Introduction to Operating Systems and SQL for Data Science

Practice 6 – Operating Systems Summary

Subjects we covered

- Processes
- Threads
- Scheduling
- Memory Management
- File Systems

Reichman University

Question #1

- Given a batch system with FIFO scheduling, and the following processes:

Process	Arrival time	Work time
A	0	4
B	3	2
C	5	15
D	18	1

- **Compute TA and Starvation time for each process**

Question #1

Solution:

Process	Arrival time	Work time
A	0	4
B	3	2
C	5	15
D	18	1

Arrival time	Work time	Start	End	TA	Starvation
0	4	0	4	4	0

Question #1

Solution:

Process	Arrival time	Work time
A	0	4
B	3	2
C	5	15
D	18	1

Arrival time	Work time	Start	End	TA	Starvation
0	4	0	4	4	0
3	2	4	6	3	1

Question #1

Solution:

Process	Arrival time	Work time
A	0	4
B	3	2
C	5	15
D	18	1

Arrival time	Work time	Start	End	TA	Starvation
0	4	0	4	4	0
3	2	4	6	3	1
5	15	6	21	16	1

Question #1

Solution:

Process	Arrival time	Work time
A	0	4
B	3	2
C	5	15
D	18	1

Arrival time	Work time	Start	End	TA	Starvation
0	4	0	4	4	0
3	2	4	6	3	1
5	15	6	21	16	1
18	1	21	22	4	3

Question #2

- Given a batch system with non-preemptive scheduling and processes with the following work time 17,5,2,18,53,3 (arrival from left to right)

  Compute TA sum when using:

- **Shortest job first**
- **First come first served**

Reichman University

Question #2

- Given a batch system with non-preemptive scheduling and processes with the following work time 17,5,2,18,53,3 (arrival from left to right)

  Compute TA sum when using:

- **<u>Solution:</u>**
- **Shortest job first**
  2+5+10+27+45+98 = 187
- **First come first served**
  27+22+24+42+95+98 = 308

Reichman University

Question #3

- Given a system with 5 processes, pid = 1,2,3,4,5 and for each process I the process $\frac{1}{i+1}$ busy with IO operations. **Compute CPU Utilization**

Reichman University

Question #3

- Given a system with 5 processes, pid = 1,2,3,4,5 and for each process I the process $\frac{1}{i+1}$ busy with IO operations. **Compute CPU Utilization**

<u>Solution:</u>

$$1 - \left(\frac{1}{2} * \frac{1}{3} * \frac{1}{4} * \frac{1}{5} * \frac{1}{6}\right) = 0.998$$

Reichman University

Question #4

Given a computer network connected to each other, each computer can be in 2 modes: broadcasting or receiving. When a computer broadcasts it sends one message throw all the connections it has, when a computer is in receiving mode, it gets messages from all its neighbors. To avoid collision between 2 messages on same communication line a computer should “own” the communication line before it broadcasts, if the line already taken it waits for it to be released.

Reichman University

Question #4

In all computers the following code responsible for broadcasting:

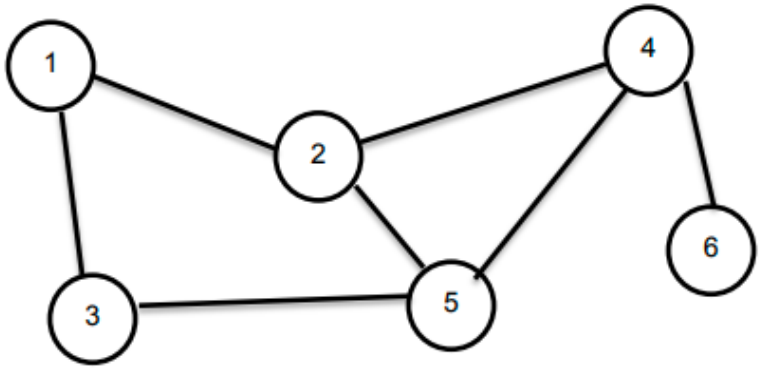
```
while(true){
//receive messages from all neighbors
  List<Message> messages = ReceiveMessages(CommunicationLines);
//broadcast all new messages to all neighbors
  foreach(Message m in messages){
    if(NewMessage(m)){//broadcast only new messages
    //grab ownership on all outgoing communication lines
      foreach(CoomunicationLine l in CommunicationLines)
        l.AcquireOwnership();//blocks until line is free
    //broadcast the message
      foreach(CoomunicationLine l in CommunicationLines)
        l.Broadcast(m);//line must be owned before broadcasting
    }
  }
}
```

Reichman University

Question #4

- Each computer receives messages from all its neighbors, then the computer sends the messages it did not send before and for each message it should broadcast, it takes ownership over connected communication lines and sends the message.

