Section A

A1.

1)

Ready

-The process is currently waiting to be assigned a processor by scheduler to start running. This state starts immediately after new state for the process.

Running

- It means that the process’ instructions are being executed by a processor. This state starts immediately after the process is assigned a processor by the scheduler

Created

- The process has just been created and will be queued to be put into ready state by being assigned memory.

2)

I would merge the Created and Ready states as after being created it only need to be assigned memory to transition to ready state.

A2.

1)

- Processes takes more time and resources to start than threads.

- Threads share resources with each other which provides efficient communication

- Threads provide multiprogramming in a process, allowing it to utilize more resources and achieve efficiency.

2)

A thread shouldn’t cost any extra resources as it is a part of the process it is created in. A process needs to be assigned a processor and memory while threads are just a part of the process and will utilize the given resources to the processor.

A3.

* Page tables can be very large due to having the user space, so having a multi-level decreases indexing times.
* It also reduces the size of the page table in physical memory.

A4.

* More flexible for file sizes as they can be increased easily.
* No external fragmentation due to the growth characteristic of files in this allocation method.

A5.

Long term scheduler

* Responsible for deciding what processes are created from the job queue and putting them into the ready state.

Short term scheduler

* Responsible for deciding when a process would be put into running state with the given resources.

Section B

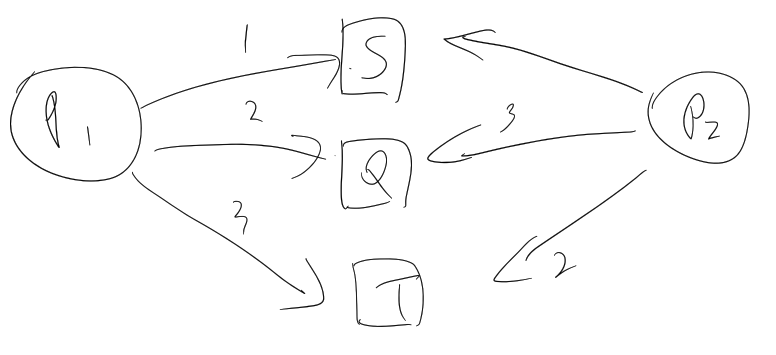
B1.

1)

No, as the processes take a proper order of turns to hold-and-wait via semaphores (mutual exclusion) and doesn’t end up with a circular wait.

2)

Yes, as they do not take a proper order of turns. See graph for explanation and visualization.



P1 will access Q before P2, and P2 will access T before Q, which will result in a circular wait scenario, and thus a deadlock.

B2.

1)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SSTF | 131 | 85 | 913 | 948 | 1022 | 1470 | 1509 | 1750 | 1764 |  |  |
| SCAN | 913 | 948 | 1022 | 1470 | 1509 | 1750 | 1764 | 2999 | 131 | 85 |  |
| C-SCAN | 913 | 948 | 1022 | 1470 | 1509 | 1750 | 1764 | 2999 | 0 | 85 | 131 |

|  |  |
| --- | --- |
|  | Total Number of Tracks |
| SSTF | (133-131)+(131-85)+(913-85)+(948-913)+…. = 1681 |
| SCAN | (2999-133)+(2999-85)=5780 |
| C-SCAN | (2999-133)+(2999)+131=5998 |

B3.

a)

SJF

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Arrival | A |  | B | C |  | D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Time | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| Current | A | A | A | A | A | A | A | A | A | A | A | A | B | B | B | B | B | B | B | B | C |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Arrival |  |  |  |  |  |  |  |  |  |
| Time | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| Current | C | C | C | C | D | D | D | D |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | B | C | D |
| Arrival | 0 | 2 | 3 | 5 |
| Burst | 12 | 8 | 5 | 4 |
| Start | 0 | 12 | 20 | 25 |
| Complete | 12 | 20 | 25 | 29 |
| Turnaround time | 12 | 18 | 22 | 24 |
| Waiting time | 0 | 10 | 17 | 20 |
| Response time | 0 | 10 | 17 | 20 |

