

Guidedog 手册

手册与完整 Odin API 参考

01 / 源文件

写出明确的含义。

02 / 结构

理清各部分的关系。

03 / 发布

屏幕与纸上都易读。

目录

教程	3
第一个项目	3
制作一本小书	4
专题指南	7
编写文档	7
域	12
项目如何工作	16
数学与证明	17
用图表解释概念	18
适合阅读的排版	19
模板	20
图片、视频、音频与下载	26
国际化	27
链接到其他项目	30
声明对象类型	31
用 autodoc 编写 Python 文档	32
为 Odin 生成文档	34
实用任务	39
构建并安装 Guidedog	39
构建并发布文档	40
迁移 Sphinx 项目	41
为代码库生成文档	41
维护本项目的文档	42
参考	43
命令行参考	43
配置参考	47
读取器、渲染器与构建器	53
错误信息	54
与 Sphinx 的兼容性	57
内部机制	61
Guidedoc 引擎	61
先有不变式，再谈算法	62
发布完整的一代输出	63
内存与所有权	64
关于本手册	67
构建验证记录	67
Guidedog	69
创建项目	69

安装	69
从源码构建	70
用 Guidedog 翻译	70
阅读与嵌入	71
致谢与许可	71
API Reference	73
autodoc	74
cli	78
core	80
defaults	134
fetch	135
gds	137
gettext	143
graphdog	159
highlight	166
highlight/guidedoc	186
host	187
i18n	214
inventory	223
jinja	226
odindoc	240
project	248
pyscan	272
readers/commonmark	287
readers/rst	299
readers/rst/inline	317
readers/sphinx	322
render/html	329
render/text	330
render/typst	330
treесitter	331
typst	337
索引	343

写一份源码，讲清楚道理，同时发布到网页与纸上。

Guidedog 将 reStructuredText 和 Markdown 源文件构建为 HTML 网站和 PDF 图书。它无需 Python 环境或外部 LaTeX 安装。转换引擎名为 **Guidedoc**，PDF 由 Typst 排版，网站由 Guidedog 直接生成。

小型项目只需三样东西：源文档、目录和 `conf.toml`。从 [第一个项目](#) 开始。具体任务查阅 [实用任务](#)，准确约定查阅 [参考](#)。

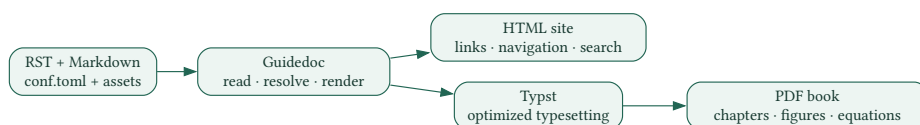


图 1 一套源文件，两种出版形式。

教程

按顺序阅读这些页面，每一步都有可检查的结果。

第一个项目

目标很小：在浏览器中显示一页，并将同一页输出为 PDF。先准备 Guidedog。PDF 可使用内置编译器，或安装 Typst 0.15.1 及以上版本。见 [构建并安装 Guidedog](#)。

在 macOS 上，运行 `brew install insanai/guidedog/guidedog` 安装发行版。此命令会添加官方 tap，安装包已包含 PDF 编译器。

创建项目

在尚不存在 `notes` 的目录中运行：

```
guidedog quickstart notes -q -p "Field Notes" -a "Your name" -v 1.0
```

Guidedog 写入配置、首份 RST 文档和可编辑模板，并为图片、音频、视频创建目录。它拒绝覆盖已有项目。

```
notes/  
  conf.toml  
  index.rst  
  _templates/  
    layout.html  
    book.typ  
  _static/  
    guidedog.css  
    guidedog.js  
  assets/  
    images/  
    audio/  
    video/  
  _build/
```

RST 文件保存正文，模板控制呈现。`quickstart` 会将 `_templates/layout.html` 和 `_templates/book.typ` 直接写入项目，而不是藏在外部主题或包缓存中。随时可查看和修改。

编写第一个页面

将 `notes/index.rst` 替换为：

```
Field Notes
=====

A useful explanation begins with a small example.

Measuring a square
-----

If the side has length :math:`s`, its area is :math:`s^2`.

.. math::

    A = s^2

.. note::

    State the units before comparing measurements.
```

构建与检查

```
guidedog build html notes -W
guidedog build pdf notes -W
```

打开 `notes/_build/html/index.html` 和 `notes/_build/pdf` 中的 PDF。`-w` 使警告导致构建失败，应保留在文档检查中。HTML 包含导航和正文；PDF 增加书籍的页眉、页脚等版式。公式按各目标格式排版，设置详见 [数学与证明](#)。

修改内容

```
guidedog serve notes
```

打开命令打印的地址，编辑源文件并保存，然后刷新浏览器。提供 HTML 前，Guidedog 检查源文件修改时间，若有更新则重建。它不会主动推送浏览器刷新。用 `Ctrl+C` 停止服务器。

不修改任何内容，再次构建。未变化的文档会复用，无需重新读取每一页来证明输出仍有效。接着阅读 [制作一本小书](#)。

制作一本小书

站点是文档组成的图，图书则需要阅读顺序。`toctree` 同时提供两者。

添加章节

创建 `notes/measurement.rst`：

```
Measurement
=====

A measurement has a value and a unit.
Write both. A bare number leaves the reader guessing.

.. _measurement-units:

Units
-----

Keep one unit within a comparison.
```

将 `notes/index.rst` 的内容替换为：


```
Field Notes
=====

These notes explain how to make a comparison that another person can check.

.. toctree::
   :maxdepth: 2
   :numbered:

   measurement

Begin with :doc:`measurement`.
For the unit rule, see :ref:`measurement-units`.
```

文档名省略 `.rst`。标签命名的是文档中的一个概念。链接整章时用文档链接，链接某段论证时用标签。

添加图表

将以下内容放入 `measurement.rst`：

```
.. graphviz::
   :caption: A comparison needs a common unit.
   :alt: Values are converted to one unit before they are compared.

   digraph comparison {
       rankdir=LR;
       "values" -> "common unit" -> "comparison";
   }
```

Guidedog 使用链接的 Graphviz 库。HTML 中显示 SVG，PDF 中显示图片，无需浏览器图表解释器。题注说明结论，替代文本解释图形内容。

构建书籍

```
guidedog build html notes -W
guidedog build pdf notes -W
```

侧边栏和图书目录中都会出现这一章。在浏览器中点击章节链接，并在 PDF 中检查其目标位置。

接下来可阅读 [编写文档](#)。保持章节简短，每章讲清一个主题。

专题指南

这些章节解释模型，为你的选择提供必要的背景。

编写文档

Guidedog 支持 reStructuredText (`.rst`) 和 MyST Markdown (`.md`)。两者都解析为同一种语义文档树，再生成 HTML 网站和 PDF 图书。

选择源文件语言

需要完整的域、复杂表格和指令时，使用 reStructuredText。偏好 Markdown 语法，同时需要 Sphinx 风格的指令和角色时，使用 MyST Markdown。

项目中的 Markdown 使用 MyST 方言。单文件转换 `guidedog convert` 默认使用 CommonMark，除非显式指定 MyST。

标题与结构

文档标题标识页面，章节标题组织正文。RST 标题用下划线符号，也可加相同的上划线。线条长度不得短于标题。

建议的标题符号：

```
Document Title
=====

Section Heading
-----

Subsection
~~~~~

Sub-subsection
^^^^^^^^^^^^^^

Paragraph Heading
.....
```

项目内的标题层级应保持一致。MyST Markdown 使用 `#`、`##`、`###` 等标准写法。

文字格式与行内标记

行内标记应表达含义，而不只是装饰：

表 1 行内标记

样式	reStructuredText	MyST Markdown
加粗	<code>**strong text**</code>	<code>**strong text**</code>
强调／斜体	<code>*emphasized text*</code>	<code>*emphasized text*</code>
行内代码	<code>`inline code`</code>	<code>`inline code`</code>
下标	<code>:sub:`text`</code>	<code>{sub}`text`</code>
上标	<code>:sup:`text`</code>	<code>{sup}`text`</code>

正文中的星号和反引号可用反斜杠转义，例如 `\*not italic\*`。

列表

Guidedog 支持四种列表：

无序列表

```
* First item
* Second item with multiple lines
  of continuing explanation.
* Third item
```

有序列表

```
1. Numbered item
2. Second item
#. Automatically numbered item
#. Next auto-numbered item
```

定义列表

定义列表将术语和说明配对。术语独占一行，紧接着是缩进的定义：

```
Workspace
  A bounded memory buffer supplied by the caller for document conversion.

Session
  A host-level coordinator that manages memory budgets and document caches.
```

字段列表

字段列表用于结构化元数据和参数说明：

```
:Authors: Jane Doe, John Smith
:Version: 1.2
:Status: Active
```

代码块

用 `code-block` 或 `sourcecode` 显示带语法高亮的代码。Guidedog 支持超过 35 种编程语言：

```
.. code-block:: python
:linenos:
:caption: Server entry point
:emphasize-lines: 2, 4-5

def main():
    app = create_app()
```

```
app.run(host="0.0.0.0", port=8080)
```

代码块选项：

- `:linenos:` 在代码旁显示行号。
- `:caption: Title:` 在代码块上方或下方添加说明。
- `:emphasize-lines: 1, 3-5:` 突出显示指定行。
- `:name: label:` 为代码块指定引用标签。

用 `literalinclude` 直接引入外部源文件：

```
.. literalinclude:: ../src/server.py
   :language: python
   :lines: 1-25
   :linenos:
```

提示块

提示块突出显示注释、警告和补充建议：

```
.. note::
    Helpful background context or implementation detail.

.. tip::
    Suggested best practices or workflow shortcuts.

.. important::
    Essential requirements that must not be overlooked.

.. warning::
    Conditions that could lead to unexpected behavior or lost work.

.. caution::
    Potential pitfalls or sensitive operational steps.

.. seealso::
    References to related chapters, specifications, or external guides.
```

Guidedog 也支持 `danger`、`error`、`hint` 和 `attention`。自定义标题的提示块使用 `admonition`：

```
.. admonition:: Design Rationale

    Explains why a particular architecture was chosen.
```

表格

Guidedog 支持简单表格、网格表格和指令生成的表格。

简单表格

简单表格用横线划分列的范围：

```
=====
A      B      A and B
=====
False  False  False
True   False  False
True   True   True
=====
```

网格表格

网格表格支持多行单元格和跨行、跨列合并：

```

+-----+-----+-----+
| Header row, column 1 | Column 2 | Column 3 |
+-----+-----+-----+
| Cell with multiple   | Second   | Third     |
| paragraphs of text.  | column   | column    |
+-----+-----+-----+

```

列表表格

`list-table` 用嵌套列表创建表格，便于在源码中阅读和维护较宽的表格：

```

.. list-table:: Project configurations
   :widths: 25 25 50
   :header-rows: 1

   * - Target
     - Builder
     - Description
   * - Website
     - ``html``
       - Static HTML documentation site
   * - Book
     - ``pdf``
       - Typeset PDF book powered by Typst

```

CSV 表格

`csv-table` 用逗号分隔的数据创建表格：

```

.. csv-table:: Comparative metrics
   :header: "Name", "Time (ms)", "Memory (MB)"
   :widths: 40, 30, 30

   "Cold build", 42, 12
   "Incremental", 3, 4

```

图片与插图

用 `image` 和 `figure` 插入图片、截图和图表：

```

.. image:: /assets/architecture.png
   :width: 600px
   :align: center
   :alt: System architecture diagram

.. figure:: /assets/flow.png
   :scale: 80%
   :align: center
   :alt: Execution flowchart

   Data flows sequentially through reader, resolver, and renderer.

```

`figure` 在图片周围添加说明文字，并可指定引用标签。

用目录树组织文档

`toc-tree` 定义文档层级、导航菜单，以及 HTML 和 PDF 的阅读顺序：

```

.. toctree::
   :maxdepth: 2
   :caption: User Guide
   :numbered:

   installation
   quickstart
   configuration

```

`toctree` 的常用选项：

- `:maxdepth: N`: 目录中包含的标题层级深度。
- `:caption: Title`: 导航条目上方显示的分类标题。
- `:numbered::` 为章节和标题添加编号。
- `:titlesonly::` 只列出文档标题，不显示内部章节。
- `:hidden::` 记录文档阅读顺序，但在正文中显示列表。
- `:glob::` 允许用通配符匹配文档，例如 `tutorials/*`。

交叉引用与链接

显式目标标签可保持引用稳定，不依赖文档文件名。

目标标签与 `:ref:`

在标题、表格或插图前放置目标标签：

```
.. _storage-model:

Storage model
-----

Caller storage is bounded and measured upfront.
```

项目中的任意文档都可引用此目标：

```
See :ref:`storage-model` for details.
See :ref:`custom link text <storage-model>`.
```

用 `:doc:` 引用文档

按路径链接到其他文档，省略扩展名：

```
Read the :doc:`configuration guide <../reference/configuration>` for details.
```

用 `:term:` 引用术语

用 `glossary` 定义术语：

```
.. glossary::

    Workspace
        A fixed memory arena allocated by the caller for conversion passes.

    Pass
        An in-memory transformation step operating on the document AST.
```

用 `:term:` 链接到术语：

```
Conversion runs within an allocated :term:`workspace`.
```

用 `:download:` 提供文件下载

将文件复制到下载目录，并生成链接：

```
Download the :download:`starter template <files/starter.toml>`.
```

外部链接

链接到外部网址：

```
Visit `Typst` <https://typst.app>`_ for typography details.
```

替换与包含

定义可重复使用的短语、符号或图片：

```
.. |version| replace:: 1.0.0
.. |brand| replace:: Field Notes

Welcome to |brand| version |version|.
```

在多个文件中共用 reStructuredText 内容：

```
.. include:: ../shared/warnings.rst
```

编写 MyST Markdown

MyST Markdown 通过围栏块表达相同的指令和角色语义：

```
# Project overview

Here is a paragraph with bold text, italic emphasis, and `inline code`.

``{note}
This note is rendered identically to a reStructuredText note directive.
```

``{code-block} python
:caption: Example function
:linenos:

def add(a, b):
 return a + b
```

``{toctree}
:maxdepth: 2
:caption: Navigation

first-chapter
second-chapter
```

See {ref}`storage-model` or consult the {doc}`../reference/configuration`.
```

## 域

域是一组指令和角色，用来描述某种编程语言或知识领域中的对象，并建立交叉引用。

Guidedog 内置了以下 Sphinx 风格的域：

- **标准域** (`std`)：命令行选项、环境变量、术语和界面标签等通用对象。
- **Python 域** (`py`)：模块、类、函数、方法、属性和异常。
- **C 域** (`c`)：函数、类型、结构体、联合体、枚举、枚举值、变量和宏。
- **C++ 域** (`cpp`)：类、概念、模板、命名空间、方法和表达式。
- **JavaScript 域** (`js`)：函数、方法、类和属性。
- **Odin 域** (`odin`)：原生 Odin 包、过程、结构体、联合体和枚举，见 [为 Odin 生成文档](#)。



## 设置主域

在 `conf.toml` 中用 `primary_domain` 设置默认域，默认为 `"py"`。启用后可省略域前缀，例如用 `.. function::` 代替 `.. py:function::`，用 `:func:`name`` 代替 `:py:func:`name``。

在文档中局部切换当前域：

```
.. default-domain:: c
```

## 标准域

标准域描述命令行工具、环境变量和文档交叉引用。

### 指令

```
.. program:: mytool

.. option:: -c <config>, --config <config>
 Path to the configuration file.

.. envvar:: GUIDEDOG_THEME
 Specifies the default theme stylesheet.
```

### 角色

- `:doc:`path``: 按相对路径链接到项目文档。
- `:ref:`label``: 链接到显式目标标签 `.. _label:`。
- `:term:`term``: 链接到 `glossary` 中定义的术语。
- `:option:`--config``: 链接到 `option` 中描述的命令行选项。
- `:envvar:`VARIABLE``: 链接到 `envvar` 中描述的环境变量。
- `:command:`name``: 标记系统命令名。
- `:file:`path``: 标记文件或目录路径，可包含 `{variable}` 占位符。
- `:kbd:`Ctrl+C``: 标记键盘按键。
- `:menuselection:`File --> Save As``: 表示菜单操作顺序。
- `:guilabel:`Submit``: 表示按钮或界面元素。
- `:pep:`8``: 链接到 Python 改进提案。
- `:rfc:`7231``: 链接到 IETF RFC 文档。

## Python 域

Python 域描述 Python 接口和签名。

### 指令

```
.. py:module:: pipeline.reader
 :synopsis: Document reading and normalization.

.. py:class:: DocumentReader(source_path: str, encoding: str = "utf-8")
 Base class for document readers.

.. py:method:: parse(content: bytes) -> Document
 Parses raw bytes into a document tree.
 :param content: Raw source content bytes.
```

```

: return: Parsed document tree instance.
: raises ValueError: If the source format is unrecognized.

.. py:attribute:: encoding
: type: str

The character encoding used for text decoding.

