NAME

pbmtext - render text into a bitmap

SYNOPSIS

pbmtext [**−font** *fontfile*] [**−builtin** *fontname*] [**−space** *pixels*] [**−lspace** *pixels*] [*text*]

DESCRIPTION

Takes the specified text, either a single line from the command line or multiple lines from standard input, and renders it into a bitmap.

In the bitmap, each line of input is a line of output. Formatting characters such as newline have no effect on the formatting; like any unprintable character, they turn into spaces.

The bitmap is just wide enough for the longest line of text, plus margins, and just high enough to contain the lines of text, plus margins. The left and right margins are twice the width of the widest character in the font; the top and bottom margins are the height of the tallest character in the font. But if the text is only one line, all the margins are half of this.

OPTIONS

-font,**-builtin**

By default, pbmtext uses a built-in font called bdf (about a 10 point Times-Roman font). You can use a fixed width font by specifying **-builtin fixed**.

You can also specify your own font with the **-font** flag. The *fontfile* is either a BDF file from the X window system or a PBM file.

If the *fontfile* is a PBM file, it is created in a very specific way. In your window system of choice, display the following text in the desired (fixed-width) font:

```
M ",/^_[`jpqy| M

/  !"#$%&'()*+ /
< ,-./01234567 <
> 89:;<=>?@ABC >
@ DEFGHIJKLMNO @
_ PQRSTUVWXYZ[ _
{ \]^_`abcdefg {
} hijklmnopqrs }
~ tuvwxyz{|}~  ~

M ",/^_[`jpqy| M
```

Do a screen grab or window dump of that text, using for instance **xwd**, **xgrabsc**, or **screendump**. Convert the result into a pbm file. If necessary, use **pnmcut** to remove everything except the text. Finally, run it through **pnmcrop** to make sure the edges are right up against the text. **pbmtext** can figure out the sizes and spacings from that.

-space *pixels*

Add *pixels* pixels of space between characters. This is in addition to whatever space surrounding characters is built into the font, which is usually enough to produce a reasonable string of text.

pixels may be negative to crowd text together, but the author has not put much thought or testing into how this works in every possible case, so it might cause disastrous results.

-B -lspace *pixels*

Add *pixels* pixels of space between lines. This is in addition to whatever space above and below characters is built into the font, which is usually enough to produce a reasonable line spacing.

pixels must be a whole number.

pixels may be negative to crowd lines together, but the author has not put much thought or testing into how this works in every possible case, so it might cause disastrous results.

USAGE

Often, you want to place text over another image. One way to do this is with **ppmlabel**. **ppmlabel** does not give you the font options that **pbmtext** does, though.

Another way is to use **pbmtext** to create an image containing the text, then use **pnmcomp** to overlay the text image onto your base image. To make only the text (and not the entire rectangle containing it) cover the base image, you will need to give **pnmcomp** a mask, via its **-alpha** option. You can just use the text image itself as the mask, as long as you also specify the **-invert** option to **pnmcomp**.

If you want to overlay colored text instead of black, just use **ppmchange** to change all black pixels to the color of your choice before overlaying the text image. But still use the original black and white image for the alpha mask.

If you want the text at an angle, use **pnmrotate** on the text image (and alpha mask) before overlaying.

SEE ALSO

pnmcut(1), **pnmcrop**(1), **pnmcomp**(1), **ppmchange**(1), **pnmrotate**(1), **pbmtextps**(1), **ppmlabel**(1), **pbm**(5)

AUTHOR