Li-Pro.Net Sphinx Primer

Version 0.0.5+RTDEXT

The LP/N Documentation Team

Oct 01, 2020

Li-Pro.Net
Jena, Germany
Email: info@li-pro.net

Table of Contents

Summary of Li-Pro.Net Sphinx Primer

Abstract This document is an effort of the Li–Pro.Net community with many references, best practices and examples, and aims to improve technical writing of any kind of documents and publications. In short words: How to write Li-Pro.Net documentation with Sphinx.

Involved Components

- *Sphinx*
- *Docutils*
- *reStructuredText*

Audience

- Project members / maintainer
- Hard- and software developer
- Integrators and testers
- Technical writer / editor

Status preliminary (*some mature, much in progress*)

Version 0.0.5

Release 0.0.5+RTDEXT

Date Oct 01, 2020

Authors The LP/N Documentation Team

- Stephan Linz <linz@li-pro.net>

Organization Li–Pro.Net

Contact Stephan Linz <linz@li-pro.net>

Address Jena, Germany

Legal Notice of Li-Pro.Net Sphinx Primer


```
Creative Commons Legal Code

Attribution-ShareAlike 3.0 Unported
```

See Listing 1.1, *License text of the Li-Pro.Net Sphinx Primer* (page 179), for the complete text that comes within this document.

Version history of Li-Pro.Net Sphinx Primer

Table 1: Li-Pro.Net Sphinx Primer Document Revisions

Version	Change	Passed	Date
0.0.5	extended content in chapter “Extensions”, template finetuning, extend local extension	Stephan Linz	2020-10-01
0.0.4	initial content in chapter “Concepts”	Stephan Linz	2020-09-14
0.0.3	initial content in chapter “Extensions”	Stephan Linz	2020-09-11
0.0.2	initial content in chapters “Cheat Sheet” and “Themes”	Stephan Linz	2020-09-08
0.0.1	base document skeleton	Stephan Linz	2020-09-08
0.0	preliminary, project created	Stephan Linz	2020-09-05

PREAMBLE

How to write Li-Pro.Net documentation with Sphinx.

Excerpts from the Sphinx Tutorial by Eric Holscher and Documentation Style Guide by Bareos GmbH & Co. KG and others. See [Has19] for an introduction to *Sphinx*.

This documentation is built using *Sphinx*, a static-site generator designed to create structured, semantic, and internally consistent documentation. Source documents are written in *reStructuredText*, a semantic, extensible markup syntax similar to Markdown. *reStructuredText* is a better tool than Markdown for documentation.

- reStructuredText Primer – introduction to reStructuredText
  - reStructuredText Quick Reference
  - reStructuredText 1-page cheat sheet
- Sphinx Markup – detailed guide to Sphinx's markup concepts and reStructuredText extensions
- Sphinx/Rest Memo – serve as quick reference for reStructuredText and Sphinx syntax

---

Note: *Sphinx* and *reStructuredText* can be very flexible. For the sake of consistency and maintainability, this how to guide is *highly opinionated* about how documentation source files are organized and marked up.

---

1 Concepts

Section author: Stephan Linz <linz@li-pro.net>

Let's document the project. A lot of these *reStructuredText* syntax examples are covered in the *Sphinx* reStructuredText Primer. The outline for this chapter has been taken from Documenting Python and adapted and extended for our needs.

1.1 Use of whitespace

All *reStructuredText* files **use an indentation of three (3) spaces**; no tabs are allowed. The maximum **line length is 80 characters** for normal text, but tables, deeply indented code samples and long links may extend beyond that. *Code example* bodies should use *normal four-(4)-space* indentation.

Make generous use of blank lines where applicable; they help group things together.

1.1.1 Indentation

Indentation is meaningful in *Sphinx* and *reStructuredText* text. Usually, indenting a section means that is "belongs to" the line it is indented under.

for example

```
.. figure:: path-to-image.*

   This is the caption of the figure. Notice that it is indented under
   the line defining the figure.
```

The rules for indentation are:

- Use **spaces, not tabs**.
- Generally, indent **three (3) spaces**.
- Code example, indent **four (4) spaces**, except *reStructuredText* examples.

The exception to the three (3) spaces rule is *Unordered (bullet) Lists* (page 31) and *Ordered (numbered) Lists* (page 32), where indentation follows the content of the list item.

unordered (bulleted) list

(2) spaces

```
* This is a list item.

  This is some additional content related to first item. Notice that
  it is indented to the same column as the first line of content.
  In this case, that's three (2) spaces.


.
.
.

* The N-th item in a list.
```

ordered (numbered) list

(4) spaces

```
1.  This is a list item.

    This is some additional content related to Item 1. Notice that
    it is indented to the same column as the first line of content.
    In this case, that's three (3) spaces.


.
.
.

10. The tenth item in a list.

    This related content will be indented four (4) spaces.
```

1.2 Sphinx naming

1.2.1 Roles

A role or "custom interpreted text role" is an inline piece of explicit markup, see *Inline Markup* (page 28) and *Explicit Markup* (page 35). It signifies that the enclosed text should be interpreted in a specific way. *Sphinx* uses this to provide semantic markup and cross-referencing of identifiers, as described in the appropriate section.

The general syntax is `:rolename:`content``. Like *Directives* (page 15), roles are extensible. Own roles can be created. They are used inside other text structures.

Docutils supports the following roles (incomplete list):

:emphasis:
emphasis – `:emphasis:`emphasis`` – equivalent of `*emphasis*`

:strong:
strong – `:strong:`strong`` – equivalent of `**strong**`

:literal:
literal – `:literal:`literal`` – equivalent of ``` ``literal`` ```

:code:
code – `:code:`code`` – equivalent of ``` ``code`` ```

:subscript:
subscript – `:subscript:`subscript`` – subscript text

The example The Fibonacci numbers (without inline role for *Mathematics* (page 62)).

```
.. |gE| unicode:: U+02267 .. GREATER-THAN OVER EQUAL TO

*f*\ :subscript:`n` = *f*\ :subscript:`n-1` + *f*\ :subscript:`n-2`
for *n* |gE| 3 with *f*\ :subscript:`1` = *f*\ :subscript:`2` = 1
```

Which gives $f_n = f_{n-1} + f_{n-2}$ for $n \geqq 3$ with $f_1 = f_2 = 1$

:superscript:
superscript – `:superscript:`superscript`` – superscript text

The example The elementary charge (without inline role for *Mathematics* (page 62)).

```
.. |sdot| unicode:: U+022C5 .. DOT OPERATOR

*e* = 1.602176634 |sdot| 10\ :superscript:`-19` C
```

Which gives $e = 1.602176634 \cdot 10^{-19}$ C

:math:
math – `:math:`mathematic equations`` – for *Mathematics* (page 62) equations

:pep-reference:

pep-reference – `:pep-reference:`pep-reference`` – equivalent to `:pep:`pep reference number`` – for *External References* (page 71) into the PEP (Python Enhancement Proposal) index

:rfc-reference:

rfc-reference – `:rfc-reference:`rfc-reference`` – equivalent to `:rfc:`rfc reference number`` – for *External References* (page 71) into the RFC (Request for Comments) index

:title-reference:

title-reference – `:title-reference:`title-reference`` – for titles of books, periodicals, and other materials

See also:

- Refer to Roles for roles provided by *Docutils*.
- Refer to Roles for roles added by *Sphinx*.

1.2.2 Directives

A directive is a generic block of *Explicit Markup* (page 35). Besides roles, it is one of the extension mechanisms of *reStructuredText*, and *Sphinx* makes heavy use of it.

Basically, a directive consists of a **name**, **arguments**, **options** and **content**. Keep this terminology in mind, it is used in section *Explicit Markup* (page 35) describing custom directives. Looking at this example, that allows marking a block of content with special meaning.

basic directive syntax looks like this

the example

```
.. directive:: arg1 arg2 ...
   :option1: value
   :option2: value
   :option5: value
   ...

   Multiline content of the directive,
   ...
```

This line is no longer part of the block controlled by the directive.

directive That is the *directive name*. It is given two arguments here.

arg1, arg2, ... *Arguments*. The last argument can contain spaces (depending on the directive implementation).

:option0:, :option1:, ... :option9: *Options* are optional. As you can see, options are given in the lines immediately following the arguments and indicated by the colons.

Multiline content of the directive, The *directive content* follows after a blank line and is indented relative to the directive start.

Directives are supplied not only by *Docutils*, but *Sphinx* and custom extensions can add their own. Directives are written as a block.

Docutils supports the following directives (incomplete list):

- *Admonitions* (page 64): attention, caution, danger, error, hint, important, note, tip, warning and the generic admonition. (Most themes style only “note” and “warning” specially.)
- *Images and Figures* (page 41):
  - image
  - figure (an image with caption and optional legend)

- Additional body elements:
  - contents (a local, i.e. for the current file only, table of contents)
  - section numbering (automatically)
  - container (a container with a custom class, useful to generate an outer `<div>` in *HTML*)
  - rubric (a heading without relation to the document sectioning)
  - topic, sidebar (special highlighted body elements)
  - parsed-literal (literal block that supports inline markup)
  - epigraph (a block quote with optional attribution line)
  - highlights, pull-quote (block quotes with their own class attribute)
  - compound (a compound paragraph)
- Special *Tables* (page 48):
  - table (a table with title)
  - csv-table (a table generated from comma-separated values)
  - list-table (a table generated from a list of lists)
- Special directives and *Include a Shared File* (page 38):
  - raw (include raw target-format markup)
  - include (include *reStructuredText* from another file) – in *Sphinx*, when given an absolute include file path, this directive takes it as relative to the source directory
  - class (assign a class attribute to the next element)[1]
- *HTML* specifics:
  - meta (generation of *HTML* `<meta>` tags, see also HTML Metadata below)
  - title (override document title)
- Influencing markup:
  - default-role (set a new default role)
  - role (create a new role)

  Since these are only per-file, better use *Sphinx*'s facilities for setting the `default_role`.
- References and *Substitutions* (page 39):
  - target footnotes (for each external URL target)
  - replacement text (for a substitution)
  - unicode characters (used in substitution)

[1] When the default domain contains a **class** directive, this directive will be shadowed. Therefore, *Sphinx* re-exports it as **rst-class**.

> **Warning:** Do *not* use the directives sectnum, header and footer.

See also:

- Refer to Directives for directives provided by *Docutils*.
- Refer to Directives for directives added by *Sphinx*.

1.2.3 Domains

A domain is a collection of *explicit* (page 35) and *inline* (page 28) markup (*reStructuredText Directives* (page 15) and *Roles* (page 13)) to describe and link to objects belonging together, e.g. elements of a programming language. Directive and role names in a domain have names like `domain:name`, e.g. `.. c:function:: int main(int argc, char **argv, char **env)` or `:c:func:`main``.

An object is the basic building block of *Sphinx* documentation. Every "object directive" (e.g. function or object) creates such a block; and most objects can be cross-referenced to.

The Standard Domain collects all markup that does not warrant a domain of its own. Its directives and roles are not prefixed with a domain name.

There is a set of directives allowing documenting command-line programs:

Table 1.1: Sphinx directives for command-line programs

short description	directive (target)	role (reference)
Following document options for the program.	`.. program:: name`	
Describes a command line argument or switch.	`.. option:: name args, ...`	`:option:`name arg``
Describes an environment variable.	`.. envvar:: name`	`:envvar:`name``

There is also a very generic object description directive, which is not tied to any domain. This directive produces the same formatting as the specific ones provided by domains, but does not create index entries or cross-referencing targets:

Table 1.2: Sphinx directives for unspecific objects without referencing

short description	directive (target)	role (reference)
Describes an unspecific element.	`.. describe:: text`	
Describes an unspecific object.	`.. object:: text`	

Originally, *Sphinx* was conceived for a single project, the documentation of the *Python* language. Shortly afterwards, it was made available for everyone as a documentation tool, but the documentation of *Python* modules remained deeply built in – the most fundamental directives, like function,

were designed for *Python* objects.

Since *Sphinx* has become somewhat popular, interest developed in using it for many different purposes: *C*/*C++* projects, *JavaScript*, or even *reStructuredText* markup (like in this documentation). The following specific domains are provided by *Sphinx* (without additional extensions):

- The C Domain (name **c**)
- The C++ Domain (name **cpp**)
- The *JavaScript* Domain (name **js**)
- The Math Domain (name **math**)
- The *Python* Domain (name **py**)
- The *reStructuredText* Domain (name **rst**)

See also:

- Refer to Domains for domains provided by *Sphinx*.

1.3 Parts, Chapters, Titles, Sections

Every *Sphinx* document has multiple level of headings. Section headers are created by underlining (and optionally overlining) the section title with a punctuation character, at least as long as the text.

Normally, there are no heading levels assigned to certain characters as the structure is determined from the succession of headings. However, for this documentation, here is a suggested convention as covered in the *Sphinx* reStructuredText Primer to use them in this order:

- # for title – with overline, for parts
- * for subtitle – with overline, for chapters
- =, for sections
- -, for subsections
- ^, for subsubsections
- =, for paragraphs

They give structure to the document, which is used in navigation and in the display in all output formats. The part section header is not used at all. All regular documents starts with a title heading underlined by #. Therefore the specific names part, chapter, section,... might not match the actual context. Generally we speak about “sections” (or “section headings” or “section markers”).

Note: With *reStructuredText*, there is no leaving out a section level. If you write a chapter it is not possible to continue with a paragraph. Instead the next section must be of the type title.

If you try to do it overwise (chapter 1 * with overline → paragraph ”), the “paragraph” is treated as a “title”. And if you continue by another chapter in the same file (chapter 2 * with overline → title #), **`sphinx-build`** got confused and at least produces a warning (*Title level inconsistent*) and possibly renders the result incorrectly.

---

the convention

```
##################################
Part -- Number Signs above and below
##################################

with overline, for parts

**********************************
Chapter -- Asterisks above and below
**********************************

with overline, for chapters

Title -- Number Signs
#####################

Suptitle -- Asterisks
*********************

Section -- Equal Signs
======================

Subsection -- Hyphens
---------------------

Subsubsection -- Circumflex
^^^^^^^^^^^^^^^^^^^^^^^^^^^

Paragraph -- Double Quotes
"""""""""""""""""""""""""""""""
```

1.4 Table of Contents Tree

Now would be a good time to introduce the `.. toctree::`. One of the main concepts in *Sphinx* is that it allows multiple pages to be combined into a cohesive hierarchy. Since *reStructuredText* does not have facilities to interconnect several documents, or split documents into multiple output files, *Sphinx* uses a custom directive to add relations between the single files the documentation is made of, as well as tables of contents.

The `.. toctree::` directive is the central element and a fundamental part of this structure. Consider this example:

`.. toctree::`

For more details, see `toctree` directive.

The example

```
.. toctree::
   :maxdepth: 2

   install
   support
   (many more files listed here)
```

Which gives

```
index
├── install
├── support
├── (many more files here)
    ├── (many more sub-files here)
```

The above directive example will output a TOC (Table of Contents) in the page where it occurs, using the individual TOCs (including "sub-TOC trees") of the files given in the directive body. The `:maxdepth: 2` argument tells *Sphinx* to include 2 levels of headers in it's output. It will output the 2 top-level headers of the pages listed; by default, all levels are included. This also tells *Sphinx* that the other pages are sub-pages of the current page, creating a "tree" structure of the pages.

This accomplishes two things:

- Tables of contents from all those files are inserted, with a maximum depth of argument `:maxdepth:`, that means one nested heading. `.. toctree::` directives in those files are also taken into account.
- *Sphinx* knows that the relative order of the files `install`, `support` and so forth, and it knows that they are children of the shown file, the library index. From this information it generates "next chapter", "previous chapter" and "parent chapter" links.

In the end, all files included in the build process must occur in (only) one `.. toctree::` directive; *Sphinx* will emit a warning if it finds a file that is not included, because that means that this file will not

be reachable through standard navigation.

The special file `index.rst` at the root of the source directory is the "root" of the TOC tree hierarchy; from it the "Contents" page is generated.

Note: The TOC Tree is also used for generating the navigation elements inside *Sphinx*. It is quite important, and one of the most powerful concepts in *Sphinx*.

1.4.1 Sidebar navigation menu

The `index.rst` file serves as a front-page to the documentation and contains the main tables of content, defined using `.. toctree::` directives. These `.. toctree::` directives control the sidebar navigation menu. To add a new document to a table of content, add the file name (without the `.rst` extension) to the relevant list of file names in `index.rst` or any other (but only one) "sub-TOC trees".

1.4.2 Secondary sub-TOC trees

Collections of documents are mostly given their own table of content on an individual page (see, for example: *Appendix* (page 179) and *Glossary* (page 191)). In these cases, the page containing the `.. toctree::` serves as a sort of intro page for the collection. That intro must, itself, be included in the *Sidebar navigation menu* (page 23). The contents of a `.. toctree::` appear as section links in another `.. toctree::` it is included in. That is, if a `.. toctree::` in `index.rst` lists `.. glossary::`, and `glossary.rst` has a `.. toctree::`, then the contents of that second `.. toctree::` will appear in the *Sidebar navigation menu* (page 23), as sub-items to *Glossary* (page 191).

Indeed, this is precisely the case in this Li-Pro.Net Sphinx Primer document currently.

1.4.3 How this document uses main and secondary TOC

- Major topics get a `.. toctree::` in `index.rst`

  Major topics include things like:

  - Each major parts (*Extensions* (page 81), *Themes* (page 171),...)
  - Large, general categories like Releases, Contributing, or Building

  Major topic tables of content include both sub-collection intro pages and also individual pages that don't fit into a sub-collection.

  The `:caption:` attribute of the `.. toctree::` directive may but not must defines the section label in the *Sidebar navigation menu* (page 23).

- Within a large topic, documents are grouped into collections of related pages, defined by a `.. toctree::` on a topic intro page.

Intro pages (pages that contain secondary `.. toctree::` directives) may include additional content, introducing the collection or providing contextual way-finding. However, this is not always necessary or desirable. Use your judgment, and avoid stating things just for the sake of having some text. ("Here are the pages in this collection.")

We also (very occasionally) include `.. toctree::` directives in sub-collection pages, such as:

- *BibTeX Citations* (page 84),
- *Spelling Checker* (page 83),
- ...
- *Read the Docs Sphinx Theme* (page 173),
- ...

Tip: If it not obvious where a new document should appear in the navigation, the best practice is to simply ask about it in the GitHub issue driving the new page.

Note: For way-finding purposes, we sometimes create an *Unordered (bullet) Lists* (page 31) of page links rather than a `.. toctree::` directive (for example, see `index.rst`). We do this when using a `.. toctree::` would create redundant links in the *Sidebar navigation menu* (page 23).

1.5 Paragraphs

The paragraph is the most basic block in a *reStructuredText* document. Paragraphs are simply chunks of text separated by one or more blank lines. As in *Python*, indentation is significant in *reStructuredText*, so all lines of the same paragraph must be left-aligned to the same level of indentation. General rules can be looked up under *Use of whitespace* (page 11).

the example

```
Paragraphs are separated by blank lines. Line breaks in the source␣
↪code do not create line breaks in the output.

This means that you *could*, in theory,
include a lot of arbitrary line breaks
in your source document files.
These line breaks would not appear in the output.
Some people like to do this because they have been trained
to not exceed 80 column lines, and they like
to write :file:`.txt` files this way.
Please do not do this.

There is **no reason** to put a limit on line length in source files␣
↪for documentation, since this is prose and not code.
Therefore, please do not put arbitrary line breaks in your files.
```

which gives Paragraphs are separated by blank lines. Line breaks in the source code do not create line breaks in the output.

This means that you *could*, in theory, include a lot of arbitrary line breaks in your source document files. These line breaks would not appear in the output. Some people like to do this because they have been trained to not exceed 80 column lines, and they like to write `.txt` files this way. Please do not do this.

There is **no reason** to put a limit on line length in source files for documentation, since this is prose and not code. Therefore, please do not put arbitrary line breaks in your files.

1.5.1 Quotes (block quotation) Element

Block quoted paragraphs are quoted by just indenting them more than the surrounding paragraphs.

the example

```
This line is not a block quote. Block quotes are indented,
and otherwise unadorned.

```

(continues on next page)

(continued from previous page)

```
   This is a block quote.

   --Adam Michael Wood - `Technical Content Writer`_

.. _`Technical Content Writer`:
   http://adammichaelwood.com/portfolio/
```

which gives This line is not a block quote. Block quotes are indented, and otherwise unadorned.

> This is a block quote.
>
> —Adam Michael Wood - Technical Content Writer

.. pull-quote::

Pull-quoted paragraphs are similar to block quotes but are directives for small selection of text to "pull out and quote", typically in a larger typeface.

The example

```
This line is not a pull quote.
Pull quotes are directive content.

.. pull-quote::

   This is a pull quote.

   --Adam Michael Wood - `Technical Content Writer`_

.. _`Technical Content Writer`:
   http://adammichaelwood.com/portfolio/
```

Which gives This line is not a pull quote. Pull quotes are directive content.

> This is a pull quote.
>
> —Adam Michael Wood - Technical Content Writer

1.5.2 Line Blocks

Line blocks are useful for addresses, verse, and adornment-free lists. They are quoted by just a | pipe sign in front of each single line.

the example

```
| Each new line begins with a
| vertical bar (``|``).
```

(continues on next page)

(continued from previous page)

```
|     Line breaks and initial indents
|     are preserved.
```

which gives

Each new line begins with a
vertical bar (“|”).
    Line breaks and initial indents
    are preserved.

1.5.3 Doctest Blocks

Doctest blocks are interactive *Python* sessions cut-and-pasted into docstrings. They do not require the *literal blocks* (page 57) syntax. The doctest block must end with a blank line and should not end with an unused prompt, see Doctest blocks in *Sphinx* for more informations.

the example

```
>>> print('this is a Doctest block')
this is a Doctest block

>>> print('Python-specific usage examples; begun with ”>>>”')
Python-specific usage examples; begun with ”>>>”
>>> print('(cut and pasted from interactive Python sessions)')
(cut and pasted from interactive Python sessions)

>>> a = [51,27,13,56]
>>> b = dict(enumerate(a))
>>> print(b)
{0: 51, 1: 27, 2: 13, 3: 56}
```

which gives

```
>>> print('this is a Doctest block')
this is a Doctest block
```

```
>>> print('Python-specific usage examples; begun with ”>>>”')
Python-specific usage examples; begun with ”>>>”
>>> print('(cut and pasted from interactive Python sessions)')
(cut and pasted from interactive Python sessions)
```

```
>>> a = [51,27,13,56]
>>> b = dict(enumerate(a))
>>> print(b)
{0: 51, 1: 27, 2: 13, 3: 56}
```

1.6 Inline Markup

If you want to make sure that text is shown in monospaced fonts for code examples or concepts, use double backticks around it. It looks `like this` on output.

the example

```
You can use **backticks** for showing ``highlighted`` code.
```

which gives You can use **backticks** for showing `highlighted` code.

Refer to Inline markup added by *Sphinx*.

All the standard *reStructuredText* inline markups are quite simple, use:

- one asterisk: `*text*` for emphasis (*italics*),
- two asterisks: `**text**` for strong emphasis (**boldface**), and
- backquotes: ``` ``text`` ``` for code samples as shown above (`literal`).

If asterisks or backquotes appear in running text and could be confused with inline markup delimiters, they have to be escaped with a backslash or encapsulated by *Roles* (page 13):

one escaped asterisk

the example

```
*italics \*with\* asterisk*,
**boldface \*with\* asterisk**
```

which gives *italics *with* asterisk*, **boldface *with* asterisk**

two escaped asterisks

the example

```
*italics \*\*with\*\* asterisks*,
**boldface \*\*with\*\* asterisks**
```

which gives *italics **with** asterisks*, **boldface **with** asterisks**,

two escaped backquotes

the example

```
*italics \`\`with\`\` backquotes*,
**boldface \`\`with\`\` backquotes**
```

which gives *italics “with“ backquotes*, **boldface “with“ backquotes**

escaped backquote and asterisks

the example

```
:literal:`literal \`\`with\`\` backquotes **and** asterisks`
```

which gives `literal ``with`` backquotes **and** asterisks`

Be aware of some restrictions of this markup:

- it may not be nested (see nested inline markup in *Docutils* To Do List),
- content may not start or end with whitespace: `* text*` is wrong,
- it must be separated from surrounding text by non-word characters. Use a backslash escaped space to work around that: `thisis\ **one**\ word` (thisis**one**word).

1.7 File, Directory, Path

File and directories (or generally paths) are formated by `:file:` inline markup. Backslashes (Windows paths) \ have to written as \\. The name of an executable program should be documented by `:program:` inline markup. This may differ from the file name for the executable for some platforms. In particular, the `.exe` (or other) extension should be omitted for Windows programs. For OS-level command use `:command:` inline markup.

`:file:`

For more details, see `file` role; about the `program` role in *Semantic Descriptions and References* (page 73), and about the `command` role in *Writing about User Interface* (page 75).

The example

```
| :file:`/bin/bash` or :file:`bash` -- but better is :command:`bash`
| :file:`/usr/local/bin/myapp` -- but better is :program:`myapp`

| :file:`filename.txt`
| :file:`/path/filename.txt`
| :file:`/path/subdir/` (ends with a ``/``)

| :file:`..\\MyApp\\core.conf`
| :file:`C:\\ProgramData\\MyApp\\core.conf`
| :file:`C:\\ProgramData\\MyApp\\` (ends with a ``\``)

| :file:`/usr/share/man/man{N}` (ends with a variable mark, *N* = 1.
↪.9)
```

Which gives

`/bin/bash` or `bash` – but better is **`bash`**

`/usr/local/bin/myapp` – but better is **`myapp`**

`filename.txt`

`/path/filename.txt`

`/path/subdir/` (ends with a `/`)

`..\MyApp\core.conf`

`C:\ProgramData\MyApp\core.conf`

`C:\ProgramData\MyApp\` (ends with a `\`)

`/usr/share/man/man`*N* (ends with a variable mark, *N* = 1..9)

1.8 Lists, Definition Lists

List markup is natural: just place an asterisk or hyphen at the start of a paragraph and indent properly. The same goes for numbered list (number or letter with tailed dot); they can also be automatically numbered using a # sign.

Nested lists are possible, but be aware that they must be separated from the parent list items by blank lines.

1.8.1 Unordered (bullet) Lists

Bullet lists contains list item elements which are uniformly marked with bullets. Bullets are typically simple dingbats (symbols) such as circles and squares.

bulleted lists ( )

the example

```
* This is a bulleted list ...
* ... use asterisks.
* It has fife items, the third
  item uses two lines.
* Are unindented at the first level.
* Must have a blank line before and after.

  - This is a bulleted list ...
  - ... use hyphens.
  - Are indented at the first level to stand out from the
    previous paragraph.

    - the blank line requirement means that nested list items
      will have a blank line before and after as well

    - you may *optionally* put a blank line *between* list items
```

which gives

- This is a bulleted list ...
- ... use asterisks.
- It has fife items, the third item uses two lines.
- Are unindented at the first level.
- Must have a blank line before and after.
  - This is a bulleted list ...

- ... use hyphens.
- Are indented at the first level to stand out from the previous paragraph.
  * the blank line requirement means that nested list items will have a blank line before and after as well
  * you may *optionally* put a blank line *between* list items

1.8.2 Ordered (numbered) Lists

Enumerated lists (a.k.a. "ordered" lists) are similar to bullet lists, but use enumerators instead of bullets. An enumerator consists of an enumeration sequence member and formatting, followed by whitespace. Different enumeration sequences are possible, e.g. Arabic or Roman numerals or alphabet characters.

numbered lists ( )

the example

```
1. This is a numbered list.
2. Start each line with a number and period.
3. Can begin on any number.

8. Must have a blank line before and after.
9. Can have nested sub-lists.

   a. nested lists are numbered separately
   #. nested lists need a blank line before and after

#. Can have automatic number with the ``#`` character.
```

which gives

1. This is a numbered list.
2. Start each line with a number and period.
3. Can begin on any number.

8. Must have a blank line before and after.
9. Can have nested sub-lists.
   a. nested lists are numbered separately
   b. nested lists need a blank line before and after
10. Can have automatic number with the # character.

1.8.3 Definition (description) Lists

Definition Lists contains a list of terms and their definitions. Each list item element contains a term, optional classifiers, and a definition.

definition list (<dl>)

the example

```
Definition list
   a list with several term-definition pairs in the form

   .. parsed-literal::

      **TERM**
         DEFINITION (*description of term*)

   Terms
      should not be indented

   Definitions
      should be indented under the term

Line spacing
   there should be a blank line between term-definition pairs
```

which gives

Definition list a list with several term-definition pairs in the form

TERM
DEFINITION (*description of term*)

Terms should not be indented

Definitions should be indented under the term

Line spacing there should be a blank line between term-definition pairs

1.8.3.1 Field (description) Lists

Field lists are special definition lists. They may also be used for two-column table-like structures resembling database records (label & data pairs). *Sphinx* extends standard docutils behavior for Field Lists and intercepts field lists specified at the beginning of documents and adds some extra (optional) functionality.

field list

the example

```
:Date: 2001-08-16
:Version: 1
:Authors: - Me
          - Myself
          - I
:Indentation: Since the field marker may be quite long, the second
   and subsequent lines of the field body do not have to line up
   with the first line, but they must be indented relative to the
   field name marker, and they must line up with each other.
:Parameter i: integer
```

which gives

Date 2001-08-16

Version 1

Authors

- Me
- Myself
- I

Indentation Since the field marker may be quite long, the second and subsequent lines of the field body do not have to line up with the first line, but they must be indented relative to the field name marker, and they must line up with each other.

Parameter i integer

1.9 Explicit Markup

"Explicit markup" is used in *reStructuredText* for most constructs that need special handling, such as footnotes, specially-highlighted paragraphs, comments, and generic directives.

An explicit markup block begins with a line starting with two dots followed by whitespace (".. ") and is terminated by the next paragraph at the same level of indentation. There needs to be a blank line between explicit markup and normal paragraphs. This may all sound a bit complicated, but it is intuitive enough when you write it.

1.9.1 Comments

Every explicit markup block which is not a valid markup construct (like the footnotes above) is regarded as a comment.

However, it must have some text in the ".. " line, otherwise it is ignored, and content will be displayed (indented).

the example

```
.. This is a comment
..
   _so: is this!
..
   [and] this!
..
   this:: too!
..
   |even| this:: !
```

1.9.2 Directives

Directives (page 15) are generic blocks of explicit markup. Besides *Roles* (page 13), it is one of the extension mechanisms of *reStructuredText*, and *Sphinx* makes heavy use of it. Basically, a directive consists of a name, arguments, options and content. Keep this terminology in mind, it is used in one of the next chapter describing custom directives.

the example

```
.. cpp:function:: char* foo(x)
                  char* foo(y, z)
   :noindexentry:

   Return a line of text input from the user.
```

which gives

char *foo(x)

char *foo(y, z)

Return a line of text input from the user.

`.. cpp:function::` is the directive name. It is given two arguments here, the remainder of the first line and the second line, as well as one option `:noindexentry:`. As you can see, options are given in the lines immediately following the arguments and indicated by the colons.

The directive content follows after a blank line and is indented relative to the directive start.

If you want to suppress the addition of an entry in the shown index, you can give the directive option flag `:noindexentry:`. If you want to typeset an object description, without even making it available for cross-referencing, you can give the directive option flag `:noindex:` (which implies `:noindexentry:`).

Hint: As far as possible, all examples in this document use the `:noindexentry:` option to keep the automatically created index as clean as possible but still be able to reference it.

1.9.3 Footnotes

For footnotes, use `[#]_` to mark the footnote location, and add the footnote body at the bottom of the document after a "Footnotes" rubric heading.

the example

```
Lorem ipsum [#]_ dolor sit amet ... [#]_

.. rubric:: Footnotes

.. [#] Text of the first footnote.
.. [#] Text of the second footnote.
```

which gives Lorem ipsum[1] dolor sit amet ...[2]

You can also explicitly number the footnotes for better context.

Text of the first footnote.
Text of the second footnote.

1.9.4 Citations

Citations are identical to footnotes except that they use only non-numeric labels such as `[note]_` or `[GVR2001]_`. Citation labels are simple reference names (case-insensitive single words consisting of alphanumerics plus internal hyphens, underscores, and periods; no whitespace). Citations may be rendered separately and differently from footnotes.

the example

```
Here is a citation reference: [CIT2002]_.

.. [CIT2002] This is the citation. It's just like a footnote,
   except the label is textual.
```

which gives Here is a citation reference: [CIT2002].

To use a professional bibliography, you should use the *Sphinx* extension *BibTeX Citations* (page 84).

1.10 Reuse Content

Sphinx supports several ways to reuse content within and across projects.

1.10.1 Include a Shared File

`.. include::`

For more details, see Including an External Document Fragment in *Docutils*.

You can store complex content, such as tasks, or code samples, in a file that is then included in multiple *reStructuredText* document files.

If you are working on multiple documents, you can save entire topics in shared files, and include those files in multiple documents.

You add a shared file to content in your project with the `.. include::` directive. For example:

```
.. include:: /{absolut-document-subdirectory}/{file}.rsti
.. include:: {relative-document-subdirectory}/{file}.rsti
```

The contents of the shared file will then be built in the document.

> **Caution:** Include paths are relative to the file in the document project, not the file in shared content.

Standard data files intended for inclusion in *reStructuredText* documents are distributed with the *Docutils* source code, located in the `docutils` package in the `docutils/parsers/rst/include` directory. To access these files, use the special syntax for standard include data files, angle brackets around the file name:

```
.. include:: <isonum.txt>
```

Note: You must reference the shared file from a file within the document. You cannot use a direct TOC reference to files outside of the document directory.

1.10.2 Substitutions

Substitutions are a useful way to define a value which is needed in many places. Substitution definitions are indicated by an explicit markup start (”`.. `”) followed by a vertical bar, the substitution text (which gets substituted), another vertical bar, whitespace, and the definition block.

A substitution definition block may contain inline-compatible directives such as *Images and Figures* (page 41), *Downloadable Files* (page 72), or other Substitution Directives:

- Replacement Text
- Unicode Character Codes
- Date

For more information, see reStructuredText Primer, section *Substitutions*, or refer the Substitution References. *Sphinx* provides additional predefined Substitutions.

`.. replace::`

The example

```
.. |RST| replace:: reStructuredText

Here, :rst:`|RST|` will be replaced by |RST|.
```

Which gives Here, |RST| will be replaced by reStructuredText.

1.10.2.1 Styled Reference

You can also create a reference with styled text, nested inline markup.

the example

```
.. |gh| replace:: :strong:`GitHub`
.. _`gh`: https://github.com/

Here, :rst:`|gh|` will be replaced by |gh|.

You can use the hyperlink reference by appending a :rst:`”_”` at the␣
↪end
of the vertical bars and :rst:`|gh|_` will be replaced by |gh|_.
```

which gives Here, |gh| will be replaced by **GitHub**.

You can use the hyperlink reference by appending a ”_” at the end of the vertical bars and |gh|_ will be replaced by **GitHub**.

1.10.2.2 Use Prolog and Epilog

The *Sphinx* configuration values `rst_prolog` and `rst_epilog` in `conf.py` contains a list of global substitutions that can be used from any file. The (incomplete) list for this document is given below:

"|project|" → leads to: "Li-Pro.Net Sphinx Primer"

"|author|" → leads to: "The LP/N Documentation Team"

"|publisher|" → leads to: "Li–Pro.Net"

"|copyright|" → leads to: "2020, Li–Pro.Net, The LP/N Documentation Team and individual contributors."

