BACHELOR OF COMPUTER APPLICATIONS

SEMESTER 4

DCA2203

SYSTEM SOFTWARE

Unit 8

Text Editor

Table of Contents

1. INTRODUCTION

In the previous unit, the discussion was the Interpreter, overview of the compiler, difference between compiler and interpreter, scanning, symbol table, parsing expression and assignment, control statements, and Simple interpreter design.

In this unit, will study the text editor. An overview of the editing process, user interface, and editor structure. Will also discuss the interactive debugging system, debugging functions and capabilities, its relationship with other parts of the system, and the user interface criteria.

1.1 Learning Objectives:

After studying this unit, the learner would be able to:

- *List the various types of editors*
- *List the various types of user interface*
- *Explain about the editor structure*
- *Describe the interactive debugging system*
- *Describe the debugging functions and capabilities*
- *Discuss the relationship with other parts of the system*
- *discuss the user interface criteria*

2. TEXT EDITOR: AN INTRODUCTION

Let's talk generally about an editor before we go into any discussions about text editors. The name itself implies that the major function of an editor is editing. Apart from editing, the editor does other supporting services editing. There are various functions which are supposed to be performed by an editor. The major functions of an editor are editing, traveling, viewing, and displaying. The function of the editor is also to create files. A file is a group of characters defined by its creator. The creator may define a file as a program file or a database file or a document file. The term document includes objects such as computer programs, text, equations, table linkers, loaders and Operating Systems (OS), diagrams, line art, and photographs or anything one can get on a printed form. It is only when we create a file that the above-said operations can be performed. The editor must be capable of allowing the user to create a file of his interest.

The term editing means modifying, updating, deleting, and formatting of the text. Traveling means the ability to travel the text in the file. This allows the user to move the cursor over the text. The viewing option allows the user to create a view of the file. This involves the selection of font, style, colors, and the like. This view created can be called an abstract view.

Displaying and printing text is another function of the editor. The editor should be able to generate the printable view of text objects and display it on the screen. That is editor should be able to convert the abstract view into the physical view and should also be able to print it. The editors should be able to provide facilities for all the functions mentioned above. Many other advanced features like spell checking and grammar checking as the document is created. The editors can have the facilities for editing the image and pictures too.

If a file is created in an editor, ASCII codes for all the letters, numbers, and characters are stored in the main memory by the editor. The file is created in the main memory (RAM) only. When the creation of the file is completed, the file is stored in the secondary memory either on the floppy disk or on hard disk with the corresponding extension.

There are various types of Editors. Depending upon the types of characters to be edited, editors can be broadly classified as:

i) Text Editors and
ii) Graphical Editors.

Depending upon how the editor edits, the editor can also be classified into

i) Stream Editor,
ii) Line Editor,
iii) Screen Editor and
iv) Structure Editor.

In this unit, will discuss mainly text editors. A text editor is a computer program that lets users enter, change, store, and usually print text. These texts are characters and numbers. The texts are encoded by the computer and input/output devices which are again arranged so that users or other programs can understand them. In a text editor, the primary elements being edited are character strings of the target text.

Typically, a text editor provides an "empty" display screen (or "scrollable page") with a fixed-line length and visible line numbers. The lines can then be filled in line by line with text.. A special command line lets you move to a new page, scroll forward or backward, make necessary changes in the document, save the document and perform other actions. A text editor allows the user to create the source program in the form of text in the main memory. Generally, the user prepares High-Level Language program as a source program. The creation, editing, modification, deletion, and updating of documents or files can be done with the help of the text editor. The scope of the editing is limited to the text only. Text editors can enter program language source statements or create documents such as technical manuals. It can also be used to create a file containing C programs, C++ programs, assembly language programs and similar programs. They support the creation and modification of documents and allow modifying or updating of files. For example, editing the text when writing a document, send emails, make a Web page, program, and many more purposes.

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. E.g., Notepad is provided in Windows OS whereas gedit or kedit editor is found in Linux OS.

Text editors can be considered the primary interface to the computer for all types of users who manipulate computer-based information.

Text editors are mostly used by (but not limited to):

- Programmers
- Authors and secretaries of authors
- Knowledge workers (who compose, organize, study and manipulate computer-based information)

The editors can be developed using programming tools like C, C++, VB, and VC++. An example of a popular text editor in IBM's large or mainframe computers is XEDIT. The two most commonly used text editors in UNIX systems are Emacs and vi.

Some of the popular text editors available are:

ae: ae, is a simple, easy-to-use text editor. It has modes to emulate the behavior of other text editors.

Cooledit: Cooledit is a popular, fast text editor for use in **X window systems.** Its features include anti-aliased fonts, Unicode support, and extensibility via the Python programming language

Dedit: DEdit is a simple editor for use in X **window system** with GNOME installed. It can read compressed files and display Japanese characters.

Ee: It is an editor that novices can begin to use immediately; the Easy Editor features pop-up menus.

Elvis: Elvis is a modern implementation of VI editor that comes with many new features and extensions.

Emacs: Emacs is one of the most popular text editors.

