FORSCHUNGSZENTRUM JÜLICH GmbH
Zentralinstitut für Angewandte Mathematik
D-52425 Jülich, Tel. (02461) 61–6402

Beratung und Betrieb, Tel. (02461) 61–6400

Technical Information

FZJ-ZAM-TKI-0286
Wolfgang Frings
03.02.98

PStools - working with PostScript files

1 Introduction

Working with PostScript files is sometimes difficult. The following tools should facilitate this task. Most of them have been written by Angus Duggan and are freeware. Some others have been written at KFA.

2 PSUtils

The utilities in this section are able to manipulate PostScript files containing multiple pages. Page selection and rearrangement are supported, including arrangement into signatures for booklet printing, and page merging for n-up printing.

A known bug of these tools is that they accept most of the DSC comments of the PostScript file but not all. These comments are introducted by '%%' e.g. '%%Page:'.

You can use the program *ghostview* to test whether the tools will work correctly on your PostScript file. If *ghostview* shows the file with the correct page numbering you can use the tools without problems. In the other case you will have to use one of the filters described in chapter 3 on page 5.

The following programs exist:

psbook	rearranges pages in PostScript file into signatures
psresize	alters document paper size
psselect	selects pages and page ranges from a PostScript file
psnup	puts multiple pages onto each physical sheet of paper
pstops	performs general page rearrangement and selection in a PostScript file

Contents

2.1 psbook

Syntax

```
psbook [-q ] [-ssignature] [ infile [outfile ] ]
```

Description

psbook rearranges pages from a PostScript document into ”‘signatures”’ for printing books or booklets, creating a new PostScript file. The input PostScript file should follow the Adobe Document Structuring Conventions. The *- s* option selects the size of signature which will be used. The signature size is the number of sides which will be folded and bound together; the number given should be a multiple of four. The default is to use one signature for the whole file. Extra blank sides will be added if the file does not contain a multiple of four pages. Psbook normally prints the page numbers of the pages rearranged; the *- q* option suppresses this.

Example The following command can be used to rearrange the pages of a document:

```
psbook - 4 in.ps out.ps
```

Then the page ordering is as follows if the document contained 7 pages:

* 1 2 7 6 3 4 5

2.2 psresize

Syntax

```
psresize [-w width] [ -hheight ] [ -p paper] [ -W Width] [ -HHeight ] [ -P
Paper] [ -q ] [ infile [ outfile ] ]
```

Description *psresize* rescales and centres a document on a different size of paper. The input PostScript file should follow the Adobe Document Structuring Conventions. The **-w** option gives the output paper width, and the **-h** option gives the output paper height, normally specified in **cm** or **in** to convert PostScript’s points (1/72 of an inch) to centimeters or inches. The **-p** option can be used as an alternative, to set the output paper size to **a3, a4, a5, b5, letter, legal,** or **10x14.** The default output paper size is **a4**. The **-W** option gives the input paper width, and the **-H** option gives the input paper height. The **-P** option can be used as an alternative, so set the input paper size. The default input paper size is **a4**. Psresize normally prints the page numbers of the pages rearranged; the **-q** option suppresses this.

Examples: The following command can be used to convert a document from size A4 to letter size:

```
psresize -Pa4 -pletter in.ps out.ps
```

If you have a DIN A4-sized document and want to increase the size to 40 cm x 50 cm you have to use the following command:

```
psresize -w40cm -h50cm -Pa4 in.ps out.ps
```

2.3 psselect

Syntax

```
psselect [ -q ] [ -e ] [ -o ] [ -r ] [-ppages] [ pages] [ infile [outfile ] ]
```

Description

psselect selects pages from a PostScript document, creating a new PostScript file. The input PostScript file should follow the Adobe Document Structuring Conventions. The **-e** option selects all of the even pages; it may be used in conjunction with the other page selection options to select the even pages from a range of pages. The **-o** option selects all of the odd pages; it may be used in conjunction with the other page selection options. The **-ppages** option specifies the pages which are to be selected. *Pages* is a comma separated list of page ranges, each of which may be a page number, or a page range of the form ***first-last***. If ***first*** is omitted, the first page is assumed, and if ***last*** is omitted, the last page is assumed. The prefix character '_' indicates that the page number is relative to the end of the document, counting backwards. The **-r** option causes **psselect** to output the selected pages in reverse order. Psselect normally prints the page numbers of the pages rearranged; the **-q** option suppresses this. If any of the **-r, -e,** or **-o** options are specified, the page range must be given with the **-p** option. This is for backwards compatibility with previous versions.