```

支持的指令包括: `py:module`、`py:currentmodule`、`py:function`、`py:data`、`py:class`、`py:method`、`py:staticmethod`、`py:classmethod`、`py:attribute`、`py:property`、`py:exception`、`py:decorator` 和 `py:type`。

## 角色

- `:py:func:`name``: 链接到 Python 函数。
- `:py:class:`name``: 链接到 Python 类。
- `:py:meth:`name``: 链接到 Python 类方法或实例方法。
- `:py:attr:`name``: 链接到 Python 属性。
- `:py:data:`name``: 链接到 Python 模块级数据。
- `:py:exc:`name``: 链接到 Python 异常类。
- `:py:mod:`name``: 链接到 Python 模块。

## C 域

C 域描述 C 语言库和头文件。

## 指令

```

.. c:namespace:: gd

.. c:struct:: Workspace

A contiguous memory buffer for in-memory document parsing.

.. c:member:: size_t capacity

Total byte size of the storage slab.

.. c:function:: int gd_convert(Workspace *ws, const char *input, char *output,
size_t out_len)

Converts source text into HTML.

: param ws: Pointer to the initialized workspace.
: param input: Null-terminated input string.
: param output: Destination buffer.
: param out_len: Size of destination buffer.
: returns: 0 on success, or an error code.

```

## 角色

- `:c:func:`name``: 链接到 C 函数。
- `:c:member:`name``: 链接到 C 结构体或联合体的成员。
- `:c:data:`name`` / `:c:var:`name``: 链接到 C 变量。
- `:c:type:`name``: 链接到 C 的 typedef 或类型。
- `:c:struct:`name``: 链接到 C 结构体。
- `:c:union:`name``: 链接到 C 联合体。
- `:c:enum:`name``: 链接到 C 枚举。
- `:c:macro:`name``: 链接到 C 预处理器宏。

## C++ 域

C++ 域支持现代 C++ 构造、概念、命名空间和作用域。

### 指令

```
.. cpp:namespace:: guidedog

.. cpp:concept:: template<typename T> Reader
 Specifies requirements for document reader types.

.. cpp:class:: template<typename Allocator> Builder
 Constructs AST document nodes.

.. cpp:function:: NodeId add_text(std::string_view text)
 Appends text to the active node.
```

### 角色

- `:cpp:class:`name``: 链接到 C++ 类或结构体。
- `:cpp:func:`name``: 链接到 C++ 函数或方法。
- `:cpp:member:`name`` / `:cpp:var:`name``: 链接到成员或变量。
- `:cpp:type:`name``: 链接到 C++ 类型别名或 typedef。
- `:cpp:concept:`name``: 链接到 C++20 概念。
- `:cpp:enum:`name``: 链接到 C++ 枚举。

## JavaScript 域

JavaScript 域描述浏览器脚本、库和 Node.js 模块。

### 指令

```
.. js:module:: theme

.. js:class:: ThemeController(options)
 Manages color themes and local storage persistence.

.. js:method:: toggleTheme()
 Toggles between light and dark modes.
```

### 角色

- `:js:func:`name``: 链接到 JavaScript 函数。
- `:js:meth:`name``: 链接到 JavaScript 方法。
- `:js:class:`name``: 链接到 JavaScript 类。
- `:js:data:`name``: 链接到 JavaScript 变量。
- `:js:attr:`name``: 链接到 JavaScript 对象属性。
- `:js:mod:`name``: 链接到 JavaScript 模块。

## 交叉引用约定

域中的交叉引用支持以下修饰符：

- **隐藏前缀**：在目标前加 `~`，只显示名称的最后一部分。例如 `:py:func:`~pipeline.reader.DocumentReader.parse`` 显示为 `parse()`。

- **当前作用域**：相对引用先在当前模块或命名空间中查找，再进行全局查找。
- **精确匹配**：在目标前加 `.`，从当前模块或所属对象开始查找。

## 项目如何工作

项目根目录是包含 `conf.toml` 的目录。`root_doc` 指定阅读顺序起点。Guidedog 发现并读取源文档、解析关系，发布完整输出版本。

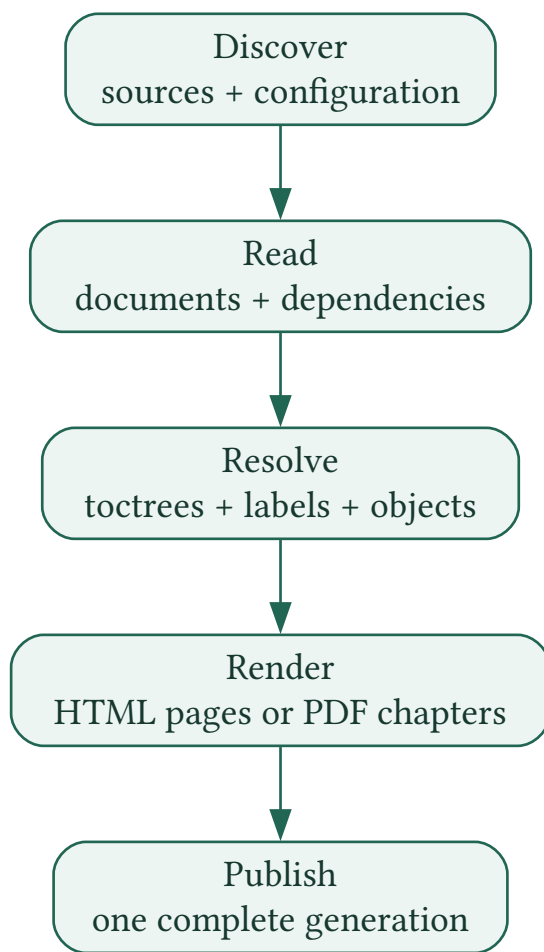


图 2 构建将发现、语义、呈现和发布分开。

### 文档与资源

文件后缀选择读取器。RST 和项目 Markdown 共享项目图。图片和下载是资源；Guidedog 复制文档使用的资源。主题资源列于 `html_static_path`。包含文件必须位于源目录或配置的 `include_roots` 内。

### 独立的项目

手册与设计记录应放在独立项目中：读者和阅读顺序不同。各自拥有配置、模板、缓存和输出。Guidedog 的 `docs/manual` 与 `docs/gds` 即采用此方式。

## 增量构建

Guidedog 记录各输出使用的输入和查找结果。源文件改变会重读，依赖改变使使用者失效，模板改变可影响每页。未变文档可从保存对象加载。

缓存是上次构建的证据，不是事实来源。`-E` 丢弃下次构建使用的保存环境，`-a` 写出所有输出。它们解决不同问题。

可复用 = 输入相同  $\wedge$  查找结果相同  $\wedge$  输出存在

模板中的显式时间戳也是页面输入。即使文字未变，打印当前分钟的页面也可能改变。需要固定构建时间时，用 `SOURCE_DATE_EPOCH`。

## 发布

构建将变化写入同级暂存目录，完成目录、刷写文件和记录后，再切换版本。读者看到完整旧版或新版。中断的提交在下次构建处理。设计和证明义务见发布完整的一代输出。

## 数学与证明

公式应让关系更容易检查。使用符号前先定义，必要时说明单位。公式应支持解释，而非取代解释。

### 行内公式与独立公式

短公式用 `math` 角色，独立公式用 `math` 指令。使用两个目标均支持的通用 LaTeX 数学子集。

```
A side of length :math:`s` encloses area :math:`s^2`.

.. math::
 :label: area-square

 A = s^2

The relation is :eq:`area-square`.
```

边长为  $s$  的正方形，面积为  $s^2$ 。

$$A = s^2 \tag{1}$$

关系见 (1)。Guidedog 为带标签公式提供稳定目标。HTML 构建器提供 MathJax 标记；PDF 渲染器将受支持的 LaTeX 子集转为 Typst 数学。不支持的 PDF 命令会被报告，不执行任意 LaTeX 包。

### 一个简短的证明

设  $U$  为已计入预算的字节数， $B$  为上限， $n$  为申请分配量。假设三者均非负，且  $U \leq B$ 。

**准入规则。** 当且仅当以下条件成立时，接受请求：

$$n \leq B - U. \tag{2}$$

**命题。** 接受请求后，仍有  $U + n \leq B$ 。

**证明。** 在 (2) 两边加  $U$ ，得到  $U + n \leq B$ 。实现检查减法形式，不先计算可能使机器整数溢出的  $U + n$ 。证明与运算顺序一致。

**边界。** 这只核算已计费的宿主存储，不限制独立 Typst 编译器的内部分配。证明的效力取决于其声明的边界。

## HTML 设置

这些项目随附 MathJax 3.2.2 及 SVG 字体数据，仅在含数学公式的页面加载。构建后离线也可阅读，无需图示脚本或网络字体服务。

你自己的项目默认从 CDN 加载 MathJax。若要使用本地副本，将它放入 `_static`，并设置：

```
html_static_path = ["_static"]
mathjax_path = "_static/tex-svg-full.js"
```

Web 公式需要 JavaScript 排版；关闭时仍显示源文。PDF 公式不依赖浏览器 JavaScript。见 [MathJax 组件契约](#)。

## 用图表解释概念

按问题选择图示。流程图解释选择，框图解释职责，时序图解释顺序，关系图解释联系。文字能更直接回答时，就用文字。

### 流程图

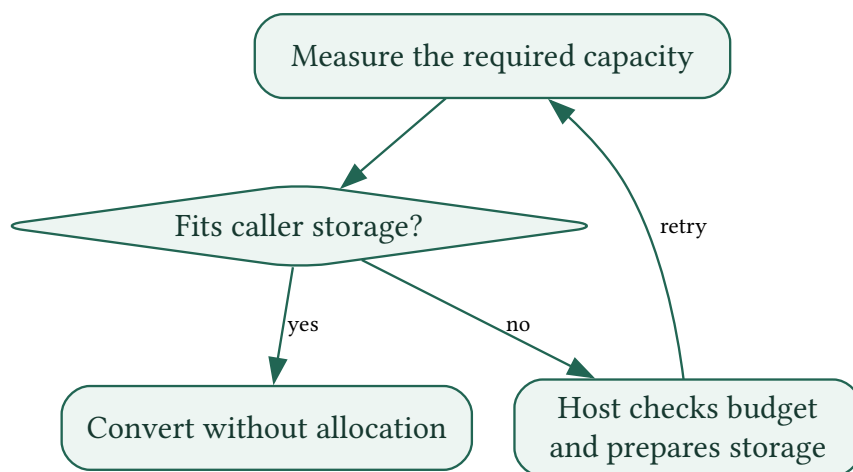


图 3 借用或扩展存储前，先检查容量。

图中在扩展步骤标明所有者，不暗示转换核心会分配内存。这一区别属于契约。

### 编写图表

```
.. graphviz::
:caption: Parsing precedes resolution.
:alt: The source becomes a tree, then references are resolved.

digraph pipeline {
 graph [rankdir=LR, bgcolor="transparent"];
```

```
node [shape=box];
"source" -> "tree" -> "resolved tree";
}
```

graphviz、graph 和 digraph 为内置指令。Guidedog 通过 Graphviz 渲染；HTML 获得 SVG 资源，PDF 嵌入渲染后的图示，两者使用同一图源。

## 让图表清晰易读

节点标签要短，较长规则写在图下。主路径尽量保持一个方向，返回边标明条件。除颜色外，也使用形状和文字；打印读者可能看不到颜色。

提供替代文本和题注。检查窄浏览器窗口和 PDF 页面。若必须用极小字号才能放下，拆成两幅图。图片和插图尺寸见 [图片、视频、音频与下载](#)。

## 适合阅读的排版

页面应在细读之前展示结构。醒目的标题、安静的导航栏和适度宽的正文栏已完成大部分工作。颜色应标明关系或操作，不独自承担含义。

### 网页

自定义布局分为项目导航、正文和当前页面纲要三部分，小屏使用导航抽屉。阅读版随附 Libertinus Serif 和 Libertinus Sans。标题用粗体衬线字；导航、标签和控件用配套无衬线字。压缩 WOFF2 文件随站点提供，不请求字体服务。

**默认浅色。** 主题控件也提供深色和系统模式，并记住同一浏览器来源上的选择。系统模式跟随操作系统。打印 HTML 使用浅色。禁用 JavaScript 后，内容和导航仍可用；主题控件与增强搜索属于渐进增强。

英文正文可在浏览器中使用整段 **Knuth–Plass 断行**。小型模板脚本测量已加载字体，建立文字盒、空格与连字断行惩罚，选择整段断点。Frank Liang 的模式方法提供英文断词候选。原始 TeX 模式和库许可证随站点提供。

增强排版保留强调和原始文字，正文宽度改变时重新计算。复制一个优化段落内的内容，得到的是源文，不含视觉换行或新增连字符。打印时恢复普通段落，交由打印布局分页。

链接、数学、代码、交互内容、窄栏和其他语言采用浏览器原生布局；无合适间距解的段落也如此。禁用 JavaScript 后仍可阅读。回退使用语言感知的 `hyphens: auto` 和 `text-wrap: pretty`；这两项 CSS 本身不保证 Knuth–Plass 优化。

### PDF 页面

书籍模板使用与 Web 相同的 Libertinus 字体，设置两端对齐、优化断行和语言感知断词，也控制间距、页眉、题注、代码块和表格线。字体、断词成本与段落宽度共同作用。强制换行通常不能正确修复松散段落。

Typst 优化 PDF 段落，Web 增强优化适用正文。它们共享源文、字体和视觉层级，但断行可能不同。见 [Typst 文本契约](#) 和 [浏览器断词契约](#)。

## 修改设计

编辑 `_static/editorial.css` 调整视觉变量，编辑 `_templates/layout.html` 调整结构，编辑 `_templates/book.typ` 调整印刷布局。保留阅读顺序、焦点指示、标签和足够对比度。

修改后构建两种格式，检查含代码、表格、图示和公式的页面。漂亮封面不能证明全书可用。完整模板契约见 [模板](#)；完整 API 见 [API Reference](#)，也包含在同一本 PDF 中。

## 模板

模板负责呈现，不重复源文档的正文。Guidedog 使用 Jinja 生成 HTML，使用 Typst 排版 PDF。

Guidedog 的模板就是普通项目文件，可直接查看和编辑。`guidedog quickstart` 会将实际使用的模板写入项目：

- `_templates/layout.html` 定义 HTML 网站的结构，使用 Jinja。
- `_templates/book.typ` 定义 PDF 的版式、字体和封面，使用 Typst。
- `_static/guidedog.css` 和 `_static/guidedog.js` 提供默认样式和浏览器交互。

模板保存在项目目录中，不隐藏在外包或二进制缓存里。直接编辑后，用 `guidedog build` 重建，用 `guidedog serve` 预览。

### HTML 模板

Guidedog 用内置 Jinja 引擎渲染根页面模板 `layout.html`。布局无需挤在一个文件中；也可以拆成局部模板、宏库和多层继承结构。

### 模块化模板结构

模板可分散在 `templates_path` 指定目录下的多个文件和子目录中；默认路径为 `["_templates"]`：

```
my-docs/
├── conf.toml
├── index.rst
└── templates/
 ├── layout.html
 ├── base.html
 ├── partials/
 │ ├── header.html
 │ ├── navigation.html
 │ ├── searchbox.html
 │ └── footer.html
 └── macros/
 └── components.html
```

Guidedog 会扫描 `templates_path` 下的全部文件和子目录。包含或导入模板时，路径以该目录为基准：



代码 1 \_templates/layout.html

```
{% extends "base.html" %}

{% block header %}
 {% include "partials/header.html" %}
{% endblock %}

{% block footer %}
 {% include "partials/footer.html" %}
{% endblock %}
```

Jinja 宏也可定义在单独的文件中，按需导入：

代码 2 \_templates/macros/components.html

```
{% macro badge(label, type="info") %}
 {{ label }}
{% endmacro %}
```

代码 3 在模板中使用宏

```
{% import "macros/components.html" as ui %}
{{ ui.badge("New", type="success") }}
```

### 用 !layout.html 继承主题

若只需修改默认主题的部分内容，无需重写整页。用 Sphinx 风格的感叹号语法继承 Guidedog 的内置布局：

代码 4 \_templates/layout.html

```
{# Inherit Guidedog's built-in layout #}
{% extends "!layout.html" %}

{# Inject custom metadata or web fonts into the document head #}
{% block extrahead %}
 {{ super() }}
 <link rel="stylesheet" href="{{ pathto('_static/custom.css', 1) }}">
{% endblock %}

{# Replace or extend the footer with custom content #}
{% block footer %}
 {% include "partials/footer.html" %}
{% endblock %}
```

`{% extends "!layout.html" %}` 或 `{% extends "basic/layout.html" %}` 会读取内置主题模板，避免再次加载项目自身的 `_templates/layout.html` 而形成递归。

