

RANDOM COMMENTS ON WORD-NAMING

Nathaniel Grossman
Department of Mathematics
UCLA
Los Angeles, CA 90024

How should one select names for newly-defined Forth words? Devotees of *Forth Dimensions* will recall many letters and articles whose authors grappled with this vexing problem. Those Forth writers whose needs and goals point toward compact code opt for short names, unambiguous but often hard to remember. Others whose interests lean toward teaching as well as those who merely want to polish programming skills by studying published Forth programs prefer long, descriptive names. In between the two extremes stretches a continuum of possibilities, no one of which is best in every circumstance.

My own preference lies with the long, descriptive names. As a teacher called upon to explain difficult definitions and lengthy proofs, I have learned to value well-chosen names for new notions. In the articles I have published in *Forth Dimensions*, descriptive word-names appear by choice. They can, if well-chosen, decrease or even eliminate the need for comments in the source code, producing cleaner and more easily readable screens and diminishing the legendary difficulties of following or debugging a Forth program. One who programs in the descriptive style always has the option of later replacing the long names by shorter aliases.

Even if the short word-names seem clear to their coiner and instantly convey to him by familiarity their meaning and usage, this convenience is not always apparent to others. How often have you tried to decipher a Forth program whose word-names serve not to clarify but to deepen the enigma? All too often, programs are published with word-names that are no more helpful than if they were chosen at random.

Whether or not it is explicitly stated, one of the chief purposes of *Forth Dimensions* must be to serve as a tutor and a source of model Forth programs. True, the Forth-83 Standard puts teachability in eleventh place -- last -- in its list of tradeoffs, but *Forth Dimensions* has its own role to play in the Forth community.

All of this brings me to a little demonstration to illustrate the previous remarks. Is there really a difference between descriptive code and "random" code? I think there is a substantial difference and the seven screens accompanying this text can show the difference.

Without claiming them to be paragons, I offer screens #1 to #3 as examples of descriptive programming. Their goal is the generation of random words, and that goal is attained by the words `RANDOM_WORD_COINER` and `RANDOM_WORD_LISTER`. Use of the words is explained in screen #0. I believe that the screens #1 to #3 are essentially self-commenting because of the choice of descriptive word-names. (Yes, I do have short aliases for those long words tucked away in the suppressed screen #4. I am a two-finger typist and I find the short aliases indispensable at debugging time.)

To illustrate the use of screens #1 to #3, I have used them to convert themselves to randomized word-names. The results are presented as screens #5 to #7. The names were generated with random lengths of five to twelve symbols. The job of conversion was made easy by the file system and editor utilities supplied with MicroMotion MasterFORTH, the implementation of Forth-83 that I use. After copying screens #1 to #3 into screens #5 to #7 of the working file, I

generated the random words and inserted them one-by-one with the search-and-replace utility to replace each of the descriptive words that I had coined. The whole job took just a few minutes.

Observe that the twin programs in screens #1 to #3 and screens #5 to #7 will be equally easy to trace and comprehend for a reader whose knowledge of English is sufficient only to allow transcription of Forth Standard words. However, most Forth users can be expected to possess at least a reading knowledge of elementary English, and it is a simple courtesy to them to publish descriptive, readable screens. I have no doubt that casual browsers and hopeful beginners will then be more encouraged to plunge deeper into Forth studies.

Henry Laxen wrote in *Forth Dimensions* V/6 about what he perceived as the devious ways of mathematicians. Who is to say that he is wrong? Long ago, St. Augustine issued a stern warning to

...beware the mathematician and all those who make empty promises.

*The danger already exists that the mathematicians have made a covenant
with the devil to darken the spirit and to confine man in the bonds of hell.*

Certainly a common and often-deserved criticism of mathematicians is that they are prone to clothing simple notions in arcane garb. But why should mathematicians have all the fun? Write your next Forth program in easy-going, descriptive style; amaze yourself by debugging it painlessly and, one month later, reading through it effortlessly. Then use the program in screens #1 to #3 to randomize its word-names, after which you can turn your metamorphosed creation loose upon the (Forth) world. I'm giving you in figure one a short list of random words to start you off. Feel free to use them.

```
SCR # 0
0 Random word generator                                FORTH 83 12AUG84NG
1
2 This program contains words to print words of random length
3 whose letters are randomly chosen from the printable ASCII
4 characters. Load screens #1 through #3. ( If your implement-
5 ation includes INCLUDE, use <filename> INCLUDE. Otherwise,
6 1 LOAD will load the necessary screens.)
7 <n1> <n2> RANDOM_WORD_COINER will print a word of random
8 length containing between n1 and n2 ( inclusive) randomly chosen
9 ASCII symbols. ( Be sure that 0 < n1 =< n2.)
10 <n1> <n2> <n3> RANDOM_WORD_LISTER will print out a list of
11 n1 words, each of random length containing between n2 and n3
12 symbols as above.
13 Implementations of RANDOM can be found in Brodie's STARTING
14 FORTH, Anderson and Tracy's FORTH Tools, Knuth's treatise, etc.
15 <n> RANDOM returns a random integer between 0 and n-1 inclusive.
```

