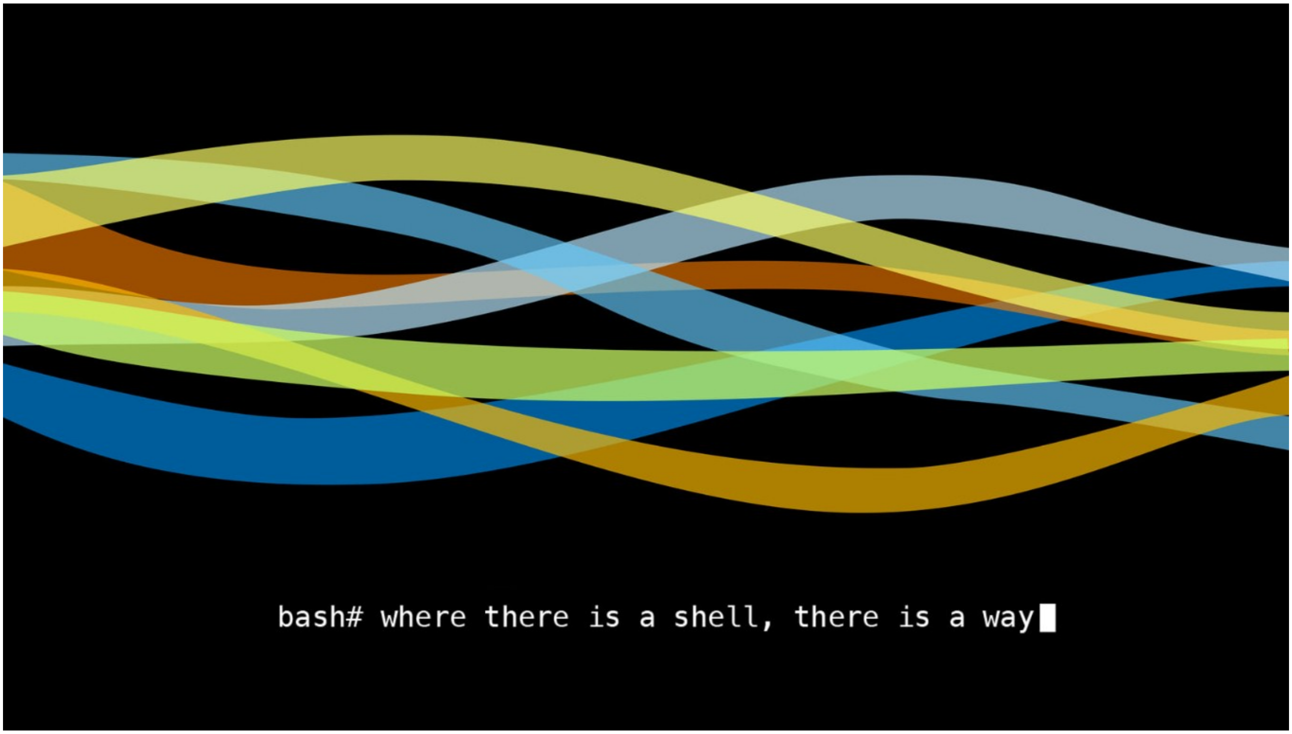
# Operating Systems

Final 2020 Solution




Saad Bin Khalid 20K-0161 31 May 2022

# Question 1

Answer the following questions

- What are the drawbacks of many to one threading model?

Many-to-one threading model maps many user-level threads to one kernel thread. The disadvantages are listed below.

  - ➔ The entire process will block if a thread makes a blocking system call.
  - ➔ Because only one thread can access the kernel at a time, multiple threads are unable to run in parallel on multicore systems.
  - ➔ It is unable to take advantage of multicore system.

- Differentiate between Monolithic and Microkernel operating system structure?

| **Monolithic** | **Microkernel** |
|---|---|
| All functionality of the kernel is in a single binary file. | Only the necessary components of kernel are compiled in a single file. |
| Little performance overhead. | Large overhead because communication is provided through message-passing mechanism. |
| Difficult to implement and extend. | Easy to extend. |

- The problem arises when a lower-priority process holds a lock needed by higher-priority process. How this problem can be resolved? Explain.

This problem is known as *priority inversion.* It is avoided by implementing a priority-inheritance protocol. According to this protocol, all processes that are accessing resources needed by a higher-priority process inherit the higher priority until they are finished with the resources. When they are finished, their priorities revert to their original values.