Jed: John E. Davis's jed editor is specifically for programmers whose features include drop-down menus that work in the console. jed loads quickly and makes editing files at a shell prompt easy and fast.

Joe: Joe's Editor or joe in short is a full-screen editor with a look and feel reminiscent of old DOS text editors like EDIT.

Nano: Nano is a free software editor.

Ted: Ted is a WYSIWYG text editor for use in X, which reads and writes `.rtf' files in Microsoft's "Rich Text Format."

The: The Hessling Editor is a configurable editor that uses the Rexx macro language. It was inspired by the XEDIT editor for VM/CMS and the Kedit editor for DOS.

Vi: Vi (pronounced "vye," or sometimes "vee-eye") is a visual, or full-screen, editor. Vi is a most popular editor on Linux and Unix-based systems.

Vim: Vim ("Vi improved") is a modern implementation of Vi whose features include syntax coloring, scrollbars and menus, mouse support, and built-in help.

Wily: Wily is a mouse-centric editor whose commands consist of various combinations of the three mouse buttons, called chords, which can be tricky to master.

Xedit: Xedit is a simple text editor that comes with and works in, X. It lets you insert, delete, copy and paste text and open and save files.

Xemacs: XEmacs is a version of Emacs with advanced capabilities for use in Xwindow system, including the ability to display images.

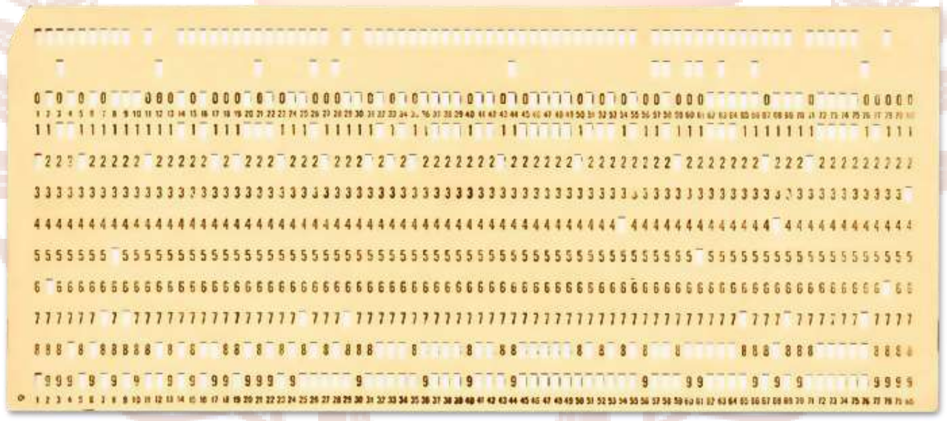
Punch cards:

Punch cards with holes punched into them to represent computer data and instructions are known as punch cards (or "punched cards"), Hollerith cards, or IBM cards. They were a common way for people to enter data into early computers. The cards were put into a card reader that was attached to a computer, which translated the holes' order into digital data.

For instance, an early computer programmer might write a program by hand, then use a punch card machine to convert it to multiple punched cards. The programmer would then bring the stack of cards to a computer and input the software there using a card reader. An illustration of a woman using a punch card machine to make a punch card is shown.

By punching holes in each column, which stand for one character, using a punch card machine like the one in the image above, data can be entered into the card. Here is an illustration of a punch card.

Example of a punch card



When a card is finished or the Return key is pushed, the information is technically "stored" on the card. If you develop a programme using punch cards (one card for each line of code), it requires a stack of cards because each card can only carry so much data.

Each card is placed in a punch card reader to read the data from the card and load the software into the computer. The punch card reader begins reading the card as soon as it is inserted, moving vertically from top to bottom. The card reader advances to the following column after reading one column. The information would be written to a computer's memory as the reader read it. The computer would be told to run the code when all cards had been loaded into memory. Punch cards would be produced as information was output (printed).

Self-Assessment Questions - 1

1. The "__________"Should generate a printable view.
2. If a file is created in an editor, then it is created in the "__________"Only.

2.1 Overview Of The Editing Process

Now let's have a look at the editor's editing process in general. An Interactive Editor is a computer program that allows users to create and revise a target document. The term document means an object such as computer programs, text, equations, tables, diagrams, photographs, etc. The document-editing process is an interactive user-computer dialogue designed to accomplish four tasks. They are:

1. Select the part of the target document to be viewed and manipulated.
2. Determine how to format this view online and how to display it.
3. Specify and execute operations that modify the target document.
4. Update the view appropriately.

Whenever the need to select the part of the document for viewing and editing, then "travelling" is done through the document to find that part of interest. Travelling implies the movement of the editing context to a new position within the text. Filtering controls the selection of what is to be viewed and manipulated. Filtering extracts the relevant subset of the target document at the point of interest, such as the next screen full of text or the next statement. Next is do "Formatting" to determine a visible representation on a display screen or other devices.

Editing means the way a document is created or altered with a set of operations such as insertion, deletion, replace, move and copy. To build a document editorone should think about its internal structure, formatting, Graphical User Interface (GUI), and spell check. The contents of this document are (i) Computer programs (ii) Text information (iii) Equations (iv) Tables

(v) Line art and (vi) Photographs.

