Programming Assignment 6

CSCE 313-503

4/15/2018

Khanh Nguyen

UIN# 525000335

Prof. Tanzir Ahmed

**Raw data for FIFO:**

|  |  |
| --- | --- |
| test1 results  with n=10k, b=50 | |
| w | time(s) |
| 5 | 22.35698 |
| 10 | 11.09 |
| 15 | 7.3737 |
| 20 | 5.504534 |
| 25 | 4.403469 |
| 30 | 3.790703 |
| 35 | 3.231393 |
| 40 | 2.798102 |
| 45 | 2.499317 |
| 50 | 2.278127 |
| 55 | 2.0927 |
| 60 | 1.944695 |
| 65 | 1.857902 |
| 70 | 1.699158 |
| 75 | 1.597411 |
| 80 | 1.512779 |
| 85 | 1.43268 |
| 90 | 1.394121 |
| 95 | 1.316433 |
| 100 | 1.283623 |
| 105 | 1.210263 |
| 110 | 1.165291 |
| 115 | 1.123334 |
| 120 | 1.10245 |
| 125 | 1.085334 |
| 130 | 1.067159 |
| 135 | 1.090014 |
| 140 | 1.039637 |
| 145 | 0.947392 |
| 150 | 0.939459 |
| 155 | 0.953246 |
| 160 | 0.905213 |
| 165 | 0.909927 |
| 170 | 0.88845 |
| 175 | 0.843838 |
| 180 | 0.827398 |
| 185 | 0.825066 |
| 190 | 0.797439 |
| 195 | 0.791124 |
| 200 | 0.761222 |
| 205 | 0.81448 |
| 210 | 0.780194 |
| 215 | 0.776044 |
| 220 | 0.77972 |
| 225 | 0.788753 |
| 230 | 0.686699 |
| 235 | 0.69843 |
| 240 | 0.700885 |
| 245 | 0.677965 |
| 250 | 0.677008 |

**Raw data for Message Queue:**

|  |  |
| --- | --- |
| test1 results | with n=10k, b=50 for MSQ |
| w | time(s) |
| 5 | 22.50347 |
| 10 | 11.17808 |
| 15 | 7.350964 |
| 20 | 5.47751 |
| 25 | 4.353654 |
| 30 | 3.58806 |
| 35 | 3.107659 |
| 40 | 2.717858 |
| 45 | 2.428585 |
| 50 | 2.17711 |
| 55 | 1.983854 |
| 60 | 1.817577 |
| 65 | 1.680864 |
| 70 | 1.561315 |
| 75 | 1.459935 |
| 80 | 1.370631 |
| 85 | 1.291864 |
| 90 | 1.225076 |
| 95 | 1.160696 |
| 100 | 1.10675 |
| 105 | 1.05965 |
| 110 | 1.02276 |
| 115 | 0.980976 |
| 120 | 0.932171 |
| 125 | 0.913709 |
| 130 | 0.874585 |
| 135 | 0.861858 |
| 140 | 0.838868 |
| 145 | 0.796538 |
| 150 | 0.732503 |
| 155 | 0.815592 |
| 160 | 0.750074 |
| 165 | 0.742437 |
| 170 | 0.73068 |
| 175 | 0.696096 |
| 180 | 0.67966 |
| 185 | 0.696555 |
| 190 | 0.651976 |
| 195 | 0.629056 |
| 200 | 0.643152 |
| 205 | 0.547049 |
| 210 | 0.559942 |
| 215 | 0.546034 |
| 220 | 0.551669 |
| 225 | 0.492242 |
| 230 | 0.552166 |
| 235 | 0.490397 |
| 240 | 0.566712 |
| 245 | 0.482213 |
| 250 | 0.471014 |

**Raw data for Shared memory with kernel semaphore:**

|  |  |
| --- | --- |
| test1 results | with n=10k, b=50 for SHM |
| w | time(s) |
| 5 | 22.54214 |
| 10 | 11.17524 |
| 15 | 7.344134 |
| 20 | 5.49342 |
| 25 | 4.328583 |
| 30 | 3.661301 |
| 35 | 3.144562 |
| 40 | 2.753719 |
| 45 | 2.473081 |
| 50 | 2.222681 |
| 55 | 2.044101 |
| 60 | 1.815996 |
| 65 | 1.65865 |
| 70 | 1.571671 |
| 75 | 1.4807 |
| 80 | 1.386465 |
| 85 | 1.29999 |
| 90 | 1.251756 |
| 95 | 1.174423 |
| 100 | 1.119311 |
| 105 | 1.082883 |
| 110 | 1.028836 |
| 115 | 0.992527 |
| 120 | 0.931815 |
| 125 | 0.913753 |
| 130 | 0.883533 |
| 135 | 0.84824 |
| 140 | 0.81932 |
| 145 | 0.805967 |
| 150 | 0.782975 |
| 155 | 0.753349 |
| 160 | 0.748067 |
| 165 | 0.727207 |
| 170 | 0.692551 |
| 175 | 0.680317 |
| 180 | 0.664644 |
| 185 | 0.658462 |
| 190 | 0.637832 |
| 195 | 0.631248 |
| 200 | 0.619953 |
| 205 | 0.60685 |
| 210 | 0.607649 |
| 215 | 0.600612 |
| 220 | 0.592817 |
| 225 | 0.58631 |
| 230 | 0.563492 |
| 235 | 0.57346 |
| 240 | 0.57346 |
| 245 | 0.552604 |
| 250 | 0.543969 |

**GRAPHS:**

For the comparison between that FIFO, Message Queue and Shared Memory IPCs, they have almost identical results, no significant difference obversed. I ran them on Ubuntu Linux system installed on my personal computer. The maximum w I ran was 250.

Differences between FIFO and Message queue and Shared Memory:

* FIFO is not limited in size, while queue and shared memory are.
* FIFO can be implemented with select() while the other two cant.
* FIFO and Message Queue are synchronized by kernel, while Shared Memory is not (must use Kernel Semaphore)

**To clear up the IPCs:**

In the Message Queue’s destructor, I used:

msgctl(serverID, IPC\_RMID, NULL);

msgctl(clientID, IPC\_RMID, NULL);

These two lines will destroy the message queues IPC created.

In the Shared Memory, I used: shmctl() function to release the memory segment used and semctl to release the semaphore arrays.

**Bonus part:**

**For the Message Queue**, I used mtype variable which is a private member of struc buf to distinguish data for Server and Client. In cread, I set mtype=1 if side is server, and mtype=2 if side is Client. In cwrite, I flipped the mtype. In other words, whenever side is server, mtype is =2, and mtype=1 when side== client.

That way I only used one buffer for two direction data transfer.

**For the Shared Memory,** instead of using two memory segments, I appended the second segment to the first segment. That way only one segment was used for data transfer (they still have different keys).

client\_data = (char\*) shmat(clientID, (void\*) 0, 0);

server\_data = (char\*) ((unsigned long)client\_data + ((unsigned long) SHM\_SIZE));