Upages – plain TEX for professionals

Stanislav Brabec

Vyšehradská 27
Praha 2 – Nové Město (Prague)
Česká republika
utx@k332.feld.cvut.cz

Abstract

When I have started my professional typography works in plain TEX, I found many things, which are done in each document. Some of them are language specific or trivial, but there exists many topics, which are strongly untrivial, and often required. TEX has many limitation, but there is (in recent time) nothing better in whole the world. Thus, we have powerful macro language but we haven't easy way to do many things: references, contents, page offsetting, interpretation of text token by token, cooperation with PostScript devices, device independent color and line drawing capabilities, easy box rotation and landscaping, making sheets and booklets, making other margins than 1 in, creating cropmarks, color signatures, color separations etc.

Certainly, there is many powerful macro systems, but they are very big, often slow, and cancels many capabilities of plain TEX. I has been particularly inspired by them, and particularly by some macros for plain TEX. But many of these macros are incompatible, if you want to use two of them, because one overwrites settings of the second.

This all is good reason to write powerful macros these things instead of creating trivial macros twice a month. Then you needn't spend much time to correct bugs caused by these trivial macros. That's why I have written my `upages.tex` macros. This chapter doesn't want to be a manual to these macros, but only an introduction with examples.

`upages.tex` macros consists on more parts. In this text I will describe the most powerful and interesting parts of them. I hope that my macros will greet many plain TEXists. They makes easy to prepare documents and to make hooks and patches.

1 Programmer structures

As I noted above, many macros are incompatible. The reason is simple: two macros redefining the same macro or variable often causes malfunctions. Programmer structures

gives you very neat and safe way to write such macros. For example, if you want to change your output routine to make at time landscape and mirror, it can be done easy by macro `\redef`. These structures helps to you in creating safe macros.

TEX is a macro language. To simplify many macros, it is often useful to have some programmer structures: stack, safe way to redefine some macros, tokens etc. `upages` brings you many macros for those rules. Some of them allow you to redefine `\par`, other to add and remove some tokens in `\everypar`, other push something to stack and return it back etc.

Another topic are macros called only in a special cases. `upages` brings new way to include these macros (mechanism is particularly similar to Pascal's `forward`) when they are required. It can save much memory.

These macros are simple for use, as shows following examples.

```
% Example of upages macro for internal definition with `@'.
% Internal definitions ends by \pull.
\def\internals{\pushedthe\catcode`\@=\letter}
% Save current font, then select other and return preceding.
\pushthefont \it text \pull
% Save current color, then type in red, then restore it.
\puththecolor \Red text \pull
% Add something to macro, then remove it (nesting required).
\redef\par\oldpar{(Here ends paragraph.)\oldpar}
     ...\par \restoredef\par\oldpar
% Add something to \everypar, then remove it (nesting not required).
\addtoks\everypar\oldeverypar{(Here starts paragraph.)\oldeverypar}
     ...\par \restoretoks\everypar\oldeverypar
% Permanently add contents of \toks0 to everyjob.
\etotokse\everyjob{\the\toks0}
% Number pushing. This is only naughty example of these macros.
Next page is \#\pushthe\pageno\incr\pageno\number\pageno\pull.
% Preliminary definition of macro \hebtext#1 in file hebrew
\friend\hebtext#1{}{hebrew}
```

These macros have many variations for different objects and types of requirements. No example above is using grouping. But end of group properly returns original meanings and cancels new stack items.

2 New output routines and device dependent setup

Classic `\plainoutput` performs easy output of single pages with margins 1 in. Man wants to typeset more general documents with different paper sizes and margins. Also output devices are not absolutely the same and often prints with different offsets, and

paper which we are using often has other dimensions than final book size. And, finally we are preparing documents with previewer, then we prints it on our printers, and final result is printed as mirror output on transparent materials or films. All these devices have different reference point, device size etc. Certainly, in most cases you can change offsets in command line, but this requires to know values for each document. If you want to create a booklet with new size, you must take a calculator and make some preprints, until you have correct position. But this all can do TEX. You indeed looses device independence of `dvi` file, but it isn't interesting to you when printing the document on your (oneness) printer is required. You can be happy, when you will send your file to the printer, and the document is exactly positioned without any proofs.

These macros gives you such possibilities. You only need one test to set reference point once forever for each output device or driver. You will do simple test, then set device preferences, and after setup you are ready.

Other part of these macros works with paper formats. Now you needn't any calculator to compute correct margins. You can easy set them! You needn't to remember paper formats – TEX does it for you.

Following example shows you the mechanism of usage of these drivers.

Example of setup files:

```
% Offset driver example.
\def\@myhpljv{%
\comment={Driver prints A4 with certain HP LJ5.}%
\devheight=297mm
\devwidth=210mm
\devhoffset=24,2mm
\devvoffset=26,0mm}

% Special driver example. (particular)
\def\@dvips{%
\def\beg@rotate#1{\rotstart{#1 rotate}}%
\def\beg@trnleft{\rotstart{270 rotate}}%
\let\end@rotate\rotfinish%
\let\end@flip\rotfinish%
\def\beg@flip{\rotstart{-1 1 scale}}%
etc...

% Example of mode definitions.
% \newmode{special driver}{offset driver}{preferences}
\newmode\preview{PasTeX}{}{}
\newmode\preprint{dvips}{myhpljv}{\pageinfo\turnmarks}
\newmode\film{dvips}{Linotype}{\mirroroutput\cropmarks\cutmarks
        \pageinfo\colorsamples\CMYKseparation}
```

```
% Example of paper size driver.
\def\USLetter{%
\totalwidth=8.5in
\totalheight=11in
\usedriver{USLetter}}
```

Following example shows, how easy is document setup:

```
\mode\preview % or \mode\preprint \mode\film etc.
\inmargin=14mm
\outmargin=17mm
\topmargin=15mm
\botmargin=21mm \withoutfootline
\doubleside
\booklet
\DIN A6
```

. . . and you will be sure to obtain what you want on all devices. No calculations, no proof prints. Described commands also rounds number of pages to be dividable by four to allow create a booklet. For such things you have there powerful mechanism of vacate pages generating.

