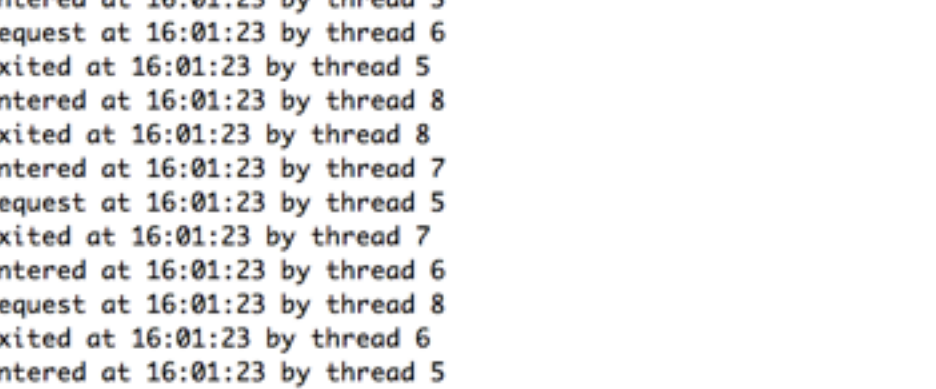


## OS: ASSIGNMENT 8.

There is no notion of ensuring that every thread has a fair chance and is checked for readiness in a certain period of time. The main principle of unbounded waiting is mutual exclusion. Which can be shown as below:



```
Assn8-cs13b1040 - bash - 80x24
bash
3rd CS entered at 16:01:23 by thread 5
6th CS Request at 16:01:23 by thread 6
3rd CS exited at 16:01:23 by thread 5
3rd CS entered at 16:01:23 by thread 8
3rd CS exited at 16:01:23 by thread 8
8th CS entered at 16:01:23 by thread 7
4th CS Request at 16:01:23 by thread 5
8th CS exited at 16:01:23 by thread 7
6th CS entered at 16:01:23 by thread 6
4th CS Request at 16:01:23 by thread 8
6th CS exited at 16:01:23 by thread 6
4th CS entered at 16:01:23 by thread 5
9th CS Request at 16:01:23 by thread 7
7th CS Request at 16:01:23 by thread 6
4th CS exited at 16:01:23 by thread 5
7th CS entered at 16:01:23 by thread 6
7th CS exited at 16:01:23 by thread 6
4th CS entered at 16:01:23 by thread 8
8th CS Request at 16:01:23 by thread 6
5th CS Request at 16:01:23 by thread 5
4th CS exited at 16:01:23 by thread 8
8th CS entered at 16:01:23 by thread 6
8th CS exited at 16:01:23 by thread 6
9th CS entered at 16:01:23 by thread 1
```

