Debugging Strategies

- It is important to **study** the **system** in depth in order to understand the system. It helps the debugger to construct different representations of systems that are to be debugged.
- **Backward analysis** of the problem traces the program backward from the location of failure message in order to identify the region of faulty code. You need to study the region of defect thoroughly to find the cause of defects.
- **Forward analysis** of the program involves tracking the program forward using breakpoints or print statements at different points in the program. It is important to focus on the region where the wrong outputs are obtained.
- You must use the **past experience** of the software to check for similar problems. The success of this approach depends on the expertise of the debugger.

Debugging Software

This software plays a vital role in the software development process. Software developers use it to find the bugs, analyze the bugs and enhance the quality and performance of the software. The process of resolving the bugs using manual debugging is very tough and time-consuming. We need to understand the program, it's working, and the causes of errors by creating breakpoints.

As soon as the code is written, the code is combined with other stages of programming to form a new software product. Several strategies like unit tests, code reviews, and pair programming are used to debug the large program (contains thousands of lines of code). The standard debugger tool or the debug mode of the Integral Development Environment (IDE) helps determine the code's logging and error messages.

The steps involved in debugging software are,

- **The bug is identified in a system and defect report is created. This report helps the developer to analyze the error and find the solutions.**
- **The debugging tool is used to know the cause of the bug and analyze it by step-by-step execution process.**
- **After identifying the bug, we need to make the appropriate changes to fix the issues.**
- **The software is retested to ensure that no error is left and checks all the new errors in the software during the debugging software process.**
- **A sequence-based method used in this software process made it easier and more convenient for the developer to find the bugs and fix them using the code sequences.**

Debugging Techniques

To perform the debugging process easily and efficiently, it is necessary to follow some techniques. The most commonly used debugging strategies are,

- **Debugging by brute force**
- **Induction strategy**
- **Deduction strategy**
- **Backtracking strategy and**
- **Debugging by testing.**

Debugging by brute force is the most commonly used technique. This is done by taking memory dumps of the program which contains a large amount of information with intermediate values and analyzing them, but analyzing the information and finding the bugs leads to a waste of time and effort.

Induction strategy includes the Location of relevant data, the Organization of data, the Devising hypothesis (provides possible causes of errors), and the Proving hypothesis.

Deduction strategy includes Identification of possible causes of bugs or hypothesis Elimination of possible causes using the information Refining of the hypothesis(analyzing one-by-one)

The backtracking strategy is used to locate errors in small programs. When an error occurs, the program is traced one step backward during the evaluation of values to find the cause of bug or error.

Debugging by testing is the conjunction with debugging by induction and debugging by deduction technique. The test cases used in debugging are different from the test cases used in the testing process.

Debugging Techniques in Embedded Systems

These techniques reduce the error count and increase the quality and functionality of the code. Debugging of the embedded systems depends on physical memory addresses and virtual memory.

There are 6 debugging techniques in an embedded system.

- **Simplify the complex data**
- **Divide and conquer**
- **Slow down the process**
- **Change only one variable at a time**
- **Creating off-line models**
- **start from a known-good state.**

Different debugging techniques are used in different cases. A combination of one or more approaches may cause errors. This process includes

- **Reproduce the bug or problem**
- **Explain the bug using input from the user**
- **Try to get all the variable values and state of the program when the bug appears**
- **Analyze the bug and find the cause of the bug**
- **Fix the bug and check all the causes of new bugs.**

Debugging Tools

Debugging tool is a computer program used to test and debug other programs. There are a lot of public domain software like **gdb** and **dbx** that you can use for debugging. Also, they offer console-based command-line interfaces. Some of the automated debugging tools include code-based tracers, profilers, interpreters, etc.

A software tool or program used to test and debug the other programs is called a debugger or a debugging tool. It helps to identify the errors of the code at the various stages of the software development process. These tools analyze the test run and find the lines of codes that are not executed. Simulators in other debugging tools allow the user to know about the display and behavior of the operating system or any other computing device. Most of the open-source tools and scripting languages don't run an IDE and they require the manual process.

Mostly used Debugging Tools are GDB, DDD, and Eclipse.

Here is a list of some of the widely used debuggers:

- **Radare2**
- **WinDbg**
- **Valgrind**
- **GDB Tool: This type of tool is used in Unix programming. GDB is pre-installed in all Linux systems if not, it is necessary to download the GCC compiler package.**
- **DDD Tool: DDD means Data Display Debugger, which is used to run a Graphic User Interface (GUI) in Unix systems.**
- **Eclipse: An IDE tool is the integration of an editor, build tool, debugger and other development tools. IDE is the most popular Eclipse tool. It works more efficiently when compared to the DDD, GDB and other tools.**
- **AppPuncher Debugger is used for debugging Rich Internet Applications**

- **AQtime debugger**
- **CA/EZ TEST is a CICS interactive test/debug software package**
- **CharmDebug is a Debugger for Charm++**
- **CodeView debugger**
- **DBG is a PHP Debugger and Profiler**
- **dbx debugger**
- **Distributed Debugging Tool (Allinea DDT)**
- **DDTLite — Allinea DDTLite for Visual Studio 2008**
- **DEBUG is the built-in debugger of DOS and Microsoft Windows**
- **Debugger for MySQL**
- **Opera Dragonfly**
- **The dynamic debugging technique (DDT)**
- **Embedded System Debug Plug-in is used for Eclipse**
- **FusionDebug**
- **Debugger OpenGL, OpenGL ES, and OpenCL Debugger and Profiler. For Windows, Linux, Mac OS X, and iPhone**
- **GNU Debugger (GDB), GNU Binutils**
- **Intel Debugger (IDB)**
- **The system is used as circuit debugger for Embedded Systems**
- **Interactive Disassembler (IDA Pro)**
- **Java Platform Debugger Architecture source Java debugger**
- **LLDB**
- **MacsBug**
- **IBM Rational Purify**
- **TRACE32 is circuit debugger for Embedded Systems**
- **VB Watch Debugger — debugger for Visual Basic 6.0**

- **Microsoft Visual Studio Debugger**
- **WinDbg**
- **Xdebug — PHP debugger and profiler**