Project #2 (Semaphores)

Continuation of Project 1

Kyumin Lee
University of Oklahoma
Norman, OK
kyuminlee@ou.edu

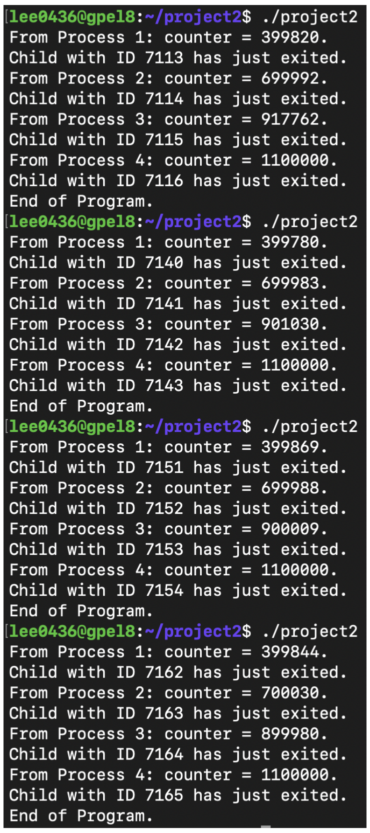
***Abstract*—Previously in Project #1, we had an issue of concurrency caused by unprotected shared memory. In this project, we fix the issue by implementing Semaphores.**

***Keywords*—Semaphores, processes, counter, total**

I. To solve the problem

To implement Semaphores, I can use the given example of the implementation. As project 1 I create 4 child processes, which means we call fork() 4 times. I need to update the increment function, so that processes increment to the target of 100,000, 200,000, 300,000, and 500,000 respectively. At the end, we need to wait() for all child processes to exit, then we always make sure to release the shared memory, deallocate Semaphore.

II. Results

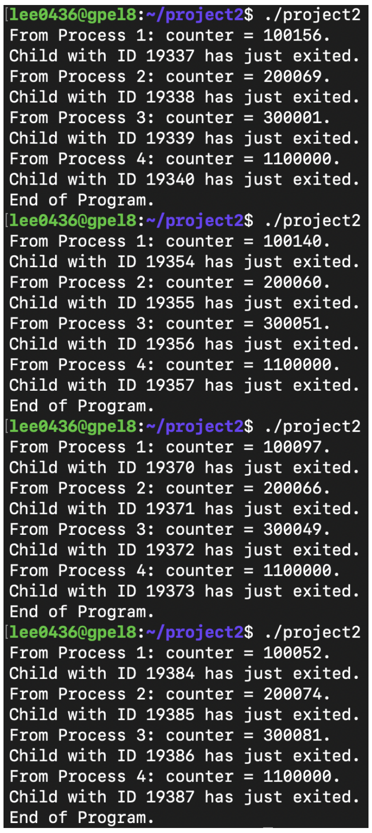


```
lee0436@gpel8:~/project2$ ./project2
From Process 1: counter = 399820.
Child with ID 7113 has just exited.
From Process 2: counter = 699992.
Child with ID 7114 has just exited.
From Process 3: counter = 917762.
Child with ID 7115 has just exited.
From Process 4: counter = 1100000.
Child with ID 7116 has just exited.
End of Program.
lee0436@gpel8:~/project2$ ./project2
From Process 1: counter = 399780.
Child with ID 7140 has just exited.
From Process 2: counter = 699983.
Child with ID 7141 has just exited.
From Process 3: counter = 901030.
Child with ID 7142 has just exited.
From Process 4: counter = 1100000.
Child with ID 7143 has just exited.
End of Program.
lee0436@gpel8:~/project2$ ./project2
From Process 1: counter = 399869.
Child with ID 7151 has just exited.
From Process 2: counter = 699988.
Child with ID 7152 has just exited.
From Process 3: counter = 900009.
Child with ID 7153 has just exited.
From Process 4: counter = 1100000.
Child with ID 7154 has just exited.
End of Program.
lee0436@gpel8:~/project2$ ./project2
From Process 1: counter = 399844.
Child with ID 7162 has just exited.
From Process 2: counter = 700030.
Child with ID 7163 has just exited.
From Process 3: counter = 899980.
Child with ID 7164 has just exited.
From Process 4: counter = 1100000.
Child with ID 7165 has just exited.
End of Program.
```

As you can see in the screenshot of the output, the total counter always results at 110,000. One thing to note about the result is that even though I gave each process a target of 100,000, 200,000, 300,000, and 500,000 respectively, their counter is consistently resulting at about 400,000, 700,000, 900,000, and 200,000 respectively.

III. Adjusting target of each process

I had given each process the target of 100,000, 200,000, 300,000, and 500,000 respectively. which resulted at about 400,000, 700,000, 900,000, and 200,000 respectively. That is about 2 to 3 times more than the target I had given them. So I've decreased the target to see if I can adjust the results of each counter. So after trying some different set of numbers, I gave them 25078, 58334, 108305, 908283 and this was the result:



```
lee0436@gpel8:~/project2$ ./project2
From Process 1: counter = 100156.
Child with ID 19337 has just exited.
From Process 2: counter = 200069.
Child with ID 19338 has just exited.
From Process 3: counter = 300001.
Child with ID 19339 has just exited.
From Process 4: counter = 1100000.
Child with ID 19340 has just exited.
End of Program.
lee0436@gpel8:~/project2$ ./project2
From Process 1: counter = 100140.
Child with ID 19354 has just exited.
From Process 2: counter = 200060.
Child with ID 19355 has just exited.
From Process 3: counter = 300051.
Child with ID 19356 has just exited.
From Process 4: counter = 1100000.
Child with ID 19357 has just exited.
End of Program.
lee0436@gpel8:~/project2$ ./project2
From Process 1: counter = 100097.
Child with ID 19370 has just exited.
From Process 2: counter = 200066.
Child with ID 19371 has just exited.
From Process 3: counter = 300049.
Child with ID 19372 has just exited.
From Process 4: counter = 1100000.
Child with ID 19373 has just exited.
End of Program.
lee0436@gpel8:~/project2$ ./project2
From Process 1: counter = 100052.
Child with ID 19384 has just exited.
From Process 2: counter = 200074.
Child with ID 19385 has just exited.
From Process 3: counter = 300081.
Child with ID 19386 has just exited.
From Process 4: counter = 1100000.
Child with ID 19387 has just exited.
End of Program.
```

I understand that the process runs in parallel and their counter isn’t protected. But I’m not so sure what kind of relationship there is between the value of the target and the result of the counter. My bad guess is that because there is 4 counters, the counter gets incremented about 4 times as much as the set target.