Imagine

Imagine

```
Imagine
  A pandoc filter that wraps around a number of external command line utilities
  used to turn a fenced codeblock into graphics or ascii art.

  Commands include:

  actdiag, blockdiag, boxes, circo, ditaa, dot, fdp, figlet, gnugraph,
  graph, graphviz, imagine, mermaid, mscgen, neato, nwdiag, packetdiag,
  pic, pic2plot, plantuml, plot, protocol, 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
       - plotutils,  https://www.gnu.org/software/plotutils/
       - gnuplot,    http://www.gnuplot.info/
     % sudo pip install:
       - blockdiag,  http://blockdiag.com
       - phantomjs,  http://phantomjs.org/
     % git clone
       - https://github.com/luismartingarcia/protocol.git
     % npm install:
       - -g mermaid

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

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

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)" ];
}

```
```

`fdp`

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

Figure 1: FSM layout by dot

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

```

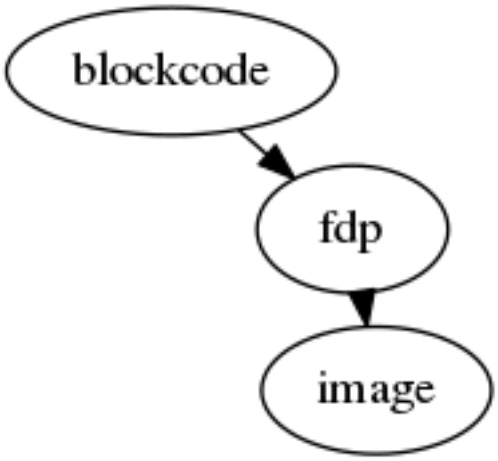


Figure 2: Created by fdp

`sfdp` *(fails)*

```
graph G {
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;
}
```
```

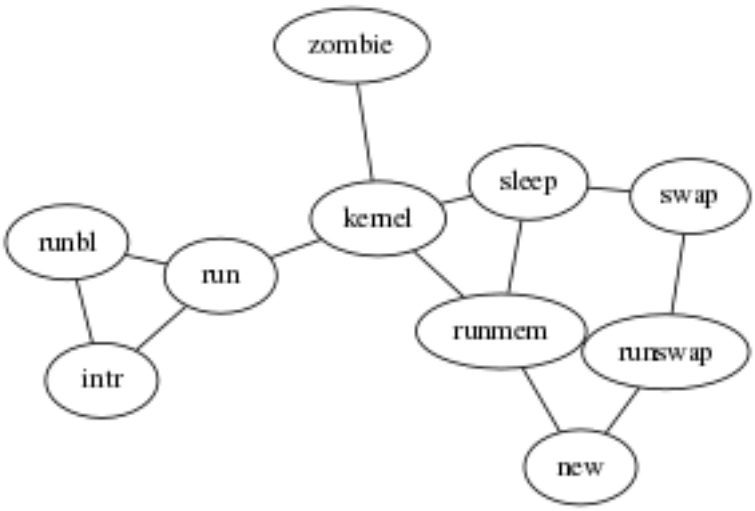


Figure 3: Created by neato

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 4: 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;
}
```
````

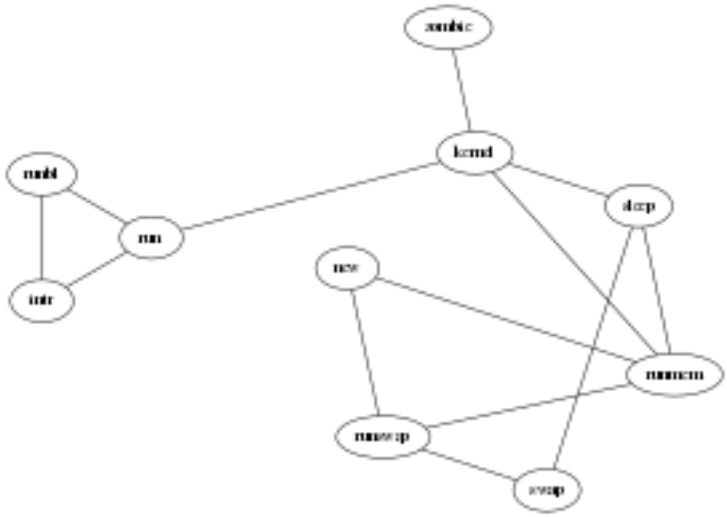


Figure 5: created by circo

BlockDiag

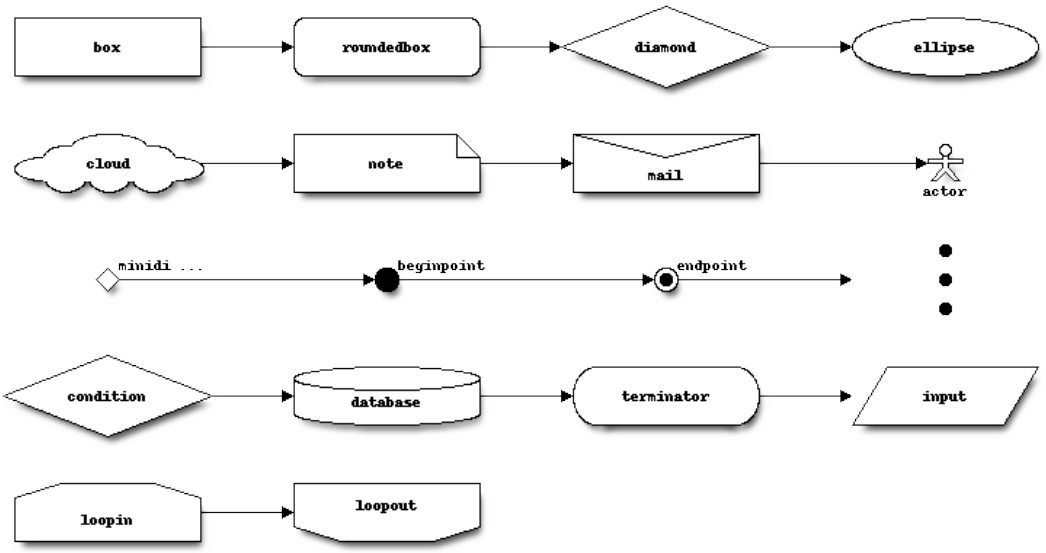
blockdiag site: `blockdiag` and its family generate diagram images from simple text files:

- 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%"}
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"];
```
```

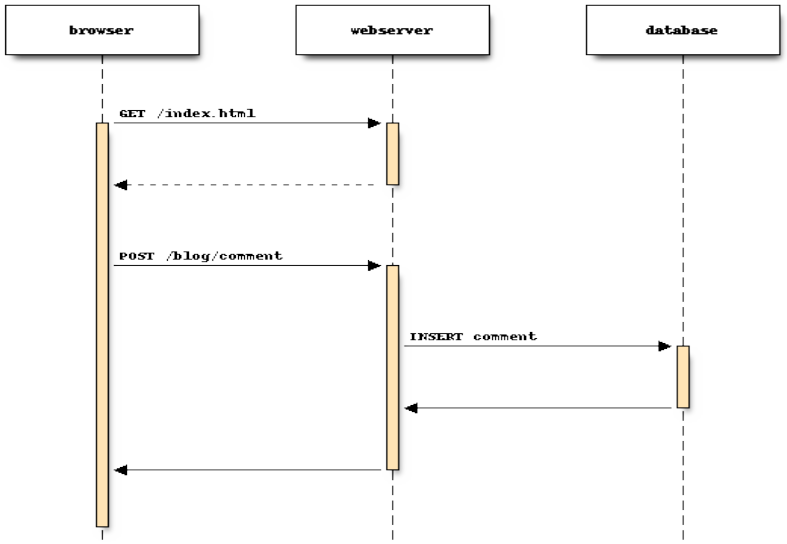
```
terminator [shape = "flowchart.terminator"];
condition -> database -> terminator -> input;
loopin -> loopout;
}
```



seqdiag

