Operating Systems Lab

Lab-4

Date: 21-06-2021

Group 9

Abhishek Raj (180010002)

Harsh Raj (180010017)

Sri Priya (180010025)

Design Decisions For implementation:

simplefs-ops.h

It contains the declarations of various methods used for operating on the simple file system.

simplefs-ops.c

It contains the definitions of various methods used for operating on the simple file system. The implementation details are mentioned in the following screenshots of the function declarations.

```
/*
creates a file with a specified name in the filesystem. A file creation
should succeed only if a file with the same name does not already exist in
the filesystem, and if the system has space for an additional file to be
created. During file creation, this function must allocate a free inode for
the file and initialize it suitably on disk. This function returns the
inode number of the newly created file on success, and -1 on a failure.
*/
int simplefs_create(char *filename)
```

```
/*
deletes a file with the specified name from the filesystem (if it exists),
and frees up the resources of the file such as its data blocks and its
inode. You may assume that the file is closed before deleting it.
*/
void simplefs_delete(char *filename)
```

```
/*
opens an existing file for reading and writing. This operation succeeds
only if the file with the same name was previously created. This operation
allocates an unused file handle from the global file handle array,
initializes it suitably, and returns the index of the newly allocated file
handle. This function returns -1 if the file open fails for any reason.
*/
int simplefs_open(char *filename)
```

```
/*
closes an open file and frees up its file handle. This operation will not
delete the file from disk
*/
void simplefs_close(int fileHandle)
```

```
/*
reads the specified number of bytes from the current offset in an open file
into the given buffer. The requested number of bytes can span multiple
blocks on disk. This function returns 0 on success and -1 on failure.
*/
int simplefs_read(int fileHandle, char *buf, int nbytes)
```

```
/*
writes the specified number of bytes from the current offset in an open
file to disk, from the given buffer. The requested number of bytes can span
multiple blocks on disk. This function returns 0 on success and -1 on
failure
*/
int simplefs_write(int fileHandle, char *buf, int nbytes)
```

```
/*
Increments the file offset in the file handle by nseek bytes. As with Linux
filesystems, the offset indicates the next byte that can be read/written
from the file.
*/
int simplefs_seek(int fileHandle, int nseek)
```

```
/*
Perform a search operation on inode with name `filename`. It is just a
helper function for the above functions.
*/
int search_file(char *filename)
```

Screenshot (Test Cases Output):

```
harshraj22 in simplefs-code on main [!$]
$ ./autograder.sh testcases expected_output
Testcases: testcases
Expected output: expected_output
Running testcase testcases/testcase0.c: Output stored in myoutput/testcase0.out
Running testcase testcases/testcase1.c: Output stored in myoutput/testcase1.out
Running testcase testcases/testcase2.c: Output stored in myoutput/testcase2.out
Running testcase testcases/testcase3.c: Output stored in myoutput/testcase3.out
Running testcase testcases/testcase4.c: Output stored in myoutput/testcase4.out
Running testcase testcases/testcase5.c: Output stored in myoutput/testcase5.out
Running testcase testcases/testcase6.c: Output stored in myoutput/testcase6.out
Running testcase testcases/testcase7.c: Output stored in myoutput/testcase7.out
Comparing expected_output/testcase0.out and myoutput/testcase0.out
Test Case Passed


Comparing expected_output/testcase1.out and myoutput/testcase1.out
Test Case Passed


Comparing expected_output/testcase2.out and myoutput/testcase2.out
Test Case Passed


Comparing expected_output/testcase3.out and myoutput/testcase3.out
Test Case Passed


Comparing expected_output/testcase4.out and myoutput/testcase4.out
Test Case Passed


Comparing expected_output/testcase5.out and myoutput/testcase5.out
Test Case Passed


Comparing expected_output/testcase6.out and myoutput/testcase6.out
Test Case Passed


Comparing expected_output/testcase7.out and myoutput/testcase7.out
Test Case Passed


Test Cases Passed: 8
Test Cases Total: 8
```