Rhetorical Analysis of ‘learn C in Y minutes’

Daman Morris

In the field of software engineering, and computer science in general, documentation of all aspects of a software system is a common task. The production of documentation for a particular software package, language, API, etc., is an instance of technical writing and can thus be subject to a rhetorical analysis. Herein I make such an analysis of the popular documentation series for programming languages, “learn X in Y minutes”. These consist of short introductions to a programming language (X) designed to be consumed in a relatively short period of time (Y minutes). The documents themselves consist primarily of a snippet of real code in the target language, with comments interjected to explain various features. I will focus on one particular document, that which gives an introduction to the C programming language: “learn C in Y minutes”[1].

Due to the short nature of the document (and of all the documents in this series), the primary goal is to give a quick tour of the C language to relatively-experienced programmers; a complete novice would gain little to no understanding from this document alone, since it assumes a basic understanding of programming languages in general. For example, it uses a variety of domain-specific vocabulary such as “array”, “pointer”, etc., without defining them in advance; it is assumed that the reader has advance knowledge of some of the technical vocabulary required to understand the text. As such, the document fulfills the exigent needs of moderate- to highly-skilled programmers looking to learn more about C.

This document is a collaborative effort, but it was originally written by Adam Bard, a software developer. Since Bard and the other authors are also moderate- to highly-skilled programmers, the writer and the audience share a significant set of skills; the difference is primarily in foreknowledge of the C programming language in particular. The writers’ relationship to their audience is thus one of peership, as anyone reading this document for its intended purpose will be a fellow programmer. Because the audience already has significant background knowledge, they are likely to value succinctness, and this is reflected in the overall structure and style of the document, which, as the title suggests, is designed to be read in a short amount of minutes (the original precursors to this document series had explicit times and languages in the title, e.g., “learn C in 5 minutes”, “learn Python in 12 minutes”, etc.).

Because the bulk of the document consists of actual code, there isn’t much in the way of layout that lends itself to meaningful analysis, so I will simply make a few brief notes here. First, the page as a whole is kept very minimal; this is probably deliberate, as there is a certain amount of prestige in the intended community to be had by keeping webpages as close to “bare HTML” as possible. The entire page is centered, with a title and brief introduction at the top and a following code block, which takes up the majority of the page. The title and introduction are, again, minimally styled for effect. Finally, the code block has a darker background and `teletype font`, both common styles for this kind of content (the font being near-universal), and the code is highlighted with different colors to give hints on what certain portions of the code are doing — this is referred to as “syntax highlighting”, and it is again near-universal in typesetting this kind of text.

It is difficult to concisely encapsulate the overall genre this document falls into; an attempt might be “technical introductory material”. It must be noted, however, that the genre can manifest in many different formats — from blog posts to books, and everything in-between. This particular document is more stable than a blog post, but it can be (and has been) edited over time and by multiple people, making it relatively less stable than a published book. That being said, this document is an excellent example of one of the most important conventions of the genre: examples, generally in the form of short code snippets. In fact, the entire document is a code snippet, with explanatory English prose embedded in code, rather than the other way around. It is thus an example of the so-called “literate programming” style, wherein code and documentation are merged in a single document, with documentation being

[1]https://learnxinyminutes.com/docs/c

provided by comments alongside the code. The popularity of literate programming has waxed and waned over time, which is an important contextual factor for understanding the document; at the moment, it is not particularly popular (outside of certain circles). At the same time, this series is itself fairly popular, so a reader is liable to interpret it as the canonical example of this style (in fact it is actually rather unusual in that in most instances, a literate program is designed to be turned into an executable program, whereas this document would not do anything useful if it were compiled, and isn't really intended to be).

The rhetorical style of the document uses primarily logos, with a bit of ethos and effectively no pathos. Credibility is built by appealing to a sense of community with the audience; as mentioned, the intended audience consists of fellow programmers and computer scientists. As such, the author makes heavy use of both technical terminology and "nerdy" language, as in Listing 1. This section contains many technical terms that the reader is assumed to know already ("array", "concrete size", "4-byte words", etc.) as well as some language designed to give the document a more formal/technical feel ("thusly" etc.).

Listing 1: A section explaining arrays.

```
// Arrays must be initialized with a concrete size.
char my_char_array[20]; // This array occupies 1 * 20 = 20 bytes
int my_int_array[20]; // This array occupies 4 * 20 = 80 bytes
// (assuming 4-byte words)

// You can initialize an array to 0 thusly:
char my_array[20] = {0};
// where the "{0}" part is called an "array initializer".
// NOTE that you get away without explicitly declaring the size of the array,
// IF you initialize the array on the same line. So, the following declaration
// is equivalent:
char my_array[] = {0};
// BUT, then you have to evaluate the size of the array at run-time, like this:
size_t my_array_size = sizeof(my_array) / sizeof(my_array[0]);
```

Because the entire document is an executable code snippet, it is a self-justifying example and an element of the logos used to persuade the reader that it represents a credible source of knowledge on the C language. Listing 2, on the other hand, provides a good example of how the author uses ethos — with the statement "C is not Python", the author is conveying a commonly-held belief in the programming community that Python is highly "magic" (meaning that certain things — like comparison chaining — are made "too easy" by the language), and since the reader is expected to agree, this increases the reader's perception of the author's expertise in the area. This snippet also has the author reasoning through an unexpected behavior of the first line (that it always evaluates to `true`) which represents an appeal to both logos and ethos.

Listing 2: An example of ethos

```
// C is not Python - comparisons don't chain.
// Warning: The line below will compile, but it means `(0 < a) < 2`.
// This expression is always true, because (0 < a) could be either 1 or 0.
// In this case it's 1, because (0 < 1).
int between_0_and_2 = 0 < a < 2;
// Instead use:
int between_0_and_2 = 0 < a && a < 2;
```

The tone of the piece is generally casual, but technical, as is common to these kinds of documents. However, because the document is produced collaboratively there are some sections where the tone differs from the overall tone of the document. For example, the prose in Listings 1 and 2 is fairly technical and formal, but in places the tone becomes more casual — later in the document one finds the lines "while loops exists" and "for C don't [*sic*] have bool as data type before C99 :(". The first uses a common expression "*thing* exists", which conveys a sense that this is an extremely obvious statement, but is fairly casual. The second, of course, has an old-style emoticon; this is probably the most informal the tone becomes in the whole piece, which differs markedly from some of the more formal portions.

Overall, the document is effective at giving a broad overview of the C programming languages (again, to already-experienced programmers). It covers most of the salient details, especially with regard to how C differs from other common programming languages. It makes effective use of logos in particular, but also ethos, to convey the information to the intended audience. However, because it has been edited by a group of people, it has some inconsistencies in tone that could be removed by revision.