在覆盖的块中调用 `{{ super() }}`，即可保留父模板内容，并在其前后追加内容。

### 布局块

内置 `layout.html` 使用与 Sphinx 约定对应的块名：

| 块                      | 用途                                                                                                                  |
|------------------------|---------------------------------------------------------------------------------------------------------------------|
| <code>doctype</code>   | 文档类型声明，默认为 <code>&lt;!DOCTYPE html&gt;</code> 。                                                                     |
| <code>htmltitle</code> | <code>&lt;head&gt;</code> 内的 <code>&lt;title&gt;</code> 元素。                                                         |
| <code>linktags</code>  | 导航和元信息链接： <code>favicon</code> 、 <code>index</code> 、 <code>search</code> 、 <code>prev</code> 、 <code>next</code> 。 |

| 块           | 用途                               |
|-------------|----------------------------------|
| css         | 样式表链接及内联 CSS 根变量。                |
| scripts     | JavaScript 脚本，包括搜索索引和交互功能。       |
| extrahead   | <head> 末尾的插入位置，用于字体、分析脚本或自定义元信息。 |
| header      | 紧邻主导导航栏之前的页头位置。                  |
| relbar1     | 顶部导航栏，包含品牌、版本、搜索和主题切换。           |
| rootrellink | 相关链接之前的导航插入位置。                   |
| relbaritems | 导航栏中的自定义项目。                      |
| sidebar1    | 左侧导航栏容器。                         |
| sidebartoc  | sidebar1 内的目录导航树。                |
| breadcrumbs | 正文上方的分层导航路径。                     |
| document    | 包裹页面正文的容器。                       |
| body        | 当前文档的 HTML 正文，即 {{ body }}。      |
| relbar2     | 底部翻页导航，链接到上一章和下一章。               |
| footer      | 页脚，包含版权、更新日期和源文件链接。              |
| sidebar2    | 右侧辅助栏，显示本目录。                     |

查找模板

Guidedog 按 `conf.toml` 中 `templates_path` 的顺序搜索目录，最后搜索内置模板。模板名相对于这些目录，不能用 `..` 越出目录。

`templates_path` 下的每个文件和子目录都是构建依赖。添加、修改或删除模板后，`guidedog build` 和 `guidedog serve` 会检测变化并重新渲染网站。

页面变量

Sphinx 已有名称的变量，Guidedog 采用相同名称，因此可迁移部分 Sphinx 主题。

| 变量                                             | 值                                                                                                                                                                                              |
|------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| body                                           | HTML 格式的文档。                                                                                                                                                                                    |
| title                                          | 文档标题。                                                                                                                                                                                          |
| pagename, docname                              | 文档名称，如 <code>usage/install</code> 。非文档页面的 <code>docname</code> 为空， <code>pagename</code> 是页面名，与 Sphinx 一样，例如 <code>genindex</code> 、 <code>py-modindex</code> 、 <code>search</code> 或模板生成的页面名。 |
| toc                                            | 由目录树生成的 HTML 导航。                                                                                                                                                                               |
| outline                                        | 文档自身各节的 HTML；没有节时空。                                                                                                                                                                            |
| prev, next                                     | 阅读顺序中的相邻页面，含 <code>url</code> （也可用 <code>link</code> ）和 <code>title</code> ；首尾无相邻页时空。                                                                                                          |
| project, version, release, copyright, language | 同名的配置项。                                                                                                                                                                                        |
| html_title, docstitle                          | <code>html_title</code> ，默认为“<project> <release> documentation”。                                                                                                                               |
| html_short_title, shorttitle                   | <code>html_short_title</code> 。                                                                                                                                                                |

| 变量                                                                                                                                       | 值                                                                                                                                                                                                                                             |
|------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>root_doc</code> , <code>master_doc</code>                                                                                          | 根文档的名称。                                                                                                                                                                                                                                       |
| <code>pathto_root</code>                                                                                                                 | 从当前页面到站点根目录的路径，例如 <code>../</code> 。                                                                                                                                                                                                          |
| <code>root_url</code> , <code>search_url</code> , <code>genindex_url</code>                                                              | 根页面、搜索页面和通用索引的链接。                                                                                                                                                                                                                             |
| <code>css_files</code> , <code>js_files</code>                                                                                           | 带有 <code>url</code> 的文件列表。先列 Guidedog 自有文件，再列 <code>html_css_files</code> 和 <code>html_js_files</code> 。                                                                                                                                      |
| <code>logo_url</code> , <code>favicon_url</code>                                                                                         | <code>_static</code> 下的 <code>html_logo</code> 和 <code>html_favicon</code> ；未设置时空。                                                                                                                                                            |
| <code>accent</code>                                                                                                                      | 主题颜色 <code>html_theme_options.accent</code> 。                                                                                                                                                                                                 |
| <code>sourcelink_url</code>                                                                                                              | 显示源码时，指向 <code>_sources</code> 下的文档源文件；否则为空。                                                                                                                                                                                                  |
| <code>last_updated</code>                                                                                                                | 设置了 <code>html_last_updated_fmt</code> 时，按该格式显示构建日期；否则为空。                                                                                                                                                                                     |
| <code>show_copyright</code> , <code>show_sphinx</code> , <code>show_guidedog</code> , <code>has_source</code> , <code>show_source</code> | <code>html_show_*</code> 和 <code>html_copy_source</code> 配置项。                                                                                                                                                                                 |
| <code>builder</code> , <code>file_suffix</code>                                                                                          | 构建器名称，例如 <code>html</code> ，以及页面后缀。                                                                                                                                                                                                           |
| <code>html_context</code> 中的每个键                                                                                                          | 值及其类型与 Sphinx 传递 Python 值的方式相同： <code>false</code> 在 <code>{% if %}</code> 中为假，数值仍为数值，数组为列表，表为字典，其键保留 <code>conf.toml</code> 顺序。<br><code>guidedog migrate</code> 迁移 <code>conf.py</code> 的 <code>html_context</code> 字面量条目；计算值保持未定义，模板将其视为假。 |

`pathto` 是 Sphinx 的模板函数。`pathto("usage/install")` 返回文档页面的 URL，`pathto("_static/logo.svg", 1)` 返回站点根目录下文件的 URL，均相对于当前页面。`singlehtml` 与 Sphinx 一样，把文档作为单页中的一节：该页上 `pathto("usage/install")` 为 `#document-usage-install`；索引和搜索页上为 `index.html#document-usage-install`。`genindex` 或下述模板页等非文档名称，在所有构建器中都是站点根目录页面。

## 由模板生成的页面

`html_additional_pages` 与 Sphinx 一样，仅用模板创建页面。每个键是页面名称，值是 `templates_path` 中的模板：

代码 5 `conf.toml`

```
root_doc = "contents"

[html_additional_pages]
index = "indexcontent.html"
download = "download.html"
```

页面模板通常继承 `layout.html` 并填充其区块，让页面与其他页面风格一致：

代码 6 \_templates/indexcontent.html

```
{% extends "layout.html" %}
{% block htmltitle %}<title>{{ shorttitle }}</title>{% endblock %}
{% block body %}
 <h1>{{ docstitle|e }}</h1>
 <p>Tutorial</p>
{% endblock %}
```

模板获得非文档页面的变量: `pagename` 是页面名(`index`), `title` 和 `body` 为空; `pathto`、`toc`、`toctree()` 和 `html_context` 值照常可用。`html`、`dirhtml`、`singlehtml` 都将其写为根目录 `<name>.html` (`html_file_suffix`), 每次构建重写。该模板或 `templates_path` 任意文件改变, 都会重建所有页面。

页面位置及与 Sphinx 的差异:

- 与 Sphinx 最后写入的行为一样, 同名模板页面替换文档页面: `index = "landing.html"` 使 `index.html` 成为落地页, 但 `index.rst` 仍提供目录条目。`singlehtml` 保留单页, 不写模板页面, 并发出警告。
- `dirhtml` 将页面写为 `download.html`, 与 `genindex.html` 和 `search.html` 并列; Sphinx 则写为 `download/index.html`。`pathto("download")` 返回该文件。
- 名称必须是根目录文件名。`"sub/page"` 会被警告拒绝 (`build.additional_page`), 因为页面链接以根目录为基准。`genindex` 等已占用文件名也会被拒绝; Sphinx 在此会覆盖其中一页。
- `guidedog migrate` 会迁移 `conf.py` 中的 `html_additional_pages`。

## 错误

模板出错会停止构建, 并报告文件、行号、该行内容和修改建议:

```
TEMPLATE ERROR template.error

 _templates/base.html:1

No filter named 'default'.

Did you mean the filter 'default'?
```

未定义的变量渲染为空, 与 Jinja 一致。

失控模板也会报错停止, 而非崩溃: 语法嵌套超过 100 层 (括号、标签, 以及长运算符、过滤器或 `elif` 链的每一环), 宏、包含或递归循环超过 200 层, 或任何递归所需栈超过 512 KiB。自包含列表与 Python 一样显示为 `[...]`。

## PDF 模板

PDF 书籍由 Typst 排版。`_templates/book.typ` 是定义 `book` 函数的普通 Typst 文件。Guidedog 按以下形式生成书籍源文:

```
#import "/_templates/book.typ": book
#show: book.with(title: ..., author: ..., version: ..., date: ...,
 lang: ..., paper: ..., numbering: ..., logo: ...)
// the chapters, one per document of the root toctree
```

因此, 决定图书外观的一切都在该文件中: 字体、页面尺寸与边距、标题、扉页、页眉和目录。

### 参数

参数	值
title	pdf_documents 中的图书 title，否则为 project。
author	图书的 author，否则为 author 配置项。
version	release。
date	today ：若设置了 today 则使用其值，否则按 today_fmt 显示构建日期。
lang	language。
paper	"a4"；若 pdf_paper_size 为 letter，则使用 "us-letter"。
numbering	当 toctree 含 numbered 选项时为 true。
logo	pdf_logo 的路径；未设置则不传入。
copyright	供版权页使用的 copyright，仅传给接受此参数的模板。
body	章节内容。

模板可增加带默认值的参数。quickstart 模板包含 accent、字体 serif、sans、mono、字号 size，以及 page-ref。将后者设为 `n => [p. #n]` 等函数，可为跨页引用附上页码。

### 模块化 Typst 模板

Typst 样式也无需放在一个文件中。可将样式、宏和封面拆成 `_templates/` 下的多个 `.typ` 文件，再用 `#import` 和 `#include` 组合：

代码 7 `_templates/book.typ`

```
#import "cover.typ": title-page
#import "typography.typ": apply-styles

#let book(
 title: "",
 author: "",
 version: "",
 date: "",
 lang: "en",
 paper: "a4",
 numbering: false,
 logo: none,
 body,
) = {
 apply-styles()
 title-page(title: title, author: author, version: version, logo: logo)
 body
}
```

### 模板可以使用什么

#### Typst 包

`#import "@preview/cetz:0.4.2"` 等包可用，首次使用时下载。设置 `pdf_packages = "offline"`，仅使用磁盘上已有的包构建。

#### 字体

可使用系统字体，与 `typst` 命令相同。通过 `pdf_font_paths` 增加目录；或设 `pdf_fonts = "embedded"`，只用 Typst 内置字体，实现逐字节可重复的书籍。

## 前置内容

`pdf_preamble` 指定 Typst 文件，其内容放在 `#show: book` 之后。若只需几条 `set` 或 `show` 规则，则无需自写完整模板。

## 文档中的 Typst

文档是 reStructuredText 或 Markdown；Typst 文件不是文档。要将 Typst 标记放入书籍，写入原始块；网页会忽略该块：

```
.. raw:: typst
 #align(center)[#text(size: 14pt)[Only in the book]]
```

`.. only:: pdf` 也可将普通内容限制在图书中。

Typst 报错时，构建会指出文件和行号，无论错误位于模板还是生成书籍。每个 PDF 的源文位于 `_build/pdf/sources/<pdf-filename>.typ`，如 `reference-en.pdf` 对应 `sources/reference-en.pdf.typ`。单本构建也保留熟悉的 `_build/pdf/book-<language>.typ` 检查副本。失败不发布内容，源码保留在 `_build/.doctrees/failed`，错误会给出路径。多本书各自保留独立失败源码。

## 编写新模板

从 `quickstart` 生成的模板开始，其中展示了各部分的用法：

1. 保留默认样式表依赖的 `layout.html` 标记，或同时替换 `_static/guidedog.css`。
2. 将重复部分移入单独文件并用 `include` 引入，或建立 `base.html`，让 `layout.html` 通过 `extends` 继承。
3. 图书可修改 `book` 函数中的规则，或写同名、同参数的新函数。

编辑时用 `guidedog serve` 预览。保存源文件后刷新浏览器，服务器会先重建变化的输入，再提供页面。

## 图片、视频、音频与下载

将资源放在解释它的文档附近，使用相对路径；若以 `/` 开头，则相对于源目录。Guidedog 会复制文档使用的资源。视频在网页上仍可播放；PDF 则标明该录制内容。

### 图片

```
.. image:: diagrams/pipeline.png
 :alt: The build pipeline
 :width: 480px

.. figure:: /screenshots/search.png

 Searching the site.
```

页面将图片复制到 `_images`，图书则将其嵌入。两种输出均支持 PNG、JPEG、GIF、SVG 和 WebP。图片缺失会产生警告：页面保留替代文本，图书省略图片。

## 视频与音频

指向视频（.mp4、.webm、.ogv、.mov、.m4v）或音频（.mp3、.ogg、.oga、.m4a、.opus、.wav、.flac、.aac）的 `image` 或 `figure` 指令会播放媒体；Docutils 对视频也采用此方式：

```
.. figure:: media/tour.mp4
 :alt: A tour of the site
 :width: 640

 The two-minute tour.

.. image:: media/pronunciation.mp3
```

页面显示浏览器播放器，并提供文件链接，供无法播放的浏览器使用。图书不能播放媒体，因此在播放器所在位置用方框标明录制内容；文本输出则写为 `[video: ...]` 或 `[audio: ...]`。

## 下载

`download` 角色可链接任意文件，并将其复制到 `_downloads:`

```
Get :download:`the example project <examples/project.zip>`.
```

## 主题文件

`_static` 保存主题使用的样式表、脚本、字体和徽标，并原样复制到站点的 `_static`。`html_static_path` 列出该目录；`html_css_files`、`html_js_files`、`html_logo` 和 `html_favicon` 指定其中的文件。内容媒体应与文档放在一起，这样移动文档和媒体后，链接仍有效。

## 国际化

翻译解释，保留结构。Guidedog 提取消息清单、合并译文，并按选定语言构建。源文本始终是参照，缺失的译文回退到源文，不凭空编造。

## 工作流程

1. 使用 `gettext` 构建器生成消息模板：

```
guidedog build gettext
```

它在 `_build/gettext` 中为每个文本域生成一个模板（.pot）。文档中的段落、标题、列表项、表格单元格、题注、提示框、术语和字段都会成为消息，并记录所在文件和行号。

2. 像 `sphinx-intl update` 一样，为各语言创建或更新清单：

```
guidedog intl update -l de -l fr
```

目录文件位于 `locales/de/LC_MESSAGES/*.po`。再次运行时，命令会像 `msgmerge` 一样合并新模板：保留已有译文；源文改变时，将旧译文标为 *fuzzy*，供复核；已删除的消息作为废弃条目（#~）保留在末尾，以便日后恢复。

3. 填写每个 `msgstr`。确认译文正确后，删除该消息的 `#, fuzzy` 行；构建时不会使用模糊译文。

```
msgid "Hello *world*, see the site_."
msgstr "Hallo *Welt*, siehe site_."
```

4. 像 `sphinx-intl stat` 一样查看翻译进度：

```
guidedog intl stat -l de
```

5. 按目标语言构建：

```
guidedog build html -D language=de
guidedog build pdf -D language=de
```

也可在 `conf.toml` 中设置 `language = "de"`。

每本书都有语言后缀：`manual.pdf` 会变为 `manual-de.pdf`。一次构建发布一个完整版本，因此保留多语言版本时应使用不同目标目录。例如，德文用 `guidedog build -b pdf docs build/books/de -D language=de`，英文用 `guidedog build -b pdf docs build/books/en -D language=en`。

与文档的其他输入相同，只有清单变化的文档才会重读；添加或删除清单时会重读所有文档。

`guidedog intl` 支持 `sphinx-intl` 的选项：`-p DIR` 指定模板目录（默认 `_build/gettext`）；`-l LANG` 指定语言，可重复使用（默认为项目语言）；`-d DIR` 指定目录文件位置（默认取 `locale_dirs` 的第一项）；`-w N` 指定行宽（默认 76）；还支持 `--no-obsolete`。

## 编写翻译

消息是段落、标题或其他元素的源文本，包含标记语法。`.rst` 文档使用 `reStructuredText`，`.md` 文档使用 `Markdown`。译文按同种语法读取，因此可保留原文的强调、链接和引用。

引用必须保持一致。译文保留消息中的每个引用及其写法，只改动周围文字和显示标题：

```
msgid "See :ref:`install` and the site_."
msgstr "Siehe :ref:`Installation <install>` und die site_."
```