3 Referencing system

References are common problem of TEX. Expansion method causes a lot of troubles with making them. Typical problem is references to pictures with picture number in `\inserts`. All simple methods can fail. You will need immediately expand number of picture, "~" mustn't be expanded at all, and, finally `\folio` needs to be expanded in time of `\shipout`. Thus I have suggested macros expanding tokens just in described order. These reference macros are based on powerful "interpretation" macros described below. These macros gives you chance to make easy references. Also "back references" are supported.

Example of reference macros:

```
\def\chapter#1{\incr\chapno\centerline{\bf#1}%
\totoc{\number\chapno: #1 .. \folio\par}}
```

This macro can be used simply as showed:

```
\chapter{Contents}
\inserttoc % inserts table of contents
\begintoc % starts writing of contents

\chapter{Introduction}
```

If you don't want to have chapter "0: Contents" in contents, you can simply use macros `\suspendtoc` and `\restoretoc`.

But this is not the only you can do: Command `\newref` generates reference macros for any file you want. It can be a list of images, index, cross references etc.

4 Interpretation macros

These macros are most powerful, difficult, mysterious and dirty[2] macros. They are true combination of most dirty tricks in the TEXbook! They takes token by token and operates with them. In these macros you are defining "processor", "eaters", "rollers" and "terminal". This macros allow you to write simply such macros as spaced text, making small letters to small caps, special expansions (see references), superprotection macros, macros for reverse order typesetting or anything other what you want. Such typesetting is many times slower than regular TEX typesetting, but makes possible many things. There are many types of macros: `\everytoken`, `\everyetoken`, `\everyatom`, `\everygeatom` etc. Differences between them are in interpretation of begin and end group characters and way to get tokens and expand them. Although these macros are versatile, it is not simple to write such interpreters: You must know way of `\if` conditions expansion, group counting etc. But it is a chance for you.

If you sometimes wanted to type something in hebrew, you certainly found, how difficult it is without `xet`. In following example you can see, how easy it is with interpreters. Macro for such things is trivial! Watch the following:

```
\newtoks\revtoks
\def\revorder{\revtoks={}%
\everyxeatom{\totoksb\revtoks{##1}}{\the\revtoks}}
{This is in normal order \revorder and this is in reverse order.}
```

That's all folks!

5 Footnotes

As noted above, `upages` has quite new output module. In this case it is easy to improve footnote style to make easy to change it (in plain TEX it is a rule for wizards). You are only changing `\footchar`, `\footdenotator`, `\footnotestyle`, `\endfootnotestyle`, `\footstrutbox`, and fonts (`\footnotefont`, `\footidentfont` and `\footdenotfont`). Following example shows, how to define footnotes:

```
\font\footnotefont=helv at 8pt
\footidentfont=helv at 7pt \let\footdenotfont=\footidentfont
\note{This is numbered note.}
```

In complex you can change footnote baselines, styles, denotation and identifier positioning.

6 Miscellaneous

This part will show you other macros from `upages`. `upages` contains many macros for easy work:

- `\reparfill` improves `\parfillskip` to prevent last line in paragraph from unwanted design.
- `\raggedleft`, `\raggedright`, `\raggedcenter`, `\raggedrldiag` etc. for different paragraph shapes.
- `\farnoindent` system for cases as `\farnoindent\medskip`, which allows `\removelastskip`.
- `\,`, `\>`, `\;`, `\!` makes to be defined outside math.
- `\inxy` and `\inyx` etc. to define transponable macros with `\xbox` and `\ybox`.
- `\framebox`, `\rectangle` etc. for drawing boxes.
- `\spaced` etc. for typing spaced text (using interpreters).
- `\hatebreak` and other penalization.
- `\setfont` for rarely used fonts.
- `\gridoriented` and whole mechanism to make grid oriented typesetting and rounding easier.

7 Device dependent functions and PostScript

To prevent changes in documents, this system suggest a set of independent macros to work with graphics, images, color etc. There is many programs, many device dependent `\special` commands. Good interface will prevent troubles and makes TEX source code as portable as possible. I'm not only, who suggests such interface. New LATEX3 will have such interface, macro package PSTricks has it's own. I hope that in nearest time will be ready single versatile special driver for all those systems, including all required functions (or warning messages). This part of `upages` is in my recent development. Following examples will work (I hope) with `upages v2`:

```
\linestyle{1pt}{\Black\fullline}
\fillstyle{\Red}
\ellipse{1cm}{3cm}{40}
\bitmappicture{mypic}
\rotated{30}\flipped\bgcolor\hbox{\Blue{Hallo}}
\setglobal{\mirror}
```

8 Future

Professional can want output with sheets containing 2, 4 or 8 pages, film saving mode canceling exposition of empty pages etc. Certainly, special commands for professional work with color would be welcome. (How perfect would be commands such as: `\screenangles`, `\CMYKseparation`, `\undercolorremoval` etc.!) But this is in recent time future. The whole purpose of this work is to get power to create books with color pictures in TEX.

My final destination is:

Typographer → TEX → dvips → exposition unit

Is there any reason to use suspicious programs, when we have TEX?