 ***DEPARTMENT OF COMPUTER ENGINEERING***

Experiment No.

|  |  |
| --- | --- |
| Semester | S.E. Semester IV – Computer Engineering |
| Subject | Operating System |
| Subject Professor In-charge | SNA |
| Assisting Teachers | Ms. Rasika Ransing |
| Laboratory | M310B – Computer Engineering Laboratory |

|  |  |  |
| --- | --- | --- |
| Student Name | Amir Shaikh | |
| Roll Number | 18102A0053 | |
| Grade and Subject Teacher’s Signature |  |  |

|  |  |  |
| --- | --- | --- |
| Experiment Number |  | |
| Experiment Title | To implement Disc Scheduling (FCFS, SSTF, SCAN, LOOK) | |
| Resources / Apparatus Required | Hardware: PC | Software:  Compiler |
| Objectives  (Skill Set / Knowledge Tested / Imparted) | File Management techniques | |
| Theory: | In computing, scheduling is the method by which work is assigned to resources that complete the work. The work may be virtual computation elements such as threads, processes or data flows, which are in turn scheduled onto hardware resources such as processors, network links or expansion cards.  A scheduler is what carries out the scheduling activity. Schedulers are often implemented so they keep all computer resources busy (as in load balancing), allow multiple users to share system resources effectively, or to achieve a target quality of service. Scheduling is fundamental to computation itself, and an intrinsic part of the execution model of a computer system; the concept of scheduling makes it possible to have computer multitasking with a single central processing unit (CPU). | |
| Code | //fcfs  size = 8    def FCFS(arr, head):    seek\_count = 0  distance, cur\_track = 0, 0    for i in range(size):  cur\_track = arr[i]  distance = abs(cur\_track - head)  seek\_count += distance  head = cur\_track    print("Total number of seek operations = ", seek\_count)  print("Seek Sequence is")    for i in range(size):  print(arr[i])  if \_\_name\_\_ == '\_\_main\_\_':    arr = [ 176, 79, 34, 60, 92, 11, 41, 114 ]  head = 50    FCFS(arr, head)  //LOOK  #include <bits/stdc++.h>  using namespace std;    int size = 8;  int disk\_size = 200;    void LOOK(int arr[], int head, string direction)  {  int seek\_count = 0;  int distance, cur\_track;  vector<int> left, right;  vector<int> seek\_sequence;    for (int i = 0; i < size; i++) {  if (arr[i] < head)  left.push\_back(arr[i]);  if (arr[i] > head)  right.push\_back(arr[i]);  }    std::sort(left.begin(), left.end());  std::sort(right.begin(), right.end());    int run = 2;  while (run--) {  if (direction == "left") {  for (int i = left.size() - 1; i >= 0; i--) {  cur\_track = left[i];  seek\_sequence.push\_back(cur\_track);  distance = abs(cur\_track - head);  seek\_count += distance;  head = cur\_track;  }  direction = "right";  }  else if (direction == "right") {  for (int i = 0; i < right.size(); i++) {  cur\_track = right[i];  seek\_sequence.push\_back(cur\_track);    distance = abs(cur\_track - head);  seek\_count += distance;  head = cur\_track;  }  direction = "left";  }  }    cout << "Total number of seek operations = "  << seek\_count << endl;    cout << "Seek Sequence is" << endl;    for (int i = 0; i < seek\_sequence.size(); i++) {  cout << seek\_sequence[i] << endl;  }  }    int main()  {    int arr[size] = { 176, 79, 34, 60,  92, 11, 41, 114 };  int head = 50;  string direction = "right";    cout << "Initial position of head: "  << head << endl;    LOOK(arr, head, direction);    return 0;  }  //SSTF  def calculateDifference(queue, head, diff):  for i in range(len(diff)):  diff[i][0] = abs(queue[i] - head)  def findMin(diff):    index = -1  minimum = 999999999    for i in range(len(diff)):  if (not diff[i][1] and  minimum > diff[i][0]):  minimum = diff[i][0]  index = i  return index    def shortestSeekTimeFirst(request, head):  if (len(request) == 0):  return    l = len(request)  diff = [0] \* l    for i in range(l):  diff[i] = [0, 0]  seek\_count = 0  seek\_sequence = [0] \* (l + 1)    for i in range(l):  seek\_sequence[i] = head  calculateDifference(request, head, diff)  index = findMin(diff)    diff[index][1] = True    seek\_count += diff[index][0]  head = request[index]    seek\_sequence[len(seek\_sequence) - 1] = head    print("Total number of seek operations =",  seek\_count)    print("Seek Sequence is")    # print the sequence  for i in range(l + 1):  print(seek\_sequence[i])    if \_\_name\_\_ =="\_\_main\_\_":    # request array  proc = [176, 79, 34, 60,  92, 11, 41, 114]  shortestSeekTimeFirst(proc, 50)  //SCAN  #include <bits/stdc++.h>  using namespace std;    int size = 8;  int disk\_size = 200;    void SCAN(int arr[], int head, string direction)  {  int seek\_count = 0;  int distance, cur\_track;  vector<int> left, right;  vector<int> seek\_sequence;  if (direction == "left")  left.push\_back(0);  else if (direction == "right")  right.push\_back(disk\_size - 1);    for (int i = 0; i < size; i++) {  if (arr[i] < head)  left.push\_back(arr[i]);  if (arr[i] > head)  right.push\_back(arr[i]);  }    // sorting left and right vectors  std::sort(left.begin(), left.end());  std::sort(right.begin(), right.end());  int run = 2;  while (run--) {  if (direction == "left") {  for (int i = left.size() - 1; i >= 0; i--) {  cur\_track = left[i];  seek\_sequence.push\_back(cur\_track);  distance = abs(cur\_track - head);  seek\_count += distance;  head = cur\_track;  }  direction = "right";  }  else if (direction == "right") {  for (int i = 0; i < right.size(); i++) {  cur\_track = right[i];  seek\_sequence.push\_back(cur\_track);  distance = abs(cur\_track - head);  seek\_count += distance;  head = cur\_track;  }  direction = "left";  }  }    cout << "Total number of seek operations = "  << seek\_count << endl;    cout << "Seek Sequence is" << endl;    for (int i = 0; i < seek\_sequence.size(); i++) {  cout << seek\_sequence[i] << endl;  }  }    int main()  {  int arr[size] = { 176, 79, 34, 60,  92, 11, 41, 114 };  int head = 50;  string direction = "left";    SCAN(arr, head, direction);    return 0;  } | |
| Output |  | |