the Adaptec, although it did work with the DTC controller.

Backing up to a removable device

The other component of my backup regime consists of backing up to floppies, using Fastback-Plus. Every month, I do a full backup of my entire hard disk to floppies; in doing so, Fastback turns off the archive bit of all the files. Every other day (or so), I do a differential backup, where Fastback copies just those files that have their archive bits 'on' — that is, just those files that have been changed or added since the last full backup; these are the same files that are copied to the '`D:`' drive by the batch file given above.

Fastback-Plus does have its problems: version 1.00 was unable to restore two out of the fifteen floppies it had created in backing up my hard disk (I did the backup with Fastback's read-after-writing verification turned on — see below). I suspect that the problem was caused by an imperfection in the way version 1.00 formatted new disks as it backed up. (If there's one program that had better be flawless, it's your backup program.)

Fastback also had trouble restoring files it had placed on a Bernoulli cartridge. If you have a Bernoulli box, I suggest you partition your hard disk into 20 Mb partitions, and backup the hard disk by `xcopy`'ing each partition to individual cartridges. This has the added advantage of not having to de-Fastback the files in order to use them.

I also noticed that if I formatted 360 K diskettes in my 1.2 Mb drive after running Fastback's installation routine, an inordinately high number of bad sectors were unjustly locked out. This problem went away after I reset the computer.

One of the nicer features of Fastback is that it allows you to exclude file and directories from the backup. I've set up my copy so that it excludes: the operating system kernel (`\command.com`, `\ibmbio.com`, and `\ibmdos.com` under PC-DOS); all the DOS programs (kept in `\DOS` on my system); and `*.dvi`, `*.log`, `*.qeb`, `*.qex`, and `park!@#.cor`. (Should my hard disk fail, I'll need to restore DOS and Fastback from their distribution diskettes anyway, in order to run Fastback to restore the other files.)

I run Fastback with 'write-verify' on, 'compression' set to 'save disks', and 'error correction' on. On my 386 system, Fastback takes about a minute per megabyte with these settings. The 'write-verify' option sounds like it offers more security than it really does: Fastback does not try to read back the information it wrote out to the floppy — all it does is compare what it read off your hard disk with the copy of that information it has in RAM.

If you use a hard-disk cache, make sure to turn it off before running a backup program — it will defeat the verification attempted by the program. (Personally, I don't bother when doing a differential backup, but before doing a full backup, I replace my `autoexec.bat` and `config.sys` files with the simplest possible versions and re-boot, to avoid any detrimental interactions that might occur between the backup program and, say, a resident program.)

For greater security, I alternate between two sets of floppies for both the full backups and the differential backups. To keep the differential sets straight, I move a Post-It marked 'Use Next' between the sets. Before doing a full backup, I put the last differential at the back of the box containing the most recent full backup, and then over-write the box containing the oldest full backup. This system not only allows easy recovery, but also allows me to dig up an early copy of a file if I find out that I've accidentally trashed a file, and backed up the trashed file.

The current version of Fastback-Plus is Version 2. It includes a separate verify feature: after the entire backup is complete, you run this option, and re-insert every diskette; Fastback will compare each file on the hard disk to the copy on your floppies (unfortunately, this still doesn't guard against an out-of-alignment floppy-disk drive). It also can be set to automatically delete unwanted history files.

Evaluation of K-Talk

C.G. van der Laan and J.R. Luyten
Rijksuniversiteit Groningen

We would like to announce the availability of a report entitled "Evaluation of K-Talk", RC-report 22, Groningen, 1988. Further information can be obtained from the authors. The Foreword of the report is reproduced below.

At the Rijksuniversiteit Groningen document preparation is done by text processors and document preparation systems 'at the desk', with possibly remote 'execution' and printing.

At the moment WordPerfect and TeX as representation of respectively text processors and document preparation systems enjoy the highest 'support category' — they are standards for the time being.

In practice most users start with WordPerfect and sometimes end with TeX.

Intermigration tools between these systems are therefore useful. K-Talk is a program that translates WordPerfect files into TeX files. This report aims to provide an answer to the question 'Is K-Talk a good tool?' Apart from 'the answer' — if any black and white answer is possible — we constructed a test collection of judiciously chosen document elements such that comparison can be made easily with future releases of K-Talk or similar products.

Furthermore is must be noted that translation of a WordPerfect document of only a few pages will produce a TeX file, preceded by a dozen or so pages of macros; moreover a macro library of considerable size is used. The average TeX user will usually not understand those macros. Adaptation of the produced document is not always simple. We also note that processing of the translated `.tex` file by e.g. LaTeX or AMS-TeX is by no means trivial.

Training

Making Paragraphs

Alan Wittbecker

TeX, as perhaps you already know, is a typesetting program for the production of beautiful pages. Using TeX to produce beautiful pages is easy if you let TeX make the design decisions already built into the program. TeX is easier to understand, however, if you have an appreciation of the history of typography and book production, not to mention computer programming.

This article, as the first of a series, presents TeX in the simplest way — as a set of instructions for typesetting text. (The tutorial on page 276, by contrast, concentrates on TeX as a computer program.) TeX instructions are added to a file, called the source file, containing text. The text has been entered with words separated by spaces and groups of words separated by punctuation and blank lines, which represent the boundaries of phrases and paragraphs.

TeX instructions describe the procedure that TeX follows when it formats the text. The instructions and the text are entered using keyboard characters (ASCII characters), so that the file can be transferred easily to other computers. TeX formats a source file to a device-independent file (DVI file) that can be printed (after going through a DVI translator and output driver) on dot-matrix printers, laser printers, or typesetters.

A TeX instruction requires a special character, the backslash (\), to be recognized as an instruction. The body of the instruction is composed of alphabetic characters, usually a word; each instruction, in general, is ended with a space. Instructions are simply entered in the text.

Instructions are gobbled up — it is permissible to anthropomorphize TeX — when the text is formatted. Characters are read and printed as characters, unless they are special, like the backslash or percent sign (%), which is the comment character; but, even these can be printed with a specific instruction. Extra blanks between words are discarded because TeX calculates an optimum interword space — therefore, you can use blanks to arrange the source file. The carriage return from the terminal is converted to a blank.

TeX Makes Paragraphs

TeX considers the paragraph the basic unit of production. A paragraph is a short composition consisting of a group of sentences. The clear separation of paragraphs can distinguish thoughts, clarify content, and increase comprehension. A paragraph is commonly indicated by starting a new line and indenting that line. Sometimes space between paragraphs is also used to distinguish them.

After TeX reads text into its memory — by the mouthful! — paragraphs are examined for goodness according to a mathematical standard of beauty (based actually on calculations of "badness"), and then separated into lines.

TeX recognizes the beginning of a paragraph by reading an alphabetic or numeric character. The paragraph instruction itself (`\par`), however, comes at the end of a paragraph, instead of the beginning. All other TeX instructions precede the text they describe. A blank line also causes the end of a paragraph; actually, two consecutive keyboard carriage returns are translated as a `\par` instruction. Thus, the simplest source file only needs blank lines separating paragraphs. Figure 1 lists a source file. Notice the use of the percent sign to comment out information — nothing that follows a percent sign on a source line is even read.

```
%    figure 1. paragraphs
\TeX\ is part of a long tradition of
```