# DVII: A TEX dvi information utility User Manual (version 0.44)

Adam H. Lewenberg

May 4, 2002

# Contents

# Chapter 1

# Introduction

`dvii` is a utility written in C that extracts information from a TeX dvi file. Information displayed can include a summary:

- File comment (usually the date the file was compiled)
- File size
- Number of (physical) pages
- Number of fonts

as well as more detailed information:

- Font names for all fonts used
- List of physical page number/TeX page number pairs
- List of all `\specials` and the page on which they appear
- List of all fonts used on a given page

For example, running the command `dvii -u test.dvi` generates the summary

```
File size: 1500 bytes (1 K)
Comment string:  TeX output 1999.12.05:1951
Page count: 6
Number of fonts: 3
```

## 1.1 Design Goals

I have designed this utility with the following goals in mind:

1. It should be *fast*, faster than `dvitype`.
2. It should be easy to use the output as a back end to PERL, enabling the easy manipulation of the data for more specific purposes.
3. It should be useful to the TeX community.
4. It should be portable.

In light of these goals, I encourage all those who use it to send me bug reports, comments, and suggestions on how it can be improved. I can be reached at `adam@macrotex.net`. In particular, I am interested in making this program compile on as many different platforms as possible, so if you have compiling problems, please let me know.

## 1.2 Some examples of use

The following examples use the file `test.dvi` whose source can be found in Appendix B.

### 1.2.1 Basic usage

The syntax for use is

```
dvii [options] dvifile
```

where `dvifile` is the name of the dvi file that you want information about. You can leave off the extension `dvi`; thus, `dvii test` and `dvii test.dvi` are equivalent. (Unless you also have a file named simply `test` in which case `dvii` will complain that `test` is not a valid dvi file. In such cases, be sure to explicitly supply the extension `.dvi`.)

### 1.2.2 Display a summary of information

```
dvii -u test.dvi
```

```
File size: 1500 bytes (1 K)
Comment string:  TeX output 1999.12.05:1951
Page count: 6
Number of fonts: 3
```

The summary tells us: the dvi file `test.dvi` is 1500 bytes long (roughly 1 kilobyte), was compiled on December 5, 1999 at 7:51 PM, has 6 pages, and contains 3 fonts.

### 1.2.3 List all information

```
dvii test.dvi

File size: 1500 bytes (1 K)
Comment string:  TeX output 1999.12.05:1951
Page count: 6
Number of fonts: 3
f:[50/cmr10/1200]::4bf16079
f:[23/cmbx10/1000]::1af22256
f:[0/cmr10/1000]::4bf16079
p:[1/1]
p:[2/2]
p:[3/3]
p:[4/4]
p:[5/-1]
p:[6/-3]
s:[2/2]:: A short special
s:[3/3]:: abcdefghijklmnopqrstuvwxyzABCDE...
s:[4/4]:: PSfile 1.eps
s:[4/4]:: PSfile 2.eps
s:[4/4]:: PSfile 3.EPS
s:[4/4]:: PSfile dog1.gif
s:[4/4]:: PSfile cat.eps
s:[6/-3]:: Some control characters: []
```

With no options, `dvii` first displays a summary of the dvi file (see section 1.2.2). The next 3 lines list the three fonts used by the dvi file.

The succeding 6 lines list page information. The format `p:[`$m/n$`]` means that this is physical page $m$ corresponding to TEX page $n$. The physical page is the page in the order that it would be displayed or printed out on a printer. The TEX page is the page number that would appear (in normal circumstances) at the bottom of the page. For more on the difference between physical and TEX pages, see section 4.1.

The last 6 lines list all the `\special`'s included in the dvi file. Note that not every dvi file will contain `\special`'s. The format is the same as that for pages; the text following the double colon `::` is the first 128 bytes of the `\special` text.

## 1.3 The `dvii` home page

The `dvii` home page can be found at

`http://www.macrotex.net/dvii/dvii.html`

# Chapter 2

# Installing

To install, copy the file `dvii` (or, in the DOS/Windows environment, the file `dvii.exe`) to a directory in your PATH.

To uninstall, remove the file `dvii` from whereever you put it.

