Name: Abhay Upadhaya Class: SE-IT Roll no: 57
Sub: OS

ASSIGNMENT 03

Q.1> What is Mutual Exclusion? Explain its significance.

Ans: 1) Mutual exclusion is one of the conditions for a deadlock to occur.
2) It must be holding true for non-shareable resources.
3) Non-sharable resources include printer, memory space, etc.

- Significance of mutual exclusion:
1) Prevents race around conditions.
2) Prevents Multiple threads to enter critical section at the same time.

Q.2> Explain race condition with example?

Ans: 1) A race condition takes place when more than one process write data items so that the final result depends on the order of execution of instructions in the multiple processes.
2) Consider two processes, P_1 and P_2, which share the global variable "x". At the time of execution process P_1 updates "x" to the value 1 and at the time of execution of another process, P_2 updates "x" to the value 2.
3) Thus, the two tasks are in a race to write variable "x". In this example the process that modifies x value lastly decides the final value of "x".

Q.3) Explain Critical Section problem. What are t requirements to solve critical-section pro

Ans: 1) The portion of the program where the shar memory is accessed is referred as the Criti section.

2) In order to avoid race conditions and faulty results, one must be able to recognize codes in Critical sections in each thread

3) The typical properties of the code that comprises critical section are as follows
- These codes reference one or more variab in a "read-update-write" way.
- At the same time, any of these variables ma changed by another thread.
- These codes modify one or more variables that can be referenced in "read-update-w fashion by another thread.
- Any part of data structure used by the cod can be modified by another thread.
- Codes modify any part of a data structure t is currently in use by another thread.

4) When one process is currently executing shared modifiable data in its critical section, n other process is to be permitted to execute in its critical section.

5) Hence, the execution of critical sections by the processes is mutually exclusive in time.

6) In a code called as critical section, only one process executes at a time.

7) In a critical section problem an algorithm n

to be designed which permits at most one process into the critical section at a time, with no deadlock.

8) The critical section's problem solution must obey mutual exclusion, progress and bounded waiting. Any process directly cannot enter in critical section.

9) First process has to obtain permission for its entry in its critical section. The segment of code which implements this appeal of process is the entry section. When process comes out of the critical section after completing its execution, there it has to execute exit section.

10) The rest of the code is the remainder-section.

11) Any solution to the critical-section problem must satisfy the following three necessary conditions:

i) Mutual exclusion:
At a time, single process should be executing in critical section (CS). If currently Process P_i is exceeding in its CS then other processes should not execute in their critical sections.

ii) Progress:
If Process's CS is free, means it is not executing in its CS. In this situation suppose some processes desire to go into their critical sections for execution.

iii) Bounded waiting:
A limit should be set on the number of times that other processes are permitted to go into their critical sections. after a process has made a request to go to its critical section and before

that request is approved.

12) Semaphore and monitor are the solutions to achieve mutual exclusions

13) A synchronization variable taking positive integer values is called semaphore. Binary semaphore only has two values 0 or 1.

14) ~~A synchronization variable taking positive~~ in

14) Hardware does not supply the semaphore. Semaphore offer the solution to critical section problem.

15) If semaphore variable takes value greater than 1 then it's called as counting semaphore.

Q. 5) ~~Differen~~ what is semaphore? Elaborate with the significance of semaphore?

Ans: 1) A semaphore S is integer variable where it can be accessed and changed only by two operations wait and signal. wait and signal are atomic operations.

2) Binary semaphores does not assume all the integer values.

3) It assumes only two values 0 and 1. On the contrary, counting semaphores can assume only non-negative values.

4) The wait operations on semaphores S, written as wait(S) or P(S), operates as follows:

```
wait(S): If S>0
         Then S:=S-1
         Else (wait on S)
```

The signal operation on semaphore S, written as signal(S) or V(S), operates as follows:

signal(S): If (one or more process are waiting on S)
Then (let one of these processes proceed)
Else S := S+1

5) Differentiate counting and binary semaphores?

Ans:

Counting Semaphore	Binary semaphore
1° Can take any initial value	1° Can take only take 0 or 1
2° V operations never blocks.	2° Both P and V operation may block.
3° Completed P and V operations do not have to alternate.	3° Completed P and V operations must alternate.
4° If the initial value is 0, the first completed operation	4° If the initial value is 0, the first completed operation must be V, if the initial value is 1, the first completed operation must be P.

6) Explain an algorithm for producer - consumer problem?

Ans: 1) In operating system Producer is a process which is able to produce data/item.

2) Consumer is a process that is able to consume the data/item produced by the Producer.

3) Both Producer and Consumer share a co
memory buffer. This buffer is a space of
size in the memory of the system which
for storage. The producer produces th
into the buffer and the consumer consu
the data from the buffer.

