Imagine

Imagine

~~~
Imagine
  A pandoc filter to turn fenced codeblocks into graphics or ascii art by
  wrapping some external command line utilities, such as:

    actdiag, asy, asymptote, blockdiag, boxes, circo, ctioga2, ditaa, dot,
    fdp, figlet, flydraw, gle, gnuplot, graph, graphviz, gri, imagine,
    mermaid, mscgen, neato, nwdiag, octave, packetdiag, pic, pic2plot,
    plantuml, plot, ploticus, protocol, pyxplot, rackdiag, seqdiag, sfdp,
    shebang, twopi


Installation

    % sudo -H pip install pandoc-imagine

    or simply save `pandoc-imagine.py` anywhere along $PATH


Dependencies

    % sudo -H pip install pandocfilters six

    and one (or more) of the packages that provide above utilities.


Pandoc usage

    % pandoc --filter pandoc-imagine.py document.md -o document.pdf


Markdown usage

    ```cmd
~~~

~~~~
  code
  ```

which will run `cmd` (if known) to proces the `code` into a png image and
replaces the fenced code block with an Image in a paragraph of its own or any
ascii art in its own CodeBlock.

Alternate, longer form:

  ```{.cmd im_opt=".." im_out=".." im_prg=<other-cmd>}
  code
  ```

- im_opt="..." will be passed onto the command line.
  Some classes already provide some defaults (as required by the command).

- im_out="...", csv-list of keywords each specifying a certain output
  - img     image in a paragraph
  - fcb     codeblock containing the original codeblock
  - stdout, codeblock containing stdout output (if any)
  - stderr, codeblock containing stderr output (if any)

- im_prg=<other-cmd>, overrides class-to-command map.
  Only useful if `cmd` itself is not an appropiate class in your document.

- im_fmt="..." for replacing the default output format (The list of available
  formats depends of the class)

If the command fails, the original fenced code block is retained unchanged.
Any info on stderr is relayed by Imagine, which might be useful for
troubleshooting.

If the command succeeds but produces no image, a line reporting the missing
image is included in the output document.

Notes:
- filenames are based on a hash of the codeblock + its attributes
- uses subdir `pd-images` to store any input/output files
- there's no clean up of files stored there
- if an output filename exists, it is not regenerated but simply linked to.
- `packetdiag` & `sfdp`s underlying libraries seem to have some problems.

Some commands follow a slightly different pattern:
- 'img' directive is ignored by commands that only produce ascii
- ctioga2 defaults to pdf instead of png
- flydraw produces a gif, not png
~~~~

  - gle also creates a .gle subdir inside the images-dir
  - gri produces a ps, which is `convert`ed to png
  - imagine reads its code as help-topics, returns codeblocks with help-info
  - plot reads its codeblock as the relative path to the file to process
  - pyxplot will have `set terminal` & `set output` prepended to its `code`
  - shebang runs its codeblock as a script with <fname>.{im_fmt} as its argument.
    - use {.shebang im_out="stdout"} for text instead of an png

Security

Imagine just hands the fenced codeblocks to plotting tools to process or simply runs them as system scripts, as-is.

Shebang's are inherently unsafe and most of the plotting tools implement their own 'little' languages, which can create beautiful images, but can also cause harm.

There is no way to check for 'side effects' in advance, so make sure to check the fenced codeblocks before running them through the filter.

Imagine class

The imagine class puts documentation of topics at your fingertips, like so:

~~~
    ```imagine
    class
    ```
~~~

Use `imagine` as class to get the module's docstring (ie this text) and/or one or more of the commands you're interested in, each on a separate line.

Thanks for helping out:

- amietn
- chdemko
- heyrict
- priiduonu

Noop's

Only codeblocks with one of Imagine's classes will be recognized and processed.

Anonymous CodeBlock

Anonymous codeblocks are not processed.

```
This code block is anonymous and not processed by Imagine.
```

A Python CodeBlock

Neither is a python codeblock processed.

```python
if processed_by(Imagine):
    raise Expection('Not ignored by Imagine!')
else:
    print "Great, if you're reading this, it passed through Imagine unharmed"
```

Asymptote

```
Codeblock class: asy

    sudo-apt-get install asymptote

    See http://asymptote.sourceforge.net/

    runs:
    > asy -o <fname>.{im_fmt} {im_opt} <fname>.asy

    class->cmd
      asy -> asy
      asymptote -> asy

Metadata options
    imagine.im_out: img,fcb
    imagine.im_log: 4
```

Notes:

- eps formatted images don't go well together with pandoc.

a plot

~~~~
```{.asy im_out="fcb,img" caption="Created by Asymptote"}
settings.outformat="png";
settings.prc=false;
settings.render=0;
import three;
size(6cm,0);
draw(O--2X ^^ O--2Y ^^ O--2Z);
triple circleCenter = (Y+Z)/sqrt(2) + X;
path3 mycircle = circle(c=circleCenter, r=1, normal=Y+Z);
draw(plane(O=sqrt(2)*Z, 2X, 2*unit(Y-Z)), gray + 0.1cyan);
draw(mycircle, blue);
draw(shift(circleCenter) * (O -- Y+Z), green, arrow=Arrow3());
```
~~~~

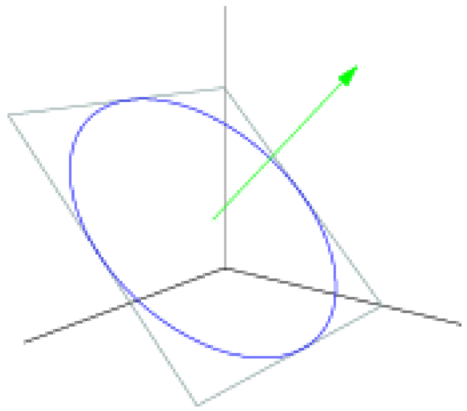


Figure 1: Created by Asymptote

a sphere

```
settings.outformat="png";
settings.prc=false;
settings.render=0;
import graph3;
size(8cm,0);
path3 myarc = rotate(18,Z) * Arc(c=O, normal=X, v1=-Z, v2=Z, n=10);
surface backHemisphere = surface(myarc, angle1=0, angle2=180, c=O, axis=Z, n=10);
surface frontHemisphere = surface(myarc, angle1=180, angle2=360, c=O, axis=Z, n=10);
draw(backHemisphere, surfacepen=material(white+opacity(0.8), ambientpen=white), meshpen=gray
draw(O--X, blue+linewidth(1pt));
```

blockdiag site:

blockdiag command

~~~
```{.blockdiag im_prg="blockdiag" im_out="fcb,img" width="100%" caption="Created by Blockdia
blockdiag {
// standard node shapes
box [shape = "box"];
roundedbox [shape = "roundedbox"];
diamond [shape = "diamond"];
ellipse [shape = "ellipse"];
note [shape = "note"];
cloud [shape = "cloud"];
mail [shape = "mail"];
beginpoint [shape = "beginpoint"];
endpoint [shape = "endpoint"];
minidiamond [shape = "minidiamond"];
actor [shape = "actor"];
dots [shape = "dots"];
box -> roundedbox -> diamond -> ellipse;
cloud -> note -> mail -> actor;
minidiamond -> beginpoint -> endpoint -> dots;
// node shapes for flowcharts
condition [shape = "flowchart.condition"];
database [shape = "flowchart.database"];
input [shape = "flowchart.input"];
loopin [shape = "flowchart.loopin"];
loopout [shape = "flowchart.loopout"];
terminator [shape = "flowchart.terminator"];
condition -> database -> terminator -> input;
loopin -> loopout;
}
```
~~~

seqdiag

~~~
```{.seqdiag im_out="fcb,img" width="80%" height="50%" caption="Created by seqdiag"}
{
browser -> webserver [label = "GET /index.html"];
browser <-- webserver;
browser -> webserver [label = "POST /blog/comment"];
webserver -> database [label = "INSERT comment"];
webserver <- database;
browser <- webserver;
~~~

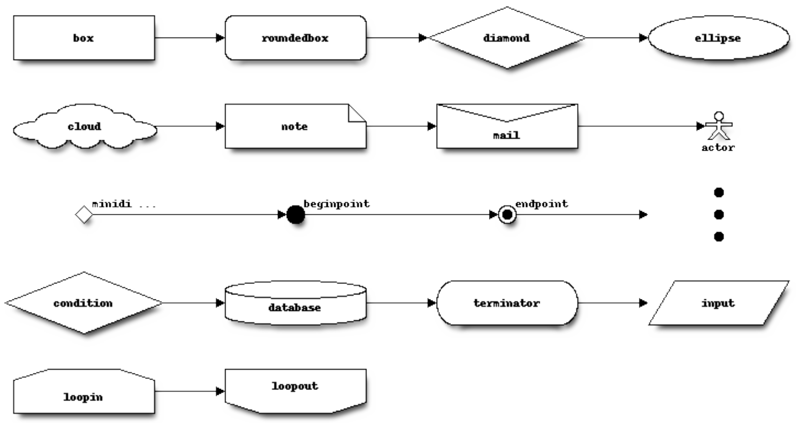


Figure 2: Created by Blockdiag

```
}

```

nwdiag

~~~
```{.nwdiag im_out="fcb,img" caption="Created by nwdiag"}
{
  network dmz {
      address = "210.x.x.x/24"

      web01 [address = "210.x.x.1"];
      web02 [address = "210.x.x.2"];
  }
  network internal {
      address = "172.x.x.x/24";

      web01 [address = "172.x.x.1"];
      web02 [address = "172.x.x.2"];
      db01;
      db02;
  }
}
```
~~~

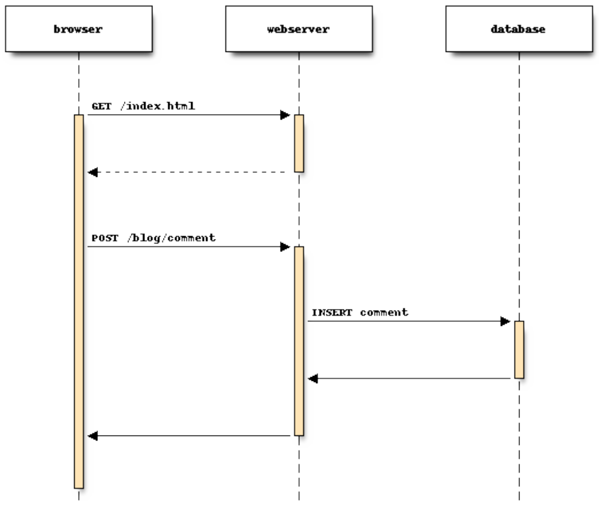


Figure 3: Created by seqdiag

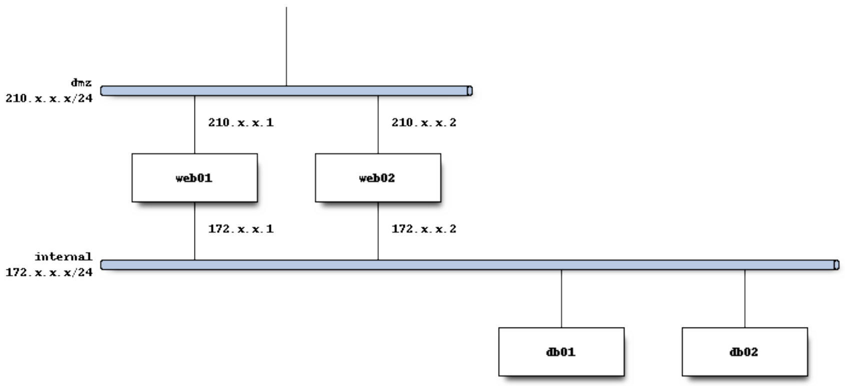


Figure 4: Created by nwdiag

actdiag

~~~
```{.actdiag im_out="fcb,img" height="60%" caption="Created by actdiag"}
{
   A -> B -> C -> D;

  lane foo {
    A; B;
  }
  lane bar {
    C; D;
  }
}
```
~~~

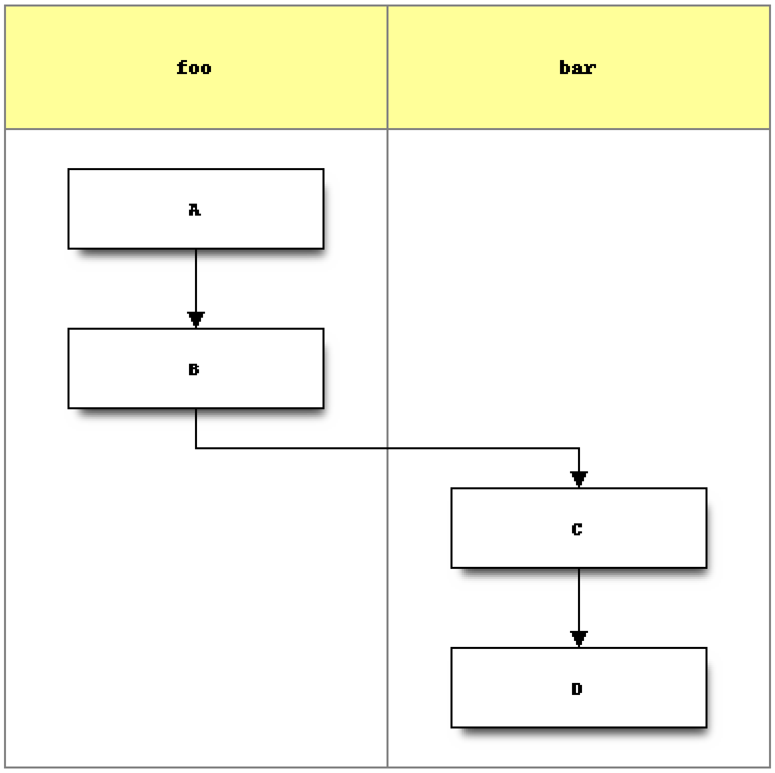


Figure 5: Created by actdiag

rackdiag

~~~
```{.rackdiag im_out="fcb,img" height="80%" caption="Created by rackdiag"}
{
  // define 1st rack
  rack {
    16U;

    // define rack items
    1: UPS [2U];
    3: DB Server
    4: Web Server
    5: Web Server
    6: Web Server
    7: Load Balancer
    8: L3 Switch
  }

  // define 2nd rack
  rack {
    12U;

    // define rack items
    1: UPS [2U];
    3: DB Server
    4: Web Server
    5: Web Server
    6: Web Server
    7: Load Balancer
    8: L3 Switch
  }
}
```
~~~

packetdiag

Unfortunately, packetdiag doesn't work properly due to a problem with some library:

```
Imagine:BlockDiag:  packetdiag -> ERROR: images do not match
```

```
{
  colwidth = 32
  node_height = 72

  0-15: Source Port
```

