FMT: 1980's text formatting to the extreme!

By Adam Bradford
CS5959
Spring 2014

This document explores the GNU utility "fmt" and its viability for use in our production environment.

Fmt is a utility that allows the user to input a text file and to reflow the text with certain options, such as width or indentation. Having looked at the source code, it is my recommendation that we continue our search for a text formatter to something more modern, and perhaps with a graphical user interface. We should look for somethings that our design and layout department would be comfortable with.

My decision is based on the following two reasons:

1. Unicode support
   Fmt does not have any built in functionality for Unicode text files, making it useless for any sort of modern text formatting in any localization that is not based on the Latin alphabet. This would alienate a large portion of the world.

2. Asserts
   The assert function was a favorite of one of my professors, and he stressed the importance of early termination of a program when impossible conditions exist. In the fmt.c file, the word assert appears exactly twice; once to include the header and once more to check that a word is no longer than the maximum word limit.

Although fmt is not a good choice for a modern company to use for text formatting, there are some good things about the implementation that show good software design and execution.

For example, there is an abundance of error checking throughout the source. Comments are descriptive and are written at a high level in order to better understand what it is a section of code is trying to accomplish.

Also, and this is unusual for an older c program, the variables are very well named as to describe what they contain. As an example here is a while condition for the "get_paragraph" function.

```
(c == '\n' || c == EOF ||
 next_prefix_indent < prefix_lead_space ||
```

in_column < next_prefix_indent + prefix_full_length)

This type of variable naming makes simple to see what conditions must be met for the loop to continue. In addition to the variable names, function names are also descriptive, have a verb, and accurately describe what it is that they do.

Continuing on the subject of functions, there are a lot of them. This is a good practice that would make it easy to debug the program if there was a problem. It's just a simple act of finding where the error is and stepping through the code, which is easy to read.

Fmt is a well written program, and completes the task it was made to do, but it does not fit text formatting needs for any workflow that requires Unicode, and also does not have any sort of visual interface. It is for these reasons I suggest broaden our search to more fully featured, modern programs.