**EXPERIMENT -1**

**File Allocation Stratergies**

## Aim

Simulate the following file allocation strategies.

a) Sequential b) Indexed c) Linked

**Program**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void contiguousAllocation() {

    struct File {

        char name[50];

        int startAddress;

        int blocks;

    } files[100];

    int fileIndex = 0, choice = 1, i;

    while (choice != 3) {

        printf("\n\nContiguous Allocation\n\n1.Add Files\n2.Search\n3.Exit\n\nEnter Choice:");

        scanf("%d", &choice);

        if (choice == 1) {

            int fileNo;

            printf("\nEnter no of files: ");

            scanf("%d", &fileNo);

            for (int i = fileIndex; i < fileIndex + fileNo; i++) {

                printf("\nEnter file name:");

                scanf("%s", &files[i].name);

                printf("Enter file starting address:");

                scanf("%d", &files[i].startAddress);

                printf("Enter no of blocks of file:");

                scanf("%d", &files[i].blocks);

            }

            fileIndex += fileNo;

        }

        else if (choice == 2) {

            char searchKey[40];

            printf("\nEnter file name: ");

            scanf("%s", &searchKey);

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

                struct File file = files[i];

                if (strcmpi(files[i].name, searchKey) == 0) {

                    printf("Starting Address: %d", files[i].startAddress);

                    printf("\nNo. of blocks: %d", files[i].blocks);

                    printf("\nBlocks: ");

                    for (int j = files[i].startAddress; j < files[i].startAddress + files[i].blocks; j++) {

                        printf("%d ", j);

                    }

                    break;

                }

            }

            if (i == fileIndex) {

                printf("File not found");

            }

        }

        else if (choice == 3)

            break;

    }

}

void linkedListAllocation() {

    struct linkedFile {

        int address;

        struct linkedFile \*next;

    };

    struct File {

        char name[50];

        int blocks;

        struct linkedFile \*start;

        struct linkedFile \*end;

    } files[100];

    int fileIndex = 0, choice = 1, i;

    while (choice != 3) {

        printf("\n\nLinked List Allocation\n\n1.Add Files\n2.Search\n3.Exit\n\nEnter Choice:");

        scanf("%d", &choice);

        if (choice == 1) {

            int fileNo;

            printf("\nEnter no of files: ");

            scanf("%d", &fileNo);

            for (int i = fileIndex; i < fileIndex + fileNo; i++) {

                printf("\nEnter file name:");

                scanf("%s", &files[i].name);

                printf("Enter no of blocks of file:");

                scanf("%d", &files[i].blocks);

                printf("Enter the blocks addresses of file:");

                struct linkedFile \*prevPtr;

                for (int j = 0; j < files[i].blocks; j++) {

                    struct linkedFile \*ptr = (struct linkedFile \*)malloc(sizeof(struct linkedFile));

                    scanf("%d", &ptr->address);

                    ptr->next = NULL;

                    if (j == 0) {

                        files[i].start = ptr;

                        prevPtr = ptr;

                    } else {

                        prevPtr->next = ptr;

                        prevPtr = ptr;

                    }

                }

                files[i].end = prevPtr;

            }

            fileIndex += fileNo;

        }

        else if (choice == 2) {

            char searchKey[40];

            printf("\nEnter file name: ");

            scanf("%s", &searchKey);

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

                struct File file = files[i];

                if (strcmpi(file.name, searchKey) == 0) {

                    printf("Starting Address: %d", file.start->address);

                    printf("\nEnd Address: %d", file.end->address);

                    printf("\nNo. of blocks: %d", file.blocks);

                    printf("\nBlocks: ");

                    for (struct linkedFile \*j = file.start; j != NULL; j = j->next) {

                        printf("%d-> ", j->address);

                    }

                    break;

                }

            }

            if (i == fileIndex) {

                printf("File not found");

            }

        }

        else if (choice == 3)

            break;

    }

}

void indexedAllocation() {

    struct File {

        char name[50];

        int blocks;

        int indexAddress;

        int addresses[100];

    } files[100];

    int fileIndex = 0, choice = 1, i;

    while (choice != 3) {

        printf("\n\nIndexed Allocation\n\n1.Add Files\n2.Search\n3.Exit\n\nEnter Choice:");

        scanf("%d", &choice);

        if (choice == 1) {

            int fileNo;

            printf("\nEnter no of files: ");

            scanf("%d", &fileNo);

            for (int i = fileIndex; i < fileIndex + fileNo; i++) {

                printf("\nEnter file name:");

                scanf("%s", &files[i].name);

                printf("Enter index block address:");

                scanf("%d", &files[i].indexAddress);

                printf("Enter no of blocks of file:");

                scanf("%d", &files[i].blocks);

                printf("Enter the blocks addresses of file:");

                for (int j = 0; j < files[i].blocks; j++)

                    scanf("%d", &files[i].addresses[j]);

            }

            fileIndex += fileNo;

        }

        else if (choice == 2) {

            char searchKey[40];

            printf("\nEnter file name: ");

            scanf("%s", &searchKey);

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

                struct File file = files[i];

                if (strcmpi(file.name, searchKey) == 0) {

                    printf("\nIndex block address: %d", file.indexAddress);

                    printf("\nNo. of blocks: %d", file.blocks);

                    printf("\nBlocks: ");

                    for (int j = 0; j < file.blocks; j++) {

                        printf("%d-> %d, ", file.indexAddress, file.addresses[j]);

                    }

                    break;

                }

            }

            if (i == fileIndex) {

                printf("File not found");

            }

        }

        else if (choice == 3)

            break;

    }

}

void main() {

    int choice = 1;

    while (choice != 4) {

        printf("\n\n1.Contiguous Allocation\n2.Linked List Allocation\n3.Indexed Allocation\n4.Exit\n\nEnter Choice:");

        scanf("%d", &choice);

        switch (choice) {

        case 1:

            contiguousAllocation();

            break;

        case 2:

            linkedListAllocation();

            break;

        case 3:

            indexedAllocation();

            break;

        }

    }

}

**Output**