译文缺少引用时，构建会像 `Sphinx` 一样发出警告（`inconsistent references in translated message`），但仍采用译文。若译文引用了原文没有的目标，该引用无法解析，构建会警告并保留原文。若读取器拒绝译文标记，也会保留原文，并在警告中列出该译文。

章节锚点不随语言变化。翻译后的标题保留原文锚点，因此站点内部和外部的链接仍然有效。

## 设置

标准设置沿用 `Sphinx` 的名称和默认值。`Guidedog` 还提供仅用于提取的排除项，可让参考附录保持原文。

设置项	含义
<code>language</code>	构建语言，例如 <code>"de"</code> 或 <code>"pt_BR"</code> 。



设置项	含义
<code>locale_dirs</code>	相对于源目录的目录文件位置。每处包含 <code>LANGUAGE/LC_MESSAGES/DOMAIN.po</code> （或编译后的 <code>.mo</code> ）。默认是 <code>["locales"]</code> 。指定多处时，采用第一个能翻译该消息的目录。
<code>gettext_compact</code>	决定文档消息所属的目录文件。 <code>true</code> （默认）：顶层文档各用一个目录文件，同一文件夹内的文档共用一个（ <code>guide/usage.rst</code> 使用 <code>guide.po</code> ）。 <code>false</code> ：每个文档一个。指定名称：所有文档共用该名称的目录文件。
<code>gettext_location</code>	写入消息位置（#： <code>../../index.rst:12</code> ），默认为 <code>true</code> 。
<code>gettext_uuid</code>	为消息的每次出现写入标识符。Guidedog 根据文档、消息及出现位置生成，因此每次构建结果相同。
<code>gettext_auto_build</code>	将每个 <code>.po</code> 编译为旁边的 <code>.mo</code> ，供其他工具使用，默认为 <code>true</code> 。Guidedog 自身读取 <code>.po</code> 。
<code>gettext_additional_targets</code>	额外翻译内容： <code>index</code> （索引项）、 <code>literal-block</code> 、 <code>doctest-block</code> 、 <code>raw</code> 和 <code>image</code> （替代文本）。
<code>gettext_allow_fuzzy_translations</code>	也使用模糊译文，默认为 <code>false</code> 。
<code>gettext_exclude_patterns</code>	仅从 POT 提取中排除的文档路径模式，如 <code>["api/**"]</code> 。默认为 <code>[]</code> 。文档仍保留在 HTML 和 PDF 中。
<code>gettext_last_translator</code> 、 <code>gettext_language_team</code>	模板头部的 <code>Last-Translator</code> 和 <code>Language-Team</code> 。
<code>figure_language_filename</code>	构建语言对应的图片文件。默认 <code>"{root}.{language}{ext}"</code> ：若 <code>language = "de"</code> ，且文件存在， <code>img/logo.png</code> 会改用 <code>img/logo.de.png</code> 。可用字段为 <code>{root}</code> 、 <code>{path}</code> 、 <code>{basename}</code> 、 <code>{ext}</code> 、 <code>{docpath}</code> 和 <code>{language}</code> 。
<code>translation_progress_class</code>	给已翻译元素添加 <code>translated</code> 类，其他元素添加 <code>untranslated</code> 类（ <code>true</code> ）；也可只添加其中一种（ <code>"translated"</code> 或 <code>"untranslated"</code> ），以使用样式表显示翻译进度。

## Guidedog 的界面用语

Guidedog 自身也生成文字：提示框标题、“Added in version 2.1”、“Fig. 1”、“Contents”，页面布局中的“Previous”、“Next”、“Navigation”和“Search”，搜索页和索引页，以及书籍中的“Version”。这些来自 `guidedog` 文本域，按以下顺序查找：

1. 项目的 `locales/LANGUAGE/LC_MESSAGES/guidedog.po`。
2. 项目的 `locales/LANGUAGE/LC_MESSAGES/sphinx.po`，与 Sphinx 一致。
3. Guidedog 内置的清单，支持德语、西班牙语、法语、日语、巴西葡萄牙语、俄语及简体中文。

消息标识符使用英文；Sphinx 有对应文字时，采用相同标识符，因此项目现有的 `sphinx.po` 仍可使用。项目自己的 `guidedog.po` 可以覆盖任意文字，或增加一种语言。

模板与 Sphinx 一样使用 `_()`、`gettext()` 和 `ngettext()` 翻译：

```
<small>{{ _('Previous') }}</small>
```

```
{{ _('Last updated on %s.')|format(last_updated) }}
```

除非在 `conf.toml` 中明确设置，`numfig_format`、`html_title` 和 `html_short_title` 会跟随语言。

## 日期

`|today|`、书籍日期和页面的 `last_updated` 采用 Sphinx 的格式规则：`today_fmt` 和 `html_last_updated_fmt` 使用 `strftime` 格式，月份和星期名称遵循 `language`。例如 `%b %d, %Y` 在英文中是“Sep 29, 2026”，在德文中是“Sept. 29, 2026”。空格式取 `_(' %b %d, %Y')`，可由目录文件翻译。上述语言的名称来自 Unicode CLDR；其他语言使用英文，与 Sphinx 遇到 Babel 不支持的语言时相同。

日期取本地时区的本地日期。设置 `SOURCE_DATE_EPOCH`（自 1970 年起的 UTC 秒数），可实现可重复构建，使各机器使用相同日期。

## 链接到其他项目

启用 `sphinx.ext.intersphinx` 后，本项目无法解析的引用会到其他项目的清单中查找，即各 Sphinx 或 Guidedog 站点发布的 `objects.inv`。在 `conf.toml` 中列出这些项目：

```
extensions = ["sphinx.ext.intersphinx"]

[intersphinx_mapping]
python = ["https://docs.python.org/3/", ""]
click = ["https://click.palletsprojects.com/", ["click.inv", ""]]
```

每项包含其他项目的页面地址及清单位置。清单可用 URL、相对于源目录的文件路径，或用 `""` 表示页面地址下的 `objects.inv`。列表按顺序尝试。这是 Sphinx 的 `intersphinx_mapping`，用 `""` 代替 Python 的 `None`；`guidedog migrate` 会进行转换。

之后，引用无需特殊写法：

```
:func:`len`, :class:`list`, and :ref:`tut-packages` link into Python's
documentation.
:ref:`click:testing` looks only in Click's inventory.
:external+python:ref:`tut-packages` never looks in this project.
```

标签和文档链接显示其他项目的标题，并附有“(in Python v3.14)”这样的提示。与 Sphinx 一样，`:doc:` 引用需指定项目名（`:doc:`python:tutorial/index``），因为 `intersphinx_disabled_reftypes` 默认为 `["std:doc"]`。

构建开始时获取清单，并在 `_build/.doctrees/_intersphinx_cache__` 中按 `intersphinx_cache_limit` 缓存若干天（默认 5 天，负值表示保留）；本地文件每次构建都读取。与 Sphinx 一样，清单 URL 重定向后采用新目录作为链接基址，缓存也记录此基址，因此使用缓存与首次获取时的链接一致。例如 `https://werkzeug.palletsprojects.com/objects.inv` 会重定向至 `/en/stable/`。`intersphinx_timeout` 以秒限制每次下载。无法访问的项目会给出警告，其引用保持未解析状态。

Guidedog 生成的 `objects.inv` 包含各页面、标签、术语和对象，其他项目可用相同方式链接到这里。

## 声明对象类型

项目可能需要标准领域不认识的名称。在 TOML 中声明对象类型后，Guidedog 无需加载 Python 扩展即可创建目标、索引项和角色。声明描述语法，不是可执行行为。

### 目标：add\_crossref\_type

```
[[crossref_types]]
directive = "setting"
role = "setting"
index = "pair: %s; setting"
```

随后，`.. setting:: DEBUG` 创建标识符为 `setting-DEBUG` 的目标及索引项，`:setting:`DEBUG`` 则链接至该目标，与 Sphinx 一致。`index` 对应 Sphinx 的 `indextemplate`：首个冒号前是类型（`single`、`pair`、`triple`、`see`、`seealso`），其后是索引项，`%s` 代表名称。

### 描述：add\_object\_type

```
[[object_types]]
directive = "django-admin"
role = "djangadmin"
index = "pair: %s; django-admin command"
name = "first-word"
display = "django-admin %s"
program = true
```

随后，`.. django-admin:: migrate [app_label]` 像普通对象一样描述命令，包含签名、内容、标识符（`django-admin-migrate`）及索引项。Sphinx 可用 `parse_node` 函数解析签名，这里改用字段表达常见处理规则：

#### name

"whole"（默认）：对象名是完整签名。"first-word"：对象名为首个空格之前的内容。例如 `pdb` 的 `b(reak) [lineno]` 命名为 `b(reak)`。

#### display

签名的显示方式，`%s` 代表签名。默认保留原样。

#### program

`true`：名称成为后续选项所属的程序，与 `.. program::` 相同。

## 指令的其他名称

扩展可以用另一个名称注册 Sphinx 自有的指令：

```
[directive_aliases]
django-admin-option = "option"
awaitablefunction = "py:function"
```

未指定领域的目标是标准指令，或默认领域的指令；`py:function` 这样的目标则明确指定领域。

## 迁移

`guidedog migrate` 找出 `conf.py` 所列、属于项目自身模块的扩展。它只读取 `add_crossref_type`、`add_object_type` 和 `add_directive` 调用，不执行代码，并写

入相应表。无法声明的内容会报告：`parse_node` 函数（需设置 `name`、`display`、`program` 以匹配），以及用 Python 编写的指令、角色和事件处理器。

## 用 autodoc 编写 Python 文档

Guidedog 通过读取源代码生成 Python 文档，不会导入软件包。docstring 和函数签名成为文档图的一部分。仅在运行时创建的对象需要显式说明。这是有意划定的边界。

### 配置方法

列出扩展，并告诉 Guidedog 软件包相对于源目录的位置，就像 `sys.path` 告诉 Python 那样：

```
extensions = ["sphinx.ext.autodoc"]
autodoc_source_paths = ["../src"]
```

`guidedog migrate` 会读取 `conf.py` 中用于定位软件包的语句，例如 `sys.path.insert(0, os.path.abspath('../'))`，并为你生成 `autodoc_source_paths`。对于安装在其他位置的软件包，必须添加其所在目录，例如依赖项的源代码检出目录；Guidedog 不会搜索 Python 自己的 `site-packages`。

接下来，这些指令的用法与 Sphinx 相同：

```
.. automodule:: shop.cart
 :members:
 :show-inheritance:

.. autoclass:: shop.Cart
 :members: add, remove
 :inherited-members:
```

支持 Sphinx 9.1 的全部选项，含义也相同：`:members:`、`:undoc-members:`、`:private-members:`、`:special-members:`、`:inherited-members:`、`:exclude-members:`、`:member-order:`、`:show-inheritance:`、`:imported-members:`、`:ignore-module-all:`、`:class-doc-from:`、`:no-value:`、`:annotation:`、`:no-index:`、`:no-index-entry:`、`:synopsis:`、`:platform:`、`:deprecated:`，以及取消 `autodoc_default_options` 的 `no-` 形式。

### 设置

这些 `conf.toml` 设置沿用 Sphinx 的名称、取值和默认值。

#### **autodoc\_source\_paths**

Guidedog 专用设置：按顺序列出查找 Python 模块的目录，路径相对于源文件目录。只读取这些目录下的文件；若符号链接指向目录外部，则拒绝访问。

#### **autoclass\_content**

指定用于描述类的文档字符串：`"class"`（默认）、`"init"` 或 `"both"`。

#### **autodoc\_class\_signature**

`"mixed"`（默认）或 `"separated"`；后者将 `__init__` 单独作为方法记入文档。

**autodoc\_default\_options**

为所有指令设置默认选项的表，例如 `members = true` 或 `member-order = "bysource"`。值为 `false` 时不添加该选项。

**autodoc\_docstring\_signature**

`true` (默认)：若文档字符串首行为 `name(args) -> result` 这样的形式，则将其作为签名。

**autodoc\_inherit\_docstrings**

`true` (默认)：没有文档字符串的成员继承基类的文档字符串。

**autodoc\_member\_order**

`"alphabetical"` (默认)、`"groupwise"` 或 `"bysource"`。

**autodoc\_preserve\_defaults**

`false` (默认)：按 `repr()` 的表示方式显示默认值；`true` 则保留源码中的写法。

**autodoc\_typehints**

`"signature"` (默认)、`"description"` (使用 `:type:` 和 `:rtype:` 字段)、`"both"` 或 `"none"`。

**autodoc\_typehints\_description\_target**

`"all"` (默认)、`"documented"` 或 `"documented_params"`。

**autodoc\_typehints\_format**

`"short"` (默认) 或 `"fully-qualified"`。

**autodoc\_use\_type\_comments**

`true` (默认)：将 `# type:` 注释作为类型注解。

**strip\_signature\_backslash**

与 Sphinx 一样，将签名中的反斜杠加倍。

**autodoc\_mock\_imports, autodoc\_type\_aliases, autodoc\_warningiserror**

接受这些设置。由于不会导入模块，因此无需模拟导入；不应用类型别名。

## Guidedog 如何推断 Python 的行为

名称按照 Python 的绑定规则解析。例如，在包的 `__init__.py` 中写入 `from .app import Flask, flask.Flask` 就指向 `flask/app.py` 中定义的类，autodoc 会标注 `:canonical: flask.app.Flask`。基类也按相同方式解析；方法解析顺序按 Python 的规则计算，继承的成员从基类源码中读取。仅在 `if TYPE_CHECKING:` 下导入的名称在运行时不存在，因此引用这些名称的注解会保留原样，与 Sphinx 一致。

文档字符串与 Python 实际存储的内容一致：处理转义，并按 Python 3.13 编译器的规则去除缩进。属性文档来自 Sphinx 分析器读取的位置：赋值行末尾或上方的 `#:` 注释，以及赋值后紧接的字符串字面量。函数签名来自 `def`。类的签名按照 Python 的查找规则来自元类的 `__call__`、`__new__` 或 `__init__`；`dataclass` 或 `NamedTuple` 的签名来自字段。`@overload` 变体会替代实现本身的签名。

## 不运行 Python 就无法获知的内容

如果源码无法揭示 Python 的实际行为，Guidedog 会明确指出涉及的对象。无法生成文档时给出警告；能生成文档但信息少于 Sphinx 时给出说明，使用 `-v` 可查看。

- 编译后的模块（`.so`、`.pyd`）没有可读取的源码，因此找不到其中的对象。
- 运行代码得到的值（`app = Flask(__name__)`、`now = datetime.now()`）不会生成 `:value:`；需要计算的默认值保留源码写法。
- 由源码搜索路径之外的装饰器（`@click.command()`）生成的函数，按被装饰的原函数生成文档。
- 源码搜索路径之外的包中的类，使用导入时的路径命名；该路径可能与定义类的模块不同。它的成员、构造函数签名以及可继承的文档字符串都无法确定。
- 无法读取内置类型的文档字符串。因此，如果某个方法没有自己的文档字符串，只能从 `dict` 继承，就不将它列入文档。
- 不会执行处理 `autodoc` 事件（`autodoc-process-docstring`、`autodoc-skip-member` 等）的 Python 扩展。
- 嵌套深度远超正常代码的模块不会进行分析，并给出 `autodoc.too_deep` 警告。Python 本身拒绝 200 层嵌套括号；Guidedog 也拒绝含数百个操作数的 `+` 链。
- 会产生倍增的工作量都有上限。沿导入和别名查找名称时，单条路径最多 48 步，总计最多 100,000 步。类型别名或常量展开最多 10,000 步，超过后保留源码写法。类的 MRO 最多追踪 100 层；单条指令最多生成 10,000 条指令，超过时给出 `autodoc.limit` 警告。只有以自身名称包含自身的类才会达到最后这一上限。

将缺失包的源码目录加入 `autodoc_source_paths`，即可解决上述大部分限制。

## 重新构建

`autodoc` 指令读取的每个 Python 文件都是该页面的依赖项。修改文件后，下次构建会重新读取页面。未找到对象的页面会在每次构建时重读，因此源码中出现该对象后即可识别。

## 安全性

构建文档绝不运行项目代码。Guidedog 将 Python 源码作为文本读取，并使用与 `include` 相同的路径限制：仅访问 `autodoc_source_paths` 下的文件，不通过链接访问外部路径。

## 为 Odin 生成文档

Odin 域为包、过程、类型和成员提供命名目标。Guidedog 也能解析 Odin 源码生成 API 页面，不调用编译器，也不执行包。生成的签名仍需有用的契约。

## 描述对象

包为后续内容提供上下文，如同 Python 模块：

```
.. odin:package:: core:strings
```

```

:synopsis: Procedures to manipulate UTF-8 encoded strings.
:imports: rt=base:runtime

.. odin:procedure:: @(require_results) clone :: proc(s: string,
allocator := context.allocator) -> (res: string, err: rt.Allocator_Error)
#optional_allocator_error

 Clones a string.

:param s: The string to be cloned.
:result res: The cloned string.
:result err: An allocator error, or ``nil``.

.. odin:struct:: Builder :: struct

 A dynamic byte buffer.

.. odin:field:: buf: [dynamic]byte