```{.seqdiag keep="True" width="80%" height="50%"}
{
browser -> webserver [label = "GET /index.html"];
browser <-- webserver;
browser -> webserver [label = "POST /blog/comment"];
webserver -> database [label = "INSERT comment"];
webserver <- database;
browser <- webserver;
}

```

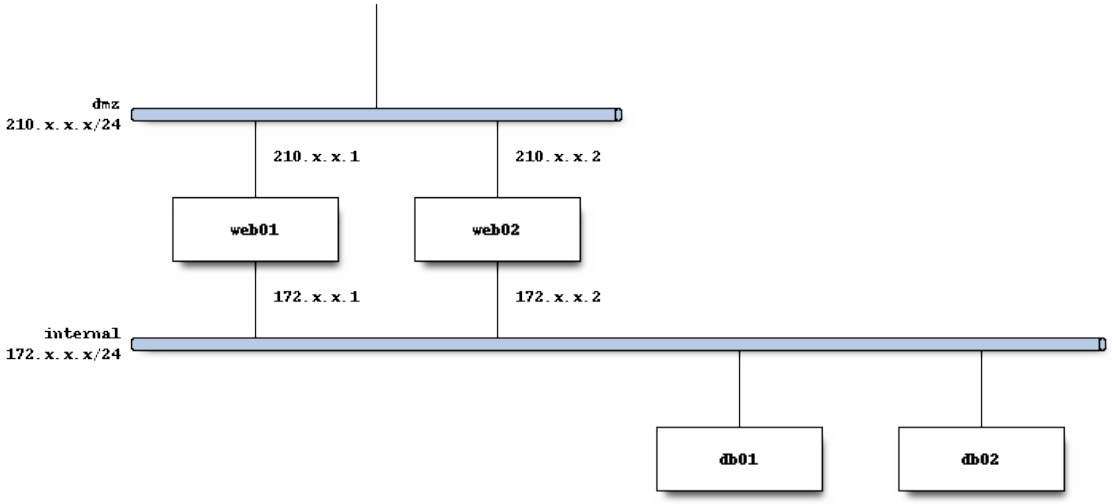


nwdiag

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

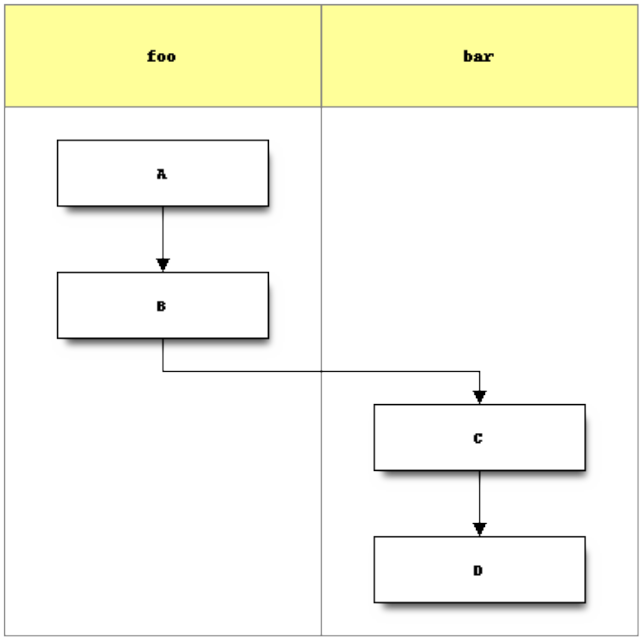
      web01 [address = "172.x.x.1"];
      web02 [address = "172.x.x.2"];
      db01;
      db02;
  }
}
```
```



actdiag

