National Textile University

Department of Computer Science

Subject:

Opreating System

Submitted to:

Sir Nasir

Submitted by:

Ahmad Fawad

Reg number:

1129

Assignment:

02

Semester:

5th

Operating Systems – COC 3071

SE 5th A – Fall 2025

After-mid Homework -1

Part 1: Semaphore theory

1. A counting semaphore is initialized to 7. If 10 wait() and 4 signal() operations are performed, find the final value of the semaphore.

Ans :

semaphore is initialized with 7

10 wait() (operations)

it will decrement 10 times in semaphore value (S-1) each execution of code for Enter . Remaining value is **-3**

4 signal() (operations)

it will increment in Semaphore 4 times (S+1) each execution of code for exit . Remaining value is **1**

final value is : 1

2. A semaphore starts with value 3. If 5 wait() and 6 signal() operations occur, calculate the resulting semaphore value.

Ans:

semaphore is initialized with 3

5 wait() (operations)

it will decrement 5 times in semaphore value (S-1) each execution of code for Enter . Remaining value is **-2**

6 signal() (operations)

it will increment in Semaphore 6 times (S+1) each execution of code for exit . Remaining value is **4**

final value is : 4

3. A semaphore is initialized to 0. If 8 signal() followed by 3 wait() operations are executed, find the final value.

Ans:

semaphore is initialized with 0

8 signal() (operations)

it will increment in Semaphore 8 times (S+1) each execution of code for exit . Remaining value is **8**

3 wait() (operations)

it will decrement 3 times in semaphore value (S-1) each execution of code for Enter . Remaining value is **5**

final value is : 5

4. A semaphore is initialized to 2. If 5 wait() operations are executed:

Ans:

a) How many processes enter the critical section?

Semaphore is initiliezwd with 2 its mean 2 proceses are successfully enter in critical section

final value : 2

b) How many processes are blocked?

If 5 wait then it decrement 5 (S-1) times so final value in blocked section is **3**

final value :3

5. A semaphore starts at 1. If 3 wait() and 1 signal() operations are performed:

Ans:

a) How many processes remain blocked?

Initial value is 1 and 3 wait () 1-3 = -**2**

1 signal operation 1 signal() -2 + 1 = **-1**

b) What is the final semaphore value?

Final value is : -1

6.

Ans

semaphore S = 3;
wait(S); wait (S-1) = 2
wait(S); wait (S-1) = 1
signal(S); S+1 = 2
wait(S); S-1 = 1
wait(S); S-1 = 0

a) How many processes enter the critical section?

4 wait() operation calls and Every wait() section will be successed

Final Value: 4 processes

b) What is the final value of S?

Final value of S is 0

7.

semaphore S = 1;
wait(S); S-1 = 0
wait(S); S-1 = -1
signal(S); S+1 = 0
signal(S); S+1 = 1

a) How many processes are blocked?

No one is in block section 1 process can go in block section but signal can woke up it again

Final value is : 0

b) What is the final value of S?

Final value is : 1

8. A binary semaphore is initialized to 1. Five wait() operations are executed without any signal(). How many processes enter the critical section and how many are blocked?

Ans:

Initial value is 1 So:
1st wait () call will be successfully enter in critical section remaing 4 go to block section

Final value in critical section is : 1 And in bloack section is 4

9. A counting semaphore is initialized to 4. If 6 processes execute wait() simultaneously, how many proceed and how many are blocked?

Ans:

Initial S = 4
Execution : 4 processes executed successfully remaining 2 go to block section
final processes : 4
blocked : 1

10. A semaphore S is initialized to 2. wait(S);

Ans:

a) Track the semaphore value after each operation.

wait(S); S-1 = 1
wait(S); S-1 = 0
wait(S); S-1 = -1
signal(S); S+1 = 0
signal(S); S+1 = 1
wait(S); S-1 = 0

b) How many processes were blocked at any time?
1 process block at any time

11. A semaphore is initialized to 0. Three processes execute wait() before any signal(). Later, 5 signal() operations are executed.

Ans:

a) How many processes wake up?