The editing functions are provided to operate on elements necessary for the type of editor used. For example, a manuscript-oriented editor is provided to operate on elements such as single characters, words, lines, sentences, and paragraphs. A program-oriented editor is used to operate on elements such as identifiers, keywords, and statements. The actual editing phase has a set of operations such as Insert, Delete, Replace, Move and Copy.

2.2 Types Of Editors

It is already mentioned earlier that the editors may be broadly classified into

(i) Textual editors (ii) Graphical editors. Editors can also be classified as Stream Editor, Line Editor, Screen Editor, and Structure Editor. (See figure 8.1)

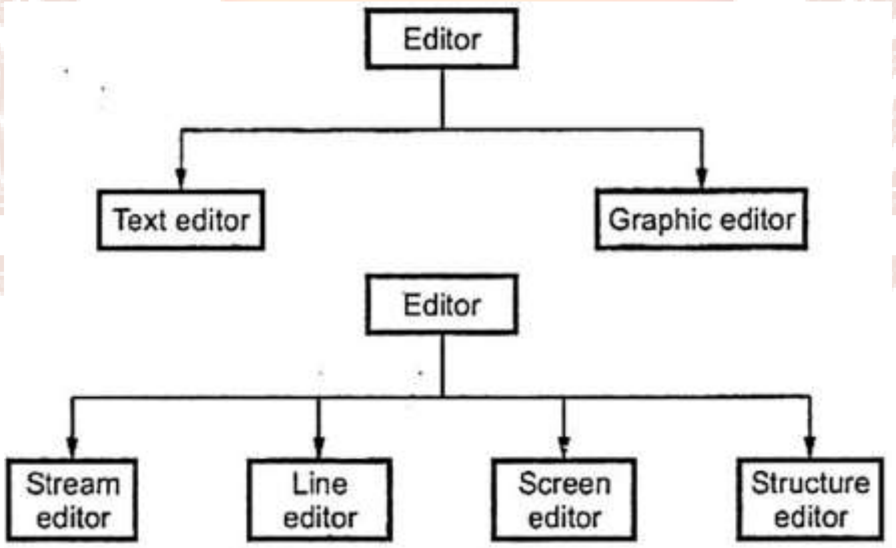


Figure 8.1: Different types of editors

- **Textual editors:** As already explained, they are the primary interface to the computer for all types of users who manipulate computer-based information.
- **Graphical editor:** A graphical editor is a program which allows the user to create the source program in Graphical User Interface (GUI) based editors. These are much user friendly.
- **Line editors:** Line editor is one of the simplest and most primitive form of editor, which uses a buffer to store information. It requires you to specify a specific line of text before making changes to it. It operates in command mode. The user gives the command to the

editor for any operation and the editor will respond to that command. (Buffer is in the main memory. That is set aside to store the information entered from the keyboard.)

- **Screen editors:** It is also called screenful-screen editors. A screen editor displays a screen full of text at a time. The user can move the cursor over the screen, position it at the point where the user desires to perform some editing, and proceed with the editing directly. The user has full control over the entire terminal. For example, the user can bring the cursor over a character to be deleted and press a delete key. It is possible to see the effect of an edit operation on the screen.
- **Stream editor:** A stream editor views the entire text as a stream of characters. These permit edit operations to cross line boundaries. Stream editor typically supports character line and context orient commands. In the stream editor, the current editing context is indicated by the position of the text pointer. This pointer can be manipulated using positioning. The stream editor maintains multiple representations.
- **Structure Editor:** The structure editor is a graphical editor that you can use to model organizational structures and relationships.

2.3 User Interface

The user interface (UI) is an editor for a user to interact with an application, as shown in figure 8.2. The user interface editors can be categorized into

(i) Interactive Editors (ii) Line Editors (iii) Screen Editors.

UI consists of a Dialog manager, presentation manager, and graphics package. The dialog manager manages the conversation between the user and the application. The Presentation manager displays the data produced by the application in an appropriate manner on the user display. A application called the graphics package gives you graphic data.

