`_mbbtype (multibyte-type character -- multibyte-type)`

The output parameter `multibyte-type` is the type of the byte character in a multibyte string. The input parameter `multibyte-type` is the type of the previous byte in the string. If `multibyte-type` is any value except 1, `_mbbtype` tests for a valid single-byte or lead byte of a multibyte character. If the value of `multibyte-type` is 1, `_mbbtype` tests for a valid trail byte of a multibyte character.

`_mbbtype` calls the *MSVCRT* library function `_mbbtype` with `integer` as `type` and `character` as `c`.

`_MBC_ILLEGAL (-- multibyte-type)` msvcrt.sf

`multibyte-type` is the `_MBC_ILLEGAL` manifest constant of the *MSVCRT* library to be used as parameters in `_mbbtype` and `_mbsbtype`.

`_MBC_LEAD (-- multibyte-type)` msvcrt.sf

`multibyte-type` is the `_MBC_LEAD` manifest constant of the *MSVCRT* library to be used as parameters in `_mbbtype` and `_mbsbtype`.

`_MBC_SINGLE (-- multibyte-type)` msvcrt.sf

`multibyte-type` is the `_MBC_SINGLE` manifest constant of the *MSVCRT* library to be used as parameters in `_mbbtype` and `_mbsbtype`.

`_MBC_TRAIL (-- multibyte-type)` msvcrt.sf

`multibyte-type` is the `_MBC_TRAIL` manifest constant of the *MSVCRT* library to be used as parameters in `_mbbtype` and `_mbsbtype`.

`_mbccpy (caddress -> character caddress -> character --)`

Copy one multibyte character from the first `caddress -> character` to the second `caddress -> character`.

`_mbccpy` calls the *MSVCRT* library function `_mbccpy` with the first address -> character as `src` and the second address -> character as `dest`.

`_mbclen (caddress -> character -- unsigned)`

`unsigned` is 1 or 2, according to the length of the multibyte character `caddress -> character` points to. For UTF-8, `signed` is always 1.

`_mbclen` calls the *MSVCRT* library function `_mbclen` with `caddress -> character` as `c`.

`_mbsbtype (unsigned caddress -> character -- multibyte-type)`

`multibyte-type` is the type of the byte at offset `unsigned` within the multibyte character string starting at `caddress -> character`. Only this byte is being examined, ignoring invalid characters before the specified byte.

`_mbsbtype` calls the *MSVCRT* library function `_mbsbtype` with `unsigned` as `count` and `caddress -> character` as `mbstr`.

`_mbslen (zero-terminated-string -- unsigned)`

`unsigned` is the length of the multibyte character string `zero-terminated-string`.

`_mbslen` calls the *MSVCRT* library function `_mbslen` with `zero-terminated-string` as `str`.

`_mbstrlen (zero-terminated-string -- signed)`

`signed` is the length of the multibyte character string `zero-terminated-string`, or -1 if the string contains an invalid multibyte character.

`_mbstrlen` calls the *MSVCRT* library function `_mbstrlen` with `zero-terminated-string` as `str`.

`_mkdir (zero-terminated-string -- flag)`

Creates a new directory with the absolute or relative path `zero-terminated-string`. `flag` is false if the operation succeeds, otherwise `flag` is true.

`_mkdir` calls the *MSVCRT* library function `_mkdir` with `zero-terminated-string` as `dirname`.

`_mktemp (zero-terminated-string -- 1st)`

Modifies `zero-terminated-string` in order to create a unique filename.

`zero-terminated-string` must have the form `baseXXXXXX`, where `base` is the unmodified part of the filename created. `_mktemp` replaces the first uppercase `X` by a lowercase letter `a` through `z` and the next five uppercase `X` by a sequence of five digits.

`_mktemp` calls the *MSVCRT* library function `_mktemp` with `zero-terminated-string` as `nameTemplate`.

