# UNIT - 4

## Logic Programming.

Logic Programming is a programming paradigm in which the set of sentences are written in logical form. The logical Programs consist of fact and rule.

The logical programming can be define into form of Logical component and control component

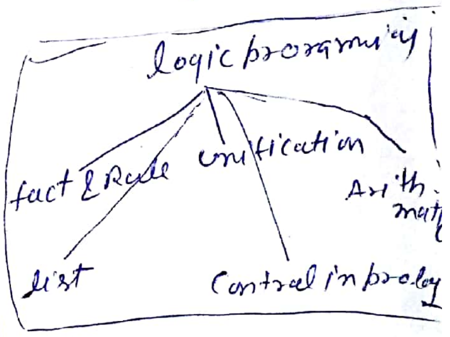
Logical programming = Logic + Control.

where Logic Represents the logic program in form of fact and rule. and control represents how algorithm can be implemented by applying the rules in particular order.

the Concept of Logic Programming is linked up with a language called prolog: prolog name is generated by PROgramming ~~and~~ in LOGic.

SWI - Prolog

```
1 ?- 10>5.
  true
2 ? - 2+3>11.
  false
3 ? write("Hello").
  Hellow
  true.
```



Facts and Rules- A Prolog program consist of a number of clauses. each clause is either a fact or a rule. Prolog Interpreter ~~[illegible]~~ will give result according to the fact and rules.

Fact A fact must start with a predicate (atom) and end with a full stop. predicate may be followed by one or more argument which are enclosed by parentheses. argument can be constant, number, variables or list. Argument are seperated by commas.

Syntax of Fact:

peed(arg1, arg2 ... argN).

Name of predicate — argument list.

eg.

test1.pl

```
man(anand).
man(arun).
woman(anuradha).
parent(anand, parth).
parent(anuradha, parth).
parent(arun, anuradha).
```

→ open prolog and select consult, select the file name test1.pl.

SWI-Prolog.

```
1 ?- parent(anand, parth).
true.

2 ?- parent(arun, anuradha).
true

3 ? parent(arun, anand).
false.
```

Rule: A rule can be viewed as an extension of a fact with added condition that also have to be satisfied for it to be true.

Syntax

head :- body.

where head is a predicate difinition just like (fact)

:- is the neck symbol ~~for~~ read as "if"

Body is one or more goal (Query)

example

test1.pl.

```
man(anand).
man(arun).
woman(anuradha).
woman(Jayshree).
parent(anand, parth).
parent(anuradha, parth).
parent(arun, anuradha).
parent(Jayshree, anuradha).            } fact
father(F,C) :- man(F), Parent(F,C).
mother(M,C) :- woman(M), Parent(M,C).  } Rule.
```

SWI-Prolog.

```
1 ?- father(X, Parth).
X = anand.

2 ?- mother(X, Parth).
X = anuradha.
```

Unification   Unification is a derivation of new rule from given by binding of variables. During Unification, any variable encountered is substituted with the value of an appropriate constant. this process is called binding

e.g

?- f(A,1) = f(2,B).

A = 2,
B = 1.

?- 1*2+3 = X+Y

X = 1*2
Y = 3

Arithmatic.

The operator = is used for unification in prolog. the use of '=' for binding with the corresponding value.

e.g.

?A = 10+20.
A = 10+20.

?- A is 10+30.
A = 40

? B is 6*2
B = 12

? X is mod(7,2)
X = 1

Lists in Prolog.   A list is an ordered sequence of object and list. in prolog list is written as its element separated by commas and enclosed in brackets eg. [a, b, c, d]. the list contain head and tail part the first part of list called head and rest of the list called tail.

eg. ?- [a, b, c, d] = [A/B].

A = a.
B = [b, c, d].

Control in Prolog: Control represent how a language computes a response to a query. the control execution is based on two rules.

1. Goal Order → that means chose left most subgoal.
2. Rule Order → that mans select the first Applicable rule.

family.pl.

