Operating Systems Laboratory (Gx) (2017-18)

Dashboard / My courses / OSLAB_GxJan2018 / 20 February - 26 February / Design and Implement a filessytem

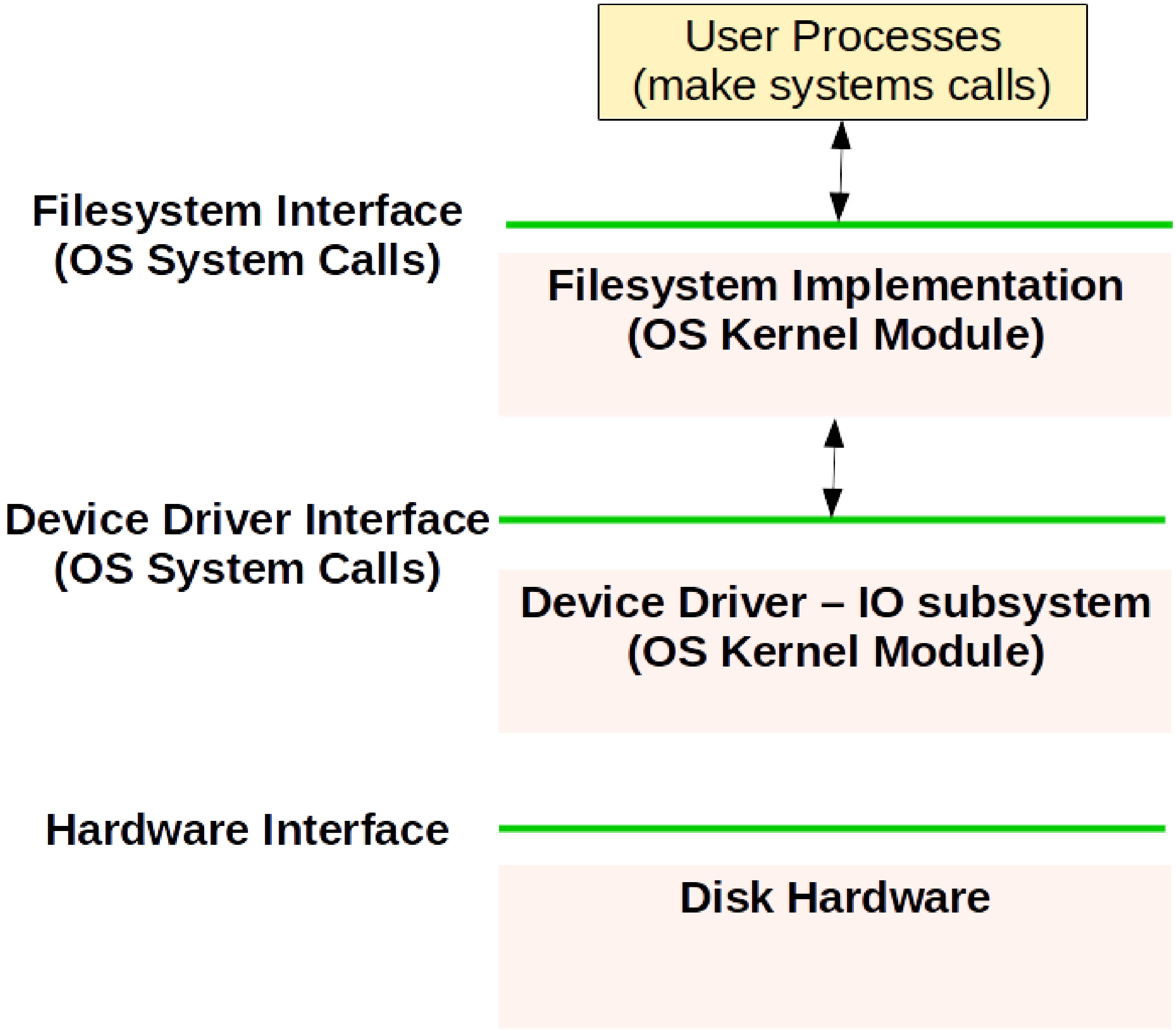
Design and Implement a filessytem

This assignment is to implement a `file system' as per our own design on a file provided by the underlying operating system. Henceforth this file system will be called as **myfs**. So, now we have two types of files. Files which are kept in **myfs** are called **myfile**. The OS on which we shall be working also support files. These files will be called **OS files**.

Let us discuss the motivation behind this assignment.

We are familiar with the following layered comprehension of file system implementation as show in the following figure!