"|LICENSE|" → leads to: "LICENSE"

"|CREDITS|" → leads to: "CREDITS"

1.10.2.3 Inline Image

You can add inline images in the document using substitutions. The following block of code substitutes arrow in the text with the image specified.

the example

```
:|lpn_16x16|: The logo as in front of this documentation.

.. |lpn_16x16| image:: /_images/lpn.*
                :alt: Li-Pro.Net.
                :height: 16px
                :width: 16px
```

which gives

 The logo as in front of this documentation.

1.11 Images and Figures

1.11.1 SVG Graphics only

All vector graphics or diagrams should be *SVG* files. This helps us keep our graphic conversion tooling simple, and generally results in higher-quality representation. *SVG* graphics with an parameterized opacity (transparency) should be possible as well as an animated *SVG*, see Figure 1.1 and Figure 1.2.

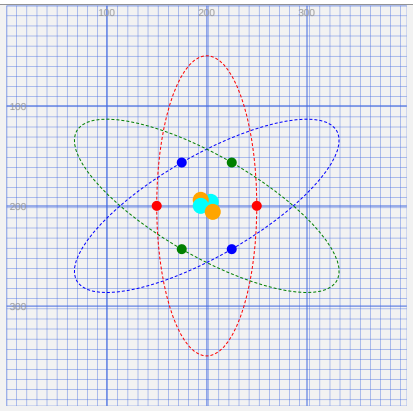


Figure 1.1: Example of transparent SVG[1]

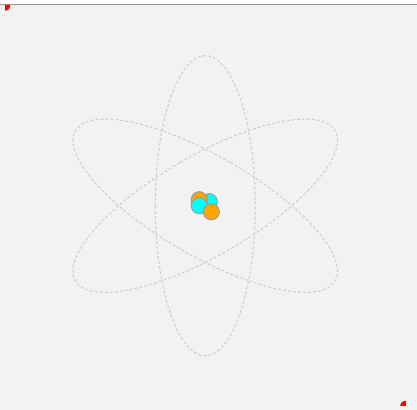


Figure 1.2: Example of animated SVG[2]

Whenever possible, you should generate your graphics as *SVG* rather than converting to *SVG* from another format. That avoids bitmap raster images embedded in a *SVG* container. The goal of *SVG* usage is to hold vector graphic as long as possible, from the editor up to the presentation. If you have to start in another vector graphic format **use lossless vector formats** whenever possible. These include EPS (Encapsulated PostScript)/PS (PostScript), AI (Adobe Illustrator Artwork), DXF (AutoCAD Drawing Exchange Format), EMF (Enhanced Metafile Format)/EMZ (Compressed Enhanced Metafile Format), WMF (Windows Metafile Format)/WMZ (Compressed Windows Metafile Format) or some special *XML* vector graphics schemes. In any case avoid embedded bitmaps, as this is a lossy format for vector informations that does not replicate scaling very well. Figure 1.3 demonstrates differences between bitmapped raster and vector graphics. The bitmap raster is composed of a fixed set of pixels, while the vector is composed of a fixed set of shapes. In the picture, scaling the bitmap reveals the pixels while scaling the vector image preserves the shapes.

Indication of provenance: STEAMcoded.org: atom1.svg (public domain for teachers and students learning to code)
Indication of provenance: STEAMcoded.org: atom.svg (public domain for teachers and students learning to code)

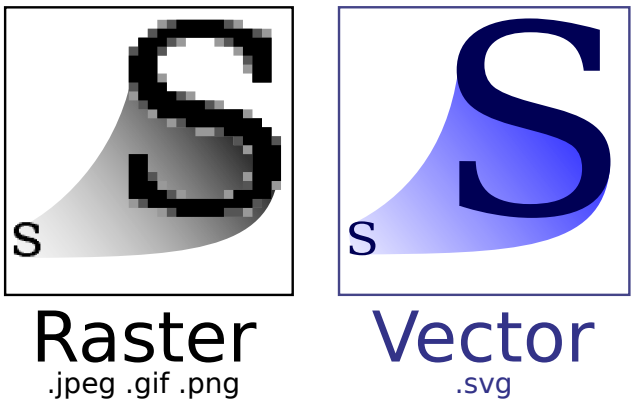


Figure 1.3: Demonstration of differences between bitmapped raster and vector images.[3]

Bitmap raster images are good for photographic images or screenshots but not for stencils, sketches, diagrams or graphs and often they do not support transparency.

—Superuser - JPEG vs. PNG vs. BMP vs. GIF vs. SVG

> Raster graphics are resolution dependent, meaning they cannot scale up to an arbitrary resolution without loss of apparent quality. This property contrasts with the capabilities of vector graphics, which easily scale up to the quality of the device rendering them. Raster graphics deal more practically than vector graphics with photographs and photo-realistic images, while vector graphics often serve better for typesetting or for graphic design.
>
> —Wikipedia - Raster graphics

> Vector graphics have the unique advantage over raster graphics in that the points, lines, and curves may be scaled up or down to any resolution with no aliasing.
>
> —Wikipedia - Vector graphics

Indication of provenance: Wikimedia: 6/6b/Bitmap_VS_SVG.svg (licensed under CC-BY-SA-2.5)

1.11.2 PNG Images only

All still bitmap raster images or photos should be *PNG* files. This helps us keep our image compression tooling simple, and generally results in higher-quality screenshots. *PNG* images with an 8-bit transparency channel should be possible as well as an animated *PNG*, see Figure 1.4 and Figure 1.5.

Figure 1.4: Example of transparent PNG[4]

Figure 1.5: Example of animated PNG[5]

Whenever possible, you should generate your images as *PNG* rather than converting to *PNG* from another format. If you have to start in another format, **use lossless formats** whenever possible. These include BMP (Bitmap Format)/DIB (Device Independent Bitmap Format), GIF (Graphics Interchange Format), and TIFF (Tagged Image File Format). Avoid JPEG (Joint Photographic Experts Group)/JFIF (JPEG File Interchange Format) if possible, as this is a lossy format that does not replicate screenshots very well. Figure 1.6 comparing lossy compression in JPEG with lossless compression in *PNG*: the JPEG artifacts can be easily visible in the background of this kind of image data, where the *PNG* image has solid color.

Indication of provenance: Wikimedia: 4/47/PNG_transparency_demonstration_1.png (licensed under CC-BY-SA-3.0)
Indication of provenance: Wikimedia: 1/14/Animated_PNG_example_bouncing_beach_ball.png (released into the public domain by its author, Holger Will)

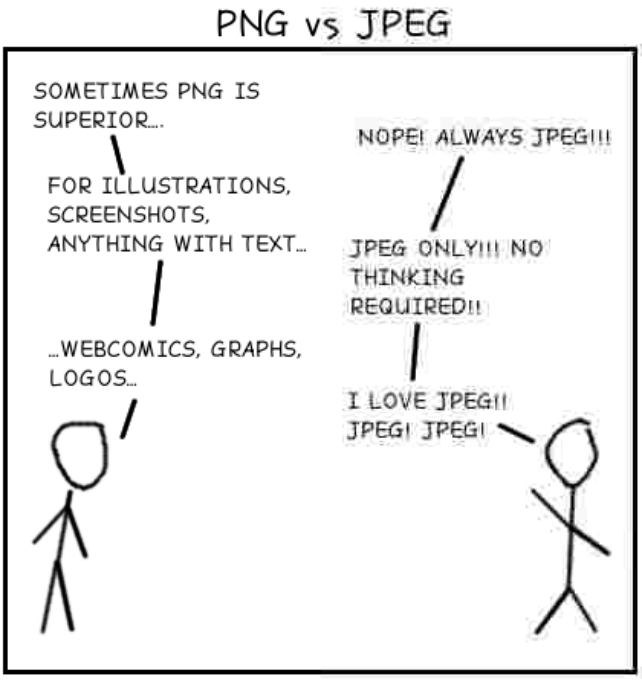


Figure 1.6: Demonstration of differences between lossy encoding and lossless method.[6]

JPEG is good for photographic images but not for sharp transitions and does not support transparency.

—Wikipedia - PNG comparison with JPEG

> The JPEG format can produce a smaller file than *PNG* for photographic (and photo-like) images, since JPEG uses a lossy encoding method specifically *designed for photographic image data*. Using *PNG* instead of a high-quality JPEG for such images would result in a large increase in file size with negligible gain in quality. In comparison, when storing images that contain text, line art, or graphics – images with sharp transitions and large areas of solid color – the *PNG* format can compress image data more than JPEG can. Additionally, *PNG* is lossless, while JPEG produces visual artifacts around high-contrast areas.
>
> JPEG's lossy compression also suffers from generation loss, where repeatedly decoding and re-encoding an image to save it again causes a loss of information each time, degrading the image. This does not happen with repeated viewing or copying, but only if the file is edited and saved over again. Because *PNG* is lossless, it is suitable for storing images to be edited.
>
> Where an image contains both *sharp transitions* and *photographic parts*, a choice must be made between the two effects:

Indication of provenance: Superuser: a/55706, http://lbrandy.com/assets/jpg_vs_png2.png (licensed under CC-BY-SA-3.0)

1.11.3 Inserting

To place an graphic or image in a document, use the `.. image::` directive (see Image).

```
.. image:: /img/{absolut-document-subdirectory}/{file}.svg
  :alt: Alt text. Every image should have descriptive alt text.

.. image:: {relative-document-subdirectory}/{file}.*
  :alt: Alt text. Every image should have descriptive alt text.
```

Note the literal asterisk (*) at the end, in place of a file extension. Use the asterisk, and omit the file extension (see reStructuredText Primer, section *Images*).

`.. image::`

The example

```
.. image:: /_images/lpn.svg
   :alt: The Li-Pro.Net logo. (explicitely as SVG)
   :height: 32px
   :width: 32px

.. image:: /_images/lpn.*
   :alt: The Li-Pro.Net logo. (scaled to the half)
   :scale: 50 %
   :align: right
```

Which gives 

1.11.4 Inserting with Captions

Use `.. figure::` directive to markup a graphic or image with a caption (see Figure).

```
.. figure:: {file-with-directory-same-as-for image}.*
  :alt: Alt text. Every image should have descriptive alt text.

  The rest of the indented content will be the (optional) caption.
  This can be a short sentence or multiline paragraph.
```

Captions can contain any other complex *reStructuredText* markup. Further paragraphs after the caption will be the (optional) legend which are also arbitrary body elements.

`.. figure::`

The example

```
.. figure:: /_images/lpn.*
   :name: the-lpn-logo
   :alt: The Li-Pro.Net logo.
   :figclass: align-center
   :align: center
   :scale: 75 %

   The |ghlpn|_ logo.

   Legend of all elements you can see in the graphic:

   .. list-table:: The legend for the |ghlpn|_ logo.
      :widths: 10 40
      :width: 50 %
      :align: center
      :header-rows: 1
      :stub-columns: 1

      * - Letter
        - Meaning
      * - L (in blue)
        - Li
      * - P (in blue)
        - Pro
      * - N (in red)
        - Net

.. |ghlpn| unicode:: Li U+02013 Pro.Net U+0040 GitHub
.. _`ghlpn`: https://github.com/lipro
```

Which gives

Figure 1.7: The Li–Pro.Net@GitHub logo.
Legend of all elements you can see in the graphic:

Table 1.3: The legend for the Li–Pro.Net@GitHub logo.

Letter	Meaning
L (in blue)	Li
P (in blue)	Pro
N (in red)	Net

1.11.5 Inserting Inline

To information on creating inline images, see *Inline Image* (page 40).

1.12 Tables

For more details, see Table in *Docutils* or Tables Basics and Tables Directives.

`.. table::`

The `.. table::` directive serves as optional wrapper of the *Grid Style* (page 49) and *Simple Style* (page 50).

`.. tabularcolumns::`

The `.. tabularcolumns::` directive gives a `column spec` for the next table occurring in the source file. The spec is the second argument to the *LaTeX* `tabulary` package's environment (which *Sphinx* uses to translate tables). For more details, see `tabularcolumns`.

1.12.1 Grid Style

For more details, see Grid Tables in *Docutils*.

the example

```
.. tabularcolumns:: p{0.132\linewidth}p{0.198\linewidth}p{0.330\
↪linewidth}
.. table:: Example table in grid style
   :name: tables-grid-example
   :widths: 20, 30, 50
   :class: longtable
   :align: center
   :width: 66%

   +------------+------------+-----------+
   | Header 1   | Header 2   | Header 3  |
   +============+============+===========+
   | body row 1 | column 2   | column 3  |
   +------------+------------+-----------+
   | body row 2 | Cells may span columns.|
   +------------+------------+-----------+
   | body row 3 | Cells may  | - Cells   |
   +------------+ span rows. | - contain |
   | body row 4 |            | - blocks. |
   +------------+------------+-----------+
```

which gives

Table 1.4: Example table in grid style

Header 1	Header 2	Header 3
body row 1	column 2	column 3
body row 2	Cells may span columns.	
body row 3	Cells may span rows.	• Cells • contain • blocks.
body row 4		

1.12.2 Simple Style

For more details, see Simple Tables in *Docutils*.

the example

```
.. tabularcolumns:: p{0.132\linewidth}p{0.198\linewidth}p{0.330\
↪linewidth}
.. table:: Example table in simple style
   :name: tables-simple-example
   :widths: 20, 30, 50
   :align: center
   :width: 66%

   ====== ====== ======
       Inputs    Output
   ------------ ------
   A      B      A or B
   ====== ====== ======
   False
   --------------------
   True
   --------------------
   True   False  True
   ------ ------ ------
   False  True
   ====== =============
```

which gives

Table 1.5: Example table in simple style

Inputs		Output
A	**B**	**A or B**
False		
True		
True	False	True
False	True	

1.12.3 List Table

.. list-table::

For more details, see List Tables in *Docutils*.

Hint: For table content that needs a higher complexity than the list table is able to support use the *flat-table* (page 86).

The example

```
.. tabularcolumns:: p{0.132\linewidth}p{0.198\linewidth}p{0.330\
↪linewidth}
.. list-table:: Example list table
   :name: tables-list-example
   :widths: 20, 30, 50
   :class: longtable
   :header-rows: 1
   :align: center
   :width: 66%

   * - Treat
     - Quantity
     - Description
   * - Albatross
     - 2.99
     - On a stick!
   * - Crunchy Frog
     - 1.49
     - If we took the bones out, it would not be
       crunchy, now would it?
   * - Gannet Ripple
     - 1.99
     - On a stick!
```

Which gives

Table 1.6: Example list table

Treat	Quantity	Description
Albatross	2.99	On a stick!

continues on next page

Table 1.6 – continued from previous page

Treat	Quantity	Description
Crunchy Frog	1.49	If we took the bones out, it would not be crunchy, now would it?
Gannet Ripple	1.99	On a stick!

1.12.4 CSV Table

.. csv-table::

For more details, see CSV Tables in *Docutils*.

Hint: In almost all cases, *csv-table* (page 53) is the easiest and most maintainable way to insert a table into a document. It should be preferred unless there is a compelling reason to use one of the other styles.

The example

```
.. tabularcolumns:: p{0.132\linewidth}p{0.198\linewidth}p{0.330\
↪linewidth}
.. csv-table:: Example CSV table
   :name: tables-csv-example
   :header: "Treat", "Quantity", "Description"
   :widths: 20, 30, 50
   :class: longtable
   :align: center
   :width: 66%

   "Albatross", 2.99, "On a stick!"
   "Crunchy Frog", 1.49, "If we took the bones out, it would not be
   crunchy, now would it?"
   "Gannet Ripple", 1.99, "On a stick!"
```

Which gives

Table 1.7: Example CSV table

Treat	Quantity	Description
Albatross	2.99	On a stick!
Crunchy Frog	1.49	If we took the bones out, it would not be crunchy, now would it?
Gannet Ripple	1.99	On a stick!

Some of the options recognized are:

:widths:

Contains a comma or space-separated list of relative column widths. The default is equal-width columns.

The special value `auto` may be used by writers to decide whether to delegate the determination of column widths to the backend.

In most cases, the best result is either the default or `auto`. If you're unsure, try it both ways and see which looks better to you.

:header:

Contains column titles. It must use the same CSV format as the main CSV data.

:delim:

Contains a one character string used to separate fields. Default value is comma. It must be a single character or Unicode code.

The only reason to use something other than a comma is when copying large blocks of content from another source that uses a different style. If you are creating new table content yourself, use the comma.

The example

```
.. csv-table:: Example CSV table with customized delimiter
   :name: tables-csv-delim-example
   :header: "Name", "Password"
   :widths: auto
   :delim: :
   :align: center
   :width: 66%

   "Peter":":literal:`QW8rTn@*emk;=J3f`"
   "Paul":":literal:`b3%C/-9\`][cnG,;{`"
```

Which gives

Table 1.8: Example CSV table with customized delimiter

Name	Password
Peter	QW8rTn@*emk;=J3f
Paul	b3%C/-9`][cnG,;{

:align:

It specifies the horizontal alignment of the table. It can be `left`, `right` or `center`.

The example

```
.. csv-table:: Example CSV table with right alignment
   :name: tables-csv-align-example
   :header: "Name", "Password"
   :delim: #
   :align: right
```

(continues on next page)

(continued from previous page)

```
6
7    "Peter"#":literal:`QW8rTn@*emk;=J3f`"
8    "Paul"# ":literal:`b3%C/-9\`][cnG,;{`"
```

Which gives

Table 1.9: Example CSV table with right alignment

Name	Password
Peter	QW8rTn@*emk;=J3f
Paul	b3%C/-9`][cnG,;{

:url:
Contains an Internet URL reference to a CSV data file.

:file:
Contains the local file system path to a CSV data file.

The example

```
1  .. csv-table:: Example CSV table from source file
2     :name: tables-csv-srcfile-example
3     :file: example.csv
4     :delim: |
5     :encoding: utf-8-sig
6     :header-rows: 1
7     :stub-columns: 1
8     :width: 66%
```

Which gives

Table 1.10: Example CSV table from source file

Name	Password
Peter	QW8rTn@*emk;=J3f
Paul	b3%C/-9`][cnG,;{

Which needs The example above processed the following CSV file content:

Listing 1.1: CSV example file (tables/csv/srcfile/example.csv)

```
1 Name|Password
2 Peter|:literal:`QW8rTn@*emk;=J3f`
3 Paul|:literal:`b3%C/-9\`][cnG,;{`
```

Note: There is no support for checking that the number of columns in each row is the same. However, this directive supports CSV generators that do not insert “empty” entries at the end of short rows, by automatically adding empty entries.

1.13 Code Example

The syntax for displaying code is the `::` mark, see Literal blocks. When it is used at the end of a sentence, *Sphinx* is smart and displays one `:` sign in the output, and knows there is a code example in the following indented block, the Indented literal (code) block. Quoted literal (code) block are unindented contiguous blocks of text where each line begins with the same non-alphanumeric printable 7-bit ASCII character.

`.. highlight::`

For more details, see `highlight` directive.

The example

```
.. highlight:: none

This is a normal text paragraph. The next paragraph
is a code sample::

   It is not processed in any way, except
   that the indentation is removed.

   It can span multiple lines.

This is a normal text paragraph again.

The next paragraph is a quoted sample -- John Doe wrote::

>> Great idea!
>
> Why didn't I think of that?

You just did!  ;-)
```

Which gives This is a normal text paragraph. The next paragraph is a code sample:

```
It is not processed in any way, except
that the indentation is removed.

It can span multiple lines.
```

This is a normal text paragraph again.

The next paragraph is a quoted sample – John Doe wrote:

```
>> Great idea!
>
> Why didn't I think of that?
```

You just did! ;-)

The handling of the `::` marker is smart:

- If it occurs as a paragraph of its own, that paragraph is completely left out of the document.
- If it is preceded by whitespace, the marker is removed.
- If it is preceded by non-whitespace, the marker is replaced by a single colon.

That way, the first sentence in the above example's first paragraph would be rendered as "... The next paragraph is a code sample:".

Sphinx extends the default language setup for each literal (code) block with the `.. highlight::` directive. That is very useful if a specific directive is not able to set the language by argument or option, even in this case here.

1.13.1 Explicit Code Blocks

Source code will be formatted by the directive `.. code-block::`. *Sphinx*, like *Python*, uses meaningful whitespace. Blocks of content are structured based on the indention level they are on.

`.. code-block::`

For more details, see `code-block` directive.

The example

```
.. highlight:: bash
   :linenothreshold: 1

A cool bit of code::

   #!/bin/bash
   # Some cool Bash code
   echo ${BASH_VERSINFO[*]}

.. highlight:: none

.. code-block:: rst
   :caption: Documentation

   A bit of **rst** which should be *highlighted* properly.

.. code-block:: python
   :caption: Script
   :linenos:

   import sys
   sys.exit(1)
```

Which gives A cool bit of code:

```
#!/bin/bash
# Some cool Bash code
echo ${BASH_VERSINFO[*]}
```

Listing 1.2: Documentation

```
A bit of **rst** which should be *highlighted* properly.
```

Listing 1.3: Script

```
import sys
sys.exit(1)
```

Valid values for the highlighting `:language:` (first argument) are:

- `none` (no highlighting)
- `python` (the default)
- `c` and `cpp` (*C*/*C++*)
- `rst` or `rest` (*reStructuredText*)
- `bash` or `ksh` or `sh` (Unix Shell scripts)
- `shell-session` (Unix Shell sessions)
- `ps1` or `posh` or `powershell` (Windows PowerShell code)
- `ps1con` (Windows PowerShell sessions)
- `dosbatch` or `winbatch` (MS-DOS (Microsoft Disk Operating System)/Windows Batch file)
- `doscon` (MS-DOS sessions)
- `cfg` or `ini` (Generic configuration file, mostly INI files)
- `sql` (Generic SQL (Structured Query Language) commands)
- `registry` (Windows Registry files produced by **`regedit`**)
- `guess` (let *Pygments* guess the lexer based on contents, only works with certain well-recognizable languages)
- ... and any other lexer alias that Pygments supports.

1.13.2 Explicit Code Includes

If the text resides in a separate file, use the `.. literalinclude::` directive:

`.. literalinclude::`

For more details, see `literalinclude` directive.

The example

```
.. literalinclude:: /docutils.conf
   :language: cfg
```

Which gives

```
;
; Docutils Configuration
;
; The configuration file consists of sections, lead by a "[section]"
; header and followed by "name: value" entries, with continuations
; in the style of RFC 822; "name=value" is also accepted. Note that
; leading whitespace is removed from values. ... Lines beginning
; with "#" or ";" are ignored and may be used to provide comments.
;
; see: https://docutils.sourceforge.io/docs/user/config.html
;

; https://docutils.sourceforge.io/docs/user/config.html#parsers
; https://docutils.sourceforge.io/docs/user/config.html
↪#restructuredtext-parser

[restructuredtext parser]
syntax_highlight = short
```

All included files could be located under `/include`. The beginning `/` means, root directory of the documentation source directory. Without it, the path is relative to the directory of the including file.

1.14 Mathematics

In *Sphinx* you can include inline math $x \leftarrow y\ x \forall y\ x - y$ (as role `:math:`x\leftarrow y\ x\forall y\ x-y``) or display math as directive block which is able to cross-referencing equations:

$$W^{3\beta}_{\delta_1\rho_1\sigma_2} = U^{3\beta}_{\delta_1\rho_1} + \frac{1}{8\pi 2}\int_{\alpha_2}^{\alpha_2} d\alpha_2' \left[\frac{U^{2\beta}_{\delta_1\rho_1} - \alpha_2' U^{1\beta}_{\rho_1\sigma_2}}{U^{0\beta}_{\rho_1\sigma_2}}\right] \tag{1.1}$$

`.. math::`

To include math in your document, just use the `.. math::` directive. For more details, see `math` directive.

The example

```
.. math::

   W^{3\beta}_{\delta_1 \rho_1 \sigma_2}
   \approx U^{3\beta}_{\delta_1 \rho_1}
```

Which gives

$$W^{3\beta}_{\delta_1\rho_1\sigma_2} \approx U^{3\beta}_{\delta_1\rho_1} \tag{1.2}$$

`:math:numref:`

The math domain (name **math**) provides the role `:math:numref:`label`` which is for cross-referencing equations defined by `.. math::` directive via their label. For more details, see `math:numref` role.

The example

```
.. math:: e^{i\pi} + 1 = 0
   :label: euler

Euler's identity, :math:numref:`euler`, was elected one
of the most beautiful mathematical formulas.
```

Which gives

$$e^{i\pi} + 1 = 0 \tag{1.3}$$

Euler's identity, Equastion 1.3, was elected one of the most beautiful mathematical formulas.

When the equation is only one line of text, it can also be given as a directive argument (as used in Euler's identity above).

:eq:

The role `:eq:`euler`` is the same as `:math:numref:`euler``. For more details, see eq role.

Recent versions of *Sphinx* include built-in support for math. There are three flavors:

- `sphinx.ext.imgmath`: uses dvipng to render the equation
- `sphinx.ext.mathjax`: renders the math in the browser using Javascript
- `sphinx.ext.jsmath`: it's an older code, but it checks out

Additionally, there are special *Sphinx* extensions provided by matplotlib that has its own math support for writing mathematical expressions and inserting automatically-generated plots:

- `matplotlib.sphinxext.mathmpl`
- `matplotlib.sphinxext.plot_directive`

See also:

See *Mathematical Plots* (page 93) for more details about the *Sphinx* matplotlib extensions with examples.

1.15 Admonitions

1.15.1 Generic Admonition

The Generic admonition is a simple titled admonition. The title may be anything the author desires. The author-supplied title is also used as a “classes” attribute value after being converted into a valid identifier form. As well as this implicitly behavior the `:class:` option value can set to any implemented specific type and overrides the computed “classes” attribute value.

`.. admonition::`

The example

```
.. admonition:: Neque porro quisquam
   :class: error

   Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus
   mattis commodo eros, quis posuere enim lobortis quis. Nullam ut
   tempus nibh.
```

Which gives

Neque porro quisquam

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus mattis commodo eros, quis posuere enim lobortis quis. Nullam ut tempus nibh.

1.15.2 Specific Admonitions

Specific Admonitions are specially marked “topics” that can appear anywhere an ordinary body element can. Typically, an admonition is rendered as an offset block in a document, sometimes outlined or shaded, with a title matching the admonition type. The following admonition directives have been implemented.

1.15.2.1 Attention Admonition

`.. attention::`

The example

```
.. attention:: Neque porro quisquam

   Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus
```

(continues on next page)

(continued from previous page)

```
4   mattis commodo eros, quis posuere enim lobortis quis. Nullam ut
5   tempus nibh.
```

Which gives

> **Attention:** Neque porro quisquam
>
> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus mattis commodo eros, quis posuere enim lobortis quis. Nullam ut tempus nibh.

1.15.2.2 Caution Admonition

`.. caution::`

The example

```
1  .. caution:: Neque porro quisquam
2
3     Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus
4     mattis commodo eros, quis posuere enim lobortis quis. Nullam ut
5     tempus nibh.
```

Which gives

> **Caution:** Neque porro quisquam
>
> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus mattis commodo eros, quis posuere enim lobortis quis. Nullam ut tempus nibh.

1.15.2.3 Danger Admonition

`.. danger::`

The example

```
1  .. danger:: Neque porro quisquam
2
3     Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus
4     mattis commodo eros, quis posuere enim lobortis quis. Nullam ut
5     tempus nibh.
```

Which gives

> **Danger:** Neque porro quisquam
>
> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus mattis commodo eros, quis posuere enim lobortis quis. Nullam ut tempus nibh.

1.15.2.4 Error Admonition

`.. error::`

The example

```
.. error:: Neque porro quisquam

   Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus
   mattis commodo eros, quis posuere enim lobortis quis. Nullam ut
   tempus nibh.
```

Which gives

> **Error:** Neque porro quisquam
>
> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus mattis commodo eros, quis posuere enim lobortis quis. Nullam ut tempus nibh.

1.15.2.5 Hint Admonition

`.. hint::`

The example

```
.. hint:: Neque porro quisquam

   Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus
   mattis commodo eros, quis posuere enim lobortis quis. Nullam ut
   tempus nibh.
```

Which gives

Hint: Neque porro quisquam

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus mattis commodo eros, quis posuere enim lobortis quis. Nullam ut tempus nibh.

1.15.2.6 Important Admonition

`.. important::`

The example

```
.. important:: Neque porro quisquam

   Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus
   mattis commodo eros, quis posuere enim lobortis quis. Nullam ut
   tempus nibh.
```

Which gives

Important: Neque porro quisquam

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus mattis commodo eros, quis posuere enim lobortis quis. Nullam ut tempus nibh.

1.15.2.7 Note Admonition

`.. note::`

For more details, see `note` directive.

The example

```
.. note:: Neque porro quisquam

   Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus
   mattis commodo eros, quis posuere enim lobortis quis. Nullam ut
   tempus nibh.
```

Which gives

Note: Neque porro quisquam

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus mattis commodo eros, quis posuere enim lobortis quis. Nullam ut tempus nibh.

1.15.2.8 Tip Admonition

`.. tip::`

The example

```
.. tip:: Neque porro quisquam

   Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus
   mattis commodo eros, quis posuere enim lobortis quis. Nullam ut
   tempus nibh.
```

Which gives

Tip: Neque porro quisquam

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus mattis commodo eros, quis posuere enim lobortis quis. Nullam ut tempus nibh.

1.15.2.9 Warning Admonition

`.. warning::`

For more details, see warning directive.

The example

```
.. warning:: Neque porro quisquam

   Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus
   mattis commodo eros, quis posuere enim lobortis quis. Nullam ut
   tempus nibh.
```

Which gives

Warning: Neque porro quisquam

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus mattis commodo eros, quis posuere enim lobortis quis. Nullam ut tempus nibh.

1.15.3 Sphinx Additional Admonitions

1.15.3.1 Seealso Admonition

.. seealso::

Many sections include a list of references to module documentation or external documents. These lists are created using the `seealso` directive.

The example

```
.. seealso:: Neque porro quisquam

   Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus
   mattis commodo eros, quis posuere enim lobortis quis. Nullam ut
   tempus nibh.
```

Which gives

See also:

Neque porro quisquam

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus mattis commodo eros, quis posuere enim lobortis quis. Nullam ut tempus nibh.

1.16 Hyperlink

The link text is set by putting a _ after some text. The ` is used to group text, allowing you to include multiple words in your link text. You should use the `, even when the link text is only one word. This keeps the syntax consistent.

The link target is defined inline or at the bottom of the section with `.. _<link text>: <target>` (*reference style*).

The Hyperlink Targets in *Docutils* provides the basic specification for external and anonymous hyperlink targets. These are also called explicit hyperlink targets.

the example

```
`A cool website`_ and `The Dino <https://docutils.sourceforge.io/>`_.

.. _A cool website: https://www.sphinx-doc.org/
```

which gives A cool website and The Dino.

1.17 Referencing

Another important *Sphinx* feature is that it allows referencing across documents. This is another powerful way to tie documents together.

The simplest way to do this is to define an explicit reference object which can then be referenced directly as internal hyperlink target or with `:ref:`refname`` or in rare cases with `:numref:`refname`` depending on the `toctree` section numbering setup. *Sphinx* also supports `:doc:`docname`` for linking to a document via built-in extension `sphinx.ext.intersphinx` and also supports autogenerated labels for each section via built-in extension `sphinx.ext.autosectionlabel`.

The Hyperlink Targets in *Docutils* provides the basic specification for internal and external hyperlink targets. These are also called explicit and also available as implicit hyperlink targets.

`:doc:`

`:ref:`

`:numref:`

For more details, see `doc`, `ref` and `numref` role.

The example

```
Clicking on this internal hyperlink will take us to the target_
↪below.

:ref:`reference-name`, this is
:ref:`the same (cool) section <reference-name>`, and again the_
↪reference
```

(continues on next page)

(continued from previous page)

```
with support of the extension **autosectionlabel** --
:ref:`concepts/referencing:A cool section` or :doc:`./referencing`.
```

```
.. _reference-name:

A cool section
""""""""""""""""""

.. _target:

The hyperlink target above points to this paragraph.
```

Which gives Clicking on this internal hyperlink will take us to the *target* (page 71) below.

A cool section (page 71), this is *the same (cool) section* (page 71), and again the reference with support of the extension **autosectionlabel** – *A cool section* (page 71) or *Referencing* (page 70).

1.17.1 A cool section

The hyperlink target above points to this paragraph.

1.18 External References

Sphinx also includes a number of predefined references for external concepts. Things like PEP's and RFC's. You can read more about this in the *Sphinx Roles* (page 13) section.

:pep:

:rfc:

For more details, see `pep` and `rfc` role.

The example

```
You can learn more about this at :pep:`8` or :rfc:`1984`.
```

Which gives You can learn more about this at **PEP 8** or **RFC 1984**.

1.19 Downloadable Files

To place a downloadable file in a document, use the *download* (page 72) role.

```
To understand the procedure better, see this
:download:`example script </_downloads/contributing/example_script.py>`.
```

Downloadable files with a dedicated context to a specific part of the documentation should also be placed exactly at this point in the source tree of the documentation. Other common artefacts should be put in the documentation root or the `/_downloads/` subdirectory, and they should be in a subdirectory with the same name as the document in which they appear (that is, the filename without the `.rst` extension).

> **Attention:** Downloads are not fully supported by all *Sphinx* builders. Especially offline documents like *LaTeX*/*PDF* will be created correctly, but will not provide additional artifacts.

:download:

For more details, see download role.

The example

```
All artifacts were selected and download by using this references:

* matplotlib example: |ellipses.py| (published as `ellipses.py`_)

.. |ellipses.py| replace::
   :download:`ellipses.py </_images/mplplots/ellipses.py>`
.. _`ellipses.py`:
   https://matplotlib.org/gallery/shapes_and_collections/ellipse_
↪demo.html#ellipse-rotated
```

Which gives All artifacts were selected and download by using this references:

- matplotlib example: `ellipses.py` (published as ellipses.py)

1.20 Semantic Descriptions and References

Sphinx also has much more powerful semantic referencing capabilities, which knows all about software development concepts.

Say you're creating a CLI (Command Line Interface) application. You can define the name of the executable application `program` and its `option` and `envvar` quite easily.

.. program::

.. option::

.. envvar::

For more details, see `program`, `option` and `envvar` directive.

The example

```
.. _dlapp:

.. rubric:: Dreamland

A fantasy command-line interface application.

.. program:: dlapp

.. option:: -f, --force

   Force the operation.

.. option:: -i <regex>, --ignore <regex>

   Ignore things that match a specific pattern.

.. envvar:: DLAPPRC

   Set location of resources.
```

Which gives

Dreamland

A fantasy command-line interface application.

-f, **--force**
Force the operation.

-i <regex>, **--ignore** <regex>
Ignore things that match a specific pattern.

DLAPPRC
Set location of resources.

That can now also be referenced quite simply.

:program:

:option:

:envvar:

For more details, see program, option and envvar role.

