Imagine

Imagine

```
Imagine
  A pandoc filter that turns fenced codeblocks into graphics or ascii art by
  wrapping some external command line utilities, such as:

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

Installation

  1. Put `imagine.py` anywhere along $PATH (pandoc's search path for filters).
  2. % sudo pip install (mandatory):
       - pandocfilters
  3. % sudo apt-get install (1 or more of):
       - graphviz,   http://graphviz.org
       - plantuml,   http://plantuml.com
       - ditaa,      http://ditaa.sourceforge.net
       - figlet,     http://www.figlet.org
       - boxes,      http://boxes.thomasjensen.com
       - plotutils,  https://www.gnu.org/software/plotutils/
       - gnuplot,    http://www.gnuplot.info/
       - asymptote,  http://asymptote.sourceforge.net/
       - pyxplot,    http://pyxplot.org.uk/
       - ploticus,   http://ploticus.sourceforge.net/doc/welcome.html

     % sudo pip install:
       - blockdiag,  http://blockdiag.com
       - phantomjs,  http://phantomjs.org/ (for mermaid)

     % git clone
       - protocol,   https://github.com/luismartingarcia/protocol.git
```

```
    % npm install:
      - -g mermaid, https://knsv.github.io/mermaid (and pip install phantomjs)
```

Pandoc usage

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

Markdown usage

```
              or                                  or
  ```cmd      |   ```{.cmd options="extras"}     |   ```{prog=cmd}
  source      |   source                          |   source
  ```         |   ```                             |   ```
  simple   with `options`   with `prog`
```

Imagine understands/consumes these fenced codeblock key,val-attributes:
- `options` used to feed extra arguments to the external command
- `prog`   used when cmd is not an appropiate document class
- `keep`   if True, keeps a reconstructued copy of the original CodeBlock

Notes:
- if `cmd` is not found, the codeblock is kept as-is.
- input/output filenames are generated from a hash of the fenced codeblock.
- subdir `pd-images` is used to store any input/output files
- if an output filename exists, it is not regenerated but simply linked to.
- `packetdiag` & `sfdp`s underlying libraries seem to have some problems.

How Imagine works

The general format for an external command looks something like:

```
   % cmd <options> <inputfile> <outputfile>
```

Input/Output filenames are generated using `pandocfilters.get_filename4code` supplying both the codeblock and its attributes as a string for hashing. If the input file doesn't exist it is generated by writing the code in the fenced codeblock. Hence, if you change the code and/or the attributes, new files will result.

Imagine does no clean up so, after a while, you might want to clear the `pd-images` subdirectory.

Some commands are Imagine's aliases for system commands. Examples are

`graphviz` which is an alias for `dot` and `pic` which is an alias for `pic2plot`. Mainly because that allows the alias names to be used as a cmd for a fenced codeblock (ie. ```graphviz to get ```dot)

Some commands like `figlet` or `boxes` produce output on stdout. This text is captured and used to replace the code in the fenced code block.

Some commands like `plot` interpret the code in the fenced code block as an input filename to convert to some other output format.

If a command fails for some reason, the fenced codeblock is kept as is. In that case, the output produced by Imagine on stderr hopefully provides some usefull info.

Security

Imagine just wraps some commands and provides no checks.

So use it with care and make sure you understand the fenced codeblocks before running it through the filter.

Imagine command

Finally, a quick way to read this help text again, is to include a fenced codeblock in your markdown document as follows:

```
```imagine
```
```

That's it, enjoy!

No cmd, no change

Anonymous CodeBlock

This code block is anonymous and not processed by Imagine.

A Python CodeBlock

```python
if processed_by(Imagine):
    raise Expection('Not ignored by Imagine!')
```

```
else:
    print "Great, if you're reading this, it passed through Imagine unharmed"
```

Granddaddy GnuPlot

gnuplot Gnuplot is a portable command-line driven graphing utility for Linux, OS/2, MS Windows, OSX, VMS, and many other platforms. The source code is copyrighted but freely distributed (i.e., you don't have to pay for it). It was originally created to allow scientists and students to visualize mathematical functions and data interactively, but has grown to support many non-interactive uses such as web scripting. It is also used as a plotting engine by third-party applications like Octave. Gnuplot has been supported and under active development since 1986.

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 keep="True" 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 linewi
set samples 200, 200
plot [-30:20] besj0(x)*0.12e1 with impulses, (x**besj0(x))-2.5 with points
```
```

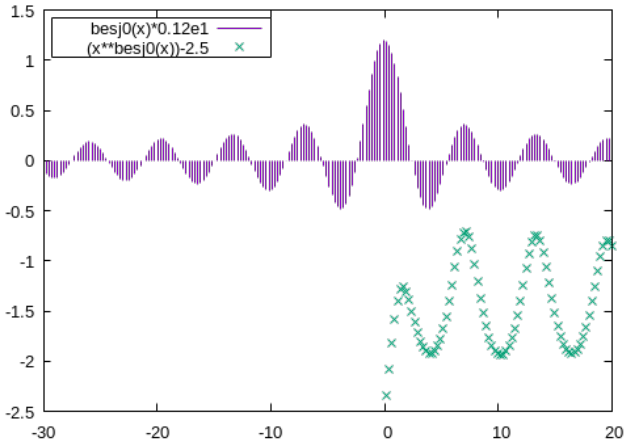

Figure 1: Created by GnuPlot

real sine

```{.gnuplot keep="True" 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 400, 400
plot [-10:10] real(sin(x)**besj0(x))
```