```

签名按 Odin 声明形式写在一行中，显示也相同。以下示例的效果如下：

```
shapes.Kind :: enum u8
```

```
Circle = 1
```

```
@(require_results) shapes.area :: proc(side: f32, scale: f32
= 1) -> (a: f32, ok: bool) #optional_ok
```

测量一个正方形。

参数: **side** – 边长。

返回:

- **a** – 面积。
- **ok** – 类型是否已知。

指令及其接受的签名：

#### odin:package 和 odin:currentpackage

可写 `core:fmt`，或项目根目录下的路径，如 `readers/sphinx`。对象名称使用包导入路径：`core:fmt.println`。`odin:currentpackage` 只改变上下文、不生成目标；`None` 清空上下文。选项有 `:synopsis:`、`:platform:`、`:deprecated:`、`:no-index:` 和 `:imports:`。后者按 `alias=package` 列出签名使用的导入别名，如 `gd=core rst=readers/rst`，使 `gd.Node_Id` 链接到 `core.Node_Id`。`:private:` 列出私有类型及过程组的私有成员，如 `State Rules catalog_text`；签名将其显示为文本，因为没有描述可链接。生成页面自动列出它们。

#### odin:procedure (或 odin:proc)

`name :: proc(params) -> results`，支持名称前属性 (`@(require_results)`)、`proc` 前的 `#force_inline`、调用约定 (`proc "c" (...)`)、默认参数 (`x := 1`、`x: int = 1`)、`$T` 多态参数、`..` 可变参数、`using`、`#c_vararg`、命名结果、标签 (`#optional_ok`) 和 `where` 子句。`name(params) -> results` 是简写。

#### odin:procgroup

`name :: proc{a, b}`：每个成员链接到对应过程。

#### odin:struct, odin:union, odin:enum, odin:bitset, odin:bitfield, odin:type

`Name :: struct($T: typeid) #packed`、`Name :: union #no_nil {A, B}`、



```
Name :: enum u8、 Name :: bit_set[Flag; u8]、 Name :: bit_field u32。
odin:type 支持独立类型和别名 (Handle :: distinct uintptr、
Callback :: proc(x: int) -> bool)。类型内容包含其成员。
```

#### odin:field 和 odin:enumerator

类型内部使用 name: Type, 可带标签 (name: string `json:"n"`) 或位宽 (low: u8 | 3); 枚举成员使用 Name 或 Name = 3。其名称为 Type.member。

#### odin:const, odin:var, odin:foreign

```
NAME :: 64 或 NAME : int : 64; name: Type、 name := value 或
name: Type = value; 外部导入使用 libc "system:c"。
```

所有对象支持 Sphinx 域选项: :no-index:、:no-index-entry:、:no-contents-entry:、:no-typesetting:。另有 :package: (在其他包中描述)、:private: 和 :deprecated: message (签名未含时显示为 @private) 和 @(deprecated="message")、:availability: Windows, Linux (内容首行说明可用平台)、:foreign: libc (外部块过程)。多行签名描述同一对象在不同目标上的形式, 只有第一行获得 ID 和索引条目。

类型位置的名称链接到对应类型, 先在签名所属包和类型中查找, 再按原名称查找。内置类型 (int、string、rawptr、typeid、any 等)、关键字、多态名称 (\$T 及后续 T, 包括外层类型中的名称)、数组长度、默认值、常量、标签和 where 子句均原样显示, 不生成链接。

## 角色与标识符

:odin:pkg:、:odin:proc:、:odin:type: (结构体、联合体、枚举、位集合、位字段等类型)、:odin:const:、:odin:var:、:odin:field:、:odin:enumerator: 和 :odin:obj: (任意对象) 用于链接。与 Python 域一样, ~ 只显示末段 (:odin:proc:~core:fmt.println 显示 println), 前导 . 匹配任意对象名称的末尾。查找先在引用所属类型和包中, 再按原名, 再取包路径: 或 / 之后部分; 因此 fmt.println 找到 core:fmt.println, sphinx.Config 找到 readers/sphinx.Config, 符合 Odin 使用包路径末段命名的方式。当前包 :imports: 的别名优先解析。设置 default-domain:: odin 或 primary\_domain = "odin" 后, 角色无需 odin: 前缀。

对象 ID 用 Sphinx 的 make\_id 处理 odin- 与全名: odin-core-fmt.println、odin-readers-sphinx.Config.docname。与 Python ID 一样保留大小写和点; 导入路径中的 : 和 / 变为连字符。ID 可读、构建间稳定, 也区分末段同名的包。包 ID 为 odin-package- 加路径。索引如“println (procedure in core:fmt)”。objects.inv 按 odin 域和类型 (odin:procedure、odin:struct、odin:field 等) 列出所有对象, 包排在前面, 如 Sphinx 的模块。

内容中的信息字段与其他域一样分组: :param name: (配合 :type name:)、命名结果的 :result name:、:returns: 和 :rtype:。

## 从源码生成

指定包相对于源目录的位置:

```
odin_autoapi_dirs = ["../src"]
odin_collections = {core = "/usr/local/lib/odin/core"} # optional
```



`odin_autoapi_dirs` 下的包按相对路径命名 (`src/shapes/round` 为 `shapes/round`)；集合内的包按 Odin 导入名称命名 (`core:strings`)。生成指令随后按 `autodoc` 的方式工作：

```
.. odin:autopackage:: shapes
 :members:
 :undoc-members:
 :member-order: groupwise

.. odin:autoproc:: shapes.area
.. odin:autotype:: Shape
```

`odin:autopackage` 输出包及文档注释，带 `:members:` 时也输出声明（全部或指定项）。`odin:autoproc`、`odin:autoprocgroupp`、`odin:autotype`、`odin:autoconst`、`odin:autovar` 各输出一个声明，可用当前包名称或 `package.name`。选项：`:members:`、`:undoc-members:`、`:private-members:`（默认省略的 `@(private)` 声明）、`:exclude-members:`、`:member-order:`，其值为 `source`（默认，先文件名后位置）、`alphabetical`、`groupwise`（分为“Types”“Procedures”“Procedure groups”“Constants”“Variables”“Foreign imports”），以及 `:no-index:`。

文档注释是声明正上方的 `//` 行或 `/* */` 块，以及字段或枚举成员的行尾注释。它们是纯文本，按以下规则转为 `reStructuredText`：

- 段落仍是段落；由 `- item` 行组成的列表仍是列表。
- 空行之后，或以冒号结尾的行（`Example:`）之后，缩进块（制表符或空格）转为 `code-block:: odin`；`Output:` 后转为 `text`。若某行比紧接段落行的文本缩进更深，则续接该段落。
- `Inputs:` 后的 `- name: text` 列表转为 `:param name:` 字段；`Returns:` 条目转为 `:result name:`，无名称则转为 `:returns:`。这与 Odin 核心库的注释形式一致。
- 以 `NOTE:` 或 `WARNING:` 开头的行成为注释或警告。
- ``code`` 是字面文本。其他会被 `reStructuredText` 解释为标记的字符（`*`、`|`、词尾 `_` 等）都进行转义，因此注释按原样显示，不产生警告。

## API 页面

设置 `odin_autoapi_dirs` 后，其中每个包也获得独立页面，在构建读取时生成，类似 `sphinx-autoapi`。`api/shapes` 使用 `odin_autoapi_options` 包含 `odin:autopackage:: shapes`；`api/index` 列出各包和简介。页面属于项目文档，可进入 `toctree`、搜索、总索引、`singlehtml` 和 PDF，但不写入源目录。设置如下：

### `odin_autoapi_dirs`

需要生成页面的包目录，相对于源目录。只读取这些目录及 `odin_collections` 内的文件；指向外部的链接会被拒绝。

### `odin_collections`

集合名称与目录的表，用于 `core:strings` 等指令参数。这些包不生成页面。

### `odin_autoapi_root`

页面目录，默认 `"api"`。若源文档已占用某页面名称，则保留源文档并发出警告。

**odin\_autoapi\_options**

默认 ["members", "undoc-members"]; 可加入 private-members。

**odin\_autoapi\_member\_order**

"source" (默认)、"alphabetical" 或 "groupwise"。

**odin\_autoapi\_add\_tocentry\_entry**

true (默认): api/index 加入根文档的第一个 toctree。

**odin\_autoapi\_generate\_api\_docs**

true (默认); false 时只为指令读取目录。

未记录包的类型 (core:io.Writer) 无法链接。使用 -n 时, 可用 nitpick\_ignore\_regex = [["odin:type", "(core|base):.\*"]] 抑制这类警告。Guidedog 的 docs/api 也采用此设置。

## 平台

按 Odin 对文件的判断, 为包的各构建目标生成文档。\*\_windows.odin、\*\_linux\_arm64.odin、\*\_amd64.odin 只适用于对应目标; #+build linux, darwin、#+build !windows (及旧式 //+build) 进一步限制。文件级 when ODIN\_OS == .Windows 若只比较 ODIN\_OS 和 ODIN\_ARCH, 其分支归对应目标, else 归其余目标。省略 \*\_test.odin 和 #+ignore 文件。所有目标共有的声明只显示一次; 部分目标的声明标明平台, 如“Availability: Windows”; 签名不同时每种签名显示一次并标明目标, 仅第一种有 ID。只支持部分目标的包用 :platform: 说明。

## 重新构建

页面读取的每个 Odin 文件都成为该页依赖。修改包源码后, 下次构建只重读记录该包的页面。包新增、删除或简介改变时, 重读包索引。guidedog serve 在提供 HTML 页面前检查 odin\_autoapi\_dirs 和 odin\_collections 的修改时间; 编辑后刷新浏览器。

## 限制

- 源码仅解析, 不编译或类型检查: 类型和常量值按原样显示, 常量值超过 100 字符则省略, 不做推断。A :: B 是类型还是常量由名称判断: 首字母大写但非全大写的名称, 或内置类型, 按类型处理。
- when 中除 ODIN\_OS 和 ODIN\_ARCH 比较之外的条件不求值, 两分支都会记录。
- 声明引用私有类型时显示为文本, 因为除非指定 :private-members:, 私有声明没有条目。生成的包通过 :private: 列出它们, 因此 -n 不会报告这类引用。
- core:odin/parser 拒绝同时带 #optional\_ok 和 where 子句的过程。此类文件会被报告, 能读取的部分仍生成文档。
- Odin 解析器处理深度嵌套需要大量栈, 数百层括号可使线程栈溢出。远超实际代码深度的文件 (约 50 层括号或类型, 或同一链约千个 else) 会以 odin.autodoc.nesting 警告省略, 其余包内容照常记录。when 条件最多求值 64 层 &&、|| 和 !, 更长则视为未求值。

---

# 实用任务

---

每页从一个具体任务开始，列出命令及其结果。

## 构建并安装 Guidedog

### 在 macOS 上用 Homebrew 安装

在 Apple 芯片或 Intel 芯片的 Mac 上，使用官方 tap `insanai/homebrew-guidedog` 安装：

```
brew tap insanai/guidedog
brew install insanai/guidedog/guidedog
guidedog --version
```

Homebrew 会选择适合处理器的发行包，并校验 SHA-256。安装包已包含 Typst、Graphviz 和 tree-sitter。生成 HTML 和 PDF 无需 Odin 编译器、Python 环境或单独安装 Typst。此版本要求 macOS 12 或更高版本。

按照 [第一个项目](#) 创建第一个项目。更新已安装的版本：

```
brew update
brew upgrade insanai/guidedog/guidedog
```

运行 `brew test insanai/guidedog/guidedog` 检查已安装的包。测试会创建项目并生成 HTML 和 PDF。

### 从源码构建

使用 `tools/odin-release.txt` 指定的 Odin 工具链。发行版使用 Odin `dev-2026-10` 和 Typst 0.15.1。动态链接版本需要 Graphviz 开发库。

### 开发版构建

在仓库根目录运行：

```
mkdir -p build
native/tree-sitter/build.sh
odin build cmd/guidedog -out:build/guidedog -o:speed -vet -strict-style
build/guidedog --version
build/guidedog formats
```

tree-sitter 构建提供固定版本的 Python 语法，用于基于源代码的 autodoc。通过系统包管理器安装 Graphviz 开发文件。没有 Typst 桥接的构建使用已安装的 typst 程序生成 PDF。可用 `--typst=/absolute/path/to/typst` 显式选择。

## 内嵌编译器

先构建原生依赖，再将其链接至 Odin 可执行程序：

```
cargo build --release --manifest-path native/typst_bridge/Cargo.toml
native/graphviz/build.sh
odin build cmd/guidedog -out:build/guidedog -o:speed \
 -define:GUIDEDOG_TYPS_T_BRIDGE=true -define:GRAPHDOG_STATIC=true
```

Typst 桥接是 Rust 代码，位于 Guidedoc 的无分配边界之外。嵌入式构建无需启动单独的 Typst 可执行文件，但不会消除编译器自身的内存开销。

在 Windows 上，将输出命名为 `build/guidedog.exe`。若希望直接输入 `guidedog`，无需路径，将可执行文件加入 `PATH`。

## 构建并发布文档

构建到项目的默认输出目录：

```
guidedog build html docs/manual -j auto -W
guidedog build pdf docs/manual -W
```

网站位于 `docs/manual/_build/html`。PDF 和生成的 Typst 源文件位于 `docs/manual/_build/pdf`。Web 服务器可将 HTML 目录作为静态文件提供；生产环境不需要 Odin 服务器。

## 选择输出目录

Sphinx 风格的命令形式明确指定源目录和目标目录：

```
guidedog build -b html docs/manual build/manual-site -W
```

输出应放在源目录之外。若发布会替换源树，Guidedog 会拒绝该布局。自定义目标使用稳定的同级缓存，如 `build/.manual-site.guidedog/.doctrees`，因此发布目录只包含发布内容。

## 预览与检查

```
guidedog serve docs/manual
guidedog build dummy docs/manual -n -W
```

dummy 构建器读取并解析项目关系，但不生成发布内容。可用于快速语义检查，不能验证最终页面布局。

## 可复现的时间

```
SOURCE_DATE_EPOCH=1790812800 guidedog build html docs/manual -W
```

这会固定构建时间。可重复性还取决于源文件、模板、字体、编译器版本和资源。只固定时钟还不够。

构建成功后发布完整目录。进程成功退出是必要条件；修改模板后仍需视觉检查。

## 迁移 Sphinx 项目

Guidedog 沿用 Sphinx 的项目结构和许多设置名称，但不运行 `conf.py` 或 Python 扩展。迁移应先把数据与可执行配置分开。

```
guidedog migrate path/to/conf.py
guidedog build html path/to/docs --diagnostics=json
```

`migrate` 读取字面量设置并写入 `conf.toml`。遇到需要人工选择的计算值，会给出报告。它不会导入项目或运行 Python 来求这些值。确认配置等价之前，先检查生成结果。

### 分三步完成

1. 先确保文档图正确，检查根文档、后缀、排除项和 `include`。
2. 再确保引用正确，检查领域、清单和自定义对象类型。
3. 最后确保呈现正确，检查模板、图表、公式及两种输出。

未知指令可能来自项目私有扩展。HTML 构建成功，并不能证明私有语义已复现。阅读诊断，再决定替换指令、配置受支持的别名，还是保留并记录限制。

受支持的功能，以及 CPython、Django 和 Flask 的验证证据，见 [与 Sphinx 的兼容性](#)。声明式自定义对象见 [声明对象类型](#)。

## 为代码库生成文档

公开名称需要契约。说明输入、结果、所有权和失败条件。只有签名，不能算手册。

### Odin

```
odin_autoapi_dirs = ["../lib"]
odin_autoapi_root = "api"
odin_autoapi_options = ["members", "undoc-members"]
```

Guidedog 通过 Odin 的解析器读取源代码，不编译或运行被记录的包。生成的页面加入文档图和对象清单。本仓库的独立 API 项目是 `docs/api`。指令、签名和选项见 [为 Odin 生成文档](#)。

### Python

```
extensions = ["sphinx.ext.autodoc"]
autodoc_source_paths = ["../src"]
```

Guidedog 解析 Python 源代码，而不导入模块，以避免导入产生副作用。运行时生成的对象因此可能缺失。若它们属于公开接口，应显式记录。准确的发现规则见 [用 autodoc 编写 Python 文档](#)。

### 跨项目链接

将生成的 `objects.inv` 随网站一起发布。其他项目即可通过 `intersphinx` 映射解析名称。为实现可重复构建，应显式设置映射。见 [链接到其他项目](#)。

## 维护本项目的文档

Guidedog 有三个项目：docs/manual、docs/api 和 docs/gds。手册和 GDS 均由 `guidedog quickstart` 创建。直接编辑生成的模板。不要把设计记录放进手册目录。

### 构建手册

在仓库根目录运行：

```
build/guidedog build html docs/manual -j auto -W
build/guidedog build pdf docs/manual -W
build/guidedog serve docs/manual
```