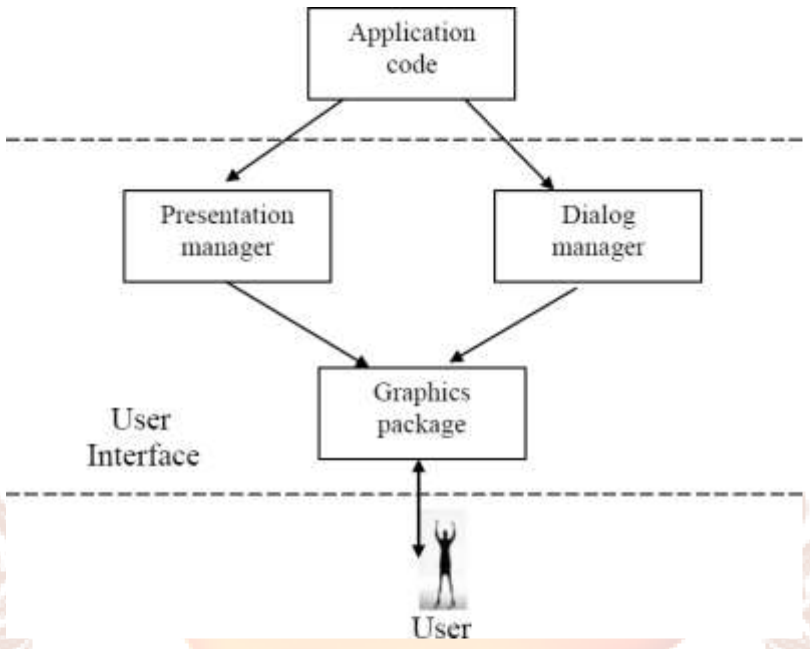


Figure 8.2: User interface

The dialog manager, is also responsible for error messages and online help function and for organizing changes in the visual context of the user. User Interface Management System (UIMS) automates the generation of user Interface. UIMS takes the specification of the presentation and dialog semantics to produce the presentation and dialog managers of the UI.

The user of an interactive editor is provided with a conceptual model of the editing system. . With a set of rules outlining how operations affect these aspects, this model offers a readily understood abstraction of the target document and its constituent parts.. In other words, this model is an abstract framework on which the editor and the world on which it operates are based.

The conceptual model essentially offers a readily understandable abstraction of the target document and its constituent parts; Besides the conceptual mode, the user interface is concerned with the input devices, the output devices, and the interaction language of the system.

Input devices are used to enter elements of the text to be edited, to enter commands, and to define editable elements. These input devices, as used with editors, can be divided into three categories:

- Text devices
- Button Devices
- Locator devices

Text or string devices are typically typewriter-like keyboards on which a user presses and releases keys, sending a unique code for each key.

Button or choice devices: These devices generate an interrupt or set a system flag. It usually causes the invocation of an associated application-program action. Such devices include a set of special function keys on an alphanumeric keyboard or on display itself.

Locator devices: These types of devices are two-dimensional analog to digital converters that are used to position a cursor symbol on the screen by observing the user's movements of the device. Such devices include joysticks, touch screen panels, data tablets, and mouse.

A locator device combined with a button device allows the user to specify either a particular point on the screen at which text should be inserted or deleted, or the start and end points of a string of characters to be operated upon.

Text devices with arrow (cursor) keys are often used to simulate locator devices. Each of these keys shown is an arrow that points up, down, left, or right. An advanced input device, namely a voice input device, translates spoken words into texts.

An output device is used to view the edited elements and results of the editing operations. Output devices like advanced CRT terminals use hardware assistance for moving the cursor, inserting and deleting characters and lines, and scrolling lines and pages.

Interaction language

The interaction language of a text editor is generally one of several common types. The categories of user interface interaction language are:

i) Typing-Oriented or Text Command oriented [For e.g.: Unix editors]

ii) Function-Key editor [For e.g.: 'C' editor]

iii) Menu-oriented user interface or Icons [For e.g.: Windows editors].

i) The typing oriented or text command-oriented: In this type of editing interface, the user communicates with the editor by typing text strings both for command names and for operands. These strings are sent to the editor and are usually displayed on the output device.

Example: Unix editors

ii) The function key interface: In this type of interface, each command has associated with it a marked key on the user's keyboard. Function-key command specification is typically coupled with cursor-key movement for specifying operands, which eliminates much typing. For the common commands in a function-key editor, usually, only a single key need to be pressed.

Example: 'C' editor

iii) The menu-oriented user interface: It is a multiple-choice set of text strings or icons, which are graphic symbols that represent objects or operations. The editor prompts the user with a menu of only those actions that can be taken at the system's current state.

Example: Windows editors

Typing-oriented systems require familiarity with the system and language, as well as some expertise in typing. Function-key oriented systems often.

have either too few keys, requiring multiple keystroke commands, or have too many unique keys. The menu-oriented user interface has a menu and provides a multiple-choice set of text strings or icons, which are graphic symbols representing objects or operations. The user can perform actions by selecting items from the menu.

Self-Assessment Questions - 2

3. Which editor allows a user to enter, change, store, and usually print text?
   a) text
   b) graphic
   c) stream
   d) line
4. Which editor allows a user to create and revise a target document?
   a) text
   b) interactive
   c) stream
   d) line
5. The "__________ "Process is an interactive user-computer dialogue.
6. A graphical editor is a program which allows the user to create the source program in "_________"based editors.
7. The use of an interactive editor is presented with a "__________ "Model of the editing system.

2.4 Editor Structure

As already said, the fundamental functions in editing are traveling, editing, viewing, and displaying. The components of the editing structure are:

- Editing Component
- Traveling Component
- Viewing Component
- Display Component
- Filter Component

Most text editors have a structure similar to the one shown in figure 8.3.

