Symbol-Table Management: An essential function of a compiler is to record the identifiers used in the source program and collect information about various attributes of each identifier .These attributes may provide information about the storage allocated for an identifier , its type and in case of procedure , the number and types of its arguments and so on .

Symbol-Table is a data structure containing a record for each identifier , with fields for the attributes of the identifier.

Error Detection and Reporting

Each phase can encounter errors. However ,after defection an error a phase must somehow deal with that error, so the compilation can proceed, allowing further errors in the source program to be detected .A compiler that stops where it finds the first error is not as helpful as it could be. The syntax and semantic analysis phases usually handle a large fraction of the error detectable by the compiler .

- **Types of Errors**

   Lexical errors: The lexical phase can detect errors where the characters remaining in the input do not form any token of the language.

   Syntax errors: The syntax analysis phase can detect errors Errors where the token stream violates the structure rules (syntax) of the language .

   Semantic errors: During semantic analysis the compiler tries to detect constructs that have the right syntactic structure but no meaning to the operation involved, e.g. to add two identifiers, one of which is the name of an array, and the other the name of a procedure .

Compile-time errors

Many errors are detected by the compiler, the compiler will generate an error message - Most compiler errors have a file name , line number, and Type of error. This tells you where the error was detected .

File name (fname.java)

```
C:\code javac fname.java
fname.java:23: cannot resolve symbol
symbol   : class string
location: class fname
     string msg;
        ^
1 error
```

Line number (23)

```
C:\code javac fname.java
fname.java:23: cannot resolve symbol
symbol   : class string
location: class fname
     string msg;
        ^
```

Type of error

```
C:\code javac fname.java
fname.java:23: cannot resolve symbol
symbol   : class string
location: class fname
     string msg;
        ^
1 error
```

Runtime Errors

Runtime errors occur while the program is running, although the compilation is successful. The causes of Runtime Errors ar[5]:

1) Errors that only become apparent during the course of execution of the program

2) External Factors - e.g.

   Out of memory

Hard disk full

Insufficient i/o privileges etc.

3) Internal Factors - e.g.

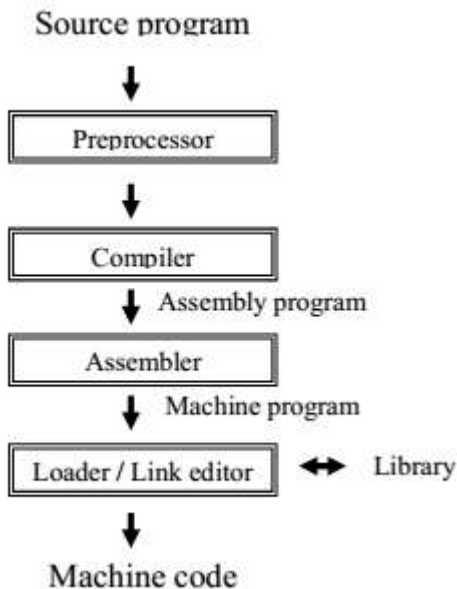Arithmetic errors

Attempts to read beyond the end of a file

Attempt to open a non-existent file

Attempts to read beyond the end of an array

etc.

A Language- Processing System :-

In addition to a compiler ,several other programs may be required to create an executable target program.

```
          Source program
                 ↓
        ┌──────────────────┐
        │   Preprocessor   │
        └──────────────────┘
                 ↓
        ┌──────────────────┐
        │     Compiler     │
        └──────────────────┘
                 ↓ Assembly program
        ┌──────────────────┐
        │     Assembler    │
        └──────────────────┘
                 ↓ Machine program
        ┌──────────────────┐
        │ Loader / Link editor │  ↔  Library
        └──────────────────┘
                 ↓
          Machine code
```

(Language- Processing System)

Preprocessing : During this stage , *comments* ,*macros* and *directives* are processed :

- *Comments* are removed from the source file.
- *Macros* :If the language supports macros ,the macros are replaced with the equivalent text,Example:

define pi 3.14

When the preprocessor encounter the word(pi) it would replace (pi) with (3.14)

- *Directives* : The preprocessor also handles directives. In 'C' language , including statement looks like:
include<"file">
this line is replaced by the actual file.

Loader - Link Editor : Is program that performs two functions:
1. **Loading** :taking relocatable machine code and placed the altered instructions and data in memory at the proper locations.
2. **Link-Editing** :Allows us to make a single program from several files . these files may have been result of different compilers and one or more may be library files.