A Permuted Index for TEX and LATEX

WILLIAM R. CHESWICK

AT&T Bell Laboratories
Room 2C416
600 Mountain Ave
Murray Hill, NJ 07974
ches@research.att.com

ABSTRACT

The Permuted Index for TEX and LATEX was written to help TEX designers find the right command among the extensive list of TEX, plain TEX, and LATEX commands.

To prepare the index, a one-line definition was written for each command. These definitions were run through the UNIX `ptx` filter, which created an entry indexed by each key word in each definition. UNIX tools were used to massage the text, and convert the `ptx` troff-style output into TEX macro calls.

This paper discusses some technical aspects of preparing the Index, and some of the problems encountered.

1. Introduction

It is easy to tell when someone is having trouble with a typesetting package: they are surrounded by heaps of nearly-identical printout. Their goal is often simple: to lower a headline or adjust spacing. When the struggle lasts more than a day, it becomes an Epic Battle.

Epic Battles seem to be a common result when a programmer attempts to stretch typesetting software beyond the novice examples. They often occur when the "safe driver" winds up on dangerous curves. The problems can be daunting, especially when the user has no TEXnician available. *The TEXbook* [Knuth] covers everything, of course. But it is actually three books in one: a beginner's manual and two levels of reference manual guarded by dangerous-bend signs. Some dangerous curves are vital, others extremely arcane and usually irrelevant. Which ones should be read, and which ignored, and by what level of user?

Epic Battles are not confined to TEX — I have seen proficient `troff` users in the same state. It seems that typesetting is a fundamentally difficult task, and these battles come from an incomplete understanding of complex typesetting programs, languages, and macro packages.

1.1 My problem

My early Epic Battles lasted as long as two days. (I was a manager at the time, and could ill-afford the time. I took it anyway.) One day I was creating a LATEX style file for a newsletter, including a table of contents. When I wrote out my contents line to a file, LATEX added lots of strange goo to my string. I had to tell TEX to stop expanding my macros and just write out the stuff I wanted. There *had* to be a command in the *TEXbook* somewhere, but how could I find it? *I* was the TEXnician, and TEXHaX wasn't available. It turns out that I wanted the `\string` command. It was hiding behind a double-dangerous curve on page 22 in the *TEXbook*. (I found it in an example dug out of `latex.tex`.) At this point I started thinking about a permuted index for TEX commands.

```
% a silly definition file
                frog    green amphibian found near water
                gnat    black insect found in the air
                gnu     animal found on land
                toad    green amphibian found on land
```


Figure 1: Some sample definitions and a permuted index of their definitions

1.2 A permuted index

A permuted (or keyword-in-context) index contains an alphabetical listing of each keyword in each command's definition. Figure 1 shows a short, silly example. The UNIX manual [Unix] has a permuted index for locating the correct text filter or command. There wasn't one for TEX, so I decided to write one. I even had a shot at the first definition:

```
\string:        don't mess with the following text I am writing out
```

2. Building the Index

The Permuted Index is generated from a file of definitions. The file is processed by a few UNIX filters to create a file of TEX macros. These are read into a LATEX file to create the Permuted Index. The index consists of three chapters summarizing the commands for TEX, plain TEX, and LATEX, and a combined permuted index of all these commands. It is well over one hundred pages long.

2.1 Command definitions

The first chore in creating the Index was to collect a complete list of commands. I had decided to index plain TEX and LATEX commands as well as TEX. Most of the TEX commands came from `tex.web`. The plain TEX commands were extracted from `plain.tex`, including a couple of internal commands that I have found useful. Barbara Beeton supplied a list that was a useful cross-check. `latex.tex`, `lplain.tex`, and `lfonts.tex` supplied the LATEX commands.

It has taken a long time to create and correct the definitions. I have tried to keep the style uniform and the definitions useful. A definition should give a fair description of the command's semantics using consistent keywords. For example, commands that manipulate tokens should have the word "token" in the definition. A symbol's definition contains the word "symbol", and math mode symbols have "math symbol" in the definition. Wording should be consistent: does `\parskip` define the "space", the "separation", or the "glue" between paragraphs?

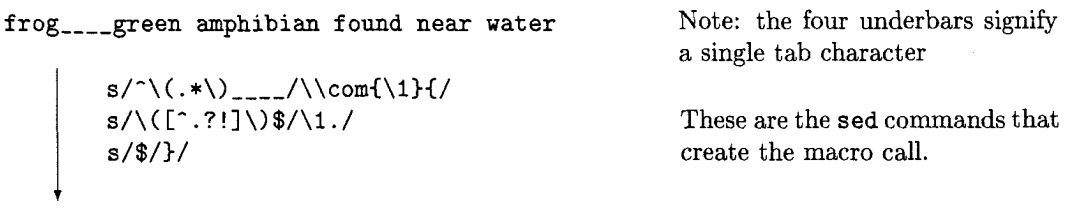

Figure 2: Generation of the command description macros

I battered my copy of the *TEXbook* hunting down definitions, and annoyed several mathematicians about the math symbols. I suspect real TEXperts could improve on some definitions, and novices could suggest keywords that should appear in some definitions.