So, what are the Producer - Consumer Pro
1. Producer Process should not produce an
data when the shared buffer is full.
2. Consumer Process should not consume any
data when the shared buffer is empty.
3. The access to the shared buffer should
mutually exclusive i.e at a time only one
process should be able to access the sha
buffer and make changes to it.
For consistent data synchronization betwee
Producer and Consumer, the above probl
should be resolve.

Q.7.7 Explain how reader writer problem can be
solved with semaphore?

Ans: 1) The readers - writers problem relates to an ob
such as a file that is shared between multipl
processes. Some of these processes are reader
i.e they only want to read the data from
object and some of the processes are write
they want to write into the object.
2) The readers - writers problem is used to ma
Synchronization so that there are no proble
with the object data.

3) For Example: If two readers access the object at the same time there is no problem. However, if two writers or a reader and writer access the object at the same time, there may be problems.

4) To solve this situation, a writer should get exclusive access to an object. However, multiple readers can access the object at the same time.

5) This can be implemented using semaphore. The codes for the reader and writer process in the reader-writer problem are given as follows-

Reader Process

```
wait (mutex);
rc++;
if (rc == 1)
wait (wrt);
signal (mutex);

wait (mutex)
rc--;
if (rc == 0)
signal (wrt);
signal (mutex);
```

Writer Process

```
wait (wrt);
signal (wrt);
```

Q.8.) Explain dining philosopher problem and s to it?

Ans:- 1) The dining philosopher problem states that are 5 philosophers sharing a circular ta they eat and think alternatively.

2) There is a bowl of rice for each of the phil and 5 chopsticks. A philosophers needs bo their right and left chopstick to eat.

3) A hungry philosopher may only eat if the are both chopsticks available.

4) Otherwise, a philosopher puts down their -ks and begins thinking again.

5) This problem is a classic synchronizatio problem as it demonstrates large cl of concurrency control problems.

- Solution of Dining Philosopher problem:

1) A solution to the Dining Philosophers Prob is to use a semaphore to represent a cho A chopstick can be picked up by executing a wait operation on the semaphore and re by executing a signal semaphore.

2) The structure of chopstick is shown belo semaphore chopstick [5];

3) Initially, the elements of the chopstic are initialized to 1 as the chopsticks are on th table and not picked up by a philosopher

4) The structure of a random philosopher is given as follows-

```
do {
    wait ( chopstick[i] );
    wait ( chopstick[(i+1) % 5]) ;

    signal ( chopstick [i]);
    signal ( chopstick [(i+1) % 5]);

  } while (1);
```

5) In the above structure, first wait operation is performed on chopstick [i] and chopstick [(i+1)%5]. This means that the philosopher I have picked up the chopsticks on his sides. Then the eating function is performed.

6) After that, signal operation is performed on chopstick [i] and chopstick [(i+1) % 5]. this means that the philosopher I have eaten and put down the chopsticks on his sides. Then the philosopher goes back to thinking.

Q.9) Explain Sleeping Barber problem? solution to it?

Ans: i) The sleeping Barber problem, states the analogy is based upon a hypothetical barber shop with one barber.

2) There is a barber shop which has one barber, one barber chair, and n chairs for waiting for customers if there are any to sit on the chair.

3) If there is no customer, then the barber sleeps in his own chair.

4) when a customer arrives, he has to wake up.

the barber.

5) If there are many customers and the barber is cutting a customer's hair, then the remaining customers either wait if there are empty chairs in the waiting room or they leave if no chairs are empty.

- Solution:

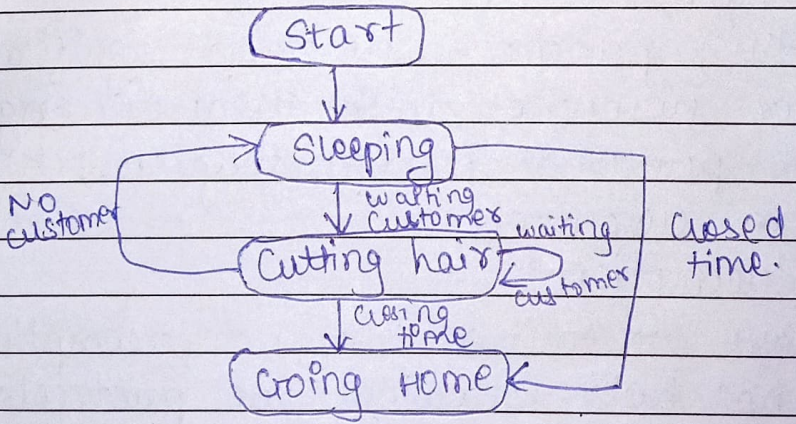
1) The solution to this problem includes three semaphores.
2) First is for the customer which counts the number of customers present in the waiting room.
3) Second, the barber 0 or 1 is used to tell whether the barber is idle or is working. And the third mutex is used to provide the mutual exclusion which is required for the process to execute.
4) In the solution, the customer has the record of the number of customers waiting in the waiting room if the number of customers is equal to the number of chairs in the waiting room then the upcoming customers leaves the barbershop.
5) When the barber shows up in the morning, he executes the procedure barber, causing him to block on the semaphore customers because it is initially 0. Then the barber goes to sleep until the first customer comes up.

