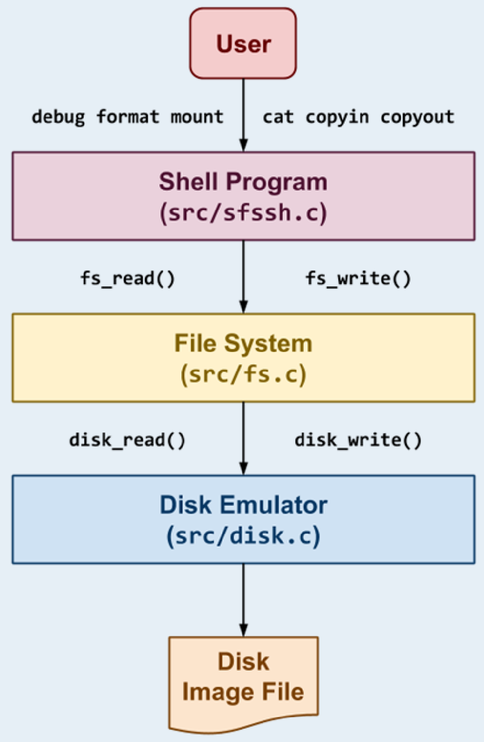
SIMPLE FILE SYSTEM

HANISHA R
1RV22CS244

OPERATING SYSTEMS
FACULTY: DR JYOTHI SHETTY

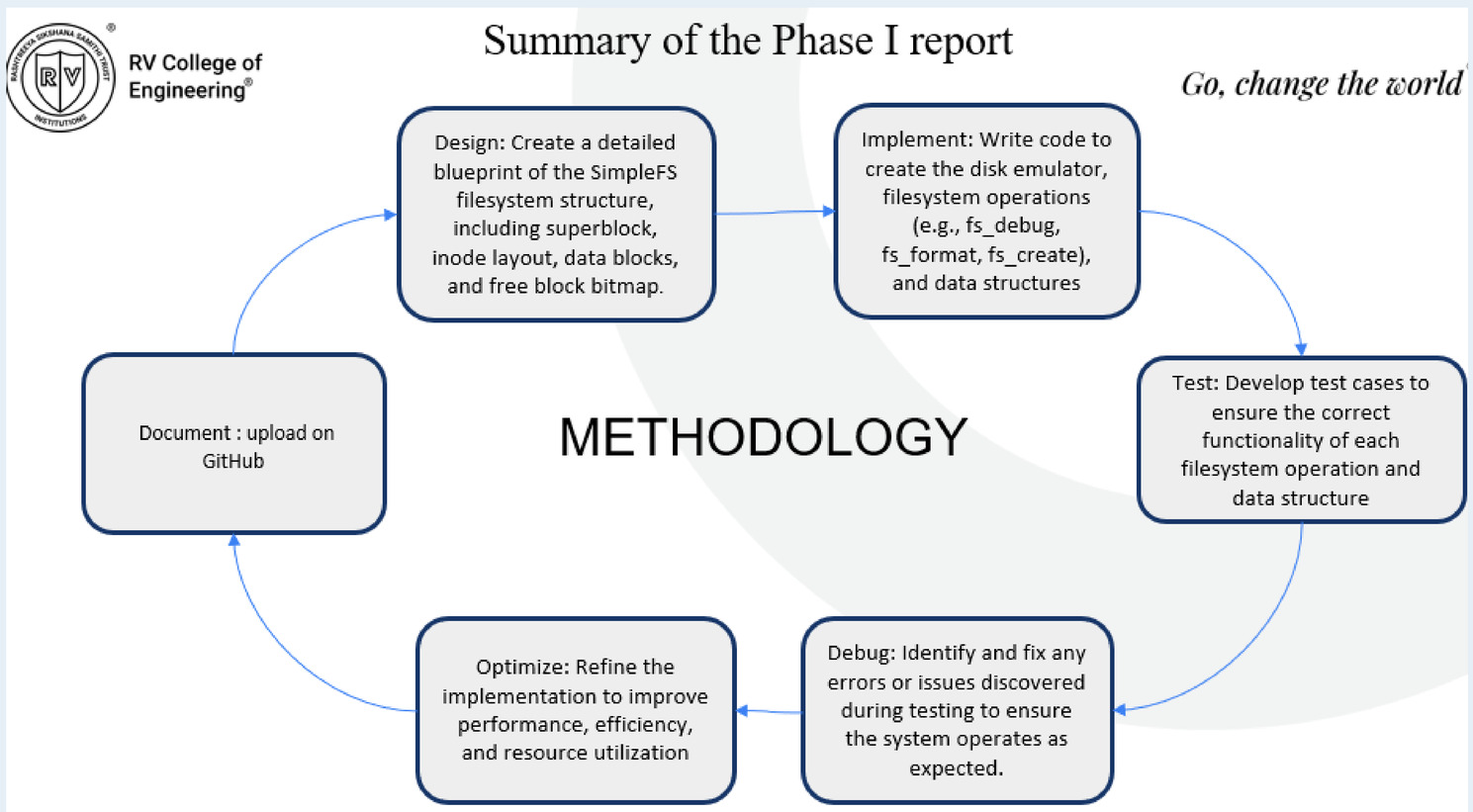


INTRODUCTION

Create SimpleFS, a simplified Unix-like File System, consisting of a shell application, file system component, and disk emulator. Develop in C to manage on-disk data structures, handle disk operations, and provide a user-friendly interface for filesystem tasks. Deliverables include source code, documentation, and a demonstration video, aiming to deepen understanding of file systems and low-level system programming.

METHODOLOGY

1. Save the filesystem assembly code in a file named fs.c
2. Save the shell interface code in a file named shell.c.
3. Open a terminal or command prompt.
4. Navigate to the directory containing the files.
5. Compile the bootloader assembly code using an assembler make
6. Run the code and implement the various function calls associated with the file operations



OUTPUTS

```
hanisha@hanisha:~$ cd os
hanisha@hanisha:~/os$ make
make: 'simplefs' is up to date.
hanisha@hanisha:~/os$ ./simplefs el 1000
opened emulated disk image el with 1000 blocks
 simplefs> format
disk formatted.
 simplefs> mount
disk mounted.
 simplefs> debug
superblock:
    magic number is valid
    1000 blocks total on disk
    100 blocks dedicated to inode table on disk
    12800 total spots in inode table
 simplefs> create
created inode 1
 simplefs> create
created inode 2
 simplefs> delete 2
inode 2 deleted.
 simplefs> cat 1
0 bytes copied