Given the following network:



Is there a scenario for deadlock – if not, explain if yes, describe it.

Reichman University

Question #4

Solution:

There might be a deadlock in here: we will take a circle of communication 2,4,5

- PC 2 owns line 2-4
- PC 4 owns line 4-5
- PC 5 owns line 5-2
- Now each PC tries to own the other line in the circle and waits because the line is already owned => **deadlock!**

Reichman University

Question #4

- **Ron said that if we will number the communication lines in global increasing order we can avoid deadlocks by a simple change in the code, explain what change should be done?**

Reichman University

Question #4

- **Ron said that if we will number the communication lines in global increasing order we can avoid deadlocks by a simple change in the code, explain what change should be done?**

<u>Solution:</u>

If each thread will require the communication lines in ascending order (lines 5-6), there will no circle created and therefore no deadlock.

Reichman University

Question #5

- **Given a process with the following memory approaches: 1,2,3,4,1,2,3,4,5**
- **Consider initial memory is empty.**

Reichman University

Question #5

Compute number of page faults when using:

a. LRU paging and number of frames in memory 3:

Page number	Hit/PF	Memory after action
1		
2		
3		
4		
1		
2		
3		
4		
5		

Reichman University

Question #5

Compute number of page faults when using:

a. LRU paging and number of frames in memory 3:

Page number	Hit/PF	Memory after action
1	PF	1
2		
3		
4		
1		
2		
3		
4		
5		

Reichman University

Question #5

Compute number of page faults when using:

a. LRU paging and number of frames in memory 3:

Page number	Hit/PF	Memory after action
1	PF	1
2	PF	1 2
3		
4		
1		
2		
3		
4		
5		

Reichman University

Question #5

Compute number of page faults when using:

a. LRU paging and number of frames in memory 3:

Page number	Hit/PF	Memory after action
1	PF	1
2	PF	1 2
3	PF	1 2 3
4		
1		
2		
3		
4		
5		

Reichman University

Question #5

Compute number of page faults when using:

a. LRU paging and number of frames in memory 3:

Page number	Hit/PF	Memory after action
1	PF	1
2	PF	1 2
3	PF	1 2 3
4	PF	2 3 4
1		
2		
3		
4		
5		

Reichman University

Question #5

Compute number of page faults when using:

a. LRU paging and number of frames in memory 3:

Page number	Hit/PF	Memory after action
1	PF	1
2	PF	1 2
3	PF	1 2 3
4	PF	2 3 4
1		
2		
3		
4		
5		

Reichman University

Question #5

Compute number of page faults when using:

a. LRU paging and number of frames in memory 3:

Page number	Hit/PF	Memory after action
1	PF	1
2	PF	1 2
3	PF	1 2 3
4	PF	2 3 4
1	PF	3 4 1
2		
3		
4		
5		

Reichman University

Question #5

Compute number of page faults when using:

a. LRU paging and number of frames in memory 3:

Page number	Hit/PF	Memory after action
1	PF	1
2	PF	1 2
3	PF	1 2 3
4	PF	2 3 4
1	PF	3 4 1
2	PF	4 1 2
3		
4		
5		

Reichman University

Question #5

Compute number of page faults when using:

a. LRU paging and number of frames in memory 3:

Page number	Hit/PF	Memory after action
1	PF	1
2	PF	1 2
3	PF	1 2 3
4	PF	2 3 4
1	PF	3 4 1
2	PF	4 1 2
3	PF	1 2 3
4		
5		

Reichman University

Question #5

Compute number of page faults when using:

a. LRU paging and number of frames in memory 3:

Page number	Hit/PF	Memory after action
1	PF	1
2	PF	1 2
3	PF	1 2 3
4	PF	2 3 4
1	PF	3 4 1
2	PF	4 1 2
3	PF	1 2 3
4	PF	2 3 4
5		

