



Software Debugging

Dr. Sangharatna Godbole
Assistant Professor
Department of CSE
NIT Warangal

Debugging

- Once errors are identified:
 - it is necessary to identify the precise location of the errors and to fix them.
- Each debugging approach has its own advantages and disadvantages:
 - each is useful in appropriate circumstances.

Brute-force method

- This is the most common method of debugging:
 - least efficient method.
 - program is loaded with print statements
 - print the intermediate values
 - hope that some of printed values will help identify the error.

Symbolic Debugger

- Brute force approach becomes more systematic:
 - with the use of a symbolic debugger,
 - symbolic debuggers get their name for historical reasons
 - early debuggers let you only see values from a program dump:
 - determine which variable it corresponds to.

Symbolic Debugger

- Using a symbolic debugger:
 - values of different variables can be easily checked and modified
 - single stepping to execute one instruction at a time
 - **break points** and **watch points** can be set to test the values of variables.

Backtracking

- This is a fairly common approach.
- Beginning at the statement where an error symptom has been observed:
 - source code is traced backwards until the error is discovered.

Example

```
int main( ){  
    int i, j, s;  
    i=1;  
    while(i<=10){  
        s=s+i;  
        i++;  
        j=j++;}  
    printf("%d",s);  
}
```

Backtracking

- Unfortunately, as the number of source lines to be traced back increases,
 - the number of potential backward paths increases
 - becomes unmanageably large for complex programs.

Cause-elimination method

- In this method, once a failure is observed, the symptoms of the failure (e.g. certain variable is having a negative value though it should be positive) are noted.
- Determine a list of causes:
 - which could possibly have contributed to the error symptom.
 - tests are conducted to eliminate each.
- A related technique of identifying errors by examining error symptoms:
 - software fault tree analysis.

Program Slicing

- This technique is similar to back tracking.
- However, the search space is reduced by defining slices.
- A slice is defined for a particular variable at a particular statement:
 - set of source lines preceding this statement which can influence the value of the variable.

Program Slicing cont ...

- Slice of a program w.r.t. program point **p** and variable **x**:
 - All statements and predicates that might affect the value of **x** at point **p**.
- $\langle \mathbf{p}, \mathbf{x} \rangle$ known as slicing criterion.

Example

An Example Program

& its slice w.r.t. $\langle 9, i \rangle$

```
1.  int main( ){  
2.  int i, s;  
3.  i=1;  
4.  s=1;  
5.  while(i<=10){  
6.      s=s+i;  
7.      i++;}  
8.  printf("%d",s);  
9.  printf("%d",i);  
10. }
```

Types of Slices

Static Slice: Statements that may affect value of a variable at a program point for *all possible executions*.

Dynamic Slice: Statements that actually affect value of a variable at a program point for *that particular execution*.

Backward Slice: Statements that *might have* affected the variable at a program point.

Forward Slice: Statements that *might be*



Applications of Slicing

- Debugging
- Program understanding
- Testing
- Software maintenance
- Complexity measurement
- Program integration
- Reverse engineering
- Software reuse

Debugging Guidelines

- Debugging usually requires a thorough understanding of the program design.
- Debugging may sometimes require full redesign of the system.
- A common mistake novice programmers often make:
 - not fixing the error but the error symptoms.

Debugging Guidelines

- Be aware of the possibility:
 - an error correction may introduce new errors.
- After every round of error-fixing:
 - regression testing must be carried out.

Program Analysis Tools

- An automated tool:
 - takes program source code as input
 - produces reports regarding several important characteristics of the program,
 - such as size, complexity, adequacy of commenting, adherence to programming standards, etc.

Program Analysis Tools

- Some program analysis tools:
 - produce reports regarding the adequacy of the test cases.
- There are essentially two categories of program analysis tools:
 - Static analysis tools
 - Dynamic analysis tools

Static Analysis Tools

- Static analysis tools:
 - Assess properties of a program without executing it.
 - Analyze the source code
 - provide analytical conclusions.

Static Analysis Tools

- Whether coding standards have been adhered to?
 - Commenting is adequate?
- Programming errors such as:
 - uninitialized variables
 - mismatch between actual and formal parameters.
 - Variables declared but never used, etc.

Static Analysis Tools

- Code walk through and inspection can also be considered as static analysis methods:
 - however, the term static program analysis is generally used for automated analysis tools.

Dynamic Analysis Tools

- Dynamic program analysis tools require the program to be executed:
 - its behaviour recorded.
 - Produce reports such as, extent of coverage achieved, adequacy of test cases, etc.



Thank You