

### Report on File System Assignment (extra credit):

1. Program features an in-memory filesystem with basic functions such as `fs_read` and `fs_write` which are used to read and write into the file(s) newly created using the `fs_create` function.
2. Output of the program:

The output of the program is as follows:

The image shows a Kali Linux desktop environment. At the top is a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. On the left is a vertical dock with icons for a web browser, file manager, terminal, and other applications. The main window is a terminal titled 'xsh \$ ftest'. It displays the output of a command, showing a large amount of hex data. The data is organized into two main sections, each starting with a line of 110 zeros followed by a line of 1000 zeros. The first section contains 10 lines of hex data, and the second section contains 10 lines of hex data. The data appears to be a large file named 'uvwxyz' that has been converted to hex. The terminal window has a title bar with 'xsh \$ ftest' and a standard Linux window control bar. The desktop background is a dark, textured image.

The IU logo is printed on the screen along with the buffer contents read from the file just written into.

- 3.

Functions used:

Shree:

```
int fs_seek(int fd, int offset);  
int fs_read(int fd, void *buf, int nbytes);  
int fs_write(int fd, void *buf, int nbytes);  
int get_next_block();  
int get_block(int c_inode_indx, int num_d_blk);
```

Saurabh:

```
int fs_create(char *filename, int mode);  
int fs_open(char *filename, int flags);  
int fs_close(int fd);  
int get_inode(char *f_name);
```

Discussed with **Pruthvi Shetty (shettypr)** about the implementation strategy.