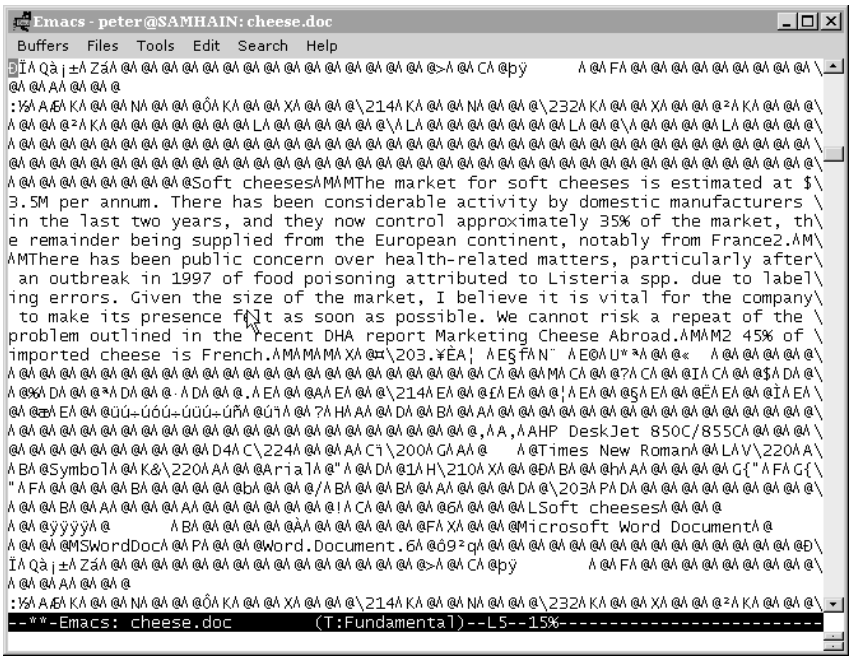

Figure 6. What you never see: the internal storage format of a word processor file, revealed by using a standard plaintext editor

1.5.1.3. Different uses for markup

By now it is becoming clear that if you wanted to search all your word processor documents for the chapter or section headings, or all references to scientific names, or mentions of other documents by title, or all phrases that were emphasized, or personal names, or product names (the list can go on and on) you couldn't just search for italics or bold, because your computer would have no way to distinguish the different uses: you'd just get all the other italics and bold coming out as well, and you'd have a huge manual task to disambiguate them.

The reason for this is that italics, bold, *etc* in themselves have no actual meaning: as we've seen, it's the conventions we associate with them that we are really looking for. Scholars call this 'semantics', the association of meaning with a symbol. All the typeface on its own says is, in effect, 'hello, I'm a different-looking style of type'; it then leaves it up to the reader to know already what that means.

While this is just fine for printing, it is not adequate if you want the computer to be able to use the meaning attached to a phrase, rather than simply the appearance. After all, the use of bold type and spacing for headings varies widely between users, designers, publishers, and

applications, and doesn't have any 'meaning' on its own anyway. What is important about the phrase 'Soft cheeses' above is that it is a *heading*, quite distinct from any other use of the same words in the sentence that follows it. While its appearance is critically important for presentation to a human reader, it is usually quite unimportant in terms of storage and recognition by computer.

So in addition to identifying behavior (like changes in appearance), markup can let you describe certain parts of the text according to their function or meaning. In the examples above, we used mostly ***visual markup***, because this is commonly used when printing is the only objective. However, there are lots of other things that people may want to do with your text apart from print it:

- display it on a screen (a bit like printing, but with printing you control the paper: you may not be able to control what kind of screen your reader has);
- archive it: preserve it over the passage of time, regardless of the system used to write it with;
- send it to someone else: transfer it to some other computer system;
- transfer it to a million other systems by putting it onto the World Wide Web;
- put it into a database or searching system so that people can refer to it and extract topics from it;
- repurpose it: a report can become a chapter of a manual; an article in the in-house magazine can become a story for the press; a research note can become part of a training system;
- print it again, but in a different style, perhaps in large type, or in Braille;
- turn it into a spoken recording, or a script for video use;
- analyze it for content, for research purposes.

To do all this kind of thing we need to use the markup to indicate the meanings, reasoning, or purpose behind the text, rather than just its appearance, and let the appearance be specified separately. This use of markup is called ***logical markup***, because it can be processed by computer logic (the terms 'visual' and 'logical' markup are from an article by Leslie Lamport[28]). Using logical markup you can call a heading a heading, a list a list, a paragraph a paragraph, and leave it to a style sheet to specify how each of those elements appears. By separating form from function, you can immediately make your text more usable, because the very uses to which it can be put are no longer cast in concrete.

In some applications, it is extremely important to impose a structure, to prescribe what can and cannot go where. Office reports, letters,

and memos; job applications; technical descriptions and specifications or manuals; books, periodicals, articles, and essays; all of them tend to use a structure where some pieces of information are essential (compulsory) and others are optional. ***Prescriptive markup*** defines what is *needed* and what is *permitted* (software is available which can help enforce compliance with prescriptive markup):

- my sister-in-law is a journalist: her articles *must* have a title, date, and byline (her name) on them, otherwise she won't get paid;
- letters or email messages *must* have the recipient's name and address, or they won't get delivered; they *ought* to have the date, usually a subject and at least one piece of textual content; they *may* have many more pieces of text of various kinds; and they *must* have the name and address of the sender;
- when you apply for a job, you *have to* give your name, address, your work experience, and a variety of other information, but some things are *optional*, like your hobbies.

There are also times when you want to indicate the existence of some special class of information, but it may or may not have a specific visual appearance. A good example is index keywords, place names, personal names, and other items readers want to be able to look up, but which just print as regular text. This is ***descriptive markup***: if you look up 'email' in the index, you'll find an entry referring to this page, but there's nothing special about the word: it has no special appearance, I just marked it for indexing.

Frequently you get descriptive and prescriptive markup, as well as visual and logical markup, working together: some pieces of information are essential for the immediate needs, others are there for future use.

1.5.1.4. Markup using SGML

SGML is a language which lets you define systems of markup to describe things in a way that a computer can use and reuse. It works on the principle that documents are typically made up of repeated occurrences of basic elements, like a house is built of many individual bricks, doors, windows, pieces of wood, fixtures, and fittings. For a house, the architect's plan describes what goes where; for an SGML document, the Document Type Definition (DTD) performs this task.

The basic elements for a building can be combined in different ways on different occasions to produce different kinds of structure called 'a house': for instance a bungalow, a town-house, a country mansion, or an inner-city tenement. A different combination, excluding some elements and including some new ones, is needed to produce instances