Example

To select the pages 3-7 from a document containing 12 pages use:

```
   psselect  -p3-7  in.ps out.ps
```

Note:

The page number given to **psselect** refers to a numbering starting with one at the beginning or end of the file. The actual page number in the document may be different.

2.4 psnup

Syntax

```
psnup [ -w width ] [ -hheight ] [ -p paper ] [ -l ] [ -r ] [ -f ] [ -c ] [-mmargin ]
[-bborder ] [-dlwidth ] [ -sscale ] [ nup ] [ -q ] [ pages] [ infile [outfile ] ]
```

Description

psnup puts multiple logical pages onto each physical sheet of paper. The input PostScript file should follow the Adobe Document Structuring Conventions. The **-w** option gives the paper width, an the **-h** option gives the paper height, normally specified in **cm** or **in** to convert PostScript's points (1/72 of an inch) to centimeters or inches. The **-p** option can be used as an alternative, to set the paper size to **a3, a4, a5, b5, letter, legal,** or **10x14.** The default paper size is **a4**. **psnup** normally uses 'row-major' layout, where adjacent pages are placed in rows across the paper. The **-c** option changes the order to 'column-major', where successive pages are placed in columns down the paper. A margin to leave around the whole page can be specified with the **-m** option. This is useful for sheets of 'thumbnail' pages, because the normal page margins are reduced by putting multiple pages on a single sheet. The **-b** option is used to specify an additional margin around each page on a sheet. The **-d** option draws a line around the border of each page, of the specified width. If the ***lwidth*** parameter is omitted, a default linewidth of 1 point is assumed. The linewidth is relative to the original page dimensions, *i.e.* it is scaled down with the rest of the page. The scale chosen by **psnup** can be overridden with the **-s** option. This is useful to merge pages which are already reduced. The **-nup** option selects the number of logical pages to put on each sheet of pater. This can be any whole number; **psnup** tries to optimise the layout so that the minimum amount of space is wasted. If **psnup** cannot find a layout within its tolerance limit, it will abort with an error message. The alternative form **-inup** can also be used, for compatibility with other n-up programs, where **i** is the number of pages. Psselect normally prints the page numbers of the pages rearranged; the **-q** option suppresses this.

Example

The use of this utility may vary but two particular applications can be seen in conjunction with *psbook(1).* For example, using lpr as the UNIX print spooler a typical command line might look like this:

```
   psbook file | psnup -2 | lpr Printername
```

If `file` is a 4 page document this command will result in a two page document printing two pages of `file` per page and it will rearrange the page order to match the input pages 4 and 1 on the first output page and pages 2 and 3 of the input document on the second output page.

Another frequent use of this tool is arranging 4 pages of a document on one sheet of paper, e.g. for course foils. This can be done with the following command:

```
psnup -4 in.ps out.ps
```

2.5 pstops

Syntax

```
pstops [ -q ] [ -b ] [ -w width ] [ -h heigth ] [-dlwidth] pagespecs] [ infile
[outfile ] ]
```

Description

`pstops` rearranges pages from a PostScript document, creating a new PostScript file. The input PostScript file should follow the Adobe Document Structuring Conventions. **`pstops`** can be used to perform a large number of arbitrary re-arrangements of Documents, including arranging for printing 2-up, 4-up, booklets, reversing, selecting front or back sides of documents, scaling, etc. pagespecs follow the syntax:

pagespecs	= [***modulo***:] ***specs***
specs	= ***spec*** [+***specs***] [,***specs***]
spec	= [-] ***pageno*** [**L**] [**R**] [**U**] [***@scale***] [**(*xoff*,*yoff*)**]

