# LECTURE – 3

# SYSTEM PROGRAMMING & SYSTEM ADMINISTRATION

## SECTION -A

REFERENCES: SYSTEM PROGRAMMING BY JOHN J. DONOVAN (TMH EDITION)

&

GOOGLE SEARCH ENGINE

# INTRODUCTION

- **Interpreters**
- **Compilers**
- **Text editors**
- **Debug monitors**
- **Programming environment**

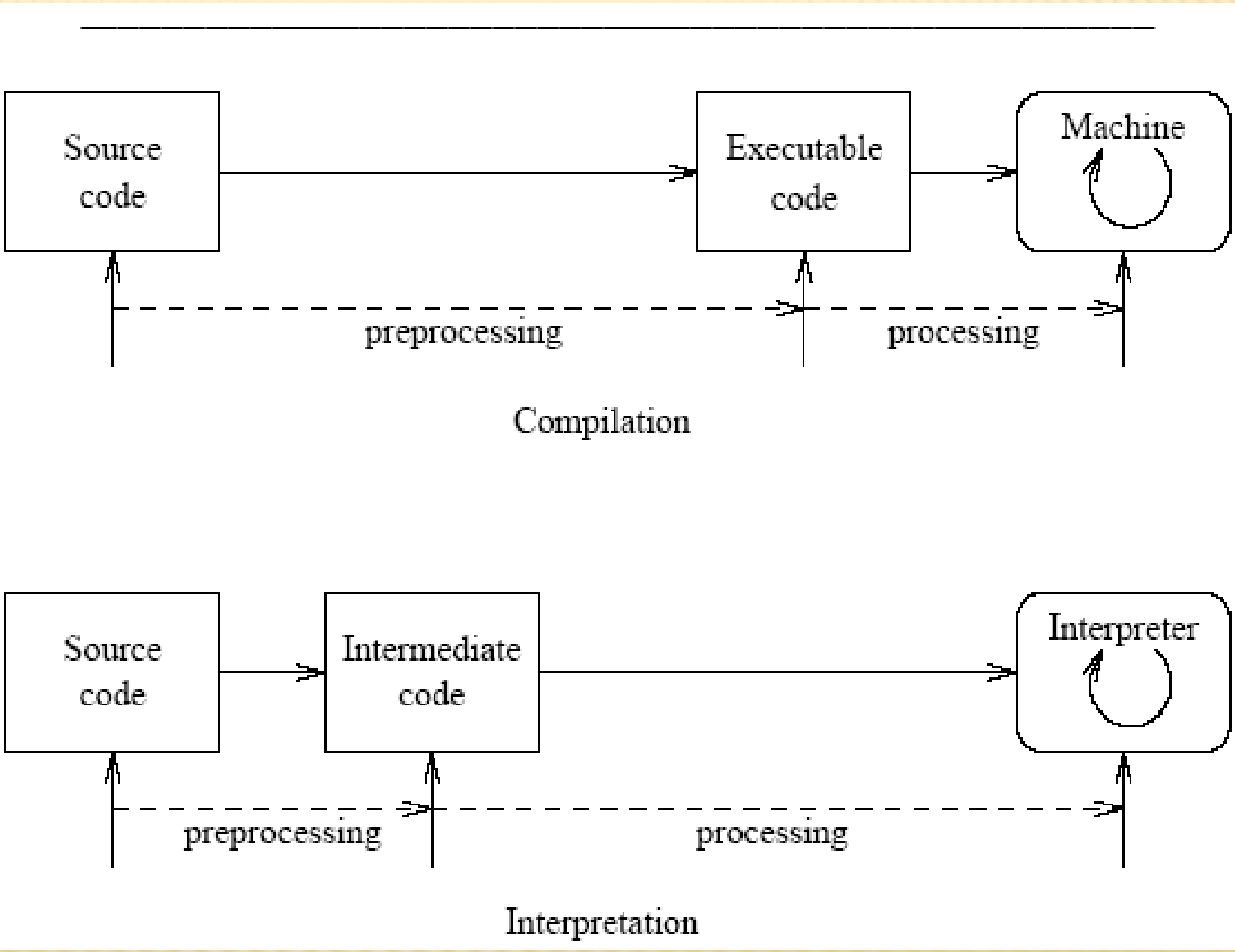
# INTERPRETERS

An *interpreter* may be a program that either

1. executes the source code directly
2. translates source code into some efficient intermediate representation (code) and immediately executes this,
3. explicitly executes stored precompiled code (**a *code* that is output from a compiler, ready to be executed.** )made by a compiler which is part of the interpreter system

# Compilers and Interpreter

- An interpreter translates some form of source code into a target representation that it can immediately execute and evaluate.
- The structure of the interpreter is similar to that of a compiler, but the amount of time it takes to produce the executable representation will vary as will the amount of **optimization** (improving a system to reduce runtime, bandwidth, memory requirements ).
- The following diagram shows one representation of the differences among compilers and interpreters


Source code
Executable code
Machine
preprocessing
processing
Compilation
Source code
Intermediate code
Interpreter
preprocessing
processing
Interpretation

## Compiler characteristics:

- spends a lot of time analyzing and processing the program
- the resulting executable is some form of machine- specific binary code
- the computer hardware interprets (executes) the resulting code
- program execution is fast

## Interpreter characteristics:

- relatively little time is spent analyzing and processing the program
- the resulting code is some sort of intermediate code
- the resulting code is interpreted by another program
- program execution is relatively slow

# Compilers

- A **compiler** is a computer program (or set of programs) that transforms source code written in a programming language (the *source language*) into another computer language (the *target language*, often having a binary form known as *object code*). The most common reason for wanting to transform source code is to create an executable program.
- The name "compiler" is primarily used for programs that translate source code from a high-level programming language to a lower level language (e.g., assembly language or machine code). A program that translates from a low level language to a higher level one is a *decompiler*.

