INTRODUCTION TO SYSTEM PROGRAMMING

• WHAT IS SYSTEM PROGRAMMING?

System programming is the activity of programming computer systems software. It is collection of different system programs.

Conceptually, computer instruction or data, anything that can be stored electronically in a system is a system software which helps you to provide better user interaction. It facilitated user to use the hardware system with its maximum efficiency.

TYPES OF SOFTWARE:

1] Open Source Software
2] Application Software
3] System Software

In system software operating system helps all utilities to enable by providing full functionality to use computer with its maximum capacity. This manages all the resources and controls the hardware so that application software can perform the task

SYSTEM SOFTWARE

System software is a collection of one or more programs used to control and coordinate the hardware and other application software.

Communicated with the hardware deviced. Controls and monitors the proper use of various hardware resources like CPU, memory, peripheral devices like monitor, printer etc

Supports the execution and development of other application software.

Few examples of system software are:

Operating System
Compliers and Interpreters
Loaders and Linkers

APPLICATION SOFTWARE

Application software is a collection of one or more programs used to solve a specific task. Generally software used in banking industry, airline/railway reservation, generation of telephone or electricity bills etc. all fall under application software.

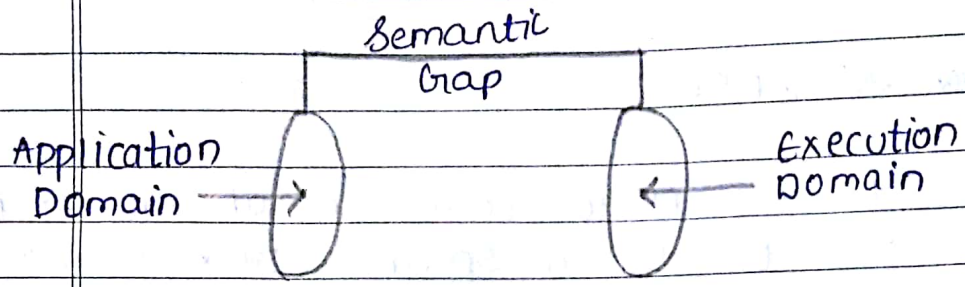
Few examples of application software are:

Word processing software
Education software
Spreadsheet software

NEED OF LANGUAGE PROCESSING

Language Processing activities arise due to difference between the manner in which a software designer describes the ideas concerning the behaviour of a software and a manner in which the ideas are implemented in a computer system

The software designer expresses the ideas in terms related to "application ~~software~~ domain" of the software, but to implement these ideas their description need to be interpreted in terms related to the "execution domain" of the computer.

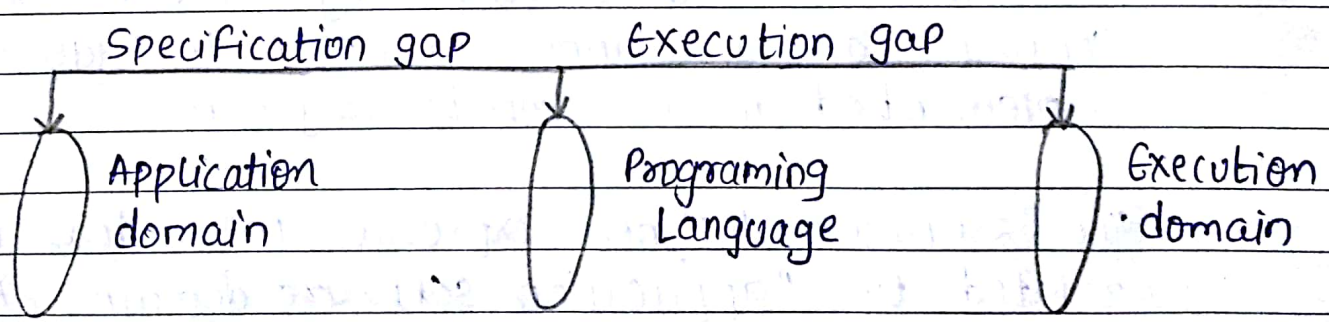


The gap between application domain & Execution domain is called semantic gap

Consequences of Semantic Gap

i) Large development time
ii) Large development efforts
iii) Poor software qualities