The example

```
.. seealso:: Working with :ref:`dlapp` (:program:`dlapp`):

   * forcing with :option:`dlapp --force`
   * ignoring with :option:`dlapp -i`
   * defaults with :envvar:`DLAPPRC`
```

Which gives

See also:

Working with *Dreamland* (page 73) (**dlapp**):

- forcing with *dlapp --force* (page 73)
- ignoring with *dlapp -i* (page 73)
- defaults with *DLAPPRC* (page 73)

Sphinx includes a large number of these semantic types, including:

- The C Domain (name **c**): c:namespace, c:struct, c:var, c:function,...
- The C++ Domain (name **cpp**): cpp:namespace, cpp:class,...
- The *JavaScript* Domain (name **js**): js:module, js:class,...
- The *Python* Domain (name **py**): py:module, py:class,...
- The *reStructuredText* Domain (name **rst**): rst:directive, rst:role,...

1.21 Writing about User Interface

Several roles are used when describing user interactions.

:guilabel:

Marks up *actual UI text* of form labels or buttons. For more details, see guilabel role.

The example

```
Press the :guilabel:`Submit` button.
```

Which gives Press the *Submit* button.

:menuselection:

Marks up the *actual UI text* of a navigation menu or form select element. For more details, see menuselection role.

The example

```
Select :menuselection:`Help` from menu.

To save your file, go to :menuselection:`File --> Save` in the Main
↪Menu.
```

Which gives Select *Help* from menu.

To save your file, go to *File* ▸ *Save* in the Main Menu.

When writing about multi-level menus, use a single `:menuselection:` role, and separate menu choices with `-->`.

Note: In some situations you might not be clear about which option, menuselection or guilabel, to use. GUI (Graphics User Interface)s in real life can sometimes be ambiguous. The general rule is:

- Actual UI (User Interface) text will always receive guilabel role unless the text could reasonably be understood to be part of a menu.
- If the actual UI text could be understood as a menu, menuselection should be used.

These both render the same on output, so don't worry too much if you get it wrong. Just use your judgment and take your best guess.

:kbd:

Marks up a sequence of literal keyboard strokes. For more details, see kbd role.

The example

```
To stop the local server, type :kbd:`CTRL+C`.
```

Which gives To stop the local server, type CTRL+C.

:command:

Marks up a terminal command. For more details, see command role.

The example

```
To build the documentation, use :command:`sphinx-build`.
```

Which gives To build the documentation, use **sphinx-build**.

To document a CLI application, you will find more information in *Semantic Descriptions and References* (page 73).

1.21.1 Other Semantic Markup

:abbr:

Marks up an abbreviation. If the role content contains a parenthesized explanation, it will be treated specially: it will be shown in a tool-tip in *HTML*. For more details, see abbr role.

The example

```
This is the :abbr:`ISDN (Integrated Services Digital Network)`
device.
```

Which gives This is the ISDN (Integrated Services Digital Network) device.

:dfn:

Marks the defining instance of a term outside the index or glossary. For more details, see dfn role.

The example

```
This library has a :dfn:`CAPI`, a Common ISDN Application
Programming Interface.
```

Which gives This library has a *CAPI*, a Common ISDN Application Programming Interface.

1.22 Glossary

Sphinx has a built-in Glossary structure that you can use to:

- Produce a consolidated glossary of terms.
- Link terms in other content to their glossary definitions.

1.22.1 Create a Glossary

.. glossary::

For more details, see `glossary` directive.

To add glossary terms, you use the directive `.. glossary::`. Write each glossary entry as a definition list, with a term, followed by a single-line indented definition.

Each glossary entry is nested below the `.. glossary::` directive. For example:

```
.. glossary::

   Sphinx
      Sphinx is a tool that makes it easy to create intelligent and
      beautiful documentation. It was originally created for the
      Python documentation, and it has excellent facilities for the
      documentation of software projects in a range of languages.

   RST
      reStructuredText is an easy-to-read, what-you-see-is-what-you-get
      plain text markup syntax and parser system. It is useful for
      in-line program documentation (such as Python docstrings), for
      quickly creating simple web pages, and for standalone documents.
      reStructuredText is designed for extensibility for specific
      application domains. The reStructuredText parser is a component
      of Docutils.

   Sublime Text
      Sublime Text is a sophisticated text editor for code, markup
      and prose. You'll love the slick user interface, extraordinary
      features and amazing performance.
```

1.22.2 Link a Term to its Glossary Entry

:term:

For more details, see index role.

When a glossary term is used in text, you can link it to its definition with the :term: role. For example, to link the term *Sphinx* to its definition, use the following syntax:

```
:term:`Sphinx`
```

The term specified must exactly match a term in Glossary directive.

You can link to a term in the glossary while showing different text in the topic by including the term in angle brackets. For example:

```
:term:`reStructuredText<RST>`
```

The term in angle brackets must exactly match a term in the glossary. The text before the angle brackets is what users see on the page.

1.23 Index

.. index::

For more details, see index directive.

Some roles and directives do already create indices automatically.

However, there is also an explicit directive available, to make the index more comprehensive and enable index entries in documents where information is not mainly contained in information units.

The directive is `.. index::` and contains one or more index entries Each entry consists of a type and a value, separated by a colon. For example:

```
.. index::
   single: execution; context
   triple: module; search; path
```

:index:

For more details, see index role.

While the *index* (page 79) directive is a block-level markup and links to the beginning of the next paragraph, there is also a corresponding role that sets the link target directly where it is used.

The content of the role can be a simple phrase, which is then kept in the text and used as an index entry. It can also be a combination of text and index entry, styled like with explicit targets of cross-references. In that case, the "target" part can be a full entry as described for the directive above. For example:

```
This is a normal reST :index:`paragraph` that contains several
:index:`index entries <pair: index; entry>`.
```

Note: The `:index:` role must contain text. This text will be printed and referenced by the index.

---

2 Extensions

Section author: Stephan Linz <linz@li-pro.net>

2.1 Spelling Checker

PyPI Package https://pypi.org/project/sphinxcontrib-spelling/

Documentation https://sphinxcontrib-spelling.readthedocs.io/

Git Repository https://github.com/sphinx-contrib/spelling

Spelling checker for *Sphinx*. It uses *PyEnchant* to produce a report showing misspelled words.

Features

1. Supports multiple source languages using the standard enchant dictionaries.
2. Supports project-specific dictionaries for localized jargon and other terminology that may not appear in the global dictionaries.
3. Suggests alternatives to words not found in the dictionary, when possible.

It consists:

- sphinxcontrib.spelling: spelling checker for *Sphinx*

2.1.1 Private Dictionaries

For more details, see Configuration Options section *Private Dictionaries*.

`.. spelling::`

The `.. spelling::` directive can be used to create a list of words known to be spelled correctly within a single file. For example, if a document refers to a person or project by name, the name can be added to the list of known words for just that single document.

When a more common list of words is needed, related to check multiple document at once, the *`spelling_word_list_filename`* (page 83) variable should be set properly.

`spelling_word_list_filename`

That is a list specifying files containing a list of words known to be spelled correctly but that do not appear in the referred language dictionary. The files should contain one word per line. Refer to the *PyEnchant* tutorial for details.

2.2 BibTeX Citations

PyPI Package https://pypi.org/project/sphinxcontrib-bibtex/

Documentation https://sphinxcontrib-bibtex.readthedocs.org/

Git Repository https://github.com/mcmtroffaes/sphinxcontrib-bibtex

Allowing *BibTeX* citations to be inserted into documentation via a `.. bibliography::` directive, and a `:cite:` role, which work similarly to *LaTeX*'s `\begin{thebibliography} ... \end{thebibliography}` environment and `\cite{cite_key}` command. For formatting, the extension relies on *Pybtex*. It consists:

- `sphinxcontrib.bibtex`: *Sphinx* interface
- `sphinxcontrib.bibtex.roles`: Doctree roles
- `sphinxcontrib.bibtex.nodes`: Doctree nodes
- `sphinxcontrib.bibtex.directives`: Doctree directives
- `sphinxcontrib.bibtex.transforms`: Doctree transforms
- `sphinxcontrib.bibtex.cache`: Cached information

Create a citation to a bibliographic entry.

Todo: In the case of the *LaTeX*/*PDF* builder the usage of `:ref:`bibliography`` leads to an **invalid and unresolved reference**.

`:cite:`

For more details, see `cite` role.

The example

```
See :cite:`sweigart2020automate` for an practical guide in Python.
```

Which gives See [Swe19] for an practical guide in Python.

For this sample you will need a corresponding bibliography for all cited references.

`.. bibliography::`

For more details, see `bibliography` directive.

The example

```
.. bibliography:: bibtex/example.bib
   :style: ldspalpha
   :encoding: utf-8
   :all:
```

Which gives All entries are placed in the central document bibliography list, mostly on the end of the document.

Which needs The example above processed the following *BibTeX* file content:

Listing 2.1: BibTeX example file (bibtex/example.bib)

```
@book{ sweigart2020automate,
    author = { Sweigart, Al },
    title = {
        Automate the boring stuff with Python :
        practical programming for total beginners
    },
    edition = { 2. },
    publisher = { No Starch Press },
    address = { San Francisco, United States of America },
    year = { 2019 },
    language = { English },
    type = { Paperback or Hardcover) },
    isbn = { 1593279922 },
    isbn10 = { 1-59327-992-2 },
    isbn13 = { 978-1593279929 },
    oclc = { 1140126955 },
    url = { https://www.amazon.com/dp/1593279922 },
    urldate = { September 2020 },
}
```

2.3 LinuxDoc

Documentation https://return42.github.io/linuxdoc/

Git Repository https://github.com/return42/linuxdoc

The LinuxDoc library with extensions of the *Linux*-Kernel documentation, you can use these extensions in common *Sphinx* projects. It consists:

- `linuxdoc.rstFlatTable`: the `.. flat-table::` reST-directive
- `linuxdoc.rstKernelDoc`: the `.. kernel-doc::` reST-directive
- `linuxdoc.kernel_include`: the `.. kernel-include::` reST-directive
- `linuxdoc.manKernelDoc`: the **`kernel-doc-man`** builder
- `linuxdoc.cdomain`: replacement for the sphinx C-domain
- `linuxdoc.kfigure`: implements scalable image handling

2.3.1 Flat list table

`.. flat-table::`

See also:

About tables: flat-table

The `.. flat-table::` (FlatTable) is a double-stage list similar to the `.. list-table::` with some additional features:

- *column-span*: with the role `:cspan:`num`` a cell can be extended through additional columns
- *row-span*: with the role `:rspan:`num`` a cell can be extended through additional rows
- *auto-span*: rightmost cell of a table row over the missing cells on the right side of that table-row. With Option `:fill-cells:` this behavior can changed from auto span to auto fill, which automatically inserts (empty) cells instead of spanning the last cell.

Options

`:header-rows: (integer)`
count of header rows

`:stub-columns: (integer)`
count of stub columns

`:widths: (list of integer)`
widths of columns

:fill-cells:
instead of auto-span missing cells, insert missing cells

Roles

:cspan:
(integer): additional columns (*morecols*)

:rspan:
(integer): additional rows (*morerows*)

The example below shows how to use this markup. The first level of the staged list is the *table-row*. In the *table-row* there is only one markup allowed, the list of the cells in this *table-row*. Exception are *comments* (`..`) and *targets* (e.g. a ref to *row 2 of table's body* (page 88)).

the example

> **Attention: line 2:** The option `:class: longtable` will not interpreted from directive `.. flat-table::` and has no effects.

```
.. flat-table:: LinuxDoc :rst:`.. flat-table::` example (table title)
   :class: longtable
   :widths: 15 15 15 15 40
   :header-rows: 2
   :stub-columns: 1

   * - :rspan:`1` head / stub
     - :cspan:`3` head 1.1-4

   * - head 2.1
     - head 2.2
     - head 2.3
     - head 2.4

   * .. row body 1 / this is a comment

     - row 1
     - :rspan:`2` cell 1-3.1
     - cell 1.2
     - cell 1.3
     - cell 1.4

   * .. Comments and targets are allowed on *table-row* stage.
     .. _`row body 2`:

     - row 2
     - cell 2.2
```

(continues on next page)

(continued from previous page)

```
    - :rspan:`1` :cspan:`1`
      cell 2.3 with a span over

      * col 3-4 &
      * row 2-3

  * - row 3
    - cell 3.2

  * - row 4
    - cell 4.1
    - cell 4.2
    - cell 4.3
    - cell 4.4

  * - row 5
    - cell 5.1 with automatic span to right end

  * - row 6
    - cell 6.1
    - .. empty cell 6.2 with automatic span to right end
```

which gives

Table 2.1: LinuxDoc `.. flat-table::` example (table title)

head / stub	**head 1.1-4**			
	head 2.1	**head 2.2**	**head 2.3**	**head 2.4**
row 1	cell 1-3.1	cell 1.2	cell 1.3	cell 1.4
row 2		cell 2.2	cell 2.3 with a span over • col 3-4 & • row 2-3	
row 3		cell 3.2		
row 4	cell 4.1	cell 4.2	cell 4.3	cell 4.4
row 5	cell 5.1 with automatic span to right end			
row 6	cell 6.1			

2.4 Program Output

PyPI Package https://pypi.org/project/sphinxcontrib-programoutput/

Documentation https://sphinxcontrib-programoutput.readthedocs.org/

Git Repository https://github.com/NextThought/sphinxcontrib-programoutput

Literally insert the output of arbitrary commands into documents, helping you to keep your command examples up to date. It consists:

- `sphinxcontrib.programoutput`: insert command output

2.4.1 Complete output

To include the output of a command into your document, use the `.. program-output::` directive provided by this extension.

`.. program-output::`

For more details, see `program-output` directive.

The example

```
.. program-output:: python --version
```

Which gives

```
Python 3.8.0
```

The whole output of `python --version`, including any messages on standard error, is inserted into the current document, formatted as literal text without any syntax highlighting. You can omit the content of the standard error stream with the `:nostderr:` option.

By default, commands are executed in the top-level source directory. You can choose an alternate working directory with the `:cwd:` option. The argument of this option is either a path relative to the current source file, or a absolute path which means that it is relative to the top level source directory.

2.4.2 Shortening the output

Lengthy output can be shortened with the `:ellipsis:` option. Its value denotes lines to omit when inserting the output of the command. Instead, a single ellipsis `...` is inserted.

the example If used with a single number, all lines after the specified line are omitted:

```
.. program-output:: python --help
   :ellipsis: 2
```

which gives The above omits all lines after the second one:

```
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
...
```

Negative numbers count from the last line backwards, thus replacing 2 with `-2` in the above example would only omit the last two lines.

the example If used with two comma-separated line numbers, all lines in between the specified lines are omitted. Again, a negative number counts from the last line backwards:

```
.. program-output:: python --help
   :caption: First two and last second lines from :command:`python --
↪help`
   :name: program-output-python-help
   :ellipsis: 2,-2
```

which gives The above omits all lines except the first two and the last two lines:

Listing 2.2: First two and last second lines from **`python --help`**

```
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
...
PYTHONDEVMODE: enable the development mode.
PYTHONPYCACHEPREFIX: root directory for bytecode cache (pyc) files.
```

2.4.3 Mimicking shell input

You can mimic shell input with the `.. command-output::` directive[1]. This directive inserts the command along with its output into the document.

`.. command-output::`

For more details, see `command-output` directive.

The example

```
.. command-output:: python --version
```

Which gives

```
$ python --version
Python 3.8.0
```

The appearance of this output can be configured with `programoutput_prompt_template`. When used in conjunction with `:ellipsis:`, the command itself and any additional text is *never* omitted.

[1] This directive is just an alias for the `.. program-output::` directive with the `:prompt:` option set.

`:ellipsis:` always refers to the *immediate output* of the command.

the example

```
.. command-output:: python --help
   :caption: First two lines from :command:`python --help` with prompt
   :name: command-output-python-help
   :ellipsis: 2
```

which gives

Listing 2.3: First two lines from **`python --help`** with prompt

```
$ python --help
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
...
```

2.4.4 Command execution and shell expansion

Normally the command is splittet according to the *POSIX* shell syntax (see `shlex`), and executed directly. Thus special shell features like expansion of environment variables will not work.

the example

```
.. command-output:: echo "$USER"
```

which gives

```
$ echo "$USER"
$USER
```

To enable these features, enable the `:shell:` option. With this option, the command is literally passed to the system shell.

the example

```
.. command-output:: echo "$USER"
   :shell:
```

which gives

```
$ echo "$USER"
```

Other shell features like process expansion consequently work, too.

the example

```
.. command-output:: ls -l $(which grep)
   :shell:
```

which gives

```
$ ls -l $(which grep)
-rwxr-xr-x 1 root root 219456 Sep 18  2019 /bin/grep
```

Warning: Remember to use `:shell:` carefully to avoid unintended interpretation of shell syntax and swallowing of fatal errors!

2.4.5 Error handling

If an unexpected exit code (also known as *return code*) is returned by a command, it is considered to have failed. In this case, a build warning is emitted to help you to detect misspelled commands or similar errors. By default, a command is expected to exit with an exit code of 0, all other codes indicate an error. In some cases however, it may be reasonable to demonstrate failed programs. To avoid a (superfluous) warning in such a case, you can specify the expected return code of a command with the `:returncode:` option.

the example

```
.. command-output:: python -c 'import sys, platform; print(sys.
↪version); sys.exit(1)'
   :returncode: 1
```

which gives

```
$ python -c 'import sys, platform; print(sys.version); sys.exit(1)'
3.8.0 (default, Jan 24 2020, 02:13:43)
[GCC 7.4.0]
```

The above command returns the exit code 1 (as given to `sys.exit()`), but no warning will be emitted. On the contrary, a warning will be emitted, should the command return 0!

Note: Upon fatal errors which even prevent the execution of the command neither return code nor command output are available. In this case an error message is inserted into the document instead.

If `:shell:` is set however, most of these fatal errors are handled by the system shell and turned into return codes instead. In this case the error message will only appear in the output of the shell. If you're using `:shell:`, double-check the output for errors. Best avoid `:shell:`, if possible.

2.5 Mathematical Plots

Attention: Matplotlib does not support labels and auto-references. You can not refer to a equation and you will never see an entry in the list of equations to `.. mathmpl::` expressions.

Only practicable and usable for *HTML*/*EPUB* and *LaTeX*/*PDF* builder.

Extension not applicable for textual output

The intended use of this *Sphinx* extension is for output systems with graphical support, not for pure text based systems like man or info pages. That is why many of the rendering functions are implemented exclusively for *HTML* or *LaTeX* output.

PyPI Package https://pypi.org/project/matplotlib/

Project Home https://matplotlib.org/

Documentation https://matplotlib.org/contents.html

Git Repository https://github.com/matplotlib/matplotlib

Documentation https://matplotlib.org/sampledoc/index.html

Git Repository https://github.com/matplotlib/sampledoc

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in *Python*. It consists:

- `matplotlib.sphinxext.mathmpl`: Matplotlib math-text in a *Sphinx* document
- `matplotlib.sphinxext.plot_directive`: Matplotlib plot in a *Sphinx* document

Runtime setup

`plot_pre_code`

Code that should be executed before each plot runs will setup over the *Sphinx* configuration option *`plot_pre_code`* (page 93). If not specified or **None or empty** it will default to a string containing: `import numpy as np; from matplotlib import pyplot as plt;`.

plot_pre_code Currently set to:

```
from matplotlib import pyplot as plt
from matplotlib import image as mpimg
from matplotlib import cm
import numpy as np
```

plot_working_directory

By default, if **None or empty**, the *plot_working_directory* (page 93) will be changed to the directory of the example, so the code can get at its data files, if any. Also its path will be added to `sys.path` so it can import any helper modules sitting beside it. This configuration option can be used to specify a central directory (also added to `sys.path`) where data files and helper modules for all code are located.

plot_working_directory Currently set to `_images/mplplots`.

plot_basedir

When a path to a source file is given, the *Sphinx* configuration option *plot_basedir* (page 94) will respect. It is the base directory, to which `.. plot::` file names are relative to. If **None or empty**, file names are **relative** to the directory where the file containing the directive is.

plot_basedir Currently set to `_images/mplplots`.

2.5.1 Expressions

See the Writing mathematical expressions for lots more information how to writing mathematical expressions in matplotlib.

With matplotlib in *Sphinx* you can include inline math $(\alpha^{ic} > \beta_{ic})$ (as role `:mathmpl:`(\alpha^{ic} > \beta_{ic})``) or display math:

$$\sum_{i=0}^{\infty} x_i \tag{2.1}$$

.. mathmpl::

The example

```
.. mathmpl::

   \left(\frac{5 - \frac{1}{x}}{4}\right)
```

Which gives

$$\left(\frac{5 - \frac{1}{x}}{4}\right) \tag{2.2}$$

2.5.2 Plots

.. plot::

See the matplotlib Pyplot tutorial and the Gallery for lots of examples of matplotlib plots.

The source code for the plot may be included in one of three ways:

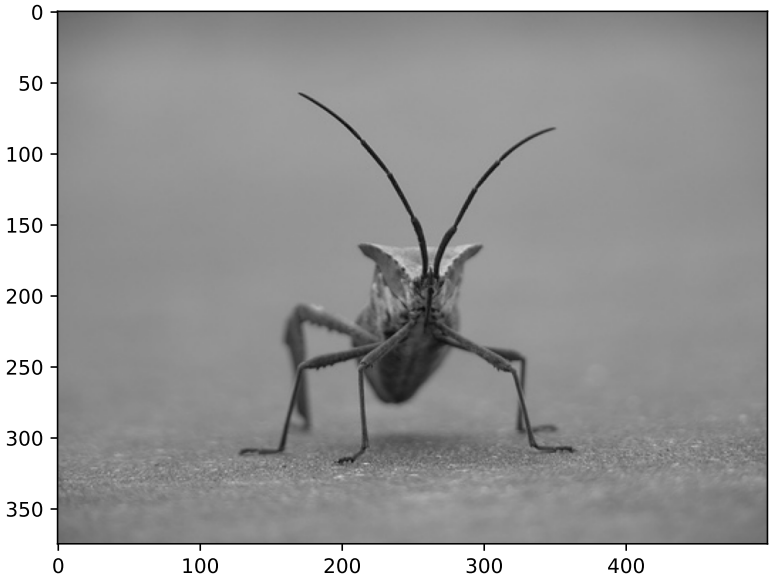
inline content

the example

```
.. plot::
   :align: center
   :scale: 75

   img = mpimg.imread('https://github.com/matplotlib/matplotlib'
                      + '/raw/master/doc/_static/stinkbug.png')
   imgplot = plt.imshow(img)
```

which gives

```
img = mpimg.imread('https://github.com/matplotlib/matplotlib'
                   + '/raw/master/doc/_static/stinkbug.png')
imgplot = plt.imshow(img)
```



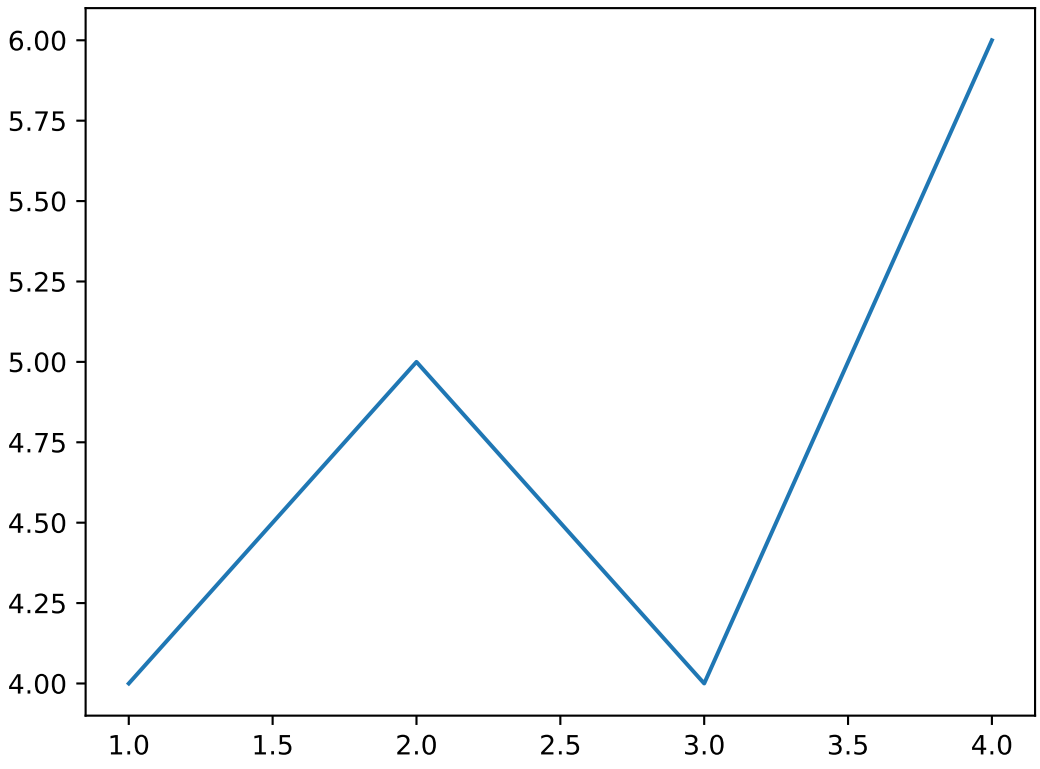
doctest content

the example

```
.. plot::
   :format: doctest
   :align: center
   :scale: 100

   >>> import matplotlib.pyplot as plt
   >>> plt.plot([1, 2, 3, 4], [4, 5, 4, 6])  # doctest: +ELLIPSIS
   [<matplotlib.lines.Line2D object at 0x...>]
```

which gives

```
>>> import matplotlib.pyplot as plt
>>> plt.plot([1, 2, 3, 4], [4, 5, 4, 6])
[<matplotlib.lines.Line2D object at 0x...>]
```



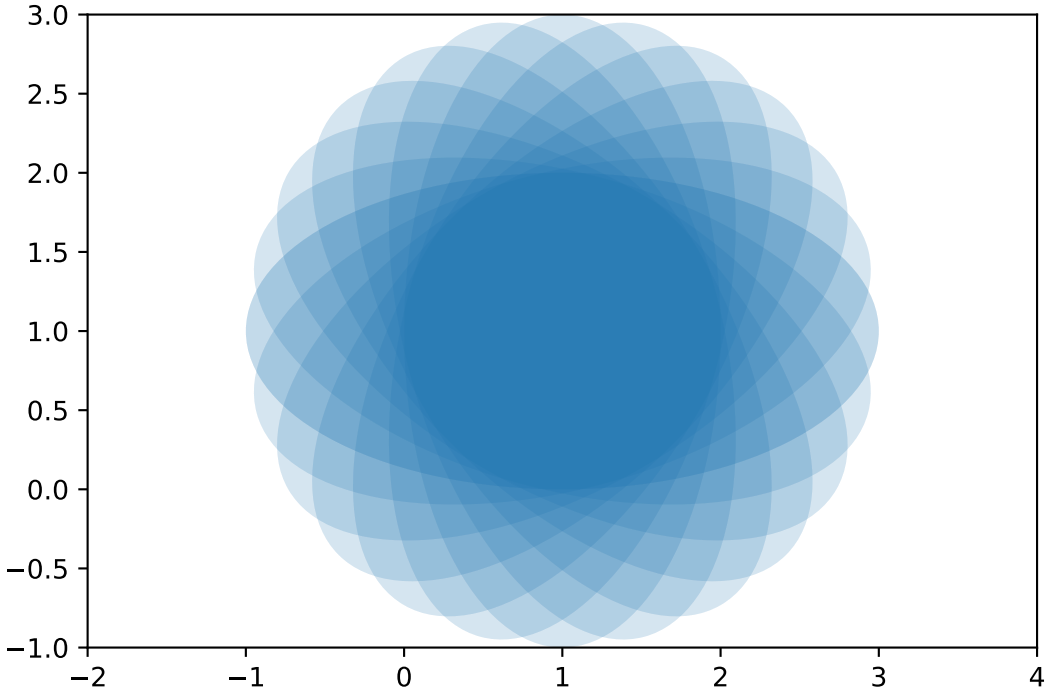
source file content

the example

```
.. plot:: ellipses.py
   :include-source:
   :encoding: utf
   :format: python
   :align: center
   :scale: 100
```

which gives

```
from pylab import *
from matplotlib.patches import Ellipse

delta = 15.0 # degrees

angles = arange(0, 360 + delta, delta)
ells = [Ellipse((1, 1), 4, 2, a) for a in angles]

a = subplot(111, aspect = 'equal')

for e in ells:

    e.set_clip_box(a.bbox)
    e.set_alpha(0.1)

    a.add_artist(e)

xlim(-2, 4)
ylim(-1, 3)

show()
```


3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2
3
4

2.5.2.1 3D-Plots

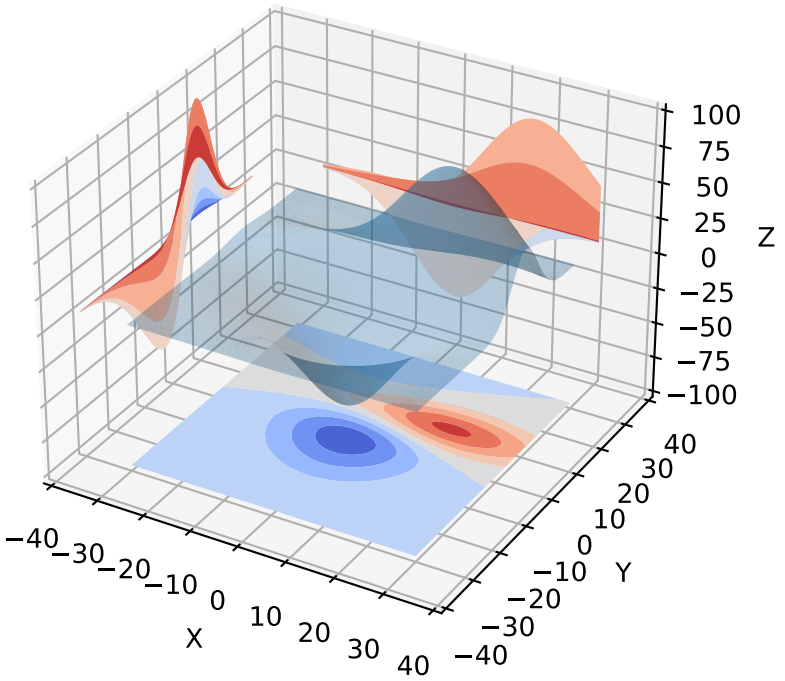
See mplot3d, mplot3d FAQ, and mplot3d API.

the example

```
.. plot::
   :format: python
   :align: center
   :scale: 100

   from mpl_toolkits.mplot3d import axes3d

   fig = plt.figure()
   ax = fig.gca(projection='3d')
   X, Y, Z = axes3d.get_test_data(0.005)
   ax.plot_surface(X, Y, Z, rstride=8, cstride=8, alpha=0.3)
   cset = ax.contourf(X, Y, Z, zdir='z', offset=-100, cmap=cm.coolwarm)
   cset = ax.contourf(X, Y, Z, zdir='x', offset=-40, cmap=cm.coolwarm)
   cset = ax.contourf(X, Y, Z, zdir='y', offset=40, cmap=cm.coolwarm)

   ax.set_xlabel('X'); ax.set_xlim(-40, 40)
   ax.set_ylabel('Y'); ax.set_ylim(-40, 40)
   ax.set_zlabel('Z'); ax.set_zlim(-100, 100)

   plt.show()
```

which gives

```
from mpl_toolkits.mplot3d import axes3d

fig = plt.figure()
ax = fig.gca(projection='3d')
X, Y, Z = axes3d.get_test_data(0.005)
ax.plot_surface(X, Y, Z, rstride=8, cstride=8, alpha=0.3)
cset = ax.contourf(X, Y, Z, zdir='z', offset=-100, cmap=cm.coolwarm)
cset = ax.contourf(X, Y, Z, zdir='x', offset=-40, cmap=cm.coolwarm)
cset = ax.contourf(X, Y, Z, zdir='y', offset=40, cmap=cm.coolwarm)

ax.set_xlabel('X'); ax.set_xlim(-40, 40)
ax.set_ylabel('Y'); ax.set_ylim(-40, 40)
ax.set_zlabel('Z'); ax.set_zlim(-100, 100)

plt.show()
```


100
75
50
25
0
−25
−50
−75
−100
Z
40
30
20
10
0
−10
−20
−30
−40
Y
−40
−30
−20
−10
0
10
20
30
40
X

2.6 PGF/TikZ LaTeX Pictures

> **Attention:** Only practicable and usable for *HTML* and *LaTeX* builder.

PyPI Package https://pypi.org/project/sphinxcontrib-tikz/

Documentation http://sphinxcontrib-tikz.readthedocs.io/

Git Repository https://bitbucket.org/philexander/tikz

Sphinx extension, which enables the use of the *PGF/TikZ LaTeX* package to draw nice pictures.

This extension relies on two software packages being installed on your computer:

1. `latex` with the `tikz` and the `amsmath` packages
2. A software package that is able to convert a *PDF* to an image. Currently, four different ways of doing this conversion are supported, called conversion "suites". Below is a list for each suite what must be installed on your computer. Only one such suite need to be installed:
   - *pdf2svg* suite: `pdf2svg` (preferred, default)
   - *Netpbm* suite: `pdftoppm` (part of the *Poppler PDF* library) and `pnmtopng` (part of the *Netpbm* package)
   - *ImageMagick* suite: `pdftoppm` (part of the *Poppler PDF* library) and `convert` (part of the *ImageMagick* package)
   - *GhostScript* suite: `ghostscript`

See Configuration in the extension documentation for more details.

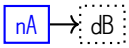
:tikz:

For more details, see Usage in the extension documentation.

inline content

The example

```
An example role :tikz:`[thick] \node[blue,draw] (a) {nA};
\node[draw,dotted,right of=a] {dB} edge[<-] (a);`
```

Which gives An example role 

.. tikz::

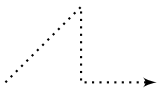
For more details, see Usage in the extension documentation.

explicit markup

The example

```
.. rst-class:: centered
.. tikz:: [dotted,thick,>=latex'] \draw[->] (0,0) -- (1,1) -- (1,0)
   -- (2,0);
   :libs: arrows
```

Which gives



from source file

The example

```
.. _tikz/srcfile/ctrloop:
.. rst-class:: centered
.. tikz:: Control system principles (PGF/TikZ example)
   :include: /extensions/tikz/srcfile/ctrloop.tex
   :libs: arrows,shapes
```

Which gives

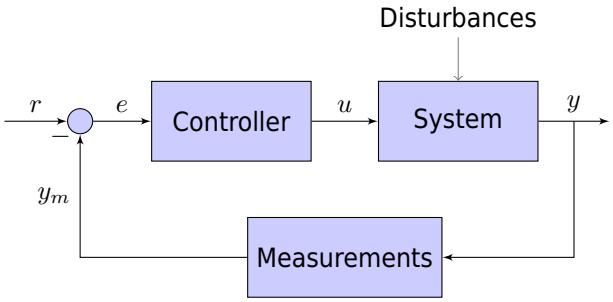


Figure 2.1: Control system principles (PGF/TikZ example)

Which needs The example above comes from the Control system principles web page and processed the following TikZ file content:

Listing 2.4: TikZ example file (tikz/srcfile/ctrloop.tex)

```
\begin{tikzpicture}[
    auto,
    node distance = 2cm,
    >=latex'
]

    \tikzstyle{block} = [
        draw,
        rectangle,
        fill = blue!20,
        minimum height = 3em,
        minimum width = 6em
    ]
    \tikzstyle{sum} = [
        draw,
        circle,
        fill = blue!20,
        node distance = 1cm
    ]
```

(continues on next page)

(continued from previous page)

```
    \tikzstyle{input} = [
        coordinate
    ]
    \tikzstyle{output} = [
        coordinate
    ]
    \tikzstyle{pinstyle} = [
        pin edge={
            to-,
            thin,
            black
        }
    ]

    % placing the blocks
    \node [input, name=input] {};
    \node [sum, right of=input] (sum) {};
    \node [block, right of=sum] (controller) {Controller};
    \node [block, right of=controller,
           pin={[pinstyle]above:Disturbances},
           node distance=3cm] (system) {System};

    % draw an edge between the controller and system block
    % to calculate the coordinate -- need it to place the
    % measurement block
    \draw [->] (controller) -- node[name=u] {$u$} (system);
    \node [output, right of=system] (output) {};
    \node [block, below of=u] (measurements) {Measurements};
    % once the nodes are placed, connecting them is easy
    \draw [draw,->] (input) -- node {$r$} (sum);
    \draw [->] (sum) -- node {$e$} (controller);
    \draw [->] (system) -- node [name=y] {$y$}(output);
    \draw [->] (y) |- (measurements);
    \draw [->] (measurements) -|
        node[pos=0.99] {$-$}
        node [near end] {$y_m$} (sum);

\end{tikzpicture}
```

2.6.1 PGF/TikZ

Homepage PGF/TikZ Home (VisualTikZ Home)

CTAN Package PGF/TikZ Package (VisualTikZ Package)

Documentation PGF/TikZ Manual (VisualTikZ Manual)

Documentation PGF/TikZ Introduction by Jacques Crémer

Git Repository PGF/TikZ Master Branch

PGF is a macro package for creating graphics. It is platform- and format-independent and works together with the most important *TeX* backend drivers. It comes with a user-friendly syntax layer called *TikZ*. Its usage is similar to pstricks and the standard picture environment. *PGF* works with plain *TeX*, *LaTeX*, and ConTeXt. Unlike pstricks, it can produce either PS or *PDF* output.

