Multi-Page Layout Design by TEX

Mohammad Ghodsi *Fahimeh Malakooti*

Computer Engineering Department
Sharif University of Technology
Tehran, Iran

Abstract

TEX is a powerful system for automatic typesetting of almost all kinds of complex documents containing formulas, tables, and figures with different page layouts. Making use of the powerful features of `plain` *TEX is, nevertheless, a hard job. To facilitate special-purpose typesetting, a set of user-defined macros is needed. In this paper, we present a set of macros written in TEX which can be used to design complicated layout of a document with several pages. An interactive environment has been design to facilitate this even further. With this software, we can design the page layouts schematically and the main parts of the input file will be produced automatically.*

1 Introduction

TEX was designed by Professor Donald Knuth of Stanford University in 1978 and its final version was released in 1989 [1,2]. Because of its powerful features, nearly-perfect quality of its output, and its availability as a public domain software, TEX has been widely used throughout the world.

TEX's powerful commands, allows automatic typesetting of almost all kinds of complicated texts, formulas, tables, and page layouts. However, effective use of as many as TEX's 300 *primitive* commands and more than 900 commands of its `plain` version, is a hard task for complicated typesetting. That is why hundreds of macro packages has been developed to facilitate using TEX for specific publications, like typesetting scientific papers, books, newsletters, etc. One of these macro-based systems is LATEX which is very popular and broadly used today [3].

Designing flexible page layouts for articles, journals, and newsletters is also hard to do in plain TEX. A user may want to design a newsletter having a logical page layout like the one shown in Figure 1. The first page may include the title, the first part of the "main article," (in box 1.1), a picture (in box 2), the first parts of first news item (in boxes 3.1 and 3.2), the first part of the second news item (in box 4.1), and two advertisements (in box 5 and 6). The second page layout may be designed such that it contains the rests of the main article (in box 1.2), the first news item (in box 3.3), the second news item (in boxes 4.2 and 4.3), and a picture in box 7. Boxes 8 and 9, can be set aside for "leftovers" of the articles that cannot fit in the designated boxes, or for other advertisements. This design can continue on for other pages of the newsletter.

One would like to be able to typeset each part of the newsletter separately and let a macro package do the job of "cut-and-paste" automatically. This is what intend to do with TEX. There are commercial typesetting sys-

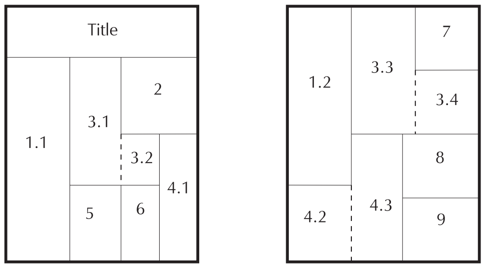

Figure 1: A sample page layout for a two-page newsletter.

tems, like `PageMaker` and `FrameMaker`, that provide such facility with nice user friendly WYSYWYG environments[1]. But, TEX is far ahead of these systems in automatic and logical typesetting of complicated documents; adding the ability to design complex page layouts is certainly a progress.

There has been several works to facilitate logical design of page layouts in TEX. Mittlebach has developed `multicol.sty` macros to do multicolumn typesetting based on LATEX[4]. Péter Huszár designed a macro to do single-page layout design [5]. Our work can be considered as an extension to [5].

In the rest of this paper, we will first introduce the overall view of our macro system briefly. Then, some implementation issues are discussed. In the last section, we conclude by mentioning how this work can be extended.

2 General Idea

The main part of the system is a set of macros written in TEX's language to handle many different issues involved. These macros work in conjunction with an interactive, user-friendly environment called `layout`, to design the page layout. A user should prepare each article as separate TEX file. He has all the TEX's capabilities for this. Thus, articles can contain tables, formulas, figures, etc.

With `layout`, user can easily draw different boxes on the monitor which wee call "layout-boxes" throughout the rest of this article. The user can assign each part of an article to a different layout-box, as he wishes, to complete the layout design. The prepared articles are preprocessed to provide the system with the approximate area covered by each article. Having this data, `layout` will provide the user with on-line data to help allocate layout-boxes that almost fit the specific articles. User can specify the areas to be *outlined*, in which case a border box is drawn automatically around that area, or *shaded*, in which case the area is outlined and filled with gray. User can also draw different lines to divide the texts.

The final layout of the pages are stored in TEX's memory. For each layout-box, its name, its top-left corner distances from top-left corner of the page, its dimensions, the page number, the next layout-box for the rest of this article, and what it contains is stored in TEX's registers.

After storing the information needed for layout-boxes, we can now typeset the article in each layout-box. The general algorithm is as follows. We process an article paragraph by paragraph, each paragraph containing different objects in TEX (including `box, glue, kern, penalty` etc.,) some with recursive structure. Each paragraph is typeset in a layout-box declared by `layout`. If the whole paragraph fit in the layout-box, we append the typeset paragraph at the end of the material in that layout-box and we progress to the next paragraph. If not, the paragraph is automatically divided into two parts; the first

