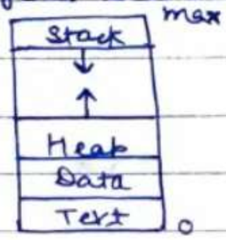
Unit - 2
Concurrent Processes

Process Model

A process is a sequential program in execution. A program defines the fundamental unit of computation of the computer. Process is an active entity.

when a program is doubled-clicked in windows then the program starts loading in memory and become a live entity. Process contains four sections:

a. ~~Data~~ Stack
b. Data
c. Heap
d. Stack.



1) Stack Section

This section contains local variable, function and return address. As stack and heap grow in opposite direction which is obvious if both grow in same direction they may overlap so it is good if they grow in opposite direction

2) Heap Section

This section is used to provide dynamic memory whenever memory is required by the program during run-time. It is provided from heap section.

3) Data Section

This section contains the global variables and static local variables.

4) Text Section

This section contain the executable instructions, constants and macros. it is read-only location and is sharable so that can be used by another process also.

Page No. ______
Date

Difference between Process and Program

Process	Program
1) Process is an operation which takes the given instruction and performs the manipulation as per the code called execution.	1) Program is a set of instructions that perform a designated task.
2) A process is entirely dependent on program.	2) Program are independent.
3) Process is a module that executes concurrently. They are separable and loadable modules.	3) Program performs a task directly relating to an operation.
4) A process includes program counter, a stack, a data section and a heap	4) A program is just a set of instructions stored on disk.
5) It is an active entity.	5) It is a passive entity.

Difference between Busy wait and Blocking wait

Busy Wait	Blocking Wait
1) Busy wait is a loop that reads that status register over and over ~~wait~~ until the busy bit becomes clear.	1) Blocking is a situation where processes wait indefinitely within a semaphores
2) It occurs when scheduling overhead is larger than excepted wait time.	2) It occurs when process resources are needed for another task.
3) It is schedule based.	3) In this, schedule based algorithm is inappropriate.

Page No. ______
Date

Principle of concurrency

Concurrency is the interleaving of processes in time to give the appearance of simultaneous execution. The fundamental problem in concurrency is processes interfering with each other while accessing a shared global resource.

```
chin = getchar();
Chout = chin
putchar(chout);
```

Concurrency can be implemented and is used a lot on single processing units, it may benefit from multiple processing units with respect to speed. If an operating system is called a multi-tasking operating system, this a synonym for supporting concurrency.

If we can load mutiple documents simultaneously in the tabs of our browser and we can still open ~~and~~ menus and perform more actions, this is concurrency.

Producer Consumer Problem

Producer process produce data item that consumer process consumes later. Buffer is used between producer and consumer. Buffer size may be fixed or variable. The producer portion of the application generates data and stores it a buffer and the consumer reads data from the buffer. The producer cannot deposit its data if the buffer is full. Similarly, a consumer cannot retrieve any data if the buffer is empty. If the buffer is not full, a producer can deposit its data. The consumer should be allowed to retrieve a data item if buffer contains.

Page No. ______
Date

```
Producer {
    while( True)
    /* produce an item and put in next Produced */
      while ( count == BUFFER_SIZE);    // do nothing
       buffer[in] = next Produced;
        in = (in+1) % BUFFER_SIZE;
        Count ++;
  }

Consumer {
    while (True)
     while ( count == 0)            // do nothing
      next Consumed = buffer [out];
        out = (out+1)% BUFFER_SIZE;
        count --;
     /* consume the item in nextConsumed */
}
```

Critical Section Problem

A critical section is a code segment where the shared variables can be accessed. An atomic action is required in a critical section i.e. only one process can execute in its critical section at a time. All the other processes have to wait to execute in their critical sections.

```
do {
        Entry Section
            (critical section)
        Exit Section
            (remainder Section)
   } while (TRUE);
```

The entry section handles the entry into the critical section. It acquires the resources needed for execution by the process. The exit section handles the exit from the critical section. It releases the resources and also informs the other processes that the critical section is free.

Solution to the Critical Section Problem

1. Mutual Exclusion:

It implies that only one process can be inside the critical section at any time. If any other processes require the critical section, they must wait until it is free.

2. Progress:

Progress means that if a process is not using the critical section, then it should not stop any other processes from accessing it. In other words, any process can enter a critical section if it is free.

