[0710] Prehistoric Error Handling

In the primitive days before the dawn of OO programming, programmers constantly ran into the following problem:

They would write the most perfect programs - programs that were clear, understandable, and well-documented. Programs that ran well, worked correctly and made total sense. In short, perfect programs! Until...

Until those lousy testers started testing. They started doing things like testing in low, low memory conditions. Or testing with faulty hard disks. Or testing with slow, slow network connections.

And they started finding bugs in these "perfect" programs! The programs were not handling these problematic conditions well at all. So the testers called "the managers" and told them about the problems. And the managers got... concerned. And the concerned managers went and talked to the programmers and told the programmers to fix the problems. And the programmers were not happy.

You see, in order to fix the problems, the programmers had to change their "perfect" program so that it respected and paid attention to things called "return codes."

Back in those days, when you called a system function, you'd get back an integer that would tell you if the function worked of not. What that meant was, if you wanted your program to be 100% bulletproof, you had to check the return code every time you called a system function. Lots of times you had to check for several different return codes and so, after each system call, you'd have to add an "if ()" statement or "if/else" ladder or possibly even a "switch()" statement.

Just think what that did to the program's code. The code - which used to be "perfect" and readable - was now forced to become bigger, slower, and much harder to read. NOOOOOOOOO!!!!!!!

And then, of course, there was the whole issue of what to do if/when your system functions started failing. Chances are there wasn't much you could do. It was all... so... primitive!