```{.actdiag keep="True" height="60%"}
{
   A -> B -> C -> D;

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



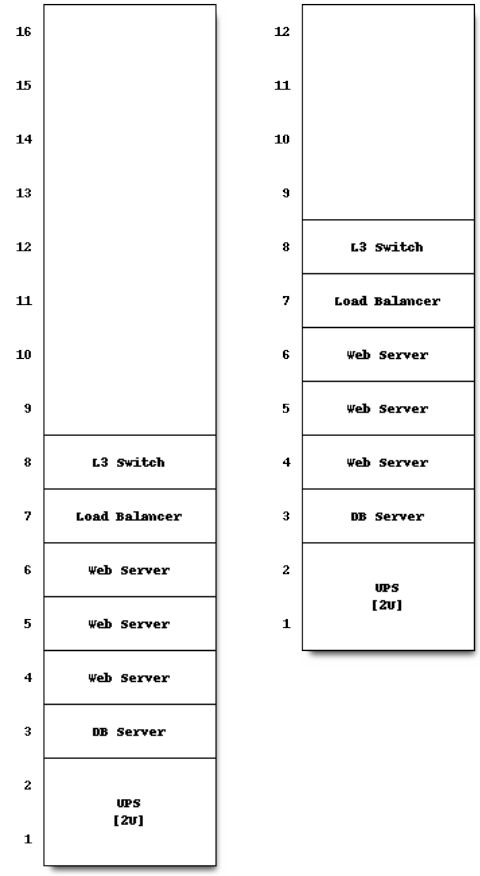
rackdiag

```
```{.rackdiag keep="True" height="80%"}
{
  // 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
  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

