The Linguistics of Forth

Recently, I introduced Forth to a computer science student that is taking linguistics as subsidiary subject. It became clear to me that linguists should love Forth for two reasons that have substantial collateral benefits:

- Its Stack based nature.
- Its simple Parser.

The Stack advantage

Programming an explicitly Stack based machine is considerably different to the vast majority of existing programming languages. It needs a different way of thinking. In the past we could always refer to the "HP-calculators" to familiarise engineers with Forth. (Note: Unfortunately, HP has given up on RPN. It is still there but only as an obscure operating mode for aged engineers.)

From a linguistic point of view this has dramatic advantages for a programming language.

Because input and output arguments are handled by the Stack, Forth words do not need parameter lists. This changes the programming style substantially, because you can put several words on a single line. This could be called "horizontal programming style".

Conventional programming languages clutter the code with parameter lists, which severely hamper the readability of the code. Usually, only one procedure call followed by its parameter list is put on a single line.

First benefit:

Given Forth's horizontal programming capability we can compose phrases and sentences. And we can put our ambition into writing "readable" code that can be understood by system engineers on its highest levels resulting in more reliable code.

Second benefit:

Horizontal programming puts more code on a single screen. Therefore, you do not have to scroll nearly as often as in other languages. That is a clear debugging advantage.

The Parser advantage

Most of the time, Forth's lexical scanner only looks out for whitespace. As a consequence, any special character may be used to compose a name. This opens up a whole new dimension for the signification of names compared to most other programming languages. Those have a rather limited inventory of "valid" characters that may be used.

Therefore, Forth's source code includes many more significant spaces compared to other languages. This makes Forth code more readable, because "reading" Forth is akin to reading a book.

And because of its simplicity, the Forth lexical scanner can do without regular expressions processing.

I asked myself, why most programming languages are so neglectant of the syntactical role of spaces. This came to my mind:

The first programming languages (FORTRAN, COBOL) appeared at a time when the source code had to be punched into cards. Every single character was costly. Omitting "unnecessary" spaces was used as a simple means for compression. Apparently, more recent programming languages upheld this as a tradition, whose justification had withered away decades ago.