modulo is the number of pages in each block. The value of modulo should be greater than 0; the default value is 1. ***specs*** are the page specifications for the pages in each block. The value of the ***pageno*** in each spec should be between 0 (for the first page in the block) and ***modulo***-1 (for the last page in each block) inclusive. The optional dimensions ***xoff*** and ***yoff*** shift the page by the specified amount. ***xoff*** and ***yoff*** are in PostScript's points, but may be followed by the units "cm" or "in" to convert to centimetres or inches, or the flag "w" or "h" to specify as a multiple of the width or height.The optional parameters **`L, R,`** and **`U`** rotate the page left, right, or upside-down. The optional ***scale*** parameter scales the page by the fraction specified. If the optional minus sign is specified, the page is relative to the end of the document, instead of the start. If ***pagespecs*** are separated by **`+`** the pages will be merged into one page; if they are separated by **`,`** they will be on separate pages. If there is only one page specification, with ***pageno*** zero, the ***pageno*** may be omitted. The shift, rotation, and scaling are performed in that order regardless of which order they appear on the command line.The **`-w`** option gives the width which is used by the "w" dimension specifier, and the **`-h`** option gives the height which is used by the "h" dimension specifier. These dimensions are also used (after scaling) to set the clipping path for each page. The **`-p`** option can be used as an alternative, to set the paper size to **`a3, a4, a5, b5, letter, legal, tabloid, statement, executive, folio, quarto`** or **`10x14`**. The default paper size is **`a4`**. The **`-b`** option prevents any bind operators in the PostScript prolog from binding. This may be needed in cases where complex multi-page re-arrangements are being done. The **`-d`** option draws a line around the border of each page, of the specified width. If the ***lwidth*** parameter is omitted, a default linewidth of 1 point is assumed. The linewidth is relative to the original page dimensions, i.e. it is scaled up or down with the rest of the page. Pstops normally prints the page numbers of the pages re-arranged; the **`-q`** option suppresses this.

Examples

To put two pages on one sheet (of A4 paper), the pagespec to use is:

```
   pstops in.ps '2:0L@.7(21cm,0)+1L@.7(21cm,14.85cm)' out.ps
```

To select all of the odd pages in reverse order, use:

```
      pstops in.ps '2:-0' out.ps
```

To rearrange pages for printing 2-up booklets, use

```
    psbook in.ps '4:-3L@.7(21cm,0)+0L@.7(21cm,14.85cm)' out.front.ps
```

for the front sides, and

```
    pstops in.ps '4:1L@.7(21cm,0)+-2L@.7(21cm,14.85cm)' out.reverse.ps
```

for the reverse sides.

To rearrange pages for printing 2-up booklets for duplex printing, use

```
    psbook file | psnup -2 | pstops
    '2:0,U1(21cm,29.7cm)' > out.ps
```

Note:

If you use the ArborText Publisher to create PostScript files you should set the '**Output structure'** to '**Page**'.

3 Filters

The filters are written in Perl so they can be used on all Unix systems supported by ZAM.

These filters "‘fix"’ PostScript files output by different programs so that they work correctly with the above tools. It depends on the producing program which filter is to be used (see table below). Files from dvips (TEX) do not require filtering.

Call the filters as follows:

```
  name_of_filter < source.ps > fixed.ps
```

`name_of_filter`	**program whose output is fixed**
fixpspps	PSPrint
fixfmps	Framemaker
fixdlsrps	DviLaser/PS (e.q. Publisher)
fixpsditps	Transcript psdit (e.q. Man Pages)
fixwwps	Windows Write
fixwpps	WordPerfect 5.0 and 5.1
fixwfwps	Word f"ur Windows
fixmacps	Macintosh PC Programme
fixtpps	Tpscript

4 Programs for working with EPS files

The following tools can manipulate encapsulated PostScript files (EPS).

`epsffit`	fits encapsulated PostScript file (EPS) into prescibed size
`epsscale`	scales EPS file
`epsrot`	rotates EPS file
`makebb`	computes a missing bounding box for an EPS file

4.1 epsffit

Syntax

```
epsffit [ -c ] [ -r ] [ -a ] [ -m ] [ -s ] llx lly urx ury [ file ]
```

Description

`epsffit` fits an EPSF file (encapsulated PostScript) to a given bounding box. The coordinates of the box are given by (***`llx,lly`***) for the lower left, and (***`urx,ury`***) for the upper right, in PostScript units (points). If no file is specified, **`epsffit`** uses the standard input. Output is always to the standard output.

Options

`-c`	Center the image in the given bounding box.
`-r`	Rotate the image by 90 degrees counter-clockwise.
`-a`	Adjust the aspect ratio to fit the bounding box. The default is to preserve the aspect ratio.
`-m`	Rotates the image to maximise the size if it would fit the specified bounding box better.
`-s`	Add a *showpage* at the end of the file to force the image to print.

Example

If you have got an EPS file with the bounding box *0 0 100 200* and you want to have this image maximised to DIN A4, centered and rotated by 90 degrees you have to use the command:

```
epsffit -c -m 0 0 595 842 in.eps out.eps
```

Note:

The heigth of the image is maximised in the given bounding box while the width is scaled relatively. If the **`-r`** option is specified it is the other way round.