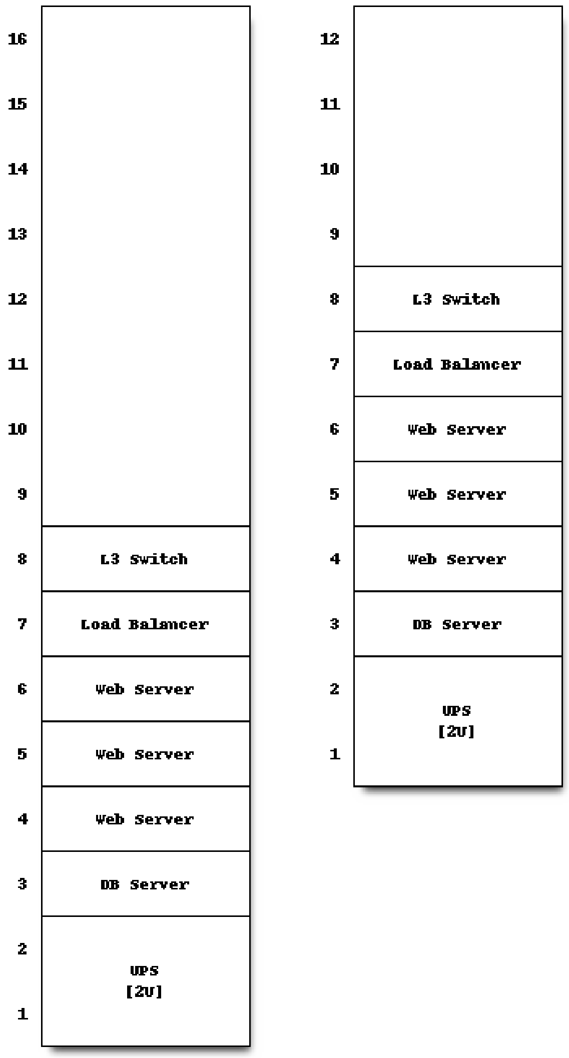


Figure 6: Created by rackdiag

```
  16-31: Destination Port
  32-63: Sequence Number
  64-95: Acknowledgment Number
  96-99: Data Offset
  100-105: Reserved
  106: URG [rotate = 270]
  107: ACK [rotate = 270]
  108: PSH [rotate = 270]
  109: RST [rotate = 270]
  110: SYN [rotate = 270]
  111: FIN [rotate = 270]
  112-127: Window
  128-143: Checksum
  144-159: Urgent Pointer
  160-191: (Options and Padding)
  192-223: data [colheight = 3]
}
```

boxes

boxes Boxes is a command line program that draws a box around its input text. It can remove and repair those boxes, too.

design 'peek'

~~~
```{.boxes im_opt="-d peek -a c -s 40x3" im_out="fcb,stdout" caption="boxes"}
boxes
```
~~~

```
/*       _\|/_
         (o o)
 +----oOO-{_}-OOo----------------------+
 |                boxes                |
 +------------------------------------*/
```

design 'ian_jones'

~~~
```{.boxes im_opt="-d ian_jones -a c -s 40x6" im_out="fcb,stdout" caption="boxes"}
There are about 52 available styles, and you can create your own if
none of them suit your needs.
```
~~~

```
                               \\\///
```

```
                               / _  _ \
                             (| (.)(.) |)
.--------------------------.OOOo--()--oOOO.-------------------------.
|There are about 52 available styles, and you can create your own if|
|                   none of them suit your needs.                   |
'-------------------------.oooO-------------------------------------'
                           (   )   Oooo.
                            \ (    (   )
                             \_)    ) /
                                   (_/
```

ctioga2

Parabolas, filling & intersection

~~~
```{.ctioga2 im_out="fcb,img" caption="Created by ctioga2" width="60%"}
title "Intersection of two parabolas"
math
plot x*x /fill=top /fill-transparency 0.8 /legend '$x^2$'
plot 50-x*x /fill=bottom /fill-transparency 0.8 /legend '$50 - x^2$'
```
~~~

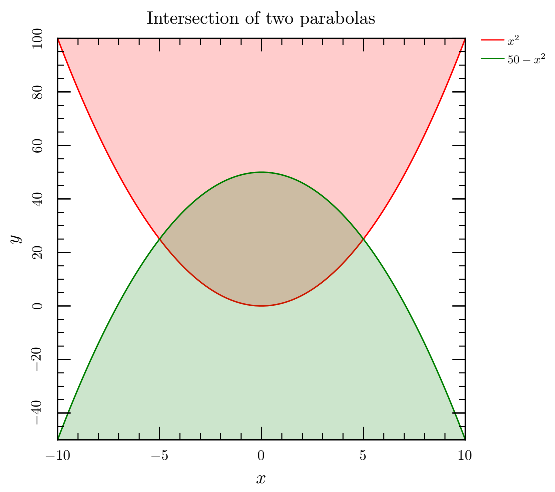


Figure 7: Created by ctioga2

a grid system

~~~
```{.ctioga2 im_out="fcb,img" caption="Created by ctioga2" width="60%"}
define-axis-style '.grid-non-left axis.left' /decoration=ticks /axis-label-text=' '
define-axis-style '.grid-non-bottom axis.bottom' /decoration=ticks /axis-label-text=' '
define-background-style '.grid-odd-column background' /background-color Blue!15
define-axis-style '.grid-2-0 axis' /decoration=None

setup-grid 3x2 /top=1mm /right=2mm /dy=2mm /dx=2mm
math


inset grid:next
  plot sin(x)
next-inset grid:next
  plot cos(x)
next-inset grid:next
  plot -cos(x)
next-inset grid:next
  plot x**2
next-inset grid:next
  plot 10*x
next-inset grid:next
  plot 0.1*x**3
end
```
~~~

plotting data

The data file's name `../dta/cr2-ex01.dat` is relative to the saved fenced code block in pd-images. Hence the `../dta` part.

~~~
```{.ctioga2 im_out="fcb,img" caption="Created by ctioga2" width="60%"}
draw-line -15,0 15,0 /style=Dashes /color=Gray
plot ../dta/ct2-ex01.dat
plot ../dta/ct2-ex01.dat@1:3
title '\centering This is a very long title about sine waves'  \
      /text-width=5cm /shift=1.3
xlabel 'My $x$ label'
ylabel 'My $y$ label'
plot ../dta/ct2-ex01.dat@'$1:$2*0.5'
plot ../dta/ct2-ex01.dat@'$1:0.5*($2-$3)'
```
~~~

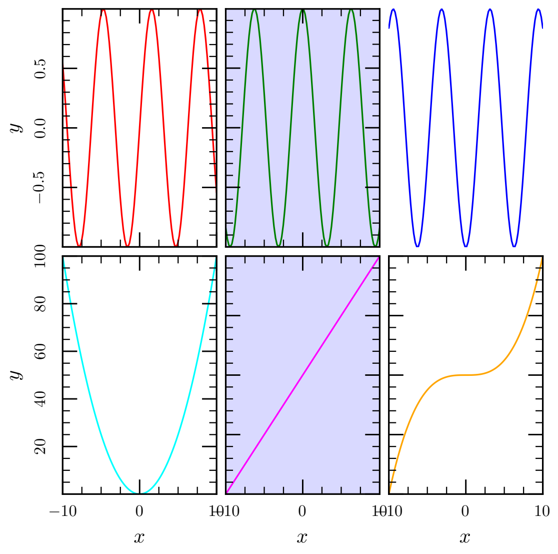


Figure 8: Created by ctioga2

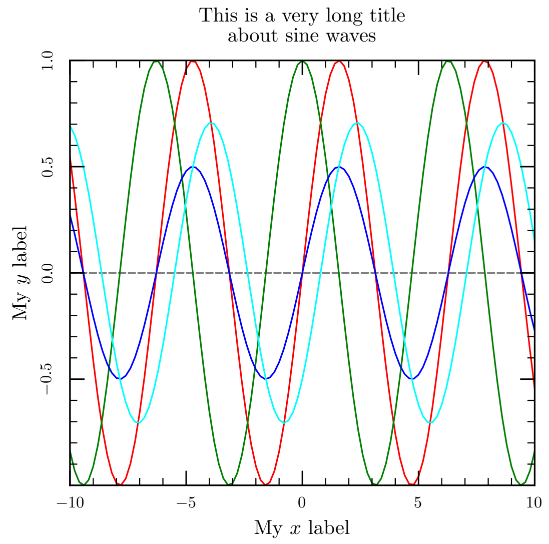


Figure 9: Created by ctioga2

ditaa site:

Rounded corners (im_opt="-r")

~~~
```{.ditaa im_opt="-r" im_out="fcb,img" width="70%" caption="Created by Ditaa"}
+--------+   +-------+    +-------+
|        +---+ ditaa +--> |       |
|  Text  |   +-------+    |diagram|
|Document|   |!magic!|    |       |
|     {d}|   |       |    |       |
+---+----+   +-------+    +-------+
    :                         ^
    |       Lots of work      |
    +-------------------------+
```
~~~

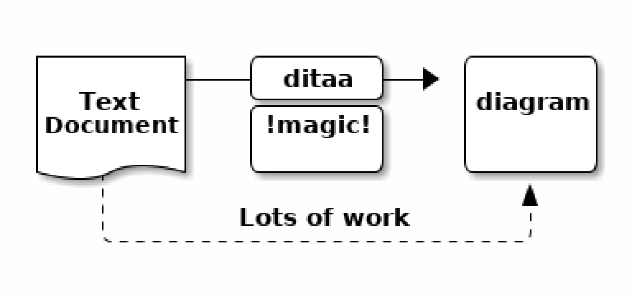


Figure 10: Created by Ditaa

Ditaa normal

~~~
```{.ditaa im_out="fcb,img" caption="Created by Ditaa"}
   +---------+   +-------+   +-------+    +--------+      +--------+
   | Document|---+ split +---|       |----|        |----->|        |
   | o  this |   +-------+   |Diagram|    | Storage|      | In/Out |
   | o  that |   |   me  |   |       |    |        |      |        |
   |  cRED{d}|-+ |   cGRE|   |   cBLK| /--| cBLU{s}|  /-*-|cPNK{io}|
   +----+----+ : +-------+   +-------+ |  +--------+  |   +--------+
        :      |     ^                 |              |
        |      v     |     /--------\  |  /--------\  |
        +------------+     | Rounded|<-/  | Rounded|-*+  *--------*
                           | Corners|     | Dashed | |   | Point  |
                           |    c33F|     |        | +-*-*  Mark  *
                           \-+------/     \-=------/     |    c1FF|
~~~

```
                                                          *---------*
```

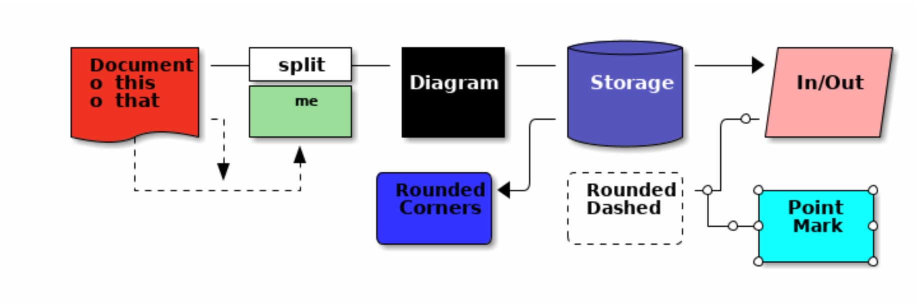


Figure 11: Created by Ditaa

ditaa reminder

~~~
```{.ditaa im_out="fcb,img" height="20%" caption="Created by Ditaa"}
/-----------------\
| Things to do    |
| cYEL            |
| o Cut the grass |
| o Buy jam       |
| o Fix car       |
| o Make website  |
\-----------------/
```
~~~

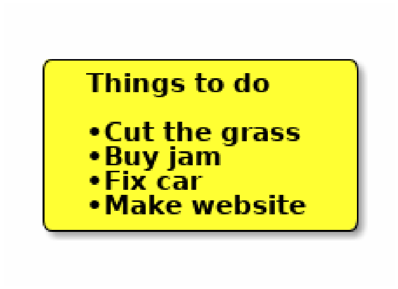


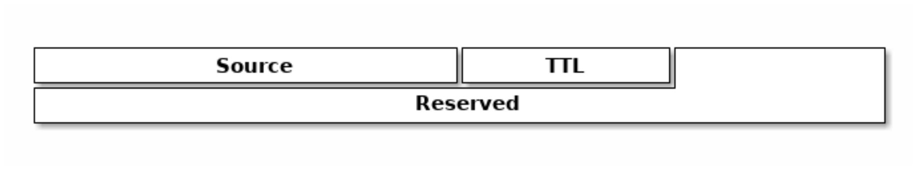
Figure 12: Created by Ditaa

Ditaa on protocol result

~~~
```{.ditaa im_out="fcb,img"}
~~~

~~~~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Source            |      TTL      |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               +
|                            Reserved                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

```
~~~~



Figlet

figlet

~~~~
```{#FIGLET .figlet im_opt="-f slant" im_out="fcb,stdout" caption="Figlet"}
figlet
```
~~~~

```
    _____       __     __
   / __(_)___ _/ /__  / /_
  / /_/ / __ `/ / _ \/ __/
 / __/ / /_/ / /  __/ /_
/_/ /_/\__, /_/\___/\__/
      /____/
```

hello world.

