You are required to submit a detailed document that contains the following

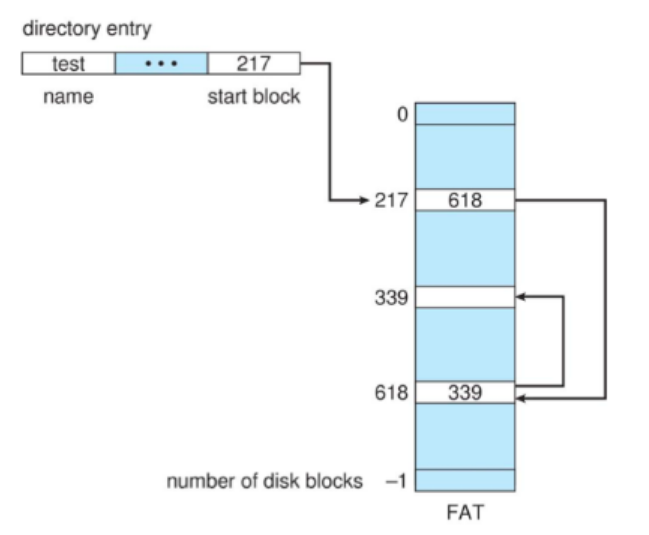
a. System design: A complete detail of your system design. In addition to the written description, you may use figures and tables to illustrate your design and its working.

1. Tree structure directory using nested dictionaries in python

Diagram

Description automatically generated

1. for keeping track of blocks assigned to a file.



1. for keeping track of unoccupied blocks.

Text

Description automatically generated

**False**

**True**

b. Choice of data structure: Which of the data structure is used in your operating system design e.g., link list, json or any other method you may have used and why?

The directory structure can be viewed as a tree having multiple directory levels. Implemented file system structure by using dictionaries and nested dictionaries in python. Dictionary data structure in python is an implementation of hash table. The hash tables are used as they have constant average case complexity for insertion, deletion and searching.

c. Memory map: A detail on how your data is organized in file and folders. Information regarding size of your system i.e., memory size and number of pages supported by the system and how your memory map handles and displays every information.

Memory map shows the distribution of files in the memory. It displays the file name including size, starting address and blocks list. The indentation indicates the hierarchy in the directory structure i.e which folder or file is present where. This functionality is achieved using recursion.

Number of blocks = 32

Block size = 16 bytes

Memory size = number of blocks \* blocks size = 512 bytes

These sizes were chosen to test the system functionalities.

d. System functionalities: A detail on the features that your system supports and how to run

your code. Also add python version and dependencies in this section if any,

When the program starts, the file system structure is loaded from the .dat file.

Functionalities we implemented are:

1. Create a file

The files created are maintained in a dictionary named dir\_str.

2. Delete a file

Remove the file from file structure. The actual content of file is also deleted and the status of the blocks is updated.

3. Make directory

The directories created are maintained in a dictionary named dir\_str.

4. Change directory

Three options available:

* Name of directory present in current directory
* “..” => to go to previous directory
* “~” => to go to root directory

5. Move a file

Move a file from source destination to the specified destination from the user. If destination have the same name file then the user is prompted to rename the file.

6. Open a file

Open the file from the current directory and store the copy of file control block from directory structure in open file table.

7. Close a file

It closes the opened file and removes it from open file table.

8. Write in a file

Writes to a file in append mode and assigns the block using File Allocation Table (FAT) technique. File must be opened in w (write) or rw (read/write) mode before writing.

9. Write in a file at specific position

Writes to a specific point in the file by overwriting data at the location specified. File must be opened in w (write) or rw (read/write) mode before writing.

10. Read a file

Reads from the first word and returns the entire content. File must be opened in r (read) or rw (read/write) mode before reading.

11. Read a file from specific position

Reads from the start memory location for size number of characters. File must be opened in r (read) or rw (read/write) mode before reading.

12. Move text in a file

This allows the user to move data forward or backward. The input should be of the form: Move(Filename, start, to, size) where filename is the name of the file, start is the starting location of data to be moved, to is the location where the data should be placed and size is the size of the data to be moved. File must be opened in rw (read/write) mode before moving.

13. Truncate text in a file

It reduces the size of the file to size. Data within the file in memory location after size is deleted. The status of blocks (from which data is deleted) is updated to available.

14. Show memory map

It shows the distribution of files in the memory. It displays the file name including size, starting address and blocks list.

The indentation indicates the hierarchy in the directory structure i.e which folder or file is present where.

15. Exit

Exits the program and stores the updated structure in the .dat file.

How to run Program:

The menu is displayed on the console. The user can select the option to perform the operations accordingly.

Python: 3.9

Libraries:

* Os
* Json
* Math

e. Limitations of your system: Explicitly write constraints of your system here

For implementing the following functions we have to be in current directory:

* Write in file
* Read from file
* Write on a specific position
* Read from specific position
* For moving a file a path given should be correct
* Create a new directory

f. Output: Screenshots of the executed program

Initial Memory Map:

Text, letter

Description automatically generated

1. Create a file

A picture containing table

Description automatically generated

A picture containing table

Description automatically generated

2. Delete a file

A picture containing text

Description automatically generated

A picture containing table

Description automatically generated

3. Make directory

A picture containing text

Description automatically generated

Text

Description automatically generated with low confidence

4. Change directory

A picture containing diagram

Description automatically generated

5. Move a file

Text

Description automatically generated with low confidence

Text

Description automatically generated with low confidence

6. Open a file

A picture containing table

Description automatically generated

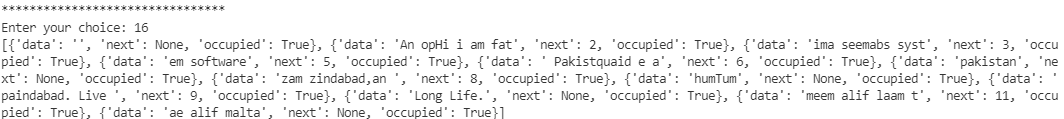


7. Close a file

8. Write in a file

A picture containing text

Description automatically generated





Text

Description automatically generated



9. Write in a file at specific position

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

10. Read a file

Text, table

Description automatically generated

11. Read a file from specific position

Text

Description automatically generated

12. Move text in a file

Text, letter

Description automatically generated

Text

Description automatically generated

13. Truncate text in a file

Table

Description automatically generated with medium confidence

Text

Description automatically generated with low confidence

g. References: List all the sources you have used as references in implementing your work.

Stackoverflow

Programiz:

https://www.programiz.com/python-programming/json

Medium:

https://medium.com/@daniel.tooke/variables-and-memory-addresses-in-python-6d96d672ed3d