**Kamala Education Society’s**

**Pratibha Institute of Business Management**

**Chinchwad, Pune-411019**

**2021-2022**

**SYNOPSIS REPORT**

**ON**

**File System Development**

**BY**

**Jay Padmakar Bhavsar**

**Course: MCA II**

**Roll No.:21041**

**Guide Name: Mr. Shubham Nagure**

**INDEX**

1. **Introduction to project**
   1. **Existing System & It’s Drawbacks**
   2. **Purpose(need) of System**
2. **Proposed System** 
   1. **Problem statement/s**
   2. **Objectives of proposed system**
   3. **Project Overview. (Functionality of Modules)**
   4. **Functional requirements**
   5. **Nonfunctional requirements**
   6. **Scope of the system**
3. **System Analysis & Design**

**3.1 On Disk File System Organization**

**3.2 Overview of the File System**

**3.3 Flow Chart**

1. **Hardware & Software Requirements**
2. **Future scope of project**
3. **Conclusion**
4. **Introduction to project**

**1.1Existing System & It’s Drawbacks**

Open-source local file systems, such as Linux Ext4 ,XFS, and Btrfs ,remain a critical component in the world of modern storage.

For example, many recent distributed file systems, such as Google GFS and

Hadoop DFS, all replicate data objects (and associated metadata) across local file systems. On smart phones, most user data is managed by a local file system; for example, Google Android phones use Ext4 and Apple’s iOS devices use HFSX. Finally, many desktop users still do not backup their data regularly in this case, the local file system clearly plays a critical role as sole manager of user data.

There are few drawbacks like Power Management, not every hardware will be compatible - older hardware may not have the drivers and some little used hardware may not have also, or have fewer resources compared to what it have in other OS;

not every software and webAPPS are compatible - the software part is almost non-existant nowadays (except for games, Linux still have only a few games), but webAPPS (the ones you use in a browser) may not work, specially those that use much security like the ones used by banks.

**1.2Purpose(need) of System**

Our aim is to be to create a simple fs so that new developers can easily onboard to the project and get understanding of FS easily.

In order to creatively think the new ideas to build it differently with more options to explore.

**2.Proposed System**

**2.1Problem statement/s**

To understand the existing File System which are stable in market to solve the issues regrading file/data accessing accuracy, Data Redundancy , Data Security etc

**2.2 Objectives of proposed system**

Our objective is to:

* to make FS which can manage memory efficiently.
* Job schedule -Round Robin Scheduling.
* Files Handling-Reading file, Writing Files and Sharing files.
* Memory management technique-Fragments, Swapping.

**2.3 Project Overview. (Functionality of Modules)**

To build an File System we used Progamming Language- C

Debugger- GCC

1) fs.h -

* It is an Header File to declare the Structure of the File System.
* It Consist of Structure of SuperBlock, Inode, Disk Block
* Function declaration to access the File create,allocate,read,write Files
* Function declaration to create, mount, synchronize File System.

2) fs.c-

* It consist of the actual implementation or body of the Function which are declare in header file fs.h
* Where we code the logic to for File system Structure and File Structure and its access methods.

3) test\_file.c: It is an driver code.

**2.4 Functional requirements**

* File creation: The file system should allow users to create new files in the file system, specifying the file name and initial contents if necessary.
* File deletion: The file system should allow users to delete files from the file system.
* File modification: The file system should allow users to modify the contents of existing files.
* File reading: The file system should allow users to read the contents of files in the file system.
* Directory creation: The file system should allow users to create directories to organize files within the file system.
* Directory deletion: The file system should allow users to delete directories from the file system.
* Directory modification: The file system should allow users to rename directories, move files between directories, and modify directory permissions.
* Metadata management: The file system should manage file and directory metadata, such as file size, creation date, permissions, and ownership.
* File system consistency: The file system should ensure the consistency of the file system data structures to prevent data corruption or loss.
* File system security: The file system should provide mechanisms for ensuring file system security, such as access control lists and encryption.
* File system performance: The file system should provide efficient access to files and directories, with minimal overhead and latency.
* File system scalability: The file system should be scalable to support large numbers of files and directories, and to handle increasing workload as usage grows.

