Documentation

IPFS

December 14, 2023

Software Documentation for Interplanetary Filing system created by students from the third semester of FAST Nuces – Islamabad Campus

By

Muhammad Ali Naveed

Hamza Omer

Ariyan Chaudhary

Interplanetary Filing System (IPFS)

Documentation

This documentation's purpose is to familiarize the user with the workings of all the functions inside the filing system. There are a few prerequisites that the user needs to install before running this project.

Installation Prerequisites

To run the Interplanetary Filing system, these two packages must be installed on your computer.

**Grpahviz**

Click on this link to download [GraphViz](https://graphviz.org/download/)

**OpenSSL (sha1)**

Click on this link to download [OPENSSL](https://slproweb.com/products/Win32OpenSSL.html)

Use version C++17 or later for Filesystem Library to Function

Classes

|  |  |  |  |
| --- | --- | --- | --- |
| **Unified Functions Operations (UFO)** | | | |
| **Function name** | parameters | return value | complexity |
| //takes a hexadecimal number in string, multiplies it by 2 and returns the result.  static std::string hexMultiplyTwo(std::string hexNum) |  |  |  |
| //takes a char value in hex and returns it in decimal int  static int hexCharToInt(char num) |  |  |  |
| //takes decimal int and returns its hex value in char type  static char intToHexChar(int num) |  |  |  |
| //takes identifier space bits as input, and sets the size for future calculations with UFO class  static void setSize(int space) |  |  |  |
| //takes two strings containing hexadecimal number and returns their sum stored in string  static std::string addHex(std::string firstHex, std::string secondHex) |  |  |  |
| //takes integer i returns a string which stores hex value of 2^i  static std::string hexPower\_i(int i) |  |  |  |
| //compares two hex numbers.  static bool isEqual(std::string hexOne, std::string hexTwo) |  |  |  |
| //subtracts second parameter from the first parameter, both of which are hexadecimal numbers in string.  //second parameter cannot be greater than first parameter.  static std::string subtract(std::string hexOne, std::string hexTwo) |  |  |  |
| //can only be called after UFO::setSize() function has been called before  //takes hash generated from sha1 and truncates it according to digitSize  static std::string truncateHash(std::string hexNum) |  |  |  |
| //takes a path stored in string. wherever it finds '\' , it adds another one : D  static std::string addSlashFilePath(std::string path) |  |  |  |
| //takes path in parameters, and extracts the last part of the path which is the file name with its extension  static std::string extractNameFromPath(std::string path) |  |  |  |
| //takes file path in parameter, and remove its extension. only catering .txt :(  static std::string removeNameExtension(std::string path) |  |  |  |
| //takes filePath and checks if a file exists at that path with the same name  //if file of that name already exists, it modifies the name appropriately, and returns the path.  static std::string fixFilePath(std::string path) |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **RFT (Routing Table)** | | | |
| **Function name** | parameters | return value | complexity |
| //Deletes RFT from Memory  bool deleteRFTNode(MachineNode\* deleteMachine) | MachineNode\* | Bool | O(N[[1]](#footnote-1)) |
| //Inserts a new Node inside the Routing Table  void insertRFTNode(MachineNode\* newMachine, int entryNumber) | MachineNode\* | void | O(N) |