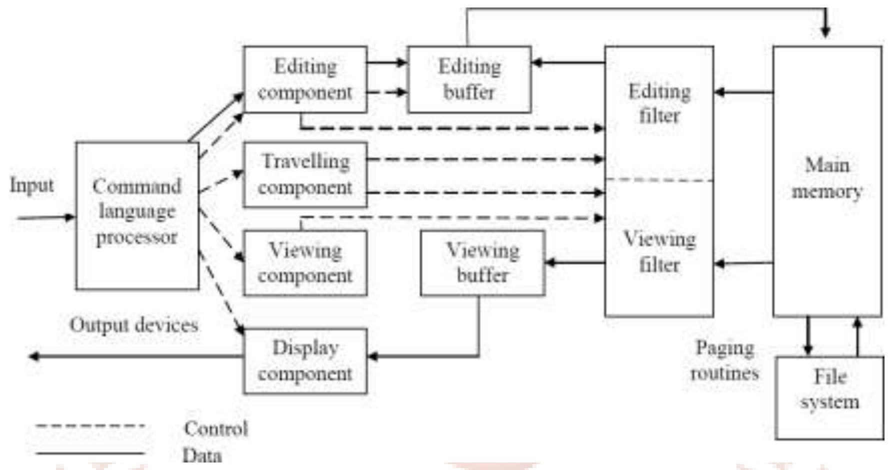


Figure 8.3: Typical Editor structure

As shown in figure 8.3, the command language processor accepts input from the user's input devices and analyses the tokens and syntactic structure of the commands. In this sense, the command language processor functions much like the lexical and syntactic phases of a compiler; just as in a compiler, the command language processor may invoke semantic routines directly. In a text editor, these semantic routines perform functions such as editing and viewing. Alternatively, the command language processor may produce an intermediate representation of the desired editing operations. An interpreter that invokes the appropriate semantic routines then decodes this intermediate representation. The use of an intermediate representation allows the editor to provide a variety of user-interaction languages with a single set of semantic routines driven from a common intermediate representation.

The semantic routines involve traveling, editing, viewing, and displaying functions. In editing a document, the start of the area to be edited is determined by the current editing pointer maintained by the editing module. The editing module is a collection of modules dealing with editing tasks. The user can set or reset the current editing pointer explicitly with traveling commands, such as the next paragraph and next screen. Or editing pointer can be set or reset implicitly by the system. The traveling module of the editor performs the setting of the current editing and viewing pointers.

The editing module invokes the editing filter when the user issues an editing command . The editing filter then filters the document to generate a new editing buffer based on the current editing pointer and the editing filter parameters. These parameters are specified by the user and the system as well. These parameters are information about the range of text that can be affected by an operation. Filtering consists of the selection of contiguous characters that begins at the current point. The semantic routines of the editing component then operate on the editing buffer, essentially a filtered subset of the document data structure.

Similarly, viewing a document determines the start of the area to be viewed by the current viewing pointer. This pointer is maintained by the viewing module of the editor, which is a collection of modules responsible for determining the next view. The current viewing pointer can be set or reset explicitly by the user with a traveling command or implicitly by the system the previous editing operation. The viewing component invokes the viewing module when the display needs to be updated. . This component filters the document to generate a new viewing buffer based on the current viewing pointer and the viewing filter parameters.

In line editors, the viewing buffer can contain the current line. In screen editors, this buffer may contain a rectangular cutout of the quarter-plane of text. This viewing buffer is then passed to the display module of the editor, which produces a display by mapping the buffer to a rectangular subset of the screen, called a window or viewport.

While independent, the editing and viewing buffers can be related in many ways. In the simplest case, editing and viewing buffers are identical when the user edits the material directly on the screen (see figure 8.4).

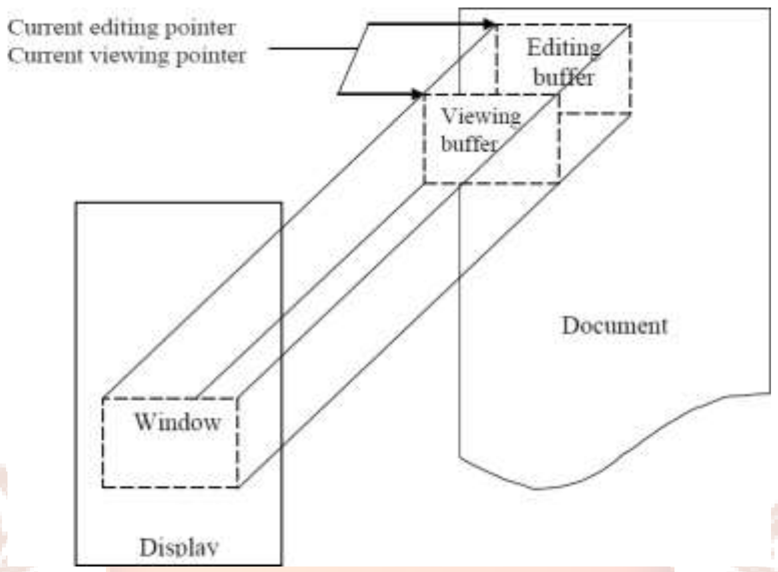


Figure 8.4: Simple relationship between editing and viewing buffers

The editing and viewing buffers can also partially overlap, or one may be completely contained in the other. Windows typically cover either the entire screen or a rectangular portion of it. Mapping viewing buffers to windows that cover only part of the screen is especially useful for editors on modern graphics-based workstations. Such systems can support multiple windows, simultaneously showing different portions of the same file or portions of different files. This approach allows the user to perform inter-file editing operations much more effectively than with a system having a single window.