Most command definitions are easy. For example:

```
|\time| current time of day
```

Some are not. Consider `\expandafter`, which has the following definition in the *TEXbook* (p. 213):

> TEX first reads the token that comes immediately after `\expandafter`, without expanding it; let's call this token t. Then TEX reads the token that comes after t (and possibly more tokens, if that token has an argument), replacing it by its expansion. Finally TEX puts t back in front of that expansion.

My definition:

```
|\expandafter|  expand the token following the next token
```

It doesn't have to be complete, just enough to guide the careening driver.

Proficient TEXperts (and others) can certainly find definitions that miss the mark. I welcome corrections and suggestions.

2.2 The command definition input files

The definitions for LATEX, plain TEX, and TEX are stored on separate files. The definitions are used in three different ways:

1. The filters in Figure 2 generate the `com` macros in three separate files. These are used in the chapters that summarize the command definitions.

2. They are processed by `ptx` to create the file with the index macro calls.

3. Lines beginning with `%%%\def` are extracted and the `%%%` is stripped off. These are special definitions that are read directly into the document before the commands and index are read.

Figure 3 demonstrates the processing steps on a very simple command definition file. Notice that a single command definition permutes to five index entries.

Each definition file has a simple format. There is one line per definition. Each line has a command, followed by a tab character, followed by the definition. Lines beginning with `%%%\def` are special definitions, described below. Other lines beginning with a `%` are ignored. Figure 4 shows some actual TEX definitions.

```
% This is a sample input file
% Comments and special lines have '%' in column one
% Note: four underbar characters stand for a single tab character
frog____green amphibian found near water
```

```
egrep -v "^%"
```

Strip comments and special entries from the file.

```
frog____green amphibian found near water
```

```
awk -F"[____]"  \
        '{ print $1 "{" type "}"  \
                "____--- " $2 }' \
        type=P
```

Format for the `ptx -r` option. `type` is set to "P" in this example.

```
frog{P}____frog--- green amphibian found near water
```

```
sed 's/\([^.!?]\)$/\1./'
```

Suffix period if no punctuation present.

```
frog{P}____frog--- green amphibian found near water.
```

```
ptx -r -f -t -w 200 -i eign
```

```
..xx "" "frog--- green" "amphibian found near water." "" frog{P}
..xx "" "" "frog--- green amphibian found near water." "" frog{P}
..xx "" "frog---" "green amphibian found near water." "" frog{P}
..xx "" "frog--- green amphibian found" "near water." "" frog{P}
..xx "" "frog--- green amphibian found near" "water." "" frog{P}
```

```
sed 's:^\.xx "\([^"]*\)" "\([^"]*\)" \
            "\([^"]*\)" "\([^"]*\)" \
        \([^{]*\){\([^}]*\)}$:      \
        \\ptx{\2}{\3}{\5}{\6}:
```

Convert from troff to TEX-style macros. Discard unused macro parameters. (The actual command is on a single line. It is broken up here to accommodate this annotation.)

```
\ptx{frog--- green}{amphibian found near water.}{frog}{P}
\ptx{}{frog--- green amphibian found near water.}{frog}{P}
\ptx{frog---}{green amphibian found near water.}{frog}{P}
\ptx{frog--- green amphibian found}{near water.}{frog}{P}
\ptx{frog--- green amphibian found near}{water.}{frog}{P}
```

Figure 3: Sample processing of a command file

```
|\-|    discretionary hyphen
|\/|    italic correction
%%%\def\showspace{{\tt\char'\ }}
+showspace+     space character
|\above|        fraction with rule thickness
|\abovedisplayshortskip|        extra glue above displays following short lines
|\abovedisplayskip|     extra glue above displays
|\abovewithdelims|      fraction with specified rule and delimiters
|\accent|       put an accent over the next character
```

Figure 4: Sample command definition input file

2.3 Editing the Index

The machine-generated permuted index in the UNIX manual was extensively hand-edited. Knuth said in the *TEXbook* that he prefers hand-editing of index entries. But I expect to make many revisions to the command definition file, and hand edits to my `ptx` output would be lost. I must make do with automated editing. There are a few things that help.

For one thing, not all words in a definition are useful keywords. By default, `ptx` consults a file named `eign` for a list of uninteresting words. The default file was not useful, since it included words like "left" and "right", which are certainly important keywords for the Index. So in the end, I built a list based on the original file and careful examination of the Index.