4.2 epsscale

Syntax

```
epsscale [ -A DIN_value ] [ -x g ] [ -y g ] [ -xf f ] [ -yf f ] infile [ outfile ]
```

Description

`epsscale` scales an EPS file into a given size. Standard Output is *`fn.1.ps`* .

Options

`-A`	Scales the image to a given DIN paper size from A0 ... A5.
`-x, -y`	Scales the x and/or y extension to the specified size ***g*** in cm.
`-xf, -yf`	Scales the x and/or y extension in the specified relation to the origin size.

Example

If you have got an EPS file with the size 10 cm x 20 cm and you want to double the size you have to use the command:

```
epsscale -xf2 in.eps out.eps
```

Note:

If only one extension is specified the other extension is scaled proportionally.

4.3 epsrot

Syntax

```
epsrot [ -90 — -180 — -270] infile [ outfile ]
```

Description

epsrot rotates an EPS file by a given angle counterclockwise. The default angle is 90. Standard output is *fn.angle.ps*.

Example

If you have got an EPS file that you want to rotate by 180 degrees you have to use the command:

```
   epsrot -180 in.eps out.eps
```

4.4 makebb

Syntax

```
makebb [ -x g ] [ -y g ] infile [outfile ]
```

Description

makebb computes a missing bounding box for an EPS file Standard output is *fn.1.ps*.

Options

If the image is bigger or smaller than DIN A 4 (21 cm x 27.9 cm), the expected size of the image must be specified by the **-x** and/or **-y** option (***g*** is the size in cm). With the help of Ghostscript **makebb** converts the given PostScript image into a bitmap (ppm format). With this file and an additional netpbm tool **pnmcrop** **makebb** is able to compute the real size of the image and thereby the bounding box. After that the bounding box comment is created in the output file.

Example

file1.eps is an EPS file with DIN A4 size but rotated by 90 degrees . For this image the bounding box comment is created with the following command:

```
   makebb -x30 in.eps out.eps
```

Note:

If one or both of the options **-x, -y** are omitted the corresponding size is assumed to be DIN A4. There could be tiny inaccuracies if the image has got a size different from DIN A4.

5 Converters between file formats

Sometimes you have to convert a file between formats like EPS, PostScript or ASCII for printing or viewing. The following tools handle these problems:

a2ps	formats an ASCII file to print on a PostScript printer
ps2epsi	converts PostScript to EPSI
ps2ascii	extracts ASCII text from a PostScript file

5.1 a2ps

Syntax

```
a2ps [ options] file1 [file2 ...]
```

Description

a2ps formats named files for printing in a PostScript printer; if no file is given, **a2ps** reads from the standard input. The output may be sent to the printer or to the standard output or saved into a file.

The format used is nice and compact: normally two pages on each physical page, borders surrounding pages, headers with useful information (page number, printing date, file name or supplied header), line numbering, pretty-printing, symbol substitution etc. This is very useful for making archive listings of programs.

Compared to the previous versions, **a2ps** version 4.8.4 provides:

1. totally different options, to be closer to that of `GNU enscript`,
2. various configuration files
3. some powerful meta-sequences to define the headers the way you want
4. a rather strong support of Latin 2, 3, 4, 5 and 6 encodings,
5. fully customizable output style: fonts (in the limit of the 13 standard fonts), background and foreground colors, line numbering style etc,
6. and of course, the ability to pretty-print sources written in quite a few various languages.

How to print

To print a file `doc.txt`, just give it to `a2ps`: the default setting should be the one you'd like:

```
a2ps doc.txt
```

`a2ps` sends the file `doc.txt` to the default printer, writing two columns of text on a single face of the sheet. Indeed, by default `a2ps` does as if you had given the option **`-2`**, standing for two virtual pages.

Say now that you want to print the C file `bar.c`, and its header `foo.h`, on 4 virtual pages, and save it into the file `foobar.ps`. Just hit:

```
a2ps foo.h bar.c -4ofoobar.ps
```

The option **`-4`** tells `a2ps` to make four virtual pages: two rows by two columns. The option **`-o`** (which is the short version of **`--output=foobar.ps`**) specifies the output file. Since the option **`-4`** does not have arguments, it can be grouped behind the same **`-`** as the one for **`-o`**. The other way round would fail, producing a file `foobar.ps4`.

Note too that the options may be before or after the files, it does not matter.