Filesystem as Layers



User Processes
(make systems calls)

Filesystem Interface
(OS System Calls)

Filesystem Implementation
(OS Kernel Module)

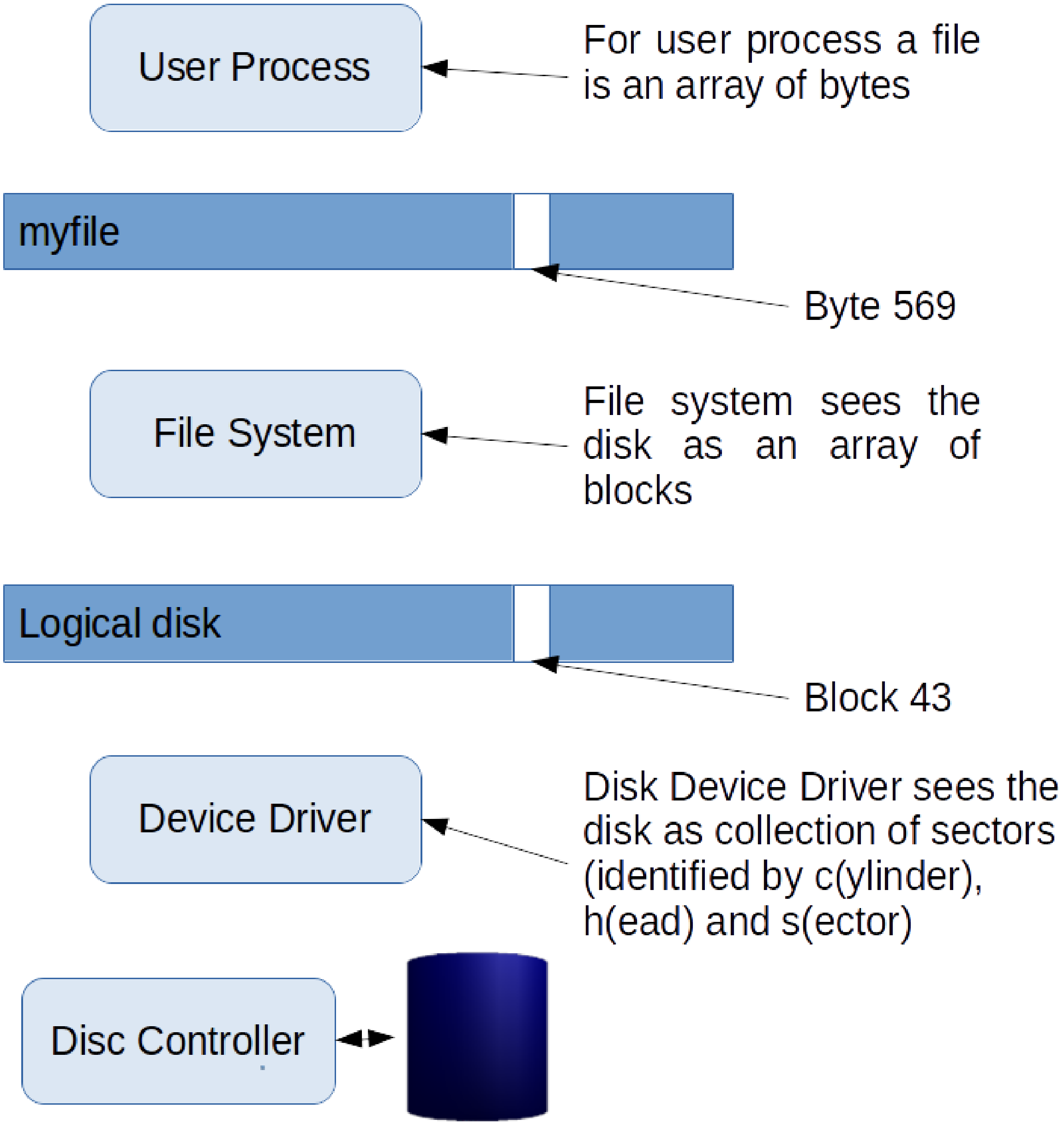
Device Driver Interface
(OS System Calls)

Device Driver – IO subsystem
(OS Kernel Module)