## 2.1 Where to get executables

You will find several executables at the `dvii` home page. Currently, there are executables for the MS-DOS, Windows 95/98/ME/NT/2000, Linux, and Solaris (Sparc and Intel) platforms. Please let me know of any that do not work.

After you download the executable, you need to rename the file to `dvii.exe` (under DOS or Windows), or `dvii` (under Unix), and place it somewhere in your PATH.

## 2.2 Where to get the source code

`dvii` is written in portable C code whose source is a single file named `dvii.c`. It should compile on any system with an ANSI C compiler. To get the source file, go to the `dvii` home page at `www.macrotex.net/dvii/dvii.html`.

# Chapter 3

# All the command line options

## 3.1 `-c`: perform simple validity check

The `-c` option does a simple validity check of the dvi file: (a) the first two bytes should be `247 2`, (b) there should be at least four `233`'s at the end of the file immediately preceded by a `2`.

This is not much of a check: it will certainly not catch dvi files whose middle bytes have become corrupted, but it will at least catch files that are clearly *not* dvi files (such as text files).

Note that the `-c` option is implicit in almost every other option, so its main use is to detect files that are probably *not* dvi files.

Ses section 15 of `dvitype.web` for more information.

Example of use:

```
dvii -c test.dvi

dvi file 'test.dvi' passed validation check (level 0).
```

## 3.2 `-C`: perform more rigorous (and slower) validity check

(Note: this is a *capital* 'C'.) This command causes `dvii` to traverse the entire dvi file parsing each opcode. If the dvi file is corrupted, this command will detect it.

However, you pay for this: `-C` is much slower than `-c`.

```
dvii -C test

dvi file 'test.dvi' passed validation check (level 1).
```

## 3.3 `-d`: dump opcodes

(This option is available in version 0.42 and later.) This option (which implies the `-C` option) displays all the opcodes to printed starting with *pre*. Here is a sample:

```
dvii -d test

o:247
o:139
o:141
o:159
o:142
...
o:248
dvi file 'test.dvi' passed validation check (level 1).
```

where `...` are many opcodes. The script `details.pl` uses this mode to create an opcode summary of the file; see section C.3 for more information on `details.pl`.

## 3.4 `-f`: display fonts

If you want to list all the fonts used in a dvi file, use the `-f` option. Note that different sizes of the same font will list separately.

```
dvii -f test

f:[50/cmr10/1200]::4bf16079
f:[23/cmbx10/1000]::1af22256
f:[0/cmr10/1000]::4bf16079
```

The file `test.dvi` calls 3 fonts: font number 50 is `cmr10` at size 1200, font number 23 is `cmr10` at size 1000, and font number 0 is `cmbx10` at size 1000. (For more information on fonts, see chapter 5.)

## 3.5 -F: display fonts on each page

This option (which implies the `-p` and `-f` options) lists the fonts used on each page.

```
dvii -F test

f:[50/cmr10/1200]::4bf16079
f:[23/cmbx10/1000]::1af22256
f:[0/cmr10/1000]::4bf16079
p:[1/1]
  start of font list
  Font [0/cmr10/1000]
  end of font list
p:[2/2]
  start of font list
  Font [0/cmr10/1000]
  end of font list
p:[3/3]
  start of font list
  Font [0/cmr10/1000]
  Font [23/cmbx10/1000]
  end of font list
p:[4/4]
  start of font list
  Font [0/cmr10/1000]
  end of font list
p:[5/-1]
  start of font list
  Font [0/cmr10/1000]
  Font [50/cmr10/1200]
  end of font list
p:[6/-3]
  start of font list
  Font [0/cmr10/1000]
  end of font list
```

## 3.6 -g: suppress control characters when showing special text

The `-g` option can be used when displaying `\special` information. If a `\special` contains text with unprintable characters, it may corrupt some computer screens. So, instead of trying to print these unpleasant characters, `dvii` will instead print a '`.`'.

Here is what the `\special` text looks like *without* the `-g` option:

```
dvii -s test.dvi

s:[2/2]:: A short special
s:[3/3]:: abcdefghijklmnopqrstuvwxyzABCDE...
s:[4/4]:: PSfile 1.eps
s:[4/4]:: PSfile 2.eps
s:[4/4]:: PSfile 3.EPS
s:[4/4]:: PSfile dog1.gif
s:[4/4]:: PSfile cat.eps
s:[6/-2]:: Some control characters: []
```

(Note: there are four control characters between the square brackets; because TEX does not like these particular characters, you cannot see them here.)

