

msvcrt — Useful routines from the MS VC++ runtime

These functions provide access to some useful capabilities on Windows platforms. Some higher-level modules use these functions to build the Windows implementations of their services. For example, the `getpass` module uses this in the implementation of the `getpass()` function.

Further documentation on these functions can be found in the Platform API documentation.

The module implements both the normal and wide char variants of the console I/O api. The normal API deals only with ASCII characters and is of limited use for internationalized applications. The wide char API should be used where ever possible.

Changed in version 3.3: Operations in this module now raise `OSError` where `IOError` was raised.

File Operations

`msvcrt.locking(fd, mode, nbytes)`

Lock part of a file based on file descriptor `fd` from the C runtime. Raises `OSError` on failure. The locked region of the file extends from the current file position for `nbytes` bytes, and may continue beyond the end of the file. `mode` must be one of the `LK_*` constants listed below. Multiple regions in a file may be locked at the same time, but may not overlap. Adjacent regions are not merged; they must be unlocked individually.

`msvcrt.LK_LOCK`

`msvcrt.LK_RLCK`

Locks the specified bytes. If the bytes cannot be locked, the program immediately tries again after 1 second. If, after 10 attempts, the bytes cannot be locked, `OSError` is raised.

`msvcrt.LK_NBLCK`

`msvcrt.LK_NBRLOCK`

Locks the specified bytes. If the bytes cannot be locked, `OSError` is raised.

`msvcrt.LK_UNLCK`

Unlocks the specified bytes, which must have been previously locked.

`msvcrt.setmode(fd, flags)`

Set the line-end translation mode for the file descriptor `fd`. To set it to text mode, `flags` should be `os.O_TEXT`; for binary, it should be `os.O_BINARY`.

`msvcrt.open_osfhandle(handle, flags)`

Create a C runtime file descriptor from the file handle *handle*. The *flags* parameter should be a bitwise OR of `os.O_APPEND`, `os.O_RDONLY`, and `os.O_TEXT`. The returned file descriptor may be used as a parameter to `os.fdopen()` to create a file object.

`msvcrt.get_osfhandle(fd)`

Return the file handle for the file descriptor *fd*. Raises `OSError` if *fd* is not recognized.

Console I/O

`msvcrt.kbhit()`

Return true if a keypress is waiting to be read.

`msvcrt.getch()`

Read a keypress and return the resulting character as a byte string. Nothing is echoed to the console. This call will block if a keypress is not already available, but will not wait for Enter to be pressed. If the pressed key was a special function key, this will return `'\000'` or `'\xe0'`; the next call will return the keycode. The Control-C keypress cannot be read with this function.

`msvcrt.getwch()`

Wide char variant of `getch()`, returning a Unicode value.

`msvcrt.getche()`

Similar to `getch()`, but the keypress will be echoed if it represents a printable character.

`msvcrt.getwche()`

Wide char variant of `getche()`, returning a Unicode value.

`msvcrt.putch(char)`

Print the byte string *char* to the console without buffering.

`msvcrt.putwch(unicode_char)`

Wide char variant of `putch()`, accepting a Unicode value.

`msvcrt.ungetch(char)`

Cause the byte string *char* to be “pushed back” into the console buffer; it will be the next character read by `getch()` or `getche()`.

`msvcrt.ungetwch(unicode_char)`

Wide char variant of `ungetch()`, accepting a Unicode value.

Other Functions

`msvcrt.heapmin()`

Force the `malloc()` heap to clean itself up and return unused blocks to the operating system. On failure, this raises [OSError](#).