|  |  |  |  |
| --- | --- | --- | --- |
| **BTreeNode** | | | |
| **Function name** | parameters | return value | complexity |
| //Inserts a new Element inside the BTree  BTreeNode\* insert(std::string val, std::string newpath, BTreeNode\* root) | BTreeNode\* root and Values | Recursively returns Nodes | O(H)[[2]](#footnote-2) |
| //Helper! Changes the values inside the Node while Insertion  int setval(std::string val, std::string newpath, BTreeNode\* n, std::string\* p, std::string\* pp, BTreeNode\*\* c) |  |  |  |
| //Works along with setval to shift around values during insertion  void fillnode(std::string val, std::string newpath, BTreeNode\* c, BTreeNode\* n, int k) |  |  |  |
| //Split Functionality when Inserting Data. Number of elements in Node > Max Keys  void split(std::string val, std::string newpath, BTreeNode\* c, BTreeNode\* n, int k, std::string\* y, std::string\* yy, BTreeNode\*\* newNode) |  |  |  |
| //Returns 1 if val is found and 0 Otherwise  int searchnode(std::string val, BTreeNode\* n, int\* pos) |  |  |  |
| //Returns the Node which contains the element being searched  BTreeNode\* search(std::string val, BTreeNode\* root, int\* pos) |  |  | O(log(Z))[[3]](#footnote-3) |
| //Recursively Moves to the Node to be deleted  int delhelp(std::string val, BTreeNode\* root) |  |  |  |
| //Copies the Successor into the node to be deleted  void copysucc(BTreeNode\* node, int i) |  |  |  |
| //Deletes the Node from memory by overwriting its value  void clear(BTreeNode\* node, int k) |  |  |  |
| //Initiated when deletion leads to unordinary case  void restore(BTreeNode\* node, int i) |  |  |  |
| //RightShift an entry from Sibling node since one of the sibling has keys < MIN  void rightshift(BTreeNode\* node, int k) |  |  |  |
| //Same as rightshift but in opposite Direction  void leftshift(BTreeNode\* node, int k) |  |  |  |
| //If shifting cannot be performed after deletion, then siblings are merged  void merge(BTreeNode\* node, int k) |  |  |  |
| //Deletion Call initiated through this  BTreeNode\* del(std::string val, BTreeNode\* root) | Hex Key and Root | Recursively returns Node | O(log(Z)) |
| //Graphviz PNG of BTree is Generated Using this  void generateDot(std::ofstream& dotFile, int& nodeCounter, bool isPath = false) |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **BTree** | | | |
| **Function name** | parameters | return value | complexity |
| //Insert a newNode  void insert(std::string key, std::string path) | Hex Key and file Path | void | O(H) |
| //Delete an Existing Node  void deleteNode(std::string key) | Hex Key | void | O(H) |
| //Returns the Current Smallest Node in the BTree  BTreeNode\* smallestNode() |  |  | O(log(Z)) |
| //Returns the Current Largest Node in the BTree  BTreeNode\* greatestNode() |  |  | O(log(Z)) |
| //Returns the path of the Hash being Searched  std::string search(std::string fileKey) |  |  | O(log(Z)) |
| //Creates PNG of BTree of a Particular Machine. Call This Function from BT of any Machine  void generateBTreeDotFile() |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **MachineNode** | | | |
| **Function name** | parameters | return value | complexity |
| //Creates the Routing Tables of all the Machines. Called After A Machine is Added or Removed  void reconstructRFT(MachineNode\* head) | Machine Head (DHT) |  | O(M\*N\*log(N))[[4]](#footnote-4) |
| //Return the Machine in the DHT that is at or next to p(HEX ID)  MachineNode\* findMachineSuccessor(MachineNode\* head, std::string p) |  | MachineNode | O(log(N)) |
| //Adds an RFT when reconstructRFTS is called and allocates memory for RFTNode  void addRFTNodeToMachine(RFTNode\*& FTHead, MachineNode\* newMachine, int entryNumber) |  |  |  |
| //Searches for the Machine that Comes after the hashValue given to the Function in the DHT  MachineNode\* search(std::string hashValue) | Hex Key | MachineNode | O(log(N)) |
| //Searches for the File in the DHT and Prints the File Path on Console  void fileSearch(std::string fileKey) | Hex Key | Void | O(log(N)) |