If you send `foobar.ps` to a printer, you'll discover that the keywords were highlighted, that the strings and comments have a different face. Indeed, `a2ps` is a pretty-printer: if ever it knows the (programming) language in which your file is written, it will try to make it nice looking on the paper.

But too bad: `foo.h` is only one virtual page long, and `bar.c` takes three. Moreover, the comments are essential in those files. And even worse: the system's default printer is out of ink. Thanks god, precious options may help you:

```
a2ps -A4 foo.h bar.c --prologue=gray -P lw
```

Here **`-A`** is used to specify that you Allow `a2ps` to put several files on the same sheet, **`-P lw`** means to print on the printer named **`lw`**, and finally, the long option **`--prologue`** asked to use one of the alternative printing styles of `a2ps`. There are other available prologues, and you can even design yours.

If `a2ps` did not have the behavior expected, this may be because of the default settings given by your system administrator. Checking those default values is easy:

```
    a2ps --list-options
```

Remember that the on line help is always available. Moreover, if your screen is small, you may *pipe* it into **`more`**. Just trust this:

```
    a2ps --help | more
```

Options for Global behavior:

`-q, --quiet, --silent`	be really quiet
`-v, --verbose`	tell what we are doing
`-V, --version`	print version and exit successfully
`-=`***`shortcut`***`,` `--user-option=`***`shortcut`***	use the ***`shortcut`*** defined by the user.
`--report[=`***`language`***`]`	generate PreScript report on all/***`language`*** style. This file must be later processed by `a2ps` using the ***`PreScript`*** style.
`--guess`	`a2ps` won't print anything, but will act as `file` does: it returns for each file the language in which `a2ps` believes the file is written. This can be very useful on broken systems to understand why a file is written with a bad style sheet.
`-h, --help`	print a short help, and exit successfully.
`--list-options`	Give a extensive report on `a2ps` configuration and installation.
`--list-features`	Known medias, encodings, languages, prologues, printers and user option shortcuts are reported.

Options for Sheets:

`-M, --media=`***`media`***	use output media ***`media`***. Currently ***`media`*** may be `A3`, `A4`, `A5`, `B4`, `B5`, `Letter`, `Legal`, `Tabloid`, `Ledger`, `Statement`, `Executive`, `Folio`, `Quarto`, `10x14`. `A4dj`, `Letterdj` are also defined for Desk Jet owners, since that printer needs bigger margins.
`-r, --landscape`	print in landscape mode
`-R, --portrait`	print in portrait mode
`--columns=`***`num`***	specify the number of columns of virtual pages per physical page.
`--rows=`***`num`***	specify the number of rows of virtual pages per physical page.
`--major=`***`direction`***	specify whether the virtual pages should be first filled in rows (***`direction`*** = `rows`) or in columns (***`direction`*** = `columns`).
`-1`	Predefined layout: 1 x 1 portrait, 80 chars/line, major rows (i.e. alias for **`--columns=1 --rows=1 --portrait --chars-per-line=80 --major=rows`**).
`-2`	2 x 1 landscape, 80 chars/line, major rows.
`-3`	3 x 1 landscape, major rows.
`-4`	2 x 2 portrait, 80 chars/line, major rows.
`-5`	5 x 1 landscape, major rows.
`-6`	3 x 2 landscape, 80 chars/line, major rows.
`-7`	7 x 1 landscape, major rows.
`-8`	4 x 2 landscape, 80 chars/line, major rows.
`-9`	3 x 3 portrait, 80 chars/line, major rows.
`-j, --borders=`***`bool`***	print borders around virtual pages.
`-A, --compact=`***`bool`***	Compact mode for a sequence of files. This option allows the printing of more than one file in the same physical page. This option makes sense only when the number of virtual pages is greater than 1. Otherwise, the beginning of each file will be printed in a new sheet.
`--margin[=`***`num`***`]`	Specify the size of the margin (***`num`*** PostScript points, or 12 points

without arguments) to leave in the inside (i.e. left for the front side page, and right for the back side). This is intended to ease the binding.

Options for Pages:

`--line-numbers=`***`number`*** precede each ***`number`*** line with its line number.

`-C` Alias for **`--line-numbers=5`**.

`-f, --fontsize=`***`size`*** scale font to ***`size`*** for body text. ***`size`*** is a float number. To change the base font, change the current prolog.

`-l`***`num`***`, --chars-per-line=`***`num`*** Set the number of columns per page. ***`num`*** is the real number of columns devoted to the body of the text, i.e., no matter whether lines are numbered or not.