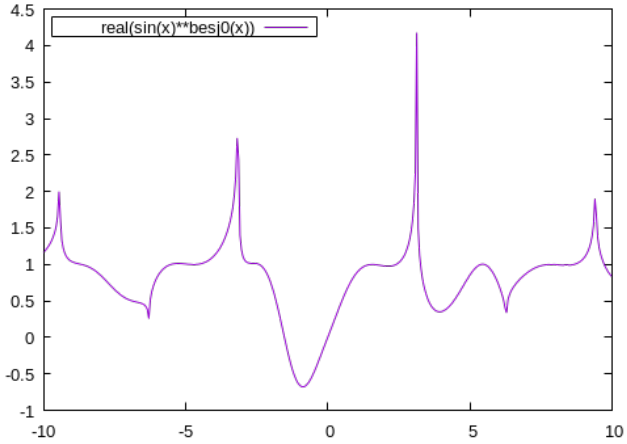

Figure 2: Created by GnuPlot

Surface

```{.gnuplot keep="True" 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
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)
```

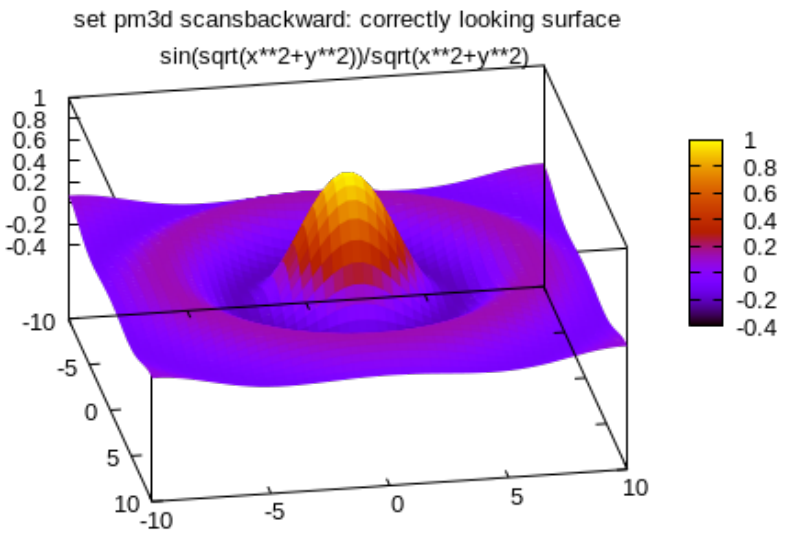

Figure 3: Another GnuPlot example

Interlocking Tori

```
```{.gnuplot keep="True" 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
splot cos(u)+.5*cos(u)*cos(v),sin(u)+.5*sin(u)*cos(v),.5*sin(v) with lines,       1+cos(u)+.
```
```

GNU's plotutils package

plotutils site It includes:

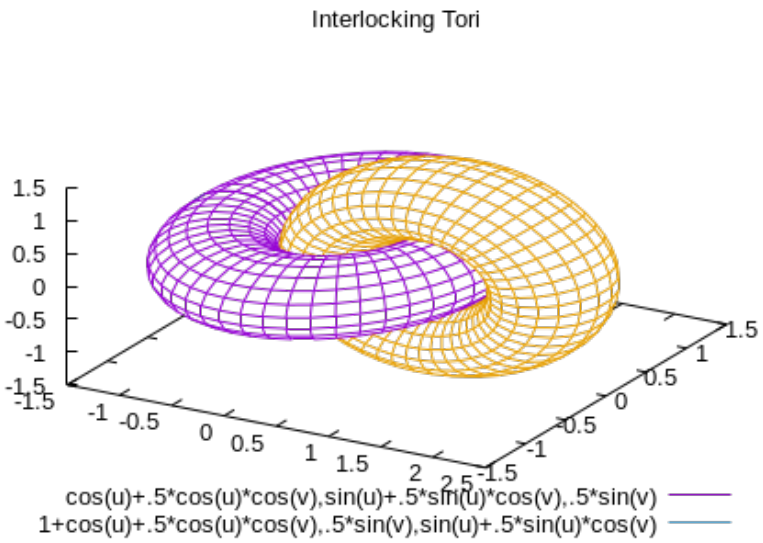

Figure 4: Gnuplot's interlocking Tori example

- GNU *graph*, which plots 2-D datasets or data streams in real time. Being designed for command-line use, it can be used in shell scripts. It produces output on an X Window System display, in SVG format, in PNG format, in PNM format, in pseudo-GIF format, in WebCGM format, in Illustrator format, in Postscript format, in PCL 5 format, in HP-GL/2 format, in Fig format (editable with the xfig drawing editor), in ReGIS format, in Tektronix format, or in GNU Metafile format.

  Output in Postscript format may be edited with the idraw drawing editor. idraw is available in the ivtools package from Vectaport, Inc. Both xfig and idraw are free software.

- 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.

We developed these command-line programs to replace the Unix command-line programs graph, plot, and spline. The GNU versions are far more powerful, and are free software.

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 options="-X x-axis -Y y-axis -f 0.1 --bitmap-size 200x200" keep="True" caption="F
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).