| //Removes the File from the BTree of the Machine that contains it and also from Hard Drive  bool fileRemove(std::string key, bool reassigning = 0) | Hex Key | bool | O(log(N)) |
| //Changes the FilePath when a file is being shifted to a new Machine  void createFileInNewDirectory(MachineNode\* currentNode, std::string oldPath, std::string Key, bool reassigning = 0) |  |  |  |
| // Counts the Number of Duplicates of a certain Hash  int countDuplicates(std::string filePath) |  |  |  |
| //Extracts the File Path amongst the many from a string for Duplicates  std::string extract(std::string dupFilePath, int numFile) |  |  |  |
| //is called when a new machine is added, pulls its own files from its next machine.  void pullFiles() |  |  | O(F)[[5]](#footnote-5) |
| //is called when a new machine is deleted, push its own files to the next machine before its deleted  void pushFiles() |  |  | O(F) |
| //Creates the GraphViz for RFT  void generateDotRFT() |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Distributed Hash Table (DHT)** | | | |
| **Function name** | parameters | return value | complexity |
| //Function that takes a String and returns sha1 of it in a string  std::string sha1(const std::string& input) |  |  |  |
| //Creates the Folder for a Machine  bool createDirectory(std::string machineName, std::string machineID) |  |  |  |
| //Removes the Folder for a Machine  bool removeDirectory(std::string machineName, std::string machineID) |  |  |  |
| //only deletes the folders created in the .\Machines  void deleteAllDirectory() |  |  |  |
| //Creates the Directory and Reassignes the Files when a New Machine is Added  bool machineAdd(std::string Mname) | Machine Name | Bool | O(M+F) |
| //Removes a Machine from DHT and Reassinges Files Accordingly  bool machineRemove(std::string mName) | Machine Name | Bool | O(M+F) |
| //Updates all the Routing Tables  void reconstructRFTs() |  |  |  |
| //Locates the Machine to Add the File and Adds it in its BTree  bool fileAdd(std::string filePath, std::string machineName = "", std::string manualID = "") | Machine From where Query Generates[[6]](#footnote-6) |  | O(log(M)+log(Z)) |
| //called when a machine is added to pull files from the next machine  void reassignFilesAdd(MachineNode\* newlyAdded) |  |  |  |
| //called when machine is deleted to push machines to the next machine.  void reassignFilesDel(MachineNode\* newlyAdded) |  |  |  |
| //Searches for a File in the DHT  void fileSearch(std::string fileKey, std::string machineName = "") |  |  | O(log(M)+log(Z)) |
| //Locates the Machine which Contains the File and Removes it  void fileRemove(std::string fileKey, std::string machineName = "") |  |  | O(log(M)+log(Z)) |
| //Generates the GraphViz for DHT  void generateDot() |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interplanetary Filing System (IPFS)** | | | |
| **Function name** | parameters | return value | complexity |
| // Displays the user interface for all operations in the IPFS. Generates the function calls of the other classes.  void displayMenu() |  |  |  |
| //Asks for the Option and Performs the Required Operation  void interface() |  |  |  |
| //Takes User input for File add and Calls DHT FileAdd  void fileAdd() |  |  |  |
| //Takes User Input and Adds Machine in DHT  void machineAdd() |  |  |  |
| //Takes user Input and Removes Machine  void machineRemove() |  |  |  |
| //Searches for the File using routing Tables  void fileSearch() |  |  |  |
| //Removes file from BTree and Directory if Exists  void fileRemove() |  |  |  |
| //Generates GraphViz of BTree  void printBTree() |  |  |  |
| //Generates GraphViz of RFT  void printRoutingTable() |  |  |  |
| //Generates GraphViz of DHT  void printDHT() |  |  |  |

1. N is the identifier Space that the User Entered [↑](#footnote-ref-1)
2. H is the Height of the Tree [↑](#footnote-ref-2)
3. Z is the number of entries in the BTree [↑](#footnote-ref-3)
4. M is the Number of Machines in the DHT [↑](#footnote-ref-4)
5. F is the Number of Files in IPFS [↑](#footnote-ref-5)
6. In FileAdd, FileRemove, and Search, we ask the user the machine from where he would like to start the Query [↑](#footnote-ref-6)