## BOUNDED WAITING

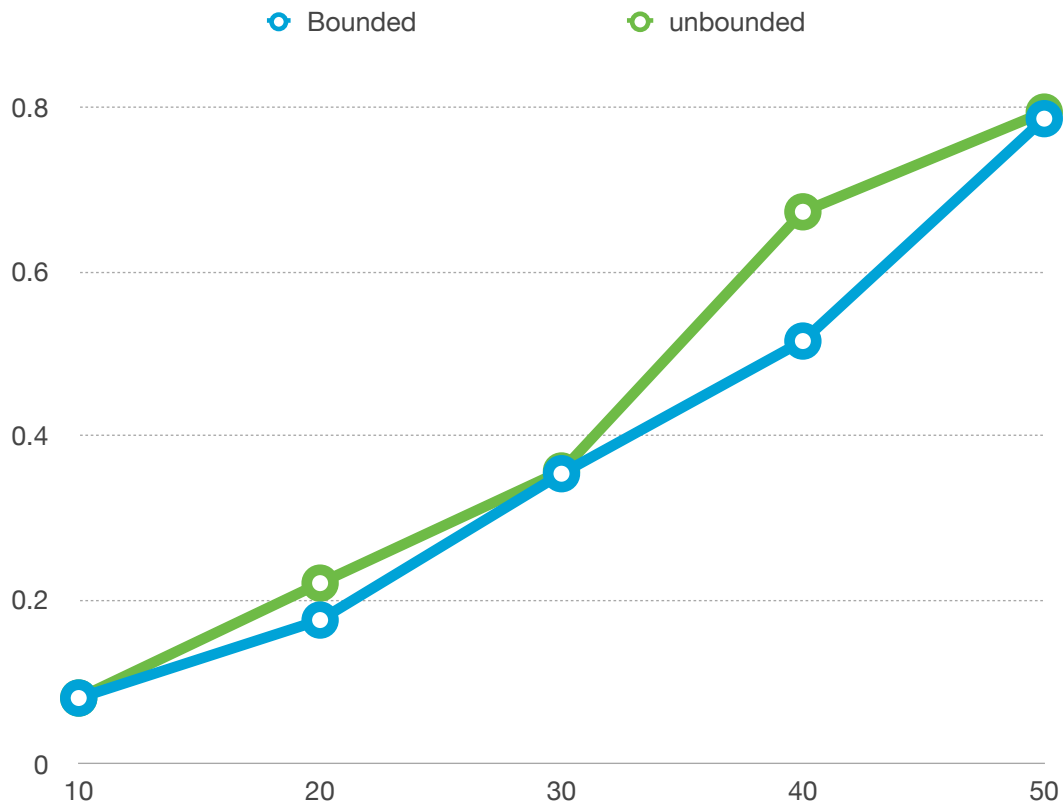
Apart from mutual exclusion there exists a bound, or limit, on the number of times that other processes are allowed to enter their critical sections after a thread has made a request to enter its critical section and before that request is granted. In this case the number of thread is equal to the total number of threads. After each thread has accessed it's critical section it will check for all the remaining threads sequentially. Ensuring that each thread has a fair chance and that there is no starvation.

The following is an example of bounded waiting:

```
8th CS entered at 15:52:08 by thread 5
10th CS Request at 15:52:08 by thread 9
8th CS exited at 15:52:08 by thread 5
10th CS entered at 15:52:08 by thread 9
10th CS exited at 15:52:08 by thread 9
6th CS entered at 15:52:08 by thread 10
9th CS Request at 15:52:08 by thread 5
6th CS exited at 15:52:08 by thread 10
6th CS entered at 15:52:08 by thread 3
7th CS Request at 15:52:08 by thread 10
6th CS exited at 15:52:08 by thread 3
7th CS entered at 15:52:08 by thread 10
7th CS Request at 15:52:08 by thread 3
7th CS exited at 15:52:08 by thread 10
9th CS entered at 15:52:08 by thread 5
8th CS Request at 15:52:08 by thread 10
9th CS exited at 15:52:08 by thread 5
7th CS entered at 15:52:08 by thread 3
```



## COMPARISON



The graph above shows a comparison between unbounded and bounded waiting's performance.

The x-axis represents the number of threads, whereas the y-axis is the average waiting time in seconds. The average waiting time is the time elapsed between the when the thread is ready to enter the critical section and when the thread actually gets access to the critical section.

One would assume that the performance of Unbounded waiting would be much greater, however as we can clearly see from the graph, on an average Bounded waiting tends to have a better performance over unbounded waiting. The data here may be a bit doubtful as we have measured it across a small number of threads, however we can still observe the general pattern.

One of the main reasons for better performances in bounded waiting is that despite its complex algorithm, bounded waiting ensures that each process gets checked at least once every  $n$  times. Where the number  $n$  can vary from algorithm to algorithm. It eradicates one of the main problems when it comes to mutual exclusion, namely starvation.

By ensuring that there is no starvation, bounded waiting effectively reduces the average waiting time, thereby increasing the performance, as shown above.