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

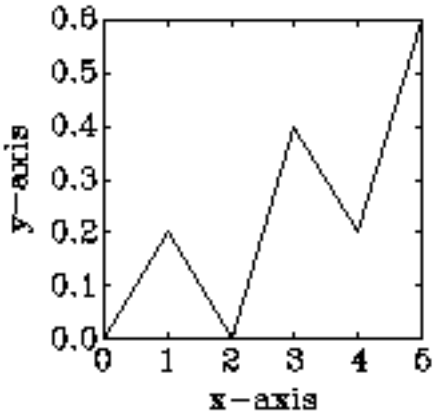

Figure 5: PlotUtil's graph

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 options="--bitmap-size 300x200" keep="True" caption="Created by plot"}
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 on the GNU implementation, and some sample pic macros contributed by the late W. Richard Stevens.

```
```{.pic keep="True" width="80%" caption="Created by pic"}
.PS
box "START"; arrow; circle dashed filled; arrow
```
```

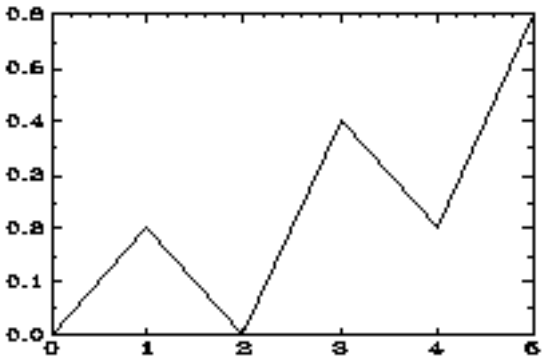

Figure 6: Created by plot

```
circle diam 2 thickness 3 "This is a" "big, thick" "circle" dashed; up
arrow from top of last circle; ellipse "loopback" dashed
arrow dotted from left of last ellipse to top of last box
arc cw radius 1/2 from top of last ellipse; arrow
box "END"
.PE
```

Asymptote

Asymptote: The Vector Graphics Language.

Asymptote is a powerful descriptive vector graphics language that provides a natural coordinate-based framework for technical drawing. Labels and equations are typeset with LaTeX, for high-quality PostScript output. A major advantage of Asymptote over other graphics packages is that it is a programming language, as opposed to just a graphics program.

Features of Asymptote:

- provides a portable standard for typesetting mathematical figures, just as TeX/LaTeX has become the standard for typesetting equations;
- generates high-quality PostScript, PDF, SVG, or 3D PRC vector graphics;
- embeds 3D vector PRC graphics within PDF files;

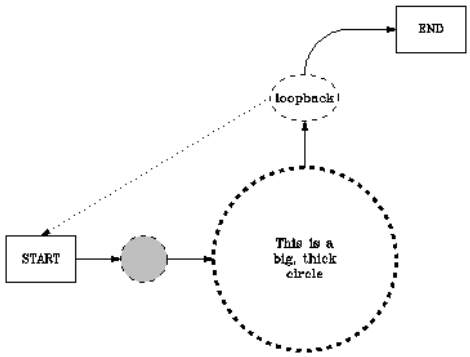

Figure 7: Created by pic

- inspired by MetaPost, with a much cleaner, powerful C++-like programming syntax and IEEE floating-point numerics;
- runs on all major platforms (UNIX, MacOS, Microsoft Windows);
- mathematically oriented (e.g. rotation of vectors by complex multiplication);
- LaTeX typesetting of labels (for document consistency);
- uses simplex method and deferred drawing to solve overall size constraint issues between fixed-sized objects (labels and arrowheads) and objects that should scale with figure size;
- fully generalizes MetaPost path construction algorithms to three dimensions;
- compiles commands into virtual machine code for speed without sacrificing portability;
- high-level graphics commands are implemented in the Asymptote language itself, allowing them to be easily tailored to specific applications.

Notes:

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

a plot

````
```{.asy keep="True" 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());
```
````

a sphere

````
```{.asy keep="True" caption="Created by Asymptote"}
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);
````
````

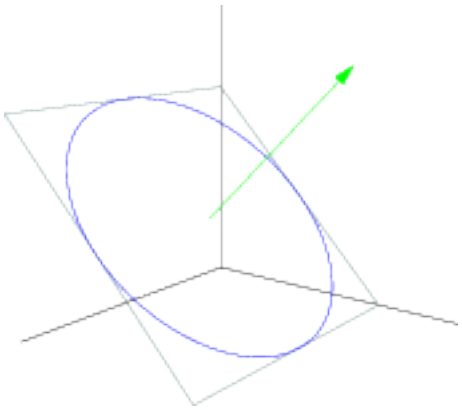

Figure 8: Created by Asymptote

```
surface backHemisphere = surface(myarc, angle1=0, angle2=180, c=0, axis=Z, n=10);
surface frontHemisphere = surface(myarc, angle1=180, angle2=360, c=0, axis=Z, n=10);
draw(backHemisphere, surfacepen=material(white+opacity(0.8), ambientpen=white), meshpen=gray
draw(O--X, blue+linewidth(1pt));
```

PyxPlot

ex01

```
```{.pyxplot keep="True" caption="Created by PyxPlot"}
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)
```
```

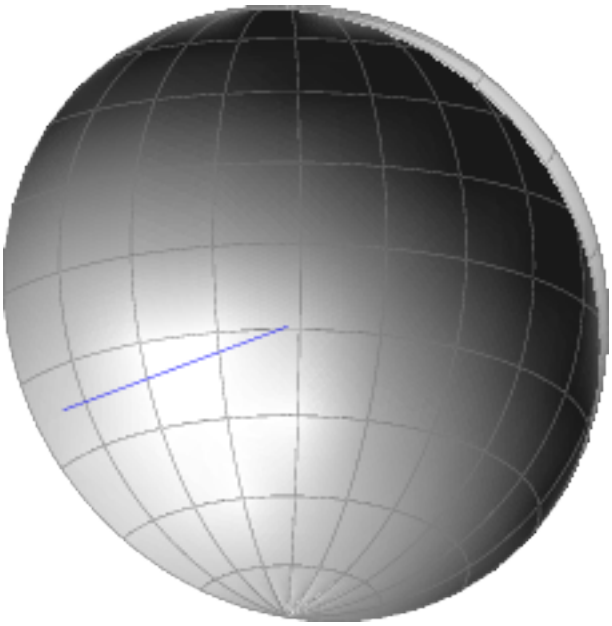

Figure 9: Created by Asymptote

Ploticus

Ploticus

Ploticus is a free GPL software utility that can produce various types of plots and graphs like shown here and here. Data input is usually csv files or text files such as used with sort, awk, etc. Output options are GIF, PNG, PostScript, SVG and some others. HTML imagemaps are supported. Ploticus can produce just-in-time plots in dynamic web content systems, or in batch production settings. It can be invoked from your command line, in shell scripts, via system() calls in web content environments and other programs, or via the libploticus API.

