HI-TEX Cutting & Pasting

Michael Ballantyne
Michael Spivak
Yoke Lee

The Problem

As more and more macro packages are written for TEX, the problem of exceeding TEX's memory capacity becomes more common and more acute. In fact, versions of TEX with larger memory have already been created to help alleviate this problem, but such versions don't run on the personal computers that most of us have.

Some macro packages try to skirt the problem by selectively loading only needed subsets, but this strategy can fail when many different sorts of constructions are required in the same file. It remains true, however, that different collections of macros are normally required for different parts of a file, with insuperable problems arising only when several different collections need to be loaded at once.

For example, a large macro package might succeed in typesetting individual complicated tables, but cause TEX's memory limits to be exceeded when used in conjunction with other macro packages, or when numerous tables have to be held over for inserts. In this case, as a last resort, one could: (1) make a special file to individually typeset all the tables required for a book or paper, one to a page; (2) leave the proper amount of blank space in the main file for each table; (3) print the two files, cut the tables from the special file, and paste them into the blank spaces in the main file.

Though an analogous procedure is required when a photograph has to be inserted in a book, it seems singularly unattractive when the inserted material is just more text that has already been typeset by TEX. But some of the allure may be restored when the computer is used to do the cutting and pasting.

The DVIPASTE Solution

Our "solution" involves a little macro package, `dvipaste.tex`, and a C program, `dvipaste.c`. In the case of the table example discussed above, we would first make a file, say `tables.tex`, of the form

```
\input dvipaste
\input ⟨macros for tables⟩
\setbox0\hbox{⟨table 1⟩}
\sendout{\box0}
\setbox0\hbox{⟨table 2⟩}
\sendout{\box0}
 ...
\end
```

When this file is run through TEX, it will produce `tables.dvi`, and an auxiliary file `tables.dat`. Printing `tables.dvi` will produce the various tables, one to a page, each positioned at the bottom left corner of the page. The file `tables.dat` will contain a sequence of lines like

```
123.45pt .4pt 233.567pt
```

where line n contains the height, depth, and width of the table on page n.

Now the main file, say `book.tex`, will also have `\input dvipaste` at the beginning, but here each table will be replaced by

```
\paste{tables}{n}
```

where n is the number of the page on which the desired table is printed in the `tables` file. A `\paste{...}{...}` can appear anywhere, for example, as

```
\centerline{\paste{tables}{n}}
```

or

```
\midinsert{\paste{tables}{n}}
```

etc. TEX will replace each such `\paste` command with a blank box having exactly the right height, depth and width (which it reads from `tables.dat`), at the same time inserting an informative little `\special`, which most drivers will happily ignore. When `book.dvi` is printed, exactly the right amount of space appears for each table.

It would appear that we haven't done much more than the procedure outlined in the previous section, except that the amount of blank space for each table has been measured for us by TEX. Now, however, we can use the `dvipaste` program,

```
dvipaste book newbook
```

to use the file `book.dvi` to produce a new file `newbook.dvi`. In creating this new `.dvi` file, `dvipaste` will examine the `tables.dvi` file and extract `.dvi` commands to be placed at the position of the various `\special`'s that were inserted by the `\paste`'s. These extra `.dvi` commands have the effect of causing the tables to be printed in precisely the places occupied by blank spaces in `book.dvi`. Thus, `newbook.dvi` will print exactly what `book.dvi` would have printed if the table specifications had been part of `book.tex` (which is not to say that `newbook.dvi` is exactly the same file that `book.dvi` would be).

Although this solution may not be ideal, it involves only an insignificant amount of extra time,

not to say that `newbook.dvi` is exactly the same file that `book.dvi` would be).

Although this solution may not be ideal, it involves only an insignificant amount of extra time, since `dvipaste` runs much quicker than TEX, and little extra work. It's true that an extra file is required, but this isn't an overwhelming inconvenience—it might even be *more* convenient to keep all the tables in a separate file. This illustration used a single auxiliary file `tables.tex`, but any number could actually be used.

How it Works

`\sendout#1`, defined in `dvipaste.tex`, writes a line to the `.dat` file giving the height, depth and width of `#1`, and then adds a `\vskip` down to the bottom of the page, followed by

```
\special{beginpaste:}%
\noindent\rlap{\smash{#1}}%
\special{endpaste:}%
\vrule height1sp width1sp depth0pt
\eject
```

The `\vrule` is obviously not meant to be seen. The only important thing is that it's *there*, more precisely that it's *here*, right back at the point where the `\noindent` began. This means that the `.dvi` commands between the two `\special`'s will create the table seen on the page, *starting and ending* at the lower left corner of the table.

On the other hand, `\paste{`⟨subfile⟩`}{`n`}` expands to

`\special{dvipaste:` ⟨subfile⟩n`}\vbox...`

where the `\vbox...` is a blank box with the height, depth and width given on line n of ⟨subfile⟩`.dat`. (The first `\paste` with the argument ⟨subfile⟩ causes TEX to open the file, and store all the information in an appropriate place; subsequent uses merely ferret out that information.) The `dvipaste` program looks for such specials, and replaces them by the relevant `.dvi` commands on page n of ⟨subfile⟩`.dvi`. Of course, it's a bit more complicated than that, because each font declaration in a `.dvi` file must be made just once (before the `postamble`), so font declarations from a ⟨subfile⟩`.dvi` must be deleted if they have already appeared in the main file, renumbered if they declare new fonts, etc.

Extensions

Although current drivers will presumably ignore a `\special{dvipaste:}` command, they don't *have* to! In fact, screen and printer drivers could perform the same maneuvers as `dvipaste`. A screen driver of this sort would preview the complete book, with the tables inserted, without using `dvipaste`. `dvipaste` itself might be reserved for the final run, before the `.dvi` file is sent off to the typesetter.

Availability

`dvipaste.c` and `dvipaste.tex` are copyrighted in the GNU spirit (they are distributed for a nominal charge, and must be passed on according to the same terms). For a standard IBM 360K double-sided diskette containing `dvipaste.tex`, `dvipaste.c` and an MS-DOS executable `dvipaste.exe`, send $4.00 to TEXplorators, 3701 W. Alabama, Suite 450-273, Houston, TX 77027.