3. Bounded waiting:

Bounded waiting means that each process must have a limited waiting time. It should not wait endlessly to access the critical section.

Mutual Exclusion

When a process is accessing shared variable is known as in critical section. When no two processes can be in critical section at the same time, this state is known as mutual Exclusion. It is property of concurrency control which is used to prevent race conditions.

Page No. ______
Date

Conditions for mutual Exclusion:

1) No two processes may at the same moment be inside their critical sections.
2) No assumptions are made about relative speed of processor or number of CPUs.
3) No process outside the critical section should block other processes.
4) No process should wait arbitrary long to enter its critical section.

Bakery Algorithm

Bakery algorithm is used in multiple process solution. It solves the problem of critical section for n processes. Before entering its critical section, process receives a ticket number. Holder of the smallest ticket number enters the critical section. The Baker algorithm cannot guarantee that two processes do not receive the same number. If processes P_i and P_j receives the same number, if $i<j$, then P_i is served first else P_j is served first.

```
do{
    choosing[i]=true;
    number[i] = max(number[0],number[1],...,number[n-1]) + 1;
    for (j=0; j<n; j++) {
        while (choosing[j]);
        while ((number[j]!= 0) && ((number[j],j) < (number[i],i)));
    }
    critical section
    number[i]=0;
    remainder section
} while (1);
```

Page No.
Date

Dekker's Algorithm

- It is the first known algorithm that solves the mutual exclusion problem in concurrent programming. Dekkers algorithm is used in process queuing and allows two different threads to share the same single-use resources without conflict by using shared memory for communication.

Dekker's algorithm will allow only a single process to use a resource if two processes are trying to use it at the same time. It succeeds in preventing the conflict by enforcing mutual exclusion, meaning that one process may use the resource at a time and will wait if another process is using it. This is achived with the use of two "flags" and a "token".

The flags indicate whether a process wants to enter the critical section or not, a value of 1 means TRUE that the process wants to enter the CS, while 0 or FALSE mean the opposite. The token, which can have a value of 1 or 0, indicates priority when both processes have their flags set to TRUE.

Busy waiting

```
do
{
        wait(mutex);
                // critical section
        signal(mutex);
                // remainder section
}
while(TRUE);
```

Busy waiting wastes CPU cycles that some other process might be able to use productivity.

Peterson's solution

Peterson's algorithm is used for mutual exclusion and allows two processes to share a single-use resource without conflict. It uses only shared memory for communication. Peterson's formula orginally worked only with two processes, but has since been generalized for more than two.

Assume that the LOAD and STORE instructions are atomic i.e. cannot be interrupted.

Two processes share two variables:

→ int turn;

→ Boolean flag[2]

The variable turn indicates whose turn it is to enter the critical section. The flag array is used to indicate if a process is ready to enter the critical section.

flag[i] = true implies that process Pi is ready.

Algorithm:

```
do {
    flag[i] = TRUE;
    turn= j;
    while (flag[j] && turn==j);
            critical section
    flag[i] = FALSE;
            remainder section
} while (1)
```

Two processes executing concurrently:

```
do {
 flag 1 = TRUE;
 turn=2;
 while (flag2 && turn==2)
 critical section
 flag 1 = FALSE;
 remainder section
} while(1)
```

```
do {
 flag 2 =TRUE;
 turn = 1;
 while (flag 1 && turn==1)
 critical section
 flag 2 = FALSE;
 remainder section
} while (1)
```

Page No. ____
Date

A system is said to be concurrent if it can support two or more actions in progress at the same time.
Concurrency is the property of the program.

A system is said to be parallel if it can support two or more actions executing simultaneously.
Parallel execution is the property of the machine.

Semaphores

Semaphores is simply a variable. This variable is used to solve the critical section problem and to achieve process synchronization in the multiprocessing environment.
The two most common kinds of semaphores are counting semaphores and binary semaphores. Counting semaphores can take non-negative integer values and binary semaphore can take the value 0 & 1 only.

Semaphores are integer variables that are used to solve the critical section problem by using two atomic operations, wait and signal that are used for process synchronization.

1. Wait

The wait operation decrements the value of its argument S, if it is positive. If S is negative or zero then no operation is performed.

```
wait(S)
{
    while (S<=0);
    S--;
}
```

2. Signal

The signal operation increments the value of its argument.

```
signal(S)
{
S++;
```

Page No. ______
Date

Advantages of Semaphores

1) Semaphores allow only one process into the critical section. They follow the mutual exclusion strictly.
2) There is no resource wastage because of busy waiting in semaphores.
3) semaphores are implemented in microkernel. So they are machine independent.