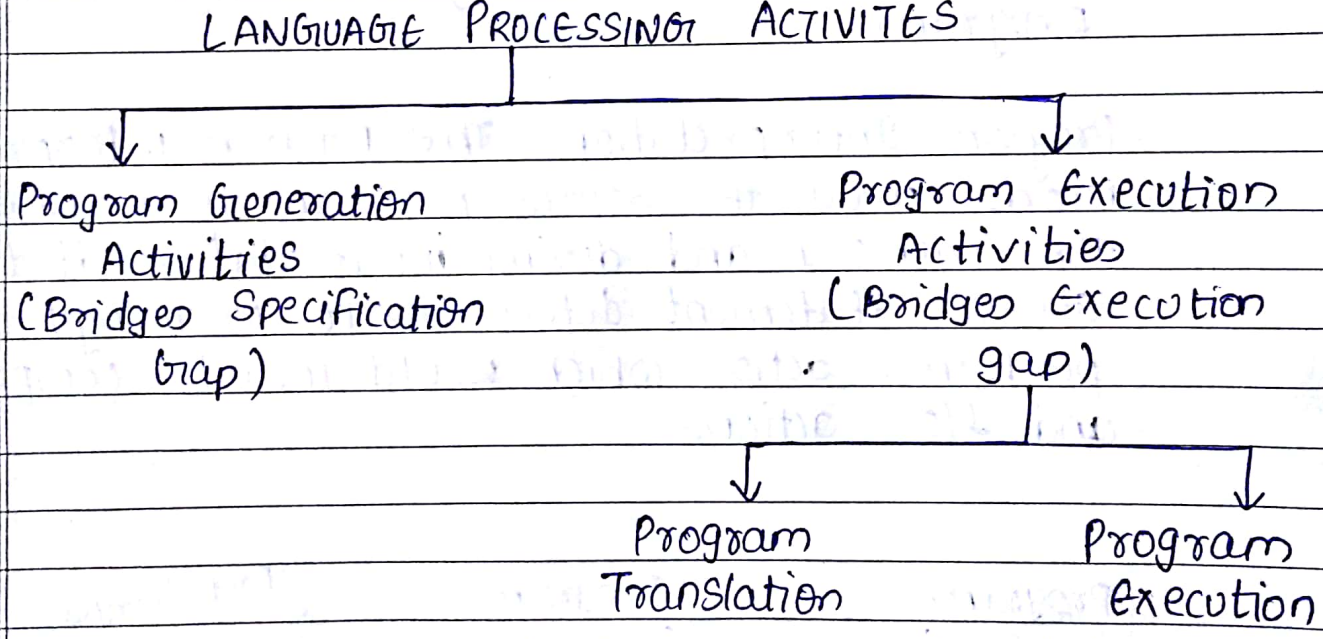
To manage this semantic gap processing language domain was introduced, it is nothing but implementation using programming language



LANGUAGE PROCESSOR

Language processor is a software which bridges the specification or execution gap. The program translated by language processor is understood by the hardware of the computer

Examples:
Compliers, Assemblers and Interpreters



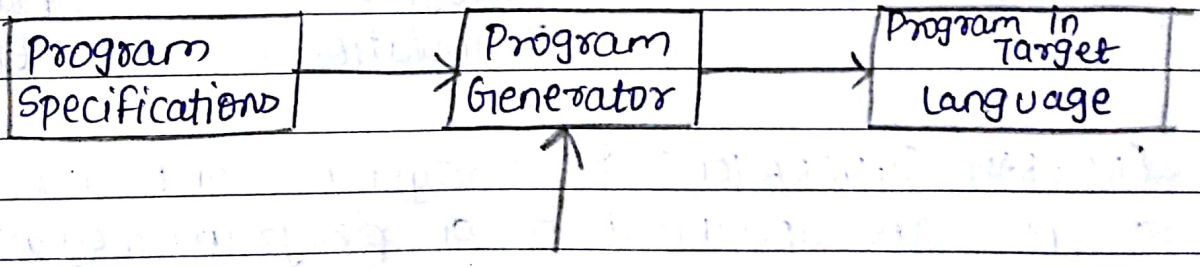
PROGRAM GENERATION: Is a system software which accepts the specification of program regenrated and throws out the errors by generating the program in the target language.

Program Execution is divided into two parts:

1] Program Translation 2] Program Interpretation

Program Translation: Here the model bridges the execution gap by translating a program written in a programming language called the source program into an equivalent program in a machine language of a computer system called as target program

Program Interpretation: The Program interpretation model reads the source program and stored it in the memory and during interpretation it takes the source statement determines it meanings and performs action which would include computational and I/O actions.