Now *with* the `-g` option:

```
dvii -s -g test.dvi

s:[2/2]:: A short special
s:[3/3]:: abcdefghijklmnopqrstuvwxyzABCDE...
s:[4/4]:: PSfile 1.eps
s:[4/4]:: PSfile 2.eps
s:[4/4]:: PSfile 3.EPS
s:[4/4]:: PSfile dog1.gif
s:[4/4]:: PSfile cat.eps
s:[6/-2]:: Some control characters: [....]
```

## 3.7 -h: show main help screen

This option shows the main help screen which begins

```
dvii -h

This is dvii 0.?? (DVI file information) by Adam Lewenberg
Send bug reports to adam@macrotex.net

dvii -h             (Print help screen)
dvii dvifile        (Print dvi information)
dvii [many possible options] dvifile
-------------------------------------------
 -H : show complete list of options
 -h : show this help screen
 -u : display dvi summary
```

```
...
```

etc.

## 3.8 `-H`: show options help screen

This option displays all the command line options:

```
dvii -H

All the command line options:
  -c : perform simple validity check
  -C : perform more rigorous (and slower) validity check
  -f : display fonts
...
```

etc.

## 3.9 `-m#`: add message digest information when displaying pages

(See Chapter 7 on message digests.) Display a 16-byte message digest for each page.

```
dvii -m test

[message digest: simple sum]
p:[1/1]::D8C977816A091771A3A631E7592DAE6D
p:[2/2]::16E5A31EF87F926DB1F86CAD165C5453
p:[3/3]::1139D12267E3D60267CC788905909170
p:[4/4]::68E5A2E9D7320743CB85769CAAE9B023
p:[5/-1]::DE72BC3345FFDF7E779C8C667DCE3F97
p:[6/-3]::12808C9489FD3430C3C8EDAD8A4EC760
```

## 3.10 `-M#`: add message digest information but ignore certain opcodes

(See Chapter 7 on message digests.) Display a 16-byte message digest for each page, but ignore certain dvii opcodes.

Depending upon the value of `#`, certain of the dvi information will be ignored when computing the message digest.

`-M0` Do not ignore any of the opcodes; equivalent to `-m`.

`-M1` Ignore all font assignments and font definitions.

`-M2` Ignore all NOPS (no ops).

`-M4` Ignore all `\special`'s.

`-M8` Ignore all pushes and pops.

These options can be combined by adding; for example, to ignore fonts *and* `\special`'s, use `-M5`. A rationale as to why you might want to ignore certain dvii page information when computing a message digest follows.

If you use page message digests to detect when a page has changed, you may get false alarms when the only thing that changes in the dvi file is the order of font definitions: the page *looks* the same but will have a different message digest because the message digest has changed.

For example, consider the two files

```
% File A                      % File B
\font\a=cmr10 at 11pt         \font\b=cmr10 at 12pt
\font\b=cmr10 at 12pt         \font\a=cmr10 at 11pt
\a A                          \a A
\b A                          \b A
\end                          \end
```

These two files generate identical looking output but different dvi code because the font numbering is different. If you use `dvii -m` you will see different fingerprints:

```
[message digest: simple sum]
A: p:[1/1]::6676847979940C046D093C93789A1812
B: p:[1/1]::6676837979940B056D093C94789A1812
                         ^
```

But, if you use `dvii -M1` you get

```
[message digest: simple sum (ignore font)]
A: p:[1/1]::B42E10A601774236B86D11CE2F8C25E4
B: p:[1/1]::B42E10A601774236B86D11CE2F8C25E4
```

By examining the fonts with the `-f` option you can see where the two dvi files differ:

```
dvii -f filea                       dvii -f fileb
f:[51/cmr10/1200]::4bf16079         f:[51/cmr10/1100]::4bf16079
f:[50/cmr10/1100]::4bf16079         f:[50/cmr10/1200]::4bf16079
f:[0/cmr10/1000]::4bf16079          f:[0/cmr10/1000]::4bf16079
```

The font `cmr10` at 1200 is font 51 in `filea`, whereas it is font 50 in `fileb`.