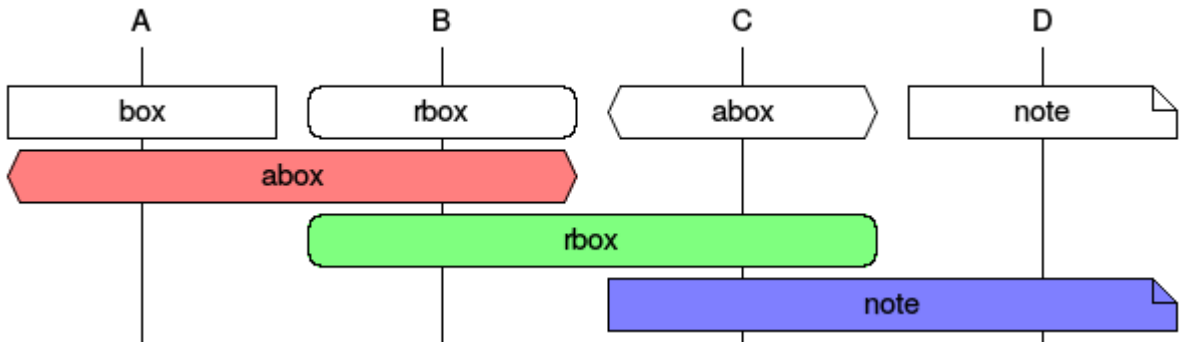
```
```{.mscgen keep="True"}
msc {

   # 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"];
}
```
```

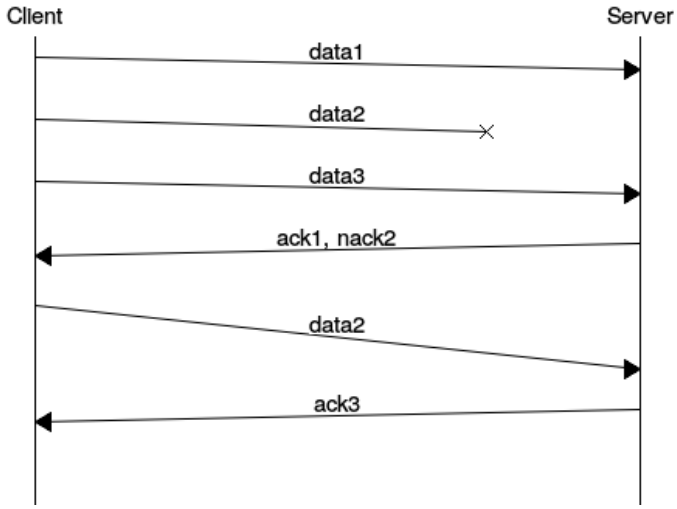


client-server interaction

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



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

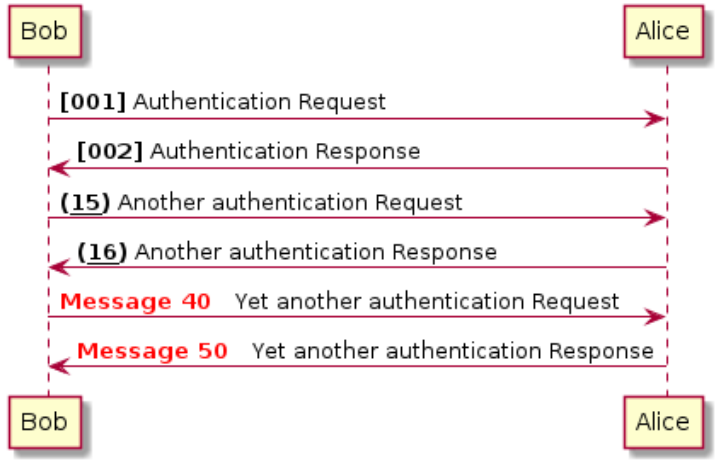
```
```{.plantuml keep="True" width="60%"}
@startuml
autonumber "<b>[000]"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

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
```
```

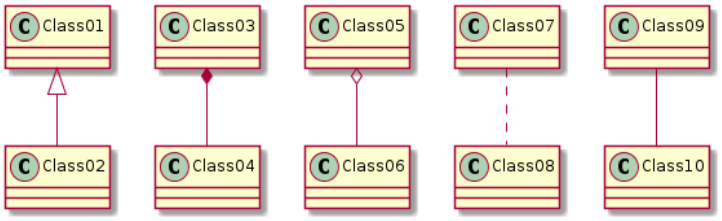
```
Bob <- Alice : Yet another authentication Response

@enduml
```



