Assigment 8

Assignment No. 8: Implement the C program for Disk Scheduling Algorithms: SSTF, SCAN, C-Look considering the initial head position moving away from the spindle.

8_disk.c

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

void cLook(int RQ[], int n, int initial, int size) {
    int TotalHeadMovement = 0, i, index;

    // Sort the request array
    for (i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (RQ[j] > RQ[j + 1]) {
                int temp = RQ[j];
                RQ[j] = RQ[j + 1];
                RQ[j + 1] = temp;
            }
        }
    }

    // Find the index of the first request greater than the initial position
    index = 0;
    while (index < n && RQ[index] < initial) {
        index++;
    }

    // C-look logic
    if (index < n) {
        for (i = index; i < n; i++) {
            TotalHeadMovement += abs(RQ[i] - initial);
            initial = RQ[i];
        }
        // Go to the first request
        if (index > 0) {
            TotalHeadMovement += abs(RQ[0] - initial);
            initial = RQ[0];
            for (i = 1; i < index; i++) {
                TotalHeadMovement += abs(RQ[i] - initial);
                initial = RQ[i];
            }
        }
    }

    printf("Total head movement (C-look) is %d\n", TotalHeadMovement);
}
```

```
void scan(int RQ[], int n, int initial, int size, int move) {
    int TotalHeadMovement = 0, i, index = 0;

    // Sort the request array
    for (i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (RQ[j] > RQ[j + 1]) {
                int temp = RQ[j];
                RQ[j] = RQ[j + 1];
                RQ[j + 1] = temp;
            }
        }
    }

    // Find the index of the first request greater than the initial position
    while (index < n && RQ[index] < initial) {
        index++;
    }

    // Scan logic
    if (move == 1) { // Move towards high values
        for (i = index; i < n; i++) {
            TotalHeadMovement += abs(RQ[i] - initial);
            initial = RQ[i];
        }
        // Move to the end of the disk
        if (initial < size - 1) {
            TotalHeadMovement += abs(size - 1 - initial);
            initial = size - 1; // Move to the end
        }
        // Now move left to the start of the requests
        for (i = index - 1; i >= 0; i--) {
            TotalHeadMovement += abs(RQ[i] - initial);
            initial = RQ[i];
        }
    } else { // Move towards low values
        for (i = index - 1; i >= 0; i--) {
            TotalHeadMovement += abs(RQ[i] - initial);
            initial = RQ[i];
        }
        // Move to the beginning of the disk
        if (initial > 0) {
            TotalHeadMovement += abs(0 - initial);
            initial = 0; // Move to the start
        }
        // Now move right to the end of the requests
        for (i = index; i < n; i++) {
            TotalHeadMovement += abs(RQ[i] - initial);
            initial = RQ[i];
        }
    }
    printf("Total head movement (Scan) is %d\n", TotalHeadMovement);
}

void sSTF(int RQ[], int n, int initial) {
    int TotalHeadMovement = 0, count = 0;

    // SSTF disk scheduling logic
    while (count < n) {
        int min = 1000, index = -1;
        for (int i = 0; i < n; i++) {
            if (RQ[i] != 1000) { // Only consider unprocessed requests
                int d = abs(RQ[i] - initial);
                if (d < min) {
                    min = d;
                    index = i;
                }
            }
        }

        if (index != -1) {
            TotalHeadMovement += min;
            initial = RQ[index];
            RQ[index] = 1000; // Mark this request as processed
            count++;
        } else {
            break; // No more requests to process
        }
    }

    printf("Total head movement (SSTF) is %d\n", TotalHeadMovement);
}

int main() {
    int RQ[100], n, initial, size, move, choice;

    // Input the number of requests
    printf("Enter the number of Requests (max 100):\n");
    scanf("%d", &n);

    if (n <= 0 || n > 100) {
        printf("Invalid number of requests. Please enter a number between 1 and 100.\n");
        return 1;
    }

    // Input the request sequence
    printf("Enter the Requests sequence:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &RQ[i]);
    }

    // Input the initial head position
    printf("Enter initial head position:\n");
    scanf("%d", &initial);

    // Input the total disk size for scan and C-look
    printf("Enter total disk size:\n");
    scanf("%d", &size);

    // Input the direction for scan
    printf("Enter the head movement direction for Scan (high = 1, low = 0):\n");
    scanf("%d", &move);

    // Choose scheduling algorithm
    printf("Select Disk Scheduling Algorithm:\n");
    printf("1. C-look\n");
    printf("2. Scan\n");
    printf("3. SSTF\n");
    printf("Enter your choice (1-3):\n");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            cLook(RQ, n, initial, size);
            break;
        case 2:
            scan(RQ, n, initial, size, move);
            break;
        case 3:
            sSTF(RQ, n, initial);
            break;
        default:
            printf("Invalid choice! Please select a valid algorithm.\n");
            break;
    }

    return 0;
}
```

Output:

```
pl-17@pl17-OptiPlex-3020:~/IT/07$ gcc 8_disk.c
pl-17@pl17-OptiPlex-3020:~/IT/07$ ./a.out
Enter the number of Requests (max 100):
15
Enter the Requests sequence:
0
10
15
18
22
48
50
92
65
```

```
48
118
182
69
128
190
Enter initial head position:
50
Enter total disk size:
200
Enter the head movement direction for Scan (high = 1, low = 0):
1
Select Disk Scheduling Algorithm:
1. C-look
2. Scan
3. SSTF
Enter your choice (1-3):
1
Total head movement (C-look) is 378

Enter your choice (1-3):
2
Total head movement (Scan) is 348

Enter your choice (1-3):
3
Total head movement (SSTF) is 334
```