## 3.11 -n#: display # bytes of special text

Normally, `dvii -s test` will display a maximum of 128 bytes of the `\special` text. For example, let `filec.tex` be the file

```
\special{0123456789abcdefghijklmnopqrstuvwxyzABCDE...}
\bye
```

(Note that the '...' indicates more special text which we omit due to space limitations in this manual.)

```
dvii -s filec
s:[1/1]:: 0123456789abcdefghijklmnopqrstuvwxyzABCDE...
```

However, you can tell `dvii` how many bytes to display by using the `-n` option:

```
dvii -s -n20 filec
s:[1/1]:: 0123456789abcdefghij
```

Finally, if you want to display *all* the `\special` text, use `-n-1`:

```
dvii -s -n-1 test
s:[1/1]:: 0123456789abcdefghijklmnopqrstuvwxyzABCDE...
```

Here, the output of `dvii -s -n-1 test` looks like `dvii -s test` only because we do not have enough page to show the entire `\special` text, but believe me, `dvii -s -n-1 test` will display all the `\special` text.

## 3.12 -p: display page information

To display a list of physical page/TEX page pairs, use the `-p` option:

```
dvii -p test

p:[1/1]
p:[2/2]
p:[3/3]
p:[4/4]
p:[5/-1]
p:[6/-3]
```

The first number indicates the physical page and the second number indicates the TEX page. For example, the sixth physical page has TEX page −3.

For more information on the difference between physical pages and TEX pages, see section 4.1.

## 3.13 `-P`: suppress the display of physical pages

(This option is available in version 0.43 and later.) This option only makes sense when you are listing page information (e.g., when using the `-p` or the `-M` option. Here is what you get with `dvii -p test`:

```
dvii -p test

p:[1/1]
p:[2/2]
p:[3/3]
p:[4/4]
p:[5/-1]
p:[6/-3]
```

and here is what you get when you add the `-P` option:

```
dvii -p -P test

p:[XX/1]
p:[XX/2]
p:[XX/3]
p:[XX/4]
p:[XX/-1]
p:[XX/-3]
```

So, why might you want to do something so silly? This is most useful when usnig the message digest feature to detect page changes (see Chapter 7). Imagine that you have message digest information for your entire 1000-page book. Later, you compile only one chapter. If you create a message digest of the chapter, the physical pages will no longer agree with the message digest for the entire book. So, suppressing the physical pages is the only way you can reliably tell if there has been a change.

## 3.14 `-s`: display specials

The `-s` lists each `\special` in the file and the page on which it appears. For example:

```
dvii -s test

s:[2/2]:: A short special
s:[3/3]:: abcdefghijklmnopqrstuvwxyzABCDE...
s:[4/4]:: PSfile 1.eps
s:[4/4]:: PSfile 2.eps
s:[4/4]:: PSfile 3.EPS
s:[4/4]:: PSfile dog1.gif
s:[4/4]:: PSfile cat.eps
s:[6/-3]:: Some control characters: []
```

Thus, there is a `\special` on (physical) page 2 consisting of the text '`A short special`'.

The `\special` text on page `6/-3` contains some control characters which do not show up on the printed page because they do not correspond to any printable characters. (Actually, the text is `[^A^B^C^D]`.)

## 3.15 -S: count number of specials

Normally, the summary that the `-u` option generates includes the number of pages and the number of fonts, but *not* the number of specials. If you want a count of the number of specials, use `-S`; note that `-S` implies `-u`

```
dvii -S test

File size: 1500 bytes (1 K)
Comment string:  TeX output 2001.01.25:0933
Page count: 6
Number of fonts: 3
Number of specials: 8
```

## 3.16 -T: give timing information

To see how long `dvii` took to run, use the `-T` option:

```
dvii -T -C test

dvi file 'test.dvi' passed validation check (level 1).
User time: 0.0 seconds
Real time: 0.0 seconds
```

The 'User time' is the time external to the computer to run; that is, if you started a stopwatch before executing the command and then stopped it after the command finished, the elapsed time corresponds to the 'User time'.

On the other hand, most modern operating systems are capable of running many programs at once, switching from one to the other. So, while it may appear to you that a command takes 5 minutes to run, the computer itself may have only spent 1 minute of the 5 minutes on your program, spending the other 4 minutes on other tasks. Hence, 1 minute would be 'Real time'.

