Ah yes, indeed. The old programmer sucked contemplatively on his pipe, waving the bowl dangerously close to my ear. *In my day you had to watch for them, the little buggers, they'd have your hand off as soon as look at you.*

He was talking about the elusive wild pointer in its many guises.

In the beginning there was C, a stupid language that had its place in the pantheon but it was basically jumped up assembler aimed at some abstract machine. The promise … he coughed alarmingly for a while, waving fragrant smoke across the room. Now there was no cancer a lot of people had started to smoke again – it was a sign of affluence and power instead of stupidity. He watched the colours of the spots dancing in front of his eyes, with a rheumy intensity.

The promise that the code would run anywhere if you recompiled it because it was all aimed at the same simple minded abstract machine.

I interjected – *this was true wasn't it?*

He stopped and regarded me over his meerschaum, the smoke falling down over the rim – *only like it is true that all books are made of paper. People would do things like attempt to go back to the beginning of a file that was a terminal session, they would write code that couldn't be bothered to check for errors and the language didn't have any error handing except for some junk added involving global variables that you had to keep testing and no-one could be bothered. No exceptions, and, of course the curse of programming, the pointer.*

You could recompile something and it would work. But once the number of lines of code (not including comments if you were lucky enough to have some) got over about two hundred it would start to fall to bits. Then you had to be disciplined. The language was so awful that people invented naming conventions for variables to check for type collisions – imagine – programmers checking because the compiler could not! People doing mental drudge work – I mean, what the fuck are computers for? Everything is a sixteen bit integer (this was a long time ago) except when it really matters and then your program won't work and you can't find out why. Don't forget, either, that before this

abortion there had already been fifteen years of research, LISP machines existed with fully integrated, extensible programming environments – all the things you take for granted were already there in essence.

So we went back to the Stone Age and lost those fifteen years. C was responsible for Unix, like Unix it was obsessed with simplicity and being clever at the same time. So we had line-mode text editors with substitution commands that were incredibly powerful. We had a replaceable command shell that did weird things like expand all of the command line arguments for you so you could never check things like ***did you really mean to delete all files in this directory?*** *The call, from the C and Unix crowd was all about being simple.*

My eyes started to glaze over, I had heard the Master on this topic before, *simplicity is right but things must be* ***complete*** *also or you are wasting your time.*

He flicked me on the nose, and the pipe got rather too near for comfort. *Listen, and I will initiate you into another great mystery :* ***simple is not correct but it can give the veneer of correct. Simple is very very dangerous in the wrong hands.***

His pipe had gone out, thank COBOL, but he still had the wit to relight the damn thing.

Simple is where the wild pointer comes from. He sucked industriously and the flame went up and down with his eyebrows. *It's easier to implement a language if you don't have to create things like strings as a proper type, instead have a pointer to a place in memory that contains the strings and write a couple of libraries if you feel generous that will allow you to do things like compare them and copy them (copying came later after everyone had written their own bug filled routine). In the same vein an array and a pointer are the same thing, a string is an array of characters, and everything is a sixteen bit integer anyway. Just for laughs make the library routines have completely unmemorable names and slightly different, but very simple, conventions. Multidimensional arrays are arrays of pointers; so you've got pointers to pointers ad infinitum. Then add in the fun-filled idea that a comma is an operator so an inexperienced Pascal programmer will not use a[0][1] but a[0,1], which is valid but not what was intended – who knows what it means? I can't remember. Oh*

yes, arrays start from 0 too. All this is post Pascal, which was a good attempt to make life easier for the programmer and simply didn't have all of these weaknesses, but it was meant for teaching and didn't have the low-level stuff in a standard form.

But the consequences of such glorious simplicity, grasshopper, the consequences are bad for all momma's li'l chil'n. Unless you are a black belt computer science type you won't understand half of this and not have the discipline necessary to make it go.

So what happens, master? The pipe was out and I was saying nothing – he will insist on smoking toffee tobacco, which smells of burnt sugar.

*People write simple programs that **work**. But they aren't **correct**, they don't check for errors because the language doesn't help you do this easily and it isn't part of the culture. Some compilers will initialise automatic variables and some won't. Code will go across machines from different vendors without any problem. But it is still incorrect.*

The wet end of the pipe came out of his mouth.

*Then you fill up a file system and the machine crashes badly and destroys a load of work, and no-one knows why. Then you have a pointer you allocated and deallocated memory to without setting back to the empty value that pisses all over something you wanted to keep, but only sometimes, because you needed to reinitialise it but had no way of finding this out. Then you have a situation where your program fails, even when it seems OK, and it can be fixed by you changing the order you declare your variables. But is it then correct? **Do you trust it?***

Underneath everything is an integer that is cast into different roles depending upon what it is supposed to do. This means that everything is nothing, everyone has to think like the fucking compiler. How many people are that good? Not many, and even the good ones need time to make their mistakes and learn. Even the good ones can be wrong sometimes. Every crappy little four line routine manipulating strings needs to be thoroughly tested and you know in your heart that someone already wrote the same routine a thousand times.

Then we get to Microsoft, who used C++ to write their stuff after a while. Every pointer was declared to be ***far Pascal****, because of the restrictions on the processors – why Pascal? Why force us to keep declaring pointers far and near? It became a kind of religion.*

Is it any wonder software is so expensive to write?