Programming Assignment 2
Disk Scheduling Algorithms

November 16, 2018

1 Disk Scheduling Algorithms

In this assignment you'll write a program implements the ***first-come first-served, shortest seek time first, SCAN, C-SCAN, LOOK, and C-LOOK scheduling*** algorithms. Calculate and output the total amount of movements for the input cylinder requests and starting position. Review the examples below for details. The first command line parameter will be the name of the input file that contains the algorithm specifications and detailed inputs, as described below. The program must generate output to the console (terminal) screen as specified below.

2 Objectives

The objectives of this assignment are to simulate the CPU scheduline of several processes for the configured *First-Come First-Served*, *preemptive Shortest Job First*, and *Round-Robin* CPU scheduling algorithms. Therefore the algorithm will need to be selected/configured as will the processes and their durations and burst cycles, as appropriate.

2.1 Algorithms

2.1.1 First-Come First-Served

The *First-Come First-Served* algorithm is just that: disk requests are fulfilled in the order of their arrival.

2.1.2 Shortest Seek Time First

The *Shortest Seek Time First* design goal is to move the shortest possible distance first - thru to the end of the current list of disk requests.

2.1.3 SCAN

SCAN works like an elevator does. Starting from the initial cylinder, the head will traverse upwards while processing requests. Upon reaching the upper cylinder, the head will then traverse to the lower cylinder while processing requests. This process will remain active as long as there are unfulfilled requests. ~~It scans down towards the nearest end and then when it hits the bottom it scans up servicing the requests that it didn't get going down. If a request comes in after it has been scanned it will not be serviced until the process comes back down or moves back up~~.

2.1.4 C-SCAN

Circular scanning (*C-SCAN*) works just like the elevator to some extent. It begins its scan toward the nearest end and works it way all the way to the end of the system. Once it hits the bottom or top it jumps to the other end and moves in the same direction.

2.1.5 LOOK

*LOOK*The LOOK algorithm is the same as the SCAN algorithm in that it also honors requests on both sweep direction of the disk head, however, this algorithm "Looks" ahead to see if there are any requests pending in the direction of head movement.

2.1.6 C-LOOK

C-LOOK uses the *LOOK* algorithm with the tweak of when the disk arm gets to the end of the disk, it jumps immediately back to the other end. As in *C-SCAN, C-LOOK* is a **circular** *LOOK* algorithm.

3 Specifications

3.1 Command line parameters

1. Your program must compile and run from the command line.
2. Input the required file name with the specified file containing disk scheduling inputs as specified below.
3. Your program should open the input text file, echo the processed input to the screen, make the necessary calculations, and then output the *search results* to the console (terminal) screen in the format described below.

4. The *command line inputs* or *cli* are the input file name as the first parameter. For example:
   `./pa2 xyz.txt`
   Review the *chkAll.sh* file in the assignment ZIP file for more examples. Note that there are more testing scripts described in the section on *Algorithmic Testing*.

3.2 Input

The inputs for the seek algorithm program, **pa2**, is simple ASCII text that may be a mix of upper and lower case letters for the keywords and integer numbers for most of the parameters. Comments are to be supported, as described below.

1. Every input file has one control element per line. For example, the seek algorithm will be specified as `use sstf` on a single input line of text, where `sstf` is the search algorithm keyword. The search keywords are specified below.

2. Each line of text may contain a comment, delimited by a leading # character. *Note that all lines that have comments will have a leading space in front of the # character.*

The keywords are as follows:

1. **use** - specifies the seek algorithm. Valid values are:
   a) **fcfs**
   b) **sstf**
   c) **SCAN**
   d) **C-SCAN**
   e) **LOOK**
   f) **C-LOOK**
2. **upperCYL** - the upper cylinder number
3. **lowerCYL** - the lower cylinder number
4. **initCYL** - the initial cylinder number
5. **cylreq** - the requested cylinder number
6. **end** - the terminal string for all the cylinder requests. (It is safe to ignore any cylinder requests after the **end** token.)

Your program should ignore everything on a line after a **#** mark and ignore additional spaces in input.

```
Use sstf       # fcfs, sstf, scan, c-scan, look, or c-look
lowerCYL 00000 # valid lower cylinder number
upperCYL 00000 # valid upper cylinder number(> lower cylinder)
initCYL 00000  # initial cylinder position( 0<initCYL<upperCYL)
cylreq 00000   # a single cylinder request
               # where the lowerCYL< cylreq< upperCYL
cylreq 00000   # a single cylinder request
cylreq 00000   # a single cylinder request
               # up to 20 requests
end
```

Each cylinder request must be bounded by the lower and upper cylinder values. In the event they are not, generate an error message and process the next cylinder request until reaching the end of the input file, signified by the **end** command.