class diagrams

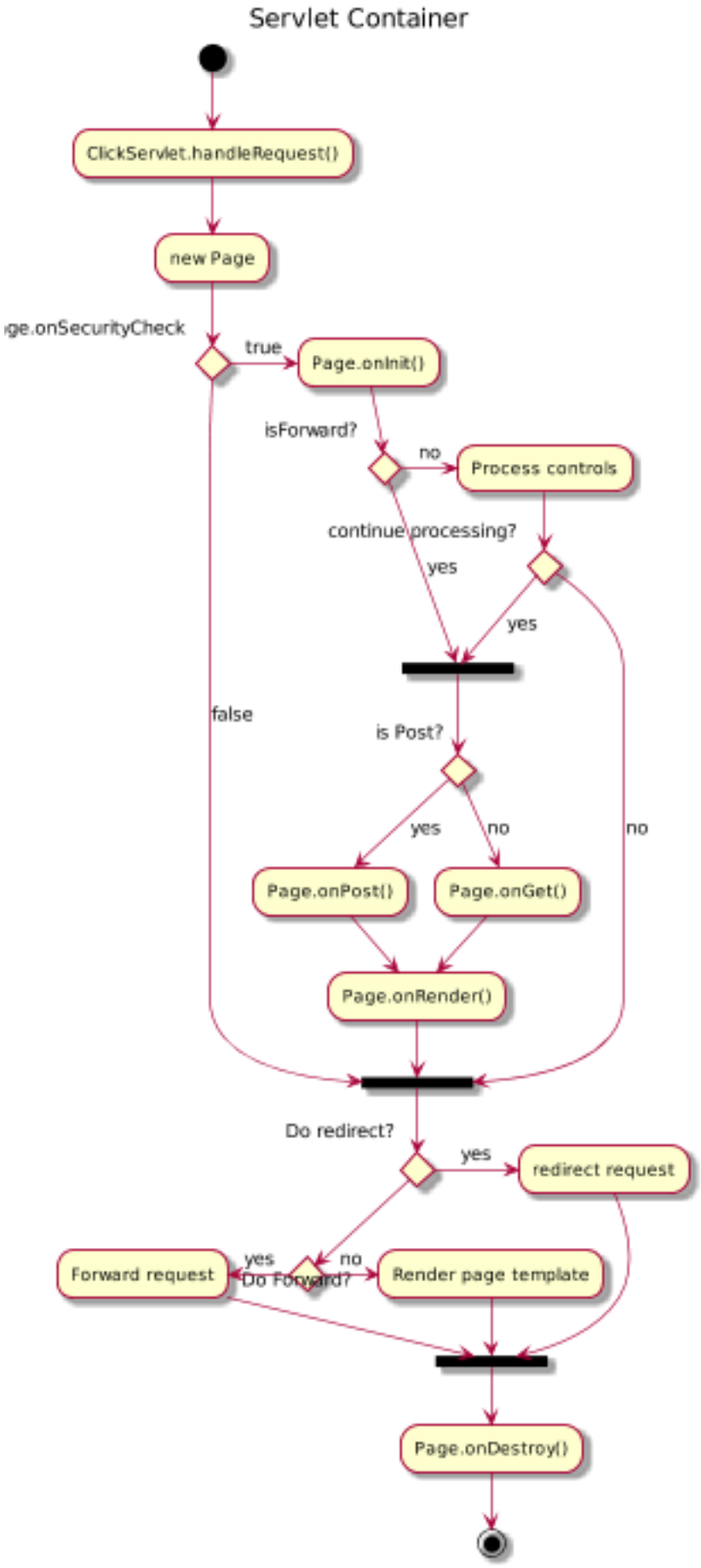
```{.plantuml keep="True" width="60%"}
@startuml
Class01 <|-- Class02
Class03 *-- Class04
Class05 o-- Class06
Class07 .. Class08
Class09 -- Class10
@enduml
```



larger plantuml

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

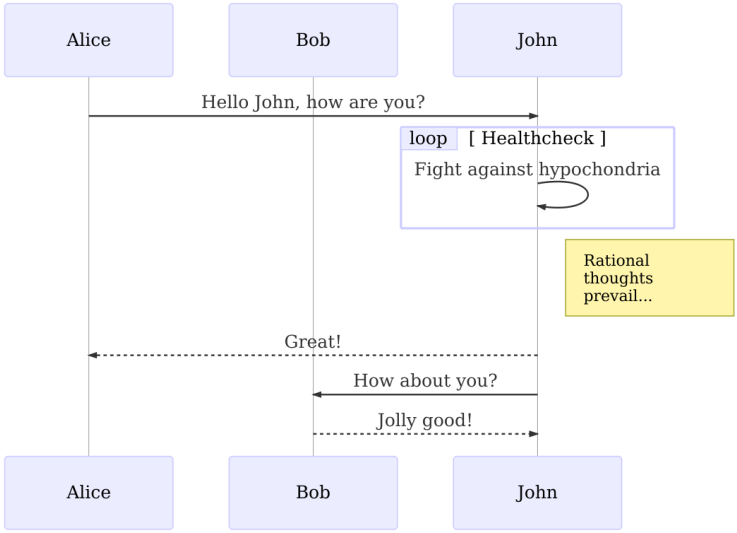
```
endif
else
-->[yes] ===RENDERING===
endif
if "is Post?" then
-->[yes] "Page.onPost()"
--> "Page.onRender()" as render
--> ===REDIRECT_CHECK===
else
-->[no] "Page.onGet()"
--> render
endif
else
-->[false] ===REDIRECT_CHECK===
endif
if "Do redirect?" then
->[yes] "redirect request"
--> ==BEFORE_DESTROY===
else
if "Do Forward?" then
-left->[yes] "Forward request"
--> ==BEFORE_DESTROY===
else
-right->[no] "Render page template"
--> ==BEFORE_DESTROY===
endif
endif
--> "Page.onDestroy()"
-->(*)
@enduml
```


