**EXPERIMENT – 35**

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.

#include <stdio.h>

#include <stdlib.h>

#define MAX\_BLOCKS 100

#define MAX\_FILE\_BLOCKS 10

int disk[MAX\_BLOCKS];

typedef struct {

int indexBlock;

int blocks[MAX\_FILE\_BLOCKS];

int blockCount;

} File;

void allocateFile(File \*file) {

int count, i, block;

printf("Enter number of blocks required for the file (max %d): ", MAX\_FILE\_BLOCKS);

scanf("%d", &count);

if (count > MAX\_FILE\_BLOCKS) {

printf("Error: Too many blocks requested.\n");

return;

}

for (i = 0; i < MAX\_BLOCKS; i++) {

if (disk[i] == 0) {

file->indexBlock = i;

disk[i] = 1;

break;

}

}

if (i == MAX\_BLOCKS) {

printf("Error: No free block available for index block.\n");

return;

}

printf("Allocating %d data blocks...\n", count);

file->blockCount = 0;

for (i = 0; i < count; i++) {

int found = 0;

for (block = 0; block < MAX\_BLOCKS; block++) {

if (disk[block] == 0) {

disk[block] = 1;

file->blocks[file->blockCount++] = block;

found = 1;

break;

}

}

if (!found) {

printf("Error: Disk full. Rolling back allocation.\n");

for (int j = 0; j < file->blockCount; j++)

disk[file->blocks[j]] = 0;

disk[file->indexBlock] = 0;

return;

}

}

printf("File allocated successfully.\n");

printf("Index Block at: %d\n", file->indexBlock);

printf("Data Blocks: ");

for (i = 0; i < file->blockCount; i++)

printf("%d ", file->blocks[i]);

printf("\n");

}

void accessBlock(File file) {

int i;

printf("Enter block number to access (0 to %d): ", file.blockCount - 1);

scanf("%d", &i);

if (i < 0 || i >= file.blockCount) {

printf("Invalid block number!\n");

} else {

printf("Accessing file block %d: Physical Block -> %d (via index block %d)\n",

i, file.blocks[i], file.indexBlock);

}

}

int main() {

File file;

int choice;

while (1) {

printf("\n--- Indexed File Allocation ---\n");

printf("1. Allocate File\n");

printf("2. Access File Block\n");

printf("3. Exit\n");

printf("Enter choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

allocateFile(&file);

break;

case 2:

accessBlock(file);

break;

case 3:

exit(0);

default:

printf("Invalid choice!\n");

}

}

return 0;

}

SAMPLE OUTPUT:

--- Indexed File Allocation ---

1. Allocate File

2. Access File Block

3. Exit

Enter choice: 1

Enter number of blocks required for the file (max 10): 4

Allocating 4 data blocks...

File allocated successfully.

Index Block at: 0

Data Blocks: 2 5 7 9

Enter choice: 2

Enter block number to access (0 to 3): 2

Accessing file block 2: Physical Block -> 7 (via index block 0)