Software Development Environment

- Operating System
- Programming Languages,
- Compilers, Editors, Linkers,
- Code-generators,
- Program Design Languages,
- Debuggers
- Workbenches, Integrated CASE tools
- Ideal Software Development Plan.

Operating System

Operating System is the system software which acts as an interface b/w the user of computer and the computer hardware. Operating system is the first software that we see when we turn on the computer and the last software we see when we shut down the system.

Uses

- Resource Manager.

Types of Operating System

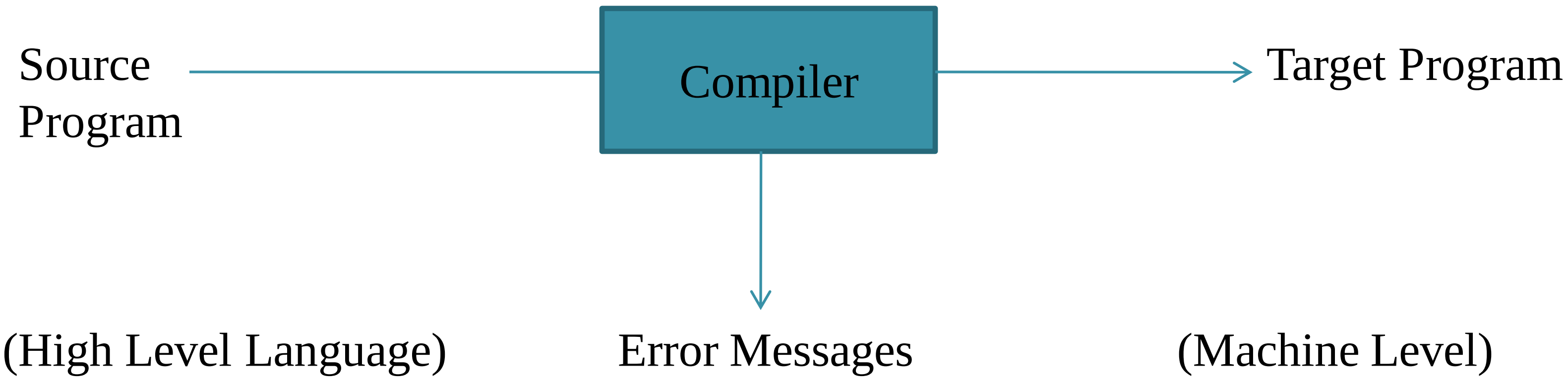
- Single User Operating System.
- Multi User Operating System.
- Real Time Operating System.

Types of Programming Languages

- Procedural Languages.
- Non Procedural Languages.
- Imperative Languages.
- Declarative Languages.
- Functional Languages.
- Logic Languages.
- Visual Languages.
- Fourth Generation Languages.

Compiler

A compiler is a software that translate program written in high level language into machine level language. Compiler translate each high level language instruction into a set of machine language instruction.



Editors

Editors are useful for coding , data entry, documentation. A number of CASE tools are also available to facilitate the document-writing process. The presence of an easy to use editor with useful features is best appreciated by the programming team , data entry operators and clerical assistants.

Syntax –Directed Editors:

Some software development environments have special editors that have the capability to verify each program line entered to assure that it is syntactically correct, without delaying entry. This added value facility may reduce the time neede to correct compiling errors; and speed up the compiling process.

