

**NAME**

metafile - graphics command interface, similar to plot(5)

**DESCRIPTION**

The *metafile* graphics format was designed with the primary goal of serving as a temporary file for routines which output to dot-matrix and other line-at-a-time devices. As a result, all of the "primitives" are completely self-contained to facilitate sorting.

A primitive is a command which can itself be plotted. Into this category fall line segments, rectangle and triangle fills, matrix and vector strings. Every primitive has a zeroth argument which contains bundled attribute information, and an extent. The extent gives the x and y minimum and maximum values which enclose the primitive. The extent is used in sorting, and typically also in describing the primitive. For example, a line segment will be described completely by its enclosing rectangle and attributes including specification of which diagonal the segment falls on. Other primitives will have additional arguments, such as vector string, which must specify the string to be output within its extent.

"Global" commands separate the primitives and allow functions which affect all commands. These are commands such as end of page, pause, open and close segment, set, unset and reset, and a special global, end of file. The end of file command is included to facilitate finding the end of file on systems which do not keep track exactly. Global commands sometimes have arguments. The open command, for instance, specifies the name of the segment. Global commands never have extents.

The metafile commands are as follows:

**F** end of file: no arguments.

When end of file is reached, all processing stops.

**E** end of page: no arguments.

This causes the device to advance to the next screen or page. If the output device is a terminal, it will beep and wait for the user to hit return before clearing the screen.

**P** pause: arguments specify the message to be printed.

This causes output to be flushed and the controlling terminal to be opened. The user is then prompted with the specified string followed by the message "- (hit return to continue)". If no string is specified, the bell is sounded without a message. After the user hits return, output continues. This command is useful when the user is required for some part of the output, such as changing paper or pens.

**D** draw global: no arguments.

This global forces flushing of output and updating of device.

**I** include file: arg0 TRUE if standard file.

The include global causes the contents of the named file to be substituted in the include command's location. If arg0 is 1 (TRUE), a standard location is searched if the file is not found in the working directory. If arg0 is 0 (FALSE), the file must be in the working directory. Include files can be nested to the number of allowed open files.

**S** set: arg0 specifies what to set (from meta.h):

SALL: place context mark on current settings.

SPAT0: set pattern 0 to the specified value.

SPAT1: set pattern 1 to the specified value.

SPAT2: set pattern 2 to the specified value.

SPAT3: set pattern 3 to the specified value.

The set command is used to globally affect certain attributes. The zeroth argument specifies the variable to set, and the arguments following specify the value. Pattern values can have two forms. The first form begins with the letter 'P', immediately followed by an integer between 0 and 11. This selects one from the following patterns: solid, thick \\\, thin \\\, mixed \\\, thick ///, thin ///, mixed ///, crisscross, web. The default pattern settings are: 0=P0, 1=P1, 2=P2, 3=P3. The second form gives the explicit values for a pattern. The set all command makes a context mark with the current settings. All settings which follow can be undone with the unset all command.