Shapes

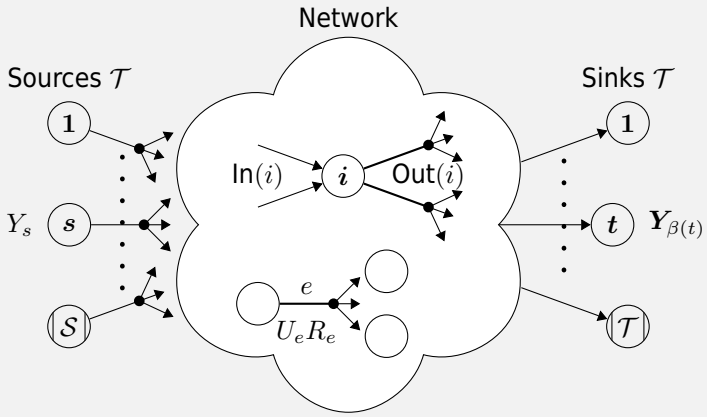


Figure 2.2: Shapes and symbols

`pgftikz/shapesyms.tex`[1], in Figure 2.2.

Indication of provenance: StackExchange (TeX): a/518945 (user194703, Dec 2 '19 at 15:56)

Positioning

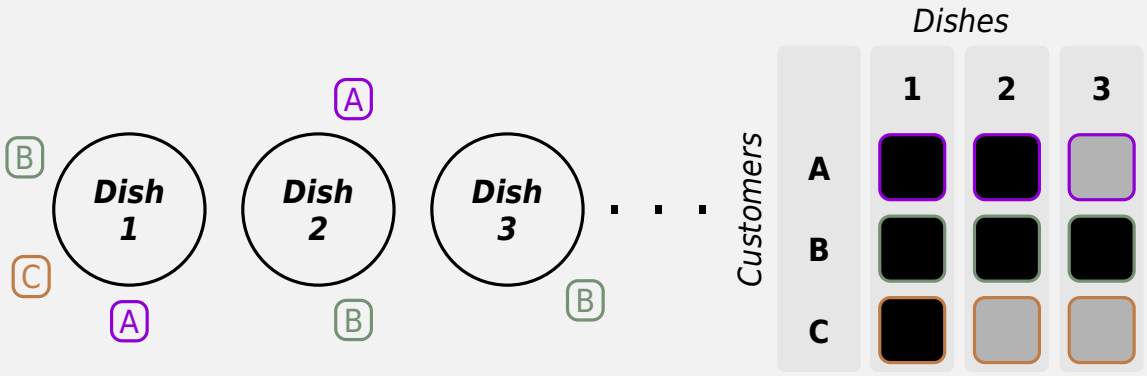


Figure 2.3: Shapes absolut and in a matrix positioned

`pgftikz/matrix.tex`[2], in Figure 2.3.

Indication of provenance: StackExchange (TeX): a/435856 (J Leon V., BSD, MIT, Beerware licences)

Construction

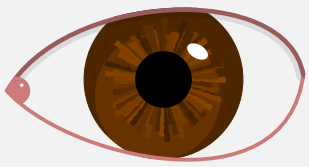


Figure 2.4: Drawing a constructive but realistic eye

`pgftikz/fancyeye.tex`[3], in Figure 2.4.

Indication of provenance: StackExchange (TeX): a/94087 (JLDiaz)

Fourier

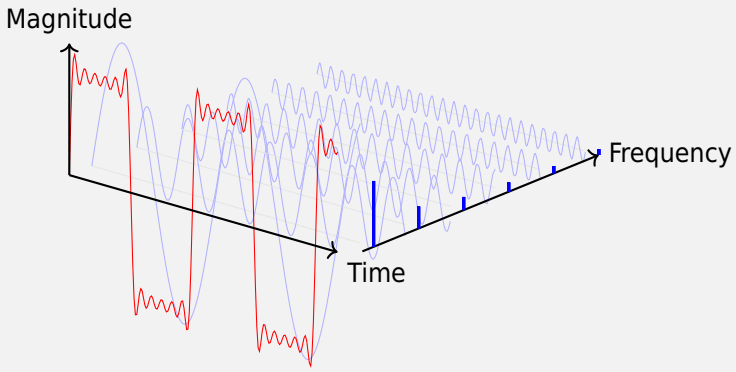


Figure 2.5: Time-frequency correspondence of the Fourier transformation

`pgftikz/fourier.tex`[4], in Figure 2.5.

Indication of provenance: StackExchange (TeX): a/127401 (Jake)

Interferences

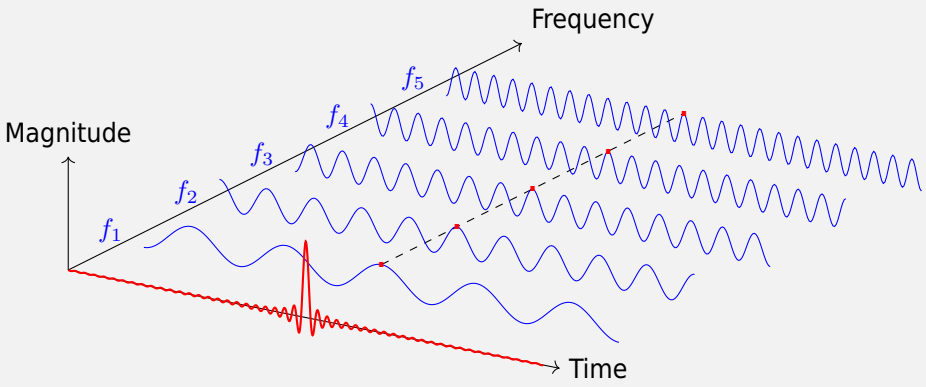


Figure 2.6: Show constructive interferences in the time domain

`pgftikz/interference.tex`[5], in Figure 2.6.

Indication of provenance: StackExchange (TeX): a/162263 (Thomas)

AM

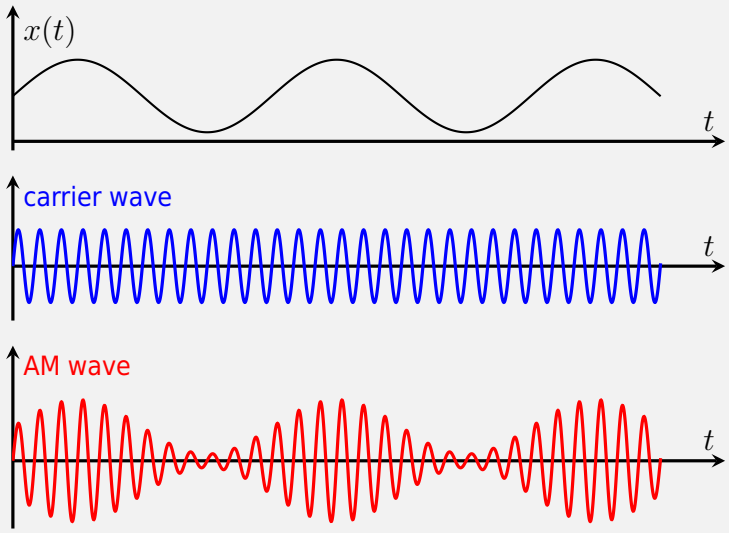


Figure 2.7: Graphical derivation of an amplitude modulated signal

`pgftikz/modula-am.tex`[6], in Figure 2.7.

Indication of provenance: Amplitude modulation (AM) (Petar Veličković, July 2016)

FM

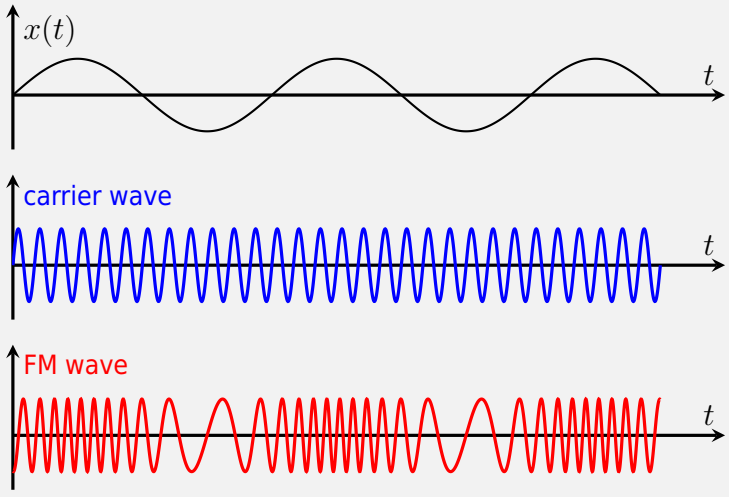


Figure 2.8: Graphical derivation of a frequency modulated signal

`pgftikz/modula-fm.tex`[7], in Figure 2.8.

Indication of provenance: Frequency modulation (FM) (Petar Veličković, July 2016)

Flowchart

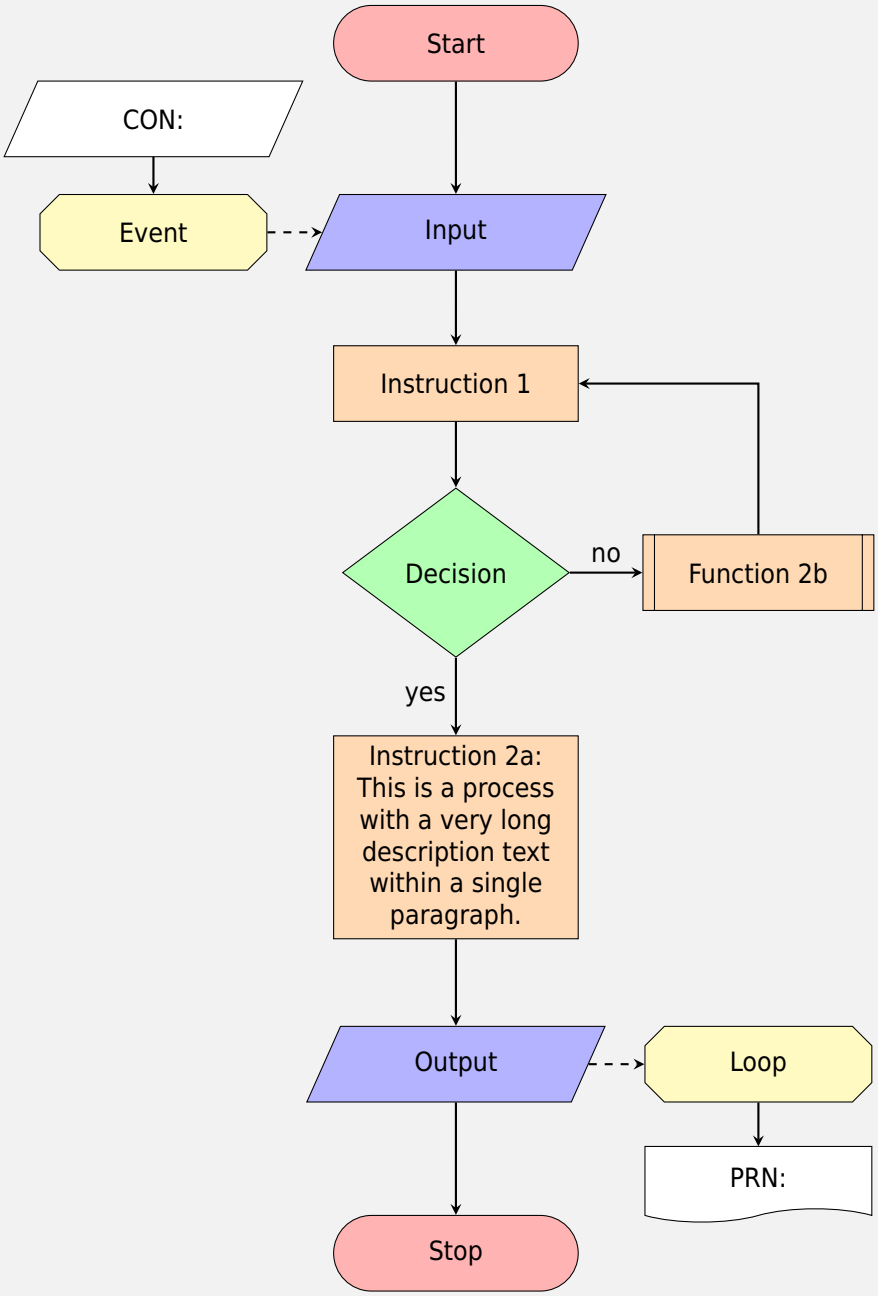


Figure 2.9: Geometrical shapes in a flowchart

`pgftikz/flowchart.tex`[8], in Figure 2.9.

Indication of provenance: TikZ Tutorial for Beginners (Part 3)—Creating Flowcharts (Josh Cassidy, August 2013)

2.6.2 CircuiTikZ

Homepage CircuiTikZ Home

CTAN Package CircuiTikZ Package

Documentation CircuiTikZ Manual

Git Repository CircuiTikZ Master Branch

CircuiTikZ provides a set of macros for naturally typesetting electrical and electronic networks, designed as a tool that is easy to use by native to a lean *LaTeX* syntax. It has therefore been based on the very impressive *PGF/TikZ* package.

Op-Amp stabilization

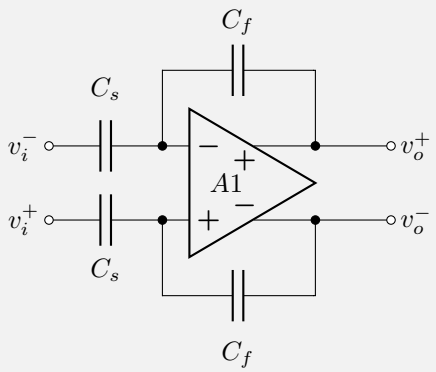


Figure 2.10: Full differential Op-Amp stabilization principles

`circuitikz/opamp-fullstab.tex`[1], in Figure 2.10.

Indication of provenance: StackExchange (TeX): q/82797 (Kit)

Inverting Op-Amp

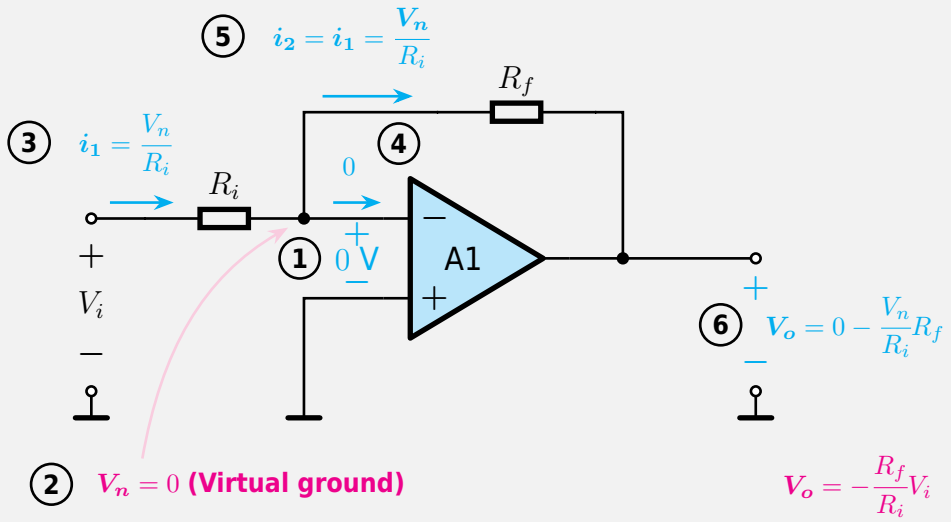


Figure 2.11: Detailed description of inverting Op-Amp principles

`circuitikz/opamp-inv.tex`[2], in Figure 2.11.

Indication of provenance: StackExchange (TeX): a/441787 (J Leon V., BSD, MIT, Beerware licences)

Schemed Structure

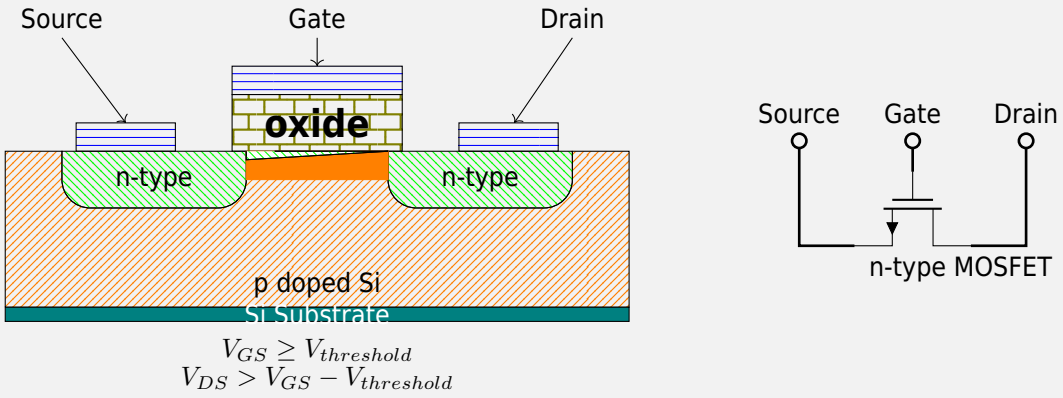


Figure 2.12: Drawing generic n-type MOSFET chip structure

`circuitikz/nmos-fet.tex`[3], in Figure 2.12.

Indication of provenance: http://wesleythoneycutt.com/drawing-mosfet-in-tikz/ (by Wesley T. Honeycutt)

Schematic

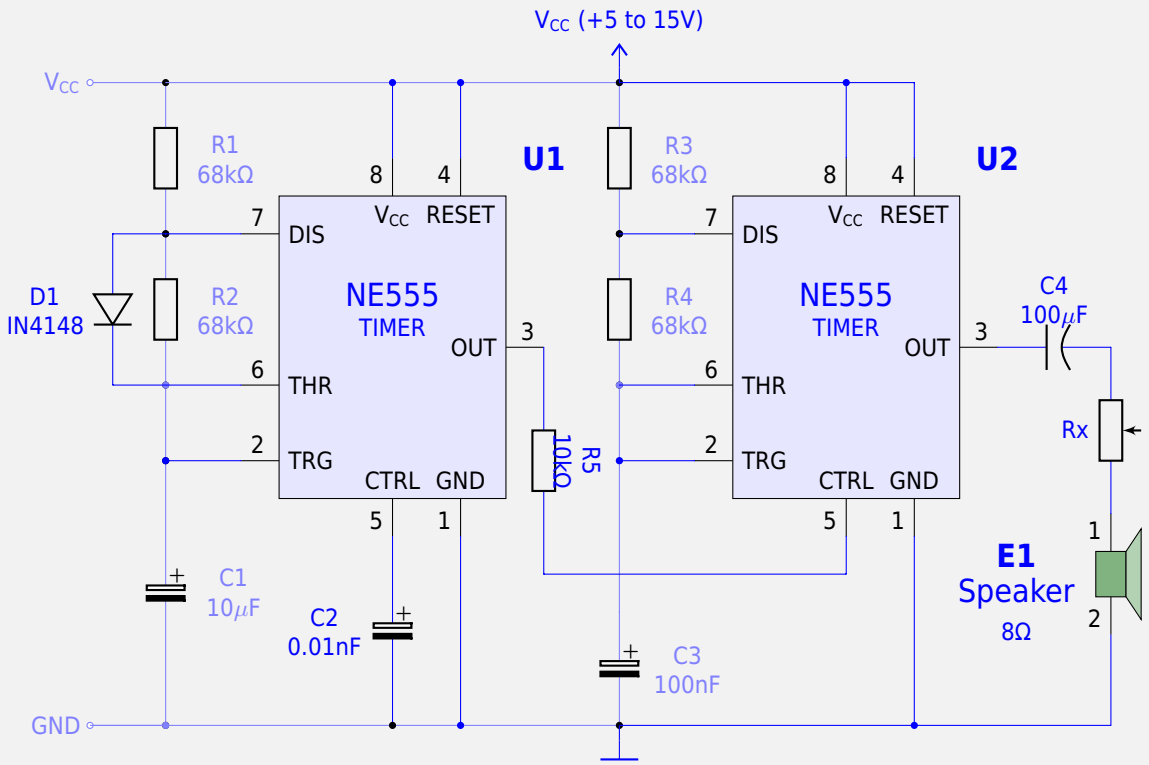


Figure 2.13: NE555 timer as “Dee-Dah” siren

`circuitikz/ne555-deedah.tex`[4], in Figure 2.13.

Indication of provenance: StackExchange (TeX): a/435123 (J Leon V., BSD, MIT, Beerware licences)

Breadboard

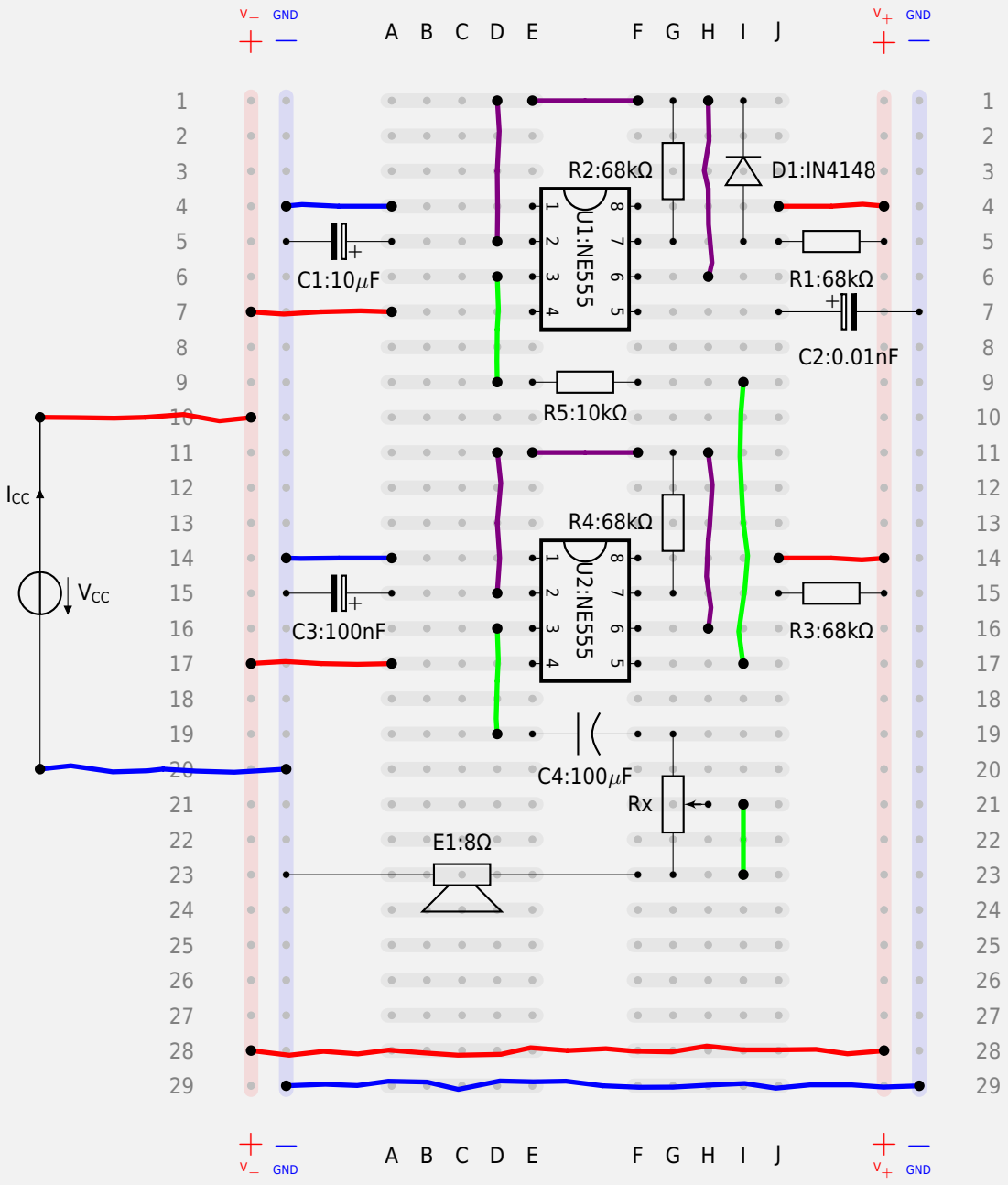


Figure 2.14: Breadboard with NE555 timer as “Dee-Dah” siren

`circuitikz/breadboard.tex`[5], in Figure 2.14.

Indication of provenance: StackExchange (TeX): q/493239 (Anshul Singhvi)

LQFP64 Package

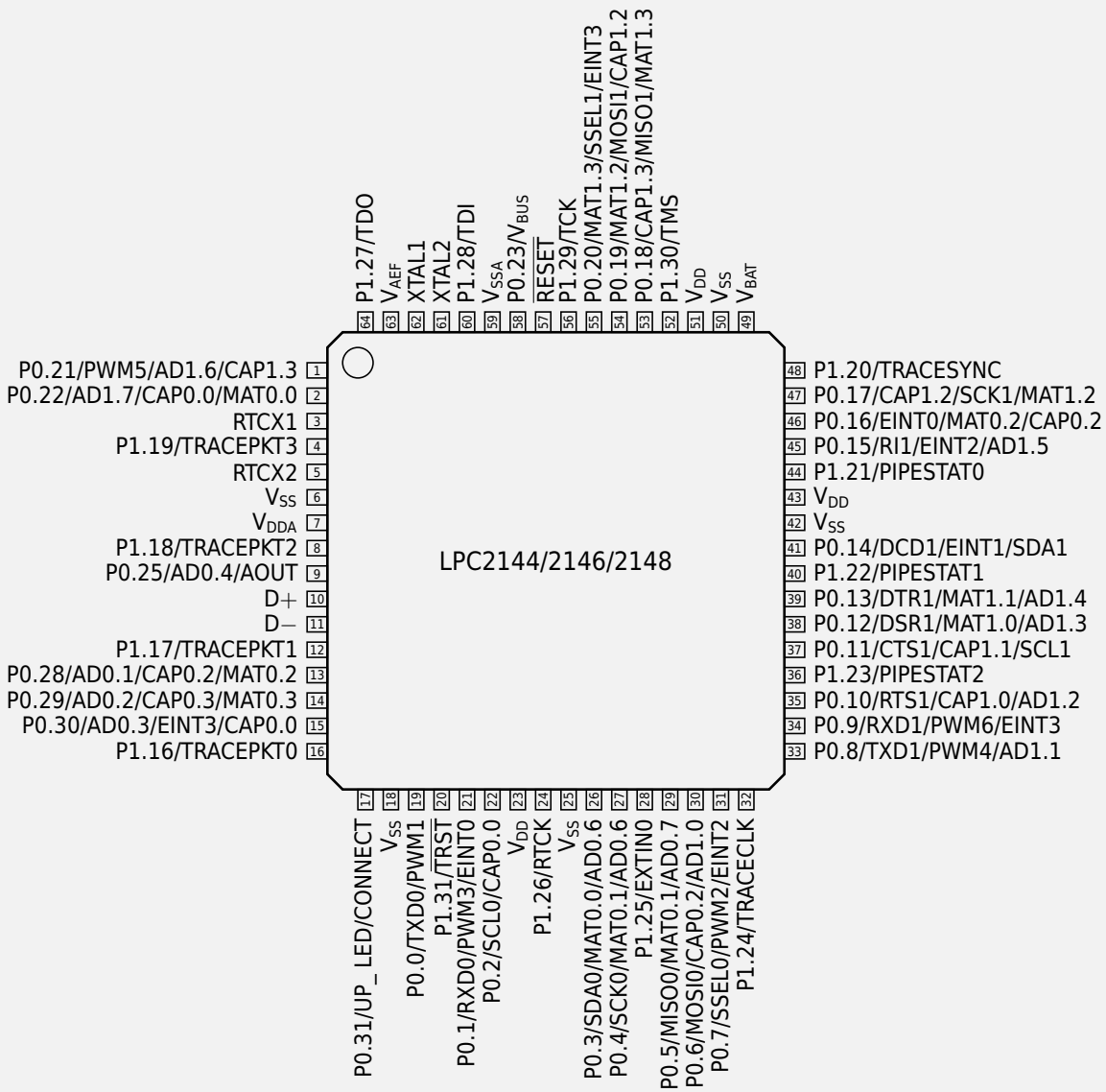


Figure 2.15: LPC2144/2146/2148 pin assignment on LQFP64

`circuitikz/lpc214x-lqfp64.tex`[6], in Figure 2.15.

Attention: Native support only since CircuiTikZ V0.9 (Oct 2019).
See also: CircuiTikZ change log

Indication of provenance: StackExchange (TeX): a/202449 (John Kormylo)

2.6.3 TikZ-Timing

CTAN Package TikZ-Timing Package

Documentation TikZ-Timing User Guide

Git Repository TikZ-Timing Master Branch

TikZ-Timing is a *TikZ* extension with macros and an environment to generate timing diagrams (digital waveforms) without much effort.

Simple time line

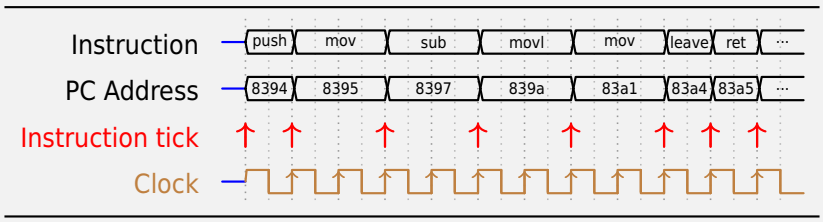


Figure 2.16: Program counter and instruction tick

`tikztiming/pcinst-tick.tex`[1], in Figure 2.16.

Indication of provenance: StackExchange (TeX): a/30906 (sdaau and Count Zero)

Divider-line

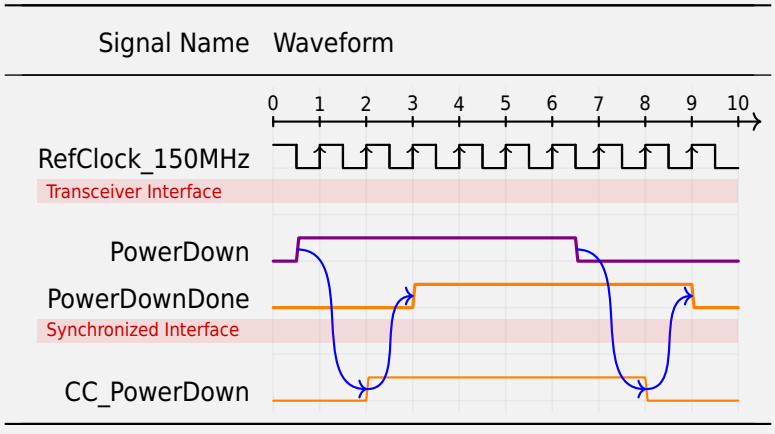


Figure 2.17: Signal interconnections and dividers

`tikztiming/divider-line.tex`[2], in Figure 2.17.

Indication of provenance: StackExchange (TeX): a/236091 (Symbol 1)

Colorized data cells

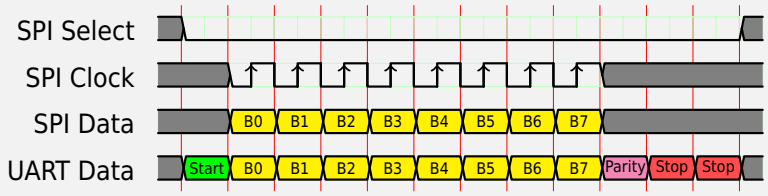


Figure 2.18: SPI vs. UART with arbitrary colored data cells

`tikztiming/colored-cells.tex`[3], in Figure 2.18.

Indication of provenance: StackExchange (TeX): a/255027 (Gonzalo Medina)

SPI operating modes

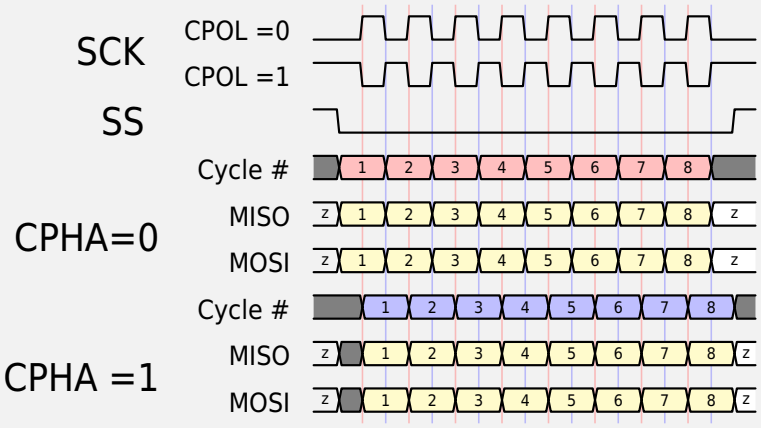


Figure 2.19: Serial Peripheral Interface operating modes

`tikztiming/spi-opmodes.tex`[4], in Figure 2.19.