The mapping of the viewing buffer to a window is accomplished by two system components. First, the viewing component formulates an ideal view, often expressed in a device-independent intermediate representation. This view may be very simple, consisting of a window's worth of text arranged so that lines are not broken in the middle of words. At the other extreme, the idealized view may be a facsimile of a page of fully formatted and typeset text with equations, tables, and figures. Second, the display module takes this idealized view from the viewing component and maps it to a physical output device in the most efficient manner possible.

Self-Assessment Questions - 3

8. The "__________" module of the editor performs the setting of the current editing and viewing pointers.
9. In viewing a document, the start of the area to be viewed is determined by the current viewing "_________".

3. INTERACTIVE DEBUGGING SYSTEM

Let us now study what an Interactive debugging system is. An interactive debugging system is a system that provides programmers with facilities to aid in the testing and debugging of programs. Although the desirability of such systems has been recognized for some time, there are relatively few actual debugging systems in practical use.

The functions of the Interactive debugging system are:

- Debugging functions and capabilities
- Relationship with other parts of the system.
- User Interface criteria.

3.1 Debugging Functions And Capabilities

The programmer can specify the most obvious requirement for a set of unit test functions. One of the functions deals with execution sequencing, which is the observation and control of the flow of program execution. After execution is suspended, other debugging commands can be used to analyze the program's progress and diagnose the error detected; then, execution of the program can be resumed. Given a good graphic representation of program progress, enabling the program to run at various speeds called gaits may be useful .

There are other functions, such as tracing and tracing back which should be provided by a debugging system. Tracing tracks the flow of execution logic and data modifications. At various levels of detail, such as a module, subroutine, branch instruction, and so forth, the control flow can be followed. Traceback shows the path of the current statement. For a given variable or parameter, trace back can show which statements modified it. Such information should be displayed symbolically.

A debugging system should also have good program display capabilities. It must display the program being debugged, complete with statement numbers. The system should be able to save all the debugging.

specifications for a recompilation so the programmer does not need to reissue all these debugging commands. A debugging system should consider the language in which the program being debugged is written. Debugger commands that initiate actions and collect data about a program's execution should be common across languages. However, a debugging system must be sensitive to the language being debugged so that procedural, arithmetic, and conditional logic can be coded in the syntax of that language.

However, to get these requirements, a debugger has several consequences for the other software. When the debugger receives control, the execution of the debugged program is suspended temporarily. . The debugger must then be able to determine the language in which the program is written and set its context accordingly.

The notation used to specify certain debugging functions varies according to the program's language. The debugger must have access to information gathered by the language translator. The internal symbol dictionary formats vary widely between different language translators. Future compilers and assemblers should aim toward a consistent interface with the debugging system. One approach is that the language translators can produce the needed information in a standard external form for the debugger regardless of the internal form used in the translator. Another possibility would be for the language translator to provide debugger interface modules that can respond to requests for information in a standard way, regardless of the language being debugged.

It is also important that a debugging system be able to deal with optimized code. Many optimizations involve the rearrangement of segments of code in the program. Blocks of code may be rearranged to eliminate unnecessary branch instructions, which provides more efficient execution. The user of a debugging system deals with the source program in its original form before optimizations are performed. The debugging of optimized code requires a substantial amount of cooperation from the optimizing compiler. In particular, the compiler must retain information about any transportation it performs on the program. Such information can be made available to the debugger and the programmer.

These are some of the advantages of debugging:

- ✓ However, some reported errors may be false alarms and hence a wastage of time.

- ✓ Error reported: Reports a problem state right away.
- ✓ Provides information: Gives helpful details about data structures and makes interpretation simple.
- ✓ Reduces useless data : Aids the developer in eliminating unnecessary and confusing information.
- ✓ Save time, energy: Cuts down on complex one-use testing code, saving time and resources for software development.

Unit Test functions:

- Execution Sequence:
- Break points
- Conditional expressions
- Gaits

The importance of the execution sequence should be investigated in order to grasp its operational principles:

- It comprises keeping an eye on and managing how program are run.
- After a certain number of instructions have been executed, It can be stopped.
- There isn't a start to the curriculum again.
- Debugging functions such as Traceback and Tracing are important :
- Under Tracing:
- Data changes and execution logic flow are monitored.
- It is only complete after several adjustments.

Under Traceback:

- demonstrates the approach utilized to begin the sentence's present execution.
- When tracing back, only the current statement's path is taken.

3.2 Relationship With Other Parts Of The System

An interactive debugger must be related to other parts of the system in many different ways. It must appear to be a part of the run-time environment and an integral part of the system. The single most important requirement for any interactive debugger is that it always be available. When an error is discovered, immediate debugging must be possible because it may be different or impossible to reproduce the program failed in another environment or at another time. Therefore, the debugger must communicate and cooperate with other operating system components, such as interactive subsystems.

