Overview

The mdadm library is a group of functions designed to implement linear memory storage on a series of JBOD disks. The core functionality of the mdadm library is to take a series of storage disks and allow the user to interface with them as one contiguous block of memory. Interfacing with mdadm functions specifically means reading from and writing to the JBOD disks. The implementation also supports caching to improve performance and networking to access remote JBOD servers. The functions below provide the mdadm services.

Functions

mdadm_mount()

Format

int mdadm_mount(void);

Parameters

- void

Description

mdadm_mount is used to mount the JBOD disks for interfacing. The function takes no parameters and returns 1 on a success and -1 on a failure. A call to mdadm_mount will fail if (1) the system is already mounted (a call to mount was made previously without calling unmount).

mdadm_unmount()

Format

int mdadm_unmount(void);

Parameters

- void

Description

mdadm_unmount is used to unmount the JBOD disks when the user is done interfacing. The function takes no parameters and returns 1 on a success and -1 on a failure. A call to mdadm_unmount will fail if (1) the system hasn't been mounted.

mdadm_read()

Format

int mdadm_read(uint32_t addr, uint32_t len, uint8_t *buf);

Parameters

- addr: an integer value representing the first byte the user wants to read from the linear JBOD disks.
- len: an integer value representing the length of memory to be read from the JBOD disks.
- buf: a pointer to a memory address for the read to be copied into.

Description

mdadm_read is used to read a length of memory from the JBOD disks. The function takes 3 parameters and returns a 1 on success and a -1 on failure. A call to mdadm_read will fail if (1) the user tries to read from an address less than 0, (2) the user tries to read a nonzero length of memory into a null buffer, (3) the user tries to read a length of memory that exceeds the JBODs address space, (4) the user tries to read more than 1024 bytes of memory in one call, or (5) the user tries to call read on an unmounted system.

mdadm_write()

Format

int mdadm_write(uint32_t addr, uint32_t len, const uint8_t *buf);

Parameters

- addr: an integer value representing the first byte the user wants to write to the linear JBOD disks.
- len: an integer value representing the length of memory to be written to the JBOD disks.
- buf: a pointer to a memory address for the write to be copied from.

Description

mdadm_write is used to write a length of memory to the JBOD disks. The function takes 3 parameters and returns a 1 on success and a -1 on failure. A call to mdadm_write will fail if (1) the user tries to write to an address less than 0, (2) the user tries to write a nonzero length of memory from a null buffer, (3) the user tries to write a length of memory that exceeds the JBODs address space, (4) the user tries to write more than 1024 bytes of memory in one call, or (5) the user tries to call write on an unmounted system.

cache_create()

Format

int cache_create(int num_entries);

Parameters

- num_entries: an integer value representing the number of entries that the user wants the cache to have.

Description

cache_create is used to create a cache to improve the fetching performance of the JBOD systems memory. The function takes 1 parameter and returns 1 on a success and -1 on a failure. A call to cache_create will fail if (1) there is already a cache running, (2) the number of entries given is less than 2, or (3) the number of entries given is greater than 4096.

cache_destroy()

Format

int cache_destroy(void);

Parameters

- void

Description

cache_destory is used to destroy a running cache when the user no longer needs it. The function takes no parameters and returns 1 on a success and 0 on a failure. A call to cache_destroy will fail if (1) there isn't a cache running when the call occurs.

jbod_connect()

Format

bool jbod_connect(const char *ip, uint16_t port);

Parameters

- *ip: a pointer to the first byte of a string representing the ip of the JBOD server that the user wants the client-side socket to connect to.
- port: an integer value representing the port number of the JBOD server that the user wants the client-side socket to connect to.

Description

jbod_connect is used to connect the user's client-side socket to a remote JBOD server. The function takes two parameters and returns *true* on a success and *false* on a failure. The function will fail if (1) the user inputs an invalid ip or (2) a network error interrupts the connection process.

jbod_disconnect()

Format

void jbod_disconnect(void);

Parameters

- void

Description

jbod_disconnect is used to disconnect from a remote JBOD server. The function takes no parameters and returns nothing. This function will not fail.