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 keep="True" caption="Created by Ploticus"}
#proc getdata
```
```

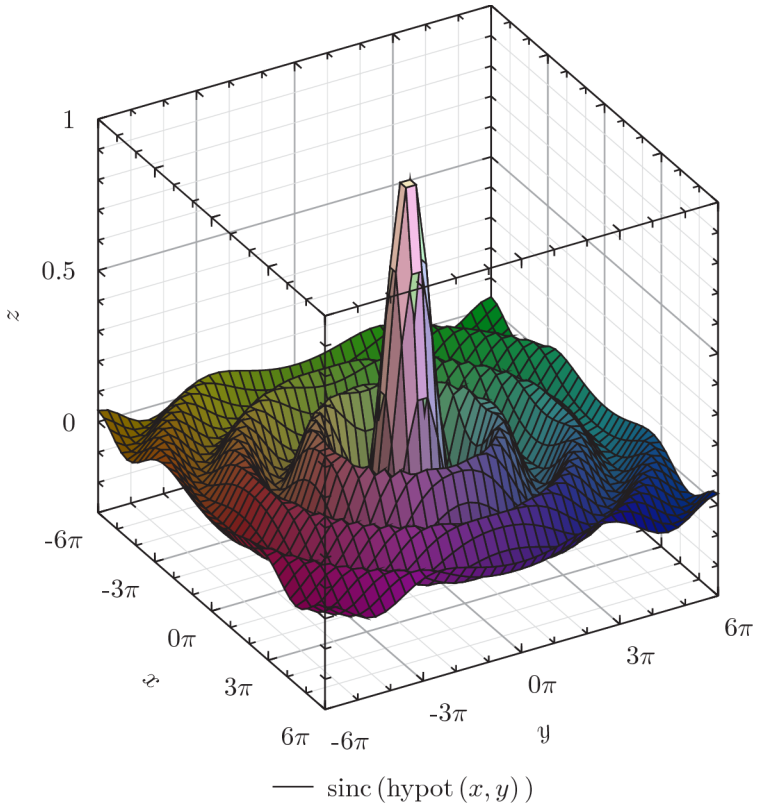

Figure 10: Created by PyxPlot

```
  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

#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

```

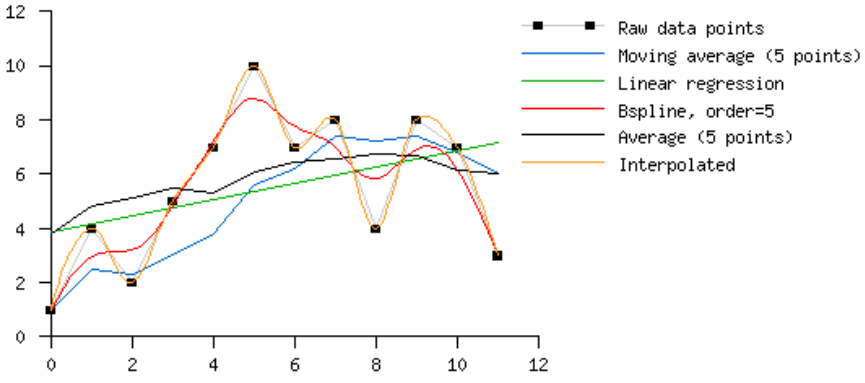

Figure 11: Created by Ploticus

Heatmap (script)

```{.ploticus keep="True" 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: 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

```

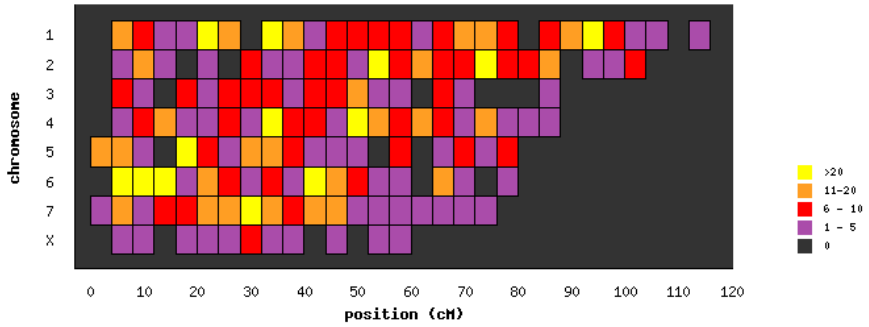

Figure 12: Created by Ploticus

Graphviz

graphviz.org site: `Graphviz` is open source graph visualization software. Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks. It has important applications in networking, bioinformatics, software engineering, database and web design, machine learning, and in visual interfaces for other technical domains.