**2.5 Nonfunctional requirements**

* Performance: The file system should provide efficient access to files and directories, with fast read and write speeds, low latency, and minimal overhead.
* Scalability: The file system should be able to handle increasing workload as usage grows, and should be able to support large numbers of files and directories.
* Reliability: The file system should be reliable, with a low probability of data loss or corruption, and should be able to recover from errors or failures.
* Availability: The file system should be available and accessible to users at all times, with minimal downtime or service disruptions.
* Security: The file system should provide mechanisms for ensuring data security, such as access control lists, encryption, and auditing.
* Compatibility: The file system should be compatible with existing operating systems, file systems, and storage devices.
* Usability: The file system should be easy to use and understand, with a simple and intuitive user interface, and clear documentation and support.
* Maintainability: The file system should be easy to maintain and update, with well-documented code, clear error messages, and easy debugging and testing tools.
* Portability: The file system should be portable across different hardware and software platforms, and should be able to run on a variety of devices and architectures.

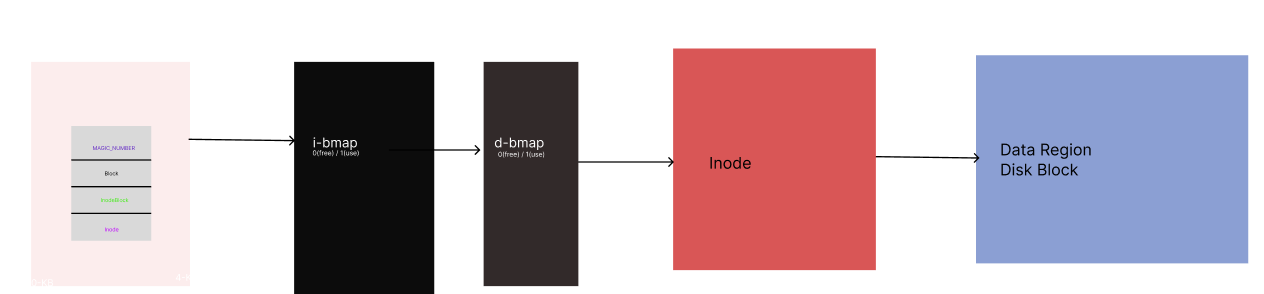
**2.6 Scope of the system**

The scope of a file system refers to the range of functionality and features that the system is designed to support. The scope of a file system will depend on the specific use case and requirements of the system.

* Local File System
* Distributed FS
* Cloud FS

**3.System Analysis & Design**

**1.On disk File System Organization**

****

**Diagram

Description automatically generated with low confidence**

1. **Overview of File System Code**

**Graphical user interface, application, Word

Description automatically generated**

**3.Flow Chart**

**Chart

Description automatically generated**

**4.Hardware & Software Requirements**

We implement an ***Container*** Concept of ***Docker*** to code in container, where we installed the required software tools like cmake, C etc.

To test out the File system we create different Container Image based on different hardware and software architecture of the various System.

To use our File system user need to mount file system to their os and can perform different tasks.

**5.Future scope of project**

As per study we develop and File System code that do basic functional operations with proper algorithm for each operation

We are going to read the File with help of suitable Algorithm to access the file data

Work on Security of the file how we can secure the Files in File System

Work on the Kernel Development

**6.Conclusion**

A file system is an essential component of any computer or computing system, responsible for managing files and directories and providing a structured way to store and access data. A file system provides an abstraction layer that hides the details of storage devices and makes it easy for users and applications to access and manage data.

File systems have evolved over time to support various use cases and requirements, from local file systems that manage files on a single computer or device, to distributed file systems that manage files across multiple devices and networks. The scope of a file system depends on the specific requirements of the system and the needs of the users, and may include features such as performance, reliability, scalability, security, and usability.

Designing and implementing a file system requires careful consideration of both functional and non-functional requirements, as well as an understanding of the underlying storage devices and operating system. With the rapid growth of data and computing needs, file systems continue to play a crucial role in managing data and providing efficient and reliable access to information.