`_mktime64 (address -> unsigned -- signed-double)`

Convert a time represented by an array of 9 unsigned single-cell values, into signed-double, the number of seconds elapsed since midnight (00:00:00), January 1, 1970, Coordinated Universal Time (UTC). `address -> unsigned` is the address of the first item of the array, whose structure is as follows:

item #	Description
0	seconds (0 - 59)
1	minutes (0 - 59)
2	hours (0 - 23)
3	day of month (1 - 31)
4	month (0 - 11; January = 0)
5	year minus 1900
6	day of week (0 - 6; Sunday = 0)
7	day of year (0 - 365; January 1 = 0)
8	0

`_mktime64` calls the *MSVCRT* library function `_mktime64` with `address -> unsigned` as `timeptr`.

`_msize (address -- unsigned)`

`unsigned` is the size of the allocated memory block at `address`. An ambiguous condition exists if `address` is not the address of the first byte of a memory block allocated with `calloc`, `malloc`, or `realloc`.

`_msize` calls the *MSVCRT* library function `_msize` with `address` as `memblock`.

`_onexit (address -- 1st)`

Add the function with code `address` to the list of functions that are to be called when StrongForth exits normally. The function may not have parameters. `1st` is equal to `address` if `_onexit` was executed successfully, otherwise `1st` is null.

`_onexit` calls the *MSVCRT* library function `_onexit` with `address` as `function`.

`_open (file-status file-control zero-terminated-string -- file-descriptor)`

Opens the file with the name `zero-terminated-string` with read and/or write access as specified by `file-control`. If `_open` creates a new file, `file-status` specifies the file permissions. If the operation is successful, `file-descriptor` is the file descriptor of the open file, otherwise `file-descriptor` is equal to `-1`.

`_open` calls the *MSVCRT* library function `_open` with `file-status` as `pmode`, `file-control` as `oflag` and `zero-terminated-string` as `filename`.

`_open_osfhandle (file-control file -- file-descriptor)`

Allocate an *MSVCRT* file descriptor `file-descriptor` with attributes `file-control` and associate it with the operating system file handle `file`.

`_open_osfhandle` calls the *MSVCRT* library function `_open_osfhandle` with `file-control` as `flags` and `file` as `osfhandle`.

`_pclose (file -- integer)`

Look up the process ID of the command processor started by the associated `_popen`, execute `_cwait` on the command processor, and close the stream on the associated pipe. `integer` is the exit status of the terminating command processor, or -1 if the operation fails.

`_pclose` calls the *MSVCRT* library function `_pclose` with `file` as `stream`.

`_pipe (file-control unsigned address -> file-descriptor -- flag)`

Opens a pipe with separate file descriptors for read and write access and the text translation as specified by `file-control`. `unsigned` is the number of bytes that are reserved for the pipe. The file descriptor for read operations and the file descriptor for write operations are stored at `address -> file-descriptor`.

`_pipe` calls the *MSVCRT* library function `_pipe` with `file-control` as `textmode`, `unsigned` as `psize` and `address -> file-descriptor` as `pfds`.

`_popen (zero-terminated-string zero-terminated-string -- file)`

Create a pipe. Asynchronously execute a spawned copy of the command processor with the second `zero-terminated-string` as the command line. The character string in the first `zero-terminated-string` specifies the type of access requested, as follows.

mode	Description
'r'	The calling process can read the spawned command's standard output using <code>file</code> .
'w'	The calling process can write to the spawned command's standard input using <code>file</code> .
'b'	Open in binary mode.
't'	Open in text mode.

`file` is a stream associated with one end of the created pipe, or null if the operation fails. The other end of the pipe is associated with the spawned command's standard input or standard output.

`popen` calls the *MSVCRT* library function `popen` with the first `zero-terminated-string` as `mode` and the second `zero-terminated-string` as `command`.

`_putch (character -- 1st)`

Type `character` on the console. If `_putch` is successful, `1st` is `character`, otherwise `1st` is EOF.

`_putch` calls the *MSVCRT* library function `_putch` with `character` as `c`.

`_putenv (zero-terminated-string -- flag)`

Adds a new local environment variable or modifies the value of an existing local environment variable. `zero-terminated-string` must be a string of the form `name=value`, where `name` is the name of the environment variable and `value` is the new value of the variable. If `name` does not exist in the local environment, it is added to the environment. If `value` is empty, `name` is removed from the list of local environment variables. `flag` is `false` if and only if the operation succeeds.

putenv calls the *MSVCRT* library function putenv with zero-terminated-string as envstring.

