Monolith64 ABI

SUBJECT TO CHANGE
Edited 2025-11-08

System Calls

System calls or **syscalls** are a way for a userland application to request services from the kernel.

Syscalls are triggered using the `0x69` exception, arguments are passed via CPU registers with the syscall number being placed in `rax` and additional arguments passed according to System V convention (1-6 ordered: `rdi`, `rsi`, `rdx`, `rcx`, `r8`, `r9`).

The currently available syscalls are as follows:

ID	Name	Args	Return	Description
`0`	`read`	`fd`, `buf`, `num`	bytes read in `rax`	Reads `size_t num` bytes into `char* buf` from file descriptor `int fd`.
`1`	`write`	*(unspecified)*	*(unspecified)*	[STUB] Implementation not yet defined.
`2`	`open`	`path`, `flags`, `mode`	file descriptor in `rax`	Opens a file located at `const char* path` with given `int flags` and `mode_t mode`. (currently `flags == mode`)
`3`	`close`	`fd`	none	Closes an open file descriptor `int fd`.
`4`	`map_page`	`va`, `pa`, `flags`	none	Maps a physical page `pa` to virtual address `va` with given `flags`.
`5`	`serial_puts`	`str`	none	Writes a null-terminated string `const char* str` to the serial port.
`6`	`serial_puthex`	`value`	none	Prints `uint64_t value` as hexadecimal over the serial port.
`7`	`serial_putc`	`ch`	none	Sends a single character `char ch` to the serial port.
`8`	`print`	`buf`, `len`	none	Writes `size_t len` bytes from `const char* buf` to the graphical terminal.
`9`	`exit`	none	none	Terminates the current running task.

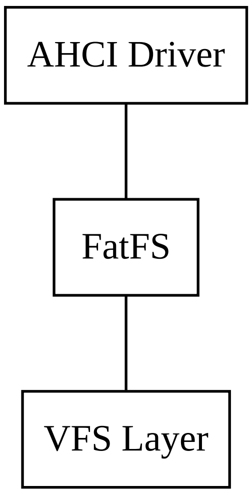
ID	Name	Args	Return	Description
`10`	`get_fb_addr`	none	fb address in `rax`, fb size in `rbx`	Returns the framebuffer base address and size to user mode.
`11`	`lb_read`	none	character in `rax`	Reads the next character from the terminal input buffer (line buffer).
`12`	`acpi_sleep`	`state`	none	Puts the system into ACPI sleep state `state`.
`13`	`opendir`	`path`	directory handle in `rax`	Opens a directory at `const char* path` and returns a handle.
`14`	`readdir`	`dirfd`, `user_fno`	result code in `rax`	Reads next directory entry into user buffer `FILINFO* user_fno`.
`15`	`read_rtc`	`user_rtc`	none	Copies current real-time clock data into `rtc_t* user_rtc`.
`16`	`exec`	`int fd`	none (replaces task)	Loads and executes a program located at file descriptor `int fd`.
`17`	`get_fb_info`	`reserved`, `user_fb_info`	status in `rax`	Copies framebuffer metadata (`fb_info`) to user buffer `struct fb_info*`.

All Invalid syscall IDs will return `0xBADCA11`.

Example

```
; open file
mov rax, 2              ; open syscall
lea rdi, [rel path]     ; path to file
mov rsi, 0              ; flags (read-only)
mov rdx, 0              ; mode
int 0x69
```

File System



Monolith64 uses a custom AHCI driver and VFS layer with FatFS [a generic FAT/exFAT filesystem module].

exFAT is currently **not** supported and only FAT32 is tested. Additionally the driver is currently read-only.

Mode flags

Please see the FatFS documentation on f_open for information on flags to pass to open.

Terminal

Monolith64 uses Flanterm as it's graphical terminal emulator, it supports **ANSI escape sequences** as defined in **ISO/IEC 6429**.

It can be written to via the print syscall [ID `8`] and read from via the line buffer obtained via syscall `11`.

Framebuffer

Monolith64 uses a linear framebuffer mapped into usermode at address `0xFB0000`, of which info can be obtained using syscall ID `17`.

Data Structures

The kernel may require specific data structures to use it's services, these are defined below.

File System

`FILINFO`

```
typedef struct {
    unsigned int fsize;          /* File size (invalid for directory) */
    uint16_t    fdate;          /* Date of file modification or directory creation */
    uint16_t    ftime;          /* Time of file modification or directory creation */
    uint8_t    fattrib;        /* Object attribute */
    char    fname[12 + 1];  /* Object name */
} FILINFO;
```

For additional file system information + examples, see the **FatFS documentation**.

Others

`rtc_t`

```
typedef struct rtc_t {
    unsigned char second;
    unsigned char minute;
    unsigned char hour;
    unsigned char day;
    unsigned char month;
    unsigned int year;

    unsigned char century;
    unsigned char last_second;
    unsigned char last_minute;
    unsigned char last_hour;
    unsigned char last_day;
    unsigned char last_month;
    unsigned char last_year;
    unsigned char last_century;
    unsigned char registerB;
} rtc_t;
```

`fb_info`

```
struct fb_info {
    uint64_t width;
    uint64_t height;
    uint64_t pitch;
    uint16_t bpp;
};
```