جامعة بيرزيت

BIRZEIT UNIVERSITY

لن يستطيعوا كسر إرادتنا، كلنا غزة

OUR WILL IS UNBREAKABLE... WE ARE ALL GAZA

Moath Abd Albaqi

1210125

O.S First Task

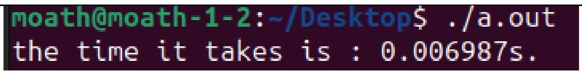
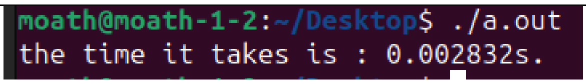
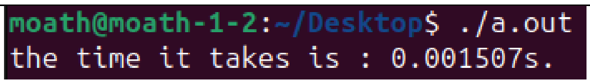
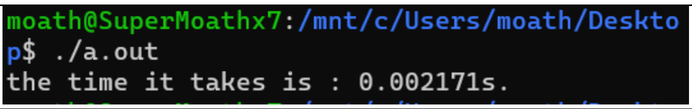
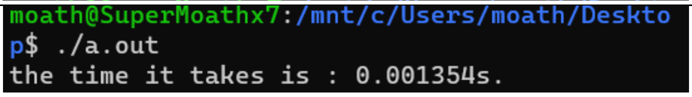
Dr. Abdel Salam Sayyad

Section: 1

Tuesday

28/11/2023

1-

Number of Processes \ threads	Time
naive approach (1 process\ 1 thread)	 `moath@moath-1-2:~/Desktop$ ./a.out` `the time it takes is : 0.006987s.`
Two processes	 `moath@moath-1-2:~/Desktop$ ./a.out` `the time it takes is : 0.002832s.`
Four processes	 `moath@moath-1-2:~/Desktop$ ./a.out` `the time it takes is : 0.001507s.`
Two joinable threads	 `moath@SuperMoathx7:/mnt/c/Users/moath/Deskto` `p$ ./a.out` `the time it takes is : 0.002171s.`
Four joinable threads	 `moath@SuperMoathx7:/mnt/c/Users/moath/Deskto` `p$ ./a.out` `the time it takes is : 0.001354s.`
Detached threads	**Can't be calculated**

The naive approach has the longest time among all tests, because there is just one process executing the program (the code).

The two-process approach gives nearly half the execution time of the naive one, because these two processes are executed in parallel. Each process computes a half of the overall work, as a result the execution time is nearly half of the naive approach, also four-process approach gives approximately half the execution time of the two-process approach because of each process will have a quarter of the whole work which executes at the same time.

In threads, the data is shared between all threads in a process as thread is the segment of it, main thread will wait the joinable threads until they finish the execution, then it will terminate.

Two joinable threads approach is nearly the same as two processes with the difference in the container, but as shown the thread is a little faster than process because process is not lightweight like thread, and it takes more time for creation and termination.

Two threads will be in the same process, with the same data, every thread will do half of the whole work with nearly third the execution time of naive approach, four- thread approach will have half the time of two threads.

Detached threads are designed to run behind the main thread without join (the main thread doesn't need to wait for them to finish the execution), if the execution time is calculated it reflects the time of the main thread, not the detached thread. This time is typically short because the main thread doesn't contain loops or something that takes a lot of time to execute (in my case, it is equal to 0.000220s).

2-

The best number of processes in my case is 4, because the array filling and multiplication is divided between 4 processes, so each process will do a quarter of the work at the same time.

Same thing for joinable threads, the best number is 4 threads.

Throughput for naïve approach =1/execution time = 1 / 0.006987 = 143.122 sec^-1.

For 2 process = 353.107 sec^-1.

For 4 process = 663.570 sec^-1.

For 2 threads = 460.617 sec^-1.

For 4 threads = 970.873 sec^-1.

Note: I divided the whole program between threads and processes not just the matrix multiplication.

The C codes are well-commented; please review them.