`Graphviz` defaults to dot

```
```{prog="dot" options="-Gsize=4,1.5" caption="FSM layout by dot" keep="True"}

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)" ];
    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)" ];
}

```

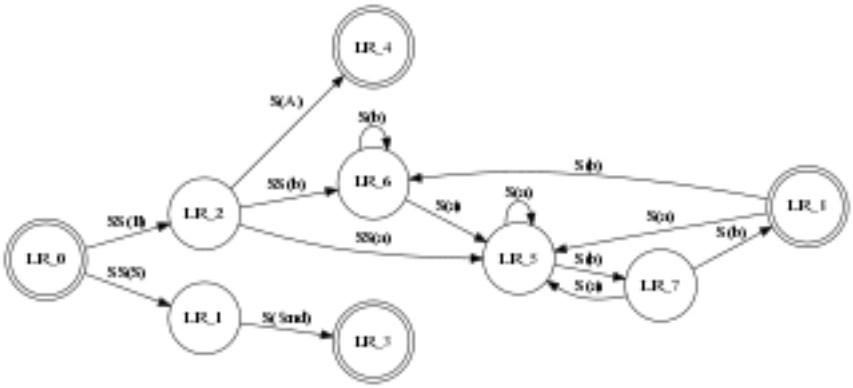

Figure 13: FSM layout by dot

fdp

```{.graphviz prog="fdp" options="-Gsize=2,3" caption="Created by fdp" keep="True"}

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

```

sfdp *(fails)*

```
graph G {
```

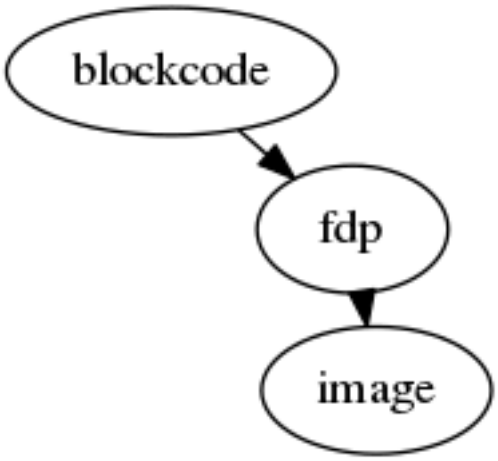

Figure 14: Created by fdp

```
size="2,2"
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;
}
```

neato

States in a kernel OS plotted by `neato`:

```
```{.graphviz prog="neato" caption="Created by neato" keep="True"}
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;
}
```

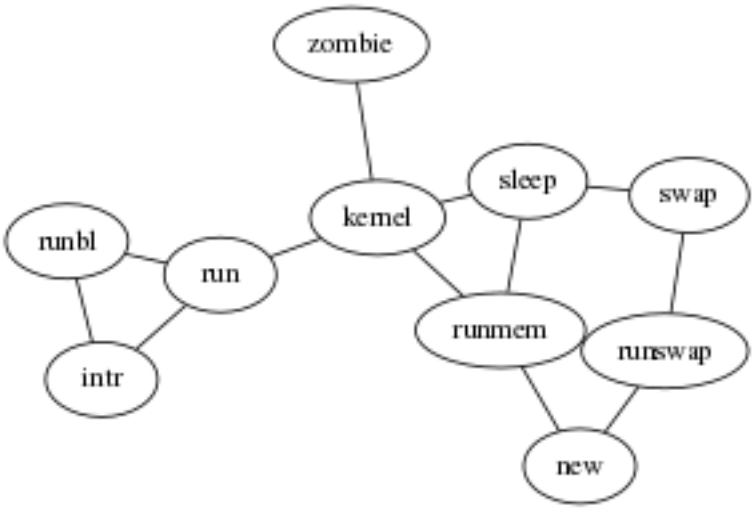

Figure 15: Created by neato

`twopi`

The same, but by `twopi`:

```{.graphviz prog="twopi" caption="Created by twopi" keep="True"}
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 16: Created by twopi

circo

Again, the same but by `circo`:

```{.graphviz prog="circo" caption="created by circo" keep="True"}

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;
}
```

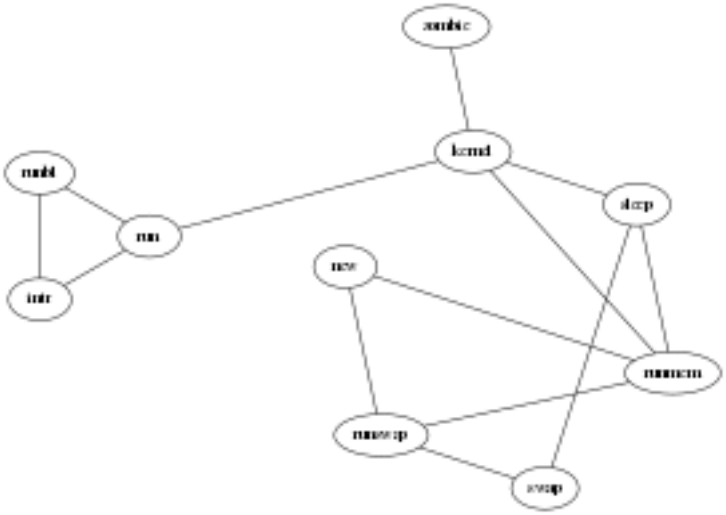

Figure 17: created by circo

BlockDiag

blockdiag site: `blockdiag` and its family generate diagram images from simple text files:

- Supports many types of diagrams.
  - block diagram (w/ blockdiag)
  - sequence diagram (w/ seqdiag)
  - activity diagram (w/ actdiag)
  - logical network diagram (w/ nwdiag)
- Generates beautiful diagram images from simple text format (similar to graphviz's DOT format)
- Layouts diagram elements automatically
- Embeds to many documentations; Sphinx, Trac, Redmine and some wikis

blockdiag

