VARUN VERMA
(Roll no - 38) CS - B1
1801010170

Operating System Assignment -2

Date of Submission :- 02/04/2020

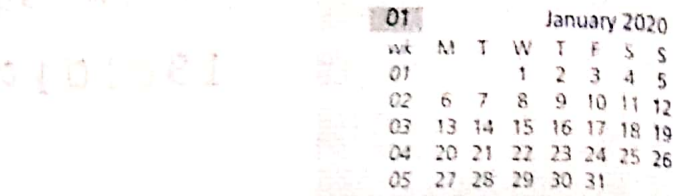
Question / Answer

Q1] Implement the Dekker's solution in order to solve the critical section problem in Producer - Consumer with finite buffer

Ans :— Dekker's solution implementation on producer process ⇒

```
while (true) {
    while ( counter == BUFFER_SIZE)
        ; //do nothing
    buffer[in] = nextProduced;
    in = (in+1) % BUFFER_SIZE;
    flag [i] = true;
    while (flag [j] == true)
    {
        if (turn == j)
        {
            flag [i] = false;
            while ( turn == j)
            {
            }
            flag [i] = true;
```

NOTES




```
        }
    }
    counter ++ ;        // Critical section
    turn = j ;
    flag [i] = false
    //remainder section
}
```

* Dekker's solution implementation on consumer process ⇒

```
while (true) {
    while (counter == 0)
        ; // do nothing
    nextConsumed = buffer [out ];
    out = (out +1) % BUFFER_SIZE ;
    flag [i] = true ;
    while ( flag [j] == true )
    {
        if (turn == j )
        {
            flag [i] = false ;
            while (turn == j )
            {
            }
            flag [i] = true ;
        }
    }
    counter -- ; // now critical section
```

Varun Verma


```
turn = j;
flag[i] = false;
// Remainder section
}
```

Q2] Implement the Peterson's solution in order to solve the critical section problem in Producer Consumer with finite buffer.

Ans :— Peterson's solution implementation on Producer process ⇒

```
while (true) {
    while (counter == BUFFER_SIZE)
        ; // do nothing
    buffer[in] = nextProduced;
    in = (in + 1) % BUFFER_SIZE;
    flag[i] = true;
    turn = j;
    while ( flag[j] && turn == j)
    {
    }
    counter++; // Critical section
    flag[i] = false;
    // Remainder section
}
```

Sunday 12

NOTES

Varun Verma

13 | 2020 Monday January

013-353 • WK 03

01 January 2020

wk	M	T	W	T	F	S	S
01			1	2	3	4	5
02	6	7	8	9	10	11	12
03	13	14	15	16	17	18	19
04	20	21	22	23	24	25	26
05	27	28	29	30	31		

Peterson's Solution implementation on Consumer process :-

```
while (true) {
        while (counter == 0)
            ; // do nothing
        nextConsumed = buffer[out];
        out = (out+1) % BUFFER_SIZE;
        flag[i] = true;
        turn = j;
        while (flag[j] && turn == j)
        {
        }
        counter--; // critical section
        flag[i] = false;
        // Remainder section
}
```

Q3] What is Dinning Philosophers problem?

Ans → Consider five philosophers who spend their lives thinking and eating. The philosophers share a circular table surrounding by five chairs, each belonging to one philosophers In the centre of the table is a bowl of rice, and the table is laid with five single chopsticks.