~~~~
```{.figlet im_out="fcb,stdout"}
hello, world!
```
~~~~

```
 _          _ _                            _     _ _
| |__   ___| | | ___    __      _____  _ __| | __| | |
| '_ \ / _ \ | |/ _ \   \ \ /\ / / _ \| '__| |/ _` | |
| | | |  __/ | | (_) |   \ V  V / (_) | |  | | (_| |_|
|_| |_|\___|_|_|\___( )   \_/\_/ \___/|_|  |_|\__,_(_)
                    |/
```

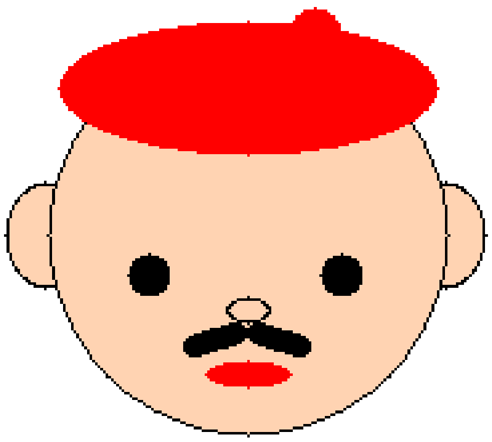
Flydraw

Notes:

- seems to only want to produce GIF, despite the manual's mention of PNG.
- only reads from stdin

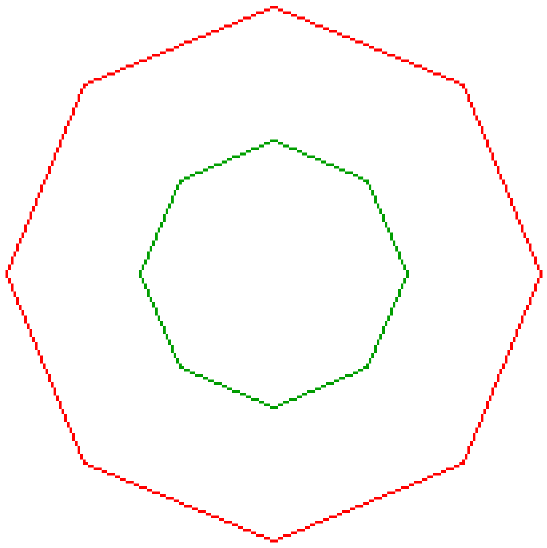
frenchman

~~~
```{.flydraw im_out="fcb,img"}
comment : from KhanAcademy
new 200,200
comment ears
fellipse 24, 100, 30, 40,255, 211, 178
fellipse 174, 100, 30, 40,255, 211, 178
ellipse 24, 100, 30, 40,black
ellipse 174, 100, 30, 40,black
comment face
fellipse 100, 100, 150, 150,255, 211, 178
ellipse 100, 100, 150, 150,black
comment nose
ellipse 100, 128, 17, 10,black
comment beret
fellipse 125, 25, 20, 20,red
fellipse 100, 45, 142, 50, red
comment mouth
fellipse 100, 152, 32, 10,red
linewidth 16
point 63, 115,black
point 135, 115 ,black
linewidth 8
line 80, 142, 96, 137, black
line 120, 142, 104, 137,black
```
~~~



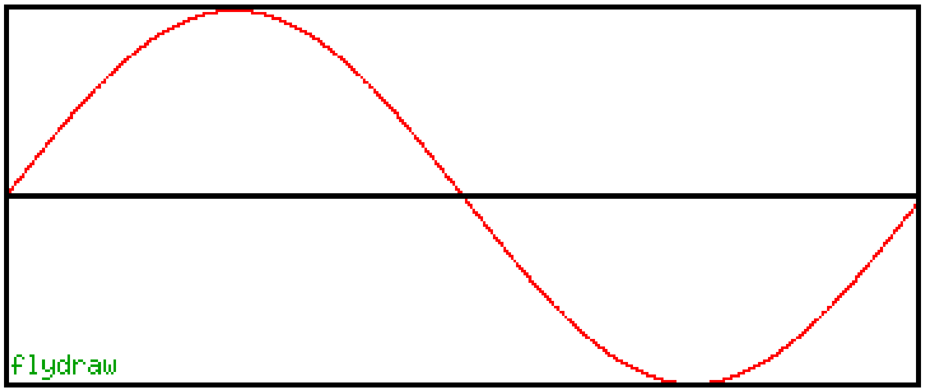
hexagons

~~~
```{.flydraw im_out="fcb,img"}
comment x=horizontal, x=0 is left
comment y=vertical,   y=0 is top
new 300,300
x0=150
y0=150
r=100
t1=0
t2=t1+2*pi
linewidth=1
plotstep 8
trange t1,t2
plot red,r*cos(t)+x0,r*sin(t)+y0
plot green,r*0.5*cos(t)+x0,r*0.5*sin(t)+y0
```
~~~



plotting a function

~~~
```{.flydraw im_out="fcb,img"}
w=360
h=150
new w,h
linewidth=1
plotstep=9000
r=-2+h/2
y0=h/2
plot red,y0-r*sin(2*pi*x/w)
linewidth=2
rect 1,1, w-1,h-1, black
line 0,y0,w,y0, black
text green,3,h-16,normal,"flydraw"
```
~~~



GLE

Baudrate

Notes:

- ../test.dat is relative to the input file in pd-images …

~~~
```{.gle im_out="fcb,img" caption="Created by GLE"}
size 18 19

amove 2 1
box 15 16 fill gray60
rmove -1 1
box 15 16 fill white
rmove 2 4
box 11 8 fill gray5

set font texcmr hei 0.6

begin graph
    fullsize
    size 11 8
    title "BAUD Rate = 9600 bit/sec"
    xtitle "Seconds"
    ytitle "Bits"
    data "../dta/test.dat"
    d1 line marker wsquare
    xaxis min -1 max 6
    yaxis min 0 max 11
end graph
~~~

```
```

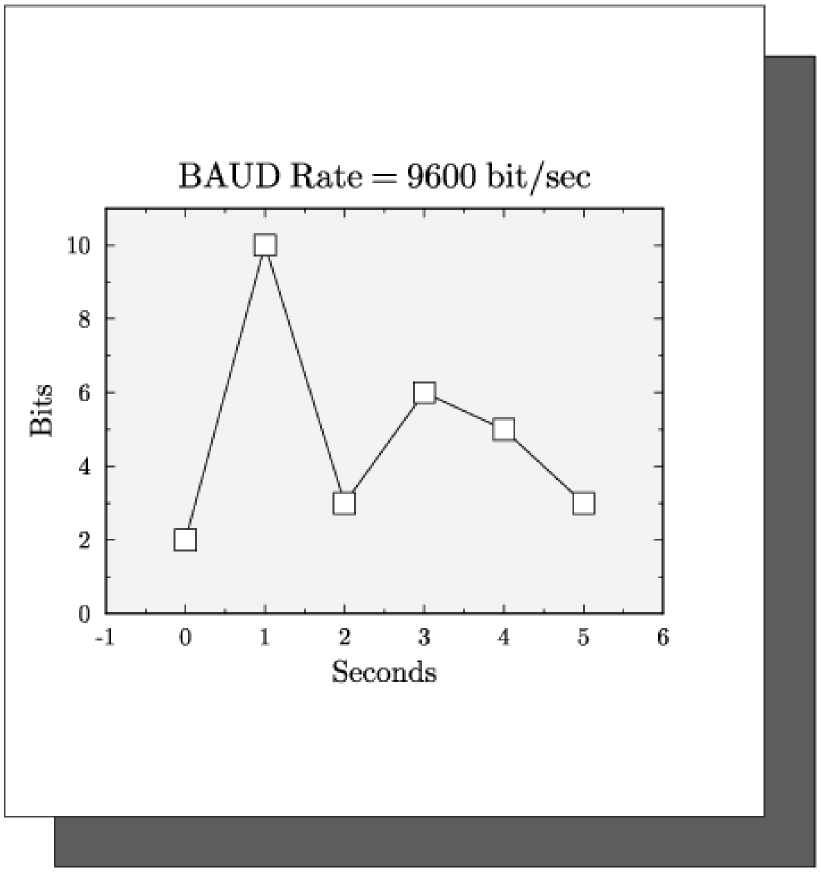


Figure 13: Created by GLE

simple 2D

~~~
```{.gle im_out="fcb,img" caption="Created by GLE"}
size 12 10

set font texcmr
begin graph
   math
   title "f(x) = sin(x)"
~~~

```
   xaxis min -2*pi max 2*pi ftick -2*pi dticks pi/2 format "pi"
   yaxis dticks 0.25 format "frac"
   let d1 = sin(x)
   d1 line color red
end graph
```

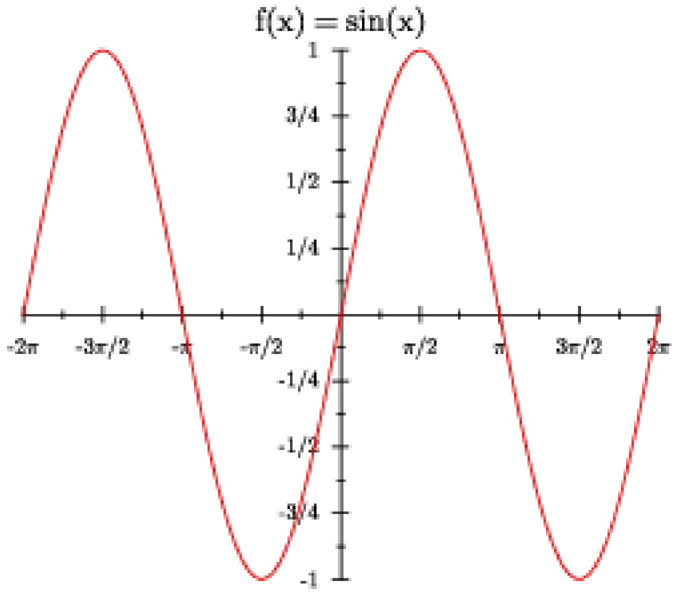


Figure 14: Created by GLE

Semi-transparant fills

Needs the `-cairo` option.

~~~
```{.gle im_opt="-cairo" im_out="fcb,img" caption="Created by GLE"}
size 10 7

set texlabels 1

begin graph
   scale auto
~~~

```
   title  "Semi-Transparent Fills"
   xtitle "Time"
   ytitle "Output"
   xaxis min 0 max 9
   yaxis min 0 max 6 dticks 1
   let d1 = sin(x)*1.5+1.5 from 0 to 10
   let d2 = 1/x from 0.01 to 10
   let d3 = 10*(1/sqrt(2*pi))*exp(-2*(sqr(x-4)/sqr(2))) from 0 to 10
   key background gray5
   begin layer 300
      fill x1,d1 color rgba255(255,0,0,80)
      d1 line color red key "$1.5\sin(x)+1.5$"
   end layer
   begin layer 301
      fill x1,d2 color rgba255(0,128,0,80)
      d2 line color green key "$1/x$"
   end layer
   begin layer 302
      fill x1,d3 color rgba255(0,0,255,80)
      d3 line color blue key "$\frac{10}{\sqrt{2\pi}}\exp\left(\frac{-2(x-4)^2}{2^2}\right)
   end layer
end graph
```

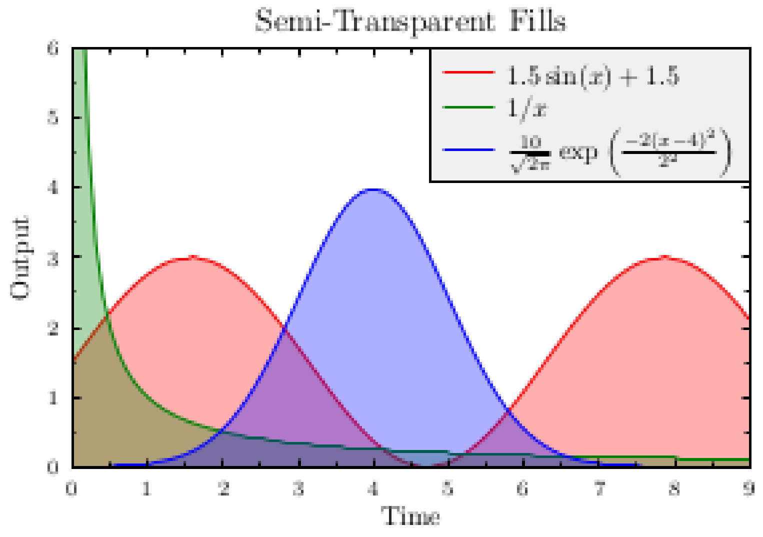


Figure 15: Created by GLE

saddle up

The following GLE script creates saddle.dta, which we want to be put in the dta directory so the file name is given relative to the pd-images directory.

~~~
```{.gle im_out="fcb,img" caption="Created by GLE"}
size 10 9

set font texcmr hei 0.5 just tc

begin letz
   data "../dta/saddle.z"
   z = 3/2*(cos(3/5*(y-1))+5/4)/(1+(((x-4)/3)^2))
   x from 0 to 20 step 0.5
   y from 0 to 20 step 0.5
end letz

amove pagewidth()/2 pageheight()-0.1
write "Saddle Plot (3D)"

begin object saddle
   begin surface
      size 10 9
      data "../dta/saddle.z"
      xtitle "X-axis" hei 0.35 dist 0.7
      ytitle "Y-axis" hei 0.35 dist 0.7
      ztitle "Z-axis" hei 0.35 dist 0.9
      top color blue
      zaxis ticklen 0.1 min 0 hei 0.25
      xaxis hei 0.25 dticks 4 nolast nofirst
      yaxis hei 0.25 dticks 4
   end surface
end object

amove pagewidth()/2 0.2
draw "saddle.bc"
```
~~~

An electronic circuit

~~~
```{.gle im_out="fcb,img" caption="Created by GLE"}
! An H-Bridge

size 13 11
include "electronics.gle"
~~~

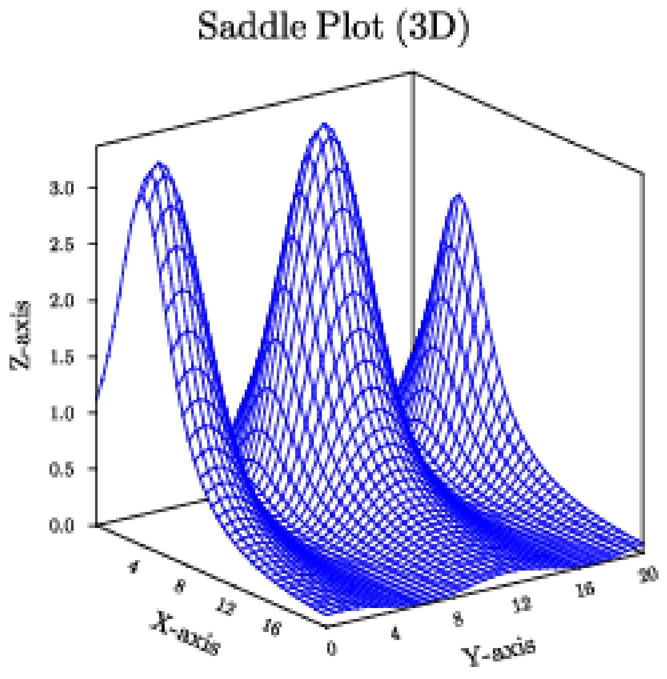


Figure 16: Created by GLE

~~~
set lwidth 0.05 cap round font psh

! Draw a grid if the line below is uncommented
drawgrid 1

! Top left of diagram
amove 2.0 9.0

! Battery leg
gsave
rline 0 -0.5
cell_v "E_1"
rline 0 -3.5
rline 5 0
rresistor_h R_4
grestore

rresistor_h R_1

gsave
rresistor_v R_2
cell_v "E_2"
grestore

rline 5 0
rresistor_v R_3
rline 0 -4
```
~~~

Gnuplot

Note:

- Imagine catches gnuplot's output on stdout and saves it to an output file. So don't `set output <name>` or Imagine will get confused and die miserably.