Indication of provenance: StackExchange (TeX): a/290027 (Habi)

PCI timing diagrams

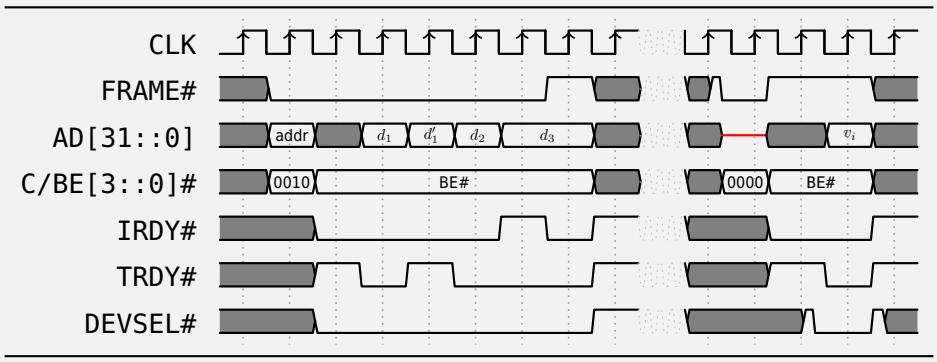


Figure 2.20: PCI Read and Interrupt Acknowledge

`tikztiming/pci-read-irqack.tex`, in Figure 2.20.
PCI timing diagrams with reference to version 2.2 of the PCI specification[5].

Indication of provenance: https://nathantypanski.com/blog/2014-10-29-tikz-timing.html

GN4124 Packet Decoder

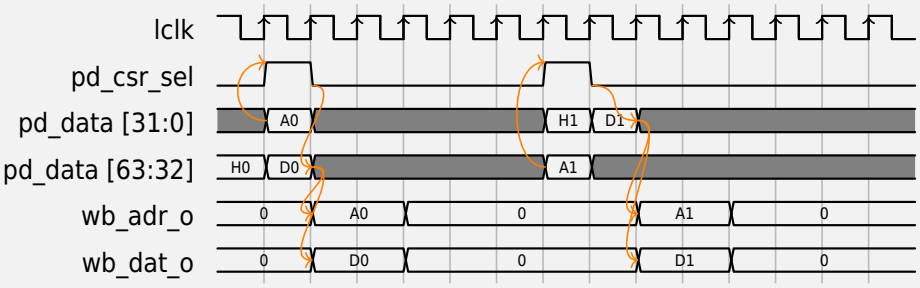


Figure 2.21: GN4124 Packet Decoder Write Request

`tikztiming/gn4124-pdwreq.tex`, in Figure 2.21.
Packet decoder timing diagram with reference to the Gennum GN4124 to Wishbone bridge user guide[6].

Indication of provenance: https://github.com/terpstra/gn4124-core/

2.6.4 TikZ Goodies

Homepage TikZ Goodies Project

TikZ Goodies is a small and simple collection of *TikZ* packages for drawing:

- **Timing-Diagrams**
- **Execution Stacks**
- **SystemC/TLM**

Timing-Diagrams

CTAN Package TikZ Goodies Timing-Diagrams Package

Documentation TikZ Goodies Timing-Diagrams README

Documentation TikZ Goodies Timing-Diagrams Example

Various kinds of timing diagrams (**not hardware-oriented**).

> **Attention:** You may want to have a look at *TikZ-Timing* (page 115) too if you want to draw some hardware-oriented timing diagrams.

Basics

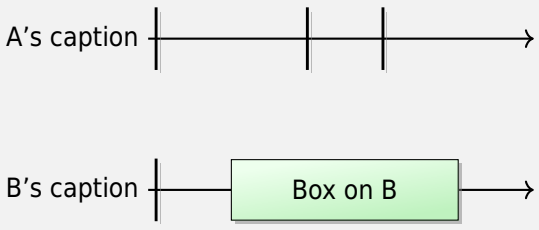


Figure 2.22: Basic Timing-Diagram example without additionals

`tikzgoodies/timing-diagrams/basic.tex`, in Figure 2.22.

Ticks & Tasks

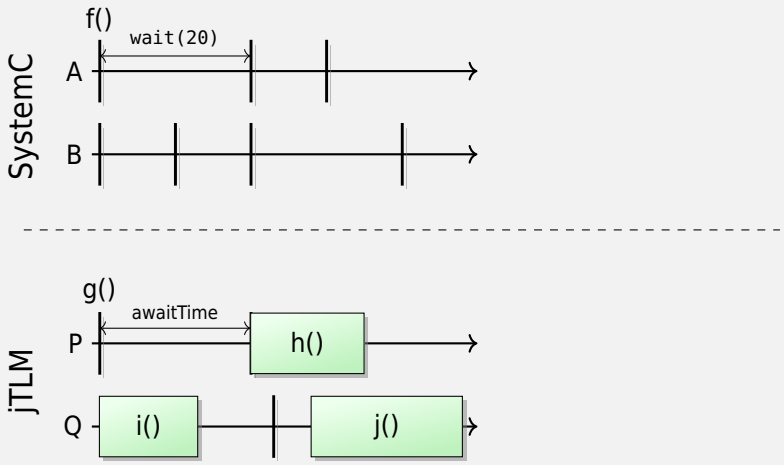


Figure 2.23: Timeline-Diagram example with ticks and tasks

`tikzgoodies/timing-diagrams/timeline.tex`, in Figure 2.23.

Annotations

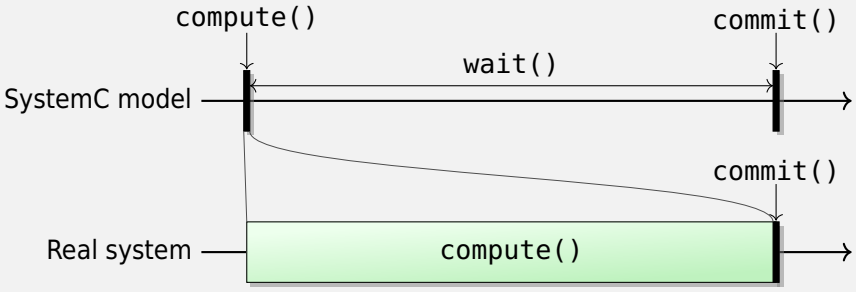


Figure 2.24: Timing-Diagram example with annotations

`tikzgoodies/timing-diagrams/annotations.tex`, in Figure 2.24.

Callouts

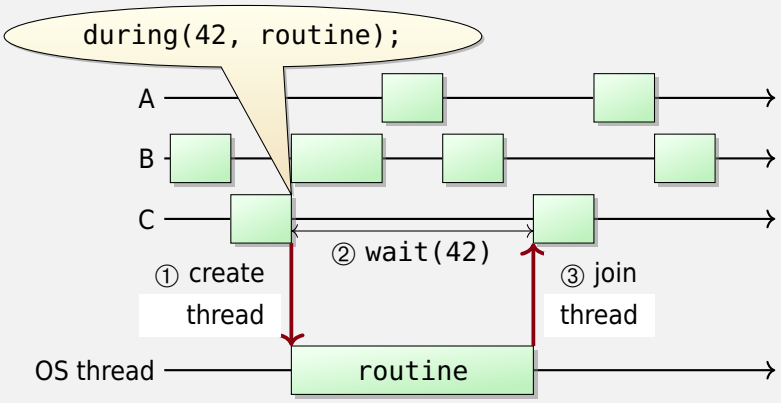


Figure 2.25: Timing-Diagram example with callouts (synchronizations)

`tikzgoodies/timing-diagrams/callouts.tex`, in Figure 2.25.

Arrows

Arrows (compact)

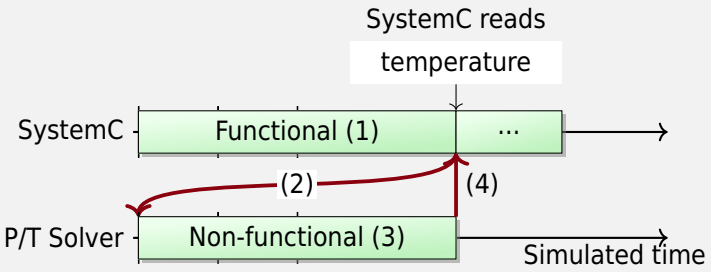


Figure 2.26: Timing-Diagram example with arrows (compact)

Arrows (stretched)

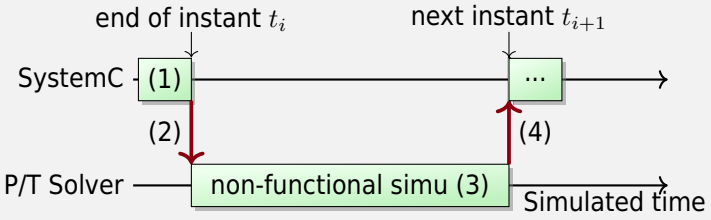


Figure 2.27: Timing-Diagram example with arrows (stretched)

`tikzgoodies/timing-diagrams/arrows-compact.tex`, in Figure 2.26.
`tikzgoodies/timing-diagrams/arrows-stretched.tex`, in Figure 2.27.

Events

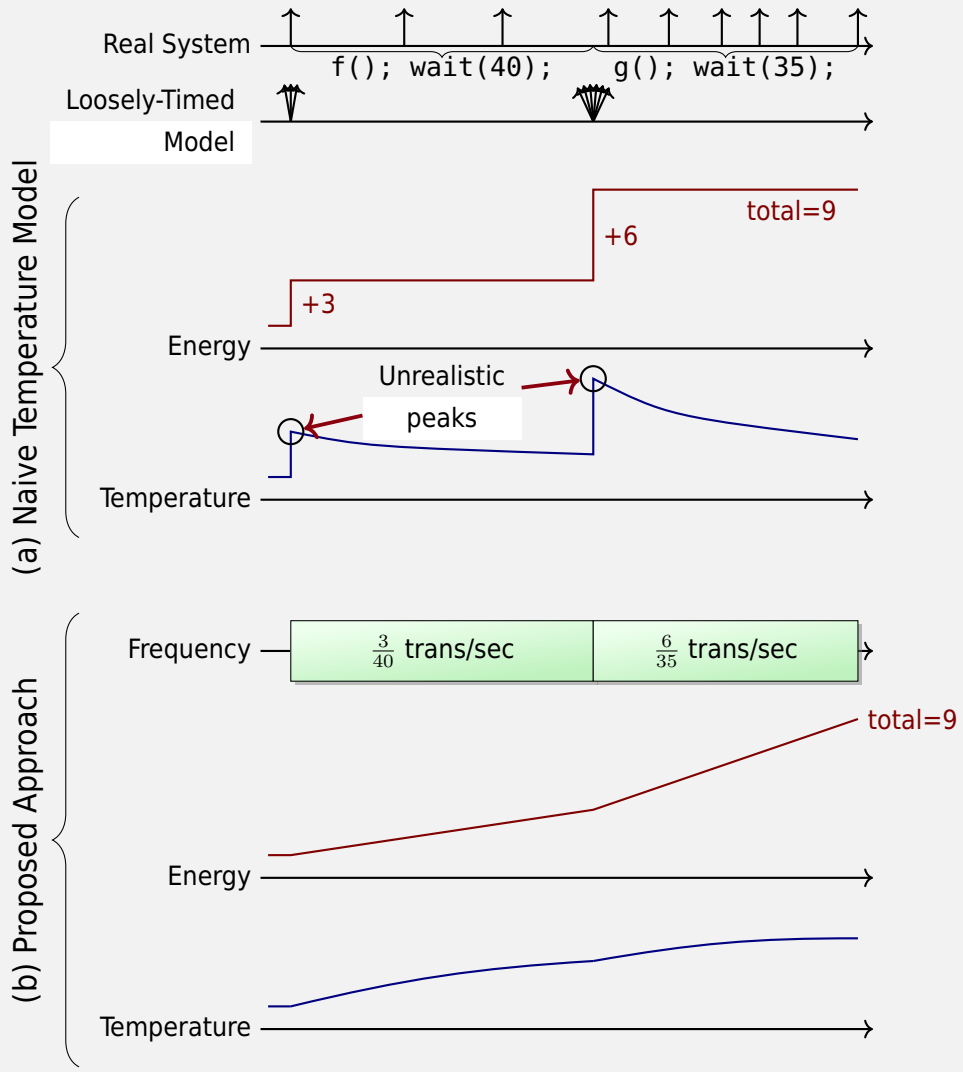


Figure 2.28: Timing-Diagram example with events as vertical arrows

`tikzgoodies/timing-diagrams/events.tex`, in Figure 2.28.

Execution Stacks

CTAN Package TikZ Goodies Drawstack Package

Documentation TikZ Goodies Drawstack README

Documentation TikZ Goodies Drawstack Example

Specific kind of execution stacks (*typically to illustrate assembly language notions*).

Basics

Minimal

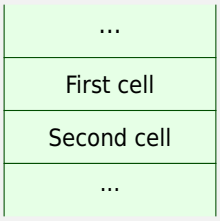


Figure 2.29: Minimalistic execution stack example

Frames

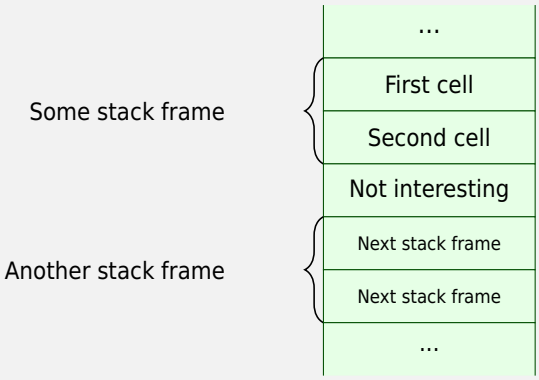


Figure 2.30: Execution stack example with grouped cells in frames

Padding

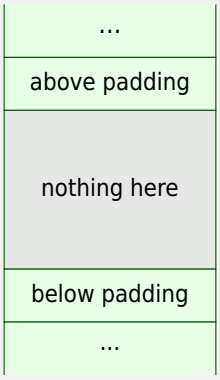


Figure 2.31: Execution stack example with cell padding

`tikzgoodies/drawstack/minimal.tex`, in Figure 2.29.
`tikzgoodies/drawstack/frames.tex`, in Figure 2.30.
`tikzgoodies/drawstack/padding.tex`, in Figure 2.31.

Pointers

Base Pointer

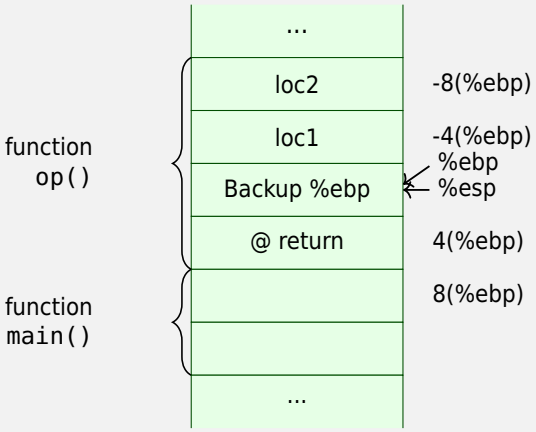


Figure 2.32: Execution stack example with base pointers

Stack Pointer

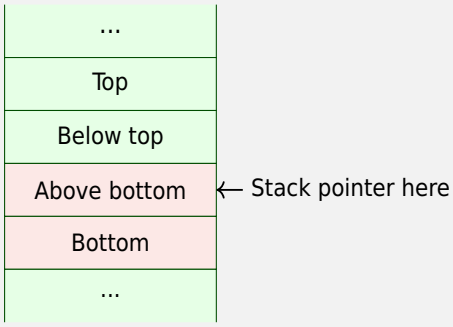


Figure 2.33: Execution stack example with stack pointers

`tikzgoodies/drawstack/base-pointer.tex`, in Figure 2.32.
`tikzgoodies/drawstack/stack-pointer.tex`, in Figure 2.33.

Memory

Structure

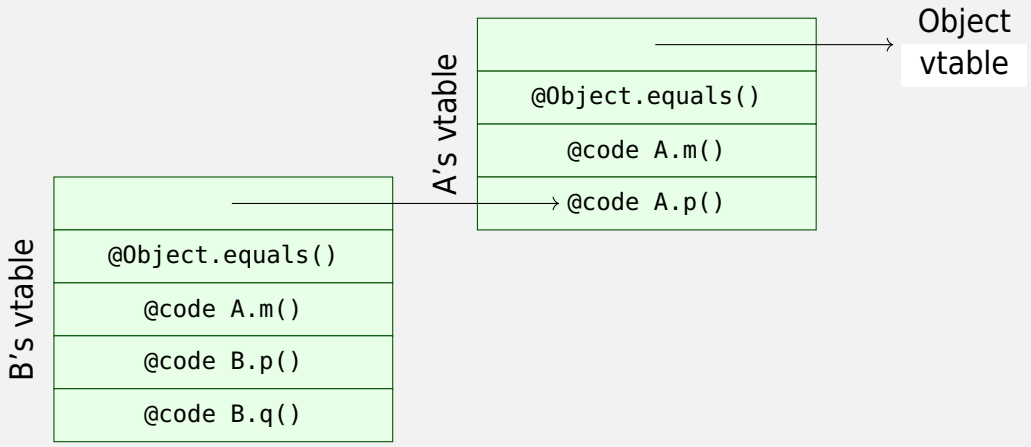


Figure 2.34: Structures without a stack structure

Stack Structure

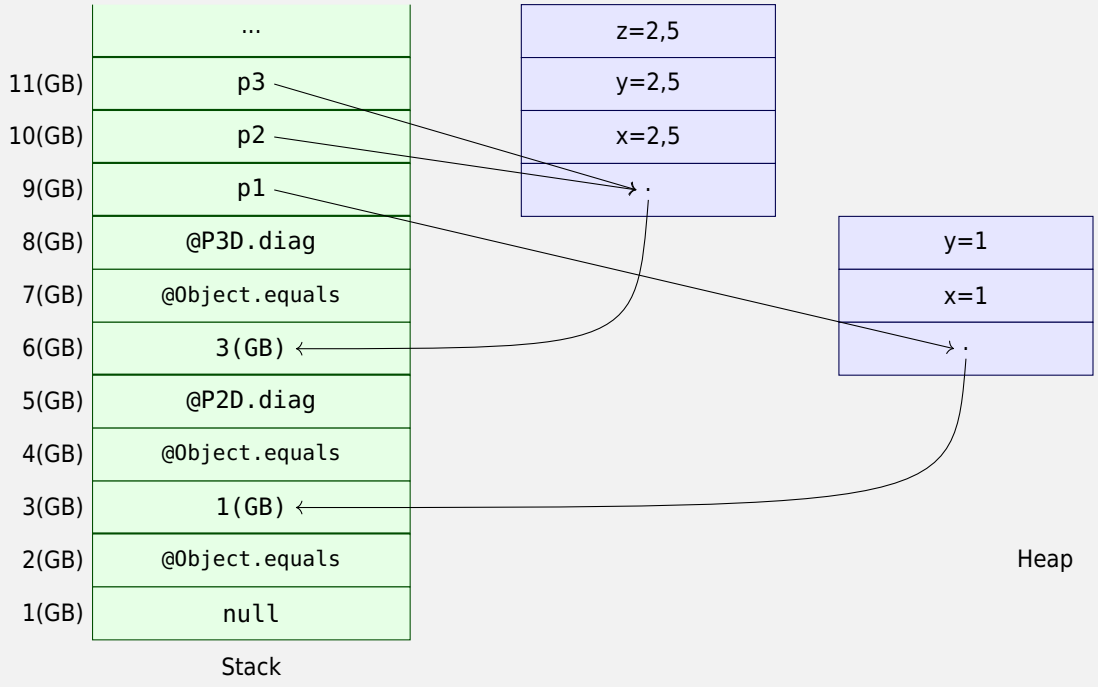


Figure 2.35: Structures and stack together

`tikzgoodies/drawstack/structure.tex`, in Figure 2.34.
`tikzgoodies/drawstack/stack-structure.tex`, in Figure 2.35.

Stylistics

Highlighting

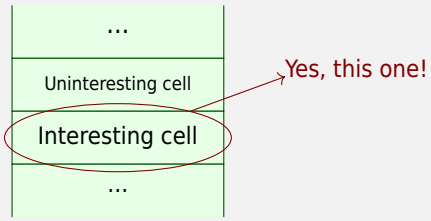


Figure 2.36: Execution stack example with highlighted note

Style

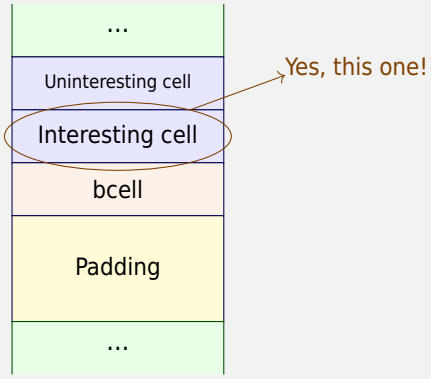


Figure 2.37: Execution stack example with changed style

`tikzgoodies/drawstack/highlighting.tex`, in Figure 2.36.
`tikzgoodies/drawstack/style.tex`, in Figure 2.37.

SystemC/TLM

Documentation TikZ Goodies TikzLM README

Documentation TikZ Goodies TikzLM Example (`tikzlm-example.pdf`)

Basic elements of typical SystemC (C++ classes for system-level modeling) and/or TLM (Transaction Level Modeling) usage.

Platform

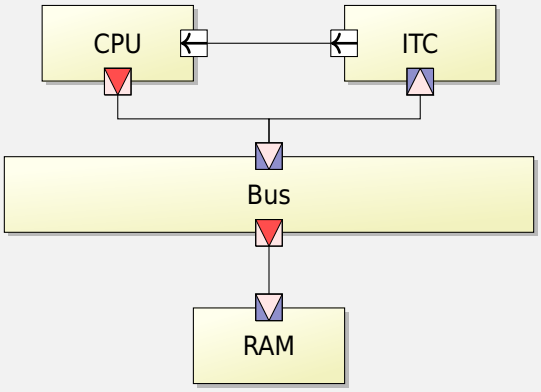


Figure 2.38: Platform show explicitly the sockets of the bus

`tikzgoodies/tikzlm/platform.tex`, in Figure 2.38.

System

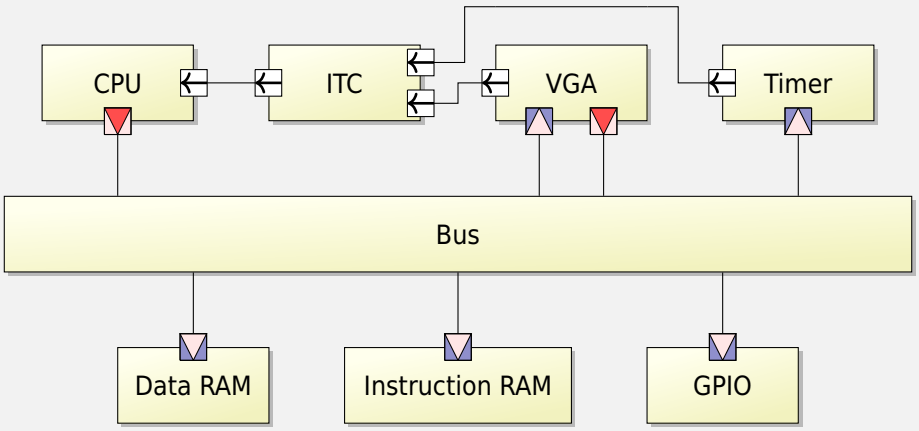


Figure 2.39: System show sockets connected through the bus

`tikzgoodies/tikzlm/system.tex`, in Figure 2.39.

Memory Map

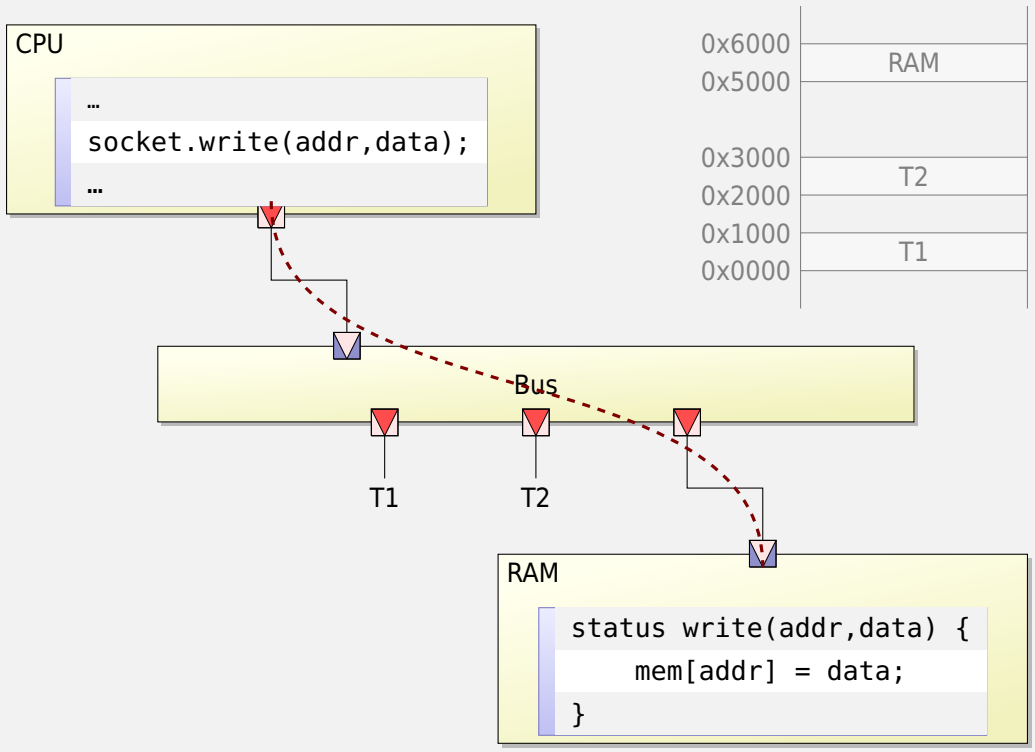


Figure 2.40: System with memory map on the side

`tikzgoodies/tikzlm/memmap.tex`, in Figure 2.40.

2.6.5 TikZ-UML

Homepage TikZ-UML Project

Documentation TikZ-UML User Guide

TikZ-UML is a *TikZ* extension to manage common *UML* diagrams. The primary goal was to propose an alternative to the pst-uml package, as *TikZ* is an alternative itself to PSTricks.

Use Case Diagram

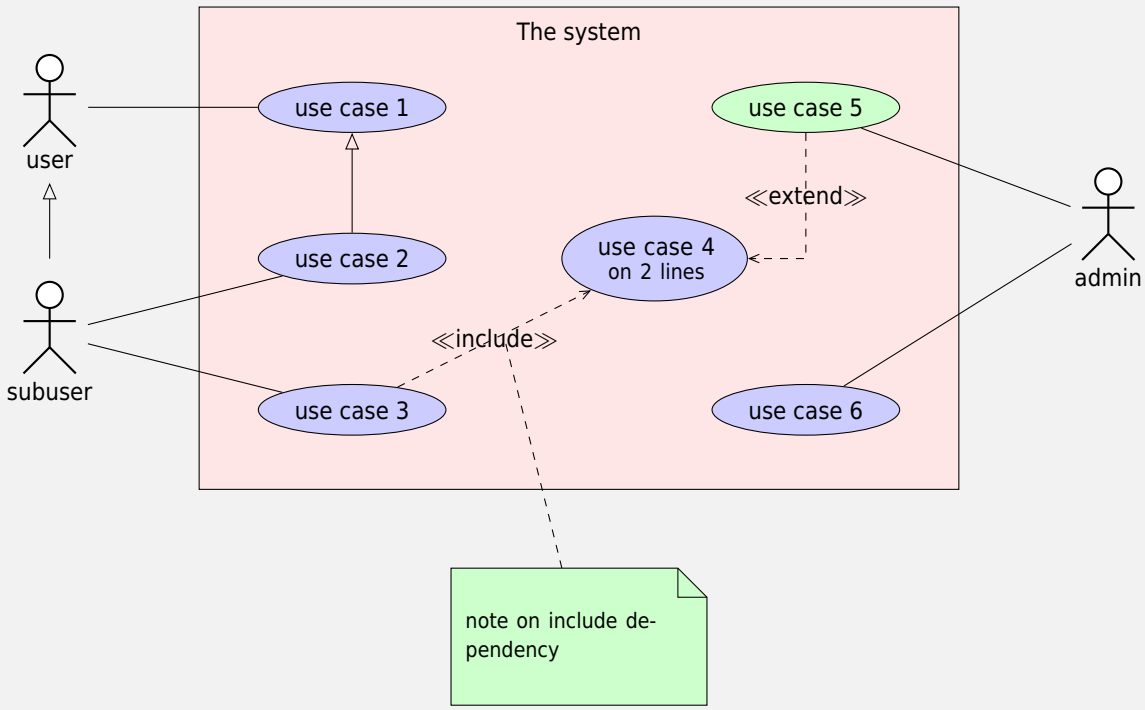


Figure 2.41: Use case diagram example

`tikzuml/usecasediag.tex`, in Figure 2.41.

Class Diagram

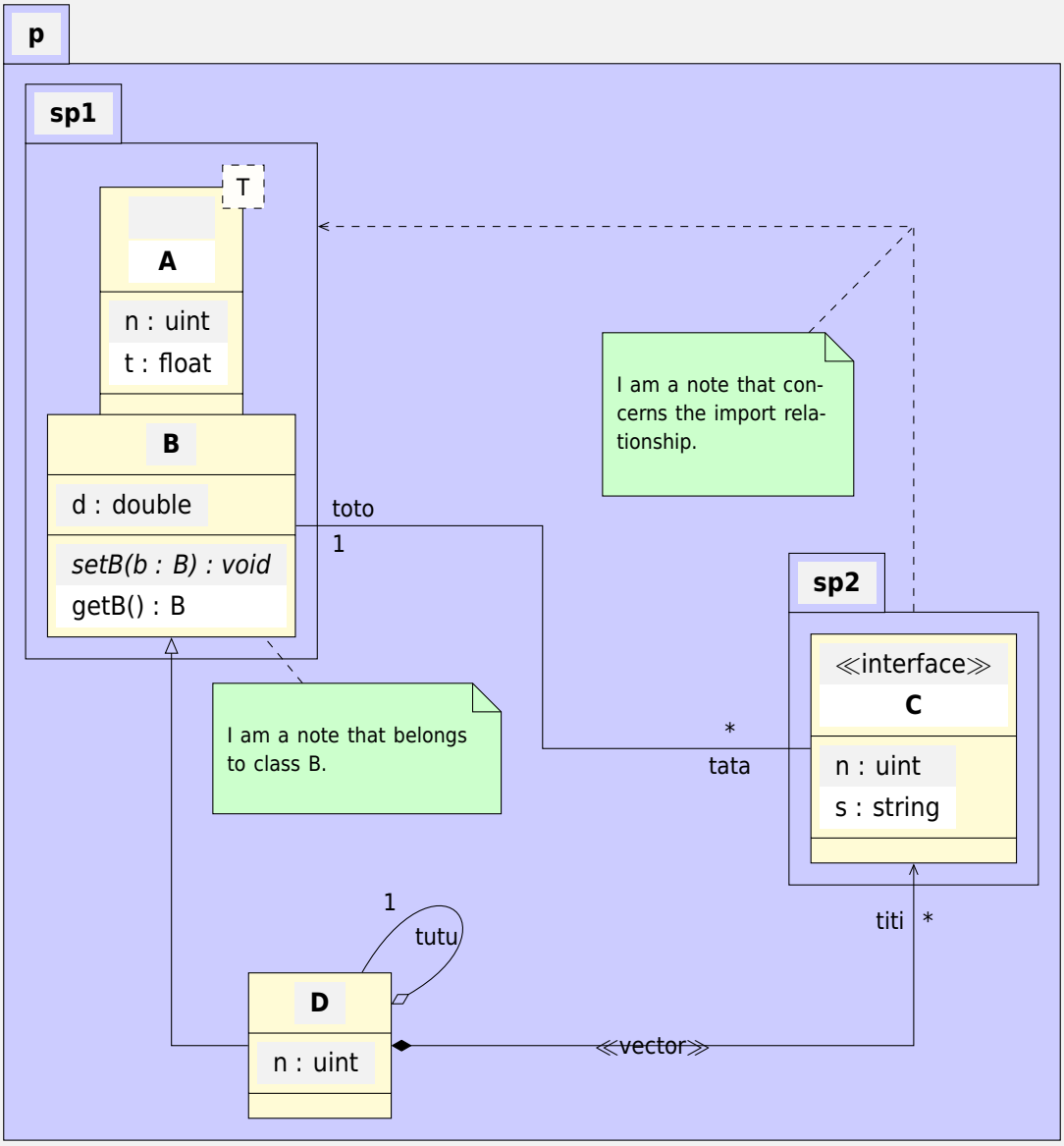


Figure 2.42: Class diagram example

`tikzuml/classdiag.tex`, in Figure 2.42.

State-Machine Diagram

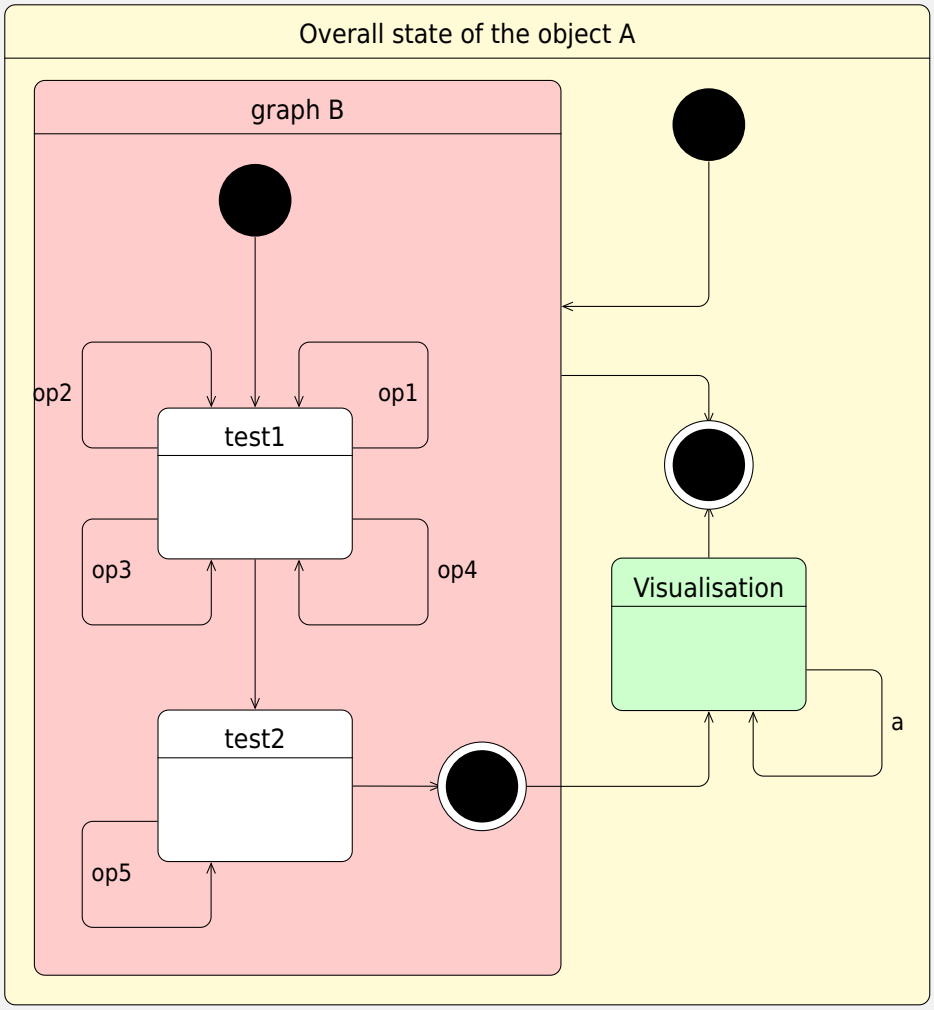


Figure 2.43: State-machine diagram example

`tikzuml/statediag.tex`, in Figure 2.43.

