the supported languages, and so on. Through this process it would be possible to end up with macro definitions for `\m@nthjan` etc. which are tailored to the set of languages supported by the `.fmt` file.

Clearly all this requires some rather complicated macros to do the job. In addition these macros would not be terribly fast but they would be invoked only once for each set of macros with language dependent text. Here I think portability and flexibility are more important than efficiency! The suffixes `.eng` for english or `.grm` for german are supplied as an argument of `\deflanguage`. Unfortunately these suffixes should be restricted to three characters because some systems (DOS and VMS) allow only that much, otherwise the full names (e.g., `english`) would be preferable.

The precise form of the macro definitions manufactured by `\makelanguage` should be of little interest to author or user. They are somewhat analogous to the PASCAL and TEX code produced by the WEB system programs TANGLE and WEAVE. These macro definitions should, however, be optimized whenever several languages use the same text. In particular, if all languages use the same text (or there is only one language defined) the replacement text for the macro should be simply this text.

Sometimes a user wants a different text (for one or several languages) than what is supplied by a macro package with its files for language dependent texts (`abcdef.eng` and `abcdef.grm` in the example above). Without precise knowledge of the macro definitions constructed by `\makelanguage` this requires a macro `\changelanguage` which could be used, e.g., in the form

```
\changelanguage
  \somemacro{\somelanguage{...}...}
```

to change the replacement texts of `\somemacro` for the languages `\somelanguage, ...` .

4. Protection

Some of the macro definitions discussed above, in particular `\setlanguage`, certainly must be protected against expansion if they are, e.g., written to an external file. Here I would, however, propose a slight deviation from LaTeX's scheme. `\setlanguage` should be defined via

```
\global\defprotect\setlanguage\setl@ng
```

with the definitions

```
\def\defprotect#1#2{\def#1{\pr@tect#1#2}}
\def\doprotect{\let\pr@tect\d@protect}
\def\noprotect{\let\pr@tect\n@protect}
\def\n@protect#1{}
\def\d@protect#1#2{\noexpand#1}
\noprotect % normally, no protection
```

Thus the sequence

```
{\doprotect
 \immediate\write{\setlanguage}}
```

would write the string '`\setlanguage`' (not '`\setl@ng`') which would still be protected when read in and written again.

5. Summary

In the preceding I have discussed the design of a scheme to handle multiple languages in TEX 3. I have intentionally left out almost all details of how such a scheme can be implemented. (At present a preliminary version of all the required macros is being tested.) For the moment it seems more urgent to agree on a design (including the user interface) than on details of how such a design can be realized through macro definitions.

◇ Peter Breitenlohner
Max-Planck-Institut für Physik
München
Bitnet: PEB@DMOMPI11

Software

Erratum:
The New Versions of TEX and METAFONT
TUGboat Vol. 10, No. 3

Donald E. Knuth

Editor's note: The following should replace the second full paragraph on page 326, column 1:

The special whatsit nodes are inserted automatically in unrestricted horizontal mode (i.e., when you are creating a paragraph, but not when you are specifying the contents of an hbox). You can insert a special whatsit yourself in restricted horizontal mode by saying `\setlanguage`⟨number⟩. This is needed only if you are doing something tricky, like unboxing some contribution to a paragraph.