Line

~~~
```{.gnuplot im_out="fcb,img" height="50%" caption="Created by GnuPlot"}
set terminal pngcairo  transparent enhanced font "arial,10" fontscale 1.0 size 500, 350
set key inside left top vertical Right noreverse enhanced autotitles box linetype -1 linewic
set samples 200, 200
~~~

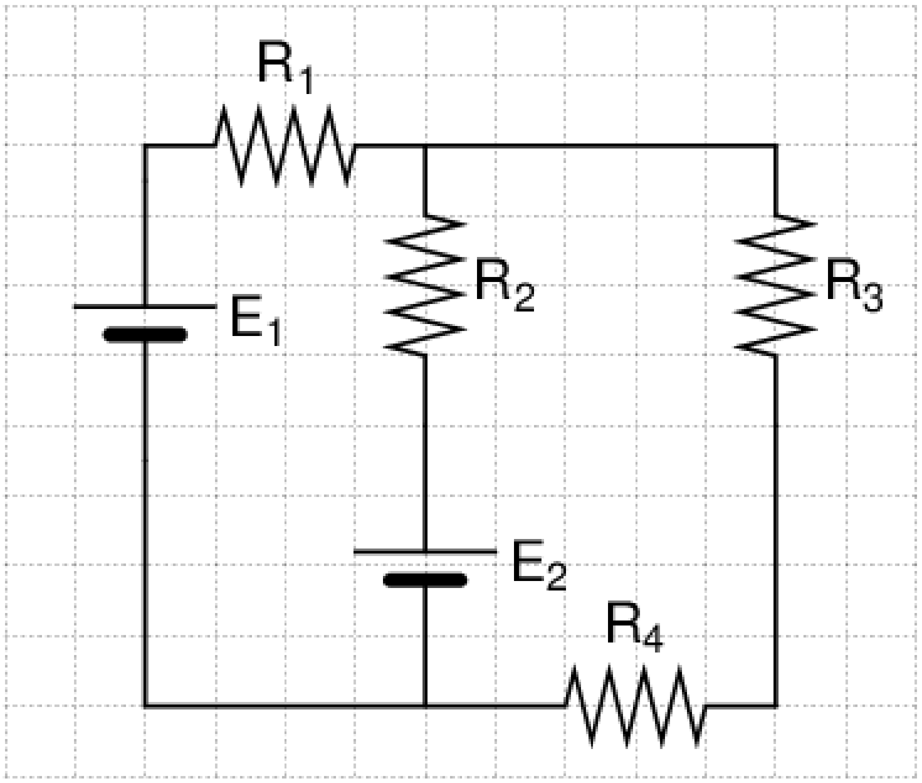


Figure 17: Created by GLE

```
plot [-30:20] besj0(x)*0.12e1 with impulses, (x**besj0(x))-2.5 with points
```

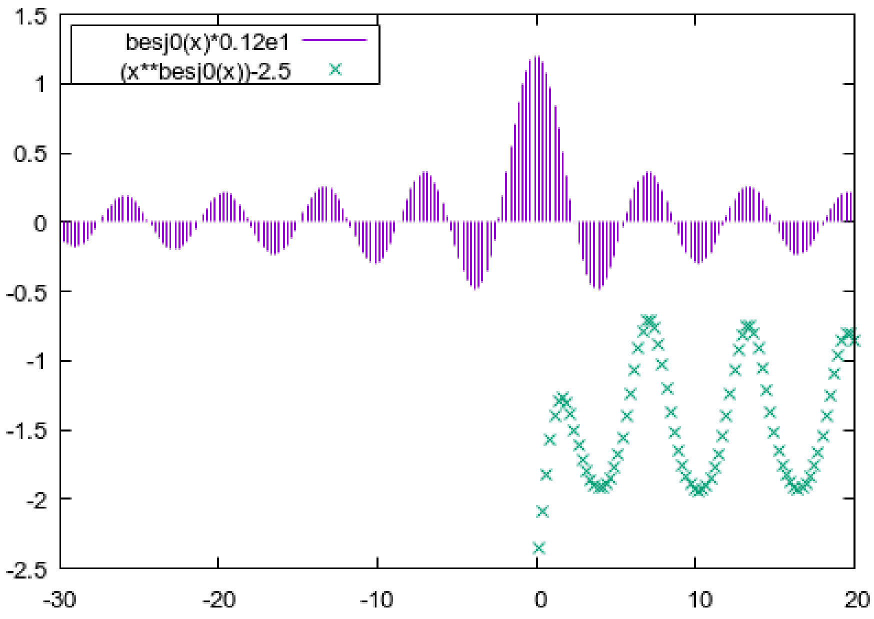


Figure 18: Created by GnuPlot

real sine

```{.gnuplot im_out="fcb,img" height="50%" caption="Created by GnuPlot"}
set terminal pngcairo  transparent enhanced font "arial,10" fontscale 1.0 size 500, 350
set key inside left top vertical Right noreverse enhanced autotitles box linetype -1 linewid
set samples 400, 400
plot [-10:10] real(sin(x)**besj0(x))
```

Surface

```{.gnuplot im_out="fcb,img" caption="Another GnuPlot example"}
set terminal pngcairo  transparent enhanced font "arial,10" fontscale 1.0 size 500, 350
set border 4095 front linetype -1 linewidth 1.000
set view 130, 10, 1, 1
set samples 50, 50
set isosamples 50, 50
```

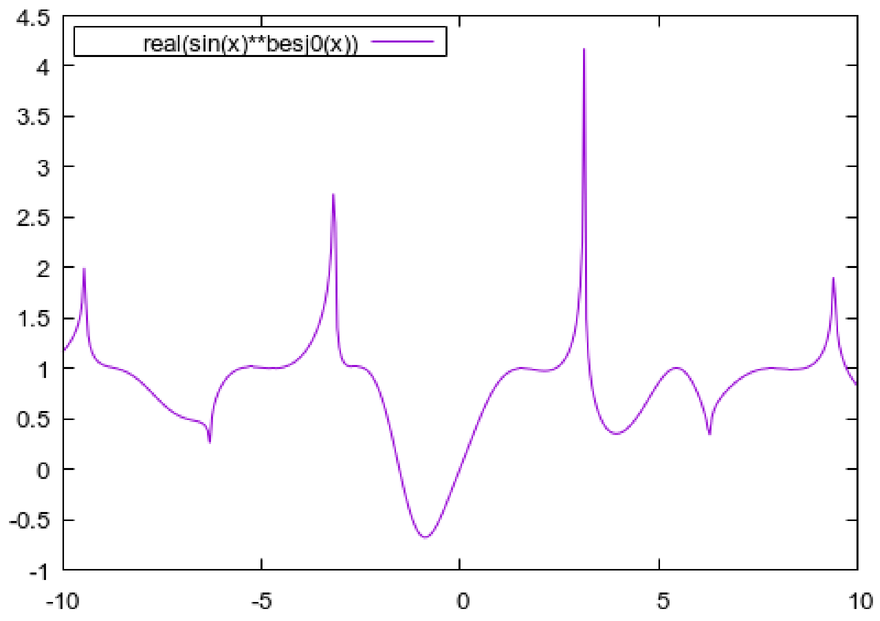


Figure 19: Created by GnuPlot

~~~~
unset surface
set title "set pm3d scansbackward: correctly looking surface"
set pm3d implicit at s
set pm3d scansbackward
splot sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
```
~~~~

Interlocking Tori

~~~~
```{.gnuplot im_out="fcb,img" caption="Gnuplot's interlocking Tori example"}
set terminal pngcairo  transparent enhanced font "arial,10" fontscale 1.0 size 500, 350
set dummy u,v
set key bmargin center horizontal Right noreverse enhanced autotitles nobox
set parametric
set view 50, 30, 1, 1
set isosamples 50, 20
set hidden3d back offset 1 trianglepattern 3 undefined 1 altdiagonal bentover
set ticslevel 0
set title "Interlocking Tori"
set urange [ -3.14159 : 3.14159 ] noreverse nowriteback
set vrange [ -3.14159 : 3.14159 ] noreverse nowriteback
~~~~

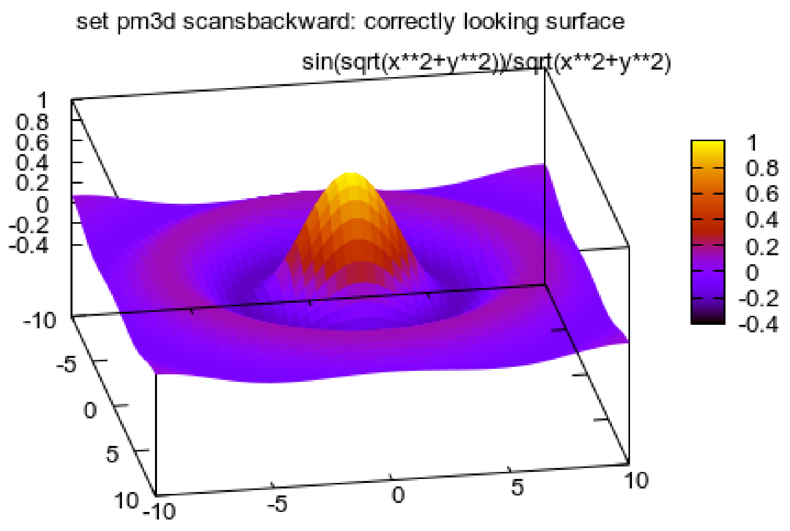


Figure 20: Another GnuPlot example

```
splot cos(u)+.5*cos(u)*cos(v),sin(u)+.5*sin(u)*cos(v),.5*sin(v) with lines,       1+cos(u)+.
```
```
```

graphviz.org site:

Graphviz defaults to dot

~~~
```{im_prg="dot" im_opt="-Gsize=4,1.5" caption="FSM layout by dot" im_out="fcb,img"}

digraph finite_state_machine {
    rankdir=LR;
    size="6,3"
    node [shape = doublecircle]; LR_0 LR_3 LR_4 LR_8;
    node [shape = circle];
    LR_0 -> LR_2 [ label = "SS(B)" ];
    LR_0 -> LR_1 [ label = "SS(S)" ];
    LR_1 -> LR_3 [ label = "S($end)" ];
    LR_2 -> LR_6 [ label = "SS(b)" ];
    LR_2 -> LR_5 [ label = "SS(a)" ];
    LR_2 -> LR_4 [ label = "S(A)" ];
~~~

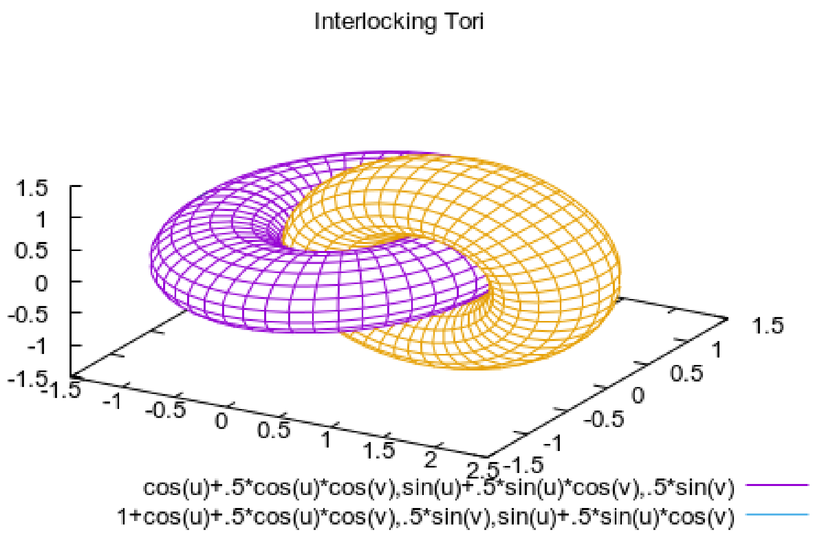


Figure 21: Gnuplot's interlocking Tori example

```
    LR_5 -> LR_7 [ label = "S(b)" ];
    LR_5 -> LR_5 [ label = "S(a)" ];
    LR_6 -> LR_6 [ label = "S(b)" ];
    LR_6 -> LR_5 [ label = "S(a)" ];
    LR_7 -> LR_8 [ label = "S(b)" ];
    LR_7 -> LR_5 [ label = "S(a)" ];
    LR_8 -> LR_6 [ label = "S(b)" ];
    LR_8 -> LR_5 [ label = "S(a)" ];
}

```
```

fdp

```
```{.graphviz im_prg="fdp" im_opt="-Gsize=2,3" caption="Created by fdp" im_out="fcb,img"}

digraph {
 blockcode -> fdp;
 fdp -> image;
 }
```

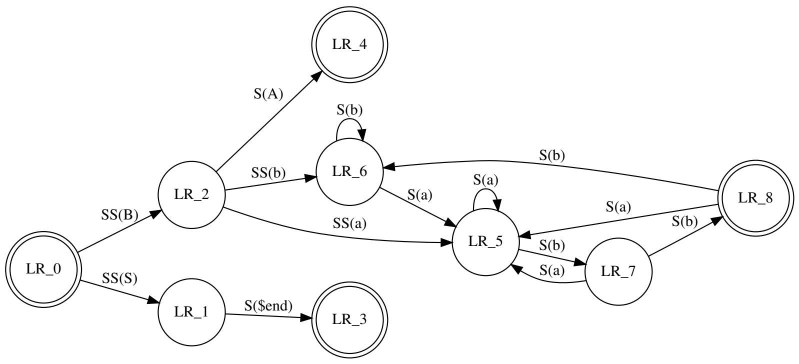


Figure 22: FSM layout by dot

```
```

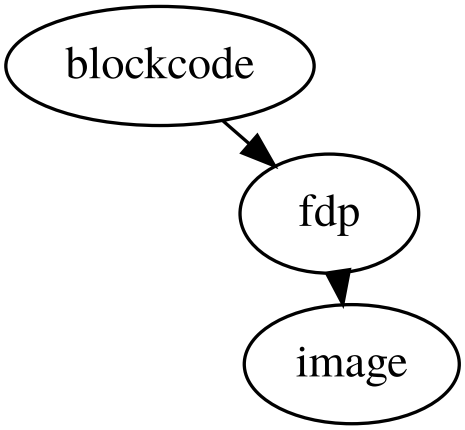


Figure 23: Created by fdp

`sfdp` *(fails)*

~~~
```{.graphviz im_prg="sfdp" caption="Not created by sfdp"}
graph G {
size="2,2"
run -- intr;
intr -- runbl;
runbl -- run;
run -- kernel;
kernel -- zombie;
kernel -- sleep;
~~~

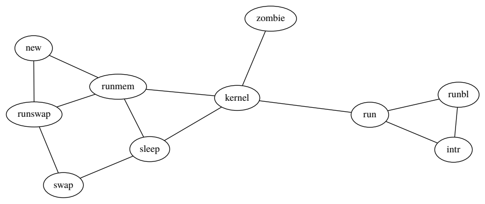


Figure 24: Not created by sfdp

```
kernel -- runmem;
sleep -- swap;
swap -- runswap;
runswap -- new;
runswap -- runmem;
new -- runmem;
sleep -- runmem;
}
```

`neato`

States in a kernel OS plotted by `neato`:

~~~
```{.graphviz im_prg="neato" caption="Created by neato" im_out="fcb,img"}
graph G {
size="3,2"
run -- intr;
intr -- runbl;
runbl -- run;
run -- kernel;
kernel -- zombie;
kernel -- sleep;
kernel -- runmem;
sleep -- swap;
swap -- runswap;
runswap -- new;
runswap -- runmem;
new -- runmem;
sleep -- runmem;
}
```
~~~

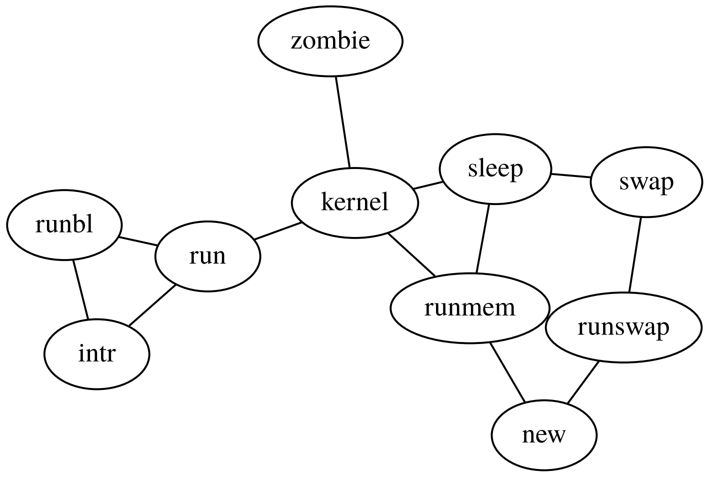


Figure 25: Created by neato

twopi

The same, but by `twopi`:

~~~
```{.graphviz im_prg="twopi" caption="Created by twopi" im_out="fcb,img"}
graph G {
size="3,2"
run -- intr;
intr -- runbl;
runbl -- run;
run -- kernel;
kernel -- zombie;
kernel -- sleep;
kernel -- runmem;
sleep -- swap;
swap -- runswap;
runswap -- new;
runswap -- runmem;
new -- runmem;
sleep -- runmem;
}
```
~~~

circo

Again, the same but by `circo`:

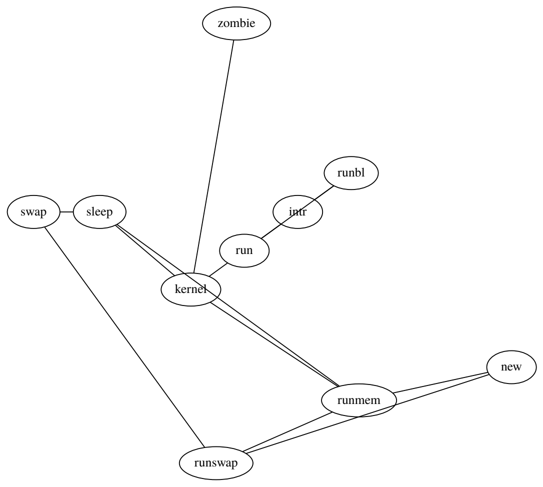


Figure 26: Created by twopi

~~~
```{.graphviz im_prg="circo" caption="created by circo" im_out="fcb,img"}

graph G {
size="3,2"
run -- intr;
intr -- runbl;
runbl -- run;
run -- kernel;
kernel -- zombie;
kernel -- sleep;
kernel -- runmem;
sleep -- swap;
swap -- runswap;
runswap -- new;
runswap -- runmem;
new -- runmem;
sleep -- runmem;
}
```
~~~

GRI

Single plot

With the following in `gri-01.dat`

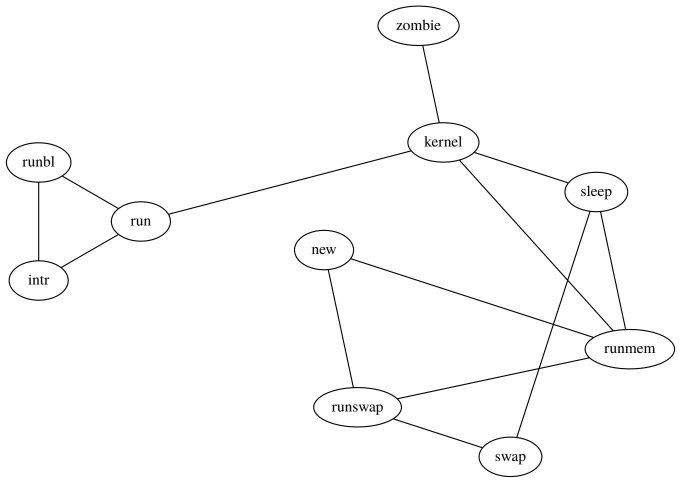


Figure 27: created by circo

```
1  8 11  9
2 22 21 20
3 11 10  9
4 20 15 10
```

plot the first two columns like so:

~~~
```{.gri im_out="fcb,img" caption="Created by Gri"}
open dta/gri-01.dat
read columns x y
draw curve
draw title "http://gri.sf.net"

```
~~~

Multiple curves

~~~
```{.gri im_out="fcb,img" caption="Created by Gri"}
`draw curves' \xname \y1name ...'`
Draw multiple y columns versus an x column.  Assumes
that the datafile is open, and that x is in the first
column, with the y values in one or more following
columns.


The number of columns is figured out from the options,
as is the name of the x-axis, and the labels to be
~~~

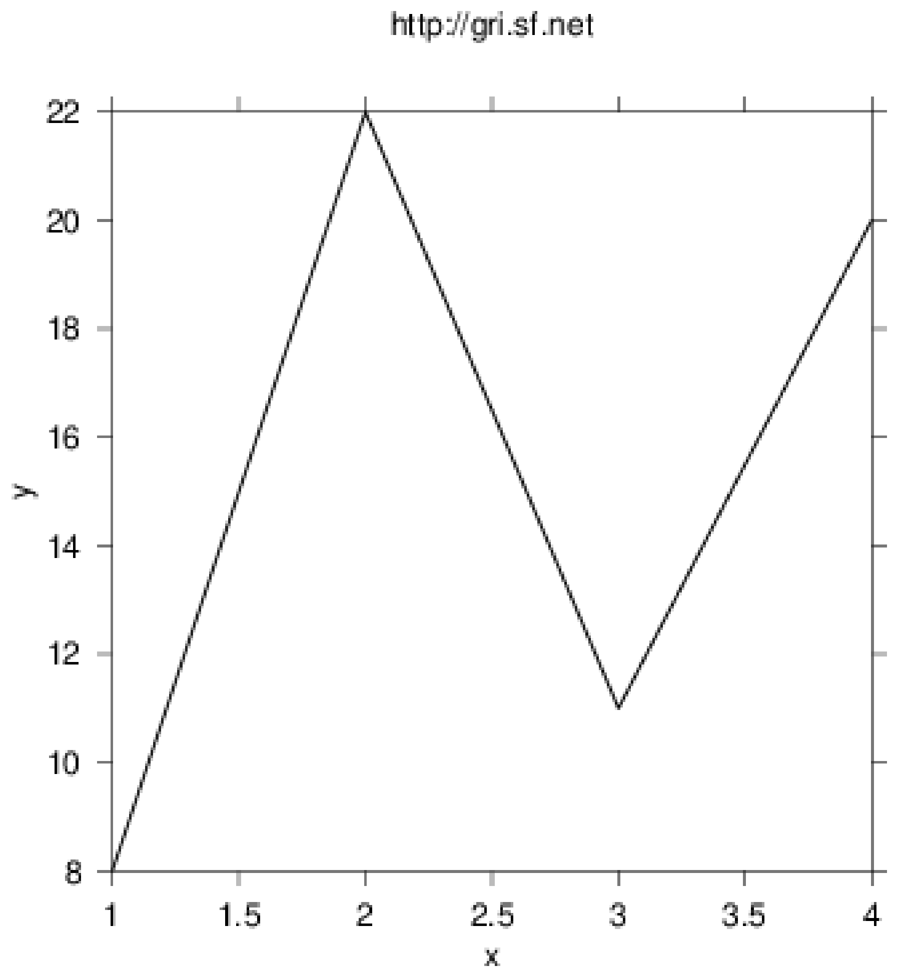


Figure 28: Created by Gri

~~~~
used on each of the y curves.
{
  # NB. the 3 below lets us skip the words 'draw'
  # and 'curves', and the name of the x-column.
  .num_of_y_columns. = {rpn wordc 3 -}
  if {rpn .num_of_y_columns. 1 >}
    show "ERROR: 'draw curves' needs at least 1 y column!"
    quit
  end if


  set x name {rpn 2 wordv}
  set y name ""


  # Loop through the columns.
  .col. = 0
  while {rpn .num_of_y_columns. .col. <}
    # The x-values will be in column 1, with y-values
    # in columns 2, 3, ..., of the file.
    .ycol. = {rpn .col. 2 +}
    rewind
    read columns x=1 y=.ycol.
    # At this point, you may want to change line thickness,
    # thickness, color, dash-type, etc.  For illustration,
    # let's set dash type to the column number.
    set dash .col.
    draw curve
    draw label for last curve {rpn .col. 3 + wordv}
    .col. += 1
  end while
}


open dta/gri-01.dat
draw curves time y1 y2 y3 y4

```
~~~~

Mermaid

sequence graph

~~~~
```{.mermaid im_out="fcb,img" width="70%" caption="Created by mermaid"}
~~~~

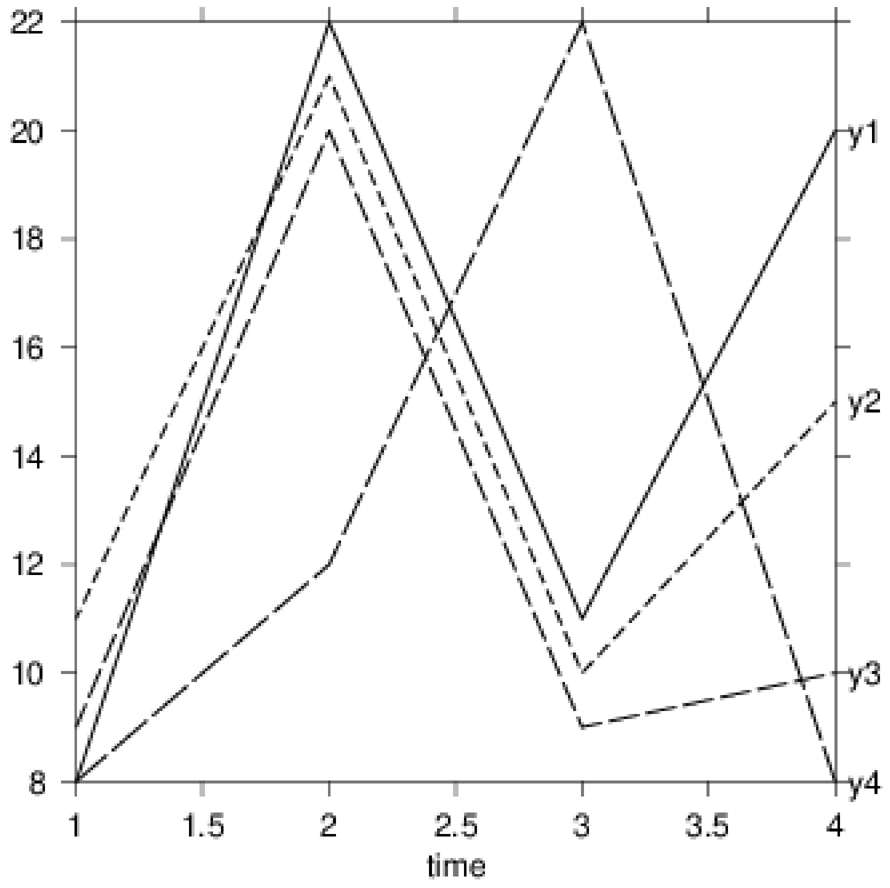


Figure 29: Created by Gri

~~~
sequenceDiagram
    participant Alice
    participant Bob
    Alice->>John: Hello John, how are you?
    loop Healthcheck
        John->>John: Fight against hypochondria
    end
    Note right of John: Rational thoughts<br/>prevail...
    John-->>Alice: Great!
    John->>Bob: How about you?
    Bob-->>John: Jolly good!
```
~~~

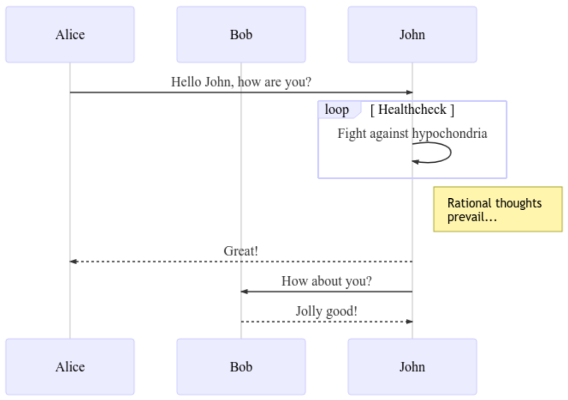


Figure 30: Created by mermaid

gantt diagram

~~~
```{.mermaid im_out="fcb,img" caption="Created by mermaid"}
gantt
    title A Gantt Diagram

    section Section
    A task           :a1, 2014-01-01, 30d
    Another task     :after a1  , 20d
    section Another
    Task in sec      :2014-01-12  , 12d
    another task      : 24d
```
~~~

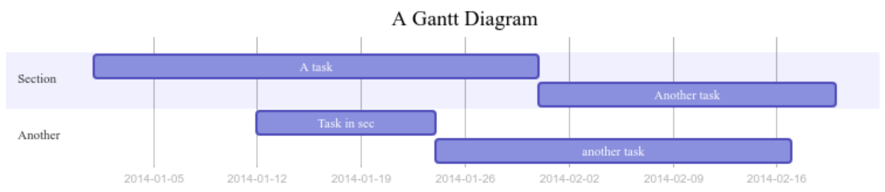


Figure 31: Created by mermaid

Mscgen site:

example w/ boxes

~~~
```{.mscgen im_out="fcb,img" caption="Created by mscgen"}
msc {

   # The entities
   A, B, C, D;

   # Small gap before the boxes
   |||;

   # Next four on same line due to ','
   A box A [label="box"],
   B rbox B [label="rbox"],
   C abox C [label="abox"],
   D note D [label="note"];

   # Example of the boxes with filled backgrounds
   A abox B [label="abox", textbgcolour="#ff7f7f"];
   B rbox C [label="rbox", textbgcolour="#7fff7f"];
   C note D [label="note", textbgcolour="#7f7fff"];
}
```
~~~

client-server interaction

~~~
```{.mscgen im_out="fcb,img" caption="Created by mscgen"}
msc {
 hscale="1.3", arcgradient = "8";

 a [label="Client"],b [label="Server"];
~~~