All block process were wake up total is 3

b) What is the final semaphore value?

final value is 2

Part 2: Semaphore Coding

Consider the Producer–Consumer problem using semaphores as implemented in Lab-10 (Lab-plan attached). Rewrite the program in your own coding style, compile and execute it successfully, and explain the working of the code in your own words.

Submission Requirements:

- Your rewritten source code
- A brief description of how the code works
- Screenshots of the program output showing successful execution

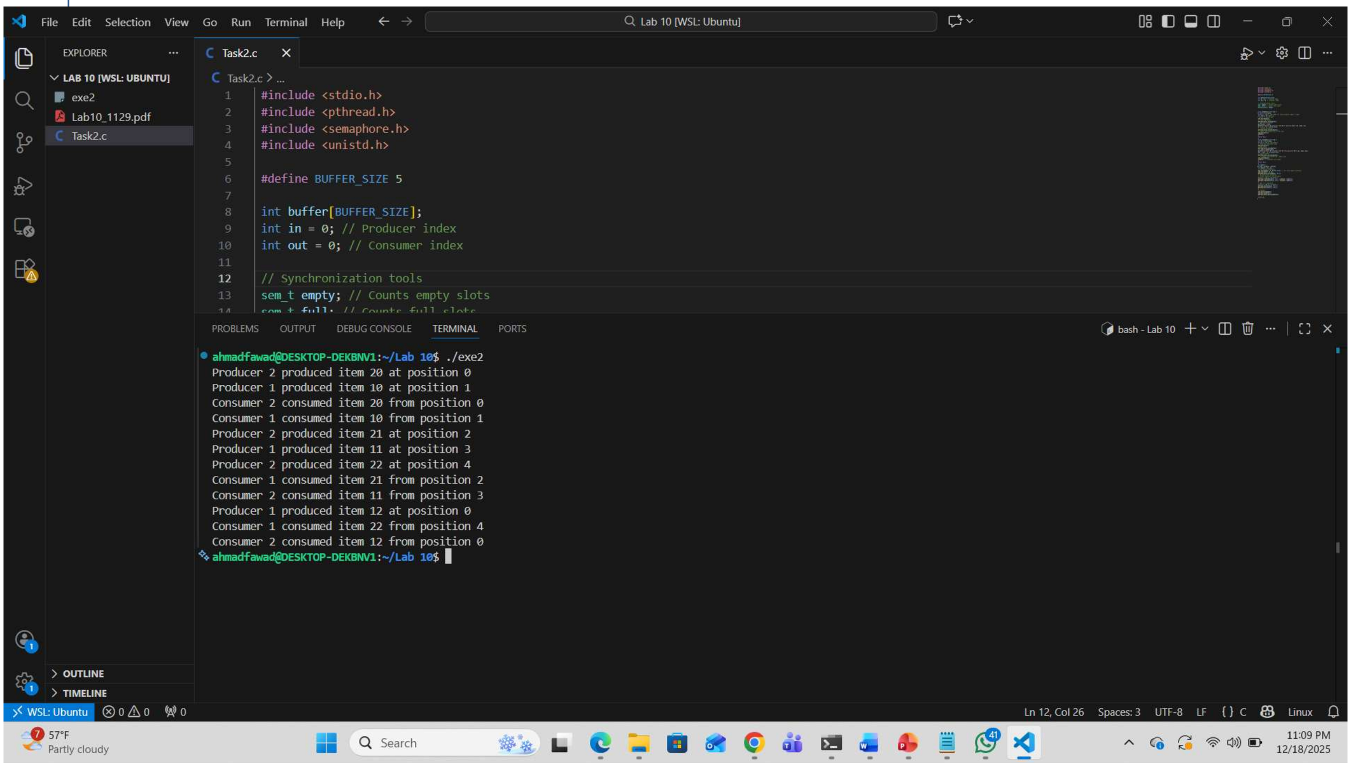
Code:

```
        #include <stdio.h>
 #include <pthread.h>
 #include <semaphore.h>
 #include <unistd.h>

 #define BUFFER_SIZE 5

 int buffer[BUFFER_SIZE];
 int in = 0; // Producer index
 int out = 0; // Consumer index

 // Synchronization tools
 sem_t empty; // Counts empty slots
 sem_t full; // Counts full slots
 pthread_mutex_t mutex;


 void* producer(void* arg) {
int id = *(int*)arg;
 for(int i = 0; i < 3; i++) { // Each producer makes 3 items
 int item = id * 10 + i;
// check for empty slot
 sem_wait(&empty);
 //  Lock the buffer
 pthread_mutex_lock(&mutex);
 // Add item to buffer
 buffer[in] = item;
 printf("Producer %d produced item %d at position %d\n",id, item, in);
 in = (in + 1) % BUFFER_SIZE;
 //  Unlock the buffer
 pthread_mutex_unlock(&mutex);
 // Signal that buffer has a full slot
 sem_post(&full);
 sleep(1);
 }
 return NULL;
```

```
}
void* consumer(void* arg) {
int id = *(int*)arg;
for(int i = 0; i < 3; i++) {
// check for available item
sem_wait(&full);
//lock buffer
pthread_mutex_lock(&mutex);
int item = buffer[out];
printf("Consumer %d consumed item %d from position %d\n",id, item, out);
out = (out + 1) % BUFFER_SIZE;
// unlock buffer
pthread_mutex_unlock(&mutex);
  // signal that buffer has  empty slot
sem_post(&empty);
sleep(2); // Consumers are slower
}
return NULL;
}
int main() {
pthread_t prod[2], cons[2];
int ids[2] = {1, 2};
// Initialize semaphores
sem_init(&empty, 0, BUFFER_SIZE); // All slots empty initially
sem_init(&full, 0, 0);
pthread_mutex_init(&mutex, NULL);
// No slots full initially
// Create producers and consumers
for(int i = 0; i < 2; i++) {
pthread_create(&prod[i], NULL, producer, &ids[i]);
pthread_create(&cons[i], NULL, consumer, &ids[i]);
}
// Wait for completion
for(int i = 0; i < 2; i++) {
pthread_join(prod[i], NULL);
pthread_join(cons[i], NULL);
}
// Cleanup
sem_destroy(&empty);
sem_destroy(&full);
pthread_mutex_destroy(&mutex);

return 0;
```

