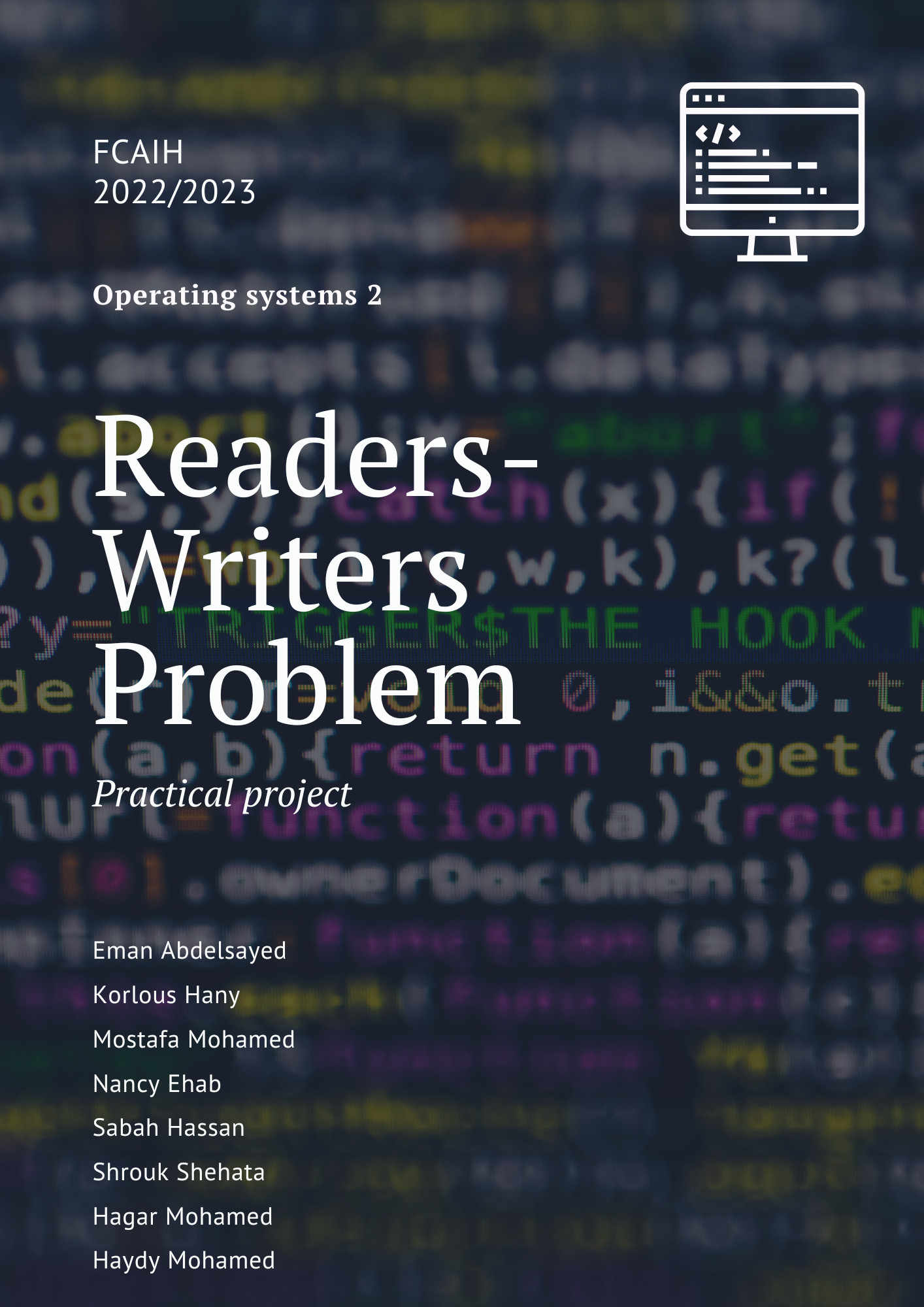
****

**Contents**

[**Writer process:** 2](#_Toc122222826)

[**Reader process:** 3](#_Toc122222827)

[**What is Deadlock?** 4](#_Toc122222828)

[**Examples of deadlock:** 4](#_Toc122222829)

[**Readers/writer locks** 5](#_Toc122222830)

[**Pseudocode example (Not Java)** 6](#_Toc122222831)

[**How to solve Deadlock :** 10](#_Toc122222832)

[**What is starvation?** 11](#_Toc122222833)

[**Examples of starvation:** 12](#_Toc122222834)

[**How to solve starvation?** 14](#_Toc122222835)

[Explanation for real world application and how did apply the problem. 14](#_Toc122222836)

**Reader writer problem**

Readers and Writers Problem. This problem occurs when many threads of execution try to access the same shared resources at a time. There are N readers to read data and K Writers to write data to shared resources.