- Differentiate between user and kernel level threads, giving the pros and cons of each?

| **User Threads** | **Kernel Threads** |
|---|---|
| Implemented by user. | Implemented by kernel. |
| Easy implementation. | Difficult implementation. |
| Requires no hardware support. | Requires hardware support. |
| If one thread is blocked, entire process will be blocked. | If one thread is blocked, another thread can continue execution. |

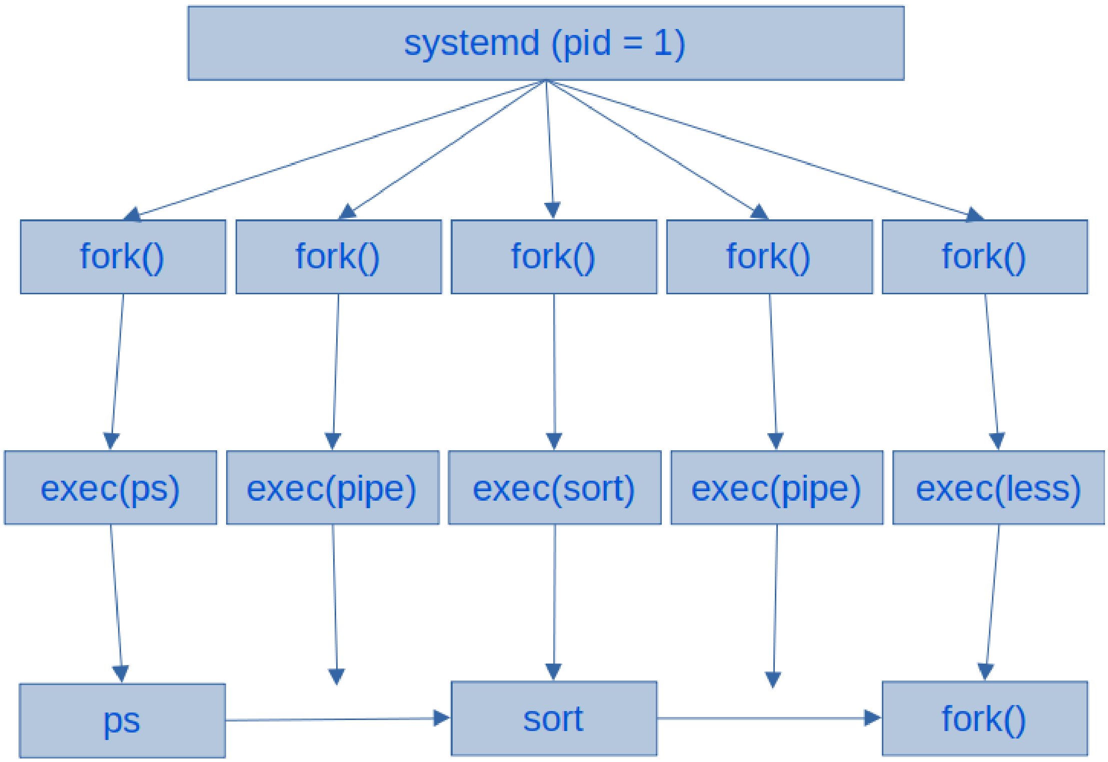
- How would you differentiate between data parallelism and task parallelism?

| Data parallelism | Task parallelism |
|---|---|
| Subsets of data are distributed on cores. | Tasks are distributed on cores. |
| Same operation is performed on each core. | Unique operation is performed on each core. |
| For example, summing the contents of a large array. | For example, performing different statistical operations on same array. |

# Question 2

Draw block diagram using fork( ) and exec ( ) for the following concatenated commands.

**ps | sort | less**



# Question 3

A single-lane bridge connects the two Vermont villages of North Tunbridge and South Tunbridge. Farmers in the two villages use this bridge to deliver their produce to the neighboring town. The bridge can become deadlocked if a northbound and a southbound farmer get on the bridge at the same time. (Vermont farmers are stubborn and are unable to back up.) Using semaphores and/or mutex locks, design an algorithm in pseudo code that prevents deadlock. Initially, do not be concerned about starvation (the situation in which northbound farmers prevent southbound farmers from using the bridge or vice versa).

```
mutex = Mutex()

southbound_farmer() {
     acquire(mutex)
     cross_bridge()
     release(mutex)
}

northbound_farmer() {
     acquire(mutex)
     cross_bridge()
     release(mutex)
}
```

