Operating system

Table of Contents

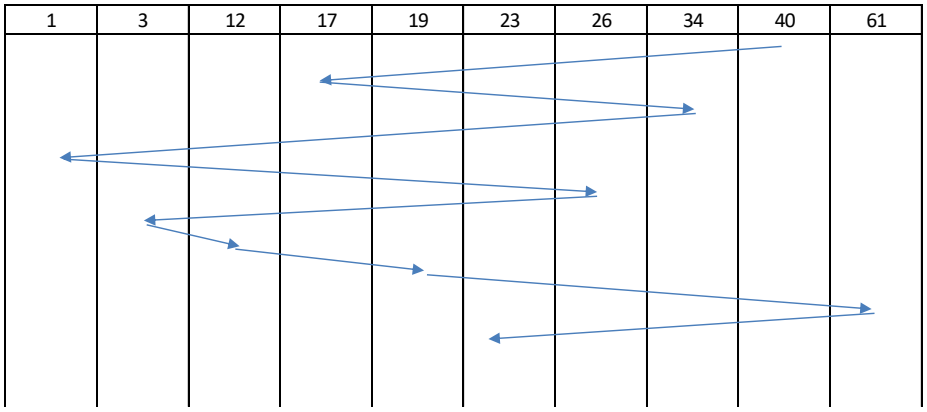
Question 1. a.

The given queue of track requests is,

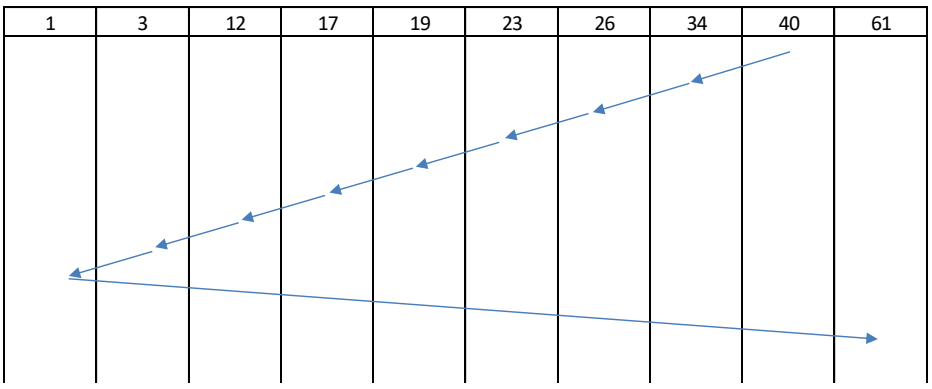
17	34	1	26	3	12	19	61	23

The head starts at the track 40.

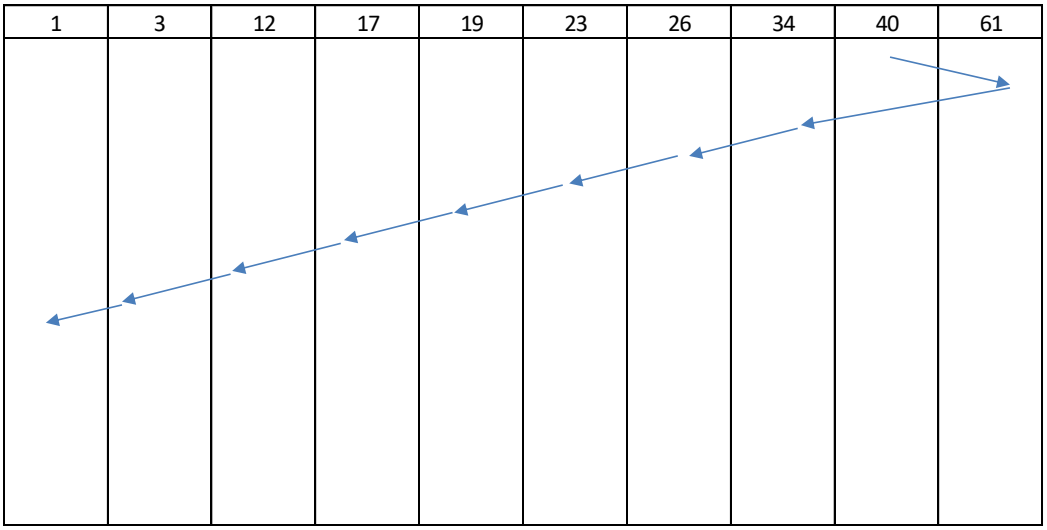
i) The arm movement for the FCFS or first come first serve basis is given below. It starts from the track 40 and then serves the queue as the request comes.



ii) The arm movement for SSTF or shortest seek time first scheme is,



iii) The arm movement for the LOOK scheme is given below. It is assumed that it starts from 40 and goes to the right first.