 simplefs> copyin hello.txt 1
33 bytes copied
copied file hello.txt to inode 1
 simplefs> copyout 1 hello.txt
33 bytes copied
copied inode 1 to file hello.txt
 simplefs> getsize 1
inode 1 has size 33
 simplefs> defrag
disk defragged.
```

```
inode 1 has size 33
 simplefs> defrag
disk defragged.
 simplefs> debug
superblock:
    magic number is valid
    1000 blocks total on disk
    100 blocks dedicated to inode table on disk
    12800 total spots in inode table
inode 1:
    size: 33 bytes
    direct data blocks: 101
```

```
 simplefs> exit
closing emulated disk.
528 disk block reads
1105 disk block writes
hanisha@hanisha:~/os$ make clean
rm simplefs disk.o fs.o shell.o
```

SYSTEM CALLS

fopen: Opens a file for reading or writing.
fclose: Closes an open file descriptor.
fread: Reads data from a file.
fwrite: Writes data to a file.
ftruncate: Truncates a file to a specified length.
fseek: Moves the file pointer to a specified position within a file.

CONCLUSION

In summary, the creation of SimpleFS involved a systematic approach, from requirements analysis to deployment. Through meticulous design, implementation, testing, and documentation, we developed a functional file system solution. Optimization techniques were applied for efficiency, and ongoing maintenance ensures its longevity. SimpleFS stands as a testament to our commitment to quality software engineering.

ACKNOWLEDGEMENTS

We express our sincere gratitude to the Principal,Dean,HOD and teacher for their kind support
received for the completion of the project