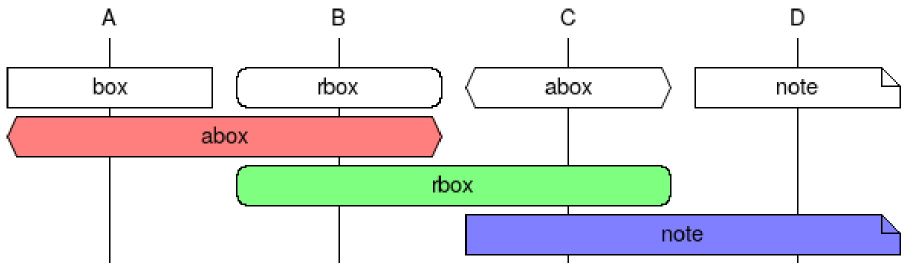


Figure 32: Created by mscgen

```
 a=>b [label="data1"];
 a-xb [label="data2"];
 a=>b [label="data3"];
 a<=b [label="ack1, nack2"];
 a=>b [label="data2", arcskip="1"];
 |||;
 a<=b [label="ack3"];
 |||;
}
```

```

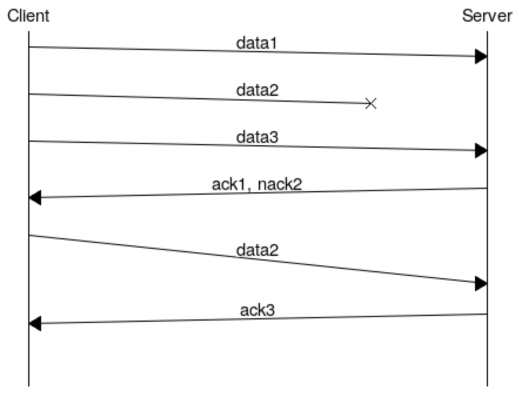


Figure 33: Created by mscgen

## *Octave*

Hints for using `Octave` as batch processor:

- `;` makes statements silent
```

- `figure(1, "visibility", "off")` prevents pop-up window
- `print(1, argv(){1});` prints to intended output filename
- octave will infer image type from output filename extension
- `imagine` calls `octave --no-gui -q <im_opt> <inpfile> <outfile>`, where
    - `<im_opt>` come from im_opt="..." in the fenced code blocks attributes
    - `<inpfile>` is `pd-images/hashed-name.octave` containing the code text
    - `<outfile>` is `pd-images/hashed-name.png` by default

Sinus plot

~~~
```{.octave im_out="fcb,img" caption="Created by Octave"}
outname = argv(){1}
figure(1, 'visible', 'off');

x = 0:0.01:2*pi;
a = sin(x);
b = cos(2*x);
c = sin(4*x);
d = 2*sin(3*x);
plot(x,a,x,b,x,c,x,d, "linewidth", 2);
set(gca, "xlim", [0,2*pi], "fontsize", 15);
title("sinusoids");

print(1, outname, '-dpng');
```
~~~

?? missing pd-images/44dff105434e960d2cfb04ee01671897ed4f363d.png

Peaks surface

~~~
```{.octave im_out="fcb,img" caption="Created by Octave"}
figure(1, 'visible', 'off');

surf(peaks);
title("peaks");

print(1, argv(){1});
```
~~~

?? missing pd-images/7ec895c9f15eea22040e146ba35f584d3616e875.png

Peaks contour

~~~
```{.octave im_out="fcb,img" caption="Created by Octave"}
figure(1, 'visible', 'off');

contourf(peaks);
title("peaks");

print(1, argv(){1});
```
~~~

?? missing pd-images/136059550aaf185a22dfd9df16ca1ae27da8dac6.png

3-D wave

~~~
```{.octave im_out="fcb,img" caption="Created by Octave"}
outname = argv(){1}
figure(1, 'visible', 'off');

x = 0:0.1:2*pi;
y = 0:0.1:2*pi;
z = sin(x)' * sin(y);
mesh(x, y, z);
xlabel("x-axis");
ylabel("y-axis");
zlabel("z-axis");
title("3-D waves");

print(1, outname, '-dpng');
```
~~~

?? missing pd-images/8e07651a4008fc2e2130a2818315e04315081885.png

Plantuml site:

sequence diagrams

~~~
```{.plantuml im_out="fcb,img" width="60%" caption="Created by plantuml"}
@startuml
autonumber "<b>[000]"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

autonumber 15 "<b>(<u>##</u>)"
~~~

```
Bob -> Alice : Another authentication Request
Bob <- Alice : Another authentication Response

autonumber 40 10 "<font color=red><b>Message 0  "
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response

@enduml
```
```

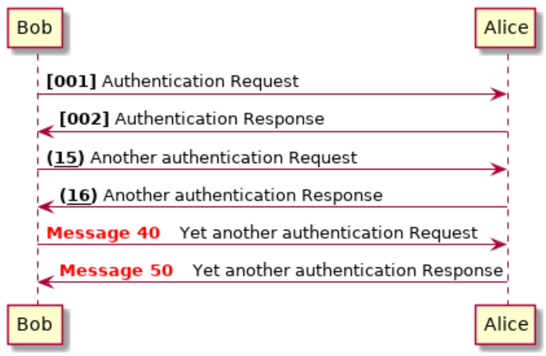

Bob
Alice
[001] Authentication Request
[002] Authentication Response
(15) Another authentication Request
(16) Another authentication Response
Message 40 Yet another authentication Request
Message 50 Yet another authentication Response
Bob
Alice

Figure 34: Created by plantuml

## class diagrams

```
```{.plantuml im_out="fcb,img" width="60%" caption="Created by plantuml"}
@startuml
Class01 <|-- Class02
Class03 *-- Class04
Class05 o-- Class06
Class07 .. Class08
Class09 -- Class10
@enduml
```
```

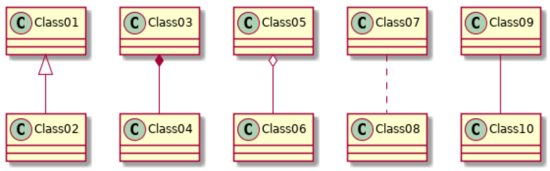


Figure 35: Created by plantuml
```

larger plantuml

~~~
```{.plantuml im_out="fcb,img" caption="Created by plantuml"}
@startuml
scale 580*690
title Servlet Container
(*) --> "ClickServlet.handleRequest()"
--> "new Page"
if "Page.onSecurityCheck" then
->[true] "Page.onInit()"
if "isForward?" then
->[no] "Process controls"
if "continue processing?" then
-->[yes] ===RENDERING===
else
-->[no] ===REDIRECT_CHECK===
endif
else
-->[yes] ===RENDERING===
endif
if "is Post?" then
-->[yes] "Page.onPost()"
--> "Page.onRender()" as render
--> ===REDIRECT_CHECK===
else
-->[no] "Page.onGet()"
--> render
endif
else
-->[false] ===REDIRECT_CHECK===
endif
if "Do redirect?" then
->[yes] "redirect request"
--> ==BEFORE_DESTROY===
else
if "Do Forward?" then
-left->[yes] "Forward request"
--> ==BEFORE_DESTROY===
else
-right->[no] "Render page template"
--> ==BEFORE_DESTROY===
endif
endif
--> "Page.onDestroy()"
-->(*)
~~~

~~~
@enduml
```
~~~

Ploticus

prefab

Ploticus scripts are pretty verbose, it also has a `prefab` method of quickly creating a graphic from a data-file, but that is not supported at the moment.

Curves script

~~~
```{.ploticus im_out="fcb,img" caption="Created by Ploticus"}
#proc getdata
  data:
  0 1
  1 4
  2 2
  3 5
  4 7
  5 10
  6 7
  7 8
  8 4
  9 8
  10 7
  11 3

#proc areadef
  rectangle: 1 1 4 3
  xrange: 0 12
  yrange: 0 12
  xaxis.stubs: inc
  yaxis.stubs: inc

#proc lineplot
  xfield: 1
  yfield: 2
  pointsymbol: radius=0.03 shape=square style=filled
  linedetails: color=gray(0.8) width=0.5
  legendlabel: Raw data points
  legendsampletype: line+symbol
~~~

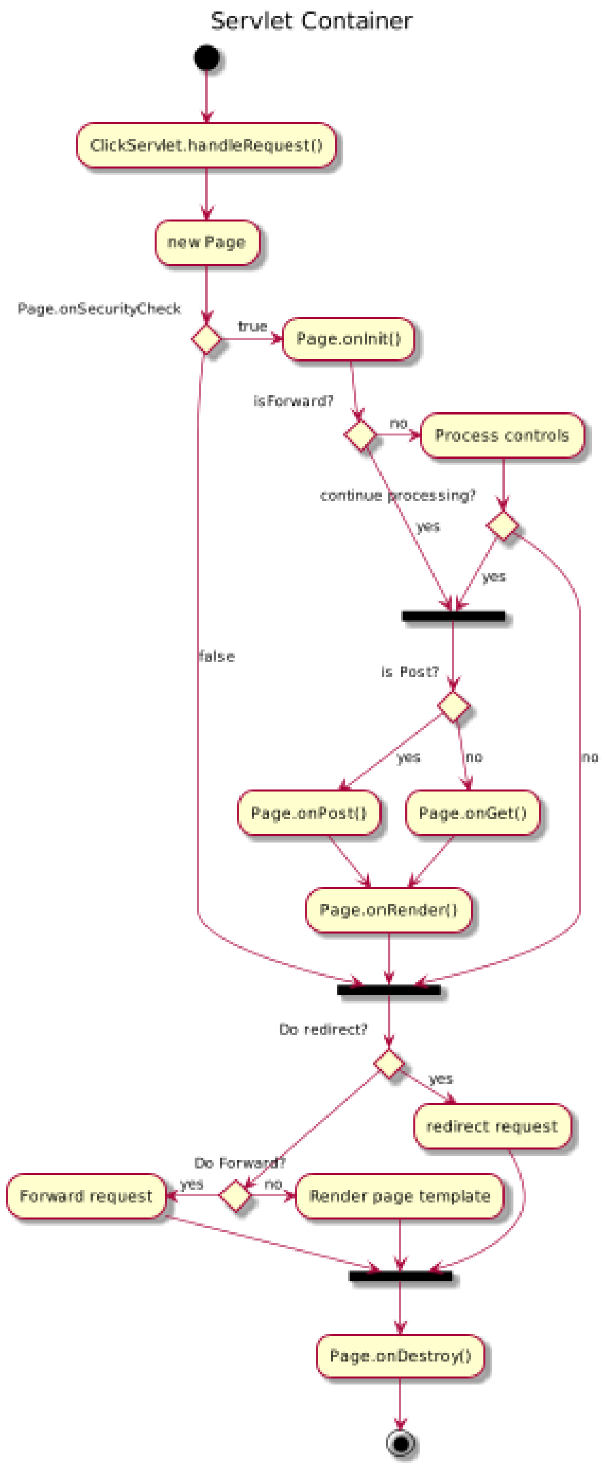


Figure 36: Created by plantuml

~~~~
#proc curvefit
  xfield: 1
  yfield: 2
  curvetype: movingavg
  order: 5
  linedetails: color=blue width=0.5
  legendlabel: Moving average (5 points)

#proc curvefit
  xfield: 1
  yfield: 2
  curvetype: regression
  linedetails: color=green width=0.5
  legendlabel: Linear regression

#proc curvefit
  xfield: 1
  yfield: 2
  curvetype: bspline
  order: 5
  linedetails: color=red width=0.5
  legendlabel: Bspline, order=5

#proc curvefit
  xfield: 1
  yfield: 2
  curvetype: average
  order: 5
  linedetails: color=black width=0.5
  legendlabel: Average (5 points)

#proc curvefit
  xfield: 1
  yfield: 2
  curvetype: interpolated
  linedetails: color=orange width=0.5
  legendlabel: Interpolated

#proc legend
  location: max+0.5 max

```
~~~~

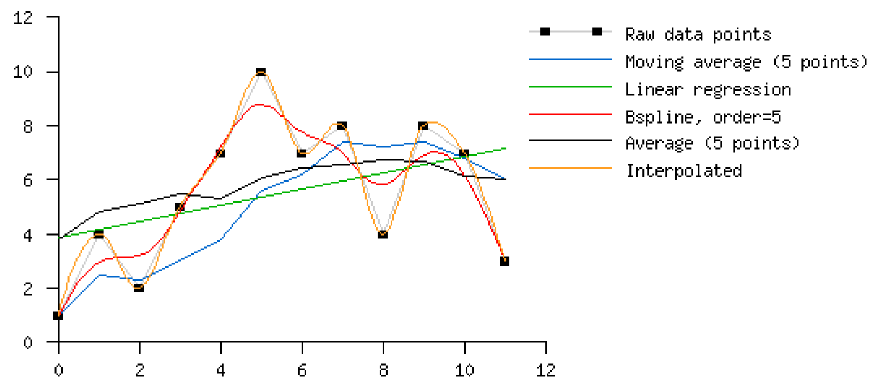


Figure 37: Created by Ploticus

Heatmap (script)

~~~
```{.ploticus im_out="fcb,img" caption="Created by Ploticus"}
#set SYM = "radius=0.08 shape=square style=filled"
#setifnotgiven CGI = "http://ploticus.sourceforge.net/cgi-bin/showcgiargs"


// read in the SNP map data file..
#proc getdata
file: dta/snpmap.dat
fieldnameheader: yes

// group into bins 4 cM wide..
filter:
   ##set A = $numgroup( @@2, 4, mid )
   @@1 @@A

// set up the plotting area
#proc areadef
rectangle: 1 1 6 3
areacolor: gray(0.2)
yscaletype: categories
clickmapurl: @CGI?chrom=@@YVAL&cM=@@XVAL
ycategories:
    1
    2
    3
    4
~~~

```
    5
    6
    7
    X

yaxis.stubs: usecategories
// yaxis.stubdetails: adjust=0.2,0
//yaxis.stubslide: 0.08
yaxis.label: chromosome
yaxis.axisline: no
yaxis.tics: no
yaxis.clickmap: xygrid

xrange: -3 120
xaxis.label: position (cM)
xaxis.axisline: no
xaxis.tics: no
xaxis.clickmap: xygrid
xaxis.stubs: inc 10
xaxis.stubrange: 0
// xaxis.stubdetails: adjust=0,0.15


// set up legend for color gradients..
#proc legendentry
sampletype: color
details: yellow
label: >20
tag: 21

#proc legendentry
sampletype: color
details: orange
label: 11-20
tag: 11

#proc legendentry
sampletype: color
details: red
label: 6 - 10
tag: 6

#proc legendentry
sampletype: color
details: lightpurple
label: 1 - 5
```

```
tag: 1

#proc legendentry
sampletype: color
details: gray(0.2)
label: 0
tag: 0


// use proc scatterplot to count # of instances and pick appropriate color from legend..
#proc scatterplot
yfield: chr
xfield: cM
cluster: yes
dupsleg: yes
rectangle: 4 1 outline


// display legend..
#proc legend
location: max+0.7 min+0.8
textdetails: size=6

```

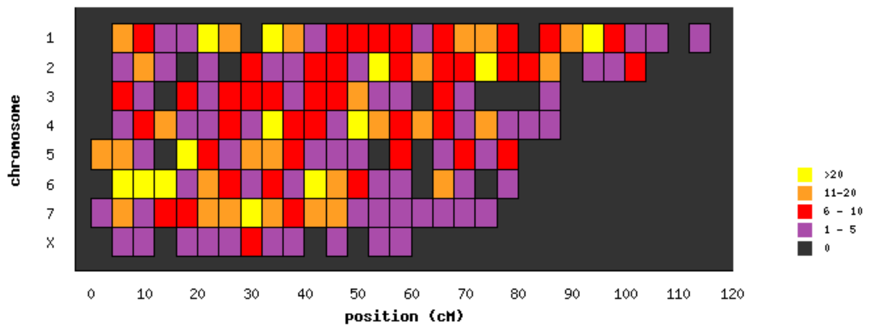


Figure 38: Created by Ploticus

Plotutils site

It includes:

- GNU *graph*, which plots 2-D datasets or data streams in real time.