_putw (file single -- 2nd)

Write the binary value of single to the file. Advance the current position of file. 2nd is equal to single if the operation succeeds, otherwise 2nd is equal to -1.

_putw calls the *MSVCRT* library function _putw with file as stream and single as binint.

_read (unsigned caddress -> character file-descriptor -- unsigned)

Read up to unsigned bytes from the file associated with file-descriptor to the buffer starting at caddress -> character. unsigned is the number of bytes actually read. unsigned is less than the length of the buffer if there are not enough bytes between the current position of the file pointer and the end of the file.

_read calls the *MSVCRT* library function _read with unsigned as buffer_size, caddress -> character as buffer and file-descriptor as fd.

_rmdir (zero-terminated-string -- flag)

Removes the directory with the absolute or relative path zero-terminated-string. flag is false if the operation succeeds, otherwise flag is true.

_rmdir calls the *MSVCRT* library function _rmdir with zero-terminated-string as dirname.

_rmtmp (-- unsigned)

Remove all temporary files created by tmpfile in the current directory. unsigned is the number of temporary files closed and deleted.

_rmtmp calls the *MSVCRT* library function _rmtmp.

**_searchenv (caddress -> character zero-terminated-string
zero-terminated-string --)**

Search for an executable file with the name specified in the second zero-terminated-string. The file is first searched in the current working directory. If it is not found, the directories listed in the environment variable specified by the first zero-terminated-string are searched. If a file matching the given name is found, its full path is copied to caddress -> character as a zero-terminated string, otherwise an empty zero-terminated string is written to caddress -> character.

Note: The memory area starting at caddress -> character shall provide space for at least 260 characters.

_searchenv calls the *MSVCRT* library function _searchenv with caddress -> character as pathname, the first zero-terminated-string as varname and the second zero-terminated-string as filename.

`_setmaxstdio (unsigned -- 1st)`

Set the maximum number of files that may be open simultaneously at the stream I/O level to unsigned. 1st is unsigned if the operation is successful. Otherwise, 1st is -1.

`_setmaxstdio` calls the *MSVCRT* library function `_setmaxstdio` with unsigned as new_max.

`_setmbcp (multibyte-codepage -- flag)`

Sets the multibyte code page to multibyte-codepage. flag is false if and only if the operation is successful.

`_setmbcp` calls the *MSVCRT* library function `_setmbcp` with multibyte-codepage as codepage.

`_setmode (file-control file-descriptor -- 1st)`

Sets the translation mode of the file associated with file-descriptor to file-control. file-control may be either `O_TEXT` or `O_BINARY`. 1st is the previous translation mode of the file.

`_setmode` calls the *MSVCRT* library function `_setmode` with file-control as mode and file-descriptor as fd.

**`_sopen (file-status sharing-mode file-control
zero-terminated-string -- file-descriptor)`**

Opens the file with the name zero-terminated-string with shared read and/or write access as specified by sharing-mode and file-control. If `_sopen` creates a new file, file-status specifies the file permissions. If the operation is successful, file-descriptor is the file descriptor of the open file, otherwise file-descriptor is equal to -1.

`_sopen` calls the *MSVCRT* library function `_sopen` with file-status as pmode, sharing-mode as shflag, file-control as oflag and zero-terminated-string as filename.

**`_spawnv (address -> zero-terminated-string zero-terminated-string
process-mode -- handle)`**

Create and execute a new process. address -> zero-terminated-string points to a list of command-line parameters. zero-terminated-string is the path of the file to be executed. process-mode is the execution mode of the new process. handle is the exit status of the new process.

`_spawnv` calls the *MSVCRT* library function `_spawnv` with address -> zero-terminated-string as argv, zero-terminated-string as cmdname and process-mode as mode.

**`_spawnve (address -> zero-terminated-string address ->
zero-terminated-string zero-terminated-string process-mode -
handle)`**

Create and execute a new process. The first address -> zero-terminated-string points to a list of environment settings. The second address ->

zero-terminated-string points to a list of command-line parameters.
zero-terminated-string is the path of the file to be executed. process-mode is the execution mode of the new process. handle is the exit status of the new process.

`_spawnve` calls the *MSVCRT* library function `_spawnve` with the first address -> zero-terminated-string as envp, the second address -> zero-terminated-string as argv, zero-terminated-string as cmdname and process-mode as mode.