Servlet Container
ClickServlet.handleRequest()
new Page
ge.onSecurityCheck
true
Page.onInit()
isForward?
no
Process controls
continue processing?
yes
yes
false
is Post?
yes
no
no
Page.onPost()
Page.onGet()
Page.onRender()
Do redirect?
yes
redirect request
Forward request
yes
Do Forward?
no
Render page template
Page.onDestroy()

mermaid

sequence graph

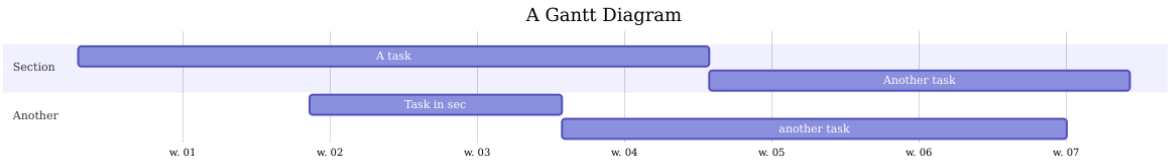
```
```{.mermaid keep="True" width="70%"}
sequenceDiagram
    participant Alice
    participant Bob
    Alice->>John: Hello John, how are you?
    loop Healthcheck
        John->>John: Fight against hypochondria
    end
    Note right of John: Rational thoughts<br/>prevail...
    John-->>Alice: Great!
    John->>Bob: How about you?
    Bob-->>John: Jolly good!
```
```



gantt diagram

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



Ditaa

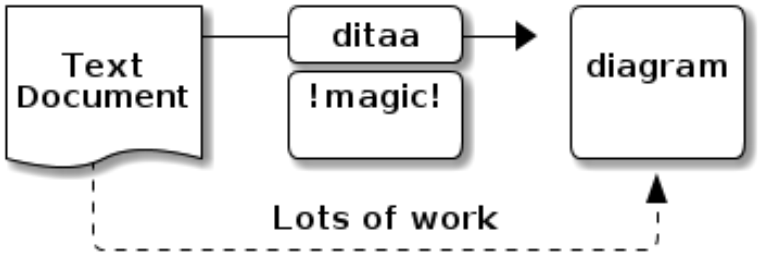
ditaa site: `ditaa` is a small command-line utility written in Java, that can convert diagrams drawn using ascii art ('drawings' that contain characters that resemble lines like | / -), into proper bitmap graphics. This is best illustrated by the following example – which also illustrates the benefits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

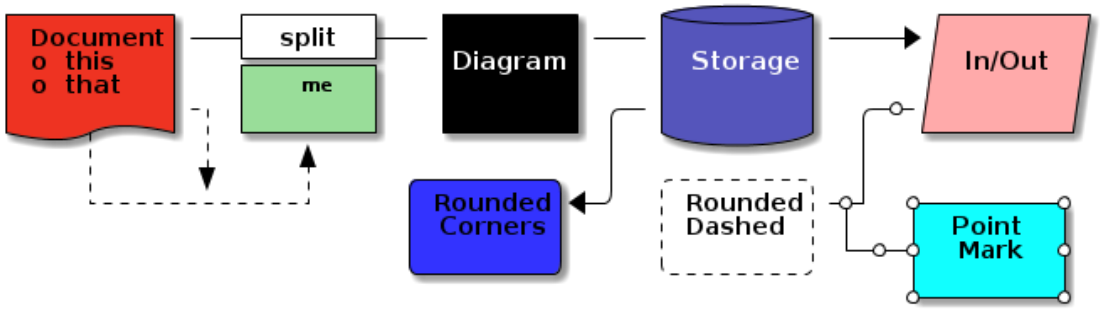
```
```{.ditaa options="-r" keep="True" width="70%"}
+--------+   +-------+    +-------+
|        +---+ ditaa +--> |       |
|  Text  |   +-------+    |diagram|
|Document|   |!magic!|    |       |
|     {d}|   |       |    |       |
+---+----+   +-------+    +-------+
    :                         ^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