[1]WYSYWYG: What You See is What You Get

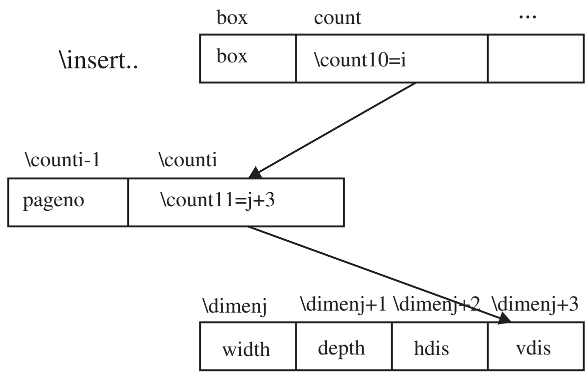

Figure 2: Structure used to store information for each layout-box.

part fills the layout-box completely, and the second part is set aside to be typeset in the next layout-box. Since the dimensions of the second layout-box is usually different, we need to "unwrap" the second part of the paragraph (which we have it in the typeset form now, and cannot convert it back into the original text form,) and re-typeset it in the new layout-box with new dimensions. This process is repeated until an actual page is filled with pieces of articles typeset by our system. The complete pages are shipped out to the TEX's output file which can then be viewed or printed out on laser printers.

Many complications involved in this process, mostly as the results of the way TEX has been implemented. Some of the implementation issues will be discussed in the next section.

3 Some Implementation Issues

Storing the layout information in TEX's memory is not straight forward. TEX has 6 different registers types. For each type, there are only 256 registers available, some of which are used by the TEX itself. `\count, \dimen, \skip, and \box` are four commonly used register types for storing integer numbers, dimensions, glues[2], and TEX boxes respectively. `\box` has a recursive structure which can contain only a character or nested boxes with `glues` and other items in between and typeset as paragraphs. `insertion` is a structure that makes available a set of one register types.

We have used `insertion` structure to store layout-box information; this is shown in Figure 2. The layout-box name and the name of the next layout-box are assigned to the `insertion` number using `\chardef` and `\toksdef` commands. `\count10` is a register used by `plain` TEXto store the number of the last registered allocated by user. We have used this counter to link the information. Four dimensions of a layout-box is stored in four `\dimen` registers. The `\box` register in the insertion will contain the material to be typeset in the layout-box. To do all these, we had to make some changes in `plain` TEX's register assignments.

Design of the macros to handle the typesetting and re-typesetting of paragraphs in layout-boxes are the main part of this project. We have redefined the output routine and `\par` macro in `plain` TEXfor this purpose [6]. When a paragraph is typeset in the layout-box, the length of the typeset material is measured. If it is more than the length of the layout-box, items are removed from the typeset paragraph line by line until the remaining part fills the layout-box. This is done by `\unvbox` commands with many cares.

The rest of the processed paragraph which is available in a typeset form, should now be

[2] glues are dimensions that can shrink or expand

re-typeset in another layout-box with different dimensions. This is the source of further complications. To mention one, there might exist hyphenation characters, "-", in this part of the paragraph. We have not found a way to remove this character automatically; we only avoid this by turning off the hyphenation is the region close to the break points.

Whenever, a layout-box is filled our output routine is called by issuing a `\break` command to do the job of "pasting"; a virtual page is filled in by pieces from different layout-boxes. We use `\kern` and `\lower` commands to paste the material typeset in layout-boxes onto the virtual page. We return to the original output routine, after the virtual page is completely full. This causes one real page to be shipped out to the `*.dvi` file, the final output file. TEX's `dvi` files are device independent and can be viewed on screen or printed out on almost all output devices using appropriate device drivers.

Lines are drawn similarly at the coordinated declared by `layout`. The lines around the outlined layout-boxes are handled in the same way. A gray layout-box is treated as a regular layout-box which will be filled by repeated overwriting of a character from a font named `gray`. This font can easily be prepared at different gray scale.

A sample scientific newsletter prepared by our system is shown at the end of this article.

4 Conclusions

In this article, we have briefly presented our work to add automatic multi-page layout design to TEX's powerful typesetting abilities. We have developed a user-friendly environment called `layout` to help users design the layout of all pages of his document. The main part of this work is the design of a set of macros written in TEX language to handle the automatic typesetting and "cut-and-pasting' We explained the main ideas used in the macros in this article. The macros use many of the limited registers and boxes available in TEX. Therefore, our current system will run out of memory if the document uses more than 40 layout-boxes. We need to find a mechanism to reuse the registers. These are also some limitations on the commands that can be used in the articles. This is because paragraphs are typeset as a `\vbox` and certain restricted horizontal commands are not allowed in the vertical mode. There are ways to avoid this also. Further work on this work continues to remove these limitations and also apply these macros to FarsiTEX, the persian version of the TEX developed by the first author [7].

References

[1] D. E. Knuth, *The TEXbook*, Addison-Wesley, 1994.

[2] D. E. Knuth, *TEX: The Program*, Addison-Wesley, 1986.

[3] Leslie Lamport, *LATEX: A Document Preparation System*, Addison-Wesley, 1994.

[4] Frank Mittelbach, *Multicolumn Environment*, TUGboat, 1992.

[5] Péter Huszár, *Over the Multicolumn*, TUGboat, Volume 13, No.2, 1992.

[6] David Saloman, *Output Routines: Examples and Techniques*, in three parts, TUGBOAT 11 #1, #2, and #4, 1990.

[7] M. Ghodsi, H. Abolhassani, and M. Sharghi, "*FarsiTEX:A System for Bilingual Typesetting of Scientific Documents,*" 2nd Iranian Computer Conference, Iran University of Science and Technology, Feb. 1994, Tehran, Iran.