UNIT - IV

~~ABSTRACT DATA TYPES~~ - CONCURRENCY -

① Concurrency ~~can~~ in software execution can occur at four different levels-
(1) Instruction level → Executing two or more machine instructions simultaneously
(2) Statement level → Executing two or more high level language statements simultaneously
(3) Unit level → Executing two or more sub program unit simultaneously
(4) Program level → Executing two or more program simultaneously

→ A concurrent algorithm is scalable if the speed of its execution increases when more processors are available.

→ Two distinct categories of concurrent unit control are-

(1) Physical Concurrency -
Assuming that more than one processor is available, several program units from the same program literally execute simultaneously.

(2) Logical Concurrency -
Assuming that there are multiple processors providing actual concurrency, when in fact the actual execution of programs is taking place in interleaved fashion on a single processor.

→ Programs that have coroutines but no concurrent sub-programs, though they are sometimes called quasi-concurrent.

→ A thread of control in a program is the sequence of program points reached as control flows through the program.

→ A program designed to have more than one thread of control is said to be multithreaded.

→ Use of concurrency are -
(1) Speed of execution of programs on machines with multiple processors.
(2) Program written to use concurrent execution can be faster than the same program written for sequential execution.
(3) Diffrent method for conceptualizing program solutions to problems.
(4) Program applications are distributed over several machines.

② Subprogram level concurrency -

→ Fundamental concepts -

(1) Task - Unit of a program, Also called process.

(2) Threads - Methods that are executed in objects.

(3) Characteristics of Task -

(i) Implicitly started

(ii) It need not wait for the task to complete its execution before continuing its own.

(iii) When execution of task is completed, control may or may not return to the unit that started that execution.

(4) Categories of Tasks -

(i) Heavyweight Tasks - executes in its own address space.

(ii) Lightweight Tasks - all run in the same address space.

(5) Synchronization -

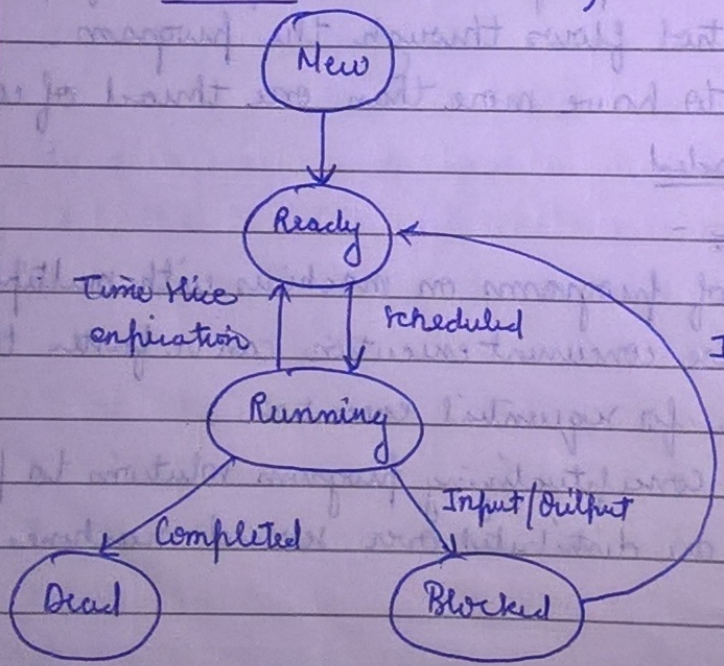
It is a mechanism that controls the order in which tasks execute.

Also it is Design issue ←

(i) Cooperation Synchronization → cooperation between Tasks

(ii) Competition Synchronization → Tasks, whoever gets the first chance

(6) Task states - (5 states)



→ Concurrency is implemented through libraries.

③ Semaphores –

It is a simple mechanism that can be used to provide synchronization of tasks. Specifically, a semaphore is a data structure that consists of an integer and a queue that stores task descriptors.

→ Task Descriptor – It is a data structure that stores all the of the relevant information about the execution state of a task.

→ Guard – It is a linguistic device that allows the guarded code to be executed only when a specified condition is true.

→ Cooperation Synchronization –

(1) FETCH subprogram – fetch the item contain the buffer.

(2) DEPOSIT subprogram – deposite the item to the buffer.

(3) WAIT subprogram – Used to test the counter of a given semaphore variable

(4) RELEASE subprogram – Release the counter for another task.