Ditaa normal

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



ditaa reminder

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

Plotutils

plotutils site It includes:

- 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.

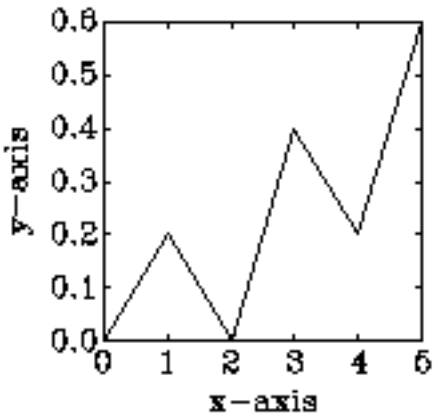
- GNU *plot*, which translates GNU Metafile format to any of the other formats.
- GNU *tek2plot*, for translating legacy Tektronix data to any of the above formats.
- GNU *pic2plot*, for translating the pic language (a scripting language for designing box-and-arrow diagrams) to any of the above formats. 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.

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"}
0.0  0.0
1.0  0.2
2.0  0.0
3.0  0.4
4.0  0.2
5.0  0.6
```



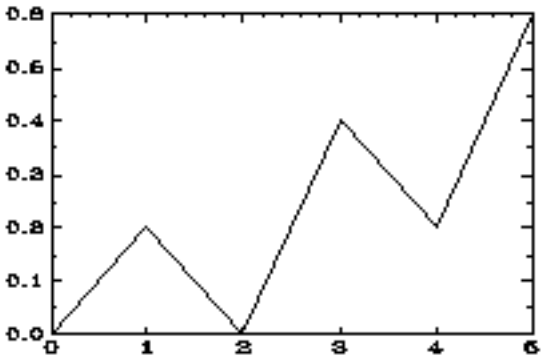
plot

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 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"}
input.meta
```

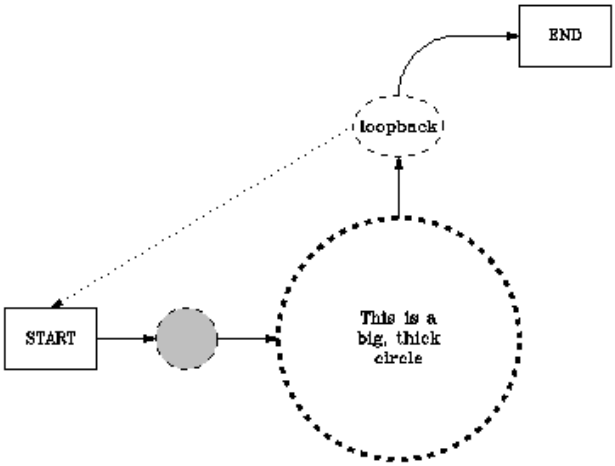


pic2plot

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%"}
.PS
box "START"; arrow; circle dashed filled; arrow
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
```



Text output

figlet

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

```
boxes
```

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

design 'ian_jones'

```
```{.boxes options="-d ian_jones -a c -s 40x6" keep="True"}
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"}
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"}
Source:16,TTL:8,Reserved:40
```

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