EXP 10

(FCFS)

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

void FCFS(int requests[], int n, int head) { int total\_head\_movement = 0;

int current\_position = head;

printf("FCFS Disk Scheduling:\n"); printf("Order of requests: ");

for (int i = 0; i < n; i++) { printf("%d ", requests[i]);

total\_head\_movement += abs(current\_position - requests[i]); current\_position = requests[i];

}

printf("\nTotal head movement: %d\n", total\_head\_movement);

}

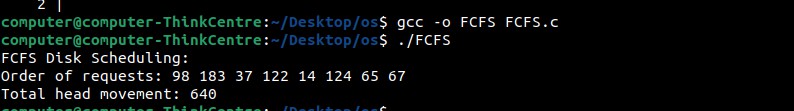
int main() {

int requests[] = {98, 183, 37, 122, 14, 124, 65, 67};

int n = sizeof(requests) / sizeof(requests[0]); int head = 53; // initial head position FCFS(requests, n, head);

return 0;

}



(SCAN)

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

void SCAN(int requests[], int n, int head, int direction) { int total\_head\_movement = 0;

int current\_position = head; for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (requests[j] > requests[j + 1]) { int temp = requests[j]; requests[j] = requests[j + 1]; requests[j + 1] = temp; } } }

printf("SCAN Disk Scheduling:\n"); printf("Order of requests: ");

if (direction == 1) { // Move towards the higher end for (int i = 0; i < n; i++) {

if (requests[i] >= current\_position) { for (int j = i; j < n; j++) {

printf("%d ", requests[j]);

total\_head\_movement += abs(current\_position - requests[j]); current\_position = requests[j];

} break; } }

total\_head\_movement += abs(current\_position - 199); // Assuming 199 is the end current\_position = 199;

for (int i = n - 1; i >= 0; i--) {

if (requests[i] < current\_position) { printf("%d ", requests[i]);

total\_head\_movement += abs(current\_position - requests[i]); current\_position = requests[i]; } }

} else { // Move towards the lower end for (int i = n - 1; i >= 0; i--) {

if (requests[i] <= current\_position) { for (int j = i; j >= 0; j--) {

printf("%d ", requests[j]);

total\_head\_movement += abs(current\_position - requests[j]); current\_position = requests[j]; }

break; } }

total\_head\_movement += abs(current\_position - 0); current\_position = 0;

for (int i = 0; i < n; i++) {

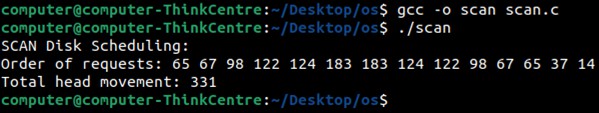
if (requests[i] > current\_position) { printf("%d ", requests[i]);

total\_head\_movement += abs(current\_position - requests[i]); current\_position = requests[i]; } } }

printf("\nTotal head movement: %d\n", total\_head\_movement); } int main() {

int requests[] = {98, 183, 37, 122, 14, 124, 65, 67};

int n = sizeof(requests) / sizeof(requests[0]); int head = 53; // initial head position

int direction = 1; // 1 for upwards, 0 for downwards SCAN(requests, n, head,

direction); return 0; }

(CSCAN)

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

void CSCAN(int requests[], int n, int head) { int total\_head\_movement = 0;

int current\_position = head;

// Sort the requests

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (requests[j] > requests[j + 1]) { int temp = requests[j]; requests[j] = requests[j + 1]; requests[j + 1] = temp;

}

}

}

printf("CSCAN Disk Scheduling:\n"); printf("Order of requests: ");

// Move to the end

for (int i = 0; i < n; i++) {

if (requests[i] >= current\_position) { for (int j = i; j < n; j++) {

printf("%d ", requests[j]);

total\_head\_movement += abs(current\_position - requests[j]); current\_position = requests[j];

}

break; } }

total\_head\_movement += abs(current\_position - 199); // Assuming 199 is the end current\_position = 0; // Jump to start

for (int i = 0; i < n; i++) {

if (requests[i] >= current\_position) { printf("%d ", requests[i]);

total\_head\_movement += abs(current\_position - requests[i]); current\_position = requests[i];

}}

printf("\nTotal head movement: %d\n", total\_head\_movement);

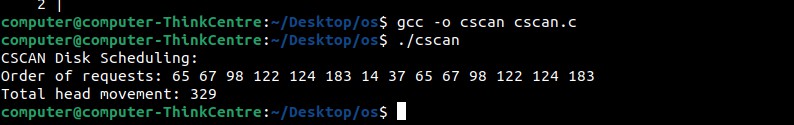
}

int main() {

int requests[] = {98, 183, 37, 122, 14, 124, 65, 67};

int n = sizeof(requests) / sizeof(requests[0]); int head = 53; // initial head position CSCAN(requests, n, head);

return 0; }



(SSTF)

#include <stdio.h> #include <stdbool.h> #include <limits.h> #include <stdlib.h>

int find\_nearest(int requests[], bool visited[], int head, int n) { int min\_distance = INT\_MAX;

int index = -1;

for (int i = 0; i < n; i++) { if (!visited[i]) {

int distance = abs(requests[i] - head); if (distance < min\_distance) {

min\_distance = distance; index = i;

}} }

return index;

}

void SSTF(int requests[], int n, int head) { int total\_head\_movement = 0;

bool visited[n];

for (int i = 0; i < n; i++) visited[i] = false; printf("SSTF Disk Scheduling:\n"); printf("Order of requests: ");

for (int i = 0; i < n; i++) {

int index = find\_nearest(requests, visited, head, n); visited[index] = true;

printf("%d ", requests[index]);

total\_head\_movement += abs(head - requests[index]); head = requests[index];

}

printf("\nTotal head movement: %d\n", total\_head\_movement);

}

int main() {

int requests[] = {98, 183, 37, 122, 14, 124, 65, 67};

int n = sizeof(requests) / sizeof(requests[0]); int head = 53; // initial head position SSTF(requests, n, head);

return 0;

}

