- -

This file, lib.doc, describes the Ghostscript library, a collection of C procedures that implement the primitive graphic operations of the Ghostscript language.

For an overview of Ghostscript and a list of the documentation files, see README.

******** The Ghostscript library ********

Ghostscript is actually two programs: a language interpreter, and a graphics library. The library provides, in the form of C procedures, all the graphics functions of the language, i.e., approximately those facilities listed in section 6.2 of the PostScript manual starting with the graphics state operators. In addition, the library provides some lower-level graphics facilities that offer higher performance in exchange for less generality.

To be specific, the highest level of the library implements all the operators in the "graphics state", "coordinate system and matrix", "path construction", "painting", "character and font", and "font cache" sections of the PostScript manual, with the following deliberate exceptions:

```
      settransfer, currenttransfer
      definefont, findfont
      FontDirectory, StandardEncoding
```

The following "device" operators are implemented:

```
      showpage (synchronizes the display)
      nulldevice
```

There are slight differences in the operators that return multiple values, since C's provisions for this are awkward. Also, the control structure for the operators involving (an) auxiliary procedure(s) (setscreen, pathforall, image, imagemask) is partly inverted: the client calls a procedure to set up an enumerator object, and then calls another procedure for each iteration. The ...show operators, charpath, and stringwidth also use an inverted control structure.

Files named gs*.c implement the higher level of the graphics library.

To use the facilities of gs?.c, a client program should include gs?.h. As might be expected, all procedures, variables, and structures available at this level begin with gs_. Structures that appear in these interfaces, but whose definitions may be hidden from clients, also have names beginning with gs_, i.e., the prefix reflects at what level the abstraction is made available, not the implementation.

Files named gx*.c implement the lower level of the graphics library. To use the facilities of gx?.c, a client program should include gx?.h. The interfaces at the gx level are less stable, and expose more of the implementation detail, than those at the gs level: in particular, the gx interfaces generally use device coordinates in an internal fixed-point representation, as opposed to the gs interfaces that use floating point user coordinates. Named entities at this level begin with gx_.

Files named gz*.c and gz*.h are internal to the Ghostscript implementation, and not designed to be called by clients.