# Question 4

Consider the following set of processes, with the length of the CPU burst given in milliseconds: Draw Gantt charts that illustrate the execution of these processes. Calculate average turnaround time, and average waiting time using the following scheduling algorithms:

| Process | Arrival | Burst | Priority |
|---|---|---|---|
| $P_1$ | 0 | 9 | 5 |
| $P_2$ | 1 | 4 | 3 |
| $P_3$ | 2 | 5 | 1 |
| $P_4$ | 3 | 7 | 2 |
| $P_5$ | 4 | 3 | 4 |

I. Preemptive Priority (Consider the lowest integer as a high priority)

| $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_2$ | $P_5$ | $P_1$ |
|---|---|---|---|---|---|---|

0 1 2 7 14 17 20 28

| Process | Turn-around time | Waiting time |
|---|---|---|
| $P_1$ | 28 - 0 = 28 | 28 - 9 = 19 |
| $P_2$ | 17 - 1 = 16 | 16 - 4 = 12 |
| $P_3$ | 7 - 2 = 5 | 5 - 5 = 0 |
| $P_4$ | 14 - 3 = 11 | 11 - 7 = 4 |
| $P_5$ | 20 - 4 = 16 | 16 - 3 = 13 |
| **Average** | 15.2 ms | 9.6 ms |

II. Round-Robin (quantum = 3)

| $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_1$ | $P_4$ |
|---|---|---|---|---|---|---|---|---|---|---|

0 3 6 9 12 15 18 19 21 24 27 28

| Process | Turn-around time | Waiting time |
|---|---|---|
| $P_1$ | 27 - 0 = 27 | 27 - 9 = 18 |
| $P_2$ | 19 - 1 = 18 | 18 - 4 = 14 |
| $P_3$ | 21 - 2 = 19 | 19 - 5 = 14 |
| $P_4$ | 28 - 3 = 25 | 25 - 7 = 18 |
| $P_5$ | 15 - 4 = 11 | 11 - 3 = 8 |
| **Average** | 20 ms | 14.4 ms |

III. Shortest Remaining Time First

| $P_1$ | $P_2$ | $P_5$ | $P_3$ | $P_4$ | $P_1$ |
|---|---|---|---|---|---|

0 1 5 8 13 20 28

| Process | Turn-around time | Waiting time |
|---|---|---|
| $P_1$ | 28 - 0 = 28 | 28 - 9 = 19 |
| $P_2$ | 5 - 1 = 4 | 4 - 4 = 0 |
| $P_3$ | 13 - 2 = 11 | 11 - 5 = 6 |
| $P_4$ | 20 - 3 = 17 | 17 - 7 = 10 |
| $P_5$ | 8 - 4 = 4 | 4 - 3 = 1 |
| **Average** | 12.8 ms | 7.2 ms |

# Question 5

Consider the following page reference string: **1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6**
How many page faults and page hits would occur for the following replacement algorithms, assuming three frames? Remember that all frames are initially empty, so your first unique pages will cost one fault each.

I. LRU replacement

| 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 4 | 4 | 4 | 5 | 5 | 5 | 1 | 1 | 1 | 7 | 7 | 7 | 2 |
| | 2 | 2 | 2 | 2 | 2 | 2 | 6 | 6 | 6 | 6 | 3 | 3 | 3 | 3 | 3 |
| | | 3 | 3 | 3 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 6 | 6 | 6 |
| * | * | * | * | ! | * | * | * | * | * | ! | * | * | * | ! | * |

| 1 | 2 | 3 | 6 |
|---|---|---|---|
| 2 | 2 | 2 | 2 |
| 3 | 3 | 3 | 3 |
| 1 | 1 | 1 | 6 |
| * | ! | ! | * |

**Hits = 5**
**Faults = 15**

## II. Optimal replacement

| 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 3 | 3 | 3 | 3 |
| | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 7 | 7 | 7 | 2 |
| | | 3 | 4 | 4 | 4 | 5 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| * | * | * | * | ! | ! | * | * | ! | ! | ! | * | * | ! | ! | * |

| 1 | 2 | 3 | 6 |
|---|---|---|---|
| 3 | 3 | 3 | 6 |
| 2 | 2 | 2 | 2 |
| 1 | 1 | 1 | 1 |
| * | ! | ! | * |