Sequence Diagram

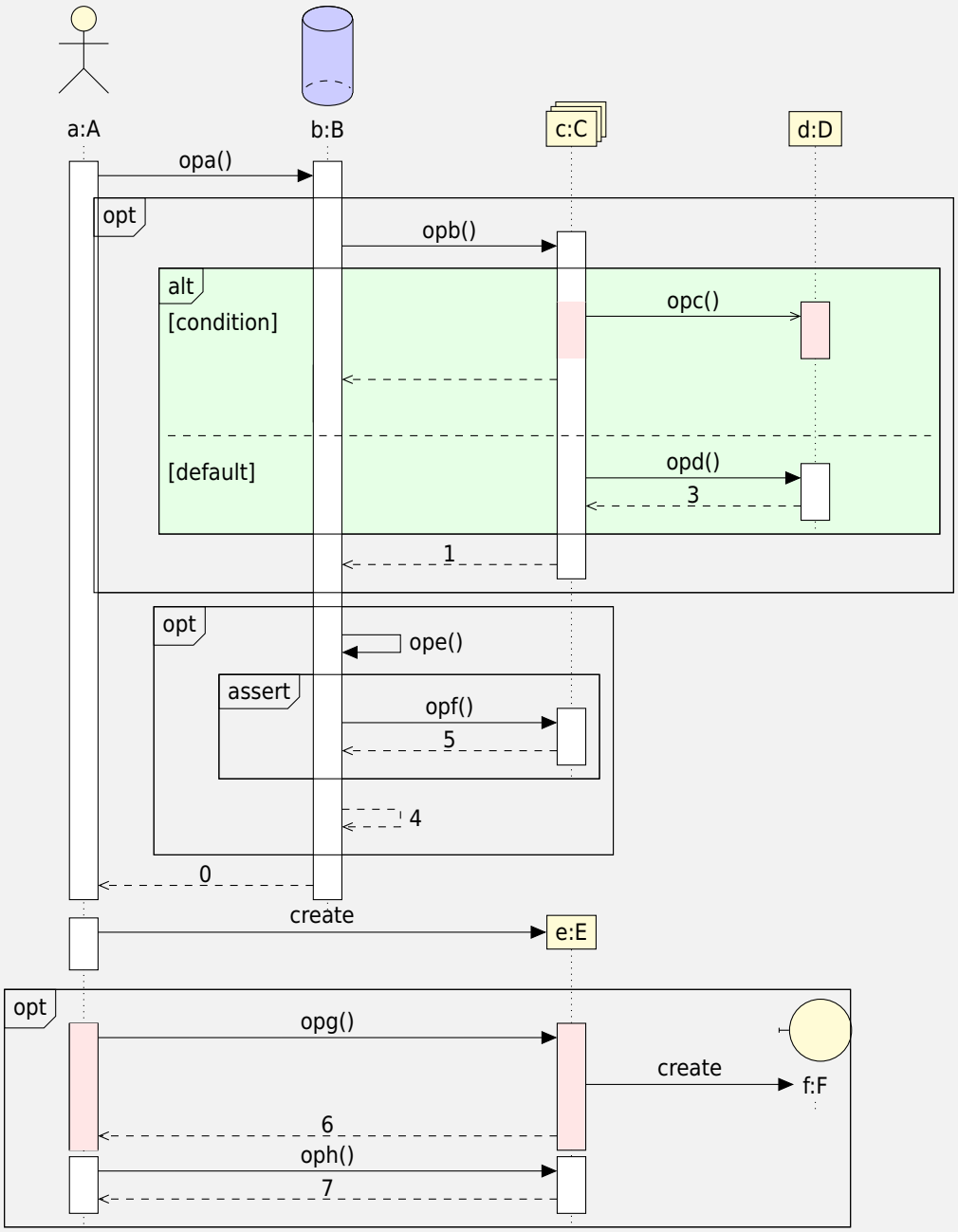


Figure 2.44: Sequence diagram example

`tikzuml/seqdiag.tex`, in Figure 2.44.

Component Diagram (still in progress)

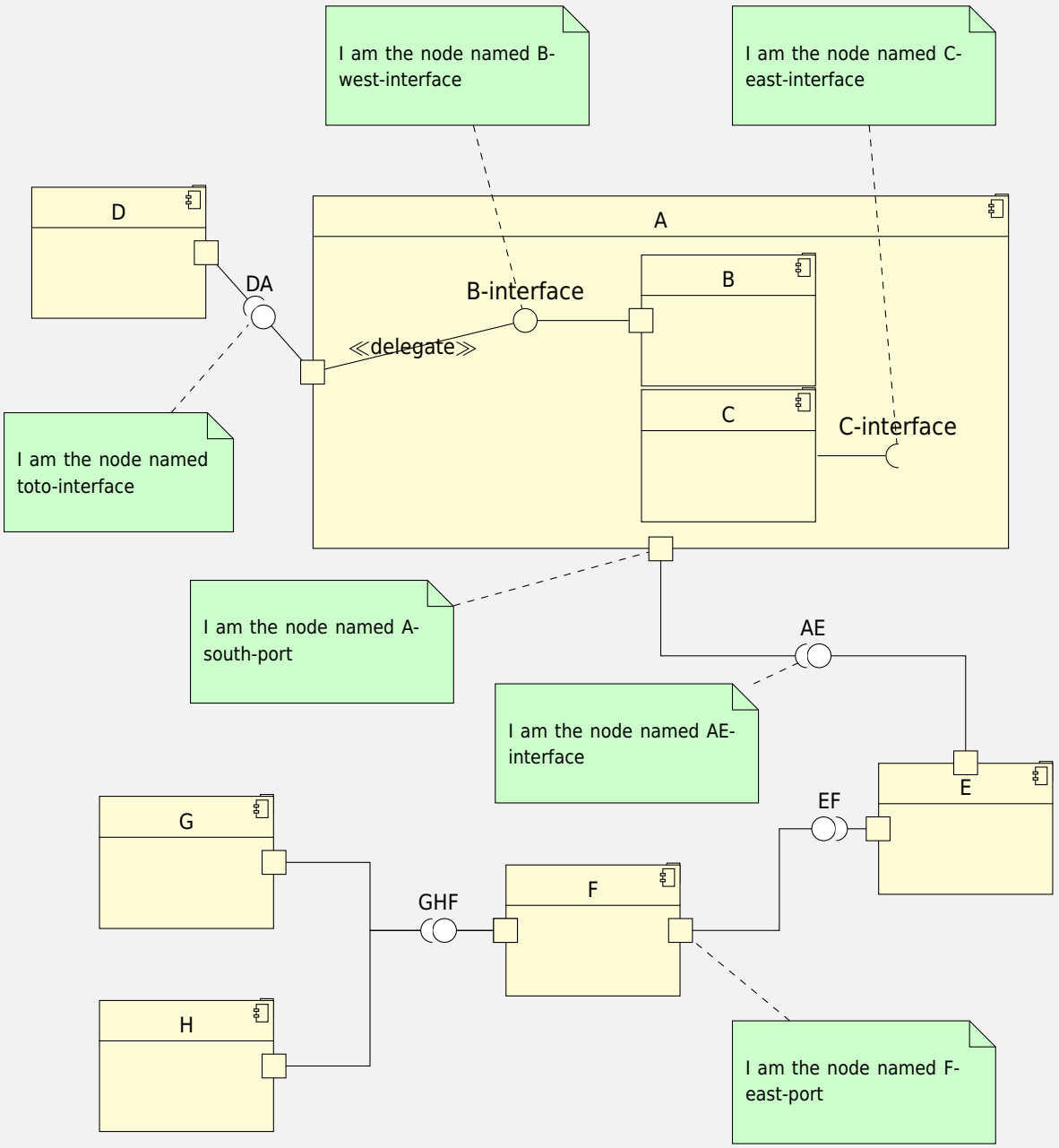
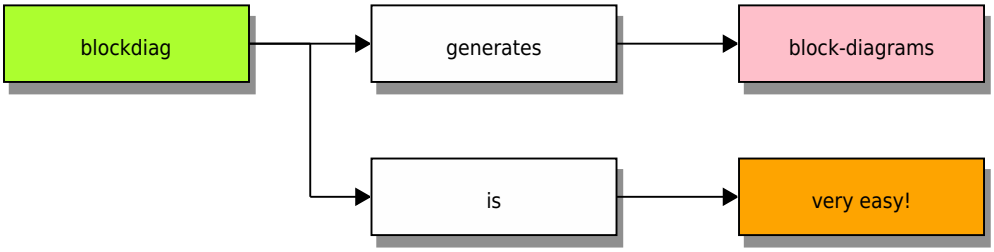


Figure 2.45: Component diagram example

`tikzuml/componentdiag.tex`, in Figure 2.45.

2.7 Block Diagram Family

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



Features

1. Supports many types of diagrams
   - block diagram (w/ blockdiag)
   - sequence diagram (w/ seqdiag)
   - activity diagram (w/ actdiag)
   - logical network diagram (w/ nwdiag)
   - rack-structure diagram (w/ rackdiag)
   - packet header diagram (w/ packetdiag)
2. Generates beautiful diagram images from simple text format (similar to Graphviz's `dot` format)
3. Layouts diagram elements automatically
4. Embeds to many documentations; *Sphinx*, Trac, Redmine, and some Wikis
5. The output image format at generating *HTML* docs is either default *PNG* or *SVG* as preferred for this document.
6. The output image format at generating *PDF* docs (through *LaTeX*) is either default *PNG* or *PDF* as preferred for this document. In case of *PDF*, the *ReportLab* library is required.

2.7.1 Block Diagram

sphinxcontrib-blockdiag is a *Sphinx* extension for embedding block diagrams. You can embed block diagrams with the `.. blockdiag::` directive.

PyPI Package https://pypi.org/project/sphinxcontrib-blockdiag/

Documentation http://blockdiag.com/en/blockdiag/sphinxcontrib.html

Git Repository https://github.com/blockdiag/sphinxcontrib-blockdiag

Sphinx extension for embedding block diagrams using blockdiag.

Features

1. Generate block-diagram from dot like text (basic feature).
2. Multilingualism for node-label (utf-8 only).

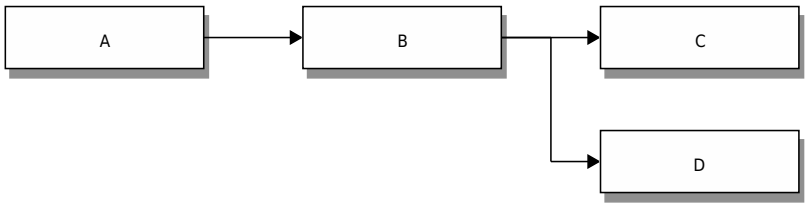
2.7.1.1 Directive Body Diagram

`.. blockdiag::`

For more details, see sphinxcontrib-blockdiag in the extension demonstration and the `README.rst` in the extension Git repository.

The example

```
.. blockdiag::
   :align: center

   blockdiag {
       A -> B -> C;
            B -> D;
   }
```

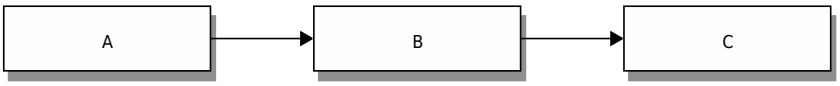


Which gives

2.7.1.2 Description Table

the example

```
.. blockdiag::
   :align: center
   :desctable:

   blockdiag {
       A -> B -> C;
       A [description = ”browsers in each client”];
       B [description = ”web server”];
       C [description = ”database server”];
   }
```



which gives

Name	Description
A	browsers in each client
B	web server
C	database server

2.7.1.3 Include Diagram

the example

```
.. blockdiag:: blockdiag/example.diag
   :caption: Style attributes to nodes and edges (Block Diagram example)
   :align: center
   :width: 480
```

which gives

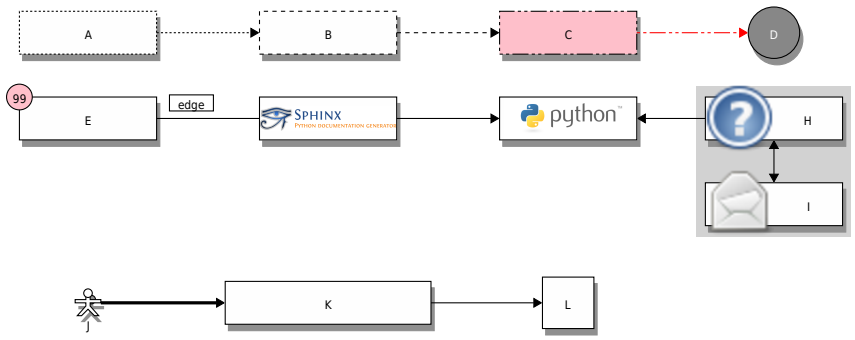


Figure 2.46: Style attributes to nodes and edges (Block Diagram example)

which needs The example above comes from the original Sample diagrams web page and processed the following file content:

Listing 2.5: Block Diagram example file (blockdiag/example.diag)

```
blockdiag {
    // Set boder-style, backgroun-color and text-color to nodes.
    A [style = dotted];
    B [style = dashed];
    C [color = pink, style = "3,3,3,3,15,3"]; //dashed_array format style
    D [shape = circle, color = "#888888", textcolor="#FFFFFF"];

    // Set border-style and color to edges.
    A -> B [style = dotted];
    B -> C [style = dashed];
    C -> D [color = "red", style = "3,3,3,3,15,3"]; //dashed_array format style

    // Set numbered-badge to nodes.
    E [numbered = 99];
```

(continues on next page)

(continued from previous page)

```
15
16      // Set background image to nodes (and erase label).
17      F [label = "", background = "https://github.com/sphinx-doc/sphinx/
    ↪raw/master/doc/_static/sphinx.png"];
18      G [label = "", background = "https://www.python.org/static/
    ↪community_logos/python-logo-master-v3-TM.png"];
19      H [icon = "https://github.com/blockdiag/blockdiag.com/raw/master/
    ↪sources/en/_static/help-browser.png"];
20      I [icon = "https://github.com/blockdiag/blockdiag.com/raw/master/
    ↪sources/en/_static/internet-mail.png"];
21      J [shape = actor]
22
23      // Set arrow direction to edges.
24      E -> F [dir = none, label = edge];
25      F -> G [dir = forward];
26      G -> H [dir = back];
27
28      group {
29          orientation = portrait;
30          color = lightgray;
31          H -> I [dir = both];
32      }
33
34      // Set width and height to nodes.
35      K [width = 192]; // default value is 128
36      L [shape = square, height = 64]; // default value is 40
37
38      // Use thick line
39      J -> K [thick]
40      K -> L;
41  }
```

2.7.2 Sequence Diagram

sphinxcontrib-seqdiag is a *Sphinx* extension for embedding sequence diagrams. You can embed sequence diagrams with the `.. seqdiag::` directive.

PyPI Package https://pypi.org/project/sphinxcontrib-seqdiag/

Documentation http://blockdiag.com/en/seqdiag/sphinxcontrib.html

Git Repository https://github.com/blockdiag/sphinxcontrib-seqdiag

Sphinx extension for embedding sequence diagrams using seqdiag.

Features

1. Generate sequence-diagram from dot like text (basic feature).
2. Multilingualism for node-label (utf-8 only).

2.7.2.1 Directive Body Diagram

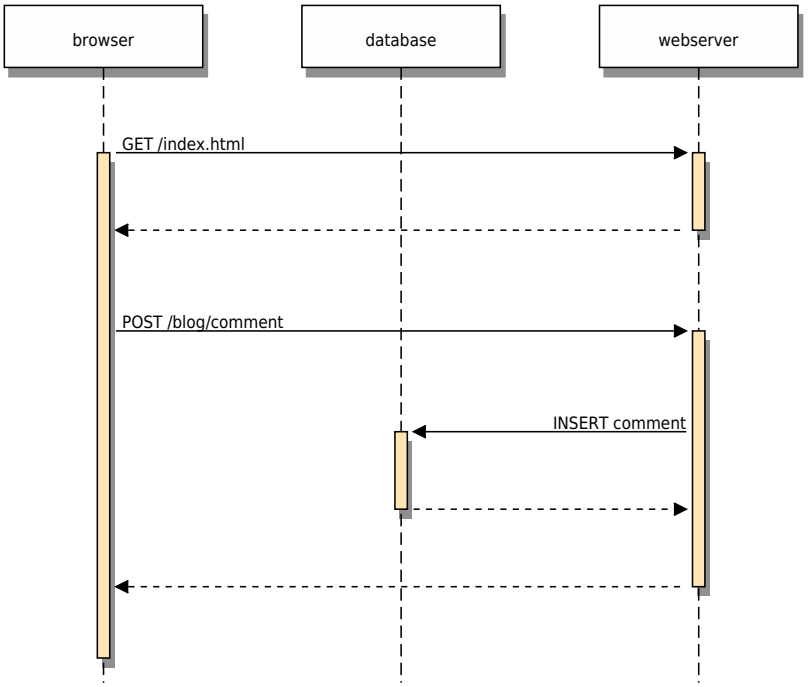
`.. seqdiag::`

For more details, see sphinxcontrib-seqdiag in the extension demonstration and the `README.rst` in the extension Git repository.

The example

```
.. seqdiag::
   :align: center

   seqdiag {
       # define order of elements
       # seqdiag sorts elements by order they appear
       browser; database; webserver;

       browser  -> webserver [label = "GET /index.html"];
       browser <-- webserver;
       browser  -> webserver [label = "POST /blog/comment"];
                   webserver  -> database [label = "INSERT comment
↪"];
                   webserver <-- database;
       browser <-- webserver;
   }
```

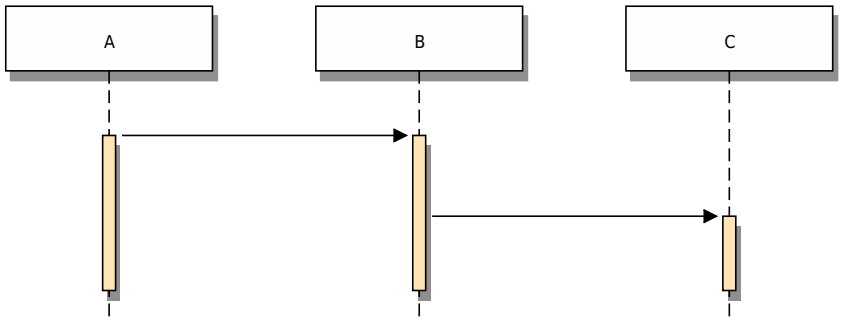
Which gives



2.7.2.2 Description Table

the example

```
.. seqdiag::
   :align: center
   :desctable:

   seqdiag {
       A -> B -> C;
       A [description = "browsers in each client"];
       B [description = "web server"];
       C [description = "database server"];
   }
```



which gives

Name	Description
A	browsers in each client
B	web server
C	database server

2.7.2.3 Include Diagram

the example

```
.. seqdiag:: seqdiag/example.diag
   :caption: Style attributes to diagram and edges (Sequence Diagram␣
↪example)
   :align: center
   :height: 480
```

which gives

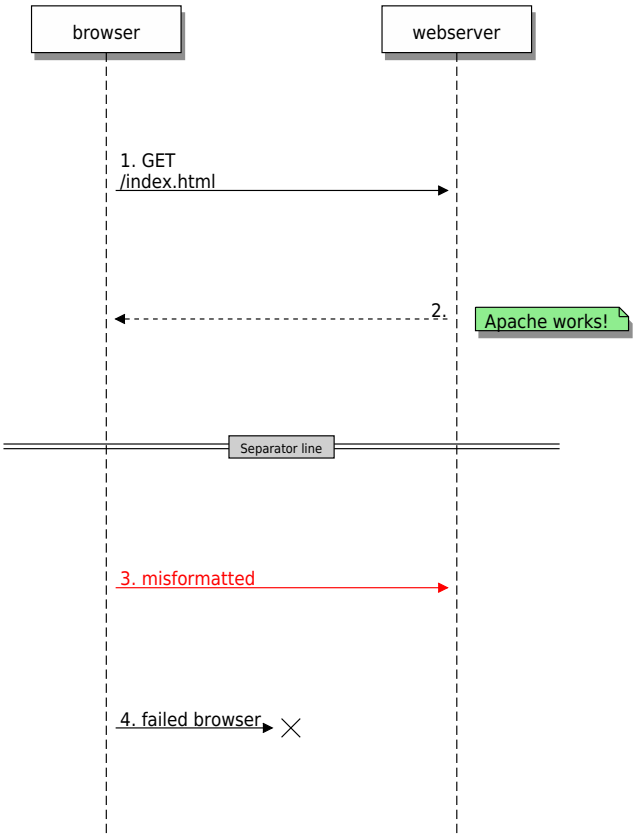


Figure 2.47: Style attributes to diagram and edges (Sequence Diagram example)

which needs The example above comes from the original Sample diagrams web page and processed the following file content:

Listing 2.6: Sequence Diagram example file (seqdiag/example.diag)

```
seqdiag {
    // Set edge metrix.
    edge_length = 300;  // default value is 192
    span_height = 80;  // default value is 40

    // Set fontsize.
    default_fontsize = 16;  // default value is 11

    // Do not show activity line
    activation = none;

    // Numbering edges automaticaly
    autonumber = True;

    // Change note color
    default_note_color = lightgreen;

    browser  -> webserver [label = "GET \n/index.html"];
    browser <-- webserver [note = "Apache works!"];

    // Separator
    === Separator line ===

    // color of edge
    browser -> webserver [label = "misformatted", color = red];

    // failed edge
    browser -> webserver [label = "failed browser", failed];
}
```

2.7.3 Activity Diagram

sphinxcontrib-actdiag is a *Sphinx* extension for embedding activity diagrams. You can embed activity diagrams with the `.. actdiag::` directive.

PyPI Package https://pypi.org/project/sphinxcontrib-actdiag/

Documentation http://blockdiag.com/en/actdiag/sphinxcontrib.html

Git Repository https://github.com/blockdiag/sphinxcontrib-actdiag

Sphinx extension for embedding activity diagrams using actdiag.

Features

1. Generate activity-diagram from dot like text (basic feature).
2. Multilingualism for node-label (utf-8 only).

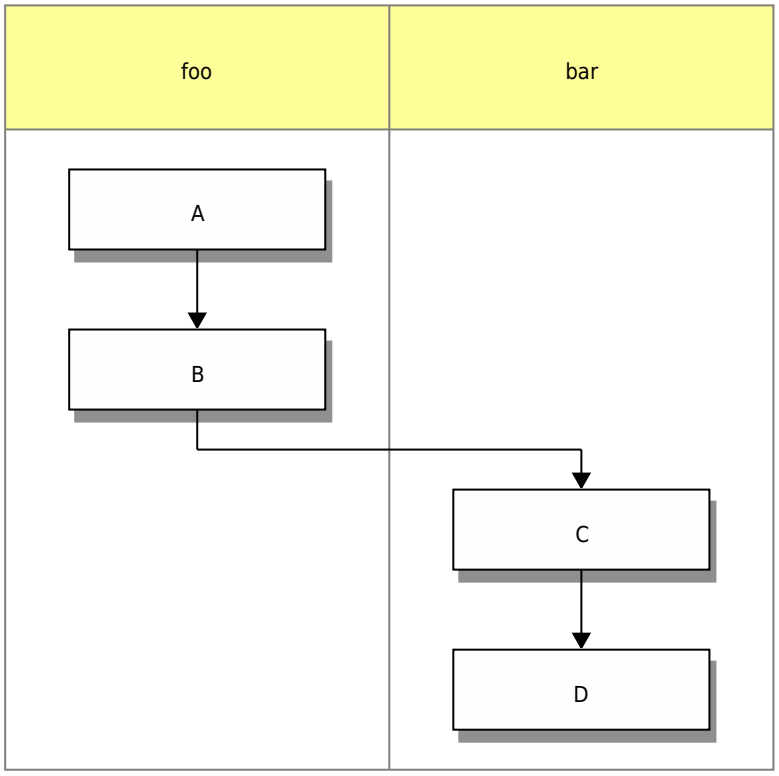
2.7.3.1 Directive Body Diagram

`.. actdiag::`

For more details, see sphinxcontrib-actdiag in the extension demonstration and the `README.rst` in the extension Git repository.

The example

```
.. actdiag::
   :align: center
   :scale: 75

   actdiag {
        A -> B -> C -> D;

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

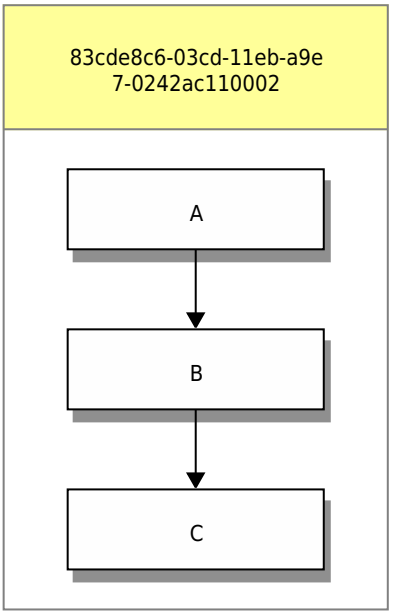


Which gives

2.7.3.2 Description Table

the example

```
.. actdiag::
   :align: center
   :scale: 75
   :desctable:

   actdiag {
        A -> B -> C;
        A [description = ”browsers in each client”];
        B [description = ”web server”];
        C [description = ”database server”];
   }
```



which gives

Name	Description
A	browsers in each client
B	web server
C	database server

2.7.3.3 Include Diagram

the example

```
.. blockdiag:: actdiag/example.diag
   :caption: Style attributes to frames and nodes (Activity Diagram␣
↪example)
   :align: center
   :scale: 75
   :width: 480
```

which gives

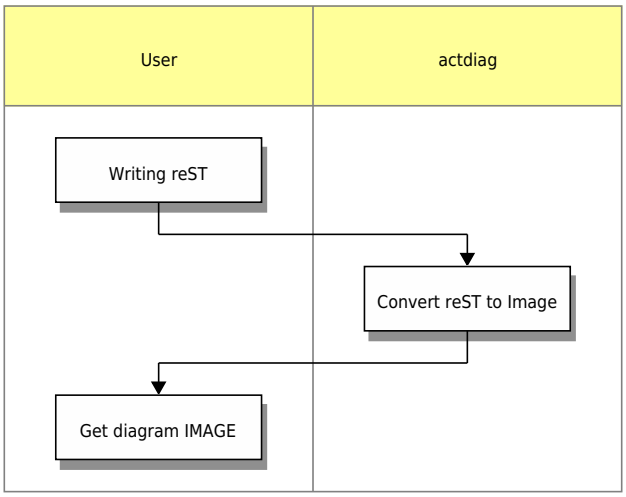


Figure 2.48: Style attributes to frames and nodes (Activity Diagram example)

which needs The example above comes from the original Sample diagrams web page and processed the following file content:

Listing 2.7: Activity Diagram example file (actdiag/example.diag)

```
actdiag {
    write -> convert -> image;

    lane user {
         label = "User";
         write [label = "Writing reST"];
         image [label = "Get diagram IMAGE"];
    }
    lane actdiag {
         convert [label = "Convert reST to Image"];
```

(continues on next page)

(continued from previous page)

```
11      }
12  }
```

2.7.4 Network Diagram

sphinxcontrib-nwdiag is a *Sphinx* extension for embedding network diagrams. You can embed network diagrams with the `.. nwdiag::`, `.. rackdiag::` and `.. packetdiag::` directives.

PyPI Package https://pypi.org/project/sphinxcontrib-nwdiag/

Documentation http://blockdiag.com/en/nwdiag/sphinxcontrib.html

Git Repository https://github.com/blockdiag/sphinxcontrib-nwdiag

Sphinx extension for embedding network diagrams using nwdiag.

Features

1. Generate network-diagram from dot like text (basic feature).
2. Multilingualism for node-label (utf-8 only).

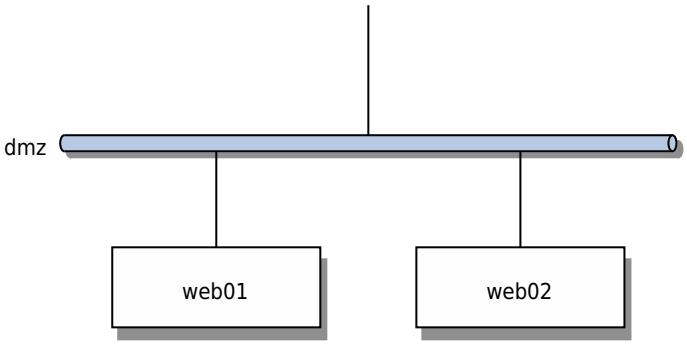
2.7.4.1 Directive Body Diagram

`.. nwdiag::`

For more details, see sphinxcontrib-nwdiag in the extension demonstration and the `README.rst` in the extension Git repository.

The example

```
.. nwdiag::
   :align: center
   :scale: 75

   nwdiag {
       network dmz {
           web01;
           web02;
       }
   }
```

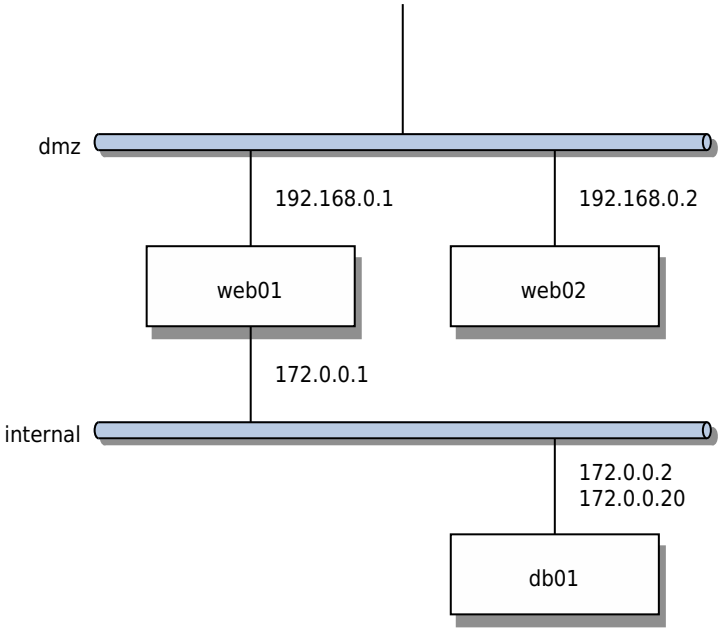


Which gives

2.7.4.2 Description Table

the example

```
.. nwdiag::
   :align: center
   :scale: 75
   :desctable:

   nwdiag {
       network dmz {
           web01 [address = ”192.168.0.1”, description = ”web server01
↪”];
           web02 [address = ”192.168.0.2”, description = ”web server02
↪”];
       }
       network internal {
           web01 [address = ”172.0.0.1”];
           db01 [address = ”172.0.0.2,172.0.0.20”, description =
↪”database server”];
       }
   }
```



which gives

Name	Description
web01	web server01
web02	web server02
db01	database server

2.7.4.3 Include Diagram

Network

the example

```
.. nwdiag:: nwdiag/example.diag
   :caption: Peer networks and grouping nodes (Network Diagram example)
   :align: center
   :scale: 75
   :width: 480
```

which gives

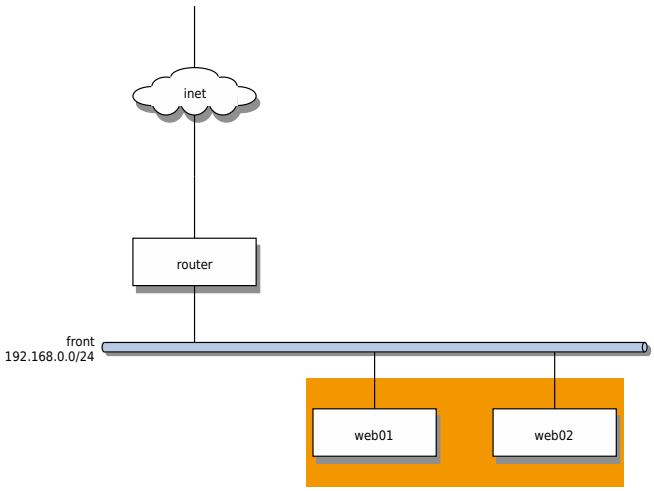


Figure 2.49: Peer networks and grouping nodes (Network Diagram example)

which needs The example above comes from the original Sample diagrams: nwdiag web page and processed the following file content:

Listing 2.8: Network Diagram example file (nwdiag/example.diag)

```
nwdiag {
    inet [shape = cloud];
    inet -- router;

    network front {
        address = "192.168.0.0/24";
```

(continues on next page)

(continued from previous page)

```
        router;
        web01;
        web02;

        // define network using defined nodes
        group db {
            web01;
            web02;
        }
    }
}
```

Rack

.. rack::

For more details, see sphinxcontrib-nwdiag in the extension demonstration and the `README.rst` in the extension Git repository.

The example

```
.. rackdiag:: rackdiag/example.diag
   :caption: Multiple racks with multiple and blocked units (Rack Diagram example)
   :align: center
   :height: 480
```

Which gives

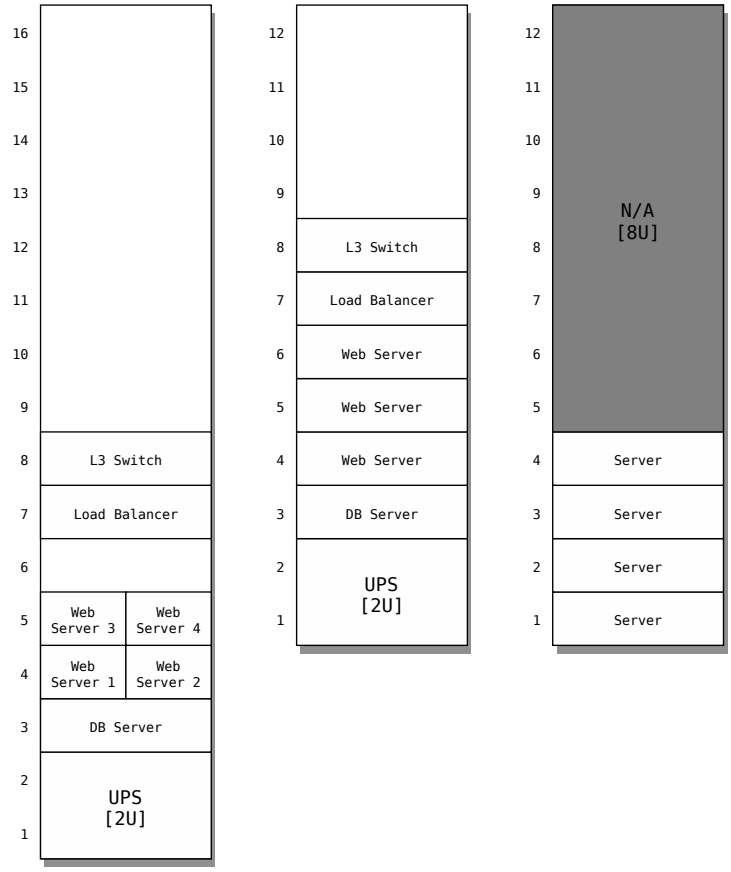


Figure 2.50: Multiple racks with multiple and blocked units (Rack Diagram example)

Which needs The example above comes from the original Sample diagrams: rackdiag web page and processed the following file content:

Listing 2.9: Rack Diagram example file (rackdiag/example.diag)

```
rackdiag {
    default_fontsize = 10;

    // define 1st (height) rack
    rack {
        16U;

        // define rack items
        1: UPS [2U, fontsize = 14];
        3: DB Server;
        // put 2 units to rack-level 4
        4: Web\nServer 1;
        4: Web\nServer 2;
        5: Web\nServer 3;
        5: Web\nServer 4;
        7: Load Balancer;
        8: L3 Switch;
    }

    // define 2nd rack
    rack {
        12U;

        // define rack items
        1: UPS [2U, fontsize = 14];
        3: DB Server;
        4: Web Server;
        5: Web Server;
        6: Web Server;
        7: Load Balancer;
        8: L3 Switch;
    }

    // define 3rd rack (with not available units)
    rack {
        12U;

        1: Server;
        2: Server;
        3: Server;
        4: Server;
        5: N/A [8U, fontsize = 14];
    }
}
```

Packet

.. packet::

For more details, see sphinxcontrib-nwdiag in the extension demonstration and the README.rst in the extension Git repository.

The example

```
.. packetdiag:: packetdiag/example.diag
   :caption: Structure of TCP Header (Packet Diagram example)
   :align: center
   :width: 480
```

Which gives

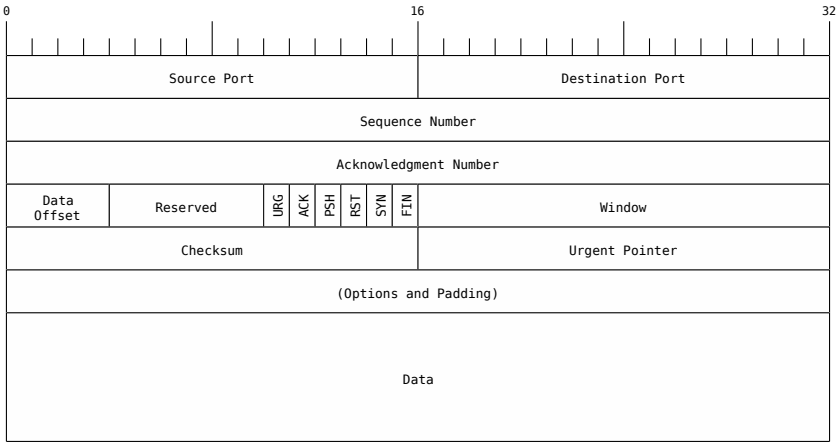


Figure 2.51: Structure of TCP Header (Packet Diagram example)

Which needs The example above comes from the original Sample diagrams: packetdiag web page and processed the following file content:

Listing 2.10: Packet Diagram example file (packetdiag/example.diag)

```
packetdiag {
    colwidth = 32;
    node_height = 40;
    default_fontsize = 12;

    0-15: Source Port;
    16-31: Destination Port;
    32-63: Sequence Number;
    64-95: Acknowledgment Number;
    96-99: Data\nOffset;
    100-105: Reserved;
```

(continues on next page)

(continued from previous page)

```
    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];
}
```

2.8 Tabbed Content

> **Attention:** Only practicable and usable for *HTML* builder.

PyPI Package https://pypi.org/project/sphinx-tabs/

Documentation https://sphinx-tabs.readthedocs.io/

Git Repository https://github.com/executablebooks/sphinx-tabs

Create tabbed content in *Sphinx* documentation when building *HTML*.

Features

1. Basic and nested tabs.
2. Grouped Tabs.
3. Code Tabs.

.. tabs::

.. tab::

For more details, see Simple Tabs in the extension demonstration and the `README.md` in the extension Git repository.

The example

```
.. tabs::

   .. tab:: **Apples**

      Apples are green, or sometimes red.