If the file is small enough (as it is in the cast of `test.dvi` above), you may see both times listed as `0.0 seconds`.

## 3.17 -u: display summary

To display a summary of a dvi file, use the `-u` option:

```
dvii -u test.dvi
File size: 1500 bytes (1 K)
Comment string:  TeX output 1999.12.05:1951
Page count: 6
Number of fonts: 3
```

The summary tells us: the dvi `test.dvi` file is 1500 bytes long (roughly 1 kilobyte), was compiled on December 5, 1999 at 7:51 AM, has 6 pages, and contains 3 fonts.

See also the `-S` option.

# Chapter 4

# Pages

## 4.1 Physical pages versus TEX pages

Every time TEX writes a page to a dvi file it inserts codes indicating the beginning (`BOP`) and end (`EOP`) of the page. The number of *physical pages* is thus the number of `BOP`'s in the dvi file.

At the start of each physical page, TEX records additional information indicating what page number will appear when the dvi file is printed. (More precisely, it stores the contents of the first ten `\count` registers, the first of which is usually the register used to store the page number printed at the bottom of the page.) This page number is the *TEX page.*

Thus, it is possible to have a dvi file that when printed consists of 10 physical pieces of papers where each page has the number '1' printed at the bottom of the page.

For example, the file `test.dvi` consists of 6 physical pages, with TEX page numbers 1, 2, 3, 4, −1, −3.

```
dvii -p test
p:[1/1]
p:[2/2]
p:[3/3]
p:[4/4]
p:[5/-1]
p:[6/-3]
```

The first number in the bracket is the physical page while the second is the TEX page.

If you want to *suppress* the display of physical pages, use the `-P` option (see section 3.13).

# Chapter 5

# Fonts

Internally, TEX refers to each font via a font number. Each font number corresponds to an external font name and a scaling factor. For example, the file `test.dvi` calls upon 3 fonts: fonts 0, 23, and 50. In this case, font 0 refers to a font named `cmr10` scaled at magnification 1000.

`dvii` ouputs font information in the following format:

`f:[`*n*`/`*filename*`/`*scale*`]::`*checksum*

The *n* refers to the font number (e.g., 0), *filename* refers to the external file name (e.g., `cmr10`), *scale* refers to the scaling factor (e.g., 1000), and *checksum* to the checksum of the font's TFM file (e.g., `4bf16079`).

Here are the fonts in `test.dvi` as displayed by `dvii`:

```
dvii -f test

f:[50/cmr10/1200]::4bf16079
f:[23/cmbx10/1000]::1af22256
f:[0/cmr10/1000]::4bf16079
```

Note that TEX views two fonts with the same external fontname but with different scaling factors as different fonts.

# Chapter 6

# Specials

As not every publishing feature is built-in to TeX, the creator of TeX, Donald E. Knuth, made sure that a facility to insert "hooks" into a dvi file was available. These hooks are called *specials* and are invoked via tha `\special` command. For example, if you put the command `\special{Hello}` in a TeX file, the resulting dvi file will contain a `\special` opcode with argument `Hello`. TeX postprocessors (such as dvi viewers or printer drivers) can then interpret the `\special`'s and take the appropriate action.

One of the more common uses of the `\special` is to insert figure files into a TeX document. For example, the LaTeX `graphics` package provides the `\includegraphics` command which will take as an argument the name of figure file and place a `\special` command in the dvi file listing the name of the file, its dimensions, and other information. Altough TeX itself knows nothing about figure files, most dvi printer drivers (such as `dvips`) know how to interpret the `\special` information and will insert the figure.

You can tell `dvii` to go through a dvi file and list the location and contents of each `\special`. This can be useful, for instance, if you want to see on which page a particular figure appears without actually viewing or printing the file.

```
dvii -s test

s:[2/2]:: A short special
s:[3/3]:: abcdefghijklmnopqrstuvwxyzABCDE...
s:[4/4]:: PSfile 1.eps
s:[4/4]:: PSfile 2.eps
s:[4/4]:: PSfile 3.EPS
s:[4/4]:: PSfile dog1.gif
s:[4/4]:: PSfile cat.eps
s:[6/-3]:: Some control characters: []
```

# Chapter 7

# Message digest

## 7.1 What is a message digest?

A short answer: a *message digest* is a numerical summary or “fingerprint” of a given set of data (such as a data file or e-mail message).

