Maharshi Shah

Dr. Valentina Korzhova

Operating Systems

23 September 2018

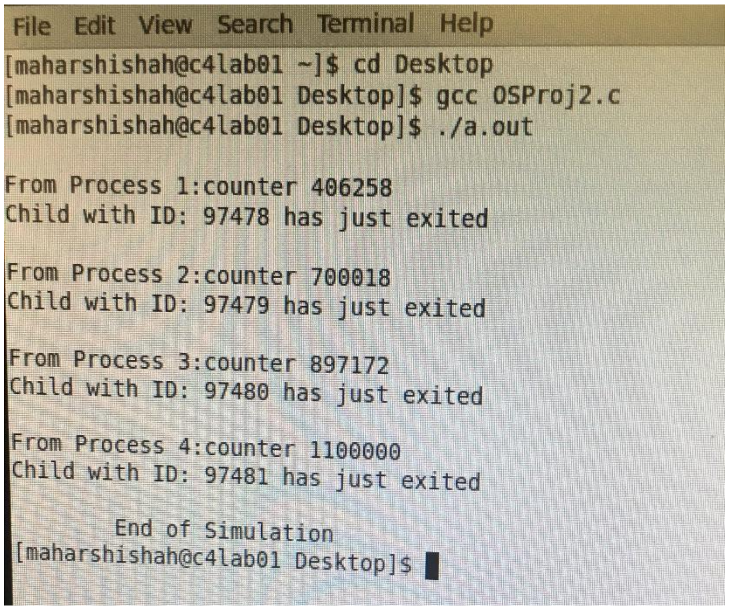
Project 2 - Semaphores

This project has four processes that share a memory location and increment the shared variable by one. Process 1 increments by one 100,000 times, Process 2 increments 200,000 times, Process 3 300,000 times and Process 4 increments 500,000. The total of all four processes is 1100,000 times and that is when the processes should terminate. Semaphores are variables that allow us to control access to a common resource by multiple processes sharing a common resource. Semaphore operations include wait(S) that blocks processes until calling process is blocked. Another one is signal(S) that allows the process to be never blocked. In this project, these operations have been implemented as POP() and VOP() functions that perform the same operations.

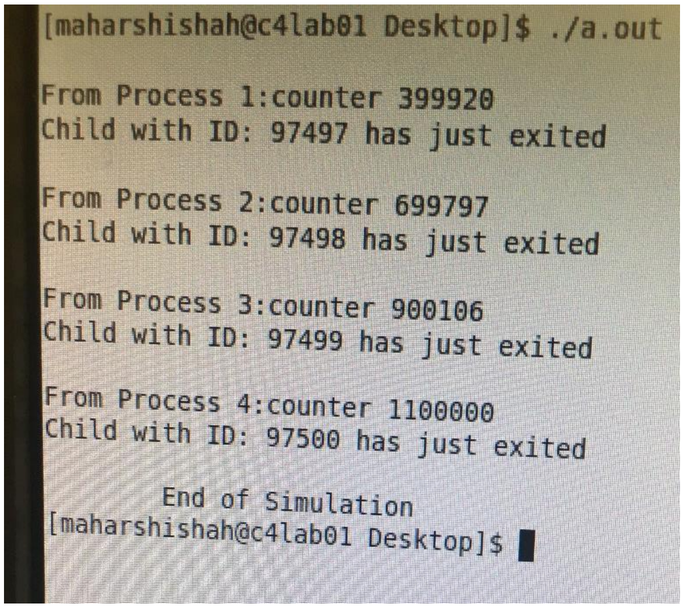
RESULTS

The following page shows the images after running the code developed that creates 4 processes and share a variable until the count reaches 1100000 times. The output has the counter of a process and the child ID of a particular parent process that returns it. In order to run the code on the C4 machines, following are the steps:

● **gcc OSProj2.c**	To compile
● **./a.out**	To run the code (More information in the readme file)



```
[maharshishah@c4lab01 ~]$ cd Desktop
[maharshishah@c4lab01 Desktop]$ gcc OSProj2.c
[maharshishah@c4lab01 Desktop]$ ./a.out

From Process 1:counter 406258
Child with ID: 97478 has just exited

From Process 2:counter 700018
Child with ID: 97479 has just exited

From Process 3:counter 897172
Child with ID: 97480 has just exited

From Process 4:counter 1100000
Child with ID: 97481 has just exited

        End of Simulation
[maharshishah@c4lab01 Desktop]$
```



```
[maharshishah@c4lab01 Desktop]$ ./a.out

From Process 1:counter 399920
Child with ID: 97497 has just exited

From Process 2:counter 699797
Child with ID: 97498 has just exited

From Process 3:counter 900106
Child with ID: 97499 has just exited

From Process 4:counter 1100000
Child with ID: 97500 has just exited

        End of Simulation
[maharshishah@c4lab01 Desktop]$
```

CONCLUSION

Concurrent programming must be done with great care and it causes an unavoidable overhead to program. Some areas where using semaphores might affect the code as a whole are: Time to initialize, overhead due to external libraries, extra time to communicate between threads, etc. This was evident when the code form Project 1 was run with four other processes that used a shared variable but with semaphore this time around. Overall, the concept of semaphores and their use in a program with parallel processes was important to understand to be able to complete the project.