

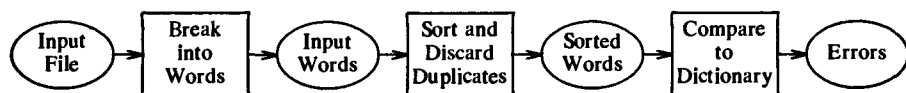
COLUMN 13: A SPELLING CHECKER

Spelling mistakes irritate readers. And for most writers, checking spelling is a boring and error-prone job. Fortunately, the problem is ideally suited for computers: dull, repetitive work that requires fast reading and a good memory.

In this column we'll study the design of the UNIX system's spelling checker `spell`. It's a beautiful and useful program, with a history rich in important lessons about program development.

13.1 A Simple Program

Steve Johnson wrote the first version of `spell` in an afternoon in 1975. His approach is straightforward: isolate the words in a document, sort them, and then compare the sorted list with the dictionary.



The output is a list of all words in the document that aren't in the dictionary. If you meant to write *stationary*, this process will catch the misspelling *stationiry* but not *stationery*, because the latter is a valid word.

Kernighan and Plauger reconstruct Johnson's program on page 133 of their *Software Tools in Pascal*. Their pipeline of programs can be paraphrased as

<code>prepare filename </code>	<code># remove formatting commands</code>
<code>translit A-Z a-z </code>	<code># map upper to lower case</code>
<code>translit ^a-z \n </code>	<code># remove punctuation</code>
<code>sort </code>	<code># put words in alphabetical order</code>
<code>unique </code>	<code># remove duplicate words</code>
<code>common -2 dict</code>	<code># report words not in dictionary</code>

The vertical bar connects the output of one program with the input to the next program. The input to the first program is filename, and the output of the last program is the list of (potentially) misspelled words.