- GNU ***plot***, which translates GNU Metafile format to any of the other formats.
- GNU ***tek2plot***, for translating legacy Tektronix data to any of the above formats.
- GNU ***pic2plot***, for translating the pic language (a scripting language for designing box-and-arrow diagrams) to any of the above formats. The pic language was designed at Bell Labs as an enhancement to the troff text formatter.
- GNU ***plotfont***, for displaying character maps of the fonts that are available in the above formats.
- GNU ***spline***, which does spline interpolation of data. It normally uses either cubic spline interpolation or exponential splines in tension, but it can function as a real-time filter under some circumstances.
- GNU ***ode***, which numerically integrates a system consisting of one or more ordinary differential equations.

Note:

- Imagine only wraps `plot` and `pic2plot` (`pic` is an alias for `pic2plot`).

graph

Each invocation of graph reads one or more datasets from files named on the command line or from standard input, and prepares a plot. There are many command-line options for adjusting the visual appearance of the plot. The following sections explain how to use the most frequently used options, by giving examples.

~~~~
```{.graph im_opt="-X x-axis -Y y-axis -f 0.1 --bitmap-size 200x200" im_out="fcb,img" captio
0.0  0.0
1.0  0.2
2.0  0.0
3.0  0.4
4.0  0.2
5.0  0.6
```
~~~~

plot

The GNU ***plot*** filter displays GNU graphics metafiles or translates them to other formats. It will take input from files specified on the command line or from standard input. The '-T' option is used to specify the desired output format. Supported output formats include "X", "png", "pnm", "gif", "svg", "ai", "ps", "cgm", "fig", "pcl", "hpgl", "regis", "tek", and "meta" (the default).

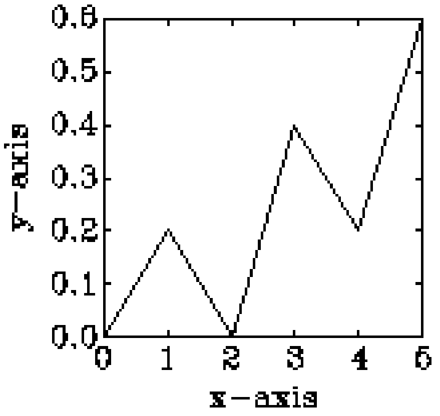


Figure 39: PlotUtil's graph

The metafile format is a device-independent format for storage of vector graphics. By default, it is a binary rather than a human-readable format (see Metafiles). Each of the graph, pic2plot, tek2plot, and plotfont utilities will write a graphics metafile to standard output if no '-T' option is specified on its command line. The GNU libplot graphics library may also be used to produce metafiles. Metafiles may contain arbitrarily many pages of graphics, but each metafile produced by graph contains only a single page.

plot, like the metafile format itself, is useful if you wish to preserve a vector graphics file, and display or edit it with more than one drawing editor.

~~~
```{.plot im_opt="--bitmap-size 300x200" im_out="fcb,img" caption="Created by plot"}
dta/input.meta
```
~~~

pic2plot

From the gnu website:

The pic language is a 'little language' that was developed at Bell Laboratories for creating box-and-arrow diagrams of the kind frequently found in technical papers and textbooks. A directory containing documentation on the pic language is distributed along with the plotting utilities. On most systems it is installed as /usr/share/pic2plot or /usr/local/share/pic2plot. The directory includes Brian Kernighan's original technical report on the language, Eric S. Raymond's tutorial

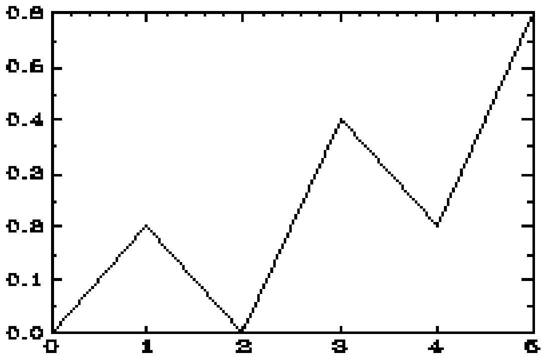


Figure 40: Created by plot

on the GNU implementation, and some sample pic macros contributed by the late W. Richard Stevens.

Protocol:

Protocol is a simple command-line tool that serves two purposes:

- Provide a simple way for engineers to have a look at standard network protocol headers, directly from the command-line, without having to google for the relevant RFC or for ugly header image diagrams.
- Provide a way for researchers and engineers to quickly generate ASCII RFC-like header diagrams for their own custom protocols.

TCP Header

~~~
```{.protocol im_out="fcb,stdout" caption="protocol"}
tcp
```
~~~

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
```

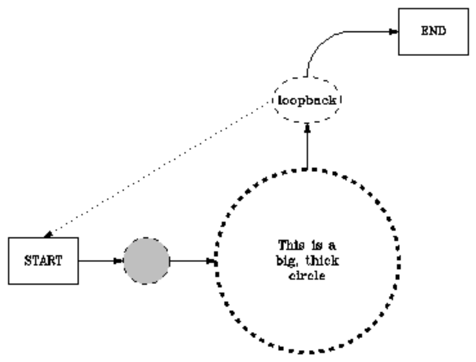


Figure 41: Created by pic

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Acknowledgment Number                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Offset|  Res. |     Flags     |             Window            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

and even custom layouts:

Custom packet

~~~
```{.protocol im_opt="--no-numbers" im_out="fcb,stdout" caption="protocol"}
Source:16,TTL:8,Reserved:40
```
~~~

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Source            |      TTL      |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               +
|                            Reserved                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

PyxPlot

ex01

```
set numerics complex
set xlabel r"$x$"
set ylabel r"$y$"
set zlabel r"$z$"
set xformat r"%s$\pi$"%(x/pi)
set yformat r"%s$\pi$"%(y/pi)
set xtics 3*pi ; set mxtics pi
set ytics 3*pi ; set mytics pi
set ztics
set key below
set size 6 square
set grid
plot 3d [-6*pi:6*pi][-6*pi:6*pi][-0.3:1] sinc(hypot(x,y)) \
     with surface col black \
```

```
    fillcol hsb(atan2($1,$2)/(2*pi)+0.5,hypot($1,$2)/30+0.2,$3*0.5+0.5)
```

SheBang

The `imagine` filter also features the `shebang` class which will run the fenced code block as a system script.

```
Codeblock class: shebang

    http://www.google.com/search?q=shebang+line

    runs:
    > <fname>.shebang {im_opt} <fname>.{im_fmt}

    class->cmd
      shebang -> shebang

Metadata options
    imagine.im_out: img,fcb
    imagine.im_log: 4
```

bash

~~~
```{.shebang im_out="fcb,stdout,img"}
#!/bin/bash
echo "This script is saved as :" $0
echo "and requires boxes to be available"
echo
echo "Its (user) executable flag is set:"
echo
echo $(ls -lpah $0 | tr -s ' ' | cut -d' ' -f1,9) | boxes -d peek
echo
echo "This script won't produce: $1"
echo
echo "But since 'im_out'-option above includes a request for img,"
echo "a line is included in the output document, like:"
echo
echo "?? missing ${1}"
echo
echo "If a shebang script returns with an exit code other than 0 (zero)"
echo "the command fails and the original code block is retained"
echo
echo "If im_out=".." includes 'stdout' (like in this case), any text"
~~~

```
echo "on stdout is included in its own CodeBlock"
```
```

This script is saved as : pd-images/10d79e342f6a160e91cf086232170c43fe8ef146.shebang
and requires boxes to be available

Its (user) executable flag is set:

/*       _\|/_
         (o o)
 +----oOO-{_}-OOo-----------------------------------------------------+
 |-rwxrw-r-- pd-images/10d79e342f6a160e91cf086232170c43fe8ef146.shebang|
 +-------------------------------------------------------------------*/

This script won't produce: pd-images/10d79e342f6a160e91cf086232170c43fe8ef146.png

But since 'im_out'-option above includes a request for img,
a line is included in the output document, like:

?? missing pd-images/10d79e342f6a160e91cf086232170c43fe8ef146.png

If a shebang script returns with an exit code other than 0 (zero)
the command fails and the original code block is retained

If im_out=.. includes 'stdout' (like in this case), any text
on stdout is included in its own CodeBlock
```

?? missing pd-images/10d79e342f6a160e91cf086232170c43fe8ef146.png

manpage

~~~
```{.shebang im_out="fcb,stdout"}
#!/bin/bash
MANWIDTH=85 man 6 figlist | col -bx | iconv -t ascii//TRANSLIT
```
~~~

```
FIGLIST(6)                        Games Manual                          FIGLIST(6)

NAME
       figlist - lists figlet fonts and control files

SYNOPSIS
       figlist [ -d directory ]

DESCRIPTION
       Lists  all  fonts  and  control  files  in figlet's default font directory.
```

```
       Replaces "figlet -F", which was removed from figlet version 2.1.

EXAMPLES
       To use figlist with its default settings, simply type

              example% figlist

       To list all the font and control files in /usr/share/fonts/figlet

              example% figlist -d /usr/share/fonts/figlet

AUTHORS
       figlist was written by Glenn Chappell <ggc@uiuc.edu>

       This manual page was written by Jonathon Abbott for the Debian Project.

SEE ALSO
       figlet(6), chkfont(6), showfigfonts(6)

v2.2.5                        31 May 2012                        FIGLIST(6)
```

Matplotlib

Agg

```
#!/usr/bin/env python

import sys
import numpy as np
import matplotlib as mpl
mpl.use('Agg')
import matplotlib.pyplot as plt

t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2*np.pi*t)
plt.plot(t, s)

plt.xlabel('time (s)')
plt.ylabel('voltage (mV)')
plt.title('A simple plot')
plt.grid(True)
plt.savefig(sys.argv[-1])
```

Fill with alpha

```
#!/usr/bin/env python

import sys
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2 * np.pi, 500)
y1 = np.sin(2 * x)
y2 = np.sin(3 * x)

fig, ax = plt.subplots()
ax.fill(x, y1, 'b', x, y2, 'r', alpha=0.2)
fig.savefig(sys.argv[-1])
```

Axis scale transformations

```
#!/usr/bin/env python

import sys
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import NullFormatter

np.random.seed(1)
# make up some data in the interval ]0, 1[
y = np.random.normal(loc=0.5, scale=0.4, size=1000)
y = y[(y > 0) & (y < 1)]
y.sort()
x = np.arange(len(y))

# plot with various axes scales
fig, axs = plt.subplots(2, 2, sharex=True)
fig.subplots_adjust(left=0.08, right=0.98, wspace=0.3)

# linear
ax = axs[0, 0]
ax.plot(x, y)
ax.set_yscale('linear')
ax.set_title('linear')
ax.grid(True)


# log
```

```
ax = axs[0, 1]
ax.plot(x, y)
ax.set_yscale('log')
ax.set_title('log')
ax.grid(True)


# symmetric log
ax = axs[1, 1]
ax.plot(x, y - y.mean())
ax.set_yscale('symlog', linthreshy=0.02)
ax.set_title('symlog')
ax.grid(True)

# logit
ax = axs[1, 0]
ax.plot(x, y)
ax.set_yscale('logit')
ax.set_title('logit')
ax.grid(True)
ax.yaxis.set_minor_formatter(NullFormatter())

fig.savefig(sys.argv[-1])
```

Coherence of two signals

```
#!/usr/bin/env python

import sys
import numpy as np
import matplotlib.pyplot as plt

plt.subplots_adjust(wspace=0.5)                # space the subplots

dt = 0.01
t = np.arange(0, 30, dt)
nse1 = np.random.randn(len(t))                 # white noise 1
nse2 = np.random.randn(len(t))                 # white noise 2
r = np.exp(-t/0.05)

cnse1 = np.convolve(nse1, r, mode='same')*dt   # colored noise 1
cnse2 = np.convolve(nse2, r, mode='same')*dt   # colored noise 2

# two signals with a coherent part and a random part
s1 = 0.01*np.sin(2*np.pi*10*t) + cnse1
```

```
s2 = 0.01*np.sin(2*np.pi*10*t) + cnse2

plt.subplot(211)
plt.plot(t, s1, t, s2)
plt.xlim(0, 5)
plt.xlabel('time')
plt.ylabel('s1 and s2')
plt.grid(True)

plt.subplot(212)
cxy, f = plt.cohere(s1, s2, 256, 1./dt)
plt.ylabel('coherence')
plt.savefig(sys.argv[-1])
```

3D image

```
#!/usr/bin/env python
import sys
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import pyplot as plt
import numpy as np


fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Create the mesh in polar coordinates and compute corresponding Z.
r = np.linspace(0, 1.25, 50)
p = np.linspace(0, 2*np.pi, 50)
R, P = np.meshgrid(r, p)
Z = ((R**2 - 1)**2)

# Express the mesh in the cartesian system.
X, Y = R*np.cos(P), R*np.sin(P)

# Plot the surface.
ax.plot_surface(X, Y, Z, cmap=plt.cm.YlGnBu_r)

# Tweak the limits and add latex math labels.
ax.set_zlim(0, 1)
ax.set_xlabel(r'$\phi_\mathrm{real}$')
ax.set_ylabel(r'$\phi_\mathrm{im}$')
ax.set_zlabel(r'$V(\phi)$')

plt.savefig(sys.argv[-1])
```