I built a script to locate pairs of adjacent lines that define the same command. In such cases, something is usually wrong: perhaps a command is defined twice, or something is amiss in the definition.

Finally, there is a filter to remove lines that are sorted in a non-obvious way. `Ptx` sorts on the ASCII character set, but it isn't obvious to a human where entries like `%` should appear. Also, actual command names tended to clutter the listing. Here are some entries that were rejected:

```
                    $— dollar sign symbol. ..... $
%— percent sign     (%). ..... %
                    \above— fraction with rule thickness. ..... \above
                    ´— acute accent (é). ..... ´
```

I can print a full list the rejected entries to make sure nothing important is ignored.

2.4 Macros

The `\ptx` macro formats each definition. It used to be fairly complex when I tried to wrap long definitions around on a line. The problem led me deep into TEX's paragraph-formatting algorithm, a wonderful but dangerous territory. After an Epic Battle, I gave up. A fairly simple version of the macro now truncates definitions.

Since the Index has to display the name of every TEX command within a line of text, it needs a good embedded verbatim environment. I took the macro from the *TEXbook*'s `manmac.tex` that uses pairs of vertical bars to delimit verbatim text. It works fine in straight text, but fails in some cases when used in an argument to a macro. Since all definitions in the Index are formatted by macros, this was quite a problem. The failure stems from the argument processing: TEX pre-scans the arguments with varying attention to the text's meaning. The LATEX `\verb` command demonstrates the problem in Figure 5.

The first `\mac` call works. The second and third die from unmatched curly braces. The fourth doesn't work because LATEX's `\verb` is not defined with `\long`. I am not sure why the fifth call fails. It appears to be trying to process the `\newif` command. (Other `\new`*xxx* definitions fail as well.) Also, a few commands caused problems with my filters: double quotes would confuse the patterns that match the troff macro fields; commands containing a blank would cause unwanted permutations by `ptx`.

It took an Epic Battle to convince me that I couldn't figure out a solution to the parameter

```
\long\def\mac#1{
        \begin{flushleft}
                #1
        \end{flushleft}}

\mac{This is a test of string: \verb|\string|}
\mac{This is a test of string: \verb|}|}
\mac{This is a test of string: \verb|{|}
\mac{This is a test of par: \verb|\par|}
\mac{This is a test of newif: \verb|\newif|}
```

Figure 5: Attempted verbatim in a macro call

problem using verbatim. I now use pairs of plus signs to delimit the names of macro calls that display the offending commands.

3. Results

The Permuted Index has been well-received. It clearly meets a need for many TEX users. I found that it was useful in its own preparation, a good sign.

The `\string` command now has the following definition in the Index:

```
|\string| expand a control sequence into character tokens
```

This is a reasonably accurate definition. Would it have helped me three years ago? I am not sure. Does a precise definition help the user find the command? Or should some commands be less precise but contain more familiar words to aid the novice.

Constructing the Index has taken far more work than I expected. Even so, further work is needed on several problems:

1. Better definitions are needed for some commands, especially those with difficult semantics.

2. I'd like to improve definitions for the novice.

3. I'd still like to solve the problems of line wrap in the index.

4. Foreign language versions could be useful. Also, other macro packages could be included.

4. Update

The Conference attendees made a number of useful suggestions. First, I am indebted to a number of people for improved definitions, especially Barbara Beeton and John Hobby. Here are a few suggestions that I will try to adopt in the next release of the Index:

1. The command names will be included in the Index, as well as listed separately as they are now.

2. The separate list of command names should be hidden in the back of the index: they are not as useful as I thought.

3. Several simple keywords, like 'page' and 'paragraph', don't yield as many entries as they should.

4. Simpler words are probably preferable to more accurate ones. For example, 'space' should probably be used where 'glue' is more precise.

5. Availability

The Permuted Index is available as Computer Science Technical Report 145 from Computer Science Research at Bell Laboratories. Send inquiries to `neera@research.att.com` or

Neera Kuckreja
Room 2C551
AT&T Bell Laboratories
600 Mountain Ave
Murray Hill, NJ 07974

After some review and revision it will be published in the *TEXniques* series, which will be available from the TEX User's Group.

At present, the source code is not available. For now, I would like to have some idea of the actual number of copies in use. I will consider special requests for the source for foreign language and other macro package versions.

Bibliography

Knuth, Donald E. *The TEXbook*. Reading, Mass.: Addison-Wesley, 1984.

UNIX Time-Sharing System Programmer's Manual, Vol. 1. 10th ed. 1989.

TEX Users Group

University of Washington

August 23 – 26, 1987