Disadvantages of Semaphores:

1) Semaphores are complicated so the wait and signal operations must be implemented in the correct order to prevent deadlocks.
2) Semaphores may lead to priority inversion where low priority processes may access the critical section first and high priority processes later.

Reader - Writer Problem

The reader-writer problem relates to an object such as a file that is shared between multiple processes. Some of these processes are readers i.e they only want to read data from object and some of the processes are writers. i.e. they want to write into the object.

The reader-writer problem is used to manage synchronization so that no problems with the object data. for ex- If two readers access the object at the same time, there is no problem. However if two writers or a reader and writer may access object at the same time, there may be problems.

To solve this problem, a writer should get exclusive access to an object i.e when a writer is accessing the object, no reader or writer may access it. However multiple readers can access the object at the same time.

Page No. ______
Date

Reader Process:

```
wait(mutex);
 rc++;
 if (rc==1)
    wait(wrt);
 signal(mutex);
  :
  : //Read
  :
 wait (mutex)
  rc--;
  if (rc==0)
  signal(wrt);
  signal(mutex);
```

writer process

```
wait (wrt);
  :
  : // write
  :
 signal (wrt);
```

Classical Problem in Concurrency

Concurrency is the interleaving of processes in time to give the appearance of simultaneous execution. Thus it differs from parallelism, which offers genuine simultaneous execution. However the issues and difficulties raised by the two overlap to a large extent:

- Sharing global resources safely is difficult
- optimal allocation of resources is difficult.
- locating programming errors can be difficult.

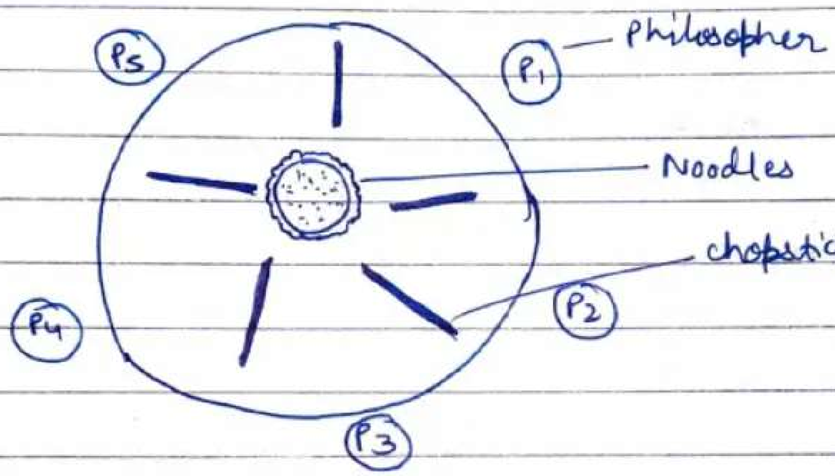
Ex:

```
chin = getchar();
chout = chin;
putchar (chout);
```

Page No.
Date

The Dining philosopher problem:

It states that K philosophers seated around a circular table with one chopstick between each pair of philosophers. There is one chopstick between each philosopher. A philosopher may eat if he can can pickup the two chopsticks adjacent to him. One chopstick chopstick may be picked by any one of its adjacent followers but not both.



Semaphore solution to Dining Philosopher:

```
Process P[i]
while true do
{   THINK;
    PICKUP(CHOPSTICK[i], CHOPSTICK[i+1 mod 5]);
    EAT;
    PUTDOWN(CHOPSTICK[i], CHOPSTICK[i+1 mod 5]);
}
```

There are three states of philosopher: THINKING, HUNGRY and EATING. Here, there are two semaphores: mutex and a semaphore array for the philosophers. Mutex is used such that no two philosophers may access the pickup or putdown at the same time. The array is used to control the behaviour of each philosopher.