Pygal

- uses python3
- needs cairosvg, tinycss, cssselect to render to png

Solid Gauges

~~~
```{.shebang im_out="fcb,img" caption="Created by Pygal"}
#!/usr/bin/env python3

import sys
import pygal

gauge = pygal.SolidGauge(inner_radius=0.70)
percent_formatter = lambda x: '{:.10g}%'.format(x)
dollar_formatter = lambda x: '{:.10g}$'.format(x)
gauge.value_formatter = percent_formatter

gauge.add('Series 1', [{'value': 225000, 'max_value': 1275000}],
          formatter=dollar_formatter)
gauge.add('Series 2', [{'value': 110, 'max_value': 100}])
gauge.add('Series 3', [{'value': 3}])
gauge.add(
    'Series 4', [
        {'value': 51, 'max_value': 100},
        {'value': 12, 'max_value': 100}])
gauge.add('Series 5', [{'value': 79, 'max_value': 100}])
gauge.add('Series 6', 99)
gauge.add('Series 7', [{'value': 100, 'max_value': 100}])

gauge.render_to_png(sys.argv[-1])
```
~~~

Basic XY line

~~~
```{.shebang im_out="fcb,img" caption="Created by Pygal"}
#!/usr/bin/env python3

import sys
import pygal
from math import cos

xy_chart = pygal.XY()
xy_chart.title = 'XY Cosinus'
~~~

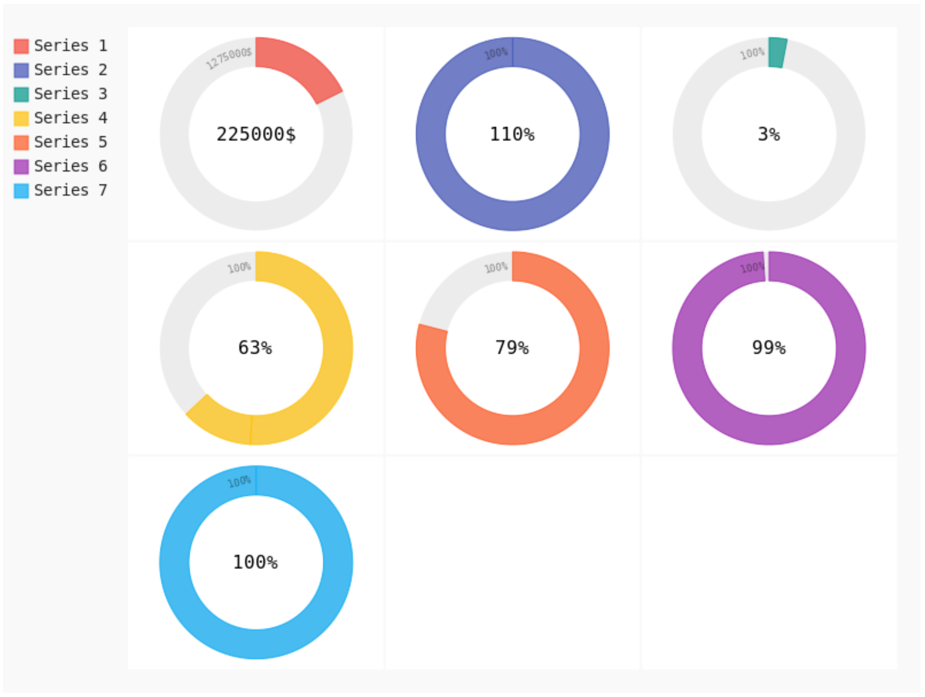


Figure 42: Created by Pygal

```
xy_chart.add('x = cos(y)', [(cos(x / 10.), x / 10.) for x in range(-50, 50, 5)])
xy_chart.add('y = cos(x)', [(x / 10., cos(x / 10.)) for x in range(-50, 50, 5)])
xy_chart.add('x = 1',  [(1, -5), (1, 5)])
xy_chart.add('x = -1', [(-1, -5), (-1, 5)])
xy_chart.add('y = 1',  [(-5, 1), (5, 1)])
xy_chart.add('y = -1', [(-5, -1), (5, -1)])
xy_chart.render_to_png(sys.argv[-1])
```
```

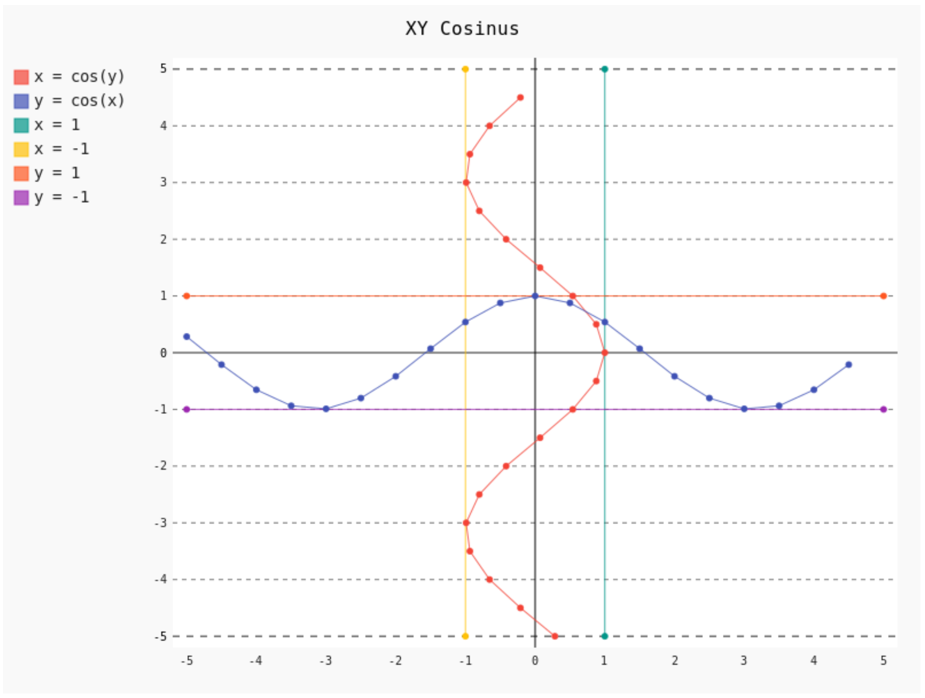

XY Cosinus
x = cos(y)
y = cos(x)
x = 1
x = -1
y = 1
y = -1
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4
5

Figure 43: Created by Pygal

Octave

Earlier example of Octave, but now run as a script.

```
```{.shebang im_out="fcb,img" caption="Created by Octave"}
#!/usr/bin/env octave

figure(1, 'visible', 'off');

x = 0:0.01:2*pi;
a = sin(x);
```

~~~
b = cos(2*x);
c = sin(4*x);
d = 2*sin(3*x);
plot(x,a,x,b,x,c,x,d, "linewidth", 2);
set(gca, "xlim", [0,2*pi], "fontsize", 15);
title("sinusoids");

print(1, argv(){1})
```
~~~

?? missing pd-images/3482c42cc7357298da476f7d32a10441018e53a7.png

ChartDirector

The yellow bars below the images created by ChartDirector are because this is the demo-version without a license.

Line Chart

```
#!/usr/bin/python
import sys
from pychartdir import *

data0 = [42, 49, NoValue, 38, 64, 56, 29, 41, 44, 57]
data1 = [65, 75, 47, 34, 42, 49, 73, NoValue, 90, 69, 66, 78]
data2 = [NoValue, NoValue, 25, 28, 38, 20, 22, NoValue, 25, 33, 30, 24]
labels = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec'
c = XYChart(600, 360, brushedSilverColor(), Transparent, 2)
c.setRoundedFrame()
title = c.addTitle("Product Line Global Revenue", "timesbi.ttf", 18)
title.setMargin2(0, 0, 6, 6)
c.addLine(10, title.getHeight(), c.getWidth() - 11, title.getHeight(), LineColor)
legendBox = c.addLegend(c.getWidth() / 2, title.getHeight(), 0, "arialbd.ttf", 10)
legendBox.setAlignment(TopCenter)
legendBox.setBackground(Transparent, Transparent)
c.setPlotArea(70, 75, 460, 240, -1, -1, Transparent, 0x000000, -1)
c.xAxis().setLabels(labels)
c.syncYAxis()
c.yAxis().setTickDensity(30)
c.xAxis().setColors(Transparent)
c.yAxis().setColors(Transparent)
c.yAxis2().setColors(Transparent)
c.xAxis().setMargin(15, 15)
c.xAxis().setLabelStyle("arialbd.ttf", 8)
c.yAxis().setLabelStyle("arialbd.ttf", 8)
```

```
c.yAxis2().setLabelStyle("arialbd.ttf", 8)
c.yAxis().setTitle("Revenue in USD millions", "arialbi.ttf", 10)
c.yAxis2().setTitle("Revenue in USD millions", "arialbi.ttf", 10)
layer0 = c.addLineLayer2()
layer0.addDataSet(data0, 0xff0000, "Quantum Computer").setDataSymbol(GlassSphere2Shape, 11)
layer0.setLineWidth(3)
layer1 = c.addLineLayer2()
layer1.addDataSet(data1, 0x00ff00, "Atom Synthesizer").setDataSymbol(GlassSphere2Shape, 11)
layer1.setLineWidth(3)
layer1.setGapColor(c.dashLineColor(0x00ff00))
layer2 = c.addLineLayer2()
layer2.addDataSet(data2, 0xff6600, "Proton Cannon").setDataSymbol(GlassSphere2Shape, 11)
layer2.setLineWidth(3)
layer2.setGapColor(SameAsMainColor)
c.layoutLegend()
c.packPlotArea(15, legendBox.getTopY() + legendBox.getHeight(), c.getWidth() - 16, c.getHeig
    ) - 25)

c.makeChart(sys.argv[-1])
```

Surface

```
#!/usr/bin/python
import sys
from pychartdir import *

dataX = [0.5, 1.9, 4.9, 1.0, 8.9, 9.8, 5.9, 2.9, 6.8, 9.0, 0.0, 8.9, 1.9, 4.8, 2.4, 3.4, 7.9
    4.8, 7.5, 9.5, 0.4, 8.9, 0.9, 5.4, 9.4, 2.9, 8.9, 0.9, 8.9, 10.0, 1.0, 6.8, 3.8, 9.0, 5.
    4.9, 4.5, 2.0, 5.4, 0.0, 10.0, 3.9, 5.4, 5.9, 5.8, 0.3, 4.4, 8.3]
dataY = [3.3, 3.0, 0.7, 1.0, 9.3, 4.5, 8.4, 0.1, 0.8, 0.1, 9.3, 1.8, 4.3, 1.3, 2.3, 5.4, 6.9
    9.8, 7.5, 1.8, 1.4, 4.5, 7.8, 3.8, 4.0, 2.9, 2.4, 3.9, 2.9, 2.3, 9.3, 2.0, 3.4, 4.8, 2.3
    2.3, 1.5, 7.8, 4.5, 0.9, 6.3, 2.4, 6.9, 2.8, 1.3, 2.9, 6.4, 6.3]
dataZ = [6.6, 12.5, 7.4, 6.2, 9.6, 13.6, 19.9, 2.2, 6.9, 3.4, 8.7, 8.4, 7.8, 8.0, 9.4, 11.9,
    15.7, 12.0, 13.3, 9.6, 6.4, 9.0, 6.9, 4.6, 9.7, 10.6, 9.2, 7.0, 6.9, 9.7, 8.6, 8.0, 13.6
    5.9, 9.0, 3.2, 8.3, 9.7, 8.2, 6.1, 8.7, 5.6, 14.9, 9.8, 9.3, 5.1, 10.8, 9.8]
c = SurfaceChart(680, 550, brushedSilverColor(), 0x888888)
c.setRoundedFrame(0xffffff, 20, 0, 20, 0)
title = c.addTitle("Surface Created Using Scattered Data Points", "timesi.ttf", 20)
title.setMargin2(0, 0, 8, 8)
c.addLine(10, title.getHeight(), c.getWidth() - 10, title.getHeight(), 0x000000, 2)
c.setPlotRegion(290, 235, 360, 360, 180)
c.setViewAngle(45, -45)
c.setPerspective(30)
c.setData(dataX, dataY, dataZ)
cAxis = c.setColorAxis(660, 80, TopRight, 200, Right)
```

```
cAxis.setTitle("Z Title Placeholder", "arialbd.ttf", 12)
cAxis.setBoundingBox(0xeeeeee, 0x888888)
cAxis.setRoundedCorners(10, 0, 10, 0)
c.setSurfaceAxisGrid(0xcc000000)
c.setContourColor(0x80ffffff)
c.setWallColor(0x000000)
c.setWallGrid(0xffffff, 0xffffff, 0xffffff, 0x888888, 0x888888, 0x888888)
c.setWallThickness(0, 0, 0)
c.setWallVisibility(1, 0, 0)
c.xAxis().setTitle("X Title\nPlaceholder", "arialbd.ttf", 12)
c.yAxis().setTitle("Y Title\nPlaceholder", "arialbd.ttf", 12)
c.makeChart(sys.argv[-1])
```

Gauge

```
#!/usr/bin/python
import sys
from pychartdir import *

value = 54
colorList = [0x0033dd, 0xaaaa00]
mainColor = colorList[1]
size = 300
outerRadius = int(size / 2 - 2)
scaleRadius = int(outerRadius * 92 / 100)
colorScaleRadius = int(scaleRadius * 43 / 100)
colorScaleWidth = int(scaleRadius * 10 / 100)
tickLength = int(scaleRadius * 10 / 100)
tickWidth = int(scaleRadius * 1 / 100 + 1)
fontSize = int(scaleRadius * 13 / 100)
readOutRadiusRatio = 0.333333333333
readOutFontSize = int(scaleRadius * 24 / 100)
m = AngularMeter(size, size, 0x000000)
m.setColor(TextColor, 0xffffff)
m.setColor(LineColor, 0xffffff)
m.setMeter(size / 2, size / 2, scaleRadius, -180, 90)
bgGradient = [0, mainColor, 0.5, m.adjustBrightness(mainColor, 0.75), 1, m.adjustBrightness(
    mainColor, 0.15)]
m.addRing(0, outerRadius, m.relativeRadialGradient(bgGradient, outerRadius * 0.66))
neonGradient = [0.89, Transparent, 1, mainColor, 1.07, Transparent]
m.addRing(int(scaleRadius * 85 / 100), outerRadius, m.relativeRadialGradient(neonGradient))
m.addRing(scaleRadius, int(scaleRadius + scaleRadius / 80), m.adjustBrightness(mainColor, 2)
m.setScale(0, 100, 10, 5, 1)
m.setLabelStyle("ariali.ttf", fontSize)
m.setTickLength( - tickLength,  - int(tickLength * 80 / 100),  - int(tickLength * 60 / 100))
```

```
m.setLineWidth(0, tickWidth, int((tickWidth + 1) / 2), int((tickWidth + 1) / 2))
smoothColorScale = [0, 0x0000ff, 25, 0x0088ff, 50, 0x00ff00, 75, 0xdddd00, 100, 0xff0000]
highColorScale = [70, Transparent, 100, 0xff0000]
m.addColorScale(highColorScale)
m.addPointer2(value, 0xff0000, -1, TriangularPointer2, 0.4, 0.6, 6)
m.setCap2(Transparent, m.adjustBrightness(mainColor, 0.3), m.adjustBrightness(mainColor, 1.5
    0.75, 0, readOutRadiusRatio, 0.015)
m.addText(size / 2, size / 2, m.formatValue(value, "{value|0}"), "ariali.ttf", readOutFontSi
    m.adjustBrightness(mainColor, 2.5), Center).setMargin(0)
m.addGlare(scaleRadius)
m.makeChart(sys.argv[-1])
```