**Hits = 9**
**Faults = 11**

## III. Second Chance

| 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 1 | 0 1 | 0 4 | 0 4 | 0 4 | 0 5 | 0 5 | 0 5 | 0 1 | 0 1 | 0 1 | 0 7 | 0 7 | 0 7 | 0 2 |
| | 0 2 | 0 2 | 0 2 | 1 2 | 0 2 | 0 2 | 0 6 | 0 6 | 0 6 | 0 6 | 0 3 | 0 3 | 0 6 | 0 6 | 0 6 |
| | 0 | 0 3 | 0 3 | 0 3 | 0 1 | 0 1 | 0 1 | 0 2 | 0 2 | 1 2 | 1 2 | 0 2 | 0 2 | 0 3 | 0 3 |
| * | * | * | * | ! | * | * | * | * | * | ! | * | * | * | * | * |

| 1 | 2 | 3 | 6 |
|---|---|---|---|
| 2 | 1 2 | 1 2 | 0 2 |
| 1 | 0 1 | 0 1 | 0 6 |
| 3 | 0 3 | 1 3 | 0 3 |
| * | ! | ! | * |

**Hits = 4**
**Faults = 16**

# Question 6

Assuming a 4-KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers):

Instruction

| page: m-n | offset: n |
|---|---|

$$4KB = 2^2 . 2^{10} B = 2^{12} B \rightarrow n = 12$$

a. 3085

$3085_{10} = 110000001101_2 \rightarrow m = 12$

no. of pages $= 2^{m-n} = 2^{12-12} = 2^0 = 1$

offset $= 110000001101_2 = 3085_{10}$

b. 42095

$42095_{10} = 1010010001101111_2 \rightarrow m = 16$

no. of pages $= 2^{m-n} = 2^{16-12} = 2^4 = 16$

offset $= 010001101111_2 = 1135_{10}$

c. 215201

$215201_{10} = 110100100010100001_2 \rightarrow m = 18$

no. of pages $= 2^{m-n} = 2^{18-12} = 2^6 = 64$

offset $= 100010100001_2 = 2209_{10}$

d. 650000

$650000_{10} = 10011110101100010000_2 \rightarrow m = 10$

no. of pages $= 2^{m-n} = 2^{20-12} = 2^8 = 256$

offset $= 101100010000_2 = 2832_{10}$

e. 2000001

$2000001_{10} = 111101000010010000001_2 \rightarrow m = 21$

no. of pages $= 2^{m-n} = 2^{21-12} = 2^9 = 512$

offset $= 010010000001_2 = 1153_{10}$

**Shortcut:**

$x = 3085 / 4K = 3085 / 2^{12} = 3085 / 4096 = 0.75317$

$m = 0$ and offset $= 0.75317 * 4096 = 3085$

# Question 7

Consider a computer system with a 32-bit logical address and 4-KB page size. The system supports up to 512 MB of physical memory. How many entries are there in each of the following (*no. of pages & frames*)?

Instruction

| page: m-n | offset: n |
| --- | --- |

$4KB = 2^2.2^{10} B = 2^{12}B \rightarrow n = 12$

no. of pages $= 2^{m-n} = 2^{20} = 1048576$

no. of frames = memory_size / frame_size = 512MB / 4KB $= 2^9.2^{20} / 2^{12} = 2^{17} = 131072$

# Question 8

A 1MB block of memory is allocated to using buddy system:

a) Show the binary tree representation

b) Show the how buddy system satisfy the following request:

R1=70KB, R2=35KB, R3=80KB, R4=60KB

--- incomplete --

# Question 9

Assume a memory access takes 40 ns, and the machine provides a Translation Lookaside Buffer (TLB) with a hit rate of 90% and a search time of 10 ns. What is the effective memory access time.

EA: effective memory access = ?

ma: memory access time = 40ns

x: tlb search time = 10ns

p: hit ratio = 0.9

$\rightarrow$ EA = p(x+ma) + (1-p)(x+ma+ma) = 0.9(10+40) + 0.1(10+80) = 0.9(50) + 0.1(90) = 54ns

# Question 10

Given the following statement for banker algorithm:

**Available**

| A | B | C | D |
|---|---|---|---|
| 6 | 3 | 5 | 4 |

| Process | Current allocation A | B | C | D | Maximum demand A | B | C | D |
|---|---|---|---|---|---|---|---|---|
| P0 | 2 | 0 | 2 | 1 | 9 | 5 | 5 | 5 |
| P1 | 0 | 1 | 1 | 1 | 2 | 2 | 3 | 3 |
| P2 | 4 | 1 | 0 | 2 | 7 | 5 | 4 | 4 |
| P3 | 1 | 0 | 0 | 1 | 3 | 3 | 3 | 2 |
| P4 | 1 | 1 | 0 | 0 | 5 | 2 | 2 | 1 |
| P5 | 1 | 0 | 1 | 1 | 4 | 4 | 4 | 4 |