**`_spawnvp (address -> zero-terminated-string
zero-terminated-string process-mode -- handle)`**

Create and execute a new process. address -> zero-terminated-string points to a list of command-line parameters. zero-terminated-string is the path of the file to be executed. `_spawnvp` uses the PATH environment variable to find the file to execute. process-mode is the execution mode of the new process. handle is the exit status of the new process.

`_spawnvp` calls the *MSVCRT* library function `_spawnvp` with address -> zero-terminated-string as argv, zero-terminated-string as cmdname and process-mode as mode.

**`_spawnvpe (address -> zero-terminated-string address ->
zero-terminated-string zero-terminated-string process-mode --`**

Create and execute a new process. The first address -> zero-terminated-string points to a list of environment settings. The second address -> zero-terminated-string points to a list of command-line parameters. zero-terminated-string is the path of the file to be executed. `_spawnvpe` uses the PATH environment variable to find the file to execute. process-mode is the execution mode of the new process. handle is the exit status of the new process.

`_spawnvpe` calls the *MSVCRT* library function `_spawnvpe` with the first address -> zero-terminated-string as envp, the second address -> zero-terminated-string as argv, zero-terminated-string as cmdname and process-mode as mode.

**`_splitpath (caddress -> character caddress -> character caddress
-> character caddress -> character zero-terminated-string --
)`**

Split the path provided by zero-terminated-string into four components. The filename extension including leading period is stored as a zero-terminated string beginning at the first caddress -> character. The filename without extension is stored as a zero-terminated string beginning at the second caddress -> character. The directory path including leading and trailing backslashes is stored as a zero-terminated string beginning at the third caddress -> character. The drive letter with a trailing colon is stored as a zero-terminated string beginning at the fourth caddress -> character.

`_splitpath` calls the *MSVCRT* library function `_splitpath` with the first caddress -> character as ext, the second caddress -> character as fname, the third caddress -> character as dir, the fourth caddress -> character as drive and zero-terminated-string as path.

`_stat (address zero-terminated-string -- flag)`

Stores information about the file with the path zero-terminated-string in a structure located at address.

`_stat` calls the *MSVCRT* library function `_stat` with address as buffer and zero-terminated-string as path.

`_stat64 (address zero-terminated-string -- flag)`

Stores information about the file with the path zero-terminated-string in a structure located at address.

`_stat64` calls the *MSVCRT* library function `_stat64` with address as buffer and zero-terminated-string as path.

`_stati64 (address zero-terminated-string -- flag)`

Stores information about the file with the path zero-terminated-string in a structure located at address.

`_stati64` calls the *MSVCRT* library function `_stati64` with address as buffer and zero-terminated-string as path.

`_strdate (caddress -> character -- zero-terminated-string)`

Copy the current system date to the buffer pointed to by `caddress -> character`, formatted mm/dd/yy with a trailing null character, where mm are two digits representing the month, dd are two digits representing the day, and yy are the last two digits of the year. The buffer must be at least 9 bytes long. zero-terminated-string is equal to the buffer.

`_strdate` calls the *MSVCRT* library function `_strdate` with `caddress -> character` as datestr.

`_strtime (caddress -> character -- zero-terminated-string)`

Copy the current local time to the buffer pointed to by `caddress -> character`, formatted as hh:mm:ss with a trailing null character, where hh are two digits representing the hour in 24-hour notation, mm are two digits representing the minutes past the hour, and ss are two digits representing seconds. The buffer must be at least 9 bytes long. zero-terminated-string is equal to the buffer.

`_strtime` calls the *MSVCRT* library function `_strtime` with `caddress -> character` as timestr.

`_tell (file-descriptor -- unsigned)`

unsigned is the current position of the file pointer of the file associated with file-descriptor, or -1 if the operation fails.

`_tell` calls the *MSVCRT* library function `_tell` with file-descriptor as handle.

`__telli64 (file-descriptor -- unsigned-double)`

`unsigned-double` is the current position of the file pointer of the file associated with `file-descriptor`, or -1 if the operation fails.

`__telli64` calls the *MSVCRT* library function `__telli64` with `file-descriptor` as `handle`.