```{.blockdiag prog="blockdiag" keep="True" width="100%" caption="Created by Blockdiag"}
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 keep="True" 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;
}

```

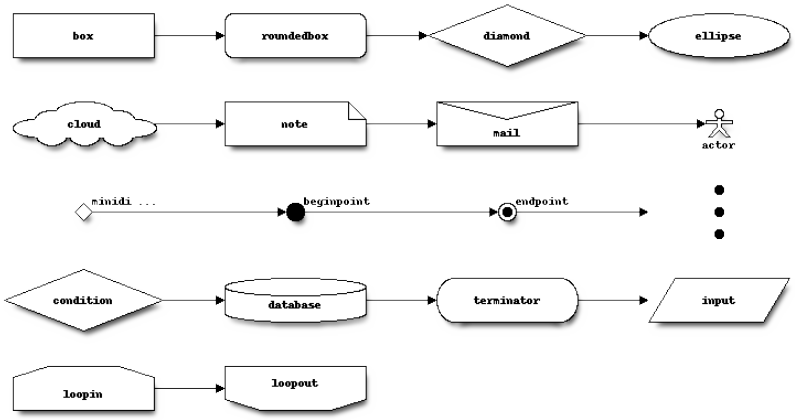

Figure 18: Created by Blockdiag

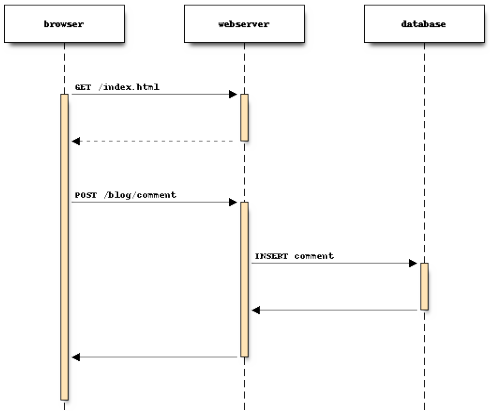

Figure 19: Created by seqdiag

nwdiag

```
```{.nwdiag keep="True" 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;
  }
}
```
```

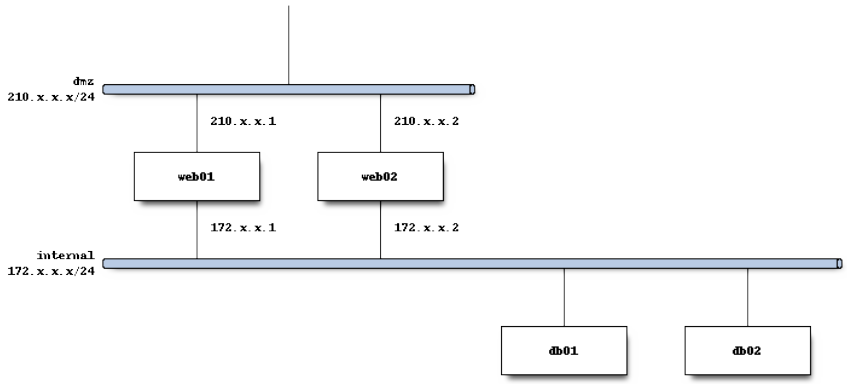

Figure 20: Created by nwdiag

actdiag

```
```{.actdiag keep="True" height="60%" caption="Created by actdiag"}
{
```
```

```
   A -> B -> C -> D;

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

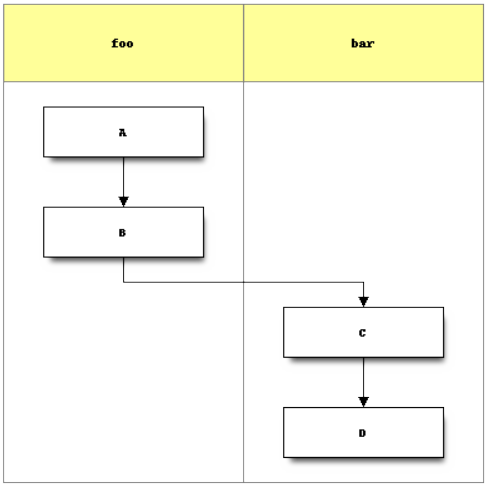

Figure 21: Created by actdiag

rackdiag

```{.rackdiag keep="True" 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
  }
}
```

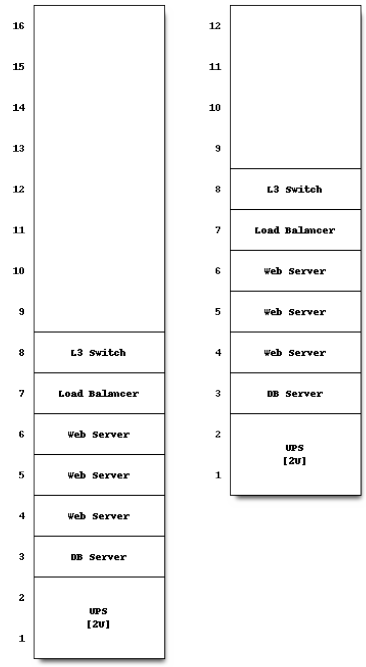

Figure 22: Created by rackdiag

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
  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]
}
```

mscgen

mscgen site: mscgen is a small program that parses Message Sequence Chart descriptions and produces PNG, SVG, EPS or server side image maps (ismaps) as the output. Message Sequence Charts (MSCs) are a way of representing entities and interactions over some time period and are often used in combination with SDL. MSCs are popular in Telecoms to specify how protocols operate although MSCs need not be complicated to create or use. Mscgen aims to provide a simple text language that is clear to create, edit and understand, which can also be transformed into common image formats for display or printing.

example w/ boxes

```
```{.mscgen keep="True" 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"];
}
```