6.) when a customer arrives, he executes customer procedure the customer acquires the mutex for entering the critical region, if another customers enters thereafter, the second one will not be able to anything until the first one has released the mutex.

7.) The customer then checks the chairs in the waiting room if waiting customers are less then the number of chairs then he sits otherwise he leaves and releases the mutex.

8.) If the chair is available the customer sit in the waiting room and increments the variable waiting value and also increases the customer's semaphore this wakes up the barber if he is sleeping.

9.) At this point, customer and barber are both awake and the barber is ready to give that person a haircut. When the hair cut is over, the customer exists the procedure and if there are no customers in waiting room barber sleeps.



Q.10) What are four conditions that create a deadl

Ans: 1. The four conditions that create a deadlock

a) Mutual enclusion:

- Non-sharable resources cannot be shared by process simultaneously.
- If resources are non-sharable then mutual -sion condition must hold for them.
- For example, It is not possible to use pointer many processes at the same time. Pointer is non-sharable but read only file is sharable resource and do not require mutually excl access.
- Therefore sharable resources such as read o file cannot be involved in a deadlock.

b) Hold and wait:

- Here we have to see that hold and wai will never occur. There should be an assu that, a process should request a resour only if it does not hold any other resource.
- In this, process will keep significant and popular resources with them forcing to wai other processes indefinitely which require these resources.

c) No Preemption:

- We have to ensure that no preemption condition does not hold. Following two protocols can be used.

- If a process at present keep some resources with it and requests another resource that cannot be immediately allocated to it, then all resources at present being held should be preempted.
- The list of the resources for which process is waiting is maintained. The preempted resources from the process are included in this list.
- The process now restart the execution provided that it can get back its old preempted resources in addition to the new ones that it is requesting.

d) Circular wait:

- To prevent the occurring of circular-wait condition inflict a total ordering of all resource types. In this case each process should request the resources in an increasing order of assigned numbers to the resources.
- Consider the ~~condition~~ collection of resources types $R = \{R_1, R_2, \ldots, R_n\}$. To this each resource type, a unique integer number is assigned.
- As resources are now recognized by their numbers it is easy to compare them.
- It is also easy to know if one resource precedes the other in our ordering.

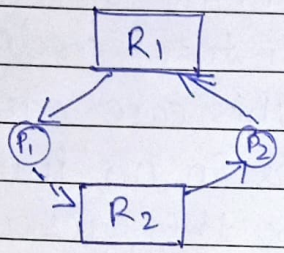
Q.11) Write short note: a) Recourse allocation Graph
b) Deadlock Avoidance
c) Dead lock detection
d) banker's algorithm.

Ans:

a) Recourse allocation Graph:

- If system has resource allocation system with only one instance of each resource type, then only this algorithm is applicable.
- Claim edge in resource allocation graph is like a future request edge.
- When process request a resource, the cla edge is converted to a request edge.
- When a process releases, the assignment edg is converted to a claim edge.

Cycle detection: $O(n^2)$



b) Deadlock Avoidance:

- Dead lock avoidance requires that the system has some information available up front. Initially every process declares the maximum number of resources need which it may requir
- An alternative method for avoiding deadlocks is to require additional information about how resources are to be requested.
- If the sequence in which process will request the resource and will release it known additionally, then we can decide for each request whether or not the process should wait.

c) Dead lock detection:

- If the system does not make use of deadlock prevention or deadlock detection algorithm, then system will observe the deadlock situation. In this situation, it is necessary that, system should have:

1) Algorithm which examines the state of the system to decide whether a deadlock has occurred or not.
2) Algorithm to recover from deadlock if it occurred
3) Both the algorithm requires the overhead including runtime cost of mainting necessary information and executing the detection algorithm and potential losses which are natural in recovering from deadlock.

d) Banker's algorithm:

- It is applicable to the resources allocation system with multiple instances of each resource type.
- It is less efficient than resource-allocation graph algorithm.
- Newly entered process should declare maximum number of instances of each resource type which it may require.
- the request should not be more than total number of resources in the system.

Q.12.) what is difference between deadlock avoidance and prevention.

Ans:

Deadlock avoidance	Deadlock Prevention
1. It blocks at least one of the conditions necessary for deadlock to occur.	1. It ensures that system does not get in unsafe state.
2. All the resources are requested together.	2. Resource requests are done according to the availability safe path.
3. Sometimes, preemption occurs more frequently.	3. In deadlock avoidance there is no preemption.
4. Deadlock prevention has low device utilization.	4. Deadlock avoidance, can block processes for too long.
5. E.g: Spooling and non-blocking synchronization algorithms are used.	5. ~~Bad~~ E.g Banker's and safety algorithm is used.