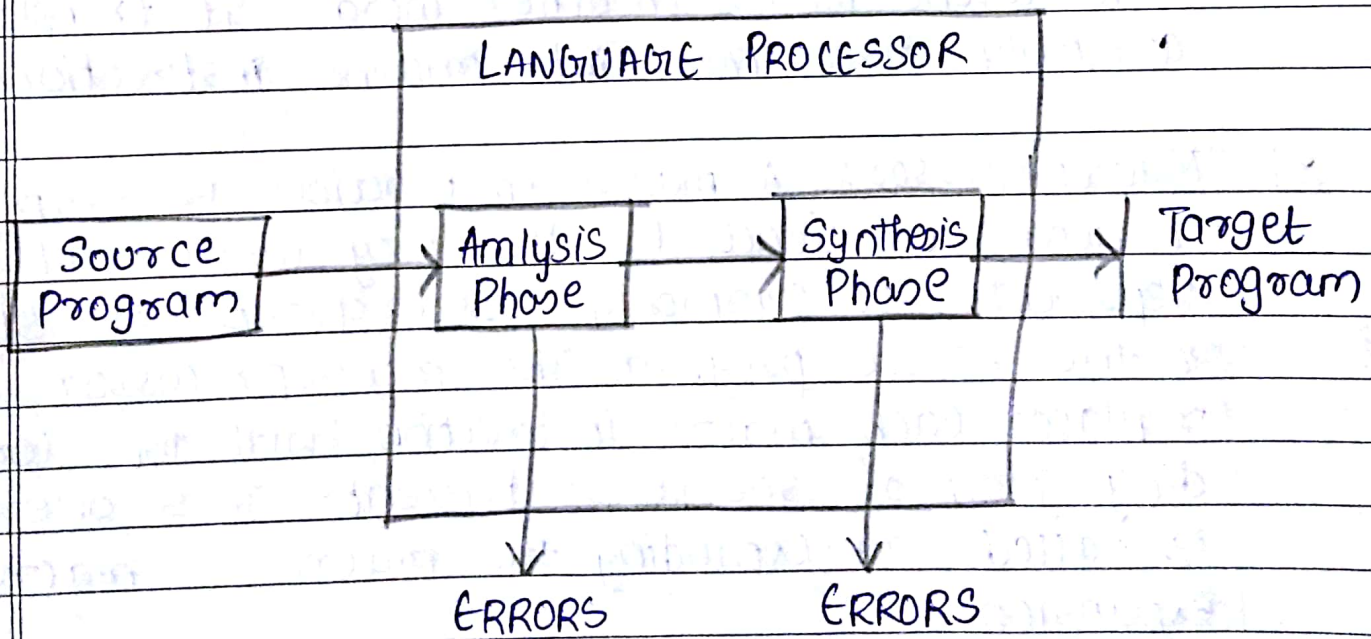


Language Processing

Language Processing is a process of analysis of the source program and synthesis of target Program

Language Processing = Analysis of Source Program + Synthesis of Target Program

Collection of language processing components enganged in analysing a source program is called analysis phase while the components enganged in synthesing a target program constitute the synthesis phase.



VARIOUS SYSTEM PROGRAMS

1] Operating System: It acts as an interface between the application program and the machine hardware. It enables us to get better user interface with the available hardware system. It enables all the application program to run smoothly to get the task done.

Eg :- Windows, MacOS

2] Assemblers: It is a translator which translates source instructions (in symbolic language) into target instructions (in machine language), on a one-to-one basis. In other words it translates assembly instructions into machine instruction.

3] Macroprocessor: A macro instruction is simply a national convience for the programmer. It represents a commonly used group of statements in the source program. The macroprocessor replaced each macro instruction with the corresponding group of source statements. This operation is called as expanding the macro or macro expansion.

4] Compliers: It is a computer program that transform human readable source code of another computer program into machine readable code which CPU can execute. The act of transforming source code into machine code is called compliation.

5] LOADERS: Is a system program which performs allocation, linking, relocation & loading the contents which are required for execution of main program. It is responsible for loading programs from executable files into the memory preparing them for execution and then executing them as per the requirements.

6] LINKERS: This takes 1/more object generated by a complier and combines them into a single executable program

7] INTERPRETER: An interpreter normally means a computer program that executes the instruction written in programming language. This maybe a program that executes the source code directly, translates source code into some efficient intermidate representation code & immediately executes it or explicitly executes stored pre complied code made by a complier which is part of interpreter system.

	STATIC BINDING	DYNAMIC BINDING
1]	It is binding that happens at compile time	1] It is binding that happens at run time
2]	Actual object is not used for binding	2] Actual object is used for binding
3]	It is also called as an early binding because it is done in compilation	3] It is called as late binding since it is done at run time
4]	Method overloading is best eg of static	4] Method overriding is best eg
5]	It is easier to debug	5] It is harder to debug
6]	Private, static, final methods shows static binding because they cannot be overriden	6] Other than private, static and final methods shows dynamic since they can be overriden