Question 1. a.

i) For the FCFS scheme, the total number of tracks travelled by the head is,

(40 - 17) + (34 – 17) + (34 - 1) + (26 – 1) + (26 – 3) + (12 – 3) + (19 – 12) + (61 – 19) + (61 – 23) = 217

There are 9 requests, so the average number of tracks travelled by the head is, $\left\lceil \frac{217}{9} \right\rceil = 25$

ii) For the SSTF scheme the total number of tracks travelled by the head are on a direction only. So, the total number of tracks travelled is calculated as, (40 – 1) + (61 – 1) = 39 + 60 = 99

There are 9 requests, so the average number of tracks travelled by the head is, $\left\lceil \frac{99}{9} \right\rceil = 11$

iii) For the LOOK scheme, the total number of tracks travelled by the head are directional. So, it is calculated as, (61 – 40) + (61 -1) = 21 + 60 = 81

There are 9 requests, so the average number of tracks travelled by the head is, $\left\lceil \frac{81}{9} \right\rceil = 9$

Question 2.a.

In the operating systems, a deadlock is a condition when a set of processes are blocked as each of the processes hold some resource that is required by any of the other processes and vice versa. So, each of the process wait for infinite time to get the resource released by the other processes. It needs at least to processes and two resources to form a deadlock condition.

The four conditions of deadlock are listed below. A deadlock condition can be avoided if any of these four conditions is not met. It needs all these four conditions to be held simultaneously to form a deadlock condition. For example, if P1 and P2 processes are in a deadlock condition as P1 is holding a resource instance R1 that is required by P2. And P2 is holding a resource instance R2 that is needed by P1. So, no process will be able to release the resources and there will be a deadlock [1].

1. **Mutual Exclusion**

The resources held by the processes must be non-shareable. As, the other processes must not be able to acquire the resources. For example, the resource instances R1 and R2 held by the processes P1 and P2 must be atomic or non-shareable. So, the processes P1 and P2 will be prevented from using the resources when needed.

2. **Partial Allocation or Hold and Wait**

The processes must hold the at least partial resources that is been allocated to some other process already and the processes are waiting for the rest of the resources. If it can release the already held resource, then there will be no deadlock as the same resource is requested by another

process and that process will complete the task by acquiring it. So, there will be no deadlock situation. For example, if P1 is not holding R1 and waiting for R2 or P1 is able to release R1 then P2 will acquire R1 as its waiting for that resource. When P2 gets two of the resources allocated to it, then it will finish the task and will release both resources, then P1 can get both and complete the task. So, there will be no deadlock. Hence, it needs to acquire partial resources and waiting for the rest of the resources that is acquired by another process that is also waiting for the resource acquired by the first process.

3. **No Pre-Emption**

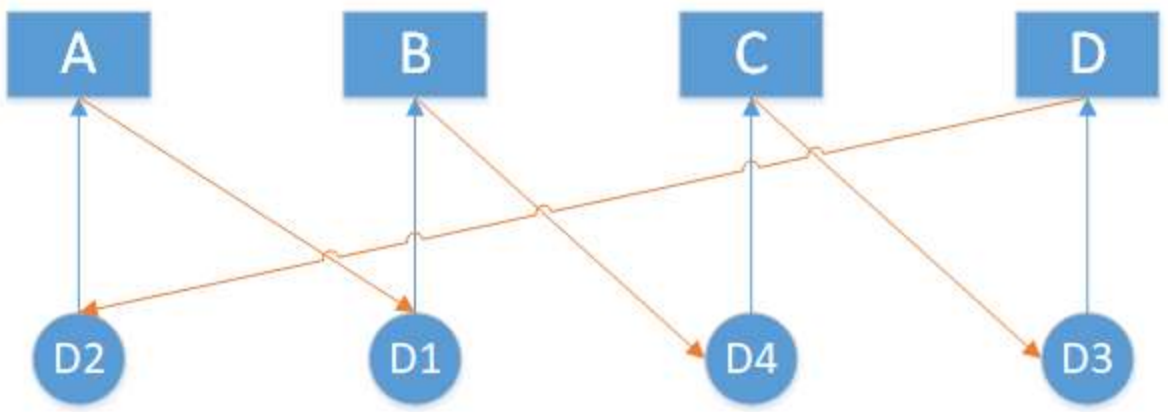
There must be no pre-emption. That is, no process will be able to release a resource allocated to it, without completing the task. Or in other words, the system must not be able to take a resource being used by a process. Then there will be no deadlock as it will assign the resource to another process to break the deadlock. For example, if there is pre-emption, then the system would take either R1 from P1 and will give it to P2 to complete the task of P2. Then the deadlock will be broken.

4. **Circular Waiting or Resource Waiting**

A circular chain of processes cannot exist in a deadlock that is each process is holding some resources that has been requested by the next process in the chain. According to the cycle theorem, if there is a cycle in the resource graph, then that is necessary for a deadlock to happen. If there is a chain of processes P1, P2, P3… and resources R1, R2, R3… in such way that P1 holds R1 and P2 is waiting for R2 and so on. Then for the last process Pn The resource will be requested by P1. And such state cant occur.

Question 2.b.

Using 4 processes and 4 instances of resources, the following deadlock situation has been created.



Process A is holding D2 device and has requested for D1. D1 is held by process B and process B has requested for D4, D4 is held by process C and process C has requested for D3. D3 is currently held by process D and process D has requested for D2 that is held by process A.