Debugging is even more important at production time than it is at application-development time. The debugger must also exist in a way consistent with the system's security and integrity components. Use of the debugger must be subject to the normal authorization mechanisms and leave the usual audit trails. The debugger must coordinate its activities with those of existing and future language compilers and interpreters.

3.3 User Interface Criteria

Let us now study the different criteria's that a user interface should bear. The interactive debugging system should:

- Be simple in its organization and familiar in its language
- Have facilities organized into a few basic categories of function
- Have function that closely neglect common user tasks
- Be able to contribute to ease of training and ease of use.

Other than the above-mentioned criteria, the user interaction should take advantage of full-screen terminal devices when they are available so that more information can be displayed and changed easily and quickly. Menus should reflect a task that a user can use. Menus should have titles that identify the task. Directions should precede any choices available to the user. Techniques such as indentation should help separate portions of the menu.

Interactive users should be supported when a screen terminal device is not present. Every action a user can take at a full-screen terminal should have an equivalent action in a linear debugging language. The command language should have a clear, logical, simple syntax. Commands should be simple rather than compound.

There should be consistent use of parameter names across the commands. Parameters should be automatically checked for errors in such attributes as type and range of values. Command formats should be as flexible as possible. The command language should minimize the use of such punctuation as parentheses, slashes, quotation marks, and other special characters. Any good interactive system should have an online HELP facility. HELP should be accessible from any state of the debugging session.

The user interface can be visualized as having two parts.

- Components of UI
  - Dialogue Manager:

Manages communication between the user and the application. It solicits a command from the user and sends it to the server.

  - Presentation manager

Displays the data from the program in a usable manner.

Conversations between the customer and the program are managed by the dialogue manager.

The presentation manager shows the details that the program generates.

An application's command dialogue is used to issue commands.

The below are few examples of how order dialogue can be implemented:

- Command languages
  - Resemble operating system command languages in certain ways.
  - Syntax as follows: <action> <parameters>
- Command menus
  - Ensure the app's casual users receive clear benefits.
  - The menus represent the most basic functions

- Direct manipulation
  - The entire device universe is available to the customer.
  - Displays the most important things in the application.

Let us see the Principles of Command Dialogue Design:

These are the principles that psychologists and engineers use to ensure that command dialogues are successful.

Engineers and psychologists have created the following features to guarantee the effectiveness of command dialogues, which include:

- The command system must be consistent
- User instructions receive immediate input
- Undo facility
- Shortcuts for experienced users
- Ease of use
- Error Handling
- To stop having to write down command information, use online assistance

The term "user interface," or "UI," refers to how a person interacts with a machine. Even while a light switch, for example, might legally be referred to as an instrument of UI, most contemporary allusions concern computers and other electronic devices.

The interaction between users and machines is made possible by UI. This essential method of communication would be impossible without it.

There are four prevalent types of user interface and each has a range of advantages and disadvantages:

- Command Line Interface
- Menu-driven Interface
- Graphical User Interface
- Touchscreen Graphical User Interface

Command Line Interface:

Although it is no longer frequently used as a fundamental user interface in ordinary consumer devices, the command line interface is still in use in some situations. Users using the command line interface must enter the proper instructions on the command line. The directive instructs the computer to open the necessary file or directory first. From there, a wide range of commands, such as those for retrieving files and running programmes, become available.

Advantages:

- Simple architecture
- very little memory utilization
- Excellent for memory-constrained or slowly operating computers
- A skilled CLI user can issue commands and complete tasks considerably more quickly than when using another type of UI.

Dis-advantages:

- learning a command language is challenging
- difficult for new users
- minimal information in error messages

Menu-Driven Interface

You can access a variety of commands or options through the menu-driven user interface in the form of a list or menu that is displayed in full-screen, pop-up, pull-down, or drop-down modes. An example of a menu-driven interface is an ATM.

Advantages:

- There is no need to keep track of a lengthy number of manual commands.
- For beginners, a simple interface
- Menu options with clear instructions

Dis-advantages:

- slower for knowledgeable users
- a menu with few choices
- To execute basic operations, you frequently need to open many menu screens.

Graphical User Interface

- The form of interface that most people are most accustomed to is the graphical user interface, or GUI. By pointing and clicking on images or icons with a mouse, tack pad, or other device, you can interact with these interfaces.

Advantages:

- Self-explanatory
- Simple to use
- There is no need to commit command lists to memory.
- enables the simultaneous use of several applications, programs, and tasks.
- solid infrastructure
- The same structure of several programs creates a sense of familiarity.

Disadvantages:

- Uses a lot of memory, yet as computers become more powerful, this becomes less of an issue.

Touchscreen Graphical User Interface

The touchscreen GUI is quite similar to the standard GUI, with the exception that you choose icons and carry out actions with your fingers or a stylus rather than a mouse or trackpad. Tablets, cellphones, and medical devices like the t:slim insulin pump all have touchscreen user interfaces. The touchscreen GUI offers a more personal way of interacting while having the same advantages and downsides as regular GUIs. Touchscreen GUIs are particularly practical because there are no peripherals required.