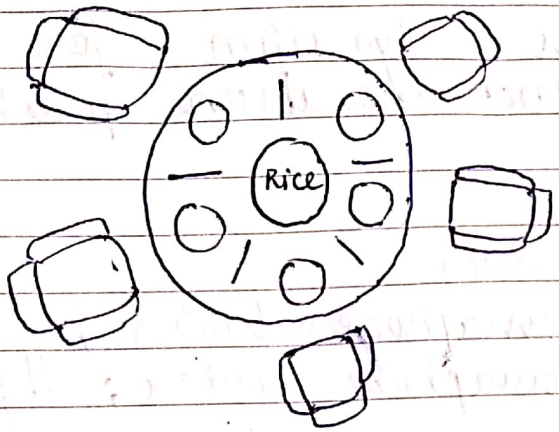
NOTES

Varun Verma

February 2020

wk	M	T	W	T	F	S	S
05						1	2
06	3	4	5	6	7	8	9
07	10	11	12	13	14	15	16
08	17	18	19	20	21	22	23
09	24	25	26	27	28	29	

2020
Tuesday
January
14
014-352 • WK 03



Situation of the dinning philosphers.

When a philosopher thinks, she does not interact with her colleagues. From time to time, a philosopher gets hungry and tries to pick up the two chopsticks that are closest to her (the chopsticks that are b/w her and her left & right neighbours). A philosopher may pick up only one chopstick at a time. Obviously, she cannot pick up a chopstick that is already in the hand of a neighbour. When a hungry philosopher has both her chopsticks at the same time, she eats without releasing her chopsticks. When she is finished eating, she puts down both of her chopsticks and starts thinking again.

NOTES

Varun Verma

15 | 2020 Wednesday January

015-351 • WK 03

01 January 2020

wk	M	T	W	T	F	S	S
01			1	2	3	4	5
02	6	7	8	9	10	11	12
03	13	14	15	16	17	18	19
04	20	21	22	23	24	25	26
05	27	28	29	30	31		

Q4] write an algorithm for the sleeping Barber synchronization problem using semaphores

Ans → shared data

```
semaphore barber ;
semaphore mutex ; // for mutual exclusion
semaphore customer ; // no. of customer waiting
int NumberOfseats ; // no. of seats vacant
```

Initially

```
barber = 0 ;
mutex = 1 ;
customer = 0 ;
NumberOfseats = n ;
```

The structure of Barber Process

```
do {
    wait ( customer) ; // if none avail. go to sleep.
    wait (mutex) ;
    NumberOfseats ++ ;
    signal (barber) ;
    signal (mutex) ;
    //haircut
} while (true) ;
```

NOTES

The structure of Customer Process

```
do {
    wait(mutex); //enter critical section
    if (Number Of Seats > 0)
    {
        Number Of Seats --;
        signal (Customer);
        signal (mutex);
        wait (barber);
        // haircut
    }
    else {
        signal (mutex);
        // leave without a haircut
    }
} while (true);
```

Q5) Give the soution of second Reader-Writer problem by using the concept of semaphore?

Ans → shared Data

```
int readcount;
int writecount;
semaphore rmutex;
semaphore wmutex;
semaphore readtry;
semaphore resource;
```

NOTES

Varun Verma

17 | 2020 Friday January

017-349 • WK 03

01 January 202(

wk	M	T	W	T	F	S	
01			1	2	3	4	
02	6	7	8	9	10	11	1
03	13	14	15	16	17	18	1
04	20	21	22	23	24	25	2
05	27	28	29	30	31		

Initially

```
readcount = 0;
writecount = 0;
rmutex = 1;
wmutex = 1;
readtry = 1;
resource = 1;
```

The structure of a Reader Process

```
do {
    wait (readtry);
    wait (rmutex);
    readcount++;
    if (readcount == 1)
        wait (resource);
    signal (rmutex);
    signal (readtry);

    // Reading is performed

    wait (rmutex);
    readcount--;
    if (readcount == 0)
        signal (resource);
    signal (rmutex);
} while (true);
```

NOTES

Varun Verma

9.00 The structure of a writer Process

10.00 do {

```
wait (wmutex);
writecount++;
if (writecount == 1)
        wait (readtry);
signal (wmutex);
wait (resource);

// writing is performed

signal (resource);
wait (wmutex);
writecount--;
if (writecount == 0)
        signal (readtry);
signal (wmutex;
} while (true);
```