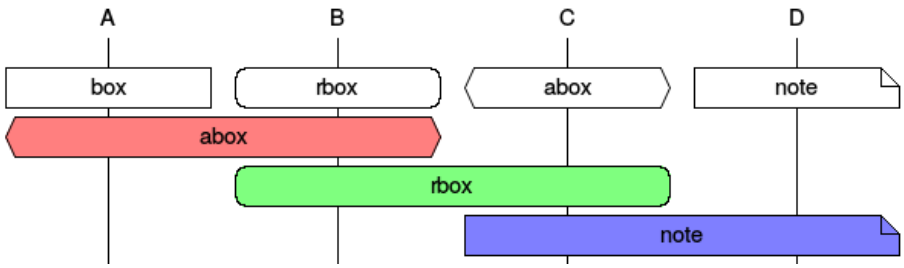

Figure 23: Created by mscgen

client-server interaction

```{.mscgen keep="True" caption="Created by mscgen"}
msc {
 hscale="1.3", arcgradient = "8";

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

 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"];
 |||;
}
```

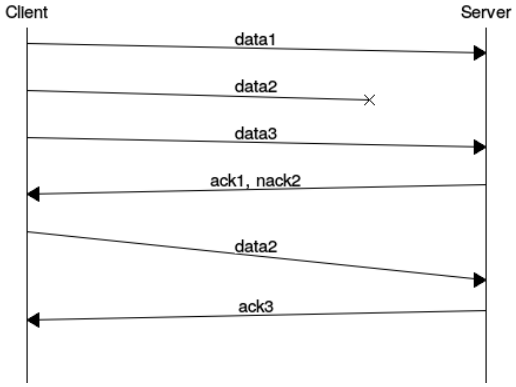

Figure 24: Created by mscgen

plantuml

plantuml site: `plantuml` is a component that allows to quickly write:

- Sequence diagram
- Usecase diagram
- Class diagram
- Activity diagram (here is the legacy syntax),
- Component diagram
- State diagram
- Object diagram
- Deployment diagram
- Timing diagram

sequence diagrams

```
```{.plantuml keep="True" 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
```

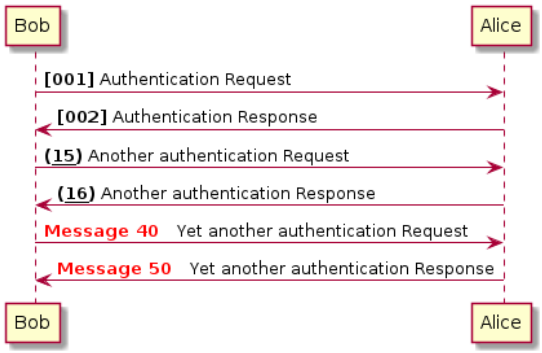

Figure 25: Created by plantuml

class diagrams

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

larger plantuml

```
```{.plantuml keep="True" caption="Created by plantuml"}
@startuml
```
```

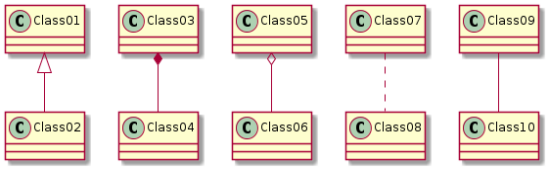

Figure 26: Created by plantuml

```
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
```

mermaid

sequence graph

```{.mermaid keep="True" width="70%" caption="Created by mermaid"}
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!
```

gantt diagram

