1. What is the difference between an operating system and middleware?

An operating system is a program that uses computer hardware to provide support for the execution of other software. A middleware on the other hand is software that mediates between the operating system and application programs.

2. What is the relationship between threads and processes?

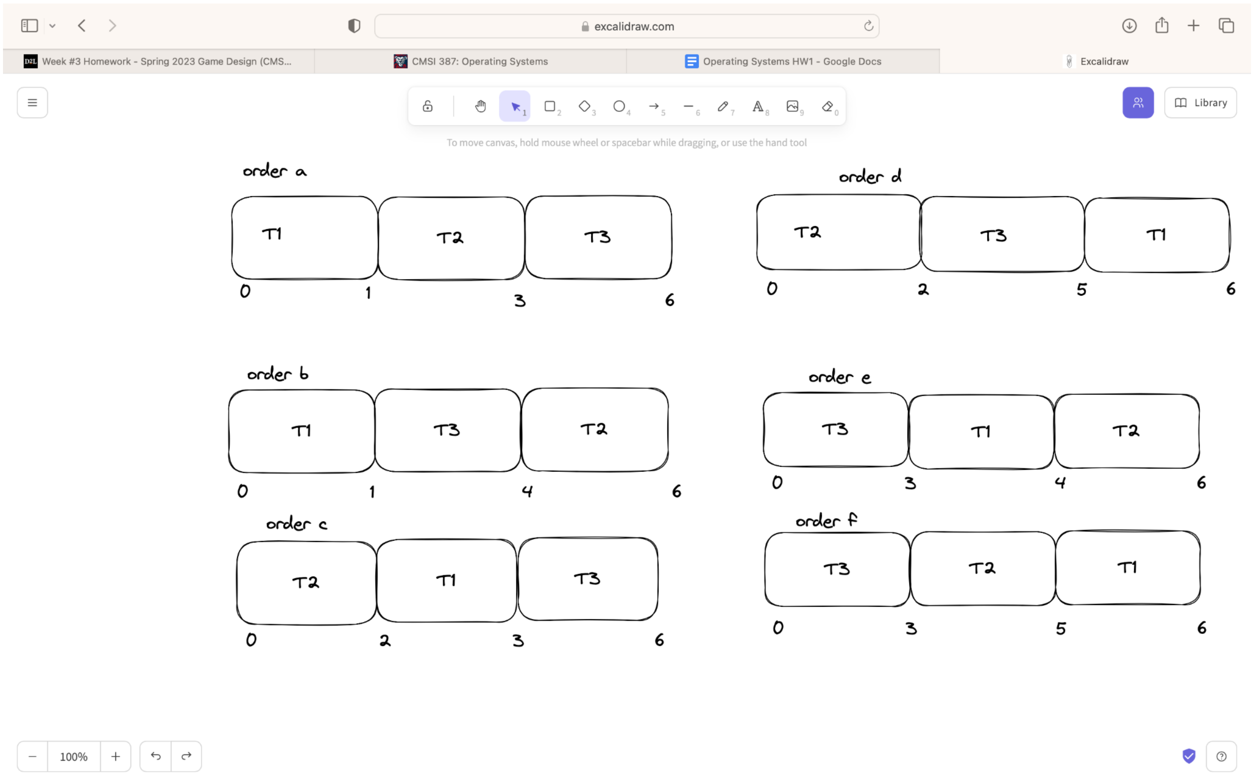
A thread is a sequence of computational steps executed in a chain while a process is an environment in which threads are executed. In other words, threads can be executed within a process

3. Of all the topics previewed in chapter one of the text book, which one are you most looking forward to learning more about? Why?

I am excited to learn about the kernel. I had no idea the operating system had a kernel that has specific functionality. I want to understand how the kernel uses hardware to make the machines (computers) that I love so much function so fluidly.