手册目录包含完整的 Odin API 参考。PDF 命令生成一本书，解释与声明相互链接。手册和独立 API 项目都从 `lib/` 生成参考，无需维护复制的 API 稿件。

单独构建 API 项目：

```
build/guidedog build html docs/api -W
build/guidedog build pdf docs/api -W
```

### 构建并检查讨论文档

```
build/guidedog gds index
build/guidedog gds check
build/guidedog gds build --target=both
```

GDS CLI 管理元数据、编号、状态和恢复。RST 发布路径调用项目引擎，生成完整 HTML 网站及每条记录的独立 PDF。选择器仅限制 PDF 范围；HTML 始终保留完整目录和连贯网站。选定 PDF 的预览使用 `_build/selected/<artifact>/pdf`，不影响 `_build/pdf` 中的完整目录。

```
build/guidedog gds build guidedoc-odin --target=pdf
build/guidedog build html docs/gds -W
build/guidedog build pdf docs/gds -W
```

普通项目命令也适用。GDS 项目使用自己的 `conf.toml` 和 `_build` 目录。编写流程见 `docs/gds/workflow.rst`。

### 发布之前

检查链接和诊断。检查宽表格与代码块。修改排版后，将 PDF 页面渲染出来检查。HTML 要在桌面和手机宽度、浅色和深色模式下测试。确保离线时数学公式和图示仍能工作。修改引擎时，运行相关库测试和集成测试。

---

# 参考

---

这些页面说明各项约定，便于搜索和查阅。需要第一个示例时，请从教程开始。

## 命令行参考

`guidedog` 提供命令行工具，用于管理、构建、预览、翻译和转换文档项目。

### 基本语法

```
guidedog <command> [arguments] [options]
guidedog --version
guidedog --help
```

## 退出码

表 6 退出状态

退出码	条件
0	成功：所有操作均已完成，没有未处理的错误。
1	构建或转换失败，或 <code>-w</code> 将警告视为错误。
2	命令行语法、参数或选项有误。
3	达到配置的资源上限，或无法提供内存。
4	文件操作或外部编译器失败。
5	内部错误；诊断信息说明如何报告。
70	发生崩溃；此前发布的输出仍然可用。

## 命令概览

表 7 CLI 命令

命令	用途
<code>quickstart [DIR]</code>	创建文档项目，并生成可直接编辑的模板和配置。
<code>build [BUILDER] [DIR]</code>	按指定的输出格式构建文档项目。
<code>serve [DIR]</code>	启动本地 HTTP 预览服务器，并在文件修改后自动重建。
<code>clean [DIR]</code>	删除生成的构建产物和缓存。
<code>convert INPUT</code>	直接转换单个文件或标准输入。
<code>migrate [conf.py]</code>	将 Sphinx 的 <code>conf.py</code> 转换为声明式 <code>conf.toml</code> 。
<code>intl &lt;update stat&gt;</code>	更新 PO 翻译目录，或报告翻译完成情况。
<code>formats</code>	列出内置的读取器、渲染器及格式符合性说明。
<code>gds &lt;COMMAND&gt;</code>	管理 Guidedog Discussions 的设计记录及其生命周期。

### guidedog quickstart

创建项目目录，其中包含 `conf.toml`、`index.rst`、`_templates/` 中的模板和 `_static/` 中的样式：

```
Interactive prompt
guidedog quickstart docs

Non-interactive creation with predefined options
guidedog quickstart docs -q -p "My Project" -a "Author Name" -v 1.0 -l en
```

选项：

- `-q`: 静默模式；跳过交互提问，使用命令行提供的值。
- `-p`, `--project NAME`: 设置项目显示名称。
- `-a`, `--author NAME`: 设置作者或组织名称。
- `-v VERSION`: 设置简短版本号，例如 `1.0`。
- `-r RELEASE`: 设置完整发布标识，例如 `1.0.0-rc1`。
- `-l LANGUAGE`: 设置项目默认语言，默认为 `en`。
- `--sep`: 分别创建 `source/` 和 `build/` 目录，而不是将源文件放在根目录。
- `--suffix EXT`: 设置源文档的默认扩展名，默认为 `.rst`。

### guidedog build

协调读取源文件、解析目录树和交叉引用，以及发布最终产物：

```
Standard project build
guidedog build html docs
guidedog build pdf docs

Sphinx-build compatibility syntax
guidedog build -b html docs _build/html
guidedog build -M html docs _build
```

支持的构建器：

- `html`: 生成带搜索和导航的完整 HTML 网站。
- `dirhtml`: 生成目录式 HTML 地址，例如 `dir/index.html`。
- `singlehtml`: 将所有文档合并为一个 HTML 页面。



- `pdf`: 用 Typst 排版 PDF 图书；别名为 `latex` 和 `latexpdf`。
- `text`: 为每个文档生成纯文本文件。
- `gettext`: 将可翻译的文本提取为 GNU gettext POT 模板。
- `dummy`: 解析并检查文档树，不写入最终文件；用于语法检查。

构建选项：

- `-a`: 写入所有输出文件，不考虑修改时间。
- `-E`: 忽略缓存，从头重建环境。
- `-W`: 将所有警告视为构建失败。
- `--keep-going`: 配合 `-W` 使用时，继续处理剩余文档，退出前报告全部错误。
- `-n`: 严格引用检查；对所有未解析的交叉引用给出警告。
- `-j N` / `-j auto`: 设置并行解析线程数；`auto` 使用与 CPU 核心数相同的线程数。
- `-D name=value`: 仅为本次运行覆盖 `conf.toml` 中的设置，例如 `-D language=de`。
- `-D table.key=value`: 只设置表类型设置（例如 `html_context`）中的一个键，并保留其他键，与 `sphinx-build` 相同。
- `-t TAG`: 定义供 `only` 指令使用的条件标签。
- `-c DIR`: 指定 `conf.toml` 所在的目录。
- `-C`: 不读取任何 `conf.toml` 文件。
- `-v`: 详细模式；显示提示信息和耗时统计。
- `-q`: 静默模式；只显示警告和错误。
- `--color=always|never|auto`: 控制终端的 ANSI 颜色输出。
- `--diagnostics=json`: 向标准错误输出写入可供程序读取的 JSON 诊断信息。
- `--budget=MIB`: 设置项目宿主持有的内存预算上限，默认为 1024 MiB。
- `--memory=ram|disk`: 指定内存预算耗尽后的处理方式。
- `--untrusted`: 省略原始嵌入标记，并禁用网络获取。这是能力限制策略，不是进程沙箱。

## guidedog serve

启动本地 HTTP 服务器和文件监视器；源文档、模板或资源修改后，自动重建并刷新浏览器：

```
Serve current project on default port (8000)
guidedog serve docs

Serve on custom port and host
guidedog serve docs --port 9000 --host 0.0.0.0
```

选项：

- `-p, --port PORT`: 本地监听端口，默认为 8000。
- `--host HOST`: 绑定的 IP 地址，默认为 127.0.0.1。

## guidedog clean

删除构建目录及缓存的环境状态：

```
guidedog clean docs
```

## guidedog convert

直接转换文件或标准输入，无需项目目录或配置：

```
Convert reStructuredText to HTML
guidedog convert guide.rst --output guide.html

Convert Markdown to PDF
guidedog convert guide.md --to pdf --output guide.pdf

Stream conversion from standard input to standard output
cat document.md | guidedog convert - --from commonmark --to html --output -
```

选项：

- `-o`, `--output FILE`: 输出文件路径；`-` 表示标准输出。
- `--to FORMAT`: 目标格式：`html`、`pdf` 或 `text`；也可根据输出文件名推断。
- `--from FORMAT`: 源读取器：`rst`、`commonmark`、`myst` 或 `typst`；也可根据输入扩展名推断。
- `--force`: 覆盖已有输出文件，不再询问。
- `--strict`: 出现任何警告即失败。
- `--raw=allow|omit`: 控制原始嵌入代码的处理策略，默认为 `omit`。
- `--emit-typst FILE`: 另存生成的中间 Typst 源码。

## guidedog migrate

将 Sphinx 的 Python 配置文件 `conf.py` 转换为声明式 `conf.toml`：

```
Convert conf.py in-place
guidedog migrate docs/conf.py

Output to a specific file
guidedog migrate docs/conf.py -o docs/conf.toml --force
```

选项：

- `-o FILE`: 写入指定路径，或用 `-` 写入标准输出。
- `--force`: 覆盖已有的 `conf.toml`。

## guidedog intl

管理多语言文档的翻译目录：

```
Step 1: Extract POT templates
guidedog build gettext docs

Step 2: Create or update language PO catalogs
guidedog intl update -l de -l fr docs

Check translation completion percentages
guidedog intl stat -l de docs
```

`intl update` 的选项：

- `-l`, `--language LANG`: 目标语言代码；可重复指定多种语言。
- `-p`, `--pot-dir DIR`: 指定已提取的 POT 消息模板所在路径。

## guidedog formats

显示已编译的读取器、渲染器、语法高亮器，以及规范测试的符合性结果：

```
guidedog formats
```

## guidedog gds

管理 Guidedog Discussions 的架构设计提案 (RFC):

```
guidedog gds new "Feature Title" --author="Name"
guidedog gds list --state=prediscussion
guidedog gds show 0003
guidedog gds promote 0003 --dry-run
guidedog gds state 0003 --to=accepted --reason="Consensus reached"
guidedog gds index
guidedog gds check --render
guidedog gds recover --rollback
```

## 配置参考

每个 Guidedog 项目都使用根目录下的 `conf.toml`。配置是声明式 TOML 数据，不会执行任意代码。

许多配置键与 Sphinx 的 `conf.py` 变量对应，因此可用 `guidedog migrate conf.py` 辅助迁移。

### 最小配置

项目只需基本信息。省略的设置使用默认值：

```
project = "Field Notes"
author = "Your Name"
version = "1.0"
release = "1.0.0"
language = "en"
root_doc = "index"
source_suffix = [".rst", ".md"]
templates_path = ["_templates"]
html_static_path = ["_static"]
pdf_paper_size = "a4"
```

## 项目元数据

表 8 元数据选项

键	类型	默认值	说明
<code>project</code>	字符串	<code>"Python"</code>	供读者识别的项目名称。
<code>author</code>	字符串	<code>""</code>	作者或组织名称。
<code>copyright</code>	字符串	<code>""</code>	显示在页脚的版权声明。
<code>version</code>	字符串	<code>""</code>	简短版本号，例如 <code>"1.0"</code> 。
<code>release</code>	字符串	<code>""</code>	完整版本号，可包含 alpha 或 beta 标识，例如 <code>"1.0.0b1"</code> 。
<code>language</code>	字符串	<code>"en"</code>	用于排版、断词和翻译的 ISO 语言代码，例如 <code>"de"</code> 或 <code>"zh_CN"</code> 。
<code>today</code>	字符串	<code>""</code>	自定义日期文字；省略时用 <code>today_fmt</code> 格式化当前日期。
<code>today_fmt</code>	字符串	<code>""</code>	文档日期的格式字符串。

## 常规选项

表 9 发现与解析选项

键	类型	默认值	说明
<code>root_doc</code>	字符串	<code>"index"</code>	作为根目录的文档。
<code>source_suffix</code>	数组	<code>[".rst"]</code>	可作为文档源文件读取的扩展名。
<code>exclude_patterns</code>	数组	<code>[]</code>	发现源文件时排除的目录和文件通配符模式。
<code>include_roots</code>	数组	<code>[]</code>	允许包含源文件的外部目录。
<code>templates_path</code>	数组	<code>[]</code>	HTML 和 PDF 模板目录；优先于内置模板。
<code>extensions</code>	数组	<code>[]</code>	启用受支持的内置 Sphinx 扩展，例如 <code>"sphinx.ext.autodoc"</code> 。
<code>primary_domain</code>	字符串	<code>"py"</code>	未写前缀的指令和角色所用的默认域，例如 <code>"py"</code> 、 <code>"c"</code> 或 <code>"odin"</code> 。
<code>highlight_language</code>	字符串	<code>"default"</code>	代码块未指定语言时使用的默认语言。
<code>pygments_style</code>	字符串	<code>""</code>	浅色主题使用的 Pygments 高亮样式。
<code>pygments_dark_style</code>	字符串	<code>""</code>	深色主题使用的语法高亮样式。
<code>smartquotes</code>	布尔值	<code>true</code>	将直引号和连字符转换为排版用标点。
<code>rst_prolog</code>	字符串	<code>""</code>	添加到每个文档开头的 reStructuredText 片段。
<code>rst_epilog</code>	字符串	<code>""</code>	添加到每个文档末尾的 reStructuredText 片段。

## 编号与数学

表 10 编号选项

键	类型	默认值	说明
<code>numfig</code>	布尔值	<code>false</code>	启用后自动为图、表和代码块编号。
<code>numfig_secnum_depth</code>	整数	<code>1</code>	图编号包含的章节层级；例如 1 可生成 Fig. 2.1。
<code>numfig_format</code>	表	见下文	"figure"、"table" 和 "code-block" 的编号前缀格式。
<code>math_number_all</code>	布尔值	<code>false</code>	启用后自动为所有独立公式编号。
<code>math_eqref_format</code>	字符串	<code>"({number})"</code>	:eq: 公式引用的格式字符串。
<code>mathjax_path</code>	字符串	URL	MathJax JavaScript 文件的 CDN 地址或本地路径。

`numfig_format` 的默认值：

```
[numfig_format]
figure = "Fig. %s"
table = "Table %s"
code-block = "Listing %s"
section = "Section %s"
```

## 诊断与严格检查

表 11 诊断选项

键	类型	默认值	说明
<code>nitpicky</code>	布尔值	<code>false</code>	对未解析的交叉引用和失效目标发出警告。
<code>nitpick_ignore</code>	数组	<code>[]</code>	免于严格引用警告的 [type, target] 配对数组。
<code>suppress_warnings</code>	数组	<code>[]</code>	不应报告的警告类别代码。
<code>keep_warnings</code>	布尔值	<code>false</code>	在发布的文档中保留警告。

## HTML 输出选项

表 12 HTML 选项

键	类型	默认值	说明
<code>html_theme</code>	字符串	<code>"guidedog"</code>	生成 HTML 时使用的主题。
<code>html_theme_options</code>	表	<code>{}</code>	传递给 HTML 主题的键值选项。
<code>html_title</code>	字符串	自动推导	浏览器标签页显示的标题。
<code>html_short_title</code>	字符串	自动推导	导航路径中使用的简短标题。
<code>html_logo</code>	字符串	<code>""</code>	相对于源目录的项目标志图片路径。
<code>html_favicon</code>	字符串	<code>""</code>	网站图标文件路径。
<code>html_static_path</code>	数组	<code>[]</code>	复制到输出 <code>_static/</code> 目录的文件夹。
<code>html_extra_path</code>	数组	<code>[]</code>	原样复制到输出根目录的文件夹。
<code>html_css_files</code>	数组	<code>[]</code>	HTML 页面加载的自定义 CSS 文件名。
<code>html_js_files</code>	数组	<code>[]</code>	HTML 页面加载的自定义 JavaScript 文件名。
<code>html_permaLinks</code>	布尔值	<code>true</code>	为段落和章节添加固定链接。
<code>html_permaLinks_icon</code>	字符串	<code>"¶"</code>	固定链接使用的符号或文字。
<code>html_baseurl</code>	字符串	<code>""</code>	站点地图和元数据使用的规范基础 URL。
<code>html_context</code>	表	<code>{}</code>	供 Jinja 模板使用的自定义变量字典。
<code>html_additional_pages</code>	表	<code>{}</code>	自定义页面： <pre>{ "page_name" = 。 "template.html" }</pre>

## PDF 与 Typst 选项

表 13 PDF 选项

键	类型	默认值	说明
pdf_paper_size	字符串	"a4"	纸张格式: "a4" 或 "us-letter"。
pdf_logo	字符串	""	图书封面的标志图片路径。
pdf_toplevel_sectioning	字符串	"chapter"	对应图书分层的章节级别: "chapter" 或 "part"。
pdf_show_urls	字符串	"no"	打印版的网址显示方式: "no"、"inline" 或 "footnote"。
pdf_preamble	字符串	""	插入生成文档头部的原始 Typst 源码。
pdf_font_paths	数组	[]	查找自定义 OTF/TTF 字体的附加目录。
pdf_packages	字符串	"download"	包解析策略: "download" 或 "offline"。
typst	字符串	"typst"	系统 Typst CLI 的程序名或绝对路径。

### 定义 PDF 图书

[[pdf\_documents]] 表定义项目要生成的一本或多本图书:

```
[[pdf_documents]]
root = "index"
file = "guide.pdf"
title = "Field Notes Complete Guide"
author = "Author Name"

[[pdf_documents]]
root = "reference/index"
file = "reference.pdf"
title = "Field Notes Reference"
author = "Author Name"
```