Page No. ____
Date

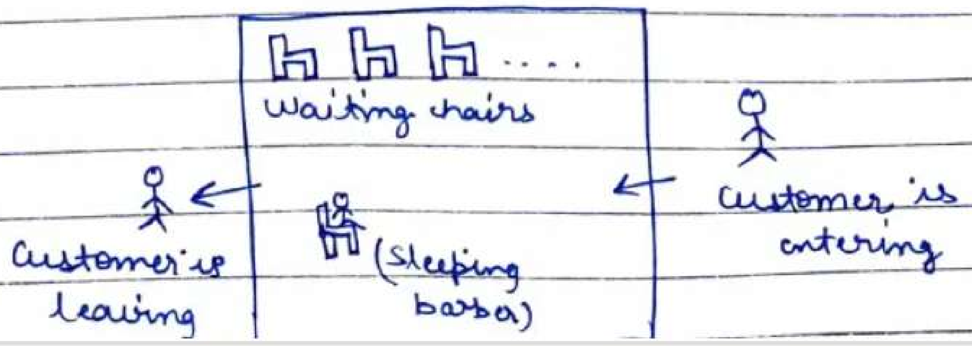
Sleeping Barber Problem:

The analogy is based upon a hypothetical barber shop with one barber. 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.

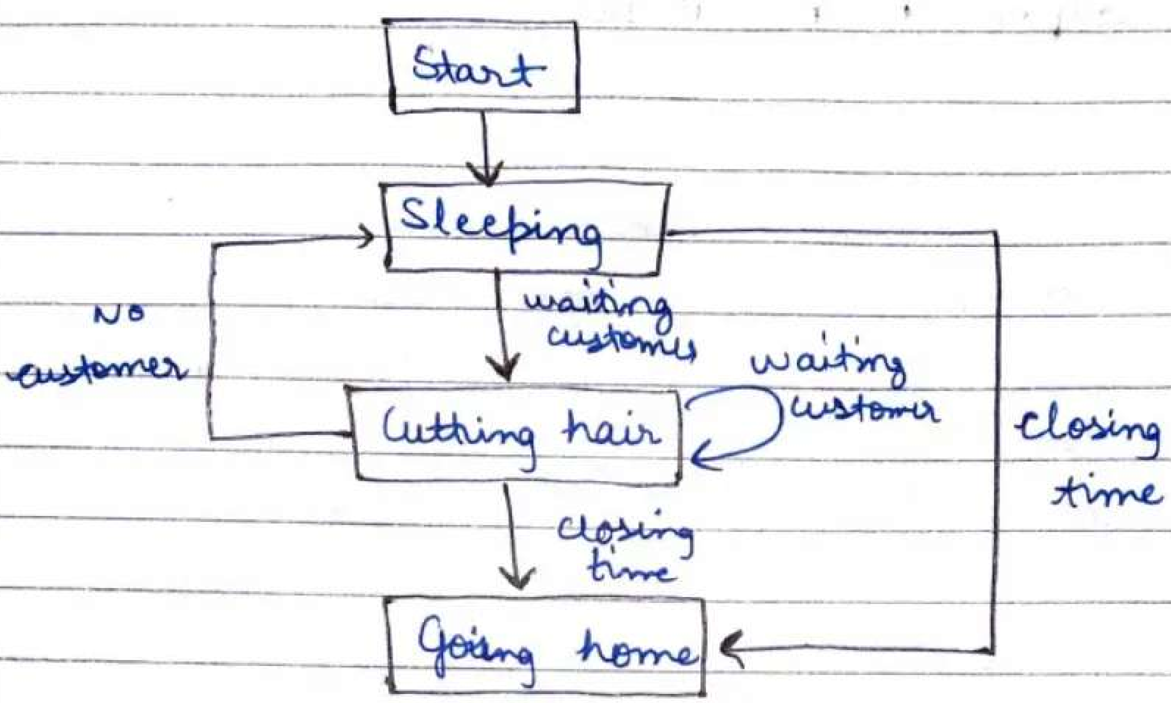
- If there is no customer, then the barber sleeps in his own chair.
- When a customer arrives, he has to wake up the barber.
- If there are many customers and the barber is cutting a customer's hair, then the remaining customers either wait if there are empty chairs in the waiting room or they leave if no chairs are empty.

Solution: The solution to this problem includes three semaphores. first is for the customer which counts the number of customers present in the waiting room. Second for barber 0 or 1 is used to tell whether the barber is idle or is working. and third mutex is used to provide the mutual exclusion which is required for the process to execute.

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 customer leaves the barbershop.



Date


Start
Sleeping
waiting customer
No customer
Cutting hair
waiting customer
closing time
closing time
Going home