   .. tab:: **Pears**

      Pears are green.

   .. tab:: **Oranges**

      Oranges are orange.
```

Which gives Apples

Apples are green, or sometimes red.

Pears

Pears are green.

Oranges

Oranges are orange.

Nested tabs are also possible.

The example

```
.. tabs::

   .. tab:: Stars

      .. tabs::

         .. tab:: The Sun

            The closest star to us.

         .. tab:: Proxima Centauri

            The second closest star to us.

         .. tab:: Polaris

            The North Star.

   .. tab:: Moons

      .. tabs::

         .. tab:: The Moon

            Orbits the Earth

         .. tab:: Titan

            Orbits Jupiter
```

Which gives Stars

The Sun

The closest star to us.

Proxima Centauri

The second closest star to us.

Polaris

The North Star.

Moons

The Moon

Orbits the Earth

Titan

Orbits Jupiter

.. group-tab::

Also tabs can stick together in groups.

The example

```
.. rubric:: operating systems

.. tabs::

   .. group-tab:: Linux

      **Linux** is Unix-like, but was developed without any Unix code.
      The Linux kernel originated in 1991, as a project of Linus
      Torvalds, while a university student in Finland.

   .. group-tab:: Mac OS X

      **Mac OS X** is a line of open core graphical operating systems
      developed, marketed, and sold by Apple Inc.

   .. group-tab:: Microsoft Windows

      **Microsoft Windows** is a family of proprietary operating systems
      designed by Microsoft Corporation and primarily targeted to
      Intel architecture based computers.

.. rubric:: integrated development environments

.. tabs::

   .. group-tab:: Linux

      **There is no dedicated or default integrated development
      environment (IDE)** *on* *Linux*. `Here is a list`_ of IDEs
      which will run natively on *Linux*.

   .. group-tab:: Mac OS X

      **Xcode** is an integrated development environment (IDE) for
      *Mac OS X* containing a suite of software development tools
      developed by Apple Inc.

   .. group-tab:: Microsoft Windows

      **Microsoft Visual Studio** is an integrated development
```

(continues on next page)

(continued from previous page)

```
      environment (IDE) from Microsoft Corporation. It is used to
      develop computer programs uses Microsoft software development
      platforms such as *Windows API*, *Windows Forms*, *Windows
      Presentation Foundation*, *Windows Store* and *Microsoft
      Silverlight*.

.. _`Here is a list`:
   https://en.wikipedia.org/wiki/Category:Linux_integrated_
↪development_environments
```

Which gives

operating systems

Linux

Linux is Unix-like, but was developed without any Unix code. The Linux kernel originated in 1991, as a project of Linus Torvalds, while a university student in Finland.

Mac OS X

Mac OS X is a line of open core graphical operating systems developed, marketed, and sold by Apple Inc.

Microsoft Windows

Microsoft Windows is a family of proprietary operating systems designed by Microsoft Corporation and primarily targeted to Intel architecture based computers.

integrated development environments

Linux

There is no dedicated or default integrated development environment (IDE) on *Linux*. Here is a list of IDEs which will run natively on *Linux*.

Mac OS X

Xcode is an integrated development environment (IDE) for *Mac OS X* containing a suite of software development tools developed by Apple Inc.

Microsoft Windows

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft Corporation. It is used to develop computer programs uses Microsoft software development platforms such as *Windows API*, *Windows Forms*, *Windows Presentation Foundation*, *Windows Store* and *Microsoft Silverlight*.

.. code-tab::

Similar to grouped tabs they can contain code areas with syntax highlighting.

The example

```
.. rubric:: main entry function

.. tabs::

   .. code-tab:: c

         int main(const int argc, const char **argv) {
             return 0;
         }

   .. code-tab:: c++

         int main(const int argc, const char **argv) {
             return 0;
         }

   .. code-tab:: python

         def main():
             return

   .. code-tab:: java

         class Main {
             public static void main(String[] args) {
             }
         }

   .. code-tab:: julia

         function main()
         end

   .. code-tab:: fortran

         PROGRAM main
         END PROGRAM main

.. rubric:: programming languages

.. tabs::

   .. code-tab:: c
```

(continues on next page)

(continued from previous page)

```
44
45          C Main Function
46
47      .. code-tab:: c++
48
49          C++ Main Function
50
51      .. code-tab:: python
52
53          Python Main Function
54
55      .. code-tab:: java
56
57          Java Main Function
58
59      .. code-tab:: julia
60
61          Julia Main Function
62
63      .. code-tab:: fortran
64
65          Fortran Main Function
```

Which gives

main entry function

C

```c
int main(const int argc, const char **argv) {
    return 0;
}
```

C++

```cpp
int main(const int argc, const char **argv) {
    return 0;
}
```

Python

```python
def main():
    return
```

Java

```
class Main {
    public static void main(String[] args) {
    }
}
```

Julia

```
function main()
end
```

Fortran

```
PROGRAM main
END PROGRAM main
```

programming languages

C

```
C Main Function
```

C++

```
C++ Main Function
```

Python

```
Python Main Function
```

Java

```
Java Main Function
```

Julia

```
Julia Main Function
```

Fortran

```
Fortran Main Function
```

2.9 Paneled Content

Todo: Evaluate the integration and coexistence of this “Paneled Content” extension with all other. See `conf.py` for more details.

Attention: Only practicable and usable for *HTML* builder.

PyPI Package https://pypi.org/project/sphinx-panels/

Documentation https://sphinx-panels.readthedocs.io/

Git Repository https://github.com/executablebooks/sphinx-panels

Create paneled content in *Sphinx* documentation when building *HTML*.

Features

1. Panels in grid or cards layout.
2. Panels with click-able link-button.
3. Panels with toggle-able content by drop-downs.
4. Panels with styling: header, footer, images, icons, *badges*, animations

For more details, see sphinx-panels in the extension demonstration and the `README.md` in the extension Git repository.

`.. panels::`
For more details, see Panels Usage.

`.. dropdown::`
For more details, see Dropdown Usage.

`.. link-button::`
For more details, see Link Buttons.

`.. div::`
For more details, see Div Directive.

`:badge:`

`:link-badge:`
For more details, see Link Badges.

`:opticon:`

`:fa:`
For more details, see Inline Icons.

Extension not applicable

This *Sphinx* extension is quite new and is under constant development. The current behavior disturbs the integration, so the extension is disabled for now (see `conf.py`). Currently known bugs are:

- annoying side effects with the *Tabbed Content* (page 159) extension by the automatically integrated and delivered Bootstrap 4.0 *CSS*
- no proper and practical *LaTeX* builder support

2.10 Email Obfuscate

> **Attention:** Only practicable and usable for *HTML* builder.

PyPI Package https://pypi.org/project/sphinxcontrib-email/

Documentation https://github.com/sphinx-contrib/email/blob/master/README.rst

Git Repository https://github.com/sphinx-contrib/email

Python 3 Fixes https://github.com/rexut/sphinxcontrib-email/tree/python3-fixes

To obfuscate an email address use something like:

```
:email:`Name Surname <user@myplace.org>`
```

That renders as Name Surname with the appropriate mailto link.

```
:email:`user@myplace.org`
```

That renders as user@myplace.org with the appropriate mailto link.

:email:

The example

```
* :email:`Name Surname <user@myplace.org>`
* :email:`user@myplace.org`
```

Which gives

-
-

3 Themes

Section author: Stephan Linz <linz@li-pro.net>

Let's decorate the project documentation. There are a lot of themes for the *Sphinx HTML* builder available on the Sphinx Themes Demo Page.

This documentation use the Read the Docs Sphinx Theme as demonstrate at the Sphinx RTD Theme Demo Page.

Another interesting and actively developed theme is The Sphinx Book Theme, the theme by The Executable Book Project.

3.1 Read the Docs Sphinx Theme

PyPI Package https://pypi.org/project/sphinx-rtd-theme/

Documentation https://sphinx-rtd-theme.readthedocs.io/

Git Repository https://github.com/readthedocs/sphinx_rtd_theme

This theme is primarily focused to be used on Read the Docs but can work with your own sphinx projects. You can find a working demo of the theme in the theme documentation.

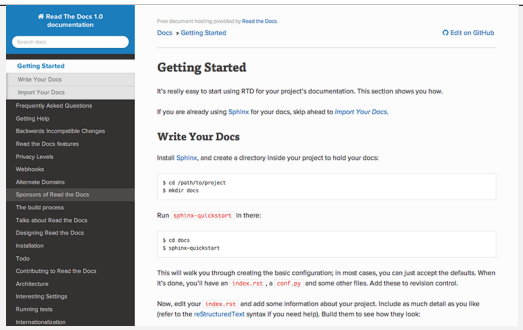


Figure 3.1: Sphinx RTD theme on Desktop

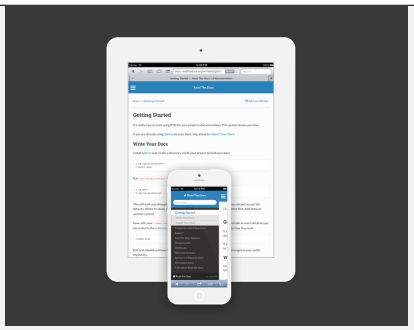


Figure 3.2: Sphinx RTD theme on Mobiles

4 Cheat Sheet

Section author: Stephan Linz <linz@li-pro.net>

We have made a cheat sheet for helping you remember the syntax for *reStructuredText* & *Sphinx* programs. The basic reStructuredText Cheat Sheet could also be very helpful.

reStructuredText & Sphinx Cheatsheet

Section Headings

```
==============
Level 1 Heading
==============

Level 2 Heading
---------------
```

```
Level 1 Heading
^^^^^^^^^^^^^^^

Level 2 Heading
+++++++++++++++
```

- Heading structure is determined only by occurance order.
- Heading overline is optional.
- Under/overlines use the following characters:

Recommended: `= - ` : . ' " ~ ^ _ * + #`
`! $ % & ( ) , / ; < > ? @ [ \ ] { | }`

Styles

```
*Italic/emphasis text*
**Bold/strong text**
``Inline literal text/code``
:sup:`super`\ Script
:sub:`sub`\ Script
```

Bullet Lists

```
* Unordered item
* Unordered item

  1. Nested ordered item
  2. Nested ordered item

     a. Nested ordered item

* Unordered item
```

Definition Lists

```
First term
   Definition of first term. It can
   span multiple lines which alined
   on same indention.

Second term
   Definition of second term.

   Indent addition paragraphs, which
   even can span multiple lines that
   aligned on same indention.
```

Lieral Code Blocks

```
Here is a literal block::

   Its content are indented.

::

   The :: marker is omitted here
```

The `::` marker will inserted a ":" in the output from the example above. To omit the ":" in the output, precede the `::` marker with white space, or use the marker on a line of its own.

Targets and Links

Anchor target
```
.. _anchorbyref:
.. _Anchor by text:
```

External target
Footnote target
Citation target
```
.. _External link name: http://example.com/
.. [1] A footnote
.. [cit1] A global citation
```

External links
```
`External link <http://example.com/>`_
`External link name`_ or `Example <External link name_>`_
```

Internal links
```
`Anchor by text`_ or `Anchor <Anchor by text_>`_
`Anchor by ref <anchorbyref_>`_
:ref:`anchorbyref`
```

Footnote
Citation
```
Reference a footnote [1]_
or a global citation [cite1]_
```

Section link
```
Section Heading
---------------
`Link <Section Heading_>`_
```

Tables

```
============  ========  ========
Simple table  Header 2  Header 3
============  ========  ========
Column 1      Column 2  Column 3
Horizontal column span  ...
----------------------  --------
...           ...       ...
============  ========  ========
```

```
+----------+----------+----------+
|   Grid   | Header 2 | Header 3 |
|   table  |          |          |
+==========+==========+==========+
| Column 1 | Column 2 | Vertical |
+----------+----------+  column  |
| Horizontal span     |   span   |
+----------+----------+----------+
```

Simple tables and grid tables can be replaced with external CSV files, using the `csv-table` directive.

Images and Figures

```
.. image:: image.png
   :height: 100px
   :width: 100px
   :align: bottom
   :target: target_
```

```
.. firgure:: image.png
   :height: 100px

   Figures are images with captions. They support
   all image options.
```

Comments

```
.. This is a single line comment, comments can span multiple lines as well.
.. This is a comment that
   span multiple line.
```

Your additional notes and comments

Figure 4.1: *Cheat Sheet* **reStructuredText & Sphinx 1/2**

Directives

reStructuredText **directives** consist of a directive **type**, **arguments** and any number of **options**. Some directives expect a block of indented **content**.

```
.. type:: arguments
   :option: option value

   Directive content
```

```
.. container:: [container class]
.. csv-table:: [table title]
   :header: CSV data for headers
   :widths: number [, number]
   :file: filename
   :encoding: encoding
   :header-rows: number
   :delim: character, "tab", or "space"
   :quote: character
   :escape: character
.. rubric:: title
```

Content Block Directives

```
.. topic:: [title]
.. sidebar:: [title]
   :subtitle: subtitle
.. admonition:: title
.. attention::
.. caution::
.. danger::
.. error::
.. hint::
.. important::
.. note::
.. tip::
.. warning::
.. seealso::
.. deprecated:: [version]
.. versionadded:: [version]
.. versionchanged:: [version]
.. math::
.. raw:: output format
```

```
.. topic:: Example

   A topic block.

.. note::
   This is a note.

.. versionchanged:: 1.3
   Something changed.

.. math::
   a_1 = b_1 + c_1

.. raw:: html

   <b>HTML output</b>
```

Table of Contents

```
.. toctree::
   :maxdepth: number (of title levels)
   :glob:
   :hidden:
   :numbered:
   :caption: caption text
   :titlesonly:

   [Include paths]
```

Code Examples

```
.. highlight:: language
   :linenothreshold: number
.. codeblock:: [language]
   :linenos:
   :emphasize-lines: numbers [, numbers]
   :caption: caption text
   :name: block target name

   [Code example, indented]
.. literalinclude:: filename
   :language: language
   :linenos:
   :emphasize-lines: numbers [, numbers]
   :encoding: encoding
   :diff: filename
   :dedent: number
```

Some of the language lexers supported by the code example `language` option: *none*, *python*, *js*, *php*, *ruby*, *perl*, *c*, *cpp*, *csharp*, *go*, *scala*, *lisp*, *coffee*, *dart*, *julia*, *lua*, *html*, *css*, *sass*, *json*, *yaml*, *diff*, *sql*, *bash*, and *shell-session*

Python Domain

Python Domain Directives

```
.. py:module:: module
.. py:class:: signature
.. py:function:: signature
.. py:data:: name
.. py:exception:: name
.. py:attribute:: name
.. py:method:: signature
.. py:staticmethod:: signature
.. py:classmethod:: signature
.. py:decorator:: signature
.. py:currentmodule:: module
```

Python Domain References

```
:py:mod:`module`
:py:class:`signature`
:py:func:`signature`
:py:data:`name`
:py:exc:`name`
:py:attr:`name`
:py:meth:`signature`

:py:const:`name`
```

Python Domain Directive Options

```
:param [type] name: description
:type name: type
:raises class: description
:var name: description
:vartype name: type
:returns description:
:rtype type:
```

There are a number of other Sphinx language domains for representing code constructs in reference documentation: *rst*, *c*, *cpp*, *js*, *ruby*, *php*, *dotnet*, *scala*, *go*, *lisp*, *coffee*, and others.

References

`:role:`title <target>``	Link to *target* with link text *title*
`:role:`!title``	Don't create link or reference
`:role:`~module.Object``	Use last element, *Object*, for link text

Cross-reference roles

```
:any:`name`
:doc:`document name`
:download:`filename`
:envvar:`name`
:keyword:`python keyword`
:option:`CLI option`
```

Cross-reference roles

```
:abbr:`long (abbreviation)`
:command:`name`
:kbd:`keystrokes`
:program:`name`
:pep:`number`
:rfc:`number`
```

```
Brought to you by **Red the Docs**.
We offer `private hosting <https://readthedocs.com/>`_
and `open source hosting <https://readthodocs.org/>`_
fo *Sphinx* documentation projects.
```

Your additional notes and comments

Figure 4.2: *Cheat Sheet* **reStructuredText & Sphinx 2/2**

A Appendix

Section author: Stephan Linz <linz@li-pro.net>

A.1 License

Listing 1.1: License text of the Li-Pro.Net Sphinx Primer

```
Creative Commons Legal Code

Attribution-ShareAlike 3.0 Unported

    CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE
    LEGAL SERVICES. DISTRIBUTION OF THIS LICENSE DOES NOT CREATE AN
    ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS
    INFORMATION ON AN ”AS-IS” BASIS. CREATIVE COMMONS MAKES NO WARRANTIES
    REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS LIABILITY FOR
    DAMAGES RESULTING FROM ITS USE.

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE
COMMONS PUBLIC LICENSE (”CCPL” OR ”LICENSE”). THE WORK IS PROTECTED BY
COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS
AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE
TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY
BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS
CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND
CONDITIONS.
```

(continues on next page)

(continued from previous page)

```
1. Definitions

 a. ”Adaptation” means a work based upon the Work, or upon the Work and
    other pre-existing works, such as a translation, adaptation,
    derivative work, arrangement of music or other alterations of a
    literary or artistic work, or phonogram or performance and includes
    cinematographic adaptations or any other form in which the Work may be
    recast, transformed, or adapted including in any form recognizably
    derived from the original, except that a work that constitutes a
    Collection will not be considered an Adaptation for the purpose of
    this License. For the avoidance of doubt, where the Work is a musical
    work, performance or phonogram, the synchronization of the Work in
    timed-relation with a moving image (”synching”) will be considered an
    Adaptation for the purpose of this License.
 b. ”Collection” means a collection of literary or artistic works, such as
    encyclopedias and anthologies, or performances, phonograms or
    broadcasts, or other works or subject matter other than works listed
    in Section 1(f) below, which, by reason of the selection and
    arrangement of their contents, constitute intellectual creations, in
    which the Work is included in its entirety in unmodified form along
    with one or more other contributions, each constituting separate and
    independent works in themselves, which together are assembled into a
    collective whole. A work that constitutes a Collection will not be
    considered an Adaptation (as defined below) for the purposes of this
    License.
 c. ”Creative Commons Compatible License” means a license that is listed
    at https://creativecommons.org/compatiblelicenses that has been
    approved by Creative Commons as being essentially equivalent to this
    License, including, at a minimum, because that license: (i) contains
    terms that have the same purpose, meaning and effect as the License
    Elements of this License; and, (ii) explicitly permits the relicensing
    of adaptations of works made available under that license under this
    License or a Creative Commons jurisdiction license with the same
    License Elements as this License.
 d. ”Distribute” means to make available to the public the original and
    copies of the Work or Adaptation, as appropriate, through sale or
    other transfer of ownership.
 e. ”License Elements” means the following high-level license attributes
    as selected by Licensor and indicated in the title of this License:
    Attribution, ShareAlike.
 f. ”Licensor” means the individual, individuals, entity or entities that
    offer(s) the Work under the terms of this License.
 g. ”Original Author” means, in the case of a literary or artistic work,
    the individual, individuals, entity or entities who created the Work
    or if no individual or entity can be identified, the publisher; and in
    addition (i) in the case of a performance the actors, singers,
```

(continues on next page)

(continued from previous page)

```
   musicians, dancers, and other persons who act, sing, deliver, declaim,
   play in, interpret or otherwise perform literary or artistic works or
   expressions of folklore; (ii) in the case of a phonogram the producer
   being the person or legal entity who first fixes the sounds of a
   performance or other sounds; and, (iii) in the case of broadcasts, the
   organization that transmits the broadcast.
h. ”Work” means the literary and/or artistic work offered under the terms
   of this License including without limitation any production in the
   literary, scientific and artistic domain, whatever may be the mode or
   form of its expression including digital form, such as a book,
   pamphlet and other writing; a lecture, address, sermon or other work
   of the same nature; a dramatic or dramatico-musical work; a
   choreographic work or entertainment in dumb show; a musical
   composition with or without words; a cinematographic work to which are
   assimilated works expressed by a process analogous to cinematography;
   a work of drawing, painting, architecture, sculpture, engraving or
   lithography; a photographic work to which are assimilated works
   expressed by a process analogous to photography; a work of applied
   art; an illustration, map, plan, sketch or three-dimensional work
   relative to geography, topography, architecture or science; a
   performance; a broadcast; a phonogram; a compilation of data to the
   extent it is protected as a copyrightable work; or a work performed by
   a variety or circus performer to the extent it is not otherwise
   considered a literary or artistic work.
i. ”You” means an individual or entity exercising rights under this
   License who has not previously violated the terms of this License with
   respect to the Work, or who has received express permission from the
   Licensor to exercise rights under this License despite a previous
   violation.
j. ”Publicly Perform” means to perform public recitations of the Work and
   to communicate to the public those public recitations, by any means or
   process, including by wire or wireless means or public digital
   performances; to make available to the public Works in such a way that
   members of the public may access these Works from a place and at a
   place individually chosen by them; to perform the Work to the public
   by any means or process and the communication to the public of the
   performances of the Work, including by public digital performance; to
   broadcast and rebroadcast the Work by any means including signs,
   sounds or images.
k. ”Reproduce” means to make copies of the Work by any means including
   without limitation by sound or visual recordings and the right of
   fixation and reproducing fixations of the Work, including storage of a
   protected performance or phonogram in digital form or other electronic
   medium.

2. Fair Dealing Rights. Nothing in this License is intended to reduce,
```

(continues on next page)

(continued from previous page)

```
limit, or restrict any uses free from copyright or rights arising from
limitations or exceptions that are provided for in connection with the
copyright protection under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License,
Licensor hereby grants You a worldwide, royalty-free, non-exclusive,
perpetual (for the duration of the applicable copyright) license to
exercise the rights in the Work as stated below:

 a. to Reproduce the Work, to incorporate the Work into one or more
    Collections, and to Reproduce the Work as incorporated in the
    Collections;
 b. to create and Reproduce Adaptations provided that any such Adaptation,
    including any translation in any medium, takes reasonable steps to
    clearly label, demarcate or otherwise identify that changes were made
    to the original Work. For example, a translation could be marked ”The
    original work was translated from English to Spanish,” or a
    modification could indicate ”The original work has been modified.”;
 c. to Distribute and Publicly Perform the Work including as incorporated
    in Collections; and,
 d. to Distribute and Publicly Perform Adaptations.
 e. For the avoidance of doubt:

     i. Non-waivable Compulsory License Schemes. In those jurisdictions in
        which the right to collect royalties through any statutory or
        compulsory licensing scheme cannot be waived, the Licensor
        reserves the exclusive right to collect such royalties for any
        exercise by You of the rights granted under this License;
    ii. Waivable Compulsory License Schemes. In those jurisdictions in
        which the right to collect royalties through any statutory or
        compulsory licensing scheme can be waived, the Licensor waives the
        exclusive right to collect such royalties for any exercise by You
        of the rights granted under this License; and,
   iii. Voluntary License Schemes. The Licensor waives the right to
        collect royalties, whether individually or, in the event that the
        Licensor is a member of a collecting society that administers
        voluntary licensing schemes, via that society, from any exercise
        by You of the rights granted under this License.

The above rights may be exercised in all media and formats whether now
known or hereafter devised. The above rights include the right to make
such modifications as are technically necessary to exercise the rights in
other media and formats. Subject to Section 8(f), all rights not expressly
granted by Licensor are hereby reserved.

4. Restrictions. The license granted in Section 3 above is expressly made
```

(continues on next page)

(continued from previous page)

```
subject to and limited by the following restrictions:

 a. You may Distribute or Publicly Perform the Work only under the terms
    of this License. You must include a copy of, or the Uniform Resource
    Identifier (URI) for, this License with every copy of the Work You
    Distribute or Publicly Perform. You may not offer or impose any terms
    on the Work that restrict the terms of this License or the ability of
    the recipient of the Work to exercise the rights granted to that
    recipient under the terms of the License. You may not sublicense the
    Work. You must keep intact all notices that refer to this License and
    to the disclaimer of warranties with every copy of the Work You
    Distribute or Publicly Perform. When You Distribute or Publicly
    Perform the Work, You may not impose any effective technological
    measures on the Work that restrict the ability of a recipient of the
    Work from You to exercise the rights granted to that recipient under
    the terms of the License. This Section 4(a) applies to the Work as
    incorporated in a Collection, but this does not require the Collection
    apart from the Work itself to be made subject to the terms of this
    License. If You create a Collection, upon notice from any Licensor You
    must, to the extent practicable, remove from the Collection any credit
    as required by Section 4(c), as requested. If You create an
    Adaptation, upon notice from any Licensor You must, to the extent
    practicable, remove from the Adaptation any credit as required by
    Section 4(c), as requested.
 b. You may Distribute or Publicly Perform an Adaptation only under the
    terms of: (i) this License; (ii) a later version of this License with
    the same License Elements as this License; (iii) a Creative Commons
    jurisdiction license (either this or a later license version) that
    contains the same License Elements as this License (e.g.,
    Attribution-ShareAlike 3.0 US)); (iv) a Creative Commons Compatible
    License. If you license the Adaptation under one of the licenses
    mentioned in (iv), you must comply with the terms of that license. If
    you license the Adaptation under the terms of any of the licenses
    mentioned in (i), (ii) or (iii) (the ”Applicable License”), you must
    comply with the terms of the Applicable License generally and the
    following provisions: (I) You must include a copy of, or the URI for,
    the Applicable License with every copy of each Adaptation You
    Distribute or Publicly Perform; (II) You may not offer or impose any
    terms on the Adaptation that restrict the terms of the Applicable
    License or the ability of the recipient of the Adaptation to exercise
    the rights granted to that recipient under the terms of the Applicable
    License; (III) You must keep intact all notices that refer to the
    Applicable License and to the disclaimer of warranties with every copy
    of the Work as included in the Adaptation You Distribute or Publicly
    Perform; (IV) when You Distribute or Publicly Perform the Adaptation,
    You may not impose any effective technological measures on the
```

(continues on next page)

(continued from previous page)

Adaptation that restrict the ability of a recipient of the Adaptation from You to exercise the rights granted to that recipient under the terms of the Applicable License. This Section 4(b) applies to the Adaptation as incorporated in a Collection, but this does not require the Collection apart from the Adaptation itself to be made subject to the terms of the Applicable License.

c. If You Distribute, or Publicly Perform the Work or any Adaptations or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution (”Attribution Parties”) in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and (iv) , consistent with Ssection 3(b), in the case of an Adaptation, a credit identifying the use of the Work in the Adaptation (e.g., ”French translation of the Work by Original Author,” or ”Screenplay based on original Work by Original Author”). The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Adaptation or Collection, at a minimum such credit will appear, if a credit for all contributing authors of the Adaptation or Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

d. Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Adaptations or Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation. Licensor agrees that in those jurisdictions (e.g. Japan), in which any exercise of the right granted in Section 3(b) of this License (the right to make Adaptations) would be deemed to be a distortion, mutilation,

(continues on next page)

(continued from previous page)

```
    modification or other derogatory action prejudicial to the Original
    Author's honor and reputation, the Licensor will waive or not assert,
    as appropriate, this Section, to the fullest extent permitted by the
    applicable national law, to enable You to reasonably exercise Your
    right under Section 3(b) of this License (right to make Adaptations)
    but not otherwise.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR
OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY
KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE,
INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY,
FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF
LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS,
WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION
OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE
LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR
ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES
ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS
BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

 a. This License and the rights granted hereunder will terminate
    automatically upon any breach by You of the terms of this License.
    Individuals or entities who have received Adaptations or Collections
    from You under this License, however, will not have their licenses
    terminated provided such individuals or entities remain in full
    compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will
    survive any termination of this License.
 b. Subject to the above terms and conditions, the license granted here is
    perpetual (for the duration of the applicable copyright in the Work).
    Notwithstanding the above, Licensor reserves the right to release the
    Work under different license terms or to stop distributing the Work at
    any time; provided, however that any such election will not serve to
    withdraw this License (or any other license that has been, or is
    required to be, granted under the terms of this License), and this
    License will continue in full force and effect unless terminated as
    stated above.

8. Miscellaneous

 a. Each time You Distribute or Publicly Perform the Work or a Collection,
```

(continues on next page)

(continued from previous page)

```
    the Licensor offers to the recipient a license to the Work on the same
    terms and conditions as the license granted to You under this License.
 b. Each time You Distribute or Publicly Perform an Adaptation, Licensor
    offers to the recipient a license to the original Work on the same
    terms and conditions as the license granted to You under this License.
 c. If any provision of this License is invalid or unenforceable under
    applicable law, it shall not affect the validity or enforceability of
    the remainder of the terms of this License, and without further action
    by the parties to this agreement, such provision shall be reformed to
    the minimum extent necessary to make such provision valid and
    enforceable.
 d. No term or provision of this License shall be deemed waived and no
    breach consented to unless such waiver or consent shall be in writing
    and signed by the party to be charged with such waiver or consent.
 e. This License constitutes the entire agreement between the parties with
    respect to the Work licensed here. There are no understandings,
    agreements or representations with respect to the Work not specified
    here. Licensor shall not be bound by any additional provisions that
    may appear in any communication from You. This License may not be
    modified without the mutual written agreement of the Licensor and You.
 f. The rights granted under, and the subject matter referenced, in this
    License were drafted utilizing the terminology of the Berne Convention
    for the Protection of Literary and Artistic Works (as amended on
    September 28, 1979), the Rome Convention of 1961, the WIPO Copyright
    Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996
    and the Universal Copyright Convention (as revised on July 24, 1971).
    These rights and subject matter take effect in the relevant
    jurisdiction in which the License terms are sought to be enforced
    according to the corresponding provisions of the implementation of
    those treaty provisions in the applicable national law. If the
    standard suite of rights granted under applicable copyright law
    includes additional rights not granted under this License, such
    additional rights are deemed to be included in the License; this
    License is not intended to restrict the license of any rights under
    applicable law.


Creative Commons Notice

    Creative Commons is not a party to this License, and makes no warranty
    whatsoever in connection with the Work. Creative Commons will not be
    liable to You or any party on any legal theory for any damages
    whatsoever, including without limitation any general, special,
    incidental or consequential damages arising in connection to this
    license. Notwithstanding the foregoing two (2) sentences, if Creative
    Commons has expressly identified itself as the Licensor hereunder, it
```

(continues on next page)

(continued from previous page)

```
shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the
Work is licensed under the CCPL, Creative Commons does not authorize
the use by either party of the trademark ”Creative Commons” or any
related trademark or logo of Creative Commons without the prior
written consent of Creative Commons. Any permitted use will be in
compliance with Creative Commons' then-current trademark usage
guidelines, as may be published on its website or otherwise made
available upon request from time to time. For the avoidance of doubt,
this trademark restriction does not form part of the License.