I. Calculate the need matrix

Since, Need[i][j] = Max[i][j] – Allocated[i][j]

| Process | A | B | C | D |
|---|---|---|---|---|
| $P_0$ | 7 | 5 | 3 | 4 |
| $P_1$ | 2 | 1 | 2 | 2 |
| $P_2$ | 3 | 4 | 4 | 2 |
| $P_3$ | 2 | 3 | 3 | 1 |
| $P_4$ | 4 | 1 | 2 | 1 |
| $P_5$ | 3 | 4 | 3 | 3 |

II. Show the safe sequence of processes. In addition to the sequence show the Available (Work Array) changes as each process terminates.

| 0 | Current allocation | | | | Need | | | | Work |
|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | A | B | C | D | |
| $P_0$ | 2 | 0 | 2 | 1 | 7 | 5 | 3 | 4 | A=6<br>B=3<br>C=5<br>D=4 |
| $P_1$ | 0 | 1 | 1 | 1 | 2 | 1 | 2 | 2 | |
| $P_2$ | 4 | 1 | 0 | 2 | 3 | 4 | 4 | 2 | |
| $P_3$ | 1 | 0 | 0 | 1 | 2 | 3 | 3 | 1 | |
| $P_4$ | 1 | 1 | 0 | 0 | 4 | 1 | 2 | 1 | |
| $P_5$ | 1 | 0 | 1 | 1 | 3 | 4 | 3 | 3 | |

| 1 | Current allocation | | | | Need | | | | Work |
|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | A | B | C | D | |
| $P_0$ | 2 | 0 | 2 | 1 | 7 | 5 | 3 | 4 | A=8<br>B=4<br>C=7<br>D=6 |
| **$P_1$** | **2** | **2** | **3** | **3** | – | – | – | – | |
| $P_2$ | 4 | 1 | 0 | 2 | 3 | 4 | 4 | 2 | |
| $P_3$ | 1 | 0 | 0 | 1 | 2 | 3 | 3 | 1 | |
| $P_4$ | 1 | 1 | 0 | 0 | 4 | 1 | 2 | 1 | |
| $P_5$ | 1 | 0 | 1 | 1 | 3 | 4 | 3 | 3 | |

| 2 | Current allocation | | | | Need | | | | Work |
|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | A | B | C | D | |
| $P_0$ | 2 | 0 | 2 | 1 | 7 | 5 | 3 | 4 | A=11<br>B=8<br>C=11<br>D=8 |
| $P_1$ | 2 | 2 | 3 | 3 | – | – | – | – | |
| **$P_2$** | **7** | **5** | **4** | **4** | – | – | – | – | |
| $P_3$ | 1 | 0 | 0 | 1 | 2 | 3 | 3 | 1 | |
| $P_4$ | 1 | 1 | 0 | 0 | 4 | 1 | 2 | 1 | |
| $P_5$ | 1 | 0 | 1 | 1 | 3 | 4 | 3 | 3 | |

| 3 | Current allocation | | | | Need | | | | Work |
|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | A | B | C | D | |
| **$P_0$** | **9** | **5** | **5** | **5** | **–** | **–** | **–** | **–** | A=18<br>B=13<br>C=14<br>D=12 |
| **$P_1$** | 2 | 2 | 3 | 3 | – | – | – | – | |
| **$P_2$** | 7 | 5 | 4 | 4 | – | – | – | – | |
| **$P_3$** | 1 | 0 | 0 | 1 | 2 | 3 | 3 | 1 | |
| **$P_4$** | 1 | 1 | 0 | 0 | 4 | 1 | 2 | 1 | |
| **$P_5$** | 1 | 0 | 1 | 1 | 3 | 4 | 3 | 3 | |

…

…

| 6 | Current allocation | | | | Need | | | | Work |
|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | A | B | C | D | |
| **$P_0$** | 9 | 5 | 5 | 5 | – | – | – | – | A=27<br>B=21<br>C=22<br>D=17 |
| **$P_1$** | 2 | 2 | 3 | 3 | – | – | – | – | |
| **$P_2$** | 7 | 5 | 4 | 4 | – | – | – | – | |
| **$P_3$** | 3 | 3 | 3 | 2 | – | – | – | – | |
| **$P_4$** | 5 | 2 | 2 | 1 | – | – | – | – | |
| **$P_5$** | **4** | **4** | **4** | **4** | **–** | **–** | **–** | **–** | |

# Question 11

Dining Philosopher problem states that there are 5 philosophers and 5 chopsticks. Using resource allocation graph, show how a deadlock occurs?