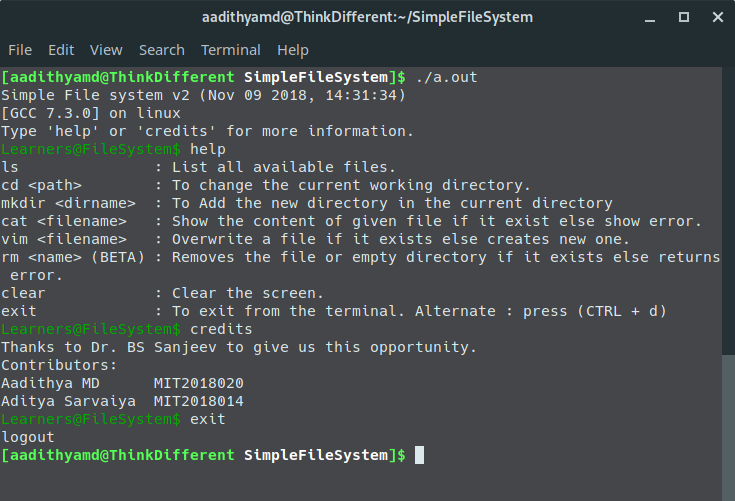
Simple File System

Group : Learners



**Aadithya MD (MIT2018020)**

**Aditya Sarvaiya (MIT2018014)**

13.11.2018

PPR Project

# 

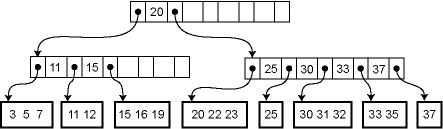
# INTRODUCTION

It’s a simple file system made using BTree. A BTree is a tree data structure that keeps data sorted and allows searches, sequential access, insertion, and deletion in logarithmic time. A BTree is a generalization of a binary search tree in a way that a node may have more than two children depending on its degree.

**Why BTree?**

There are many data structures available which we can use. Then why BTree? What is so special about it? Answer to this question can be justified by the following points:

* As BTree is self balancing tree, it takes logarithmic time (O(log n) or O(h) to be precise) for most of the operations.
* Disk operations are costly so, it is desirable to minimise it. Since BTree reduces the no of disk access required by reducing the height of the tree, it is preferred.
* Deleting or Inserting a record from/to a table with a million records or more in it could be an expensive operation if the table has to be completely rewritten. If sequential access to the underlying table is handled through the B-Tree or if the entire file is stored in the nodes of the B-Tree, deletion or insertion of a row or record in the table gets much simpler.



1.1 Simple BTree structure

# 

# STAGES

# 

# Stage 1 : Disk Library

Designing a simple disk library that reads and writes disk blocks. Our file system will be built on top of this interface.

The disk interface is as follows:



1.2 Disk Interface

The calls return an error if the underlying Unix system calls fail.

**A disk block freelist.** This is a simple data structure (for instance, a bitmap) that tracks what blocks on the disk have been allocated. This data structure is, for obvious reasons, persistent (i.e., on disk).

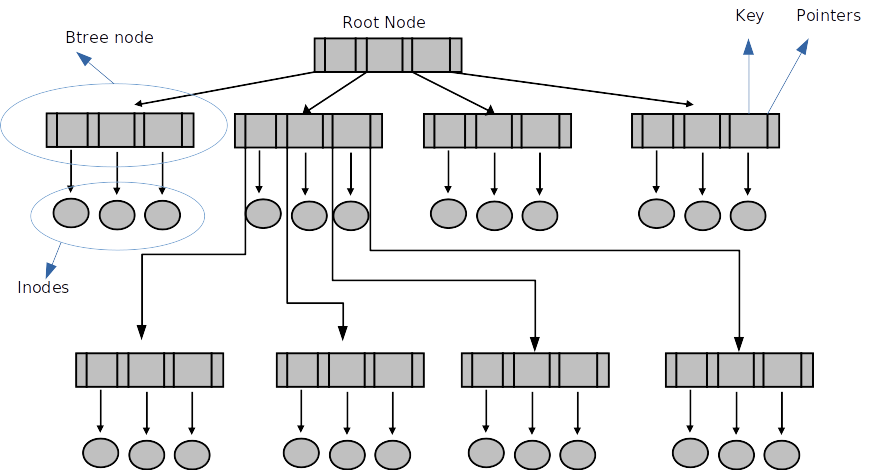
# 

1.3 Disk Block Freelist

# 

# Stage 2 : BTree Implementation

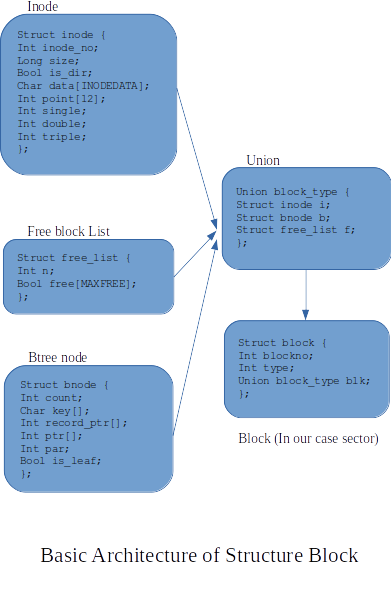
**BTree.** It is a fat height balanced search tree. BTree achieves significant reduction in number of disk accesses compared to other data structures by height reduction.



**Inodes.** An inode is persistent memory that contains pointers to disk blocks. Inodes are used to map a file to the disk representation of a file: each file has its own inode, the inode has pointers to all disk blocks that make up that file.

|  |
| --- |
| inode\_no |
| size |
| is\_dir |
| hash |
| data |
| point |
| single |
| double |

1.4 Inode Structure



# Stage 3 : Command Line Interface

|  |  |
| --- | --- |
| Function name and Argument | Description |
| help | Overview of all the available commands |
| ls | List all available files in that directory |
| cd <path> | To change the current working directory |
| mkdir <dirname> | To Add the new directory in the current directory |
| cat <filename> | Show the content of given file if it exist else show error |
| vim <filename> | Overwrite a file if it exists else creates new one |
| **(BETA)**  rm <filename>  rm <dirname> | Removes the given file or empty directory if it exists else gives an error |
| clear | Clear the terminal screen |
| exit | To exit from the terminal. Alternate : press (CTRL + d) |

# 

Link to code : <https://github.com/aadithyamd/SimpleFileSystem>

# 

# REFERENCES

1. **Book - “Introduction to algorithms” by Cormen, Leiserson, Rivest and Stein**
2. **MIT lab assignment on a file system:**

<http://web.mit.edu/6.033/1997/handouts/html/04sfs.html>

1. **Stack overflow :**

<https://stackoverflow.com/questions/4714056/how-to-implement-a-very-simple-filesystem>

1. **University of Illinois at Chicago:**

<https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/11_FileSystemImplementation.html>