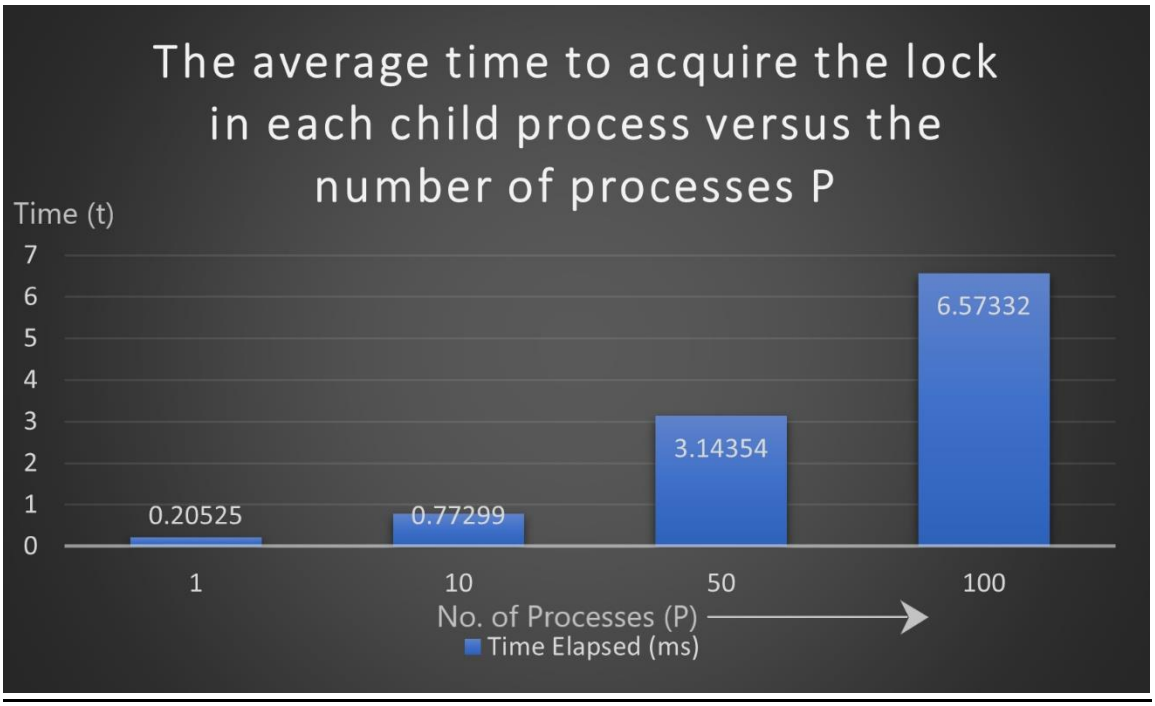
OS Assignment 2 Part C Report

1) Total execution time for the parent process versus number of processes P

- shared2a.c (Number(n)=100):

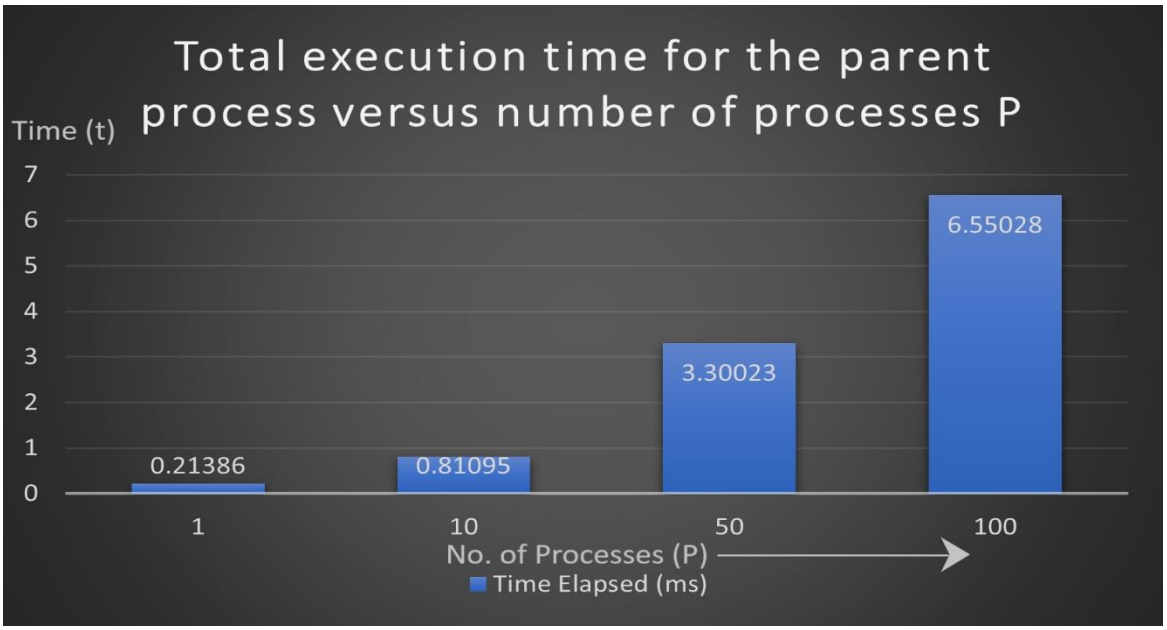
The following graph is the result of “sem_try_wait()” and “continue” statement. Here, execution time is measured in milliseconds(ms) and the number of processes is declared by “P”. We have taken 4 cases with 1, 10, 50 and 100 processes. The results are as follows:



We can conclude that the execution time increases as the number of processes increases.

- shared2b.c (Number(n)=100):

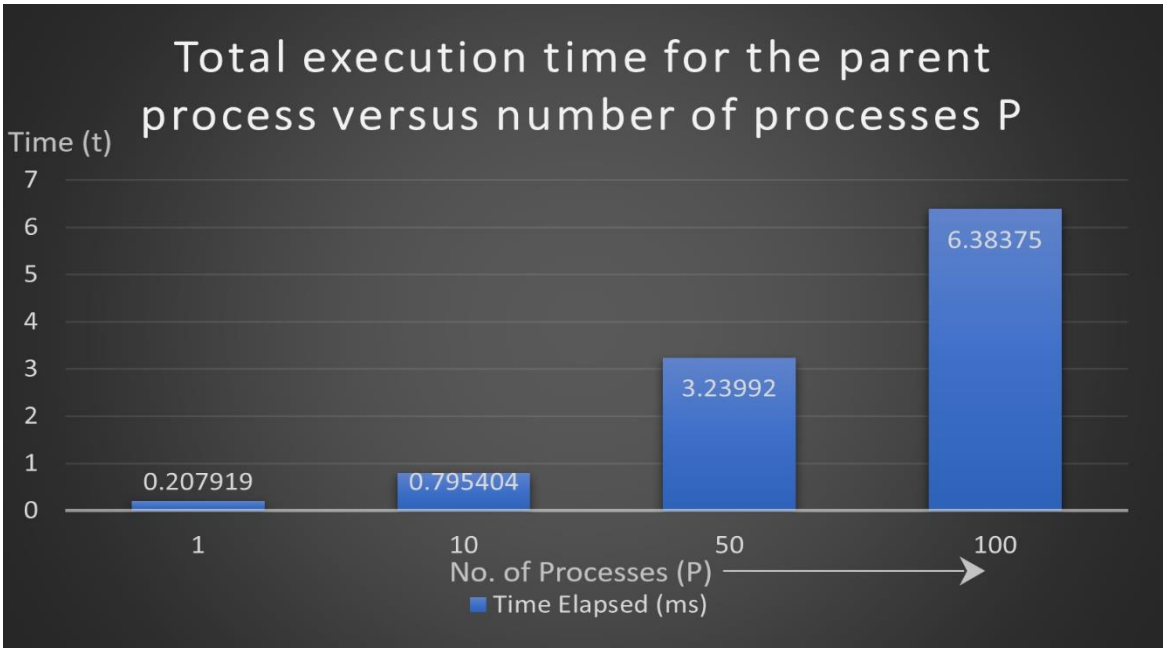
In the following graph we have replaced the “continue” statement with “sched_yield()”. Here, execution time is measured in milliseconds(ms) and the number of processes is declared by “P”. We have taken 4 cases with 1, 10, 50 and 100 processes. The results are as follows:



We can conclude that the execution time increases as the number of processes increases.

- shared2c.c (Number(n)=100):

In the following graph we have used the Blocking Semaphore(sem_wait()). Here, execution time is measured in milliseconds(ms) and the number of processes is declared by "P". We have taken 4 cases with 1, 10, 50 and 100 processes. The results are as follows:

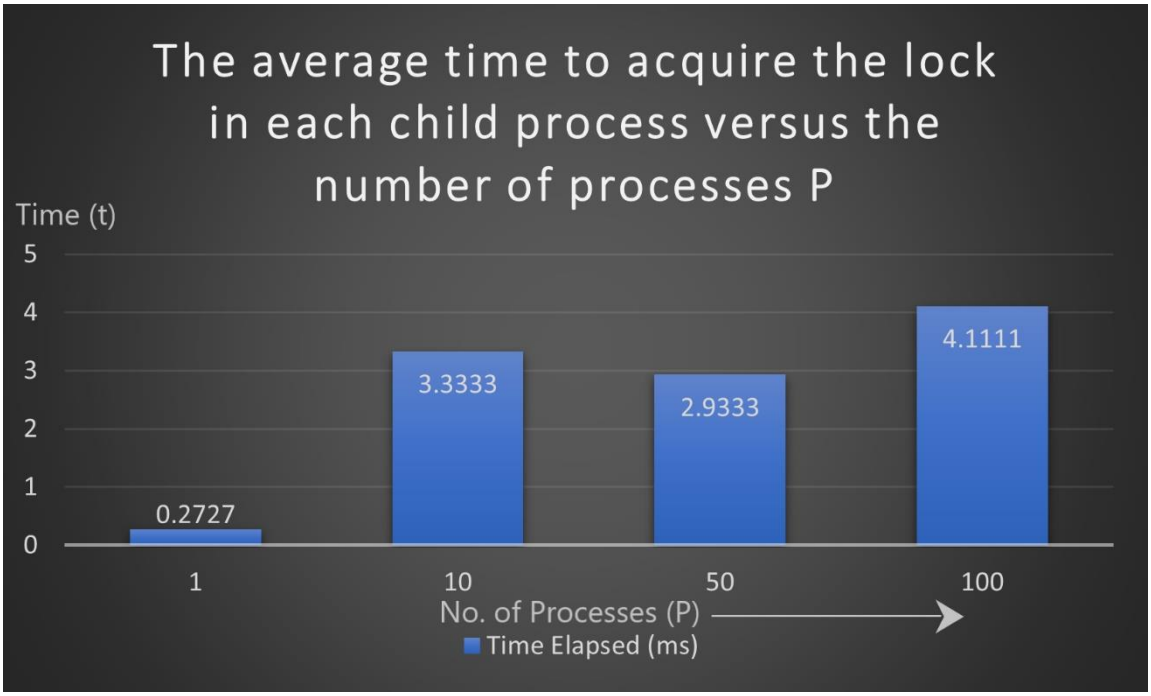


We can conclude that the execution time increases as the number of processes increases.

2) The average time to acquire the lock in each child process versus the number of processes P

- shared2a.c (Number(n)=10^{10}):

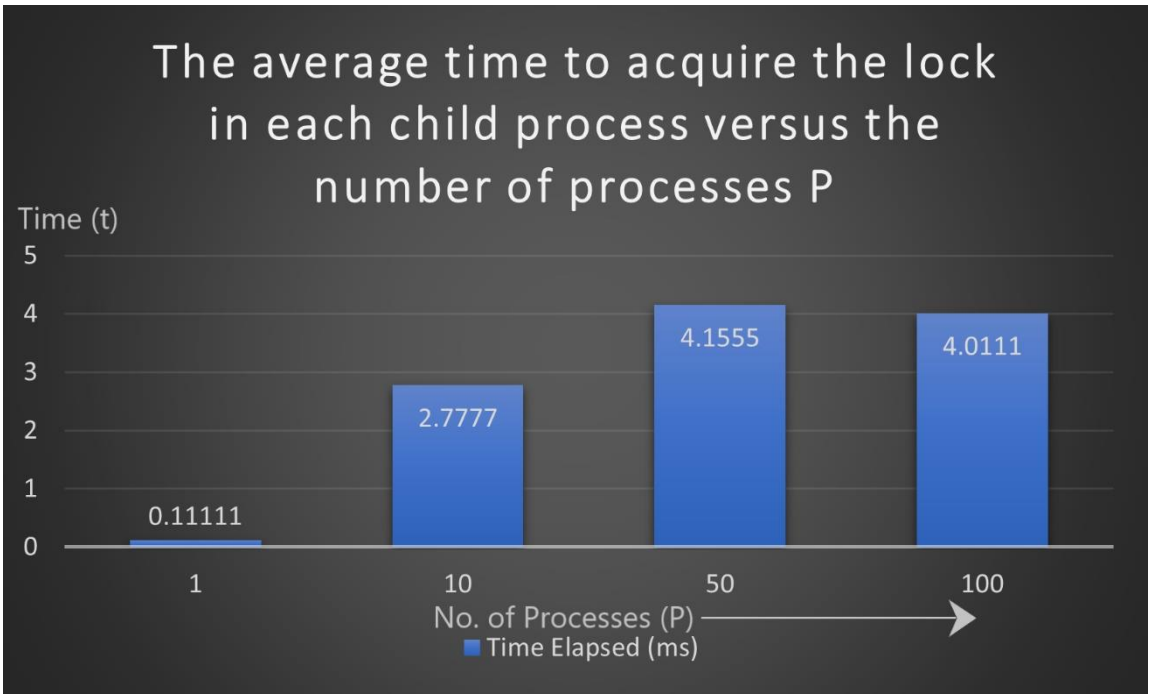
The following graph is the result of “sem_try_wait()” and “continue” statement. Here, execution time is measured in milliseconds(ms) and the number of processes is declared by “P”. We have taken 4 cases with 1, 10, 50 and 100 processes. The results are as follows:



We can conclude that the average time to acquire the lock in each child process doesn’t follow a particular pattern and is completely random.