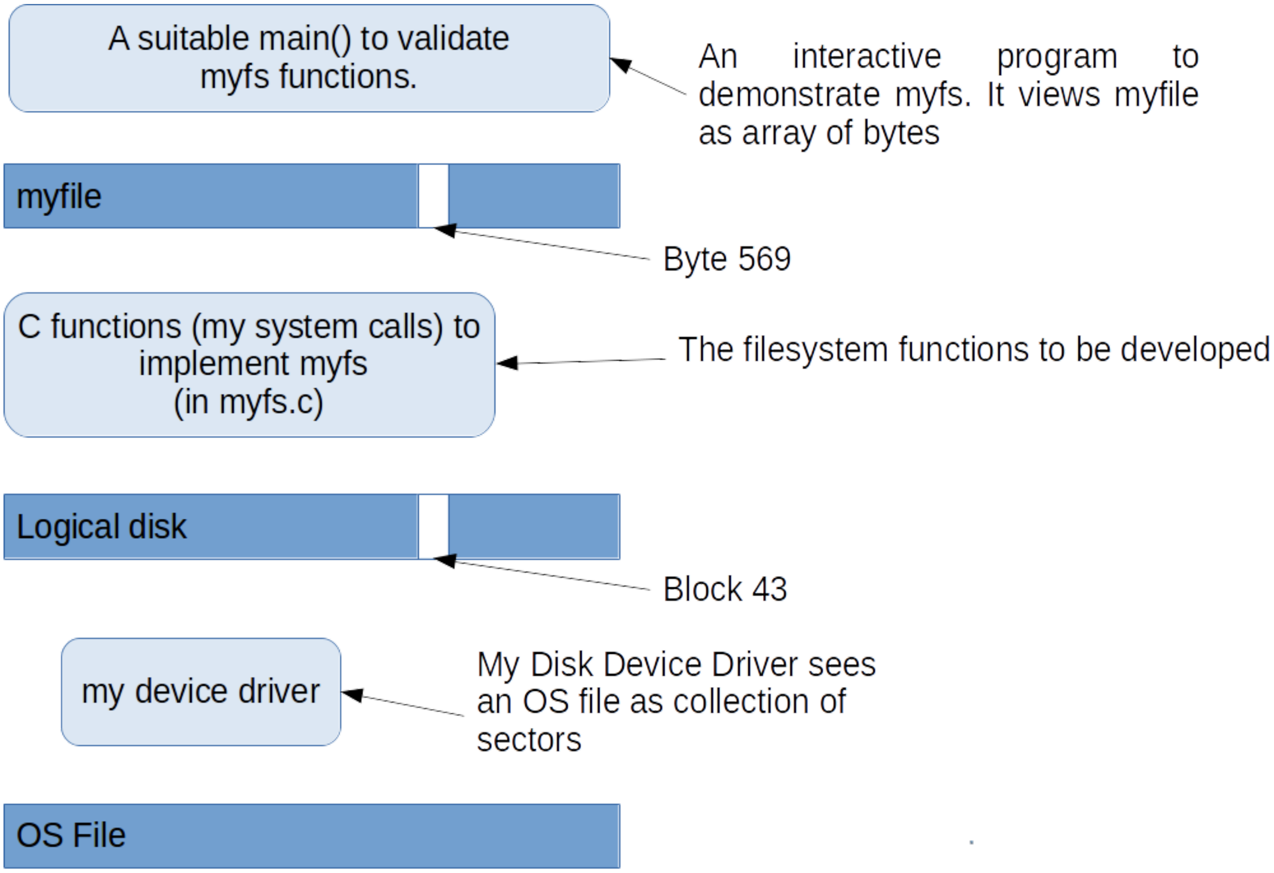
Hardware Interface

Disk Hardware

The following figure shows how a file system is viewed from different layers.


User Process
For user process a file is an array of bytes
myfile
Byte 569
File System
File system sees the disk as an array of blocks
Logical disk
Block 43
Device Driver
Disk Device Driver sees the disk as collection of sectors (identified by c(ylinder), h(ead) and s(ector)
Disc Controller

Since, **myfs** will be created on an **OS File**, for us, the following diagram depicts the situation more closely.


A suitable main() to validate myfs functions.
An interactive program to demonstrate myfs. It views myfile as array of bytes
myfile
Byte 569
C functions (my system calls) to implement myfs (in myfs.c)
The filesystem functions to be developed
Logical disk
Block 43
my device driver
My Disk Device Driver sees an OS file as collection of sectors
OS File

Now let us look at the implementation of **myfs**. Implementation of a File System (**myfs**) essentially means implementation of the File System related system calls. We should first then see what are the system calls Linux provides for File System related activities (**man syscalls** gives the huge list of all system calls. Filter it or use other sources). Please note that though the functions we will be implementing for **myfs** should be as close as possible to these system calls, we have the freedom to customize their functionality. We shall call these functions to **my system calls**. To check that **myfs** works as desired we shall provide an interactive **main()** function which will accept "command" from the user and in turn will call "**my system calls**" to perform those commands. A typical session with this main() function is given below.