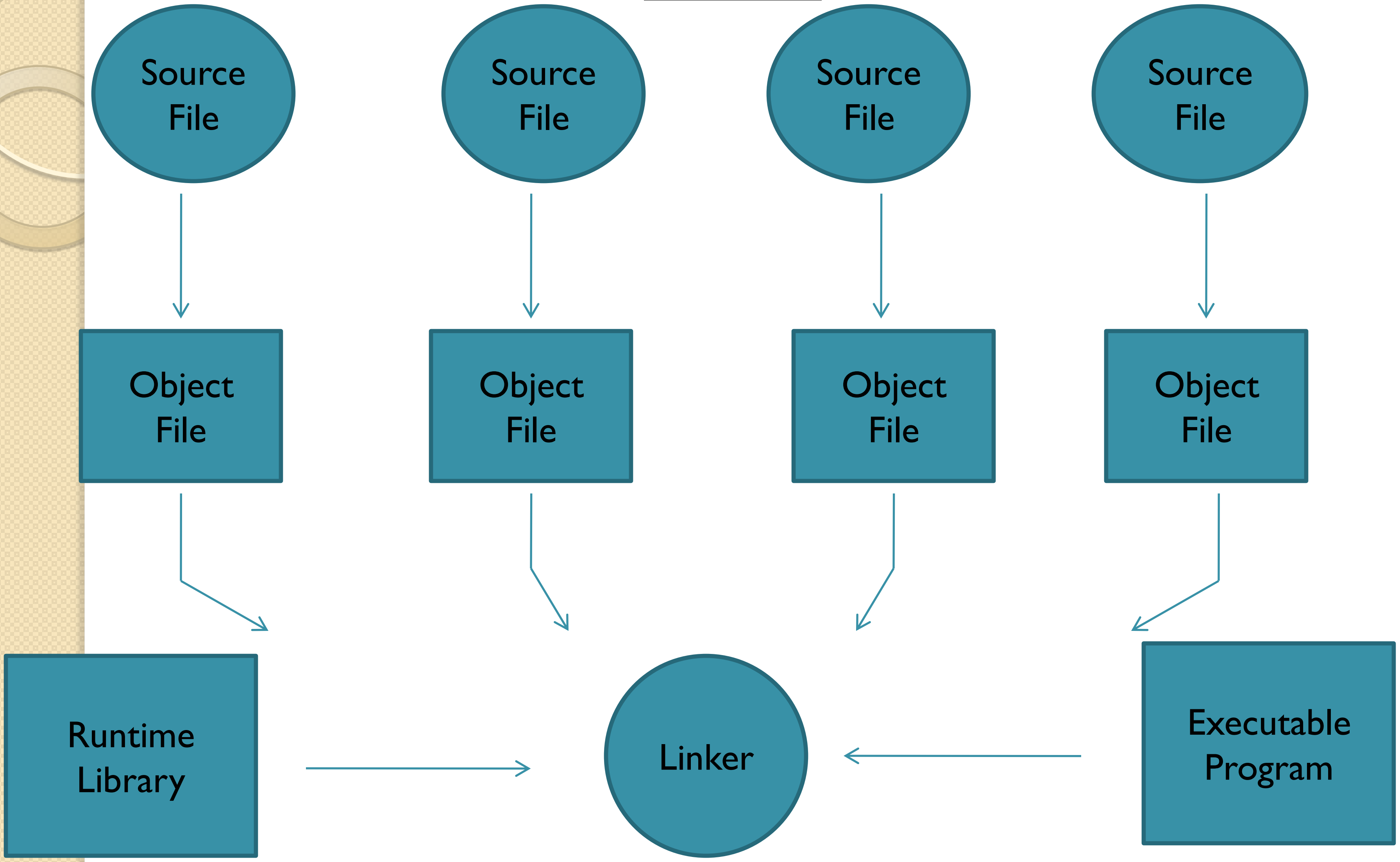
Types of Text Editors

- Vi Editor
- Notepad

Linkers

A linker is a program that combines object modules to form an executable program. Many programming languages allow you to write different pieces of code, called *modules*, separately. This simplifies the programming task because you can break a large program into small, more manageable pieces. Eventually, though, you need to put all the modules together. This is the job of the linker. Linkers can take objects from collection called a library. Some linkers do not include the whole library in the output; they only include its symbols that are referenced from other object files or libraries. Libraries for diverse purposes exist, and one or more system libraries are usually linked in by default.

Linker



Source File
Source File
Source File
Source File
Object File
Object File
Object File
Object File
Runtime Library
Linker
Executable Program

Code Generators

When we implement software using a programming language such as Ada , C or Fortran, the compiler translates the program code into a machine-understandable code. However there are other languages that may be used for implementation. These languages don’t usually have a compiler. The code created in those languages does not translate directly into machine understandable code. Code generators often shorten the length of actual code generators by the programming team by a factor of 10 or more.

Program Design Languages

Program Design Language is a technique for designing and documenting methods and procedures in software. It is related to pseudocode , but unlike pseudocode, it is written in plain language without any terms that could suggest the use of any programming language or library.

The use of PDL's won't make writing Design specifications as easier task. It is more difficult to use a formal PDL than Pseudocode because one must have to learn another set of syntax and semantics.

Debugger

Debuggers are programs that behave as an interpreter. A debugger is a program which runs other programs in such a way as to let you see every step of program execution. A debugger will let you stop the program while it is running , change the program or program variables and start the program running again. In other words we can say that debugger gives the programmer the capability to inspect the execution state of the program in a symbolic way.

Debuggers are an indispensable tool in the development process. Infact, during the course of the average software project more hours are spent in debugging the software than in compiling code.

Workbenches

Workbenches are semi Integrated CASE tools designed for extensive support to the associated programmer while developing , analyzing and documenting the SRS and the SDS for a software system. CASE workbenches (CASEW) have improved steadily and have become more available. There are several components within a CASEW tool. The most important is creation and editing of data flow diagrams, structure charts, entity relation diagrams, code object diagrams, structure charts, entity relation diagrams code-object diagrams and many other graphic representation tools used in specification and design phases.

Ideal Software Development Plan

Planning is a critically important step in any successful venture, particularly for system and software development. Moreover , planning is something that most of us do not like to do. Patience, discipline and hard work are required to plan out a system or for software development process. To keep the plan current and at the same time gain maximum benefits from the planning effort the system development plan should be an agenda item for system development team meeting.

A system level project plan needs information from the software organization as input to its planning effort. Similarly the software development planning effort requires inputs from the project plan.

Software Development Plan consists of following

- How to treat reviews, walkthroughs and Inspections.
- How to treat formal and informal software documentation.
- How to manage problem resolution.
- How to manage and central the software development plan itself as it is being executed.
- How to manage and control interfaces with other performing organizations.

Ideal Software Development Plan

Table of contents

5. Technical management and control.
6. Standards and procedures.
7. Reviews/ audits and walkthroughs.
   I. Schedules.
   II. Procedures.
   III. Responsibilities.
8. Development Environment.
9. Technical Performance Measurements.
10. Documentation.
11. Verification and Validation.
12. Maintenance.