4. Suppose thread A goes through a loop 100 times, each time performing one disk I/O operation, taking 10 milliseconds, and then some computation, taking 1 millisecond. While each 10-millisecond disk operation is in progress, thread A cannot make any use of the processor. Thread B runs for 1 second, purely in the processor, with no I/O. One millisecond of processor time is spent each time the processor switches threads; other than this switching cost, there is no problem with the processor working on thread B during one of thread A's I/O operations. (The processor and disk drive do not contend for memory access bandwidth, for example.)
   - Suppose the processor and disk work purely on thread A until its completion, and then the processor switches to thread B and runs all of that thread. What will the total elapsed time be? 2101 ms
   - Suppose the processor starts out working on thread A, but every time thread A performs a disk operation, the processor switches to B during the operation and then back to A upon the disk operation's completion. What will the total elapsed time be? 1301 ms
   - *In your opinion*, which do you think is more efficient, and why? Approach 2 is better. Less time is taken to execute thread A and B by the processor.
5. Find and read the documentation for **pthread_cancel()**. Then, using your "C" programming environment, use the information and the model provided in Figure 2.4 on page 26 of the text book to write a program in which the initial (main) thread creates a second thread. The main thread should sit on a read call of some kind, waiting to read input from the keyboard, waiting until the user presses the Enter key. At that point, it should kill off the second thread and print out a message reporting that it has done so. Meanwhile, the second thread should be in an infinite loop, each time around sleeping five seconds and then printing out a message. Try running your program. Can the sleeping thread print its periodic messages while the main thread is waiting for keyboard input? Can the main thread read input, kill the sleeping thread, and print a message while the sleeping thread is in the early part of one of its five-second sleeps?

6. Suppose a system has three threads (T1, T2, and T3) that are all available to run at time 0 and need one, two, and three seconds of processing, respectively. Suppose each thread is run to completion before starting another. Draw six different Gantt charts, one for each possible order the threads can be run in. For each chart, compute the "turnaround" time of each thread; that is, the time elapsed from when it was ready (time 0) until it is complete. Also, compute the average turnaround time for each order. Which order has the shortest average turnaround time? What is the name of the scheduling policy that produces this order?


excalidraw.com
Week #3 Homework - Spring 2023 Game Design (CMS...
CMSI 387: Operating Systems
Operating Systems HW1 - Google Docs
Excalidraw
Library
To move canvas, hold mouse wheel or spacebar while dragging, or use the hand tool
order a
T1
T2
T3
0
1
3
6
order d
T2
T3
T1
0
2
5
6
order b
T1
T3
T2
0
1
4
6
order e
T3
T1
T2
0
3
4
6
order c
T2
T1
T3
0
2
3
6
order f
T3
T2
T1
0
3
5
6
100%

Order a

- T1 turnaround time = 1s
- T2 turnaround time = 3s
- T3 turnaround time = 6s
- Average turnaround = 10/3 = 3.33s

Order b

- T1 turnaround time = 1s
- T2 turnaround time = 6s
- T3 turnaround time = 4s
- Average turnaround = 11/3 = 3.67s

Order c

- T1 turnaround time = 3s
- T2 turnaround time = 2s
- T3 turnaround time = 6s
- Average turnaround = 11/3 = 3.67s

Order D

- T1 turnaround time = 6s
- T2 turnaround time = 2s
- T3 turnaround time = 5s

- Average turnaround = 13/3 = 4.33s

Order E

- T1 turnaround time = 4s
- T2 turnaround time = 6s
- T3 turnaround time = 3s
- Average turnaround = 13/3 = 4.33s

Order F

- T1 turnaround time = 6s
- T2 turnaround time = 5s
- T3 turnaround time = 3s
- Average turnaround = 14/3 = 4.67s

Order A has the lowest average turnaround time. The scheduling policy with this order is Earliest Deadline First(EDF)..

7. Google the "C" standard library API and find out how to get information from the command line by using a **printf()** call to display a prompt, then another call [which you will look up] to get the user input. Write a program in "C" to prompt the user demographic information including name, age, class year, and any three other data times you wish. Structure the program as a "call-and-response" program such that each data item is a single question with a single answer. When all data has been obtained, display the data on the console. Each data item must be on a separate line, and it must be appropriately labeled. The output must be done using a **single** **printf()** statement.