

35. Consider a file system that brings all the file pointers together into an index block. The *ith* entry in the index block points to the *ith* block of the file. Design a C program to simulate the file allocation strategy.

AIM

To design a C program that simulates a **File Allocation Strategy** using an **Index Block**, where all the file pointers are brought together into an index block, and each entry in the index block points to the respective block of the file.

ALGORITHM

- 1. Start**
2. Define a structure FileBlock to represent a block in the file.
3. Create an array indexBlock[] to represent the index block that holds pointers to file blocks.
4. Create a function to add a new record to the file and update the index block accordingly.
5. Create a function to display the current file blocks and index block.
6. Create a function to access a specific file block using the index block.
- 7. Stop**

PROCEDURE

1. Include necessary libraries (stdio.h for input/output, stdlib.h for dynamic memory management).
2. Define a structure FileBlock to represent each file block (with data).
3. Define an array indexBlock[] to simulate the index block storing pointers to file blocks.
4. Implement functions to add new file blocks (addFileBlock()), display file blocks (displayFile()), and access specific blocks (accessFileBlock()).
5. Initialize the file system and perform operations such as adding file blocks, displaying file contents, and accessing blocks using the index.
- 6. End**

CODE:

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

```
#define MAX_BLOCKS 10
```

```
typedef struct {
    char data[100];
} FileBlock;
```

```
FileBlock file[MAX_BLOCKS];
int indexBlock[MAX_BLOCKS];
```

```

int blockCount = 0;

void addFileBlock(const char *data) {
    if (blockCount < MAX_BLOCKS) {
        snprintf(file[blockCount].data, sizeof(file[blockCount].data), "%s", data);
        indexBlock[blockCount] = blockCount;
        blockCount++;
    } else {
        printf("File system is full. Cannot add more blocks.\n");
    }
}

void displayFile() {
    if (blockCount == 0) {
        printf("No blocks in the file.\n");
        return;
    }

    printf("Index Block: ");
    for (int i = 0; i < blockCount; i++) {
        printf("%d ", indexBlock[i]);
    }
    printf("\n");

    printf("File Blocks:\n");
    for (int i = 0; i < blockCount; i++) {
        printf("Block %d: %s\n", indexBlock[i], file[indexBlock[i]].data);
    }
}

void accessFileBlock(int blockNum) {
    if (blockNum >= 0 && blockNum < blockCount) {
        printf("Accessing Block %d: %s\n", indexBlock[blockNum],
file[indexBlock[blockNum]].data);
    } else {
        printf("Invalid block number.\n");
    }
}

int main() {
    int choice, blockNum;
    char data[100];

    while (1) {
        printf("\nFile Allocation System (Index Block)\n");
        printf("1. Add File Block\n");
    }
}

```

```

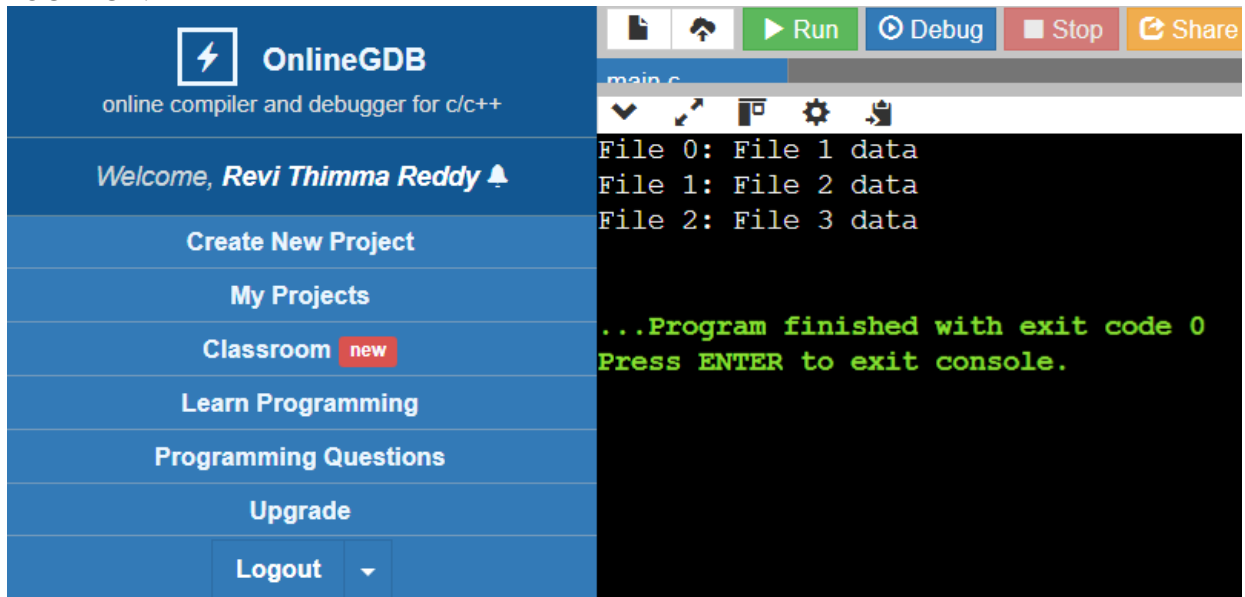
printf("2. Display File Blocks\n");
printf("3. Access a Specific Block\n");
printf("4. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
getchar();

switch (choice) {
    case 1:
        printf("Enter Data for Block: ");
        fgets(data, sizeof(data), stdin);
        data[strcspn(data, "\n")] = 0;
        addFileBlock(data);
        break;
    case 2:
        displayFile();
        break;
    case 3:
        printf("Enter Block Number to Access: ");
        scanf("%d", &blockNum);
        accessFileBlock(blockNum);
        break;
    case 4:
        exit(0);
    default:
        printf("Invalid choice. Please try again.\n");
}
}

return 0;
}

```

OUTPUT:



The screenshot shows the OnlineGDB web interface. On the left is a sidebar with navigation links: 'Create New Project', 'My Projects', 'Classroom' (with a 'new' badge), 'Learn Programming', 'Programming Questions', 'Upgrade', and a 'Logout' button. The main area displays the output of a C program. The output shows three lines of data: 'File 0: File 1 data', 'File 1: File 2 data', and 'File 2: File 3 data'. Below this, a green message states: '...Program finished with exit code 0' and 'Press ENTER to exit console.' The top of the interface includes a toolbar with icons for file operations and buttons for 'Run', 'Debug', 'Stop', and 'Share'.

```

File 0: File 1 data
File 1: File 2 data
File 2: File 3 data

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

```