`-L`***`num`***`, --lines-per-page=`***`num`*** Set the lines per virtual page for printing. The font size is automatically scaled up to fill in the whole page. This is useful for printing preformatted documents which have a fixed number of lines per page. The scaling of the font size will be suppressed if this option is used with option `--font`. The minimum number of lines per page is set at 40 and maximum is at 160. If a number less than 40 is supplied, scaling will be turned off and a warning message is printed on the terminal.

`-m, --catman` Understand UNIX manual output ie: 66 lines per page and possible bolding and underlining sequences. The understanding of bolding and underlining is there by default even if `--catman` is not specified.

`-T`***`num`***`,--tabsize=`***`num`*** set tabulator size to ***`num`***. This option is ignored if **`--interpret=no`** is given.

`--non-printable-format=`***`format`*** specify how non-printable chars are printed. ***`format`*** can be `caret` (`^A`, `M-^B` etc.), or `space` (a space is written instead of the non-printable character).

Options for Headers:

`-b[`***`text`***`]`
`--header[=`***`text`*** set the page header

`-B, --no-header` no page headers at all.

`--center-title[=`***`text`***`]`
`--left-title[=`***`text`***`]`
`--right-title[=`***`text`***`]` Set virtual page center, left and right titles to ***`text`***.

`-u[`***`text`***`]`
`--underlay[=`***`text`***`]` use ***`text`*** as *under lay* (or *water mark*), i.e., in a light gray, and under every page.

`--left-footer[=`***`text`***`]`
`--footer[=`***`text`***`]`
`--right-footer[=`***`text`***`]` Set sheet footers to ***`text`***.

Options for Input:

`-c, --truncate-lines=`***`bool`*** Cut lines too large to be printed inside the borders. The maximum line size depends on format and font size used and whether line numbering is enabled.

`-i, --interpret=`***`bool`*** interpret tab, bs and ff chars

`-X`***`name`***`, --encoding=`***`name`*** use input encoding ***`name`***. Currently `a2ps` supports:

`ascii`	Regular ASCII code.
`latin1`	Which real ISO name is ISO8859-1.
`latin2`	`(ISO8859-2)`
`latin3`	`(ISO8859-3)`
`latin4`	`(ISO8859-4)`

`latin5 (ISO8859-9)`
`latin6 (ISO8859-10)`
The support for these encodings is provided thanks to *Ogonkify*. Ogonkify is a tool that lets you extend the font support for most encodings, i.e., for most non-western-Europe alphabets. If you think the support is not good enough, you may help the extension of Ogonkify.

`hp8` HP encoding The 8 bits Roman encoding for HP.

`mac` Macintosh encoding For the Macintosh encoding. The support is not sufficient, and a lot of character may be missing at the end of the job. This is planed to be fixed.

`ibmpc` PC encoding For the regular PC encoding. Not all the characters will be printed.

`pcg` PC graphic encoding For the PC Graphic encoding.

`cp1250` CP-1250 MicroSoft's CP-1250 encoding (aka CeP).

`-t` ***`filename`***`, --title=`***`filename`*** Give the name ***`filename`*** to the files read trough the standard input.

`--prologue=`***`prologue`*** Use ***`prologue`*** as the PostScript prologue for `a2ps`. ***`prologue`*** must be in a file named ***`prologue`***`.pro`, which be in a directory of your library path. Currently the available prologues are:

`bw` (black and white) the regular presentation

`gray` same presentation, but with gray shades for comments etc.

`gray2` same as gray, but light declaration (such as function calls etc.) are mapped on the same shape as the strong declarations (such as function declaration etc.).

`color` still the same idea, but with colors.

`--print-anyway=`***`bool`*** force binary printing. By default, a whole print job is stopped as soon as a binary file is detected. To detect such a file we make use of a very simple heuristic: if the first sheet of the file contains more than 40% of non-printing characters, it's a binary file.

Options for Output:

`-o`***`file`***`, --output=`***`file`*** leave output to file ***`file`***. If ***`file`*** is `-`, leave output to the standard output.

`-n`***`num`***`, --copies=`***`num`*** print ***`num`*** copies of each page

`-D, --setpagedevice[=`***`key`***`:`***`value`***`]` Pass a page device definition to the generated PostScript output. If no ***`value`*** is given, ***`key`*** is removed from the definitions.