SRTF

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Arrival | A |  | B | C |  | D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Time | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| Current | A | A | B | C | C | C | C | C | D | D | D | D | B | B | B | B | B | B | B | A | A |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Arrival |  |  |  |  |  |  |  |  |  |
| Time | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| Current | A | A | A | A | A | A | A | A |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | B | C | D |
| Arrival | 0 | 2 | 3 | 5 |
| Burst | 12 | 8 | 5 | 4 |
| Start | 0 | 2 | 3 | 8 |
| Complete | 29 | 19 | 8 | 12 |
| Turnaround time | 29 | 17 | 5 | 7 |
| Waiting time | 17 | 9 | 0 | 3 |
| Response time | 0 | 0 | 0 | 3 |

RR

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Arrival | A |  | B | C |  | D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Time | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| Current | A | A | B | B | A | A | C | C | B | B | D | D | A | A | C | C | B | B | D | D | A |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Arrival |  |  |  |  |  |  |  |  |  |
| Time | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| Current | A | C | B | B | A | A | A | A |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | B | C | D |
| Arrival | 0 | 2 | 3 | 5 |
| Burst | 12 | 8 | 5 | 4 |
| Start | 0 | 2 | 6 | 10 |
| Complete | 29 | 25 | 23 | 20 |
| Turnaround time | 29 | 23 | 20 | 15 |
| Waiting time | 17 | 15 | 15 | 11 |
| Response time | 0 | 0 | 3 | 5 |

B4.

1)

Given that an entry is 4 bytes

2)

3) Since there is 2^5 bytes per page, the offset will be 5 bits.

4) 19 \* (5+8) = 247 bits

5) 20 \* (5+8) = 260, so roughly 9 bits

7) I would use a multi-level paging table where the root would have entries to user page tables that would point to user addresses , thus reduce paging table size and indexing time based on search sectors.

B5.

1) No,

Eg. Lock(0) needs Unlock(1) to be called due to the line “while(flag[1-self] == true && turn == 1-self)”

2) Yes it is possible, because they are processes and not threads, meaning that they do not share their resources to each other.

3) Yes, as we cannot determine the speed at which each process runs, it is the system it is running on that will decide that.

4) No, functions each could have potential starvation.

B6.

1)

FIFO

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  |  | 0 | 1 | 0 | 2 | 3 | 0 | 4 | 1 |
|  |  | 0 | 1 | 1 | 2 | 3 | 0 | 4 | 1 |
|  |  |  | 0 | 0 | 1 | 2 | 3 | 0 | 4 |
|  |  |  |  |  | 0 | 1 | 2 | 3 | 0 |
| hit |  | no | no | yes | no | no | no | no | no |

Hit rate = 1/8

Fault rate = 7/8

LRU

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  |  | 0 | 1 | 0 | 2 | 3 | 0 | 4 | 1 |
|  |  | 0 | 1 | 0 | 2 | 3 | 0 | 4 | 1 |
|  |  |  | 0 | 1 | 0 | 2 | 3 | 0 | 4 |
|  |  |  |  |  | 1 | 0 | 2 | 3 | 0 |
| hit |  | no | no | yes | no | no | yes | no | no |

Hit rate = 2/8

Fault rate = 6/8

2)

By doing it in order of page number with below reference address

0 0 0 1 1 2 3 4

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  |  | 0 | 0 | 0 | 1 | 1 | 2 | 3 | 4 |
|  |  | 0 | 0 | 0 | 1 | 1 | 2 | 3 | 4 |
|  |  |  |  |  | 0 | 0 | 1 | 2 | 3 |
|  |  |  |  |  |  |  | 0 | 1 | 2 |
| hit |  | no | yes | yes | no | yes | no | no | no |

Hit rate = 3/8 which is better than the LRU hit rate of 2/8