Underlays.

Underlays are coloured rounded boxes painted under groff source. They are great devices to emphasise and organize bigger chunks of text and to differentiate groups at-a-glance by colour. This page and the next two show a few examples.

We have two pairs of macros to provide a number of goods: different sizes and colours, horizontal and vertical dividers, a grid for layout planning, the inclusion of an external PostScript file to do millions of things.

```
.beg_under1  .end_under1   paint underlay,  your font and spacing
.beg_under2  .end_under2   paint underlay,  .nf .ft CR .ps 7 .vs .4c are provided here
```

Both pairs are working in the same way. One pair of the *.beg_.... end_...* macros enclose a chunk of text that may contain the usual formatting requests and two new ones, specific to these pairs (in blue):

```
.beg_...
text without/with formatting commands
.   div_height 0.4
text without/with formatting commands e.g. \m[red] or \f(CB

blank lines
text without/with formatting commands
.   div_Height 0.5
text without/with formatting commands
.end_...
```

The most basic jobs of the pairs are as follows:

```
The .beg_... macro starts a diversion: formats and collects text until the .end... macro is found

the .end_... macro finds out how much space is required, paints this area with the given colour,
                  then paints the formatted text over it
```

These pairs can do quite a bit more than that. The outcome can be influenced by the special formatting macros inserted into the text (see the blue lines in the white box), by blank lines in the text, by the macro parameters, by a few number registers, and by setup parameters in the *gr.setup1* file and in common variables in the *PS ToolBox*.

In addition, tabs can be converted to spaces or left alone, and a convenient line continuation mechanism is provided.

a) The *.div_height* and *.div_Height* inserts.

These macros can be called up-to 6 times each within one underlay. When called they produce space according to their parameters, then they find out the heights of the middle of these spaces. They record these heights in the *\n[u_h1] ... \n[u_h6]* and *\n[U_h1]...\n[U_h6]* number registers, in 'u' units. These heights are measured from the bottom of the underlay, upwards.

These heights can be used in PostScript inserts to good effect (see *Macro parameters* later).
In addition to recording heights, wherever *.div_height* is called, the *.end_...* macro draws a horizontal line at that height. It will use the same colour (hue) as of the underlay, only a bit darker.

b) Blank lines.

Please consult the introduction.

c) Tabs and line continuation.

Please consult the introduction.

d) Macro parameters.

Both pairs have 6 parameters, all of them are optional:

.beg_under1		
1) indent	e.g. 0.5c	default: 0.3c
.end_under1		
1) hue	in degrees: 0-359, many extras	**default: 180 (cyan)**
2) x-left of box	from page offset, in millimetres	default: 0
3) x_right		default: 150
4) "[x ... x]"	x values for vertical lines from page offset, in millimetres	default: no lines
5) include file	external PS file to include	default: no file
.beg_under2		
1) indent	e.g. 0.5c or 75	default: 97 (this is the max)
.end_under2		
1) ... 5)	the same parameters than those of .end_under1	

All x values are measured from the page offset in the *\n[s_po]* register. Initially this has the value of '.po' saved at the beginning of the groff document.

The only parameter of *.beg_under2* is different from that of its counterpart. In addition to accepting indent in 'c', you may specify the number of characters to accommodate in the box, centered.
The *.beg_...* routines set the line-length so that to place the text in the middle but, if '.nf' is declared, line-length is not observed.

The colours are all pastel colours, the specials are: -1 for gray fill, -2 for gray stroke (border, see below). Horizontal/vertical lines are drawn with a *complementary colour*: the same hue, but a bit darker. Here's a selection of colours. *Please use as few of them in the same document as possible.* More specials later.

-2	-1	0	60	120	180	240	300

PostScript inserts are drawn after the underlay and before the text. They can do anything: paint eps pictures, draw 3d objects, etc. With their use one needs to observe the following:

- They are external PostScript- or ToolBox files.
- Currently no environment variables are accepted in file names.
- They have to be 'naked', i.e they may not contain *showpage* or *P_page_port1* or *P_page_land1* or *P_page_end.*
- They may contain ASCIIHexDecode images physically included in the file but, for the time being, no ASCII85Decode images are allowed.
- Included files are given their viewports in the *u1,v1,u2,v2* variables. The *u1,v1* pair (both zeroes) mean the lower-left corner of the underlay, the *u2,v2* pair the upper-right corner, in millimetres.

  In addition to that, PS inserts have access to groff's number registers and strings. For example, the *u_h1* groff number register contains the height at the time of the first call to the *.div_height* macro. In PS terms the height above the bottom of the underlay in millimetres is:

  /height \n[u_h1] G_u2mm d_

  Please note the single backslash.

e) Pre-defined number registers and the ‘1000’ colour modifier.

```
.nr debug    0       If set to 1, a grid of 2 mm resolution will be drawn between the under-
                     lay and your text, to help in placing inserts, vertical bars, etc
.nr ut_st    0       If set to 1 (and color > -2), it strokes the outline of the underlay
                     In printing this is not necessary, but on-screen this may look better
.nr ut_wh    0       If set to 1 (and color = -2), it whites-out the underlay before stroking
colour ‘x + 1000’    It paints with the darker complementary colour of x (see the cyans below)
.nr c_gra   -1       colour for graphics                              gray
.nr c_var   20              for common variables - data - registers   brick
.nr c_pro   50              for processes - macros                    mustard
.nr c_mix  180              for all else                              cyan
.nr u_g1  0.38c      underlay dimensions: top-gap
.nr u_m1  0.22c                           top-margin (within underlay)
.nr u_m2  0.20c                           bot-margin
.nr u_g2  0.38c                           bot-gap
```

An example.

This is part of a Perl subroutine[1] and the example aims to show how to make the embeddedness of the if structure clearer by using *.div_Height* calls at the tops and bottoms of these *if ... { ... }* structures and a very short external PS file:

```
foreach my $ss (split /,/, $st) {                       # sub-strings
  if ((length $ss) == 0) { next ; }
  if (! ($ss =~ /\D/)) { if ($ss == $nu) { goto yeah ; } else { next ; } }
  if ((index $ss, "-") >=0) {
     my ($n1, $n2) = split /-/, $ss ;
     if ($n2 eq "e") { if ($nu >= $n1              ) { goto yeah ; } }
     else            { if ($nu >= $n1 && $nu <= $n2) { goto yeah ; } } next ;
  }
  if ((index $ss, "(") >=0 && (index $ss, ")") > 0) {
     my ($n1, $n2, $n3) = split /[()]/, $ss ;
     if ((modulo $n1, $n2) == (modulo $nu, $n2)) {
        if ($n3 eq "e") { if ($nu >= $n1              ) { goto yeah ; } }
        else            { if ($nu >= $n1 && $nu <= $n3) { goto yeah ; } }
     }
  }
}                                                       # foreach
```

The PS insert is very simple:

```
14 \n[U_h2] G_u2mm 128.5 \n[U_h1] G_u2mm  -2          P_tport1
14 \n[U_h6] G_u2mm 128.5 \n[U_h3] G_u2mm  -2          P_tport1
18 \n[U_h5] G_u2mm 122.0 \n[U_h4] G_u2mm 180 1000 add P_tport1
```

It gets the vertical dimensions from the *U_h1* etc registers, while the horizontal dimensions (‘14’ etc) can be read from the grid produced by setting the *debug* register to 1.

[1] Perl has many backslashes that may interfere with groff. If you need to print a backslash but deny it any meaning for groff, type ‘\e’ instead of ‘\’.