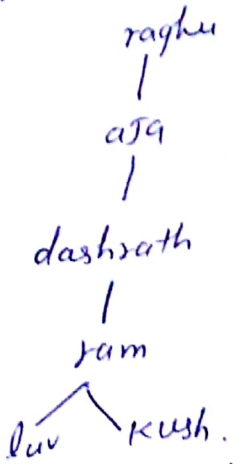
left most subgoal

e.g.
rule order

```
/* Rules */
1. gfather(X, Y) :- father(X,Z), parent(Z,Y).
2. parent(X,Y) :- father(X, Y).

/*facts*/
father(dashrath, ram)
father(raghu, aja).
father(aja, dashrath).
father(ram, luv).
father(ram, kush).
```

Tree Representation of fact.



execution of family.pl.

?- gfather(aja, ram).
true.

?- ~~parent(X, luv)~~

? parent(X, luv)
X = ram.

~~for~~ gfather(aja, ram) the first rule is
gfather(X,Y): father(X,Z), Parent(Z,Y)
gfather(X,Y) : father(aja,Z), parent(Z,Y)
parent(dasrath, ram) :- father(dasrath, ram).

left most subgoal chosen and search fact for that so Z = dasharath

## Concurrent Programming

The term concurrency can be defined as the expression of the task in term of multiple interaction sub-task that can be potentially executed at the same time.

The intraction among processes take two forms.

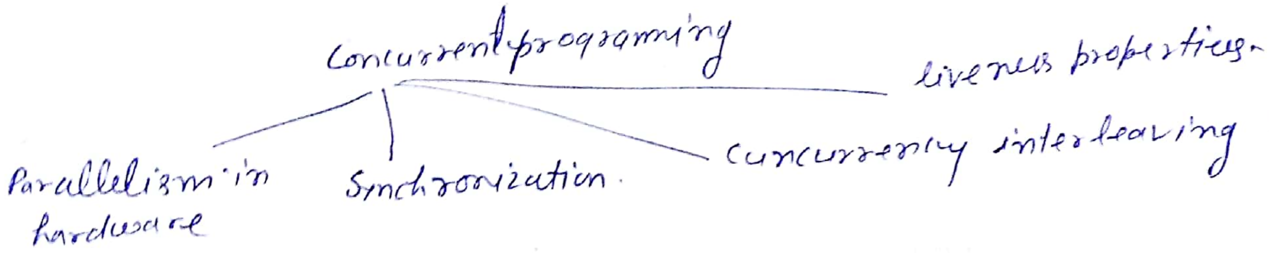
1. Communication : it involves exchang of data betn processes either through messages, or shared variable.

2. Synchronization : It involves Synchronization of one task with another generally thread execution is involved in Synchronization.

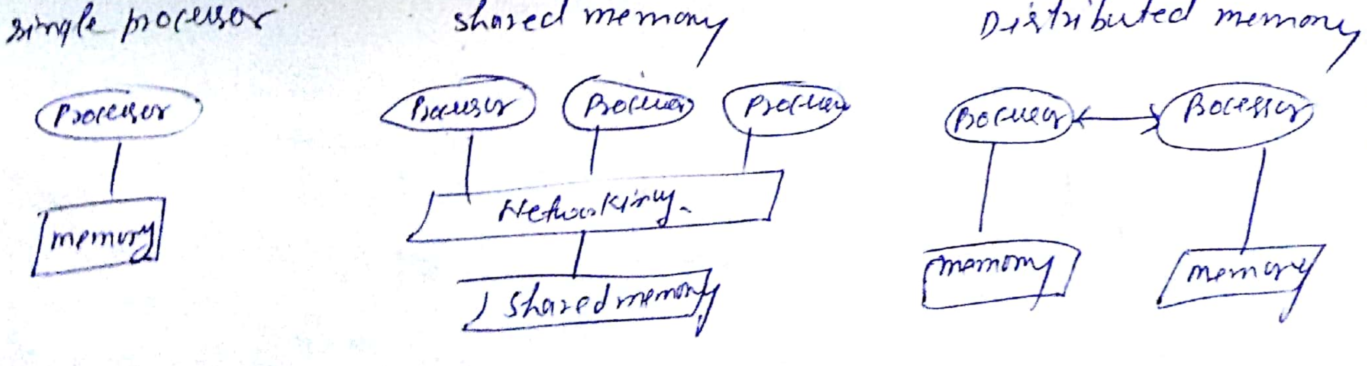
Process and thread : Process is basically an unit of work created by OS. Process has its own address space, A program can be broken down into one or more processes.