}

Output:

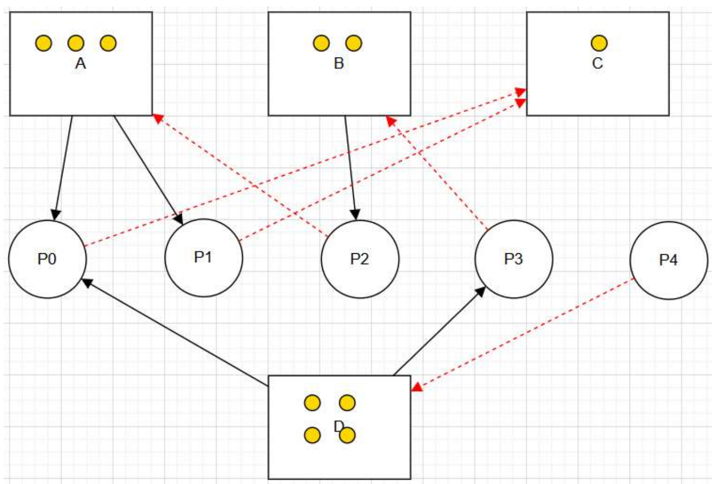


Discription:

- This program can be performed synchronization by using multi threading and semaphore
- In this program we use 2 semaphore one (empty) which track which the free space in buffer
- Second full which can track who many items should be ready for counsume
- We use mutex which can verify that at a time only one thread can be access shared resource so that race condition cant occur
- (sem_wait(&empty) producer first check that there is any free space avalible if yes it can produce data in buffer and generate signle to counsumer (sem_post(&full)
- (sem_wait(&full)) consumer will wait until buffer is free . after consuming data it will generate signal to producer that there is 1 sapace avalible (sem_post(&empty))