For example, command

```
      a2ps -DDuplex:true report.pre
```

prints file `report.pre` in duplex (two side) mode.

Page device operators are implementation dependent but they are standardized.

`-S, --statusdict=`***`key`***`[:`***`value`***`], --statusdict=`***`key`***`[::`***`value`***`]` Pass a statusdict definition to the generated PostScript output. `statusdict` operators and variables are implementation dependent; see printer's documentation for details.

If no ***`value`*** is given, ***`key`*** is removed from the definitions.

With a single colon, pass a call to an operator, for instance:

```
      a2ps -Ssetpapertray:1 quicksort.c
```

prints file `quicksort.c` by using paper from the paper tray 1 (assuming that printer supports paper tray selection).

With two colons, define variable ***key*** to equal ***value***. For instance: `a2ps -Spapertray::1 quicksort.c` produces `/papertray 1 def` in the PostScript.

`-k, --page-prefeed` enable page prefeeding. It consists in positioning the sheet in the printing area while the PostScript is interpreted (instead of waiting the end of the interpretation of the page before pushing the sheet). It can lead to an significant speed up of the printing.

`a2ps` quotes the access to that feature, so that non supporting printers won't fail.

`-K, --no-page-prefeed` disable page prefeeding.

`-P` ***name***`, --printer=`***name*** send output to printer ***name***.

`-d` send output to the default printer.

`-s`***num***`, --side=`***num*** number of sheet sides. ***num*** is of course 1 or 2.

Options for Pretty printing:

`-g, --graphic-symbols=`***bool*** set symbols graphical conversion

`-E`***language***`, --pretty-print=`***language*** use the ***language*** pretty print style sheet.

`-K, --automatic-style=`***bool*** `a2ps` will try to guess in what language is written your files, and use the corresponding style sheet.

`--strip-level=`***num*** Depending on the value of ***num***:

0 everything is printed;
1 regular comments are not printed
2 strong coments are not printed
3 every comment is canceled.

This option is valuable for instance in `java` in which case strong comments are the so called documentation comments, or in `SDL` for which some graphical editors pollutes the specification with internal data as comments.

Note that the current implementation is not satisfactory: some undesired blank lines remain. This is planed to be fixed.

Meta sequences:

Headers, footers, titles and the water mark are strings in which some meta sequences are later replaced by their actual values. In general `%` are related to general information, while `$` meta sequences are related to the current file.

On a new sheet `a2ps` first draws the water mark, then the content of the first page, then the frame of the first page, (ditto with the others), and finally the sheet header and footers. This order must be taken into account for some meta sequences (e.g., `$L`, `$l`).

Currently `a2ps` support the following meta sequences:

`%%` character `%`

`$$` character `$`

`%?#`***cond***`#`***if_true***`#`***if_false***`#` this may be used for conditional assignment. The separator (presented here as `#`) may be any character. ***if_true*** and ***if_false*** may be defined exactly the same way as regular headers, included meta sequences and the `$?` construct.

cond may be:

`l` true if printing in landscape mode,

`v` true if printing on the back side of the sheet (verso).

`1, 2, or 3` true if tag 1, 2 or 3 is not empty. See item `$t1` for explanation.

`$(`***`var`***`)`	value of the environment variable ***`var`***
`%a`	the localized equivalent for `Printed by` ***`User Name`***
`%A`	the localized equivalent for `Printed by` ***`User Name`*** `from` ***`Host Name`***
`%a{`***`username`***`}`	the localized equivalent for `Printed by` ***`username`***
`%A{`***`username@hostname`***`}`	the localized equivalent for `Printed by` ***`username`*** `from` ***`hostname`***. These two are provided in the case `a2ps` would be used by the print service: since neither of the user name nor the host name can be known at the time the files reach `a2ps`, these options should be used.
`%c`	trailing component of the current working directory
`%C`	current time in `hh:mm:ss` format
`$C`	file modification time in `hh:mm:ss` format
`%d`	current working directory
`%D`	current date in `yy-mm-dd` format
`$D`	file modification date in `yy-mm-dd` format
`%D{`***`string`***`}`	format current date according to ***`string`*** with the `strftime(3)` function.
`$D{`***`string`***`}`	format file's last modification date according to ***`string`*** with the `strftime(3)` function.
`%e`	current date in localized short format (e.g., `Jul 4, 76` in English, or `14 Juil 89` in French).
`$e`	file modification date in localized short format.
`%E`	current date in localized long format (e.g., `July 4, 76` in English, or `Samedi 14 Juillet 89` in French).
`$E`	file modification date in localized long format.
`%F`	current date in `dd.mm.yyyy` format.
`$F`	file modification date in `dd.mm.yyyy` format.
`%l`	top most line number of the current page
`$l`	current line number. To print the page number and the line interval in the right title, use `--right-title="$q:%l-$l"`.
`$L`	number of lines in the current file.
`%m`	the host name up to the first `.` character
`%M`	the full host name
`%n`	the user login name
`$n`	input file name without the directory part
`%N`	the user's pw_gecos field up to the first `,` character
`$N`	the full input file name
`%p`	current page number
`$p`	page number for this file
`%P`	total number of pages printed
`$P`	number of pages of the current file
`%q`	localized equivalent for `Page %p`
`$q`	localized equivalent for `Page $p`
`%Q`	localized equivalent for `Page %p/%P`
`$Q`	localized equivalent for `Page $p/$P`
`%s`	current sheet number
`$s`	sheet number for the current file
`%S`	total number of sheets
`$S`	number of sheet of the current file
`%t`	current time in 12-hour am/pm format
`$t`	file modification time in 12-hour am/pm format
`$t1, $t2, $t3`	Content of tag 1, 2 and 3. Tags are pieces of text `a2ps` fetches in the files,