# TEXT EDITORS

- A **text editor** is a type of program used for editing plain text files.
- Text editors are often provided with operating systems or software development packages, and can be used to change configuration files and programming language source code.
- (In computing, **configuration files**, or **config files** configure the initial settings for some computer programs. They are used for user applications, server processes and operating system settings.)

# TYPES OF TEXT EDITORS

- Some text editors are small and simple, while others offer a broad and complex range of functionality. For example, Unix and Unix-like operating systems use the vi editor (or a variant).
- Many text editors for software developers include source code syntax highlighting and automatic completion to make programs easier to read and write.
- Programming editors often permit one to select the name of a subprogram or variable, and then jump to its definition and back.

# TYPICAL FEATURES

- Cut, copy, and paste – most text editors provide methods to duplicate and move text within the file, or between files.
- Text formatting – Text editors often provide basic formatting features like line wrap, bullet list formatting, comment formatting, and so on.
- Undo and redo – As with word processors, text editors will provide a way to undo and redo the last edit.
- Syntax highlighting –highlights software code and other text that appears in an organized or predictable format.

# DEBUG MONITORS

- A debug monitor, is a tool that helps to find and reduce the number of bugs and defects in a computer program or any electrical device within or attached to the computer in order to make it act the way it should.
- For example, when an armored car drives up to a bank and the guards have to transfer money from the truck to the bank, there are special guards that stand watch to make sure no one tries to rob them thus making the transaction go smoothly.
- Those guards could be the debug monitors in the computer industry.

- If the debugging monitor locates a bug or defect in any of the equipment, it will first try to reproduce the problem which will allow a programmer to view each string (in a program) that was within the bug or defect range and try to fix it.

# PROGRAMMING ENVIRONMENT

- DEFINITION- The programming environment is the set of processes and programming tools used to create the program or software product.
- An integrated development environment is one in which the processes and tools are coordinated to provide developers an convenient view of the development process (or at least the processes of writing code, testing it, and packaging it for use).
- An example of an IDE product is Microsoft's Visual Studio .NET. And Oracle JDeveloper 10g and Eclipse (an IBM product) for Java development
- An integrated development environment (IDE) is a programming environment that has been packaged as an application program, typically consisting of a code editor, a compiler, a debugger, and a graphical user interface (GUI) builder.

- The IDE may be a standalone application or may be included as part of one or more existing and compatible applications.
- IDEs provide a user-friendly framework for many modern programming languages, such as Visual Basic, Java etc.
- IDEs for developing HTML applications are among the most commonly used. For example, many people designing Web sites today use an IDE (such as HomeSite, DreamWeaver, or FrontPage) for Web site development that automates many of the tasks involved.

# Program Generators

- Software program that enables an individual to easily create a program of their own with less effort and programming knowledge.
- With a **program generator** a user may only be required to specify the steps or rules required for his or her program and not need to write any code or very little code.
- Some great examples of a program generator are: Adventure Maker, Alice, Stagecast Creator, and YoYo Games

- **Alice** is an innovative 3D programming environment that makes it easy to create an animation for telling a story, playing an interactive game, or a video to share on the web. Alice is a teaching tool for introductory computing. It uses 3D graphics and a drag-and-drop interface to facilitate a more engaging, less frustrating first programming experience.
- Adventure Maker: The World's easiest way to create point-and-click games and virtual tours for Windows, iPhone, and iPod touch!
- Stagecast Creator: Make your own video games, share them with friends, learn thinking skills…

# APPLICATIONS

- Beginner-friendly mouse operation.
  You basically don't even need a keyboard.
- Complicated environmental configuration not required.
  Just start up and start creating sample software immediately.
- Since the module library provided even defines MCU peripheral circuits, you can operate your MCU as soon as software combination is complete.
- The Sample Application Program Generator & Organizer system is C-compliant.
- The combination of supported program modules enables highly-flexible software creation.
- Since the source file of the created software is output, it can also be used as a base for development.
- As control of the Sample Application Program Generator & Organizer system utilizes only the comment line, ROM/RAM cannot be increased.
- User programs can be registered in a library at the function level using the GUI..
- The program module, like sample software, is provided free of charge. Please note that proper operation is not guaranteed.

# SCOPE OF RESEARCH

A well-known problem in program generation is *scoping. When identifiers* (i.e., symbolic names) are used to refer to variables, types, or functions, program generators must ensure that generated identifiers are *bound to their* intended *declarations. This is the standard scoping issue in programming languages,* only automatically generated programs can quickly become too complex and maintaining bindings manually is hard. In this paper we present *generation scoping: a language mechanism to facilitate the handling of scoping* concerns. Generation scoping offers control over identifier scoping beyond the scoping mechanism of the target programming language (i.e., the language in which the generator output is expressed). Generation scoping was originally implemented as an extension of the code template operators in the Intentional Programming platform, under development by Microsoft Research. Subsequently, generation scoping has also been integrated in the JTS language extensibility tools. The capabilities of generation scoping were invaluable in the implementation of two actual software generators: DiSTiL (implemented using the Intentional Programming system), and P3 (implemented using JTS).