3.3 Example

3.3.1 Inputs

The input file, **sstf01.txt** containes the following command tokens.

```
use sstf
lowerCYL 0000
upperCYL 3000
initCYL 27
cylreq 300
cylreq 200
cylreq 100
cylreq 450
cylreq 2500
cylreq 38
end
```

3.3.2 Input confirmation

This output shown below is derived from the input file, **sstf01.txt**. Note that the inputs derived from the input file shown above are displayed in the same order as they arrived. Note:

- The configuration inputs are output with a leading **TAB** (For example: `fmt.Printf("\tInit cylinder:  %5d\n", initialPosition)`)
- The *cylinder requests* are formatted with two leading **TAB**s. (For example: `fmt.Printf("\t\tCylinder %5d\n", requests)`)

- The servicing cylinder messages use **%5d** for the currently serviced cylinder number format specifier.
- The first four lines of output confirm the input disk configuration while the remaining lines of output are the individual cylinder requests.

Note: *The four configuration parameters are in the same order for all input files. That is the **seek algorithm**, the **lower**, **upper**, and **intitial** cylinders arrive in that order for every input file.*

3.3.3 Output

Generate the output formatted as shown below. Send it to **stdout**. Given that the input file is **sstf01.txt** shown above the output file is shown below:

```
Seek algorithm: SSTF
        Lower cylinder:     0
        Upper cylinder:  3000
        Init cylinder:     27
        Cylinder requests:
                Cylinder   300
                Cylinder   200
                Cylinder   100
                Cylinder   450
                Cylinder  2500
                Cylinder    38
Servicing    38
Servicing   100
Servicing   200
Servicing   300
Servicing   450
Servicing  2500
SSTF traversal count =  2473
```

3.3.4 Error messages

There are two exception categories, an **abort** which terminates program execution, and an **error** which recognizes invalid conditions, and continues to process other cylinder requests. The format for the **abort/error** messages are as follows:

- `fmt.Printf("ABORT(11):initial (%d) > upper (%d)\n", initial, UPPER)`
- `fmt.Printf("ABORT(12):initial (%d) < lower (%d)\n", initial, LOWER)`
- `fmt.Printf("ABORT(13):upper (%d) < lower (%d)\n", UPPER, LOWER)`
- `fmt.Printf("ERROR(15):Request out of bounds:  req (%d) > upper (%d) or < lower (%d)\n", request, UPPER, LOWER)`

3.4 Testing

3.4.1 Algorithmic testing

There are 3 baseline files for each of the six scheduling algorithms included in the **ZIP** file. (Additional error test files are also available.) The filename and their corresponding seek algorithms are shown in the table below:

Table 3.1: Algorithmic Test Schema

Shell Script	Purpose
chkFcfs.sh	Test the *first come, first served* seek algorithm **only**
chkSstf.sh	Test the *shortest seek time first* seek algorithm **only**
chkScan.sh	Test the *SCAN* seek algorithm **only**
chkCScan.sh	Test the *C-SCAN* seek algorithm **only**
chkLook.sh	Test the *LOOK* seek algorithm **only**
chkCLook.sh	Test the *C-LOOK* seek algorithm **only**
chkAll.sh	Tests all the seek algorithms shown above
error01.sh	Tests if requested address is > upper or < lower, produces an error if true, discards the request and continues to process pending requests
error02.sh	Tests if the initial address is < lower, produces an ABORT if true, thereby terminating the program
error03.sh	Tests if the initial address is > upper, produces an ABORT if true, thereby terminating the program
error04.sh	Tests if upper is < lower, produces an ABORT if true, thereby terminating the program
errorALL.sh	Tests all four error tests shown above
Results were obtained using the command: *bash ShellScriptName pa2.go*	

4 Submission instructions

You must submit this assignment in **Webcourses** as a source file upload. Note that all submissions will be via Webcourses. The submitted programs will be **tested** and **graded** on **Eustis**.

Make sure to include a comment at the top of your main source file that contains the following *Academic Integrity* statement (substitute your name and NID) - *"I [name] ([NID]) affirm that this program is entirely my own work and that I have neither developed my code with any another person, nor copied any code from any other person, nor permitted my code to be copied or otherwise used by any other person, nor have I copied, modified, or otherwise used programs*

created by others. I acknowledge that any violation of the above terms will be treated as academic dishonesty."

4.1 Grading

Scoring will be based on the following rubric:

Table 4.1: Grading Rubric

Deduction	Description
-100	Cannot compile on *eustis*
-100	**Cannot read input parameter(s) specified on command line**
-100	Cannot write output to `stdout`
- 90	The program does not run from the command line without error or produces no output.
- 70	The program compiles, runs, and outputs the input file, but crashes thereafter or produces no output.
- 15	No or seriously deficient **first-come first-served** output
- 15	No or seriously deficient **shortest seek time first** output
- 15	No or seriously deficient **SCAN** output
- 15	No or seriously deficient **C-SCAN** output
- 15	No or seriously deficient **LOOK** output
- 15	No or seriously deficient **C-LOOK** output
- 5	No data validation for cylinder addressing errors - per occurrence *There are four error conditions, discussed above. Each failed error condition wiil result in a 5 point deduction.*
- 5	Minor output discrepancies - per common occurrence: Input file confirmation output doesn't match input file commands Algorithm output has minor alignment issues, etc.
Start with 100 points and deduct per the schedule above	