`__tempnam (zero-terminated-string zero-terminated-string -- zero-terminated-string)`

Generate the zero-terminated name of a file for a given directory that does not currently exist. The generated name is a concatenation of the prefix specified by the first `zero-terminated-string` and a sequential number, with no file extension. `__tempnam` uses `malloc` to allocate space for the filename. This space has to be explicitly freed when it is no longer needed. `zero-terminated-string` is the generated filename, or null if an error occurred.

If the `TMP` environment variable is defined and set to a valid directory name, unique filenames will be generated for the directory specified by `TMP`.

If the `TMP` environment variable is not defined or if it is set to the name of a directory that does not exist, `__tempnam` uses the directory specified by the second `zero-terminated-string` as the path for which it will generate a unique name.

If the `TMP` environment variable is not defined or if it is set to the name of a directory that does not exist, and if the second `zero-terminated-string` is either null or set to the name of a directory that does not exist, `__tempnam` uses the current working directory to generate a unique name.

`__tempnam` calls the *MSVCRT* library function `__tempnam` with the first `zero-terminated-string` as `prefix` and the second `zero-terminated-string` as `dir`.

`__time64 (address -> signed-double -- 2nd)`

`2nd` is the number of seconds elapsed since midnight (00:00:00), January 1, 1970, Coordinated Universal Time (UTC), according to the system clock. Additionally, store this value at `address -> signed-double`.

`__time64` calls the *MSVCRT* library function `__time64` with `address -> signed-double` as `destTime`.

`__tzset (--)`

Use the current setting of the environment variable `TZ` to assign values to three internal *msvcr*t variables. These variables are used by `__ftime` and `localtime` functions to make corrections from coordinated universal time (UTC) to local time, and by `time` to compute UTC from system time.

`__tzset` calls the *MSVCRT* library function `__tzset`.

`__umask (file-status -- 1st)`

Sets the file-permission mask of the current process to `file-status`. `1st` is the previous file-permission mask.

`_umask` calls the *MSVCRT* library function `_umask` with `file-status` as `pmode`.

`_ungetch (character -- 1st)`

Pushes `character` back to the console, assuming that `character` was the most recent character read from the console. If `_ungetch` is successful, `1st` is `character`, otherwise `1st` is EOF.

`_ungetch` calls the *MSVCRT* library function `_ungetch` with `character` as `c`.

`_unlink (zero-terminated-string -- flag)`

Delete the file with the name given by `zero-terminated-string`. `flag` is false if and only if the file was successfully deleted.

`_unlink` calls the *MSVCRT* library function `_unlink` with `zero-terminated-string` as `filename`.

`_USEDENTRY (-- heap-constant)`

msvcrt.sf

`heap-constant` is the `_USEDENTRY` manifest constant of the *MSVCRT* library.

`_utime (address -> signed zero-terminated-string -- flag)`

Sets the access time and the modification date on the file whose path is specified by `zero-terminated-string`. The two time values are expected in the given order at `address -> signed` as the number of seconds elapsed since midnight (00:00:00), January 1, 1970, Coordinated Universal Time (UTC). `flag` is false if and only if the operation succeeds.

`_utime` calls the *MSVCRT* library function `_utime` with `address -> signed` as `times` and `zero-terminated-string` as `filename`.

`_utime64 (address -> signed-double zero-terminated-string -- flag)`

Sets the access time and the modification date on the file whose path is specified by `zero-terminated-string`. The two time values are expected in the given order at `address -> signed-double` as the number of seconds elapsed since midnight (00:00:00), January 1, 1970, Coordinated Universal Time (UTC). `flag` is false if and only if the operation succeeds.

`_utime64` calls the *MSVCRT* library function `_utime64` with `address -> signed-double` as `times` and `zero-terminated-string` as `filename`.

`_write (unsigned address -> character file-descriptor -- unsigned)`

Write up to unsigned bytes from the buffer starting at `address -> character` to the file associated with `file-descriptor`. `unsigned` is the number of bytes actually written. If the operation fails, `unsigned` is less than the length of the buffer.

`_write` calls the *MSVCRT* library function `_write` with `unsigned` as `buffer_size`,
`caddress` -> `character as buffer` and `file-descriptor` as `fd`.