Process Coordination using Semaphores and Mutexes

1. **Buffer Under-flow**: Consumer may try to consume even before the producer produces it. We do not know which process runs first i.e. whether consumer or producer.

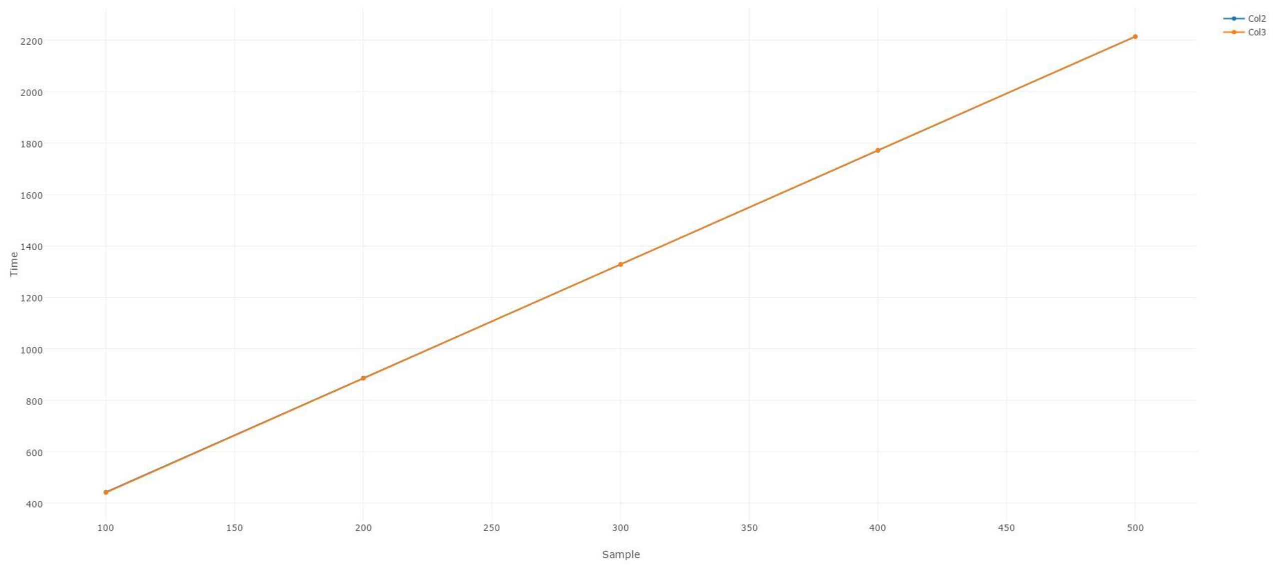
2.**Buffer over-flow**: Producer may produce more quickly and consumer is not able to consume at such speed then buffer is filled.

3.Processes do not have any communication, so they don't know each other's state.

Graph values:

s.no.	Sample(X-axis)	Time(semaphore[Y])	Time(mutex[Y])
1	100	443	441
2	200	886	885
3	300	1328	1328
4	400	1771	1772
5	500	2214	2214

GRAPH:



Conclusion: So by using semaphores and mutex we are able to control access to a shared buffer. We need to write the code in such a way that it do not lead to deadlock condition. Deadlock condition is that both processes are waiting forever. From the graph values it is observed that the code implemented using mutex is somewhat faster than the semaphore for the lower number of samples. When the samples are increased the time is same for both the implementation. For the sample of 400 we observed that semaphore implementation is quick.