Thread is a lightweight process which makes use of the address of its parent process for execution.

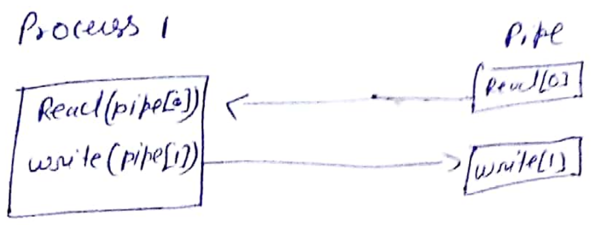
Basically Process creates one or more threads to accomplished its task.

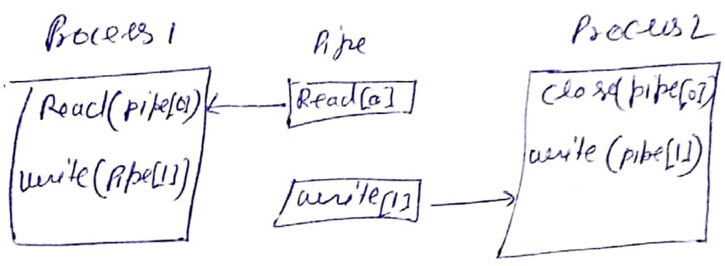


Parallelism in Hardware:



Synchronization: Parallelism raised the problem of synchronization access to shared resource. in multiprocessor system shared memory or in distributed system the Communication betⁿ the processes is using input and output.





Concurrency Interleaving:

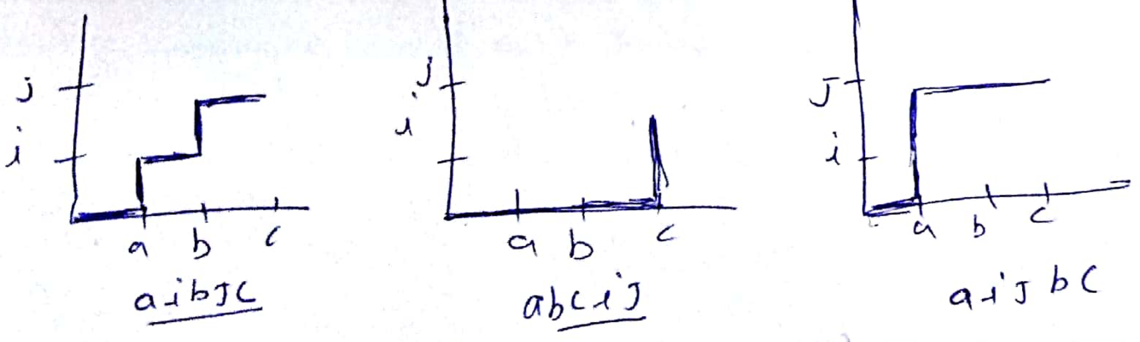
Interleaving of threads is a convenient technical device for studying the concurrent execution of processes. Based on the event in the threads the interleaving can be possible.

e.g. P1 consist three process a, b and c

P2 consist two process i and J.

the concurrent execution of P1 and P2 will generate following interleaving. as

abciJ, aibJc, .... iJaibc.



Liveness properties ←——→ liveness represent the rate of progress of the process during which the computations proceed. e.g. deadlock, livelock, fairness.

Example: Dining philosopher problem:

Solution to this problem:

- there will Never be a situation of deadlock.
- there will never be a situation of resource starvation.
  that is a situation in which philosopher wants to eat but does it.

Deadlock: is a situation in which one process depends upon the resource held by another process.

livelock: it is a kind of situation in which the execution is continued without any progress

Fairness: it is a kind of situation in which any process that wants to execute can execute in a finite amount of time.