A longer answer. A computer file is really just a sequence of numbers. For example, the file containing the single word “`Hello`” is really just the the sequence 72 101 108 108 111. The file containing the file “`Goodbye`” is the sequence 71 111 111 100 98 121 101. Of course, most computer files are much bigger with correspondingly longer sequences.

Imagine that a file with one million characters needs to be sent from one computer to another computer thousands of miles away. How can the recipient be assured that the data received is in fact the data that was sent? One way would be to send the data twice and compare the two transmissions. This might work, but would be rather slow: it would take twice as long to send the message twice.

So, instead of sending the message again a second time we send a shorter message that gives a summary of the original message. This summary is calculated at both ends, and the receiver compares her summary with the one sent by the transmitter. If they match, this is evidence that the file was received intact.

Of course, the form of this summary will affect how reliable the summary is at detecting if the data was sent correctly.

One sort of summary is called a *parity check*: add up all the numbers in the file and see if this sum is odd or even. The message digest is then either 0 (if the sum is even) or 1 (if the sum is odd). As you may imagine, this form of fingerprint is not very useful when the files get long.

Thus one desirable aspect of a message digest is that different data sets which differ in a small number of places generate different message digests. This way, changes to the data are readily detected.

One system that does this well is the MD5 message digest. It is "cryptographically secure", that is, if one file generates a given MD5 message digest, it is hard to construct a different file with the same MD5 message digest. Thus, it is suitable for "digital signatures" that are hard to forge.

## 7.2 Why use a message digest with TEX?

The last (and often trickiest) step in setting a book is "page layout". In the TEX world this corresponds to optimizing figure placement and fixing bad page breaks. This is often delicate: changing one line of text can change not just the page on which the line appears, but the pages following ("text reflow").

Thus, it would be very nice to determine, after making a small change, *which pages have changed*. One way to determine this is to look at a printout of each page and see which pages have changed. This, however, is not very practical except for the smallest documents.

My approach is to take each page's "finger print" (via a message digest) before making a change, make the change, and then take each page's fingerprint again after making the change. Then, just check which pages' fingerprints have changed.

Here is an example. Let `filex.tex` contain the following source:

```
a\eject b\eject c\eject d\eject e\eject f\end
```

The summary of this file is

```
dvii -u filex

File size: 628 bytes (1 K)
Comment string:  TeX output 1999.12.17:0937
Page count: 6
Number of fonts: 1
```

We can save and display the page information along with a message digest of each page if we use the `-M1` option:

```
dvii -p -M1 filex > before.md
cat before.md
[message digest: simple sum (ignore font)]
p:[1/1]::9C8E26458F1B019011D2F28DA18B18CC
p:[2/2]::9C8E26468F1B029011D2F28DA18B18CC
p:[3/3]::9C8E26478F1B039011D2F28DA18B18CC
p:[4/4]::9C8E26488F1B049011D2F28DA18B18CC
```

```
p:[5/5]::9C8E26498F1B059011D2F28DA18B18CC
p:[6/6]::9C8E264A8F1B069011D2F28DA18B18CC
```

We now edit `filex.tex` and change the '`b`' that appears on page 2 to a '`B`'; we now have

```
a\eject B\eject c\eject d\eject e\eject f\end
```

We next TeX the file, save the new fingerprints, and see which pages have changed using the `diff` utility:

```
tex filex
dvii -p -M1 filex > after.md
diff before.md after.md
3c3
< p:[2/2]::9C8E26468F1B029011D2F28DA18B18CC
---
> p:[2/2]::9C8E26468F1AE29011D2F28DA18B18CC
```

Note that page 2 is the only page with a different message digest.

# Appendix A

# Acknowledgements

- Heiko Oberdiek made significant suggestions on improving the performance of the code, and pointed out several errors.
- Tom Kacvinsky helped make the code work on 64-bit machines.
- Karsten Tinnefeld made some helpful suggestions on getting the code to compile on older Suns, and contributed some Linux and Sun binaries.

