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

#define BLOCKS 5

typedef struct {
    int *indexBlock;
} FileSystem;

void allocateBlocks(FileSystem *fs) {
    fs->indexBlock = malloc(BLOCKS * sizeof(int));
    for (int i = 0; i < BLOCKS; i++) {
        fs->indexBlock[i] = i; // Pointing to block i
    }
}

void displayBlocks(FileSystem *fs) {
    for (int i = 0; i < BLOCKS; i++) {
        printf("Index %d -> Block %d\n", i, fs->indexBlock[i]);
    }
}

int main() {
    FileSystem fs;
    allocateBlocks(&fs);
    displayBlocks(&fs);
    free(fs.indexBlock);
    return 0;
}
```

```
Index 0 -> Block 0
Index 1 -> Block 1
Index 2 -> Block 2
Index 3 -> Block 3
Index 4 -> Block 4


=== Code Execution Successful ===
```

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

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

typedef struct File {
    Block* head;
    Block* tail;
} File;

File* createFile() {
    File* file = (File*)malloc(sizeof(File));
    file->head = file->tail = NULL;
    return file;
}

void addBlock(File* file, int blockNumber) {
    Block* newBlock = (Block*)malloc(sizeof(Block));
    newBlock->blockNumber = blockNumber;
    newBlock->next = NULL;
    if (!file->head) {
        file->head = file->tail = newBlock;
    } else {
        file->tail->next = newBlock;
        file->tail = newBlock;
    }
}

void displayFile(File* file) {
    Block* current = file->head;
    while (current) {
        printf("Block Number: %d\n", current->blockNumber);
        current = current->next;
    }
}

int main() {
    File* myFile = createFile();
    addBlock(myFile, 1);
    addBlock(myFile, 2);
    addBlock(myFile, 3);
    displayFile(myFile);
    return 0;
}
```

```
Block Number: 1
Block Number: 2
Block Number: 3


=== Code Execution Successful ===
```