```{.mermaid keep="True" 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
```

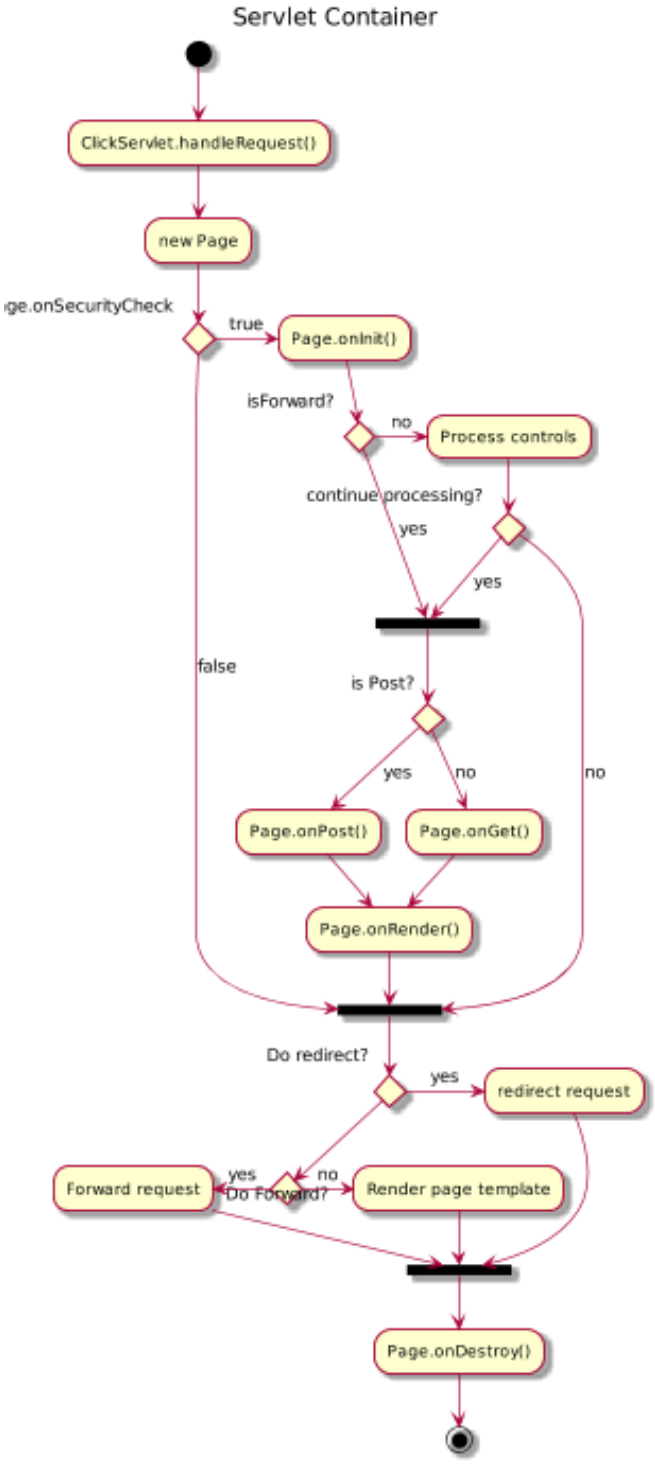

Figure 27: Created by plantuml

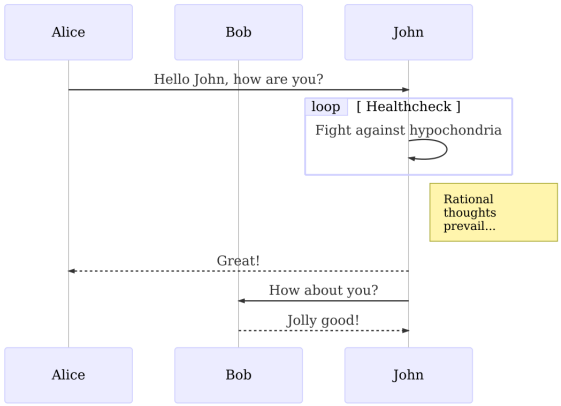

Figure 28: Created by mermaid

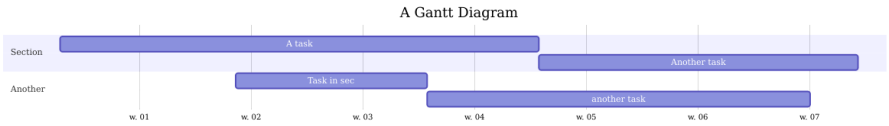

Figure 29: Created by mermaid

Ditaa

ditaa site: `ditaa` is a small command-line utility written in Java, that can convert diagrams drawn using ascii art ('drawings' that contain characters that resemble lines like | / -), into proper bitmap graphics. This is best illustrated by the following example – which also illustrates the benefits of using ditaa in comparison to other methods :)

ditaa interprets ascci art as a series of open and closed shapes, but it also uses special markup syntax to increase the possibilities of shapes and symbols that can be rendered.

ditaa is open source and free software (free as in free speech), since it is released under the GPL license.

Ditaa with options="-r"

```
```{.ditaa options="-r" keep="True" width="70%" caption="Created by Ditaa"}
+--------+   +-------+    +-------+
|        +---+ ditaa +--> |       |
|  Text  |   +-------+    |diagram|
|Document|   |!magic!|    |       |
|     {d}|   |       |    |       |
+---+----+   +-------+    +-------+
    :                         ^
    |       Lots of work      |
    +-------------------------+
```
```

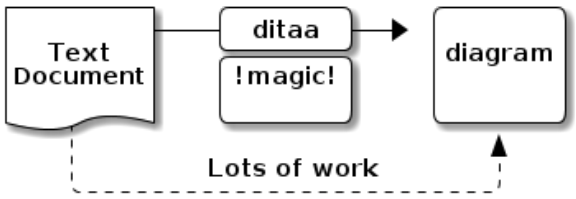

Figure 30: Created by Ditaa

Ditaa normal

```
```{.ditaa keep="True" 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|
                                                        *--------*
```

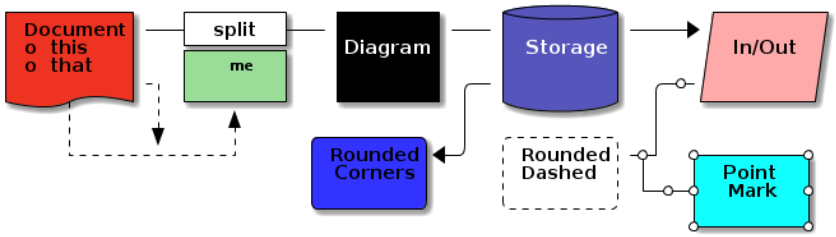

Figure 31: Created by Ditaa

ditaa reminder

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

Ditaa on protocol result

```
```{.ditaa keep="True"}
```
```

Figure 32: Created by Ditaa

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

```

Figlet

```
```{#FIGLET .figlet options="-f slant" keep="True" caption="Figlet"}
figlet
```
```

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

boxes

boxes Boxes is a command line program that draws a box around its input text. It can remove and repair those boxes, too. You can easily make your own box designs if you wish, but many designs are already provided.

design 'peek'

```
```{.boxes options="-d peek -a c -s 40x3" keep="true" caption="boxes"}
boxes
```
```

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

design 'ian_jones'

```
```{.boxes options="-d ian_jones -a c -s 40x6" keep="True" 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.
                            \ (    (   )
                             \_)    ) /
                                   (_/
```

Protocol

protocol github: 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.

A **TCP Header**:

```
```{.protocol keep="True" 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       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Acknowledgment Number                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Offset|  Res. |     Flags     |             Window            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Checksum           |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

and even custom layouts:

```{.protocol options="--no-numbers" keep="True" caption="protocol"}
Source:16,TTL:8,Reserved:40
```

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