# **Writer process:**

1. Writer requests the entry to critical section.
2. If allowed i.e. wait() gives a true value, it enters and performs the write. If not allowed, it keeps on waiting.
3. It exits the critical section.

# **Reader process:**

1. Reader requests the entry to critical section.
2. If allowed:
   * it increments the count of number of readers inside the critical section. If this reader is the first reader entering, it locks the **wrt** semaphore to restrict the entry of writers if any reader is inside.
   * It then, signals mutex as any other reader is allowed to enter while others are already reading.
   * After performing reading, it exits the critical section. When exiting, it checks if no more reader is inside, it signals the semaphore “wrt” as now, writer can enter the critical section.
3. If not allowed, it keeps on waiting.

Diagram

Description automatically generated with medium confidence

# **What is Deadlock?**

**Deadlock** is a situation that occurs in OS when any process enters a waiting state because another waiting process is holding the demanded resource. Deadlock is a frequent problem in multi-processing where several processes share a specific type of mutually exclusive resource known as a soft lock or software.

# **Examples of deadlock:**

Diagram

Description automatically generated1-

Graphical user interface, text, application, email

Description automatically generated

2-

# **Graphical user interface, text Description automatically generatedReaders/writer locks**

# **Text, letter Description automatically generatedPseudocode example (Not Java)**

3- When we use semaphore each philosopher take one stack and they are hungry because philosopher to eat need two stacks, so this is cause deadlock.

A picture containing diagram

Description automatically generated

4- when many threads of execution try to access the same shared resources at a time.

Graphical user interface

Description automatically generated

5- When we use synchronized:

When we started the threads from the Main class, both the threads called therun()method.Note that both threads are sharing a common messageobject. Now the Reader thread called the synchronizedread()method and hence acquired the lock of the messageobject.As initially the booleanemptyflag was set totrue, the Readerthread keeps executing while loop infinitely.Also, the Writer thread won't be able to execute thewrite()method as the lock of the messageobject is already acquired by the Reader thread.

Graphical user interface, text

Description automatically generated

# **How to solve Deadlock :**

Text

Description automatically generated

Text

Description automatically generated

# **What is starvation?**

Table

Description automatically generated

# **Examples of starvation:**

1-

Diagram

Description automatically generated

2-

A picture containing diagram

Description automatically generated

When we use monitor should starvation when the middle philosopher is hungry and the left and write philosopher are eating.

Timeline

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated3-

A screenshot of a computer

Description automatically generated with medium confidence

# **How to solve starvation?**

By using monitor cooperation

# Explanation for real world application and how did apply the problem.

 A practical example of a Readers and Writers problem is an airline reservation system consisting of a huge data base with many processes that read and write the data. Reading information from the data base will not cause a problem since no data is changed. The problem lies in writing information to the data base. If no constraints are put on access to the data base, data may change at any moment. By the time a reading process displays the result of a request for information to the user, the actual data in the data base may have changed. What if, for instance, a process reads the number of available seats on a flight, finds a value of one, and reports it to the customer. Before the customer has a chance to make their reservation, another process makes a reservation for another customer, changing the number of available seats to zero.

Solve:

by monitor cooperation:

***Monitor in Java -Cooperation***

•wait ():

•Call to this method causes the current thread to wait until another thread invokes the notify() or the notifyAll() method for this object. The current thread must own this object’s monitor(lock).

The thread releases ownership of this monitor and waits until another thread notifies threads waiting on this object’s monitor to wake up either through a call to the notify or the notifyAllmethod.

This method should only be called by a thread that is the owner of this object’s monitor.

•notifyAll():

•Call to this method wakes up all threads that are waiting on this object’s monitor. A thread waits on an object’s monitor by calling one of the wait methods.

The awakened threads will not be able to proceed until the current thread relinquishes (voluntarily releases) the lock on this object.

This method should only be called by a thread that is the owner of this object’s monitor.