→ Competition Synchronization –

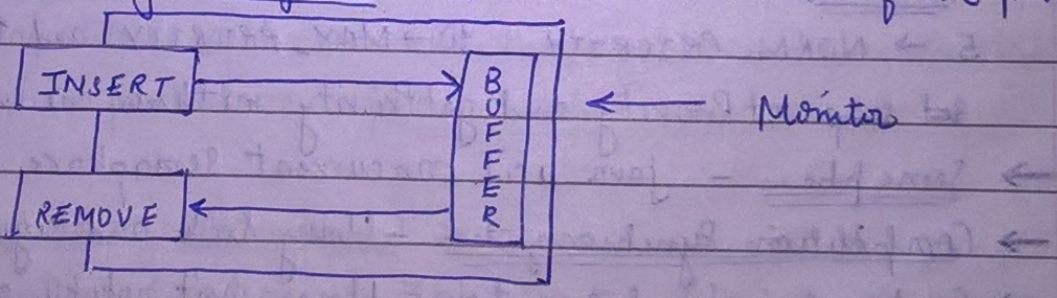
Binary semaphore – Requires only a binary-valued counter.

④ Monitors –

All synchonization operations on shared data be gathered into a single program unit called monitors.

→ Competition Synchronization – Synchonized access by allowing only one access at a time.

→ Cooperation Synchonization – It is still the task of the programmer.



⑤ Message Passing - It can be either synchronous or asynchronous
→ A rendezvous can occur only if both the sender and receiver want it to happen.
→ Both cooperation and competition synchronization of tasks can be conveniently handled.

⑥ Java Threads -
→ Concurrent units in Java are methods named run. The process in which the run methods execute is called a thread.
→ Java's threads are lightweight tasks.
→ The Thread Class -

```
class MyThread extends Thread {
    public void run() {...}
}
...
Thread myTh = new MyThread();
myTh.start();
```

→ Several methods for controlling the execution of threads are -
(1) yield method → takes no parameters
(2) sleep method → has a single parameter
myTh.join(); (3) join method → Used to force a method to delay its execution.
→ Three more thread methods are stop, suspend and resume.
→ Priorities -
5 → NORM_PRIORITY, 10 → MAX_PRIORITY and 1 → MIN_PRIORITY
setPriority and getPriority methods are used.
→ Semaphores - java.util.concurrent.Semaphore package.
→ Competition Synchronization - Using lock and keyword synchronized
→ Cooperation Synchronization - Using wait, notify and notifyAll methods
→ Nonblocking Synchronization - No waiting, java.util.concurrent.atomic pkg
→ Explicit locks - Using lock, unlock and trylock methods

⑦ C# Threads - (lightweight tasks)

Using predefined delegate, ThreadStart.

Eg: public void MyRun1() {...}

...

Thread myThread = new Thread(new ThreadStart(MyRun1));
myThread.Start();

→ A thread can be terminated using Abort method.

→ myThread.BeginInvoke(10, null, null); // asynchronously.

→ Synchronizing Threads -

Three different ways that C# threads can be synchronized - Interlocked class (for the integer), Monitor class (System.Threading) (five methods - Enter, wait, Pulse, PulseAll & Exit) & lock statement (for the critical section)

*****

→ Any method can be run in its own thread.

TYPES OF STORAGE MANAGEMENT -

① Static Storage Management -

Simplest form of allocation. Ordinarily storage for the code segments of user and system programs is allocated statically.

Advantages -

(1) Requires no run-time storage management.

(2) Efficient because no time or space is expended for storage management during execution.

Disadvantage -

Incompatible with recursive subprogram calls, with data structures whose size is dependent on computed or input data.

② Stack based Storage Management -

Sequential locations in the stack. A single stack pointer is needed to control storage management using stack. Stack pointer always points to free storage block. ←

③ Heap storage management - (Two techniques → fixed size & variable size)

A heap is a block of storage within which pieces are allocated and freed in some relatively unstructured manner.

Advantages -

(E) language permits storage to be allocated and freed at arbitrary points during program execution.

Disadvantage -

The problems of storage allocation, recovery, compaction, and reuse may be severe.

④ Garbage Collection -

It is a form of automatic memory management. This frees the programmer from needing to delete objects explicitly when they are no longer needed.

It is used for storage reclamation.

ABSTRACT DATA TYPES -

① An abstraction is a view or representation of an entity that includes only the most significant attributes.

An abstract data type is an enclosure that includes only the data representation of one specific data type and the subprograms that provide the operations for that ~~data~~ type.

An instance of an abstract data type is called an object.

② Floating-Point as an abstract data type

③ User-defined abstract datatypes -

Characteristics — type definition and have a set of operations

④ Encapsulation - a single unit containing collection of data and functions.

→ Object oriented programming in small talk, C++, Java, C#, PHP, Perl.