Part 3: RAG (Recourse Allocation Graph)

- Convert the following graph into matrix table ,



Ans:

Resource allocation Graph:-

In to matrix:-

Allocation Matrix:-

$$\begin{array}{c} \\ P_0 \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{array}
\begin{array}{cccc} A & B & C & D \\ \hline 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}$$

Request Matrix:-

$$\begin{array}{c} \\ P_0 \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{array}
\begin{array}{cccc} A & B & C & D \\ \hline 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array}
\begin{array}{c} \\ \checkmark \\ \checkmark \\ \checkmark \\ \checkmark \\ \checkmark \end{array}$$

Part 4: Banker's Algorithm

System Description:

- The system comprises five processes (P0–P3) and four resources (A,B,C,D).
- Total Existing Resources:

Total			
A	B	C	D
6	4	4	2

- Snapshot at the initial time stage:

	A	B	C	D	A	B	C	D	A	B	C	D
P0	2	0	1	1	3	2	1	1				
P1	1	1	0	0	1	2	0	2				
P2	1	0	1	0	3	2	1	0				
P3	0	1	0	1	2	1	0	1				

Questions:

1. **Compute the Available Vector**:
    - Calculate the available resources for each type of resource.
2. **Compute the Need Matrix**:
    - Determine the need matrix by subtracting the allocation matrix from the maximum matrix.
3. **Safety Check**:
    - Determine if the current allocation state is safe. If so, provide a safe sequence of the processes.
    - Show how the Available (working array) changes as each process terminates.

Banker's Algorithm:-

1 Allocation Matrix:

	A	B	C	D
P_0	2	0	1	1
P_1	1	1	0	0
P_2	1	0	1	0
P_3	0	1	0	1

Max :-

	A	B	C	D
P_0	3	2	1	1
P_1	1	2	0	2
P_2	3	2	1	0
P_3	2	1	0	1

~~Available~~:

A.V = Total - Sum Allocated.

Sum of Allocation of Resources is:-

$A = 2+1+1+0 = 4$

$B = 0+1+0 1 = 2$

$C = 1 + 0+1+0 = 2$

$D = 1+0+0+1 = 2$

Avalible vector:-

$A \rightarrow 6-4 = 2$

B = 4 - 2 = 2
C = 4 - 2 = 2
D = 2 - 2 = 0

2) Need Matrix:- Max - Allocation.

	A	B	C	D
P0	1	2	0	0
P1	0	1	0	2
P2	2	2	0	0
P3	2	0	0	1

3) Saftey check:-
Avalibe vector is : [2, 2, 2, 0]

1) check P0:-
Need : [1, 2, 0 0] ≤ [2, 2, 2, 0].
Resources will assign, full fill its need and also release allready allocated resources:-
Avalible += Avalible [P0].
[2, 2, 2, 0] + = [2, 0, 1, 1].
S = [4, 2, 3, 1]
Sequence = [P0]

2) Check P1:
Avalible vector is [4, 2, 3, 1]

Need : [0,1 ,0,2] ≤ [4,2, 3,1]

False State. Skip it.

3) Check P_2:-

Need: [2,2,0,0] ≤ [4,2, 3, 1]

Its true so resourse released after use:-

: Avalible + = Allocation [P2]

= [5,2, 4,1]

Sequense [P0, P2]

4) check P_3:

Need [2,0,0,0] ≤ [5,2, 4, 1]

True State:-

Avalible + = Allocation [P3]

= [5 , 3, 4, 2]

Sequence :-

[P0, P2, P3]

5) Again check P1:

Need [0,1, 0, 2] ≤ [5, 3, 4, 2]

True State:-

Avalible + = Allocation[P1]

Avalible = [6, 4, 4, 2] ·

Sequence = [P_0, P_2, P_3, P_1]

Submission Guidelines:

- Ensure all answers are well-explained and calculations are shown step-by-step.
- Submit your assignment on MS Team and GitHub in a PDF format.
- VIVA based Evaluation so Develop your own solution after getting help.