Systems II
Project 3, Part B
File system (writable)

1 Overview

Continuing the work from Project-3A, we will add system calls that provide the ability to create, write, and delete files.

2 Modifying the kernel to write to the file system

The following system calls should be provided to user processes so that they may alter the files on the file system:

- `WRITE (0xca110007)`: Write bytes from a buffer space into a given file.

  ```
  int write (char* filename, int offset, int length, void* buffer);
  ```

  The `filename` specifies the file to which to write. `length` bytes are copied from `buffer` and into the file starting at position `offset` in the file. It is the responsibility of the caller to ensure that `buffer` is a space that is at least `length` bytes.

  Note that the `offset` may be any value from `0` to the *length* of the file. By setting this argument to the file's current length, the written bytes are appended to the end of the file. If `offset + length` exceeds the file's length, then the file is extended.

  If `filename` is `NULL` or does not exist, this call should return `-1`. If any of `offset` through `offset + length` fall outside of the file's size, return `-2`. If `buffer` is `NULL`, return `-3`.

  Upon success, this call should return *the number of bytes written*. If the call fails, its return value depends on the nature of the failure:

  - `-1` (*invalid filename*): `filename` is `NULL` or is too long (exceeds the 60-byte limit).
  - `-2` (*nonexistant file*): `filename` does not exist in the directory.
  - `-3` (*invalid range*): If `offset` exceeds the *length* of the file.

  - `-4` (*invalid buffer*): If `buffer` is `NULL` or is an invalid address.

  - `-5` (*allocation failure*): The block device has no free blocks.

  - `-6` (*other failure*): Any other cause of failure.

- `CREATE (0xca110008)`: Create a new file and provide a name by which it can be accessed. The function prototype is:

  ```
  int create (char* filename);
  ```

  The `filename` argument should point to a null-terminated string with the name that should lead to a newly created file object. Upon success, this call should return `0`. Failure should return a negative value as follows:

  - `-1` (*invalid filename*): `filename` is `NULL` or is too long (exceeding the maximum filename length of 60 characters).

  - `-2` (*file exists*): `filename` already appears in the directory.

  - `-3` (*allocation failure*): The block device has no free blocks.

  - `-4` (*other failure*): Any other cause of failure.

- `DELETE (0xca110009)`: Delete a file by removing both its directory entry and the file object itself.

  ```
  int delete (char* filename);
  ```

  Return `0` upon success; return `-1` for an *invalid filename*, and return `-2` if the named file does not exist.

- `RENAME (0xca11000a)`: Update the directory by changing the name of a file.

  ```
  int rename (char* oldname, char* newname);
  ```

  If `oldname` exists in the directory, its value should be substituted in the directory entry with `newname`.

  Upon success, this call returns `0`. Upon an error, return:

  - `-1` (*invalid filename*): `oldname` or `newname` are either `NULL` or provide names that are too long.
  - `-2` (*nonexistant file*): `oldname` does not exist in the directory.
  - `-3` (*file exists*): `newname` already appears in the directory.
  - `-4` (*other error*): Any other cause of failure.

To test of these capabilities, write test programs that create, write, read, rename, and delete files.

3 How to submit your work

Copy into the `project-3/` directory in your shared Google Drive folder **all** of your source code: Every `.asm`, `.h`, and `.c` file. Include test programs and data files (if any) that those programs might use. It should be straightfoward to build and run your kernel and test programs.

This assignment is due at **11:59 pm** on **Tuesday, May** 5^{th}**.**