1. $myfs /* execute your program */
2. myfs> /* prompt given by your program */
3. myfs>mkfs osfile1 512 10MB /* creates your filesystem on file osfile1, blocksize is taken to be 512B */
4. myfs>mkfs osfile2 1024 20MB /* creates your filesystem on file osfile2, blocksize is taken to be 1024B */
5. myfs>use osfile1 as C: /* the filesystem on osfile1 will henceforth be accessed as C: */
6. myfs>use osfile2 as D: /* the filesystem on osfile2 will henceforth be accessed as D: */
7. myfs>cp osfile3 C:tesfile1 /* copy the file osfile3 from os to the filesystem C: as testfile1 */

8. myfs>ls C: /* see the contents of the filesystem C: */
   testfile1 ... size

9. myfs>cp C:tesfile1 C:testfile1a /* copy the file testfile1 from C: to the filesystem C: as testfile1a */

10. myfs>ls C: /* see the contents of the filesystem C: */
    testfile1 ... size
    testfile1a ... size

11. myfs>cp C:tesfile1 D:testfile2 /* copy the file testfile1 from C: to the filesystem D: as testfile2 */

12. myfs>cp D:testfile2 osfile4 /* copy the file testfile2 from C: to the to the OS as osfile4 */

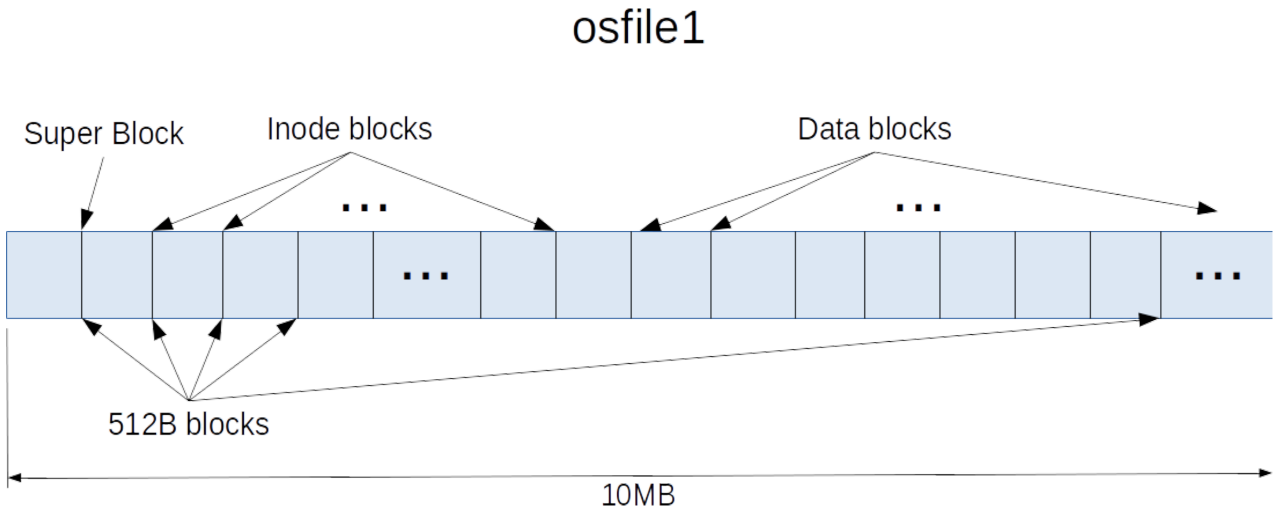
13. myfs>rm C:testfile1 /* Delete the testfile1 from C: */

14. myfs>ls C: /* see the contents of the filesystem C: */
    testfile1a ... size

15. myfs>mv D:testfile2 D:testfile2a /* Rename testfile2 of D: to testfile2a in D: */

16. myfs>exit /* exit from your program */

Let us take up the first and most important command - "**mkfs**" - make file system. This is equivalent to the "**format**" command that we use for pen drive, memory card, etc. For us, "**mkfs**" creates an empty file system on an **OS file**. The systax of this command is "**mkfs <OS File Name> <block size> <myfs size>**". For example, "**mkfs osfile1 512 10MB**" creates **myfs** on **OS File osfile1** and block size is taken to be **512B**. Pictorially **osfile1** now can be conceived as



So the decisions we have to take for **mkfs** is:

1. What will be my **superblock** size (what we should keep in the superblock)
2. What will be the size of each **inode** (what do we keep in the **inode**)meta
3. The data blocks for a folder contains the meta-data of its subfolder or files. We have to decide what meta-data we shall keep. Please note that "**mkfs**" creates the **root folde**r of **myfs**.
4. ....
5. ....

6. ....

The other commands used in the above session are self-explanatory. Make your own assumptions wherever necessary.

Try to prepare the overall design of **myfs** and associated **my system calls** and submit the same at the end of the class.

Submission status

Submission status	No attempt
Grading status	Not graded
Due date	Tuesday, 20 February 2018, 4:35 PM
Time remaining	2 hours 33 mins
Last modified	-
Submission comments	⊞ Comments (0)

Add submission

Make changes to your submission

You are logged in as Avinash Prasad (Log out)
OSLAB_GxJan2018