Reichman University

Question #5

Compute number of page faults when using:

a. LRU paging and number of frames in memory 3:

Page number	Hit/PF	Memory after action
1	PF	1
2	PF	1 2
3	PF	1 2 3
4	PF	2 3 4
1	PF	3 4 1
2	PF	4 1 2
3	PF	1 2 3
4	PF	2 3 4
5	PF	3 4 5

Reichman University

Question #5

Compute number of page faults when using:

b. LRU paging and number of frames in memory 4:

Page number	Hit/PF	Memory after action
1		
2		
3		
4		
1		
2		
3		
4		
5		

Reichman University

Question #5

Compute number of page faults when using:

b. LRU paging and number of frames in memory 4:

Page number	Hit/PF	Memory after action
1	PF	1
2		
3		
4		
1		
2		
3		
4		
5		

Reichman University

Question #5

Compute number of page faults when using:

b. LRU paging and number of frames in memory 4:

Page number	Hit/PF	Memory after action
1	PF	1
2	PF	1 2
3		
4		
1		
2		
3		
4		
5		

Reichman University

Question #5

Compute number of page faults when using:

b. LRU paging and number of frames in memory 4:

Page number	Hit/PF	Memory after action
1	PF	1
2	PF	1 2
3	PF	1 2 3
4		
1		
2		
3		
4		
5		

Reichman University

Question #5

Compute number of page faults when using:

b. LRU paging and number of frames in memory 4:

Page number	Hit/PF	Memory after action
1	PF	1
2	PF	1 2
3	PF	1 2 3
4	PF	1 2 3 4
1		
2		
3		
4		
5		

Reichman University

Question #5

Compute number of page faults when using:

b. LRU paging and number of frames in memory 4:

Page number	Hit/PF	Memory after action
1	PF	1
2	PF	1 2
3	PF	1 2 3
4	PF	1 2 3 4
1	Hit	2 3 4 1
2		
3		
4		
5		

Reichman University

Question #5

Compute number of page faults when using:

b. LRU paging and number of frames in memory 4:

Page number	Hit/PF	Memory after action
1	PF	1
2	PF	1 2
3	PF	1 2 3
4	PF	1 2 3 4
1	Hit	2 3 4 1
2	Hit	3 4 1 2
3		
4		
5		

Reichman University

Question #5

Compute number of page faults when using:

b. LRU paging and number of frames in memory 4:

Page number	Hit/PF	Memory after action
1	PF	1
2	PF	1 2
3	PF	1 2 3
4	PF	1 2 3 4
1	Hit	2 3 4 1
2	Hit	3 4 1 2
3	Hit	4 1 2 3
4		
5		

Reichman University

Question #5

Compute number of page faults when using:

b. LRU paging and number of frames in memory 4:

Page number	Hit/PF	Memory after action
1	PF	1
2	PF	1 2
3	PF	1 2 3
4	PF	1 2 3 4
1	Hit	2 3 4 1
2	Hit	3 4 1 2
3	Hit	4 1 2 3
4	Hit	1 2 3 4
5		

Reichman University

Question #5

Compute number of page faults when using:

b. LRU paging and number of frames in memory 4:

Page number	Hit/PF	Memory after action
1	PF	1
2	PF	1 2
3	PF	1 2 3
4	PF	1 2 3 4
1	Hit	2 3 4 1
2	Hit	3 4 1 2
3	Hit	4 1 2 3
4	Hit	1 2 3 4
5	PG	2 3 4 5

Reichman University

Question #6

- Given a system with i-node file management
- Block size is 256 bytes, word size is 2 bytes. A block pointer size is one word. There is one single-indirect pointer and one double-indirect pointer and 64 bytes are needed for file's metadata, rest are direct pointers.
- How much information can be saved in a file in this system?

Reichman University

Solution

- How much direct pointers there are ?
  256-64-2-2=188 bytes, pointer size is 2B so there are 94 directed pointers.
- How many pointers in an empty block? 256/2 = 128
- Hoy many data blocks? ? $94 + 1 * 128 + 1 * 128^2 = 16606$
- Block size is 256B so total: 16606 * 256 = 4251136 bytes

Reichman University