192311035

M.RAVI SAI VINAY

EXPERIMENT-36

With linked allocation, each file is a linked list of disk blocks; the disk blocks may be scattered anywhere on the disk. The directory contains a pointer to the first and last blocks of the file. Each block contains a pointer to the next block. Design a C program to simulate the file allocation strategy.

AIM:-

The aim of this program is to simulate the **Linked Allocation** strategy of file storage in a file system. Each file is represented as a linked list of blocks, where each block contains data and a pointer to the next block. This strategy allows the blocks to be scattered across the disk and eliminates the need for contiguous storage. The directory contains a pointer to the first block of the file, and each block contains a pointer to the next block.

ALGORITHM:-

1. Define a Block structure that holds data and a pointer to the next block.
2. Define a File structure to store pointers to the first and last blocks of the file.
3. Implement a function to create a new block and initialize its data and next pointer.
4. Implement a function to add a block to the file (insert the block at the end of the linked list).
5. Implement a function to display the file contents by traversing the linked list of blocks.
6. In the main function, prompt the user for the number of blocks, gather the data for each block, and display the file contents.

PROCEDURE:-

1. Initialize the File structure with firstBlock and lastBlock as NULL.
2. Prompt the user for the number of blocks and the data to store in each block.
3. For each block, create a new block, add it to the file, and link it to the next block.

4. After all blocks have been added, display the contents of the file by traversing the linked list starting from the firstBlock.
5. Print the data of each block as it is traversed.

CODE:-

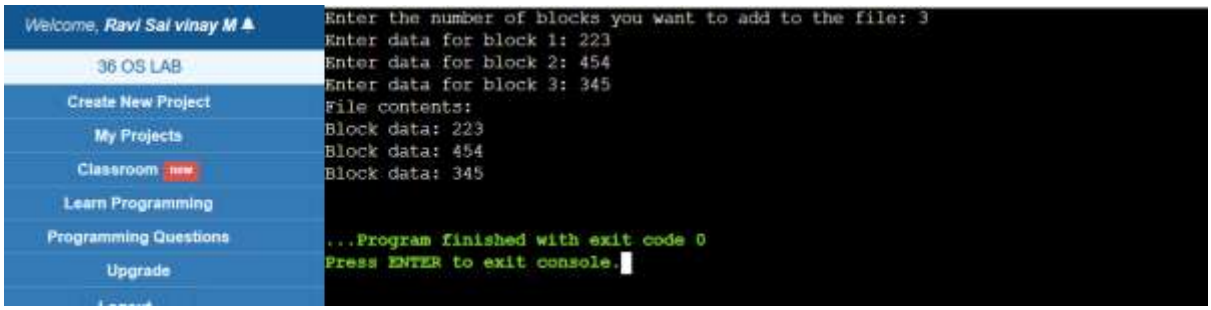
```
#include <stdio.h>

#include <stdlib.h>


// Structure for a block in the file (each block contains data and a pointer to the next block)

struct Block {

    int data;

    struct Block *next;

};


// Structure for a file (contains a pointer to the first block and the last block)

struct File {

    struct Block *firstBlock;

    struct Block *lastBlock;

};


// Function to create a new block and return its pointer

struct Block* createBlock(int data) {

    struct Block *newBlock = (struct Block*)malloc(sizeof(struct Block));

    newBlock->data = data;
```

```
    newBlock->next = NULL;
    return newBlock;
}

// Function to add a block to the file (to the end of the linked list)
void addBlock(struct File *file, int data) {
    struct Block *newBlock = createBlock(data);

    if (file->firstBlock == NULL) {
        // If this is the first block, set both first and last pointers to this block
        file->firstBlock = newBlock;
        file->lastBlock = newBlock;
    } else {
        // Add the new block to the end of the linked list
        file->lastBlock->next = newBlock;
        file->lastBlock = newBlock;
    }
}

// Function to display the file contents by traversing the linked list of blocks
void displayFile(struct File *file) {
    struct Block *current = file->firstBlock;
    if (current == NULL) {
        printf("The file is empty.\n");
```

```
    } else {
        printf("File contents:\n");
        while (current != NULL) {
            printf("Block data: %d\n", current->data);
            current = current->next;
        }
    }
}

int main() {
    struct File file = {NULL, NULL}; // Initialize the file with no blocks

    int numBlocks, data;
    printf("Enter the number of blocks you want to add to the file: ");
    scanf("%d", &numBlocks);

    // Add blocks to the file
    for (int i = 0; i < numBlocks; i++) {
        printf("Enter data for block %d: ", i + 1);
        scanf("%d", &data);
        addBlock(&file, data);
    }

    // Display the contents of the file
```

```
    displayFile(&file);


    return 0;
}
```

OUTPUT:-



RESULT:-

The program successfully simulates the **linked allocation** file storage strategy. It dynamically adds blocks to the file, linking them together using pointers. The program allows files to be represented as a linked list of blocks, where each block contains a pointer to the next block, allowing the file to be stored non-contiguously across the disk. The file contents are displayed by traversing the linked list, and each block's data is printed.