Errors in C

Errors are simply unavoidable when you develop a program, let's learn more about them!

Learning to code can be a frustrating endeavor because you are destined to encounter many red errors along the way. What makes a programmer successful isn't avoiding errors — no programmer can avoid them. Great programmers understand that errors are part of the process, and they know how to find the solution to each while learning something new from them. In this article, we'll teach you how to think about errors in your code a little differently.

There are many ways of classifying errors. For example:

- **Compile-time errors:** Errors found by the compiler. We can further classify compile-time errors based on which language rules they violate, for example:
    - Syntax errors
    - Semantic errors
- **Link-time errors:** Errors found by the linker when it is trying to combine object files into an executable program.
- **Run-time errors:** Errors found by checks in a running program.
- **Logic errors:** Errors found by the programmer looking for the causes of erroneous results.

As you recall, the programming process looks like:

The errors above are described as follows:

Compile-time errors

When you are writing C programs, your compiler is your first line of defense against errors. Here are the two types of compile-time errors:

Syntax errors

These errors occur when an invalid statement is written.

```
int x = 6    // Error: missing a terminating semicolon
Int x = 6;   // Error: Int is not a type
printf(“Error”;   // Error: missing closing parenthesis
```

Semantic errors

These statements are syntactically valid but don’t do what the programmer intends.

```
a + b = c;   // Error: value required as left operand of assignment

int i;
i = i + 2;   // Error: use of a un-initialized variable
```

Link-time errors (or Linker errors)

If your program has no compile-time errors, these link-time errors will occur when an attempt is made to actually compile. These appear when a program is missing proper prototypes, has incorrect header files, or accidentally uses **Main()** instead of **main()**.

Run-time errors

If your program has no compile-time errors and no link-time errors, it’ll *run*. Run-time errors appear during this time and can manifest in a handful of unique ways, one most common way being a Division error when dividing by zero. These types of errors can be difficult to find due to them being syntactically right and missed by the compiler.

Logic errors

Once we have removed the initial compile-time errors, link-time errors, and run-time errors, the program finally runs without hiccups. However, sometimes no output is produced or the output the program produces is just wrong. This can occur for a number of reasons. Maybe your understanding of the underlying program is flawed; maybe you didn’t write what you thought you wrote or certain lines executed in unexpected ways. These are logic errors, aptly named after errors that could arise in a coder’s logic inside the program.

That’s all of them!

Avoiding errors may sometimes feel impossible, but learning from them and understanding the different types can all be as part of the coding process as the coding itself. So remember what errors mean, how they appear, how to solve them, and keep programming!