Cruz FileSystem

This document contains every detail about the functionality, implementation and use cases of every function defined in the API developed by ThanOS.

cr_open

```
#include "cr_API.h"
crFILE *cr_open(char *path, char mode);
```

Description

Opens a `crFILE`. If `mode` is `'r'`, a reader `crFILE` will be returned. If `mode` is `'w'`, a writer `crFILE` will be returned.

Return Value and Error Handling

Upon success, the method will return a `crFILE` struct. In any other case, the method will log a message to `stderr` and return `NULL`. The method will fail in any of the next scenarios:

- If `path` is an invalid path to the file
- If `mode` is `'r'` and `path` does not lead to an existing file
- If `mode` is `'w'` and `path` leads to a file that already exists
- If `mode` is `'w'` and the filesystem has no more space
- If `mode` is invalid

Notes

A reader `crFILE` has a pointer to the last byte read from it. Once its pointer gets to the end of the file, it will be impossible to read it again. Likewise, a writer `crFILE` can be used only once to write in it. It is a one use struct.

cr_read

```
#include "cr_API.h"
int cr_read(crFILE *file_desc, void *buffer, int nbytes);
```

Description

Read `file_desc` from the last read byte (`file_desc->reader`), until the next `nbytes`.

Return Value and Error Handling

Returns the number of bytes read. If `nbytes` plus the numbers of bytes read to the moment is less than the total size of the file, returns exactly `nbytes`. In the other case, it returns `nbytes` less the numbers of bytes unread.

If the pointer to `crFILE` is `NULL`, logs a message to `stderr` and returns `-1`. If `crFILE` is a writer file, logs a message to `stderr` and returns `-1`.

Notes

Notice that when the `reader` of the `crFILE` reaches the end of the file, the amount of bytes to be read turns into `0`, so every next call to `cr_read` using that same `crFILE` will result in nothing being saved in the buffer and a return value of `0`.

cr_write

```
#include "cr_API.h"
int cr_write(crFILE *file_desc, void *buffer, int nbytes);
```

Description

Writes the content of `buffer` to `file_desc`, specifically the first `nbytes` of `buffer`. If the filesystem has not enough space for the `nbytes`, the file won't be written at all and cr_write will fail.

Return Value and Error Handling

If `crFILE` is a `NULL` pointer, logs a message to `stderr` and returns `-1`. If `crFILE` is a reader file, logs a message to `stderr` and returns `-1`. If `crFILE` has already been written, logs a message to `stderr` and returns `-1`. If the filesystem fails to find enough space for the whole file, `cr_write` will log a message to `stderr` and return `0`. Otherwise, `cr_write` will return the amount of bytes written to `file_desc`.

cr_close

```
#include "cr_API.h"
int cr_close(crFILE *file_desc);
```

Description

Closes the file saved on `file_desc` and frees its memory.

Return Value and Error Handling

This function returns `1` if the pointer to `crFILE` is `NULL`. In any other case, it logs a message to `stderr` and returns `0`.

cr_rm

```
#include "cr_API.h"
int cr_rm(char *path);
```

Description

Remove the file in `path` from the file sistem, freeing all the memory used by it.

Return Value and Error Handling

Returns `1` if succeeded, `0` otherwise.
In case that `mounted_disk` is set to `NULL`, it logs a message to `stderr`.
In case that `path` does not exist, it logs a message to `stderr`.

cr_unload

```
#include "cr_API.h"
int cr_unload(char *orig, char *dest);
```

Description

Unloads something from the virtual location `orig` to the real location `dest`. If `orig` corresponds to a file, only that file will be copied directly inside `dest`. Otherwise, `orig` corresponds to a folder. That folder will be recursively copied to `dest`.

Return Value and Error Handling

The function returns `1` if nothing goes wrong. If the real or the virtual destination and origin are wrong, this method will return `0`.

cr_load

```
#include "cr_API.h"
int cr_load(char *orig);
```

Description

Loads something from the real location `orig` to the root of the virtual file, `/`. If `orig` corresponds to a file, only that file will be copied directly inside `/`. Otherwise, `orig` corresponds to a folder. That folder will be recursively copied to `/`.

Return Value and Error Handling

This function returns `1` when it has copied at lest 1 file inside `/`

cr_mount

```
#include "cr_API.h"
void cr_mount(char *diskname);
```

Description

Opens the disk `diskname` and saves a pointer to it on the environmental variable `mounted_disk`.

Return Value and Error Handling

The function returns void. In case that `diskname` does not exist, it logs a message to `stderr` and sets the environmental variable `mounted_disk` to `NULL`.

cr_unmount

```
#include "cr_API.h"
void cr_unmount();
```

Description

Closes the disk loaded on the environmental variable `mounted_disk` and frees its memory.

Return Value and Error Handling

The function returns void. In case that `mounted_disk` is set to `NULL`, it logs a message to `stderr`.

cr_bitmap

```
#include "cr_API.h"
void cr_bitmap(unsigned block, bool hex);
```

Description

Shows a representation of the bitmap of `mounted_disk` in `stderr`.

If `block` is `0`, it shows the whole bitmap and then the amount of used blocks and the amount of free blocks left. If `block` is any number between `1` and `129` (both included), it shows the bitmap block in that position.

If `hex` is `true`, the bitmap will show every byte as a hex value. If `hex` is `false`, the bitmap will show every bit on its own.

Return Value and Error Handling

The function returns void. In case that `block` is a value different than the ones specified (`0` to `129`), it logs a message to `stderr`. If no disk is mounted, it logs a message to `stderr`.

Notes

As the bitmap contains 131072 bytes of information, calling the function with `block` set to `0` will probably overflow the console, so it is advisable to change the `stderr` file to an external file and use it as a debug log.

cr_exists

```
#include "cr_API.h"
int cr_exists(char *path);
```

Description

Given a `path`, it returns 1 if the file/folder exists and a 0 if it does not.

Return Value and Error Handling

The function returns `1` or `0`. If no disk is mounted, it logs a message to `stderr` and returns `0`.

cr_ls

```
#include "cr_API.h"
void cr_ls(char *path);
```

Description

Prints to `stdout` the files and directories inside the directory `path`.

Return Value and Error Handling

The function returns void. In case that `path` does not exist, it logs a message to `stderr`. If no disk is mounted, it logs a message to `stderr`.

cr_mkdir

```
#include "cr_API.h"
int cr_mkdir(char *foldername);
```

Description

Creates a folder in `foldername` if the complete path up to the last `/` exists.

Return Value and Error Handling

The function returns `1` if it succeeds. If the path up to the last `/` does not exist, it logs a message to `stderr` and returns `0`. If a folder with the same name in the same path exists, it logs a message to `stderr` and returns `0`. If the disk has no space left, it logs a message to `stderr` and returns `0`. If no disk is mounted, it logs a message to `stderr` and returns `0`.

Notes

Paths like `/dir1/dir2/` will be pre-processed to be `/dir1/dir2` and then created.