Course: Compiler Construction

Week 13: Error Handling & Recovery

Lecturer: Martha Gichuki

Learning outcomes Week 13: Error Handling & Recovery

At the end of the lecture, you will be able to:

i. Describe different types of errors
ii. Describe the process of Error Handling in compiler design
iii. Describe Error Recovery strategies

1. Introduction:

- ✓ An Error is a blank entry in the symbol table.
- ✓ Errors in the program should be checked, detected, reported and handled by the parser.
- ✓ When errors occur, the parser handles them and continues to parse the rest of the input.
- ✓ Errors may occur at various stages of the compilation process.
- ✓ There are many types or sources of errors that can occur in a program such as logic, run-time and compile-time errors[1].

A. Run-time errors

- ✓ These are errors that take place during program execution
- ✓ They usually occur due to adverse system parameters or invalid input data.

Two examples of run-time errors are:

i. Lack of sufficient memory to run an application or a memory conflict with another program
ii. Logical errors that occur when executed code does not produce the expected result. Logic errors are best handled by meticulous program debugging[2].

[1] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 219
[2] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 220

B. Compile-time errors

- ✓ They occur at compile-time, before program execution

Examples of compile-time errors[3]

a) Lexical errors

- ✓ They are caused by misspellings of identifiers, keywords or operators These are mainly *spelling mistakes* and *accidental insertion of foreign characters*.
- ✓ They include misspellings of identifiers, keywords, or operators and missing quotes around text intended as a string.
- ✓ They are detected by the lexical analyzer[4].

b) Logical errors

- ✓ These errors occur when programs operate incorrectly but still do not terminate abnormally (or crash).
- ✓ Unexpected or undesired outputs or other behaviour may result from a logic error, even if it is not immediately recognized as such[5].
- ✓ There is no automatic way of detecting logical errors
- ✓ Proper use of debugging tools may help programmers identify them.
- ✓ ***Examples of logical errors are*:**

i. Infinite loop(s),

ii. Unreachable code,

iii. Incorrect reasoning on the part of the programmer, e.g. use of the assignment operator = in a C program instead of the comparison operator = =. The program containing = may be well formed; however, it may not reflect the programmer's intent[6]

[3] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 219
[4] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 139
[5] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 220
[6] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 219

c) Syntax (Syntactical) errors

- ✓ These are mainly grammatical mistakes such as missing semicolon, unbalanced parenthesis or missing file reference which prevents the program from compiling successfully.
- ✓ They are the most common types of errors in a program
- ✓ They are detected by the parser and they include:

i. ill-formed constructs.

ii. misplaced semicolons

iii. extra or missing braces (unbalanced parenthesis in an expression)

iv. appearance of a case statement without an enclosing switch (in C or Java) - however, the parser allows this situation until later in the processing, when the compiler attempts to generate code and the error is then detected[7]

d) Semantic errors

- ✓ These are errors due to undefined variables, incompatible operands to an operator, incompatible value assignment or type mismatches between operators and operands etc.
- ✓ Type mismatches between operators and operands example is the return of a value in a Java method with result type void.
- ✓ They are detected by introducing some extra checks during parsing[8].
- ✓ Examples of semantics errors that the semantic analyzer is expected to recognize are: -

i. Type mismatch
ii. Undeclared variables
iii. Reserved identifier misuse
iv. Multiple declaration of a variable in a scope.
v. Accessing an out-of-scope variable.
vi. Actual and formal parameter mismatch[9].

[7] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 219
[8] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 219
[9] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 221

2. Error Detection and Reporting

- ✓ *Viable-prefix* is the property of a parser that allows early detection of syntax errors.
- ✓ The goal is to detect an error as soon as possible without further consuming unnecessary input
- ✓ The methodology used is to detect an error as soon as the *prefix of the input does not match* a *prefix of any string in the language.*
- ✓ Example: for(;), - reports an error for having two semicolons inside braces[10].

2.1. Error Handling

- ✓ Error handling is one of the most important features of any modern compiler.
- ✓ Error handlers address two challenges :-
    i. To have a good guess of possible mistakes that programmers can make
    ii. To come up with strategies to point out these errors to the user in a very clear and unambiguous manner[11].

There are three important functions of Error Handlers:

a) Error Detection
b) Error Reporting to the user
c) Implementing some error recovery strategy to handle the error.

- ✓ Error handling process should not slow down the processing time of the program[12]
- ✓ Good error handling is not easy to achieve but generally, a good error handler should:
    i. accurately and clearly report errors
    ii. recover from errors quickly
    iii. not slow down compilation of valid code[13]

[10] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 530
[11] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 230
[12] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 230
[13] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 230

2.2. Advantages of Error Handling in Compiler Design

i. Robustness

- ✓ Permits the compiler to detect, report and recover smoothly from errors as other processes continue

ii. Locating errors

- ✓ Compilers can recognize and distinguish source code errors such as syntactic, semantic or type errors which cause programs to function abnormally or generate incorrect output[14].

iii. Clear error reporting

- ✓ Compilers clearly report the nature and location of errors, enabling users to effectively fix the issues hence saving time in troubleshooting systems.

iv. Error recovery

- ✓ Compilers recover from errors and continue aggregating more through different methods like error adjustment, error synchronization, and resynchronization.
- ✓ This ensures that processes are not ended abruptly[15]

v. Incremental error gathering

- ✓ Gradual error accumulation enables compilers to execute correct program segments even if other segments contain errors.
- ✓ This is useful for large scope projects, as it enables engineers to test and investigate modules without recompiling the whole code.

vi. Improved Productivity

- ✓ Error handling enables the compiler to reduce the time and exertion spent on troubleshooting and error fixing.
- ✓ Accurate error detection, reporting and recovery assists programmers with rapid recognition and resolving issues, for efficient development cycles[16]

vii. Language Dependability

- ✓ Error handling is a fundamental part of language plan and advancement.
- ✓ To ensure there is a guarantee of the general language dependability and consistency, error handling is consolidated with systems in the compiler[17].