- U** unset: arg0 specifies what to unset (from meta.h):  
 SALL: return to previous context.  
 SPAT0: set pattern 0 to the previous value.  
 SPAT1: set pattern 1 to the previous value.  
 SPAT2: set pattern 2 to the previous value.  
 SPAT3: set pattern 3 to the previous value.  
 The unset command returns a variable to its previous value. The unset all command returns the settings to the values they had in the previous context. If no context has been marked by set all, variables are returned to their default values.
- R** reset: arg0 specifies what to reset (from meta.h):  
 SALL: reset all variables.  
 SPAT0: set pattern 0 to the default value.  
 SPAT1: set pattern 1 to the default value.  
 SPAT2: set pattern 2 to the default value.  
 SPAT3: set pattern 3 to the default value.  
 The reset command returns a variable to its default setting. The reset all command returns all variables to their initial state.
- O** open segment: arguments specify segment name.  
 The commands following up to a C (close segment) are not to be output, but are to be stored in the named segment. Segment names can contain any ascii character (except newline) in any sequence of reasonable length. Segment definitions are local to the enclosing segment. Side effects should be avoided in segments by balancing calls to set and unset. A segment cannot reference itself.
- C** close segment: no arguments.  
 The current segment is closed, which completes its usable definition.
- l** line segment: fields of arg0 are:  
 100: orientation: positive slope, negative slope.  
 060: type: solid, dashed, dotted, dotted-dashed.  
 014: width: 0, 12, 24, 48, 96 units.  
 003: color: black, red, green, blue.
- r** rectangle fill: fields of arg0 are:  
 100: toggle: OR fill, XOR fill.  
 014: pattern: choice of 4 (see set).  
 003: color: black, red, green, blue.  
 Fills the given extent with the specified pattern. Toggle (XOR) fill allows the reversal of previous fills to an area.
- t** triangle fill: fields of arg0 are:  
 100: toggle: OR fill, XOR fill.  
 060: orientation: right (& down), up, left, down.  
 014: pattern: choice of 4 (see set).  
 003: color: black, red, green, blue.  
 Fills the given half-rectangle with the specified pattern. A triangle is oriented to the right if the the area between the positive-sloped diagonal and the lower right corner of the extent is filled. Rotating this triangle ccw successively yields up, left and down triangles. Toggle (XOR) fill allows the reversal of previous fills to an area.
- p** polygon fill: fields of arg0 are:  
 100: border: no border, line border.  
 060: orientation: right (& down), up, left, down.  
 014: pattern: choice of 4 (see set).  
 003: color: black, red, green, blue.  
 The argument string gives a blank separated list of the polygon vertices in the form: "x0 y0 x1 y1 x2 y2 ... ". The coordinates must be integers ranging between 0 and 16383. The bounding box and orientation will be used to fit the original polygon into a scaled and rotated position. The last vertex will be

connected to the first, and the polygon will be filled in with the specified pattern. If a border is requested, one will be drawn of solid black zero width lines. All polygon fills will toggle, therefore other polygon and toggled triangle and rectangle fills will affect the final appearance of the image. For example, a polygon drawn inside another polygon of the same pattern will make a hole.

**m** matrix string: fields of arg0 are:

100: strike: single, double.

060: density: 10 cpi, 12 cpi, 17 cpi, 20 cpi.

014: size: normal, double width, double height, double both.

003: color: black, red, green, blue.

The upper left corner of the extent is used to place the beginning of the string specified after the command. More sophisticated drivers will use the extent for clipping, but the size of the characters will not be altered.

**v** vector string: fields of arg0 are:

060: orientation: right, up, left, down.

014: thickness: 0, 12, 24, 48, 96 units.

003: color: black, red, green, blue.

The string specified following the command will be made to fit within the given extent.

**s** print segment: fields of arg0 are:

060: orientation: right, up, left, down.

014: thickness: 0, 12, 24, 48, 96 units.

003: color: black, red, green, blue.

The segment whose name is specified in the arguments will be oriented according to arg0 and made to fit in the given extent. The thickness and color of the lines in the segment will be changed also according to arg0. In the case of area fill, it is the pattern rather than the width which will change. The segment must have been previously defined using the open segment global. Note that matrix strings will not transfer well since they cannot be oriented or scaled.

The metafile has two basic formats. The first format is meant to be user readable, and has the form:

```
c arg0 xmin ymin xmax ymax 'args
```

Where c is the single letter command, arg0 is the octal value for arg0, xmin ymin xmax ymax are the extent (ranging from 0 to 16283), and the optional args following the backquote are additional arguments, terminated by a newline. If the command is a global, the extent is not present. If the global has no arg0, 0200 is appropriate. Any global which has a following string must have a value for arg0 (< 0200). Comments are permitted on lines beginning with a pound sign ('#').

The second format is roughly equivalent, but packs the extrema into two bytes each. It takes between one quarter and one third as much space, and much less processing to use this type of file, hence it is the default format for all of the programs. Conversion between formats is accomplished with cv(1).

## FILES

The standard location for metafiles used by the programs is /usr/lib/meta/, but can be changed by setting the environment variable MDIR. This is useful for systems where the owner does not have access to the /usr/lib/ directory. It also allows the user to create his own metafiles for vector characters and other symbols.

## BUGS

The command for line segment ('l') is awkward at best.

## AUTHOR

Greg Ward

## SEE ALSO

cv(1), meta(3), pexpand(1), primout(3), psort(1)