Creative Commons may be contacted at https://creativecommons.org/.
```

A.2 Credits

Listing 1.2: Authors cited when creating the Li-Pro.Net Sphinx Primer

```
_____________________________________________________________________________

AUTHORS (in alphabetical order)

_____________________________________________________________________________

Armin Ronacher <armin.ronacher@active-4.com>  Sphinx Code and Documentation
Bareos GmbH & Co. KG <info@bareos.org>        Bareos Documentation
David Goodger <goodger@python.org>            Docutils and reStructuredText
Eric Holscher <eric@ericholscher.com>         Sphinx Tutorial
Georg Brandl <georg@python.org>               Sphinx Code and Documentation
Richard Jones <rjones@ekit-inc.com>           ReStructuredText Primer
Stephan Linz <linz@li-pro.net>                Li-Pro.Net Sphinx Primer
_____________________________________________________________________________
```

A.3 The Gimmick

The gimmick for your pastime.

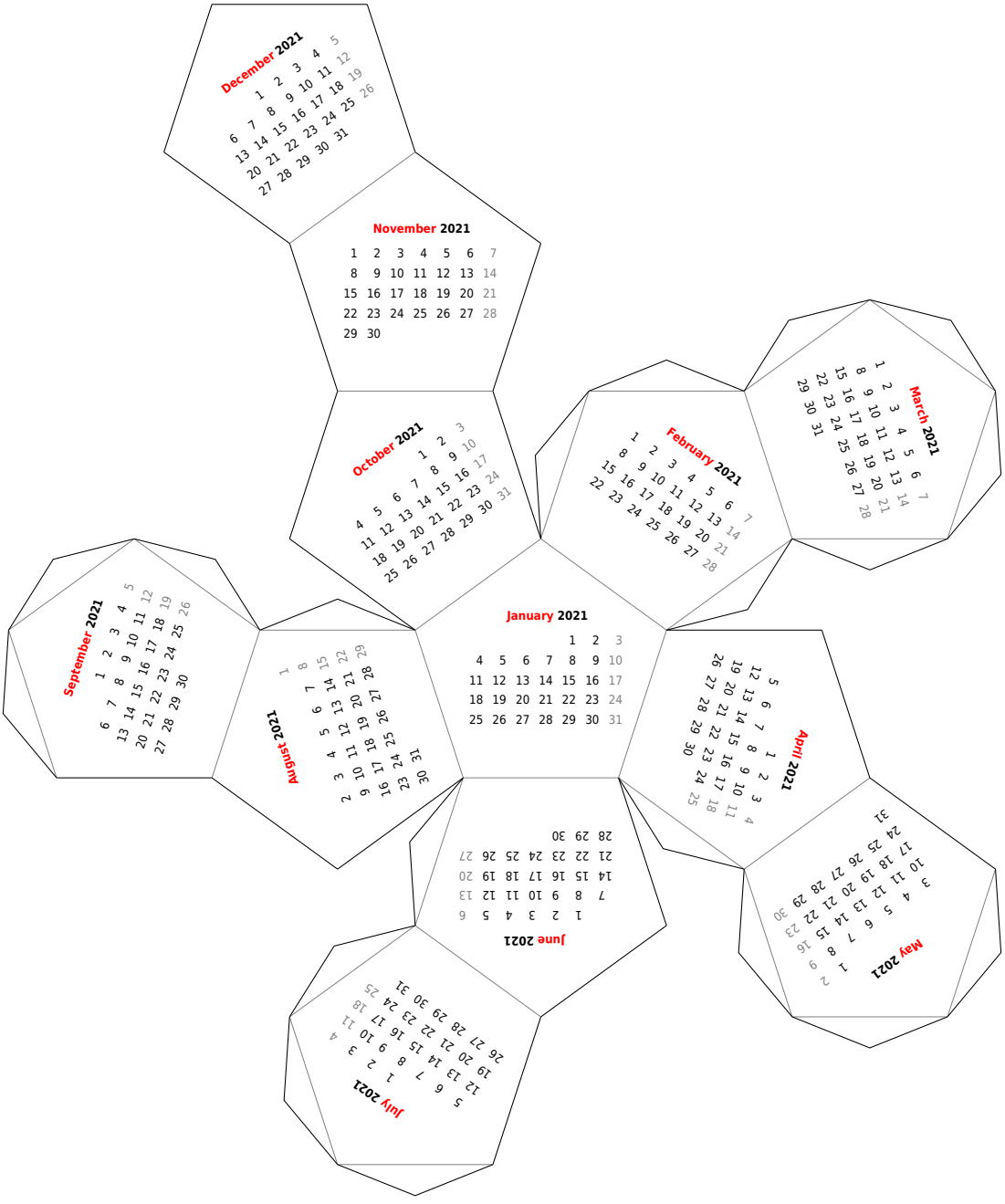


Figure 1.1: A new year is coming soon ...

B Glossary

Section author: Stephan Linz <linz@li-pro.net>

B.1 Terms

B.1.1 Commons

Docutils Docutils is an open-source text processing system for processing plaintext documentation into useful formats, such as *HTML*, *LaTeX*, man-pages, open-document or *XML*. It includes *reStructuredText*, the easy to read, easy to use, what-you-see-is-what-you-get plaintext markup language.

See also:

- English Wikipedia: reStructuredText

LaTeX LaTeX is a document preparation system for high-quality typesetting. It is most often used for medium-to-large technical or scientific documents but it can be used for almost any form of publishing. LaTeX uses the *TeX* typesetting program for formatting its output, and is itself written in the *TeX* macro language.

See also:

- English Wikipedia: LaTeX

Pybtex Pybtex is a *BibTeX*-compatible bibliography processor written in *Python*. Pybtex aims to be 100% compatible with *BibTeX*. It accepts the same command line options, fully supports *BibTeX*'s `.bst` styles and produces byte-identical output. Additionally, Pybtex is Unicode aware and Pybtex supports bibliography formats other than *BibTeX*: BibTeXML (BibTeX as XML) and YAML (YAML Ain't Markup Language).

PyEnchant PyEnchant is a *Python* binding for *Enchant*.

Pygments Pygments is a generic syntax highlighter written in *Python* which supports a wide range of over 500 languages with related lexers and other text formats and is ready for new languages and formats added easily.

ReportLab ReportLab Toolkit is an Open Source *Python* library for generating *PDF*s and graphics.

See also:

- https://www.reportlab.com/opensource/
- https://www.reportlab.com/dev/docs/
- https://hg.reportlab.com/hg-public/
- https://pypi.org/project/reportlab/

reStructuredText reStructuredText (**RST**, **ReST**, or **reST**) is a file format for textual data used primarily in the *Python* programming language community for technical documentation. It is part of the *Docutils* project of the *Python* Doc-SIG (Documentation Special Interest Group).

See also:

- English Wikipedia: reStructuredText

Sphinx Sphinx is a documentation generator written and used by the *Python* community. It is written in *Python*, and also used in other environments. Sphinx converts *reStructuredText* files into *HTML* websites and other formats including PDF, EPub, Texinfo and man.

reStructuredText is extensible, and Sphinx exploits its extensible nature through a number of extensions–for auto generating documentation from source code, writing mathematical notation or highlighting source code, etc.

See also:

- English Wikipedia: Sphinx (documentation generator)

B.1.2 Operating Systems

Linux

GNU (GNU's Not Unix!)/Linux Linux (trademarked as Linux™, handled by the LMI (Linux Mark Institute)) is a family of open source *Unix*-like operating systems based on the Linux kernel, an operating system kernel first released on September 17, 1991, by Linus Torvalds. Linux is typically packaged in a Linux distribution. Distributions include the Linux kernel and supporting system software and libraries, many of which are provided by the GNU Project. Many Linux distributions use the word *"Linux"* in their name, but the FSF (Free Software Foundation) uses the name **GNU/Linux** to emphasize the importance of GNU software, causing some controversy.

See also:

- English Wikipedia: Linux

POSIX (Portable Operating System Interface) POSIX is a family of standards specified by the IEEE (Institute of Electrical and Electronics Engineers) Computer Society for maintaining compatibility between operating systems. POSIX defines the API (Application Programming Interface), along with command line shells and utility interfaces, for software compatibility with variants of *Unix* and other operating systems. POSIX has been standardized by the *Austin Group* since 1997 (SUS (Single UNIX Specification)), and later since 2001 by the IEEE Computer Society (**IEEE Std 1003.1**). Before 1997, POSIX comprised several standards.

See also:

- English Wikipedia: POSIX
- POSIX.1-2017 (**POSIX.1-2008 with Technical Corrigenda 1 and 2**): standard ratified in 2017 as **IEEE Std 1003.1-2017**
- POSIX.1-2008 (**Base Specifications, Issue 7**): standard ratified in 2008 as *IEEE Std 1003.1-2008*
- POSIX.1-2001 (**Single UNIX Specification version 3**): standard ratified in 2001 as *IEEE Std 1003.1-2001*
- POSIX.1c (**Threads Extensions**): initially standardized in 1995 as *IEEE Std 1003.1c-1995*
- POSIX.1b (**Real-Time Extensions**): initially standardized in 1993 as *IEEE Std 1003.1b-1993*
- POSIX.2 (**Shell and Utilities**): initially standardized in 1992 as *IEEE Std 1003.2-1992*
- POSIX.1 (**Core Services**): initially standardized in 1988 as *IEEE Std 1003.1-1988*

Unix Unix (trademarked as UNIX™) is a family of multitasking, multiuser computer operating systems that derive from the original AT&T (American Telephone and Telegraph Company) Unix, development starting in the 1970s at the Bell Labs research center by Ken Thompson, Dennis Ritchie, and others. The origins of Unix date back to the mid-1960s when the MIT (Massachusetts Institute of Technology), Bell Labs and GE (General Electric) were developing *Multics*, a time-sharing operating system for a 36-bit mainframe computer. In the late 1980s, an open operating system standardization effort now known as *POSIX* provided a common baseline for all operating systems; IEEE based *POSIX* around the common structure of the major competing variants of the Unix system, publishing the first *POSIX* standard in 1988. In the early 1990s, a separate but very similar effort was started by an industry consortium, the COSE (Common Open Software Environment) initiative, which eventually became the SUS administered by *The Open Group*. Starting in 1998, *The Open Group* and IEEE started the *Austin Group*, to provide a common definition of *POSIX* and the SUS, which, by 2008, had become the *Open Group Base Specification*.

See also:

- English Wikipedia: Unix

B.1.3 Programming Languages

C C is a general-purpose, imperative procedural computer programming language supporting structured programming, lexical variable scope, and recursion, with a static type system. It was designed to be compiled to provide low-level access to memory and language constructs that map efficiently to machine instructions, all with minimal runtime support. C has been standardized by the ANSI (American National Standards Institute) *X3J11* since 1989 (**ANSI C**) and by the ISO (International Organization for Standardization)/IEC (International Electrotechnical Commission) JTC1/SC22/WG14 (**ISO C**).

See also:

- English Wikipedia: C (programming language)
- English Wikipedia: Compatibility of C and C++
- English Wikipedia: C18 (C standard revision): standard ratified in 2018 as **ISO/IEC 9899:2018**
- English Wikipedia: C11 (C standard revision): standard ratified in 2011 as *ISO/IEC 9899:2011*
- English Wikipedia: C99: standard ratified in 1999 as *ISO/IEC 9899:1999*
- English Wikipedia: C95 (C version): Amendment 1 ratified in 1995 as *ISO/IEC 9899:1990/AMD1:1995*
- English Wikipedia: C90 (C version): standard ratified in 1990 as *ISO/IEC 9899:1990*
- English Wikipedia: C89 (C version): standard ratified in 1989 as *ANSI X3.159-1989*

C++ C++ is a general-purpose programming language as an extension of the *C* programming language, or *"C with Classes"*. Modern C++ implementations now has object-oriented, generic, and functional features in addition to facilities for low-level memory manipulation. C++ is standardized by the ISO/IEC JTC1/SC22/WG14 since 1998.

See also:

- English Wikipedia: C++
- English Wikipedia: Compatibility of C and C++
- English Wikipedia: C++17: standard ratified in 2017 as **ISO/IEC 14882:2017**
- English Wikipedia: C++14: standard ratified in 2014 as *ISO/IEC 14882:2014*
- English Wikipedia: C++11: standard ratified in 2011 as *ISO/IEC 14882:2011*
- English Wikipedia: C++03: standard ratified in 2003 as *ISO/IEC 14882:2003*
- initially standardized in 1998 as *ISO/IEC 14882:1998*

ES (ECMAScript)

ECMAScript ES is a general-purpose programming language, standardized by Ecma International since 1997 according to the document ECMA-262. It is a *JavaScript* standard meant to ensure

the interoperability of Web pages across different Web browsers. ES is standardized by the ISO/IEC JTC1/SC22 since 1998.

See also:

- English Wikipedia: ECMAScript
- English Wikipedia: ECMAScript engine
- English Wikipedia: List of ECMAScript engines
- ES Edition 11: standard ratified in 2020 as **ECMA-262-11:2020**
- ES Edition 5.1: standard ratified in 2011 as *ISO/IEC 16262:2011*
- ES Edition 2: initially standardized in 1998 as *ISO/IEC 16262:1998*

JS (JavaScript)

JavaScript JS is a programming language that conforms to the *ECMAScript* specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

Alongside *HTML* and *CSS*, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it for client-side page behavior, and all major web browsers have a dedicated JavaScript engine to execute it.

See also:

- English Wikipedia: JavaScript
- English Wikipedia: JavaScript engine
- English Wikipedia: List of JavaScript engines

Python Python is an interpreted, high-level and general-purpose programming language. Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, a free and open-source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

CPython is the reference implementation of Python. It is written in *C*, meeting the *C*89 standard with several select *C*99 features. Python's development is conducted largely through the PEP process, the primary mechanism for proposing major new features, collecting community input on issues and documenting Python design decisions. Python coding style is covered in **PEP 8**.

See also:

- English Wikipedia: Python (programming language)
- English Wikipedia: CPython
- [Swe19]

B.1.4 Technologies

Badges Web buttons, badges or stickers are small images in some WWW (World Wide Web) pages which are typically used to promote programs that were used to create or host the site. They may also be used to promote compliance with web standards such as passing W3C (World Wide Web Consortium) *HTML* validation or to comply with an application's terms of service or present the status of any 3rd party service like CI/CD pipelines or review processes.

See also:

- English Wikipedia: Web badge

BibTeX BibTeX is a widely used bibliography management tool in *LaTeX*, with BibTeX the bibliography entries are kept in a separate file and then imported into the main document.

See also:

- English Wikipedia: BibTeX

CircuiTikZ CircuiTikZ is a *LaTeX* package which provides a set of macros for naturally typesetting electrical and electronic networks. Designed as a tool that is easy to use by native to a lean *LaTeX* syntax it has therefore been based on the very impressive *PGF/TikZ* package. CircuiTikZ was initiated as tool for creating exercises and exams. The use of CircuiTikZ is, of course, not limited to academic teaching. The package gets widely used by engineers for typesetting electronic circuits for articles and publications all over the world.

See also:

- CircuiTikZ Package
- CircuiTikZ Manual
- CircuiTikZ Home

CORBA (Common Object Request Broker Architecture)

MOF (Meta-Object Facility) MOF is an OMG (Object Management Group) standard for model-driven engineering. Its purpose is to provide a type system for entities in the *CORBA* architecture and a set of interfaces through which those types can be created and manipulated.

See also:

- *OCL* and *QVT*
- English Wikipedia: Meta-Object Facility
- MOF 2.5.1: initially standardized in 2016 **OMG Meta Object Facility Core Specification 2.5.1 2016/10/01**
- MOF 2.5: initially standardized in 2015 *OMG Meta Object Facility Core Specification 2.5 2015/06/05*
- ISO 19508:2014: **MOF 2.4.2** standard formally published in 2014 **ISO/IEC 19508:2014(E) 2014/04/05**

- MOF 2.4.2: initially standardized in 2014 *OMG Meta Object Facility Core Specification 2.4.2 2014/04/03*
- MOF 2.4.1: initially standardized in 2013 *OMG Meta Object Facility Core Specification 2.4.1 2013/06/01*
- MOF 2.0: initially standardized in 2006 *Meta Object Facility Core Specification 2.0 2006/01/01*
- ISO 19502:2005: *MOF 1.4.1* standard formally published in 2005 *ISO/IEC 19502:2005(E) 2005/05/05*
- MOF 1.4: initially standardized in 2002 *Meta Object Facility Specification 1.4 2002/04/03*

CSS (Cascading Style Sheets) CSS is a style sheet language used for describing the presentation of a document written in a markup language like *HTML*. CSS is a cornerstone technology of the WWW, alongside *HTML* and *JavaScript*. In addition to *HTML*, other markup languages support the use of CSS including plain *XML* and *SVG*. The CSS specifications is standardized by the W3C/TR/CSS since 1996.

See also:

- English Wikipedia: CSS
- CSS 2.1: standard ratified in 2011 **W3C REC-CSS2-20110607**
- CSS 2.0: standard ratified in 1998 *W3C REC-CSS2-19980512*
- CSS 1.0: initially standardized in 1996 *W3C REC-CSS1-961217*

Enchant Enchant is a free software project developed as part of the AbiWord word processor with the aim of unifying access to the various existing spell-checker software.

See also:

- English Wikipedia: Enchant (software)

ePub (Electronic Publication)

EPUB (Electronic Publication) EPUB is an e-book file format that uses the `.epub` file extension. The term is short for electronic publication and is sometimes styled **ePub**. EPUB is a technical standard published by the IDPF (International Digital Publishing Forum) and became an official standard in 2007, superseding the older OEB (Open eBook) standard. EPUB is also standardized by the ISO/IEC JTC1/SC34 since 2014.

See also:

- English Wikipedia: EPUB
- EPUB 3.2: standard ratified in 2019 **EPUB 3.2**
- EPUB 3.0.1: standard ratified in 2020 **ISO/IEC 23736 (parts 1-6)**
- EPUB 3.0.1: standard ratified in 2014 *OPS 3.0.1*
- EPUB 3.0: standard ratified in 2014 *ISO/IEC TS 30135 (parts 1-7)*

- EPUB 3.0: standard ratified in 2011 *OPS 3.0*
- EPUB 2.0.1: standard ratified in 2010 *OPS 2.0.1 v1.0.1*
- EPUB 2.0: initially standardized in 2007 *OPS 2.0 v1.0*

HTML (Hypertext Markup Language) HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as *CSS* and scripting languages such as *JavaScript*. The HTML specifications is standardized by the W3C/TR/HTML since 1997 and ISO/IEC JTC1/SC34 since 1998.

See also:

- English Wikipedia: HTML
- English Wikipedia: HTML5: latest live standard was released in 2017 **W3C REC-HTML52-20171214**
- English Wikipedia: HTML4: standard ratified in 1999 *W3C REC-HTML40* and 2000 *ISO/IEC 15445:2000*
- English Wikipedia: HTML3: standard ratified in 1997 *W3C REC-HTML32*
- English Wikipedia: HTML2: initially standardized in 1995 as *RFC 1866* (**RFC 1866**)

OCL (Object Constraint Language) OCL is a declarative language describing rules applying to *UML* models and is now part of the *UML* standard but as separate document. Initially, OCL was merely a formal specification language extension for *UML*. OCL may now be used with any *MOF* meta-model, including *UML*, and is a precise text language that provides constraint and object query expressions on any such kind of meta-model that cannot otherwise be expressed by diagrammatic notation. OCL is a key component of the new OMG standard recommendation for transforming models, the *QVT* specification.

See also:

- *UML*, *MOF* and *QVT*
- English Wikipedia: Object Constraint Language
- OCL 2.4: standard ratified in 2014 **Object Constraint Language 2.4 2014/02/03**
- ISO 19507:2012: **OCL 2.3.1** standard formally published in 2012 **ISO/IEC 19507:2012(E) 2012/05/09**
- OCL 2.3.1: standard ratified in 2011 *Object Constraint Language 2.3.1 2012/01/01*
- OCL 2.2: standard ratified in 2010 *Object Constraint Language 2.2 2010/02/01*
- OCL 2.0: standard ratified in 2006 *Object Constraint Language 2.0 2006/05/01*
- UML 1.3 (*Chapter 7*): initially standardized in 2000

PDF (Portable Document Format) PDF is a file format developed by Adobe in 1993 to present documents, including text formatting and images, in a manner independent of application software, hardware, and operating systems. Based on the PS language, each PDF file encapsulates

a complete description of a fixed-layout flat document, including the text, fonts, vector graphics, raster images and other information needed to display it. PDF is standardized by the ISO TC171/SC2/WG8 since 2008, and no longer requires any royalties for its implementation.

ISO standardized subsets of PDF:

- English Wikipedia: PDF/X: since 2001, series of *ISO 15929* and *ISO 15930* standards
- English Wikipedia: PDF/A: since 2005, series of *ISO 19005* standards
- English Wikipedia: PDF/E: since 2008, series of *ISO 24517*
- English Wikipedia: PDF/VT: since 2010, *ISO 16612-2*
- English Wikipedia: PDF/UA: since 2012, *ISO 14289-1*

See also:

- English Wikipedia: PDF
- English Wikipedia: History of the Portable Document Format (PDF)
- PDF 2.0: standard ratified in 2017 as **ISO 32000-2:2017**
- PDF 1.7: initially standardized in 2008 as *ISO 32000-1:2008*

PGF (Portable Graphic Format)

TikZ (TikZ ist kein Zeichenprogramm)

PGF/TikZ PGF/TikZ is a pair of languages for producing vector graphics (for example: technical illustrations and drawings) from a geometric/algebraic description, with standard features including the drawing of points, lines, arrows, paths, circles, ellipses and polygons. PGF, is a lower-level language, while TikZ, which is written in *TeX*, is a set of higher-level macros that use PGF.

See also:

- English Wikipedia: PGF/TikZ
- PGF/TikZ Package and VisualTikZ Package
- PGF/TikZ Manual and VisualTikZ Manual
- PGF/TikZ Introduction
- PGF/TikZ Home and VisualTikZ Home

PNG (Portable Network Graphics) PNG is a raster-graphics file format that supports lossless data compression. PNG was developed as an improved, non-patented replacement for GIF with support for interactivity and animation. The PNG specification is standardized by the W3C/TR/PNG since 1996 and ISO/IEC JTC1/SC24/WG7 since 2003 as an open standard.

See also:

- English Wikipedia: PNG
- PNG 1.2: standard ratified in 2004 **ISO/IEC 15948:2004**
- PNG 1.2: standard ratified in 2003 *REC-PNG-20031110*

- PNG 1.0: initially standardized in 1996 as *RFC 2083* (**RFC 2083**)

QVT (Query/View/Transformation) QVT is a standard set of languages for model transformation defined by the OMG.

See also:

- *OCL* and *MOF*
- English Wikipedia: QVT
- QVT 1.3: standard ratified in 2016 **MOF 2.0 Query/View/Transformation Specification 1.3 2016/06/03**
- QVT 1.2: standard ratified in 2015 *MOF 2.0 Query/View/Transformation Specification 1.2 2015/02/01*
- QVT 1.1: standard ratified in 2011 *MOF 2.0 Query/View/Transformation Specification 1.1 2011/01/01*
- QVT 1.0: initially standardized in 2008 *MOF 2.0 Query/View/Transformation Specification 1.0 2008/04/03*

SVG (Scalable Vector Graphics) SVG is an *XML*-based vector image format for two-dimensional graphics with support for interactivity and animation. The SVG specification is standardized by the W3C/TR/SVG since 1999 as an open standard.

SVG drawings can be dynamic and interactive. Time-based modifications to the elements can be described in SMIL (Synchronized Multimedia Integration Language), or can be programmed in a scripting language (e.g. *ECMAScript* or *JavaScript*). The W3C explicitly recommends SMIL as the standard for animation in SVG.

See also:

- English Wikipedia: SVG
- SVG 2.0: latest standard draft was released in 2020
- SVG 1.1 Second Edition: standard ratified in 2011 **W3C REC-SVG11-20110816**
- SVG 1.1: standard ratified in 2003 *W3C REC-SVG11-20030114*
- SVG 1.0: initially standardized in 2001 *W3C REC-SVG-20010904*

SysML (Systems Modeling Language)

BPMN (Business Process Model and Notation)

UML (Unified Modeling Language) UML is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system. In 1997, UML was adopted as a standard by the OMG, and has been managed by this organization ever since. In 2005, UML was also published by ISO as an approved standard.

In 2007 with the release of UML 2.1.2, two new significant specifications for certain technology areas have split off. These are: *SysML*, *BPMN*.

See also:

- *OCL*, *MOF* and *QVT*
- English Wikipedia: Unified Modeling Language
- UML 2.5.1: standard adopted from 2.5 in 2017 **OMG Unified Modeling Language 2.5.1 2017/12/05**
- UML 2.5: standard released in 2012 and ratified in 2015 *OMG Unified Modeling Language 2.5 2015/03/01*
- ISO 19505-1:2012 and ISO 19505-2:2012: **UML 2.4.1** standard formally published in 2012 **ISO/IEC 19505-1:2012(E) 2012/05/06** and **ISO/IEC 19505-2:2012(E) 2012/05/07**
- UML 2.4.1 and UMLDI 1.0: standard ratified in 2011 *OMG UML Superstructure Specification 2.4.1 2011/08/06*, *OMG UML Infrastructure Specification 2.4.1 2011/08/05* and *OMG UML Diagram Interchange 1.0 2006/04/04*
- UML 2.4 and UMLDI 1.0: standard ratified in 2011 *OMG UML Superstructure Specification 2.4 2010/11/14*, *OMG UML Infrastructure Specification 2.4 2010/11/16* and *OMG UML Diagram Interchange 1.0 2006/04/04*
- UML 2.3 and UMLDI 1.0: standard ratified in 2010 *OMG UML Superstructure Specification 2.3 2010/05/05*, *OMG UML Infrastructure Specification 2.3 2010/05/03* and *OMG UML Diagram Interchange 1.0 2006/04/04*
- UML 2.2 and UMLDI 1.0: standard ratified in 2009 *OMG UML Superstructure Specification 2.2 2009/02/02*, *OMG UML Infrastructure Specification 2.2 2009/02/04* and *OMG UML Diagram Interchange 1.0 2006/04/04*
- UML 2.1.2 and UMLDI 1.0: standard ratified in 2007 *OMG UML Superstructure Specification 2.1.2 2007/11/02*, *OMG UML Infrastructure Specification 2.1.2 2007/11/04* and *OMG UML Diagram Interchange 1.0 2006/04/04*
- UML 2.1.1 and UMLDI 1.0: standard ratified in 2007 *OMG UML Superstructure Specification 2.1.1 2007/02/05*, *OMG UML Infrastructure Specification 2.1.1 2007/02/06* and *OMG UML Diagram Interchange 1.0 2006/04/04*
- UML 2.0: standard ratified in 2005 *OMG UML Superstructure Specification 2.0 2005/07/04* and *OMG UML Infrastructure Specification 2.0 2005/07/05*
- ISO 19501:2005: *UML 1.4.2* standard formally published in 2005 *ISO/IEC 19501:2005(E) 2005/04/01*
- UML 1.5: standard ratified in 2003 *OMG Unified Modeling Language Specification 1.5 2003/03/01*
- UML 1.4: standard ratified in 2001 *OMG Unified Modeling Language Specification 1.4 2001/09/07*
- UML 1.3: standard ratified in 2000 *OMG Unified Modeling Language Specification 1.3 2000/03/01*

- UML 1.2: standard ratified in 1999
- UML 1.1: initially standardized in 1997

TeX TeX is a computer language designed for use in typesetting system; in particular, for typesetting math and other technical material. It has been noted as one of the most sophisticated digital typographical systems and is also used for many other typesetting tasks, especially in the form of *LaTeX*, ConTeXt, and other macro packages.

See also:

- English Wikipedia: TeX

TikZ-Timing TikZ-Timing is a *LaTeX* package which provides macros and an environment to generate timing diagrams (digital waveforms). The *TikZ* package is used to produce the graphics.

See also:

- TikZ-Timing Package
- TikZ-Timing User Guide

TikZ-UML TikZ-UML is a *LaTeX* package with a set of specialized commands and environments in front of the wide range of possibilities given by the *PGF/TikZ* library to define a set of specific *UML* diagrams: class diagrams, use case diagrams, state-machine diagrams, sequence diagrams and component diagrams.

See also:

- TikZ-UML Project
- TikZ-UML User Guide

XML (Extensible Markup Language) XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures. Several schema systems exist to aid in the definition of XML-based languages. The XML specification is standardized by the W3C/TR/XML since 1998 as an open standard.

See also:

- English Wikipedia: XML
- XML 1.1 Second Edition: standard ratified in 2006 **W3C REC-XML11-20060816**
- XML 1.1: standard ratified in 2004 *W3C REC-XML11-20040204*
- XML 1.0 Fifth Edition: standard ratified in 2008 *W3C REC-XML-20081126*
- XML 1.0: initially standardized in 1998 *W3C REC-SVG-20010904*

Listings

List of Tables

List of Figures

List of Equations

List of Downloads

Note: *List of Downloads* is not fully supported for *LaTeX*. All entries in the list are not linked and can not be provided together with the document.

Legal Notice of Li-Pro.Net Sphinx Primer

All artifacts were selected and download by using this reference URLs:

- `LICENSE`
- `CREDITS`

Extensions

All artifacts were selected and download by using this reference URLs:

- **PGF/TikZ examples**
  1. `pgftikz/shapesyms.tex`, used for: *Shapes and symbols* (page 105)
  2. `pgftikz/matrix.tex`, used for: *Shapes absolut and in a matrix positioned* (page 106)
  3. `pgftikz/fancyeye.tex`, used for: *Drawing a constructive but realistic eye* (page 106)
  4. `pgftikz/fourier.tex`, used for: *Time-frequency correspondence of the Fourier transformation* (page 107)
  5. `pgftikz/interference.tex`, used for: *Show constructive interferences in the time domain* (page 107)
  6. `pgftikz/modula-am.tex`, used for: *Graphical derivation of an amplitude modulated signal* (page 108)

7. `pgftikz/modula-fm.tex`, used for: *Graphical derivation of a frequency modulated signal* (page 108)
8. `pgftikz/flowchart.tex`, used for: *Geometrical shapes in a flowchart* (page 109)

- **CircuiTikZ examples**

1. `circuitikz/opamp-fullstab.tex`, used for: *Full differential Op-Amp stabilization principles* (page 110)
2. `circuitikz/opamp-inv.tex`, used for: *Detailed description of inverting Op-Amp principles* (page 111)
3. `circuitikz/nmos-fet.tex`, used for: *Drawing generic n-type MOSFET chip structure* (page 111)
4. `circuitikz/ne555-deedah.tex`, used for: *NE555 timer as "Dee-Dah" siren* (page 112)
5. `circuitikz/breadboard.tex`, used for: *Breadboard with NE555 timer as "Dee-Dah" siren* (page 113)
6. `circuitikz/lpc214x-lqfp64.tex`, used for: *LPC2144/2146/2148 pin assignment on LQFP64* (page 114)

- **TikZ-Timing examples**

1. `tikztiming/pcinst-tick.tex`, used for: *Program counter and instruction tick* (page 115)
2. `tikztiming/divider-line.tex`, used for: *Signal interconnections and dividers* (page 115)
3. `tikztiming/colored-cells.tex`, used for: *SPI vs. UART with arbitrary colored data cells* (page 116)
4. `tikztiming/spi-opmodes.tex`, used for: *Serial Peripheral Interface operating modes* (page 116)
5. `tikztiming/pci-read-irqack.tex`, used for: *PCI Read and Interrupt Acknowledge* (page 117)
6. `tikztiming/gn4124-pdwreq.tex`, used for: *GN4124 Packet Decoder Write Request* (page 117)

- **TikZ-UML examples**

1. `tikzuml/usecasediag.tex`, used for: *Use case diagram example* (page 128)
2. `tikzuml/classdiag.tex`, used for: *Class diagram example* (page 129)
3. `tikzuml/statediag.tex`, used for: *State-machine diagram example* (page 130)
4. `tikzuml/seqdiag.tex`, used for: *Sequence diagram example* (page 131)
5. `tikzuml/componentdiag.tex`, used for: *Component diagram example* (page 132)

- **TikZ Goodies examples**

1. **Timing-Diagrams**
    1. `tikzgoodies/timing-diagrams/basic.tex`, used for: *Basic Timing-Diagram example without additionals* (page 118)
    2. `tikzgoodies/timing-diagrams/timeline.tex`, used for: *Timeline-Diagram example with ticks and tasks* (page 119)
    3. `tikzgoodies/timing-diagrams/annotations.tex`, used for: *Timing-Diagram example with annotations* (page 119)
    4. `tikzgoodies/timing-diagrams/callouts.tex`, used for: *Timing-Diagram example with callouts (synchronizations)* (page 120)
    5. `tikzgoodies/timing-diagrams/arrows-compact.tex`, used for: *Timing-Diagram example with arrows (compact)* (page 120)
    6. `tikzgoodies/timing-diagrams/arrows-stretched.tex`, used for: *Timing-Diagram example with arrows (stretched)* (page 120)
    7. `tikzgoodies/timing-diagrams/events.tex`, used for: *Timing-Diagram example with events as vertical arrows* (page 121)
2. **Execution Stacks**
    1. `tikzgoodies/drawstack/minimal.tex`, used for: *Minimalistic execution stack example* (page 122)
    2. `tikzgoodies/drawstack/frames.tex`, used for: *Execution stack example with grouped cells in frames* (page 122)
    3. `tikzgoodies/drawstack/padding.tex`, used for: *Execution stack example with cell padding* (page 122)
    4. `tikzgoodies/drawstack/base-pointer.tex`, used for: *Execution stack example with base pointers* (page 123)
    5. `tikzgoodies/drawstack/stack-pointer.tex`, used for: *Execution stack example with stack pointers* (page 123)
    6. `tikzgoodies/drawstack/structure.tex`, used for: *Structures without a stack structure* (page 124)
    7. `tikzgoodies/drawstack/stack-structure.tex`, used for: *Structures and stack together* (page 124)
    8. `tikzgoodies/drawstack/highlighting.tex`, used for: *Execution stack example with highlighted note* (page 125)
    9. `tikzgoodies/drawstack/style.tex`, used for: *Execution stack example with changed style* (page 125)
3. **SystemC/TLM**

1. `tikzgoodies/tikzlm/platform.tex`, used for: *Platform show explicitly the sockets of the bus* (page 126)
2. `tikzgoodies/tikzlm/system.tex`, used for: *System show sockets connected through the bus* (page 126)
3. `tikzgoodies/tikzlm/memmap.tex`, used for: *System with memory map on the side* (page 127)

List of Issues (To-Do)

Todo: In the case of the *LaTeX*/*PDF* builder the usage of `:ref:`bibliography`` leads to an **invalid and unresolved reference**.

(The original entry (page 84) is located in /home/docs/checkouts/readthedocs.org/user_builds/lpn-doc-sphinx-primer/checkouts/0.0.5/source/extensions/bibtex.rst, line 30.)

Todo: Evaluate the integration and coexistence of this “Paneled Content” extension with all other. See `conf.py` for more details.

(The original entry (page 168) is located in /home/docs/checkouts/readthedocs.org/user_builds/lpn-doc-sphinx-primer/checkouts/0.0.5/source/extensions/sphinx-panels.rst, line 11.)

Bibliography

[CIT2002] This is the citation. It's just like a footnote, except the label is textual.

[Has19] Jan Ulrich Hasecke. *Software-Dokumentation mit Sphinx*. CreateSpace (was part of Amazon.com Inc.), today Kindle Direct Publishing (KDP), Seattle, United States of America, 2. edition, 2019. ISBN 1793008779. ISBN-10: 1-79300-877-9, ISBN-13: 978-1793008770, OCLC: 1190007667, URL: https://www.amazon.com/dp/1793008779 (March 2020).

[Swe19] Al Sweigart. *Automate the boring stuff with Python : practical programming for total beginners*. No Starch Press, San Francisco, United States of America, 2. edition, 2019. ISBN 1593279922. ISBN-10: 1-59327-992-2, ISBN-13: 978-1593279929, OCLC: 1140126955, URL: https://www.amazon.com/dp/1593279922 (September 2020).

Index

Symbols

A

B

C

D

E

M

N

O

P

Q

R

S

T

U

W

X