

Project 5: s2fs (Super Simple File System)

- **Handed out:** Wednesday, April 27, 2022

- **Due dates:**

- Friday, May 6, 2022

Introduction

The goal of this project is to implement a simple pseudo file system called name `s2fs`. A pseudo file system is not backed by disk but resides in memory and is usually used to provide information regarding the kernel to the user (e.g., `proc` file system).

Please read the following articles and sample codes to understand how to write a simple file system using `libfs`. Note that these articles are written for older Linux versions, and thus some APIs might have been renamed/removed.

Program your module in `s2fs.c` and `s2fs.h` (as needed). Create `Makefile` that support *all*, *clean*, *install* and *uninstall* rules (and more as needed), similar to that of project 2.

Recommended Background Reading

- [Creating Linux virtual filesystems](#)
- [Sample lwnfs code](#)
- [Writing a File System in Linux Kernel](#)
- [File System Implementation](#)

Part 1: Create a Pseudo File System

[20 points] Design a kernel module called `s2fs`, a mountable pseudo file system.

Tasks:

- Define `struct file_system_type s2fs_type`.
- Use `mount_nodev` to mount a pseudo file system.
- Define a function named `int s2fs_fill_super(...)` to fill a superblock and pass it as an argument for `mount_nodev`.
- During the module init, register the `s2fs_type` filesystem. During the module exit, unregister the file system.

Deliverables:

- Run the following commands and take a screenshot of the output. Name your screenshot as `s2fs1.png`

```
$ mkdir mnt
$ sudo insmod s2fs.ko # Inserting the filesystem kernel module
$ sudo mount -t s2fs nodev mnt # Mount at dir mnt with option nodev
$ mount # Print out all mounted file systems
$ sudo umount ./mnt # Unmount the file system
$ sudo rmmod s2fs.ko # Remove the kernel module
```

Part 2: Implement File Operations

Now we will add inode and file operations to `s2fs`.

Part 2.1: Create an inode

[15 points] Write a function to create an inode, called `static struct inode *s2fs_make_inode(struct super_block *sb, int mode)`. The function accepts two inputs, the superblock of the filesystem and the mode which decides the type (a directory vs a file) and the permission. Make sure you set the `i_ino` field of inode with `get_next_ino()`.

Part 2.2: Create a directory

[15 points] Write a function to create a directory, called `static struct dentry *s2fs_create_dir(struct super_block *sb, struct dentry *parent, const char *dir_name);`

Part 2.3: File operations

[15 points] To handle a file, the filesystem needs to know how to open, read and write a file. Write three functions, `s2fs_open`, `s2fs_read_file`, and `s2fs_write_file`. In this project we will not use the open and the write functions so they will simply return 0. The read function should return "Hello World!" string to the user.

Create a `s2fs_fops` of type `file_operations` and assign `.read`, `.write` and `.open` with the functions you wrote here.

Part 2.4: Create a file

[15 points] Write a function to create a file, called `static struct dentry *s2fs_create_file(struct super_block *sb, struct dentry *dir, const char *file_name)`. The function should create a file with the name stored in `file_name` inside directory pointed by `dir` dentry. Set `s2fs_fops` as `file_operations`.

Part 2.5: Putting it all together

[10 points] Update `s2fs_fill_super()` function (Part 1) so that after mounting, `s2fs` creates (1) a directory named `foo` in the root directory, and (2) a file named `bar` inside the subdirectory `foo`. Use the `s2fs_create_dir` and `s2fs_create_file` functions, defined in Parts 2.2 and 2.4.

Part 2.6: Deliverables

[10 points] Run the following commands and take a screenshot of the output. Name your screenshot as `s2fs2.png`

```
$ sudo insmod s2fs.ko
$ sudo mount -t s2fs nodev mnt # mount the filesystem
$ cd mnt/foo # change the directory
$ cat bar # read bar. check if you can see ``Hello World!''
$ cd ../../
$ sudo umount ./mnt # unmount
$ sudo rmmod s2fs.ko
```

Make a folder named with your SBU ID (e.g., 112233445), put `Makefile`, `s2fs.c`, `s2fs.h` (if exists), and the screenshots `s2fs{1,2}.png` file in the folder, create a single gzip-ed tarball named `[SBU ID].tar.gz`, and turn the gzip-ed tarball to Blackboard.