The graphical user interface, followed by the touchscreen variant, is by far the most popular of the four different forms of the user interface. The GUI continues to be the favored standard despite the other technologies that already exist and are developing. The simplicity and usability are largely to blame for this.

The majority of end users find it simpler to understand graphical user interfaces since they don't have to memorize or input complicated commands and because the icons and menus are typically self-explanatory.

Self-Assessment Questions - 4

10. Which of the following functions of the Interactive debugging system?
    a) Debugging functions and capabilities
    b) Relationship with other parts of the system.
    c) User Interface criteria.
    d) all of the above
11. In a good graphic representation of program progress, the program runs at various speeds called "________ ".
12. A debugging system should provide functions such as "_________and "________".
13. Which functions can be used to track the flow of execution logic and data modifications in debugging system?
    a) Tracing
    b) Trace back
    c) gait
    d) travelling
14. Which of the following determines the programming language and sets its context accordingly?
    a) Tracer
    b) gait
    c) User Interface
    d) debugger

4. SUMMARY

Let us recapitulate the important concepts discussed in this unit:

- The major function of an editor is editing. The editor performs other supporting services to editing like editing, traveling, viewing, and displaying.
- Traveling means the ability to travel the text in the file. The viewing option allows the user to create a view of the file. Displaying generates a printable view and displays it on the screen.
- The editors may be broadly classified into two types as (i) Textual editors (ii) Graphical editors. Editors can also be classified as Stream Editor, Line Editor, Screen Editor and Structure Editor.
- A text editor is a computer program that lets users enter, change, store, and usually print text.
- Text editors are often provided with operating systems or software development packages. They can be considered the primary interface to the computer for all types of users who manipulate computer-based information.
- Some of the popular text editors are Emacs, Jed, Joe, Nano, Vi, Vim, Xedit.
- The document-editing process is an interactive user-computer dialogue designed to accomplish different tasks.
- The user of an interactive editor is presented with a conceptual model of the editing system.
- Besides the conceptual mode, the user interface is concerned with the input devices, the output devices, and the interaction language of the system.
- An interactive debugging system provides programmers with facilities that aid in the testing and debugging programs.
- A debugging system should also provide functions such as tracing and trace back.
- An interactive debugger must be related to other parts of the system and must appear to be a part of the run-time environment and an integral part of the system.
- The criteria of User Interface are that the interactive debugging system should be simple in its organization and familiar in its language.

5. GLOSSARY

Interactive Debugging: System that provides programmers with facilities to

aid in the testing and debugging of programs.

Locator device: Two-dimensional analog to digital converters are used to position a cursor symbol on the screen by observing the user's movements of the device.

Text editor: A computer program that lets a user enter, change, store, and usually prints text.

Typing oriented interaction language: In editing interfaces, the user communicates with the editor by typing text strings both for command names and for operands.

User Interface (UI): An editor for a user to interact with an application as to produce the presentation and dialog managers of the UI.

User Interface Management System (UIMS): Automates the generation of user Interface and takes the specification of the presentation and dialog semantics.

6. TERMINAL QUESTIONS

Short Answer Questions

1. Explain what a text editor is. Write down the names of at least five popular text editors.
2. What is the different tasks document-editing process accomplishing? Explain.
3. Differentiate between Line editors and Stream editors.
4. Draw the diagram of a typical editor structure and explain about it.
5. What are the functions of interactive debugging systems? Explain the debugging functions and capabilities of interactive debugging systems.

7. ANSWERS

Self-Assessment Questions

1. editor
2. main memory (RAM)
3. a) text
4. b) interactive
5. document-editing
6. Graphical User Interface (GUI)
7. conceptual
8. traveling
9. pointer
10. d) all of the above
11. gaits
12. tracing, trace back
13. a) Tracing
14. d) debugger

Short Answer Questions

1. A text editor is a computer program that lets users enter, change, store, and usually print text. (Refer to section 8.2 for more details)
2. The document editing process accomplishes four different tasks. They are: 1. Select the part of the target document to be viewed and manipulated 2. Determine how to format this view online and how to display it (Refer to section 8.2.1 for more details)

   Line editors operate in command mode, where you can specify a specific line of text before making changes to it whereas a stream
3. editor views the entire text as a stream of characters. (Refer to section 8.2.2 for more details)
4. (Refer to section 8.2.4 for more details)
5. The functions of the Interactive debugging system are: (i) Debugging functions and capabilities (ii) Relationship with other parts of the system (iii) User Interface criteria. (Refer to section 8.3 for more details)

8. SUGGESTED BOOKS

- Leland L. Beck (1990). *System Software*, Addison-Wesley Publishing Company.
- M. Joseph. (2007). System Software. Firewall Media.
- *system programming*. (n.d.). Retrieved 09 06, 2012, from http://www.itswtech.org/Lec/Manal(system%20programming)/ch7_ system_software_tools.pdf
- *Text Editor*. (n.d.). Retrieved 09 16, 2012, from http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Text_editor.html