So, there is a deadlock.

Question 2.c.

I) A record format helps to understand the organisation of records in a file. In many operating systems, different file formats are used. Fixed-length and variable-length records are examples of two such formats to organise records in a file.

Fixed-length format

There are predetermined number of columns to arrange the records and there is an end-of record marker after that predetermined number of columns [1]. For example, the record length is 80 columns.

Field 1	Field 2	Field 3
Field 1	Field 2	Field 3
Field 1	Field 2	Unused Space

Each row represents a record. Each record has same length fields.

Variable-length format

The length of a record is not restricted pre-deterministically. The end-of-record market is available after the last field of a record. Same is true for each record in a file.

Field 1	Field 2	Field 3
Field 1	Field 2	Field 3
Field 1	Field 2	

Each row represents a record. Each record has variable length fields.

II) The relative file name is when it is with the respect of the root or the source of the file. It includes the path of the file from the root. For example, /var/home/User1/picture.png

And on the other, the absolute file name is the name of the file that is the picture.png. No filepath is attached to this name.

Question 3.a.

- chmod

It is used to change permissions related to the files and directories. It is used in Linux, Unix and Unix-like operating systems. The full form is change mode.

The syntax is,

```
chmod [options] permissions file name
```

Options may not be specified in basic syntax. The permissions set for group or single user will be applied to the file as defined by the filename field [2].

An example is,

```
chmod u=rw,g=r,o=r myfile
```


The file myfile will have permission read and write for user, read for groups and read for others.

```
[root@localhost ~]# chmod -help
BusyBox v1.24.2 (2017-05-25 17:33:59 CEST) multi-call binary.

Usage: chmod [-Rcvf] MODE[,MODE]... FILE...

Each MODE is one or more of the letters ugoa, one of the
symbols +-= and one or more of the letters rwxst

        -R      Recurse
        -c      List changed files
        -v      List all files
        -f      Hide errors
[root@localhost ~]#
```

And,

```
[root@localhost mydir]# chmod 745 myfile.txt
[root@localhost mydir]# ls
myfile.txt
[root@localhost mydir]# ls -l
total 4
-rwxr--r-x    1 root     root            46 Sep 24 18:03 myfile.txt
[root@localhost mydir]#
```

- grep

It is used to search for some patterns in a file. It then displays all instances found by it. Grep stand for globally search for regular expression and print out.

The syntax is, `grep [options] pattern [files]`
An use is, `$grep -c "a" myfile.txt`
It will print all instances of a in the myfile.txt.

```
[root@localhost mydir]# grep -c "a" myfile.txt
1
[root@localhost mydir]#
```

And, grep is case sensitive during search. Following screenshot supports that.

```
[root@localhost mydir]# cat >myfile.txt
This is a sunny day. There is no sign of rain
^Z[1]+  Stopped                 cat 1>myfile.txt
[root@localhost mydir]# ls -l
total 4
-rw-r--r--    1 root     root            46 Sep 24 18:03 myfile.txt
[root@localhost mydir]# grep -c "a" myfile.txt
1
[root@localhost mydir]# grep -help
^Z[2]+  Stopped                 grep -help
[root@localhost mydir]#
[root@localhost mydir]# grep -c "Sunny" myfile.txt
0
[root@localhost mydir]# grep -c "sunny" myfile.txt
1
[root@localhost mydir]#
```

- ls

It is used to list all files found in the OS. There are parameters to restrict the output. By default, it lists all files in the current directory.

The syntax is, `ls -[option]`

An example, `$ ls -l`

Where l stands for long format.

```
[root@localhost mydir]# ls -help
total 0
[root@localhost mydir]#
```

After creation of the myfile.txt,

```
[root@localhost mydir]# cat >myfile.txt
This is a sunny day. There is no sign of rain
^Z[1]+  Stopped                 cat 1>myfile.txt
[root@localhost mydir]# ls -l
total 4
-rw-r--r--    1 root     root           46 Sep 24 18:03 myfile.txt
[root@localhost mydir]#
```

- mkdir

It is used to make a new directory. The full form is Make Directory.

The basic syntax is, `mkdir [directory name]`

An example is, `$ mkdir mydir`

```
[root@localhost ~]# mkdir mydir
[root@localhost ~]# chdir mydir
[root@localhost mydir]#
```

And

```
[root@localhost ~]# mkdir mydir
[root@localhost ~]# chdir mydir
[root@localhost mydir]# mkdir -help
mkdir: invalid option -- h
BusyBox v1.24.2 (2017-05-25 17:33:59 CEST) multi-call binary.

Usage: mkdir [OPTIONS] DIRECTORY...

Create DIRECTORY

        -m MODE Mode
        -p      No error if exists; make parent directories as needed
[root@localhost mydir]#
```

References

[1] A. Silberschatz, G. Gagne, and P. B. Galvin, *Operating System Concepts*. Wiley, 2018.

[2] E. J. Ray and D. S. Ray, *Unix and Linux: Visual QuickStart Guide*. Pearson Education, 2014.