CMSI 387
HOMEWORK #1
TIGERLILLY ZIETZ, JOSH SEAMAN, NICK MORGAN

---

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

   The difference between operating systems and middleware is that they rely upon different underlying providers of lower-level services. As well as, that the operating system makes use of the features supported by the hardware to provide services to the API, while the middleware, “sits on top of the operating system”, provides the services in its API by making use of the features supported by an underlying operating system.

2. What is the relationship between threads and processes?

   A process is a container that holds a thread, or multiple threads, and protects them from unwanted interactions with other unrelated threads running on the same machine. Also, a thread running on one process doesn’t have the ability to unintentionally that is being used by another process.

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

   Of the topics previewed in chapter one of the textbook we are all very excited to learn about the topics in Chapters 9 and 10 of the textbook. We are excited to learn more about how machines are able to interact and communicate remotely and synchronously while providing support for all machines connected.

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.)

a. 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?

100*(10ms + 1ms) + 1ms + 1000ms = 2101ms = 2.101s

b. 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?

100*(10ms + 1ms) + 1ms + (1000ms-100*(8ms)) = 1301ms = 1.301s

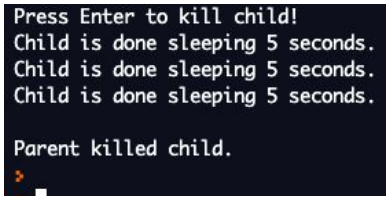
c. In your opinion, which do you think is more efficient, and why?

It is more efficient to follow option b, because the difference in computation time would become more and more apparent as the scale is increased towards infinity.

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 textbook 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.

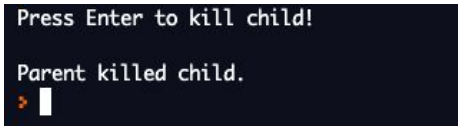
Code is included in our team repo

Can the sleeping thread print its periodic messages while the main thread is waiting for keyboard input?

Yes: 

(processed three periodic messages while it waited for our eventual 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?

Yes: 

(processed the input, killed the sleeping thread, and printed the message before the first periodic cycle)

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?

The ordering of **Chart 1** has the shortest turnaround time at approximately 3.3 seconds. **Shortest Job First** is the name of the scheduling policy that produces this order. Shortest Job First is the scheduling policy that in which the process that has the shortest execution time is chosen for the next execution.

Chart 1	1s	2s	3s	4s	5s	6s
T1						
T2						
T3						

1	T1	T2	T3
Turnaround time (s)	1	3	6
Average turnaround time (s)			3.3

2	1s	2s	3s	4s	5s	6s
T1						
T3						
T2						

2	T1	T3	T2
Turnaround time (s)	1	4	6
Average turnaround time (s)			3.7

3	1s	2s	3s	4s	5s	6s
T2						
T1						
T3						

3	T2	T1	T3
Turnaround time (s)	2	3	6

Average turnaround time (s)			3.7

4	1s	2s	3s	4s	5s	6s
T2						
T3						
T1						

4	T2	T3	T1
Turnaround time (s)	2	5	6
Average turnaround time (s)			4.3

5	1s	2s	3s	4s	5s	6s
T3						
T2						
T1						

5	T3	T2	T1
Turnaround time (s)	3	5	6
Average turnaround time (s)			4.7

6	1s	2s	3s	4s	5s	6s
T3						
T1						
T2						

6	T3	T1	T2
Turnaround time (s)	3	4	6
Average turnaround time (s)			4.3

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.

   Code is included in our team repo