[14] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 222
[15] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 223
[16] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 220
[17] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 220

2.2.1. Disadvantages of error handling in compiler design:

i. Increased complexity

- ✓ Error handling in compiler design can increase the complexity of the compiler making it *more challenging to develop, test, and maintain*.
- ✓ Complex error handling mechanisms are *difficulty to monitor* i.e. ensuring they work correctly and are able to find and fix errors is cumbersome[18].

ii. Reduced performance

- ✓ Error handling in compiler design may impact the performance of the compiler more so if the error handling mechanism is *time-consuming and computationally intensive*.
- ✓ As a result, the compiler may take longer to compile programs and may require more resources to operate[19].

iii. Increased development time

- ✓ Developing an effective error handling mechanism can be *a time-consuming process* since it requires significant testing and debugging to ensure that it works as intended.
- ✓ This can slow down the development process and result in longer development processes.

iv. Difficulty in error detection

- ✓ While error handling is designed to identify and handle errors in the source code, it can also make it more difficult to detect errors.
- ✓ This is because the error handling mechanism may *mask some errors*, making it harder to identify them.
- ✓ Additionally, if the error handling mechanism is not working correctly, it may fail to detect errors altogether[20].

3. Error Recovery

- ✓ The basic requirement for the compiler is to simply stop and issue a message, and stop the compilation process.
- ✓ There are some common recovery methods and not all strategies are supported by all parser generators[21]

[18] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 221
[19] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 221
[20] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 221
[21] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 220

3.1. Error Recovery Strategies

✓ There are four common error-recovery strategies that can be implemented in the parser to deal with errors in the code[22]. These are: -

A. Panic Mode: -

✓ Upon encountering an error anywhere in a statement, a parser ignores the rest of the statement by not processing input from erroneous input.

✓ It is the easiest strategy and it prevents the parser from developing infinite loops.

Advantages

✓ It's easy to use.

✓ The program never falls into the infinite loops.

Drawback

❑ The method may lead to semantic or runtime errors in subsequent stages[23].

B. Statement Mode: -

✓ When a parser encounters an error, it tries to take corrective measures so that the rest of the statement inputs allow the parser to parse ahead.

✓ Example: - inserting a missing semicolon, replacing comma with a semicolon, etc.

Advantage

✓ Used in many errors repairing compilers.

Drawback

❑ A wrong replacement may cause the program to fall into an infinite loop. Parser designers should prevent such wrong corrections from happening[24].

[22] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 220
[23] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 220
[24] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 221

C. Error Productions: -

- ✓ Common errors that may occur in the code are known to the compiler designers and therefore designers create augmented grammar to be used, as productions that generate erroneous constructs when these errors are encountered.

Example: Write 5 x instead of 5 * x

Add the production E → … | E E

Advantage

- ✓ Generally, syntactic phase errors are recovered by error productions.

Drawback

- ❑ A difficulty method to maintain since change in the grammar necessitates a change of the corresponding production(s)[25] .

D. Global Correction: -

- ✓ The parser considers the program in hand as a whole, checks what the program is intended to do and tries to find a closest match for it, which is error-free.
- ✓ When an erroneous input (statement) X is fed, it creates a parse tree for some closest error-free statement Y. This may allow the parser to make minimal changes in the source code

Advantage

- ✓ It makes very few changes in processing an incorrect input string.

Drawback

- ❑ It is simply a theoretical concept, which due to the complexity (time and space), it has not been implemented in practice yet[26].

[25] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 221
[26] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 221

3.2. Symbol Tables and Error Recovery

- ✓ Semantic errors are recovered by using a symbol table for the corresponding identifier
- ✓ If data types of two operands are not compatible, the compiler automatically performs type conversion.

Advantage

- ✓ It allows basic type conversion, which is generally done in real-life calculations.

Drawback

- ❑ The only possible conversion is implicit type conversion[27].

Content Covered in Week 13: Error Handling & Recovery

At the end of the lecture, we were able to:

i. Describe different types of errors
ii. Describe the process of Error Handling in compiler design
iii. Describe Error Recovery strategies

Course Text Books

1. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; *Addison- Wesley Pub Co*, ISBN: 0201100886 (2007))
2. Compiler Construction: Principles and practices; Kenneth C. Louden; *Cengage Learning*; 1st edition, ISBN-10 : 0534939724 (1997)
3. Basics of Compiler Design: Torben Mogensen; *DIKU University of Copenhagen Universitetsparken 1 DK-2100 Copenhagen DENMARK* (2007)
4. Engineering a Compiler: 2nd edition; Keith D. Cooper and Linda Torczon; *Morgan Kaufmann Publishers* ISBN: 978-0-12-088478-0 (2003)
5. Compiler Design: Santanu Chattopadhyay; *PHI Learning Publishers*, ISBN 812032725X, 9788120327252 (2005)

[27] Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 221