## MyST Markdown 选项

表 14 MyST Markdown 选项

键	类型	默认值	说明
<code>myst_enable_extensions</code>	数组	<code>["dollarmath"]</code>	语法扩展： "colon_fence"、"deflist"、 "dollarmath"、"fieldlist"、 "tasklist"、"substitution"。
<code>myst_heading_anchors</code>	整数	<code>0</code>	自动生成标题锚点的层级深度； <code>0</code> 表示禁用。
<code>myst_substitutions</code>	表	<code>{}</code>	Markdown 文档可使用的变量替换，例如 <code>{key = "value"}</code> 。
<code>myst_url_schemes</code>	数组	<code>["http", ...]</code>	识别为外部链接的 URI 协议。
<code>myst_commonmark_only</code>	布尔值	<code>false</code>	仅使用 CommonMark 语法，不启用 MyST 扩展。

## Intersphinx 选项

```
extensions = ["sphinx.ext.intersphinx"]
intersphinx_cache_limit = 5
intersphinx_timeout = 30

[intersphinx_mapping]
python = ["https://docs.python.org/3/", ""]
click = ["https://click.palletsprojects.com/", ["click.inv", ""]]
```

完整流程见 [链接到其他项目](#)。

## Python autodoc 选项

表 15 Autodoc 选项

键	类型	默认值	说明
<code>autodoc_source_paths</code>	数组	<code>[]</code>	供静态分析查找 Python 模块的目录。
<code>autoclass_content</code>	字符串	<code>"class"</code>	类文档字符串的来源： "class"、"init" 或 "both"。
<code>autodoc_member_order</code>	字符串	<code>"alphabetical"</code>	成员排序方式： "alphabetical"、"bysource" 或 "groupwise"。
<code>autodoc_typehints</code>	字符串	<code>"signature"</code>	类型提示的位置： "signature"、"description" 或 "none"。
<code>autodoc_default_options</code>	表	<code>{}</code>	所有 <code>auto*</code> 指令使用的默认选项。

完整发现规则见 [用 autodoc 编写 Python 文档](#)。



## Odin API 文档选项

表 16 Odin 域选项

键	类型	默认值	说明
odin_autoapi_dirs	数组	<code>[]</code>	扫描 Odin 包的源目录。
odin_autoapi_root	字符串	<code>"api"</code>	生成 Odin API 文档的目标子目录。
odin_autoapi_options	数组	<code>["members", "undoc-members"]</code>	筛选要包含的成员。
odin_autoapi_member_order	字符串	<code>"source"</code>	成员排序方式: <code>"source"</code> 或 <code>"alphabetical"</code> 。

Odin 域的完整用法见 [为 Odin 生成文档](#)。

## 国际化选项

表 17 i18n 选项

键	类型	默认值	说明
locale_dirs	数组	<code>["locales"]</code>	查找 gettext 翻译目录的文件夹。
gettext_compact	布尔值	<code>true</code>	将同一目录内文档的消息合并到一个翻译目录。
gettext_uuid	布尔值	<code>false</code>	在 POT 消息中写入稳定的 UUID。
gettext_auto_build	布尔值	<code>true</code>	构建时将 <code>.po</code> 编译为二进制 <code>.mo</code> 目录。
figure_language_filename	字符串	<code>"{root}.\n{language}\n{ext}"</code>	选择本地化图片的文件名模板。

翻译流程见 [国际化](#)。

## 读取器、渲染器与构建器

读取器理解源语言；渲染器输出一棵文档树；构建器协调项目并发布输出。这是不同职责。

表 18 源文件处理路线

输入	读取器	路线
项目中的 RST	RST 与内置的 Sphinx 层	将项目图生成 HTML、Typst/PDF、文本或翻译清单。
项目中的 Markdown	兼容 MyST 的 Markdown 层	使用同一项目图和构建器。
<code>convert file.md</code>	默认使用 CommonMark	将一份文档生成 HTML 或 Typst/PDF。
<code>convert file.typ</code>	原生 Typst 适配器	在 Guidedoc 语义树之外编译 Typst。

## 构建器

### html

每份文档一个 HTML 文件，附带索引、搜索和清单。

### dirhtml

目录式文档 URL。

### singlehtml

一个主 HTML 页面，包含各文档章节和本地引用。

### pdf

构建项目定义的 Typst 图书；LaTeX 构建器别名也选择这条路线。

### text

每份文档一个纯文本文件。

### gettext

用于翻译的消息模板。

### dummy

读取并解析项目，不发布输出文件。

## 覆盖率是一种证据

`guidedog formats` 列出可执行文件实际包含的适配器。兼容性页面记录固定测试集和真实项目检查。通过 CommonMark 测试不代表完整支持 MyST；能读取 RST 也不代表复现所有 Python 扩展。见 [与 Sphinx 的兼容性](#)。

## 错误信息

诊断报告应说明问题、标明位置，并给出有用的下一步。Guidedog 将这些信息保存为结构化数据。终端显示增加颜色和源码标记；JSON 保留数据，不含终端装饰。

## 如何阅读诊断报告

```
UNDEFINED SUBSTITUTION
```

```
rst.substitution.undefined
```

```
guide/page.rst:4: See |logo| here.
 ^^^^^
```

```
The substitution "|logo|" is not defined.
No output was written.

Add a definition, e.g. ".. |name| replace:: text".
```

报告始终按以下顺序分为四部分：

1. **标题栏**。标题用大写字母说明问题类型；警告和说明分别以 `WARNING:` 和 `NOTE:` 开头。右侧显示报告代码，如 `rst.substitution.undefined`，这是可用于搜索的稳定名称。
2. **位置**。显示文件名、行号、源码行，以及问题文本下方的标记。若问题不涉及某一行，如目录缺失或参数错误，只显示路径，或不显示位置。
3. **发生了什么**。先解释问题；必要时再说明它对输出的影响，例如本次没有写入任何内容。
4. **如何处理**。支持彩色的终端用绿色显示具体建议。如果有多种修复方式，会依次列出，最可能适用的放在前面。

颜色只强调文字已表达的含义。`--color=never` 关闭颜色；`--color=always` 则在输出不是终端时也保留颜色。

### 每条提示都指明下一步

Guidedog 对自身消息也遵守这一要求：提示先给出动作（“Add”“Pass”“Set”“Rename”“Run”），再说明对象，如 `--budget=MIB`、`conf.toml` 中的 `root_doc` 或指令选项。`tests/hints_test.odin` 检查每条可发出的消息；提示为空、重复错误内容，或只说“report this”而不说明报告材料，测试都会失败。另允许两种已指出修正方法的形式：建议（“Did you mean numfig?”），以及无需操作时的“Nothing to do:”和原因。

若问题是 Guidedog 自身的错误，提示会明确说明，并请你提交所运行的命令、导致问题的输入，以及 `guidedog --version` 的输出。

### 供程序读取的报告

`--diagnostics=json` 将每条报告作为一个 JSON 对象写入标准错误，每行一个，便于编辑器和持续集成读取：

```
{ "schema": 1, "code": "rst.substitution.undefined", "severity": "error",
 "title": "UNDEFINED SUBSTITUTION", "message": "The substitution \"|logo|\" is not
 defined.",
 "hint": "Add a definition, e.g. \".. |name| replace:: text\".", "path": "guide/
 page.rst",
 "line": 4, "column": 5, "end_line": 4, "end_column": 11, "source": "See |logo| here.",
 "category": "" }
```

此处为便于阅读而折行；实际每个对象占一行。`schema` 是格式版本，仅字段变化时更新，读取其余内容前应先检查。列号从 1 开始，按 Unicode 标量计数，而非字节或终端格数。`source` 是完整原始行。终端仅显示错误附近的有限窗口，转义控制字符、测量宽字符后对齐指示符。`category` 是 `suppress_warnings` 使用的警告类别，也可为空。

### 内存不足时

内存不足也会生成诊断，不会导致崩溃。分为两种情况：

**host.budget (达到内存预算)**

构建达到允许持有的内存预算 (`--budget=MiB`, 默认 1 GiB)。报告指出步骤; 若是模板, 还指出申请内存的行。终端中会询问一次: 提高预算、改用磁盘工作内存, 或停止。其他环境直接停止; 可提前用 `--budget` 或 `--memory=disk` 选择。提示也会用数字给出预算, 例如:

```
Build again with --budget=256, or with --memory=disk to keep working memory
on
disk (slower); or split the work into smaller files.
```

建议值取“当前持有量加上被拒步骤所需内存的四倍”(该步骤还需保存输出及写入副本)与“旧预算的两倍”中的较大值, 向上取整到 64 MiB。当步骤能装入可用内存时, 不超过机器当时的可用量; 否则提示会明确说明, 并先建议 `--memory=disk`:

```
Build again with --memory=disk to keep working memory on disk (slower):
the step
needs a budget of about 1408 MiB, more than the 900 MiB this machine has
free. Or
free that memory and build with --budget=1408, or split the document.
```

**host.memory (内存不足)**

系统拒绝了预算允许的内存申请。命令停止, 不使用被拒的存储: 构建不发布内容, GDS 修改要么全部应用、要么不应用, `convert` 要么写出整页、要么不写。释放内存或减少线程 (`-j 1`) 后重试。发布决定一旦做出, 切换输出无需再申请内存, 因此拒绝申请不会造成半发布。若构建发现先前未完成的发布, 却没有内存规划恢复, 就保留现场, 交给下次构建。

两种情况都以状态 3 退出。

**退出状态**

每条命令都以以下状态之一结束, 脚本可据此区分失败类型:

状态	含义
0	成功。输出已写入; 若为构建命令, 也已发布。
1	输入有问题: 文档错误、策略拒绝 (未使用 <code>--raw</code> 的原始内容)、文档无法读取, 或 <code>-W</code> 、 <code>--strict</code> 将警告视为错误。
2	命令用法错误: 未知参数或命令、缺失或无效的值, 或者 <code>conf.toml</code> 、 <code>-D</code> 中的无效设置。
3	达到限制: 内存预算 ( <code>--budget</code> )、嵌套深度 ( <code>--max-depth</code> )、节点数 ( <code>--max-nodes</code> )、栈 ( <code>--stack-kib</code> )、源文件过大, 或系统无法提供内存。
4	文件或外部程序失败: 路径无法读取或写入、文件读取期间发生变化, 或者 Typst 无法启动或完成。
5	Guidedog 内部发生错误, 报告会请你提交问题。
70	Guidedog 崩溃时, 会打印 <code>GUIDEDOG CRASHED (internal error)</code> , 以及执行的命令、故障和 <code>guidedog --version</code> 的输出, 并给出报告地址: 构建指定的问题跟踪器, 或提供该构建的人。构建仅在最后发布, 因此崩溃会保留之前的输出。
130	命令被 Ctrl+C 中断, 已发布文件保持不变。

无论构建因何失败，上一代输出及构建记录均保持不变。修复问题后重新构建即可。

## 与 Sphinx 的兼容性

在有助于作者迁移项目的地方，Guidedog 遵循 Sphinx，并逐项衡量兼容性。它不执行 Python 配置或任意 Python 扩展。**构建成功只能证明构建完成，不能证明完全等价。**

### 项目与发布

表 20 项目操作

操作	已实现的行为
项目布局与构建参数	采用 Sphinx 风格的源目录、输出目录及参数： <code>-a -E -W -n -D -t -j -b -M -c -C</code> 。
配置	使用熟悉的 Sphinx 设置名，以 TOML 数据配置。迁移时转换字面量，并报告需要计算的值。
HTML 构建器	支持 <code>html</code> 、 <code>dirhtml</code> 和 <code>singlehtml</code> 。单页锚点包含文档名称。
其他构建器	支持 <code>text</code> 、 <code>gettext</code> 和 <code>dummy</code> 。 <code>pdf</code> 使用 <code>Typst</code> ； <code>latex</code> 和 <code>latexpdf</code> 也选择这条处理路线。
模板	Guidedog 自带的 Jinja 实现、保留类型的 <code>html_context</code> 、布局继承及附加页面。
增量构建	源文件、依赖项、模板、配置和引用查询的变化，使受影响的输出失效。
发布	一次发布完整的一代输出。构建失败时保留上一代已发布内容。

`-D` 的布尔值接受 `true`、`false` 或 `1`、`0`。若项目有 `contents` 而没有 `index`，可将 `contents` 作为根文档，但会给出警告。附加模板页面放在站点根目录，即使使用 `dirhtml` 也是如此。具体约定见 [模板](#)。

未变化的页面也会再次报告诊断，因此 `-w` 在全新构建和未变化的构建中含义一致。删除文档后，其页面也会移除。发布使用可恢复的日志；若提交中断，下次构建会完成恢复。不变式及平台边界见 [发布完整的一代输出](#)。

不提供 `epub`、`man`、`texinfo`、`linkcheck`、`doctest`、`coverage` 和 `changes` 构建器。使用这些名称时会给出纠正说明。仓库中的 `tools/linkcheck` 是独立的验证工具。

源文件语言

表 21 固定版本的一致性验证

读取器	证据与限制
reStructuredText	98 个语料源文件与对照的 Docutils 树及标识符一致。对比不包括源码位置和若干内部管理属性。
CommonMark 0.31.2	全部 652 个规范示例通过测试。
MyST 0.16.1 reference	218 个读取器测试样例中有 215 个一致，10 个 Sphinx 构建样例中有 8 个一致。其余差异均已记录。

这些测试使用固定版本，不能据此推断与所有后续版本兼容。运行 `guidedog formats` 可查看当前可执行程序的读取器和渲染器验证信息。各读取器目录还包含一致性说明。

`rst_prolog` 插在开头的书目信息字段之后，`rst_epilog` 插在源码末尾，两者均不插入 Markdown。条件成立时，`only` 和 `ifconfig` 保留其中的章节。若条件章节改变了周围的章节层级，其位置可能与 Sphinx 不同。MyST 的 `eval-rst` 内不允许章节标题，会报告错误。

引用与领域

Guidedog 实现了 `toctree`、`标签`、`术语表`、`章节与插图编号`，以及 `ref`、`doc`、`numref`、`term`、`download`、`any`、`keyword`、`option`、`envvar` 和 `token` 引用。对象可以出现在目录树、侧边栏和页面大纲中。

已实现标准、Python、C、C++、JavaScript、reStructuredText 和数学领域。Odin 领域是 Guidedog 自有的扩展，描述包、声明、签名及生成的 API 页面。仅基于源码的发现机制及其限制见 [为 Odin 生成文档](#)。

参数、返回值、异常和变量字段组成结构化描述。规范名称成为引用和清单中的别名。Python 默认值和注解保留源码写法；Sphinx 可能通过 Python 的反解析器重新打印。C++ 声明发布与 Sphinx 兼容的标识符版本和符号清单。仍有嵌套深度限制。嵌套括号的解析时间为线性。

`numref` 替换一个编号占位符，并报告未编号的目标。章节按阅读顺序编号。每份文档只编号一次；第二个编号目录树会报告冲突。格式替换和标识符细节见引用测试。

内置通用索引、模块索引和搜索。搜索匹配词语前缀，不采用 Sphinx 的英语词干算法。

声明式对象类型可替代一类常用的 Python 扩展注册。`object_types`、`crossref_types` 和 `directive_aliases` 描述相应数据。自定义 Python `parse_node` 改为声明式的名称、显示和程序规则。迁移会报告无法表达的代码。参见 [声明对象类型](#)。

## 内置扩展的行为

表 22 扩展的边界

行为	状态
<code>todo</code> , <code>ifconfig</code> , <code>extlinks</code> , <code>autosectionlabel</code>	内置支持。对硬编码的 <code>extlinks</code> 链接，可建议改用角色。
<code>graphviz</code>	链接的 Graphviz 库生成静态图形，并遵守资源访问边界。
<code>intersphinx</code>	支持本地和获取的清单。Windows 当前仅支持本地文件。
<code>githubpages</code>	内置静态站点所需的发布文件。
<code>mathjax</code> 和 <code>imgmath</code>	HTML 公式使用 MathJax。PDF 将支持的 LaTeX 子集转换为 Typst。
<code>myst_parser</code>	已实现的项目级 Markdown 层。
<code>doctest</code> 指令	显示内容。项目构建器不执行其中的测试。
<code>autodoc</code>	读取 Python 源码，不导入或执行包。

基于源码的 `autodoc` 无法发现所有运行时创建的对象。编译模块、外部装饰器、计算所得的值及扩展事件钩子都有明确限制。参考行为来自 Python 3.13 和固定版本的 Sphinx 文档生成器测试。完整设置及例外见 [用 autodoc 编写 Python 文档](#)。

`autosummary`、`napoleon`、`viewcode` 和任意 Python 扩展不属于这条处理路线。配置中出现不支持的扩展时会报告；未知指令或角色会在源码位置报告。**发布之前必须检查被省略的内容**。其他主题名称会被接受，但使用 Guidedog 自有的主题实现。

## 信任与资源限制

普通构建接受项目的原始 HTML、Typst、模板和网络清单。本地读取限于源目录及明确共享的根目录。符号链接不能扩大权限。模板、静态文件、`include`、图片、字体和 API 源码均遵守同一规则。

Graphviz 图片资源相对于所属文档解析，并受路径边界限制。支持的图片数据会嵌入图形。`imagepath` 和 `fontpath` 不会扩大边界。Typst 在包含可读目录的专用根目录中运行，确保图书模板及前置内容仅访问声明的资源。