# Appendix B

# Source files

Here is the source for `test.tex`:

```
% test.tex
% Test file for dvii (last changed: 5 December 1999)
%
% A blank page
\eject

This is page 2/2.
\eject

This is page 3/3 with a short special. Also, a rule. \vrule width1cm
depth1cm height 1cm\relax
\special{A short special}
\eject

This is page 4/4 with a {\bf long} special. Note the control
character; this is meant to confuse dvii when it cuts of the long
special text.
\special{%
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789%
3.141592653589793238462643383279502884197169399375105820974944592%
The quick brown fox jumped over the lazy dog.%
}
\eject

This is page 5/5 with 5 specials.
\special{PSfile 1.eps}
\special{PSfile 2.eps}
\special{PSfile 3.EPS}
\special{PSfile dog1.gif}
\special{PSfile cat.eps}
\eject

\pageno = -1
\font\a=cmr10 scaled 1200
{\a This is page 6/-1 with font cmr10 scaled 1200.}
```

```
\eject

\pageno = -3
This page has a special with control characters. (Oooh!)
\special{Some control characters: []}
Followed by several blank pages.

\eject\eject\eject\eject
\bye
```

# Appendix C

# Perl scripts

Here I describe some Perl scripts that use the output from `dvii`.

## C.1 `fontdiff.pl`

### C.1.1 Introduction

The Perl script `fontdiff.pl` looks at two dvi files and compares the fonts used in each displying the differences. This can be useful when you have compiled a TEX file on two different computer systems and you are worried that the fonts on the two systems may be installed differently.

Let us look at an example.

We have two dvi files: `test1.dvi` and `test2.dvi`. Here are the fonts listings for these two files:

```
d:\tex\dvii>dvii -f test1.dvi
f:[50/cmbx10/1100]::1af22256
f:[33/cmsl10/1000]::70ae304a
f:[0/cmr10/1000]::4bf16079

d:\tex\dvii>dvii -f test2.dvi
f:[51/cmbx10/1100]::1af22256
f:[50/cmr10/1100]::4bf16079
f:[0/cmr10/1000]::4bf16079
```

If we run the Perl script `fontdiff.pl` on these two files we get the following:

```
d:\tex\dvii>perl fontdiff.pl test1.dvi test2.dvi
Fonts in test1.dvi NOT in test2.dvi:
------------------------------
  f:[NN/cmsl10/1000]::70ae304a
------------------------------

Fonts in test2.dvi NOT in test1.dvi:
```

```
------------------------------
  f:[NN/cmr10/1100]::4bf16079
------------------------------

Fonts in common that have DIFFERENT checksums:
------------------------------
------------------------------
```

Note that by default the `fontdiff.pl` script ignores font numbering when finding font differences. If you want to see which fonts are in common add the `-l` option:

```
d:\tex\dvii>perl fontdiff.pl -l test1.dvi test2.dvi
Fonts in test1.dvi NOT in test2.dvi:
NOTE: fonts marked with * are in BOTH files
------------------------------
* f:[NN/cmbx10/1100]::1af22256
  f:[NN/cmsl10/1000]::70ae304a
* f:[NN/cmr10/1000]::4bf16079
------------------------------

Fonts in test2.dvi NOT in test1.dvi:
NOTE: fonts marked with * are in BOTH files
------------------------------
* f:[NN/cmbx10/1100]::1af22256
  f:[NN/cmr10/1100]::4bf16079
* f:[NN/cmr10/1000]::4bf16079
------------------------------

Fonts in common that have DIFFERENT checksums:
------------------------------
------------------------------
```

### C.1.2 How are fonts "different"?

What does it mean for 2 TEX fonts to be ”different”? There are four characteristics of a TeX font called from a .dvi file:

1. name (e.g., cmr10 or ptmb)
2. font number
3. scaling (e.g., 1000 or 1200)
4. checksum

The Perl script `fontdiff.pl` can ignore any of these parameters (except font name) when finding font differences. Here are the options:

```
-c : ignore checksum when finding font differences
-C : do NOT ignore checksum when finding font differences (DEFAULT)
-n : ignore font number when finding font differences (DEFAULT)
-N : do NOT ignore font number when finding font differences
-s : ignore font scaling when finding font differences
-S : do NOT ignore font scaling when finding font differences (DEFAULT)
-l : list fonts for each file
```

