

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

To design a C program that simulates the **Linked Allocation File System**, where each file is represented as a linked list of disk blocks, and the directory contains pointers to the first and last blocks of the file. Each block contains a pointer to the next block, and the blocks may be scattered anywhere on the disk.

ALGORITHM

1. Start
2. Define a structure FileBlock to represent each disk block. Each block contains data and a pointer to the next block.
3. Create a structure File to represent the file, which contains pointers to the first and last blocks of the file.
4. Create functions for file operations such as adding a new block, displaying the file contents, and accessing specific blocks.
5. Implement a function to add a new block to the file, updating the directory with the first and last block pointers.
6. Implement a function to display the file contents by traversing through the linked list of blocks.
7. Implement a function to access a specific block in the file by following the linked list of blocks.
8. Stop

PROCEDURE:

1. Include necessary libraries (stdio.h for input/output and stdlib.h for dynamic memory management).
2. Define a FileBlock structure to represent a block with data and a pointer to the next block.
3. Define a File structure that holds pointers to the first and last blocks of the file.

4. Create functions to add new blocks to the file, display file contents, and access specific blocks.
5. Initialize the file and perform operations such as adding blocks, displaying contents, and accessing specific blocks.
6. End

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct Block {
    int blockNumber;
    struct Block* next;
} Block;

typedef struct File {
    char name[20];
    Block* firstBlock;
    Block* lastBlock;
} File;

File* createFile(const char* name) {
    File* newFile = (File*)malloc(sizeof(File));
    strcpy(newFile->name, name);
    newFile->firstBlock = NULL;
    newFile->lastBlock = NULL;
    return newFile;
}

void addBlock(File* file, int blockNumber) {
    Block* newBlock = (Block*)malloc(sizeof(Block));
    newBlock->blockNumber = blockNumber;
    newBlock->next = NULL;

    if (file->lastBlock == NULL) {
        file->firstBlock = newBlock;
    } else {
        file->lastBlock->next = newBlock;
    }
    file->lastBlock = newBlock;
}

void displayFile(File* file) {
    printf("File: %s\n", file->name);
    Block* current = file->firstBlock;
    while (current != NULL) {
        printf("Block Number: %d\n", current->blockNumber);
        current = current->next;
    }
}
```

```

void freeFile(File* file) {
    Block* current = file->firstBlock;
    Block* nextBlock;
    while (current != NULL) {
        nextBlock = current->next;
        free(current);
        current = nextBlock;
    }
    free(file);
}

int main() {
    File* myFile = createFile("example.txt");
    addBlock(myFile, 1);
    addBlock(myFile, 3);
    addBlock(myFile, 5);

    displayFile(myFile);

    freeFile(myFile);
    return 0;
}

```

OUTPUT:

```

main.c
File: example.txt
Block Number: 1
Block Number: 3
Block Number: 5

...Program finished with exit code 0
Press ENTER to exit console.

```