	according to the style. For instance, in `mail-folder` style, tag 1 is the title of the mail, and tag 2 its author.
`%T`	current time in 24-hour format `hh:mm`
`$T`	file modification time in 24-hour format `hh:mm`
`%*`	current time in 24-hour format with seconds `hh:mm:ss`
`$*`	file modification time in 24-hour format with seconds `hh:mm:ss`
`$v`	the sequence number of the current input file
`$V`	the total number of files
`%W`	current date in `mm/dd/yy` format
`$W`	file modification date in `mm/dd/yy` format
	All format directives can also be given in format

Examples:

Attention: Per default **`a2ps`** puts **two** logical pages on **one** physical.

```
a2ps test.txt
```

- sends the output to the printer specified in $PRINTER; two logical pages on one physical page; 80 chars per line

```
a2ps -1 -o test.ps test.txt
```

- generates the PostScript file `test.ps`; one logical page on a physical page; 80 chars per line

```
a2ps -1 -l88 test.txt
```

- sends the output to the printer with 88 chars per line; layout like older versions of a2ps

```
a2ps -1 -r -l132 test.txt
```

- sends the output to the printer with landscape layout; one logical page on a physical page; 132 chars per line

```
a2ps -B -1 --border=no -o test.ps test.txt
```

- generates the PostScript file `test.ps`: no border; no header; one logical page on a physical page; 80 chars per line

```
a2ps -A -Eperl -o listing.ps perldat1 perldat2 perldat3
```

- generates the PostScript file `listing.ps`: pretty print (Perl); two logical pages on a physical page; compact mode (more than one file on the same physical page)

Help and futher documentation

`a2ps --help`
: gives a list of all available options

`/usr/local/pstools/doc/`
: contains DVI and PostScript files of the users manual

5.2 ps2epsi

Syntax

```
ps2epsi infile.ps [outfile.epsi ]
```

Description

`ps2epsi` is a utility, based on Ghostscript, which takes an input PostScript file and generates a new output file which conforms to Adobe's 'Encapsulated PostScript Interchange' or EPSI format. This is a special form of encapsulated PostScript (EPS) which adds a bitmap version of the finally displayed page (in the form of PostScript comments) to the beginning of the file. This bitmap can be used by programs which understand EPSI (i.e. Publisher) to give a preview version of the PostScript on screen. The displayed quality is often not very good (eg. low resolution, no colours), but the final printed version uses the 'real' PostScript, and thus has the normal full PostScript quality.

Example:

If you want to convert the PostScript file *`file.ps`* to EPSI you have to use:

```
   ps2epsi file.ps > file.epsi
```

5.3 ps2ascii

Syntax

```
ps2ascii [ infile.ps [ outfile.txt ]
```

Description

`ps2ascii` is a shellscript based on ghostscript to extract ASCII text from a PostScript file. If *`outfile.txt`* is omitted the output goes to stdout. If both *`infile.ps`* and *`outfile.txt`* are omitted, ps2ascii acts as a filter, reading from stdin and writing on stdout.

Example:

If you want to convert the PostScript file *`file.ps`* to ASCII you have to use:

```
   ps2ascii file.ps > file.ascii
```