Continuing the example started in the above section, if we *don't* want to ignore font numbering we would use the `-N` option:

```
d:\tex\dvii>perl fontdiff.pl -N test1.dvi test2.dvi
Fonts in test1.dvi NOT in test2.dvi:
------------------------------
  f:[50/cmbx10/1100]::1af22256
  f:[33/cmsl10/1000]::70ae304a
------------------------------

Fonts in test2.dvi NOT in test1.dvi:
------------------------------
  f:[51/cmbx10/1100]::1af22256
  f:[50/cmr10/1100]::4bf16079
------------------------------

Fonts in common that have DIFFERENT checksums:
------------------------------
------------------------------
```

Note that more fonts are considered different than before: in file `test1.dvi` the font `cmbx10` scaled at magnification 1000 has font number 50 while in `test2.dvi` that same font has font number 51.

*Note.* If you look back at the output of the example at the end of the previous section, you will see that the fonts are listed with their font numbers replace with `NN`; this is because, unless overridden by the `-N` option, font numbers are ignored and hence we do not even want to see them.

Here is the output when we ignore everything we can ignore: checksum, scaling, and font number:

```
d:\tex\dvii>perl fontdiff.pl -n -s -c test1.dvi test2.dvi
Fonts in test1.dvi NOT in test2.dvi:
------------------------------
  f:[NN/cmsl10/SSSS]::70ae304a
------------------------------

Fonts in test2.dvi NOT in test1.dvi:
------------------------------
------------------------------
```

## C.2 `specials.pl`

This Perl script takes as it single argument a dvi file and outputs a a dvips-style list of pages containing `\special`'s.

Let us look at what happens with our old friend `text.dvi`. Here is a list of the `\special`'s:

```
d:\tex\dvii>dvii -s test.dvi
s:[2/2]:: A short special
s:[3/3]::
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01234567893.141592
6535897932384626433832795028841971693993751058209749445592
s:[4/4]:: PSfile 1.eps
s:[4/4]:: PSfile 2.eps
s:[4/4]:: PSfile 3.EPS
s:[4/4]:: PSfile dog1.gif
s:[4/4]:: PSfile cat.eps
s:[6/-3]:: Some control characters: []
```

and here is the output from the `specials.pl` script:

```
d:\tex\dvii>perl specials.pl test.dvi
-pp2,3,4,-3
```

You could then take this string and print just the pages containing the `\special`'s using `dvips`:

```
d:\tex\dvii>dvips -pp2,3,4,-3 test.dvi
```

Although the most common `\special`'s encountered are those for including an EPS (Encapsulated Postscript File) figures, and in most cases those are the *only* `\specials`'s, it may happen that there are pages containing other kinds of `\specials`'s (for example when the document incorporates color). So, how can you get a list of just those pages that have included figures? The answer is to use the `--grep` option to display only those `\special`'s that contain the string `PSfile` (the standard indicator of an included EPS file). So, to print only those pages that contain an included EPS figure:

```
d:\tex\dvii>perl specials.pl --grep PSfile test.dvi
-pp4
```

## C.3 `details.pl`

The Perl script `details.pl` uses the dump opcode option `-d` of `dvii` (see section 3.3 to generate opcode statistics. For example, to list the op codes in order of a dvi file, type `details.pl -d` as in this example:

```
details.pl -d test

o:pre
o:bop
o:push
o:down3
...
o:post
```

where `...` are the intervening opcodes.

To get statistics on how many times each opcode is called, use the `-D` option

```
details.pl -D test

set_char_0:0
set_char_1:0
...
pop:25
right1:0
right2:7
...
```

(I have only shown a few of the 255 opcodes.)

Finally, if you want a succint summary of the opcodes used, use the `-u` option, or no option at all:

```
details.pl -u test

SUMMARY
Number of characters set (total) : 314
  at or below position 127       : 314
  above position 127             : 0
Number of rules                  : 1
Number of nops                   : 0
Number of bops                   : 6
Number of eops                   : 6
Number of font changes           : 9
Number of specials               : 8
Number of font definitions       : 3
Number of undefined opcodes      : 0
```

The “Number of characters set” reflects the total number of characters set, that is, if you were to type this document on a typewriter how many times you would have to hit a key. (Sort of.)

Note that strictly speaking this script could just about as easily use `dvitype` to do its work, but where is the fun in that?