

Case 1: Looking up Tasks Precedence

Design:

All APIs are from the *windows.h* library.

- **Threads:** Three Threads; one writer thread and two looking up threads. All threads are created using the *CreateThread* API .

```
Thread1= CreateThread(NULL,0,Writer, NULL, CREATE_SUSPENDED, &Thread1ID);  
Thread2= CreateThread(NULL,0,LookUp1,NULL, CREATE_SUSPENDED, &Thread2ID);  
Thread3= CreateThread(NULL,0,LookUp2, NULL, CREATE_SUSPENDED, &Thread3ID);
```

And all threads are created suspended and resumed using *ResumeThread(Handle)* API.

```
ResumeThread(Thread1);
```

- **Semaphores:** Two binary semaphores with initial value of 1 and maximum value of 1, created using the *CreateSemaphore* API. One semaphore, semaphore_m, for the writer process so that other processes get blocked when it's executing and another one, semaphore_r, for blocking other processes when modifying the value of r which determines how many Lookup processes are active.

```
semaphore_r = CreateSemaphore(NULL,1,1, &sem_r);  
semaphore_m = CreateSemaphore(NULL,1,1,&sem_m);
```

The semaphores are decremented using the *WaitForSingleObject* API and released using the *ReleaseSemaphore* API.

```
WaitForSingleObject(semaphore_m,INFINITE);
```

```
ReleaseSemaphore(semaphore_m,1,NULL);
```

- **Variables:** Two global variables *r* for tracking the number of looking up functions and *shared_variable* that the writer process modifies.

- **Others:** *clock()* from *time.h* library to tell us about the execution time of each process.

Operation and testing:

- Upon start all threads are suspended when pressing '1' on keyboard the writer and looking up process 1 are activated where the writer blocks the looking up the process using *semaphore_m* and modifies the shared variable.

When it's done it unblocks the looking up process which increments the *r* variable, blocks the writer process and reads the *shared_variable*.

When it's done it decrements *r* and unblocks the writer process once again.

```

look up process 1
writer process blocked
look up process 1 id=13304 and execution time is 2246
  looked up item in process 1= 0
writer unblocked
writer process blocked
look up process 1 id=13304 and execution time is 2247
  looked up item in process 1= 0
writer unblocked
writer process blocked
look up process 1 id=13304 and execution time is 2247
  looked up item in process 1= 0
writer unblocked
writer process blocked
look up process 1 id=13304 and execution time is 2247
  writer process
  looked up item in process 1= 0
writer unblocked
other processes blocked
writer process id=10536 and execution time is 2247
shared variable is modified= 1
writer process done
writer process
writer process blocked
look up process 1 id=13304 and execution time is 2248
  looked up item in process 1= 1
writer unblocked
other processes blocked
writer process id=10536 and execution time is 2248
shared variable is modified= 2

```

- On pressing '2' on the keyboard looking up process 2 is resumed and now there's a race condition on the *r* variable so the *semaphore_r* becomes useful to prevent multiple 2 processes from modifying the same variable and blocking the writer process multiple times.

Only when $r=1$ does the writer process get blocked as it means that there's a process reading and when $r=0$ it gets unblocked which means that the processes are done.

```
writer process blocked
look up process 1 id=13304 and execution time is 1587078
  looked up item in process 1= 128
writer unblocked
look up process 2
other processes blocked
writer process id=10536 and execution time is 1587078
shared variable is modified= 129
writer process done
writer process
writer process blocked
look up process 1 id=13304 and execution time is 1587084
  looked up item in process 1= 129
look up process 2 id=6052 and execution time is 1587084
  looked up item in process 2= 129
look up process 1 id=13304 and execution time is 1587084
  looked up item in process 1= 129
look up process 2 id=6052 and execution time is 1587084
  looked up item in process 2= 129
look up process 1 id=13304 and execution time is 1587087
  looked up item in process 1= 129
look up process 2 id=6052 and execution time is 1587087
  looked up item in process 2= 129
look up process 1 id=13304 and execution time is 1587088
  looked up item in process 1= 129
look up process 2 id=6052 and execution time is 1587088
  looked up item in process 2= 129
look up process 1 id=13304 and execution time is 1587090
  looked up item in process 1= 129
look up process 2 id=6052 and execution time is 1587090
```

Also by resuming process 2, the processes will keep looking up the variable and the writer process will remain blocked without the ability to modify its value again.

Case 2: writer precedence

Design:

- Similar to case1 but we add a *readtry* semaphore to allow the older processes to finish reading and prevent new processes from reading so that the writer process doesn't remain blocked forever.