- shared2b.c (Number(n)=10^{10}):

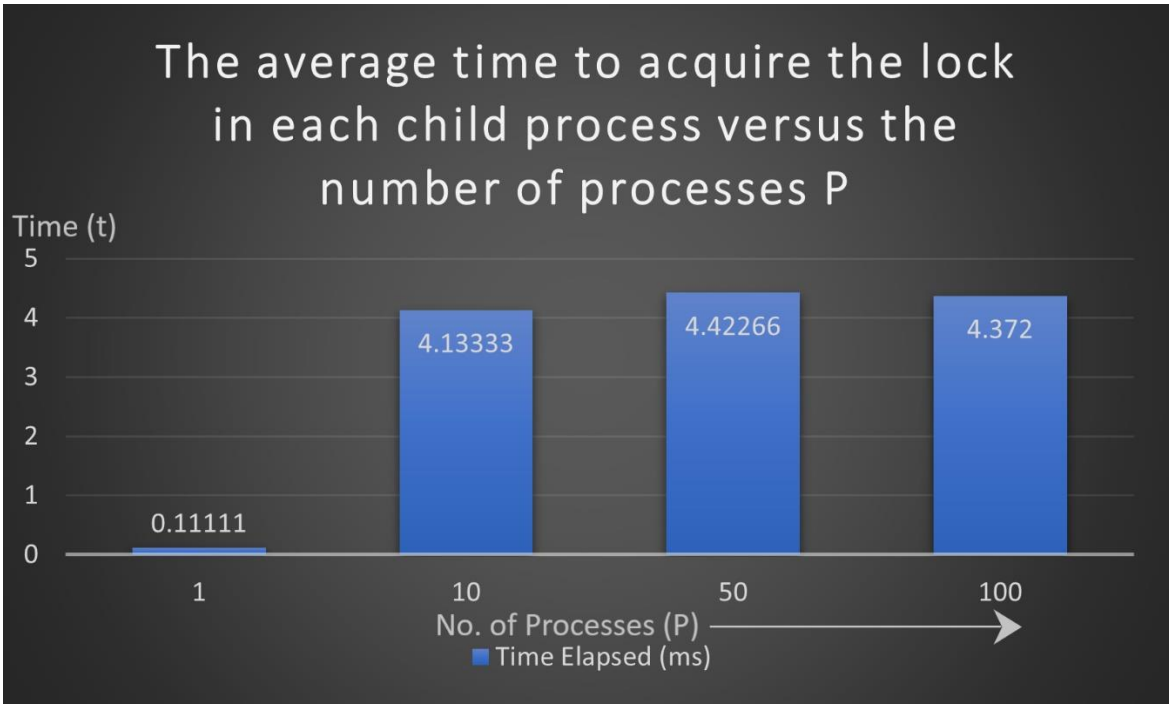
In the following graph we have replaced the “continue” statement with “sched_yield()”. Here, execution time is measured in milliseconds(ms) and the number of processes is declared by “P”. We have taken 4 cases with 1, 10, 50 and 100 processes. The results are as follows:



We can conclude that the average time to acquire the lock in each child process doesn't follow a particular pattern and is completely random.

- shared2c.c (Number(n)=10^{10}):

In the following graph we have used the Blocking Semaphore(sem_wait()). Here, execution time is measured in milliseconds(ms) and the number of processes is declared by "P". We have taken 4 cases with 1, 10, 50 and 100 processes. The results are as follows:



We can conclude that the average time to acquire the lock in each child process doesn't follow a particular pattern and is completely random.

Profiling Results:

Total execution time for the parent process versus number of processes P

	1	10	50	100
sem_try_wait() and continue	0.20525	0.77299	3.14354	6.57332
sem_try_wait() and sched_yield()	0.21386	0.81095	3.30023	6.55028
sem_wait()	0.207919	0.795404	3.23992	6.38375

The average time to acquire the lock in each child process versus the number of processes P

	1	10	50	100
sem_try_wait() and continue	0.2727	3.3333	2.9333	4.1111
sem_try_wait() and sched_yield()	0.11111	2.7777	4.1555	4.0111
sem_wait()	0.11111	4.13333	4.42266	4.372