Operation and testing:

Test case 1:

```

while(getche() != '1');
    ResumeThread(Thread1);

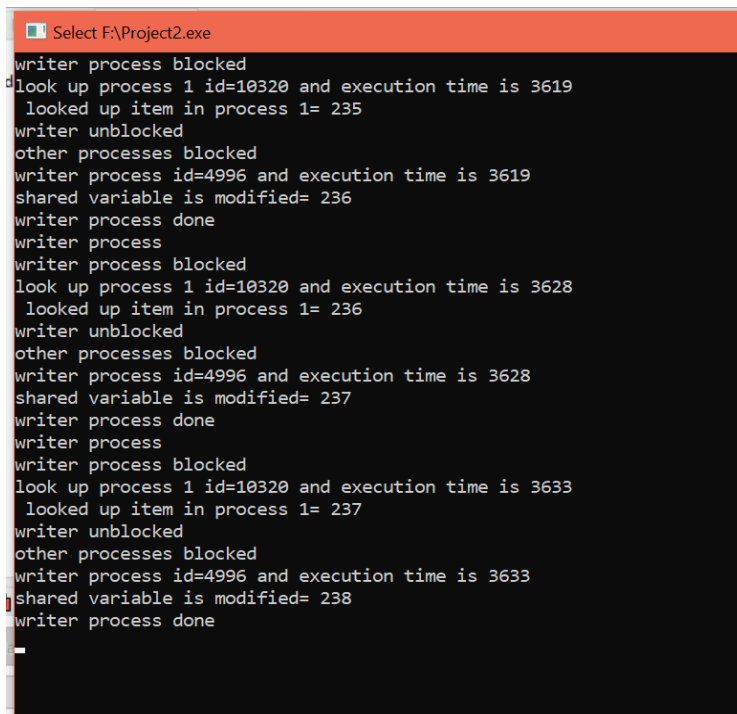
    ResumeThread(Thread2);

while(getche() != '2');

    ResumeThread(Thread3);

```

Similar operation upon as before: when by pressing '1' the writer process the writer and first look up process operate normally



```

Select F:\Project2.exe
writer process blocked
look up process 1 id=10320 and execution time is 3619
  looked up item in process 1= 235
writer unblocked
other processes blocked
writer process id=4996 and execution time is 3619
shared variable is modified= 236
writer process done
writer process
writer process blocked
look up process 1 id=10320 and execution time is 3628
  looked up item in process 1= 236
writer unblocked
other processes blocked
writer process id=4996 and execution time is 3628
shared variable is modified= 237
writer process done
writer process
writer process blocked
look up process 1 id=10320 and execution time is 3633
  looked up item in process 1= 237
writer unblocked
other processes blocked
writer process id=4996 and execution time is 3633
shared variable is modified= 238
writer process done

```

- When pressing '2' the writer process doesn't get infinitely blocked. Because when the writer process is done, process 1 executes and blocks all other processes, then releases them. Similarly, process 2 executes and blocks all other processes then releases them when it's done so the writer process is able to execute again.

Meanwhile in the previous case when the first process starts it blocks the writer process until all looking up processes are done.

```
look up process 2 id=964 and execution time is 41933
looked up item in process 2= 1268
writer unblocked
other processes blocked
writer process id=4996 and execution time is 41933
shared variable is modified= 1269
writer process done
writer process
writer process blocked
look up process 1 id=10320 and execution time is 41934
looked up item in process 1= 1269
look up process 2 id=964 and execution time is 41934
looked up item in process 2= 1269
writer unblocked
other processes blocked
writer process id=4996 and execution time is 41934
shared variable is modified= 1270
writer process done
writer process
writer process blocked
look up process 1 id=10320 and execution time is 41934
looked up item in process 1= 1270
look up process 2 id=964 and execution time is 41934
looked up item in process 2= 1270
writer unblocked
other processes blocked
writer process id=4996 and execution time is 41934
shared variable is modified= 1271
writer process done
```

Test case 2:

```
while(getche() != '1');
```

```
ResumeThread(Thread2);
```

```
while(getche() != '2');
```

```
ResumeThread(Thread1);
```

```
ResumeThread(Thread3);
```

On pressing '1' look up process1 is resumed and on pressing '2' the writer and looking up process2 are resumed.

```

writer process blocked
look up process 1 id=16172 and execution time is 2739
  looked up item in process 1= 0
writer unblocked
writer process blocked
look up process 1 id=16172 and execution time is 2742
  looked up item in process 1= 0
writer unblocked
writer process blocked
look up process 1 id=16172 and execution time is 2744
  looked up item in process 1= 0
writer unblocked
writer process blocked
look up process 1 id=16172 and execution time is 2746
  looked up item in process 1= 0
writer unblocked
writer process blocked
look up process 1 id=16172 and execution time is 2748
  looked up item in process 1= 0
writer unblocked
writer process blocked
look up process 1 id=16172 and execution time is 2753
  looked up item in process 1= 0
writer unblocked
writer process blocked
look up process 1 id=16172 and execution time is 2755
  looked up item in process 1= 0
writer unblocked
writer process blocked

```

we can see that by pressing '1' the look up process behaves normally.

```

look up process 2 id=13328 and execution time is 57640
looked up item in process 2= 115
look up process 1 id=16172 and execution time is 57646
  looked up item in process 1= 115
writer unblocked
other processes blocked
writer process id=5808 and execution time is 57646
shared variable is modified= 116
writer process done
writer process
writer process blocked
look up process 2 id=13328 and execution time is 57646
looked up item in process 2= 116
look up process 1 id=16172 and execution time is 57656
  looked up item in process 1= 116
writer unblocked
other processes blocked
writer process id=5808 and execution time is 57656
shared variable is modified= 117
writer process done
writer process
writer process blocked
look up process 2 id=13328 and execution time is 57656
looked up item in process 2= 117
look up process 1 id=16172 and execution time is 57662
  looked up item in process 1= 117
writer unblocked
other processes blocked
writer process id=5808 and execution time is 57662

```

On pressing '2' the writer and look up process 2 operate in the same way and the writer process never gets blocked infinitely.