`--untrusted` 将读取范围缩至源目录，省略原始内容、不安全 URL、外部资源、项目 Typst 模板和前置内容，并给出诊断。它不获取清单或 Typst 包。请求读取资源的图表会以代码显示，并给出警告。此模式不限制原生编译器的内存或运行时间。

主机预算默认为 1 GiB，分配前先预留额度。页面工作区使用后释放；项目增大时，保留的项目图和清单仍会占用内存。读取工作线程协调任务准入。请求被拒绝时，会指出失败操作及可用的纠正选项。

`--memory=ram` 在托管预算达到上限时停止。`--memory=disk` 允许托管工作存储使用内存映射的临时文件；保留的主机存储仍计入 RAM 预算。`--disk-budget` 限制映射存储，策略还保留磁盘余量。非交互命令不会等待回答。可通过 `GUIDEDOG_AVAILABLE_MIB` 提供已知的机器可用容量。



`--max-depth`、`--max-nodes` 和 `--stack-kib` 限制文档处理。改变限制会使读取缓存和输出缓存失效。深度与栈容量必须匹配。原生 Graphviz、tree-sitter 和 Typst 的分配不属于主机预算；若需硬性上限，应使用操作系统限制。参见 [错误信息](#) 和 [内存与所有权](#)。

## 验证与实际项目

`tools/sphinxdiff` 对比 42 个固定版本的 Sphinx 测试根目录，检查页面、标识符、正文链接、清单、编号及诊断。存储的结果及其 README 解释了有意保留的差异，例如更精确的源码位置、稳定的阅读顺序编号，以及对省略的目标专用原始内容给出明确报告。

2026 年 10 月 1 日的远程验证对 CPython、Django 和 Flask 执行了全新及未变化的 HTML、PDF 构建。全部十二次构建成功，诊断和链接结果保持基线一致。另有 1,039 项 Linux 测试和 94 项针对性 AddressSanitizer 测试通过。

这些项目含有不支持的 Python 扩展结构，其诊断仍是验证证据的一部分。退出状态为零不表示所有私有指令都已复现。若发布要求零诊断，请使用 `-w`。

历史测量和源码修订记录保存在 `docs/manual/evidence/manuals.md`。最新远程评审报告位于 `build/review-remote-20261001/report.txt`。这些数据描述的是记录的工作负载，并非普遍适用的速度或内存上限。



# 内部机制

这些章节解释实现背后的推理，描述当前契约。GDS 在独立项目中保存决策与评审状态。

## Guidedoc 引擎

Guidedoc 是转换核心，借用调用方提供的存储。它读取源文、构建语义树、执行处理阶段、测量输出，再输出字节；不打开文件，也不发布产物。

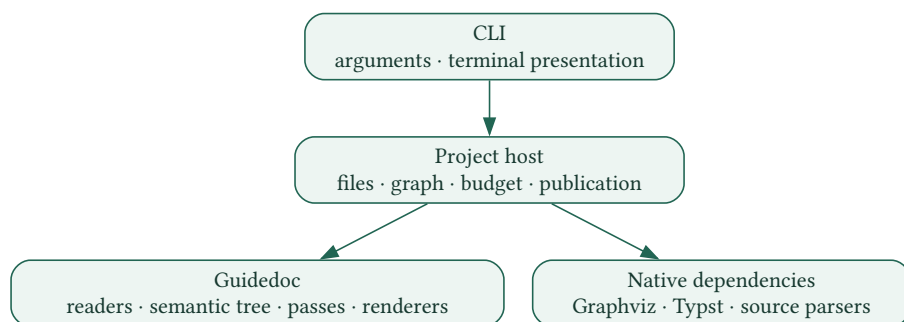


图 4 外部操作由宿主负责；文档语义由 Guidedoc 负责。

## 边界

核心转换不进行分配。调用方提供节点列、文本存储、诊断和输出容量。宿主准备这些缓冲区时可以分配。第三方编译器有自己的分配规则。这项保证针对转换边界，而非整个进程。

## 树结构

节点使用有类型的 ID。热数据列保存节点种类和子树范围；冷数据表保存各类节点需要的载荷。扁平前序表示可跳过整个子树，无需追踪堆指针链。

对节点  $i$ ，设  $e_i$  为不包含在子树中的结束索引。子树占据  $[i, e_i)$ 。叶节点有  $e_i = i + 1$ ；父节点在后代之后闭合。区间证明见 先有不变式，再谈算法。

## API

核心 API 明确直接：初始化存储、转换、检查结果、重置。注册表是包含读取器、处理阶段和渲染器描述符的值。没有运行时插件发现或全局注册副作用。少量操作便于其他 Odin 程序调用。

所有权是关键规则。工作区视图在重置或复用时无效。宿主持有的结果必须拥有所有超过会话生命周期的文本。准确签名见 `core`；它与独立 `docs/api` 项目从相同 Odin 源码生成。手册 PDF 将参考合入同一本书，提供内部链接。设计记录是 `docs/gds` 中的 GDS 0003。

## 先有不变式，再谈算法

有用的不变量说明算法运行时什么始终成立，也说明算法结束后调用方可以依赖什么。先写不变量，再写循环。

### 前序遍历中的子树区间

假设树按前序输出：节点先于子节点输出，每个子树占连续区间。设  $e_i$  为节点  $i$  及其全部后代之后的第一个索引。

$$\text{subtree}(i) = [i, e_i), \quad i < e_i \leq N. \quad (3)$$

**命题。** 将索引推进到  $e_i$ ，即可跳过整个子树。

**证明。** 连续性保证所有后代位于  $e_i$  之前。右端不包含的定义保证所有后续节点位于  $e_i$  或之后。因此推进到该索引，恰好跳过子树。渲染器使用这些假设前，验证器会先检查。

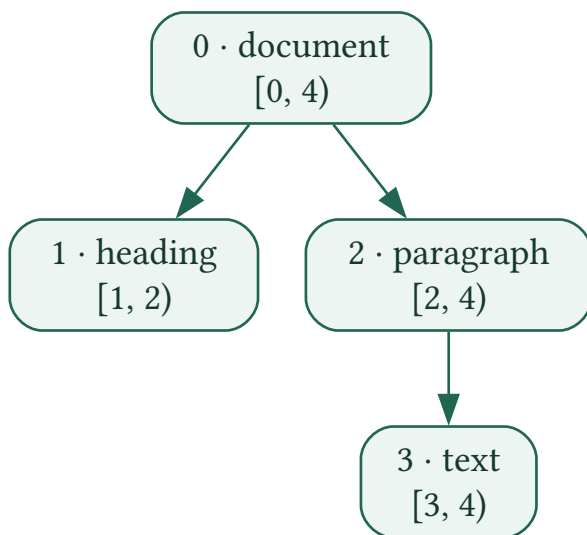


图 5 父节点的区间包含各子节点的区间。

完整扫描只访问每个节点一次，遍历成本为  $O(N)$ 。跳过子树只需更新一次索引，但载荷处理仍有自身成本。常数时间跳过，不等于常数时间渲染。

## 先测量，后输出

渲染器先测量字节数  $m$ 。设调用方的输出容量为  $c$ 。

$$m \leq c \implies \text{可以开始输出}.$$

测量与输出遍历采用相同规则。成功输出必须恰好写入  $m$  字节。失败时不返回有效的部分产物。发布由宿主负责，因此未完成的字节不能替换先前输出。

## 缓存有效性

缓存键必须覆盖所有能改变结果的数据。只有源文件还不够：包含文件、模板、配置、引用查找、资源和适配器版本都会影响输出。记录的依赖仍一致时才允许复用。即使键匹配，输出缺失也会使复用失效。

测试在核心边界安装会失败的分配器，并比较输出内容，也审查导入。只统计分配而不检查结果，可能漏掉因拒绝申请而静默丢失文档内容的问题。

## 发布完整的一代输出

读者不应看到半更新的网站。这是发布保证，与解析和渲染是不同问题。

## 不变式

设  $P$  为已发布输出树。准备阶段保持  $P$  不变；在决定提交前，下一棵树  $Q$  已完整。提交日志记录输出及其配套记录。

**准备可能失败。** 宿主放弃未完成版本。**已决定的提交可能中断。** 下次构建根据日志完成或恢复，而不根据暂存目录名称猜测。

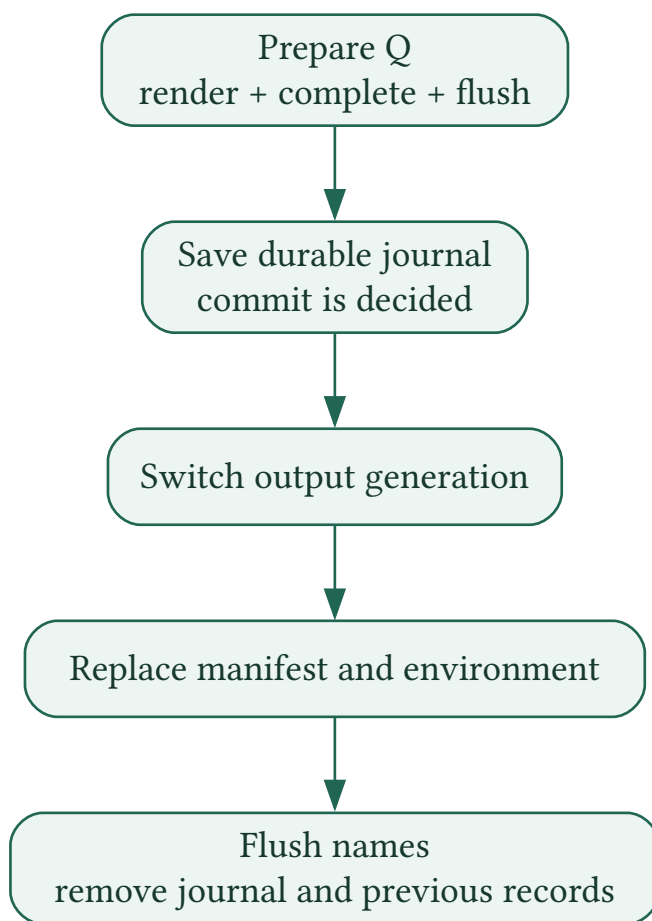


图 6 持久化日志将准备阶段与已决定的提交分开。

### 为什么缓存放在输出之外

交换目录会移动发布目录内的全部内容。若日志也在其中，就会在描述的操作中被移动，在最需恢复时失去稳定位置。Guidedog 将记录保存在发布树之外的稳定目录，因此自定义输出使用同级缓存。

### 持久性与可见性

刷写文件内容不等于刷写目录名称。删除日志前，算法处理这两者。各平台的目录交换方式不同；恢复协议据此设计。实现见 `lib/project/commit.odin` 和 `lib/project/publish.odin`。

这保护项目发布，不构成与外部托管服务之间的分布式事务。将成功构建复制到服务器，是部署工具负责的另一项操作。

### 内存与所有权

预算是一份记账契约。只有计费生命周期与其所支付的存储生命周期一致，预算才有意义。

## 计费跟随存储

分配前先预留字节；释放存储后再解除计费。替换缓冲区时，新旧缓冲区可能同时存在，重叠期间应计入完整新分配。只计增量会低估峰值。

替换时的峰值 = 其他存活存储 + 旧缓冲区 + 新缓冲区。

会话拥有的输入由会话回收；调用方拥有的输入仍由调用方负责。`read_owned` 和 `read_kept` 明确区分这两种情况。读取长度减少，并不会缩小已分配空间。

## 已知为零，不等于未知

内存探测可能确认没有余量，也可能无法查明余量，这是不同状态。策略用单独的已知标记区分，而非都用零表示。Linux 上，可读取的 cgroup 限制沿可见祖先层级生效。仅看容器宿主机的空闲内存可能误导。

## 报告的寿命长于工作区

会话结束后显示的报告必须拥有标题、源文和修正文字，不能借用已释放缓冲区。内部源范围使用字节偏移；用户列号按 Unicode 标量计数；终端标记另行考虑显示宽度。

## 仍然存在的边界

Guidedog 计量自己管理的工作存储。Typst 编译器和原生依赖自行管理分配。大型 PDF 的内存可能远超宿主预算。远程 CPython 验证中，包含 Typst 的峰值 RSS 约为 14.5 GiB。这是该工作负载的证据，不是通用上限。达到托管预算时的选择见 错误信息。



---

# 关于本手册

---

本手册介绍已经实现的工具，以及有意保留的限制。独立的 GDS 项目记录提案、决策和评审历史。改善排版并不会改变提案的状态。

本手册按实际使用过程组织：通过教程入门，通过专题理解原理，通过操作指南完成任务，再查阅参考资料核对细节。内部实现部分解释不变量和证明。

行文以小例子、明确的假设和直接的陈述为主。目标是让读者能够检验解释是否成立。页面采用克制的配色、醒目的标题、适合阅读的衬线字体，以及易于使用的导航。它并不模仿任何作者的措辞或版式。

本项目通过 `guidedog quickstart` 创建，并由 Guidedog 构建。正文使用 RST，图表使用 Graphviz，公式使用 LaTeX 数学记法。只有 PDF 排版模板仍使用 Typst。

## 构建验证记录

2026-10-01 的远程验证对 CPython、Django 和 Flask 分别进行了全新及增量 HTML、PDF 构建。十二条命令全部成功，诊断和链接检查结果与既有基线一致。Linux 测试通过 1,039 项，定向 sanitizer 测试通过 94 项。这些结果只证明所测试版本的行为，并不表示完全兼容 Sphinx。已知限制见 [与 Sphinx 的兼容性](#)。

文档维护命令见 [维护本项目的文档](#)。





---

# Guidedog

---

[English](#) · [简体中文](#) · [日本語](#) · [한국어](#) · [हिन्दी](#) · [Español](#) · [Deutsch](#)

写一份源码，讲清楚道理，发布到网页与纸上。

Guidedog 将 reStructuredText 和 Markdown 构建为 HTML 文档网站和 PDF 图书。它使用 Odin 编写。转换引擎名为 **Guidedoc**，PDF 由 Typst 排版。无需 Python 环境或 LaTeX 安装。

Sphinx 启发了它的写作流程。Guidedog 是独立系统：配置使用声明式 TOML，API 提取采用静态分析，绝不执行 Python 配置或扩展。完全兼容 Sphinx 不是目标。

## 创建项目

```
guidedog quickstart docs
guidedog build html docs
guidedog build pdf docs
guidedog serve docs
```

项目包含 `conf.toml`、`index.rst`、`_templates/layout.html`、`_templates/book.typ` 和 `_static/`。模板是普通文件，可直接编辑。HTML 提供导航、搜索、代码高亮、图表和公式。PDF 提供章节、交叉引用、插图、索引及按语言优化的排版。

默认采用浅色主题，也可选择深色或跟随系统。手册首页和仓库概览会按浏览器语言选择版本。已保存的选择优先；直接进入某个语言版本或章节时，会保留该链接的选择。

```
guidedog convert notes.md --to pdf --output notes.pdf
guidedog clean docs
guidedog migrate conf.py
```

## 安装

每个版本都提供 Linux (x86-64 和 ARM64)、macOS (Apple 芯片和 Intel) 和 Windows (x86-64) 的压缩包。每个压缩包都带有 Typst、Graphviz 和 tree-sitter，构建网站和图书无需再安装其他软件。Linux 需要 glibc 2.35 或更新版本以及 libcurl；macOS 需要 12 或更新版本。下载与系统对应的压缩包和 SHA256SUMS，校验后安装：

```
sha256sum -c --ignore-missing SHA256SUMS # on macOS: shasum -a 256 -c --
ignore-missing SHA256SUMS
tar -xzf guidedog-linux-amd64.tar.gz
sudo install guidedog-linux-amd64/guidedog /usr/local/bin/
```

在 macOS 上, 通过官方 [Guidedog Homebrew tap](#) 安装。Homebrew 会自动选择 Apple 芯片或 Intel 芯片对应的安装包。该 tap 也支持 Linux:

```
brew tap insanai/guidedog
brew install insanai/guidedog/guidedog
guidedog --version
```

在 Windows 上, 解压压缩包并把其文件夹加入 PATH; Graphviz 的 DLL 和 config8 要与 guidedog.exe 放在一起。这些程序没有代码签名: 在 macOS 上, 用 `xattr -d com.apple.quarantine guidedog` 解除浏览器下载文件的隔离属性。各平台的限制见版本说明。

## 从源码构建

安装 Odin dev-2026-10 (`tools/install-toolchain.sh` `odin` DIR 会下载并校验它)、C 编译器, Linux 上还需要 `libcurl`、`expat` 和 `zlib` 的开发包。先构建原生库, 再构建 Guidedog:

```
native/tree-sitter/build.sh
native/graphviz/build.sh
tools/release/build.sh
```

之后 `build/guidedog` 使用已安装的 Typst 0.15.1 生成 PDF。若要像发布版本那样内置 Typst 编译器, 安装 Rust 和 Cargo 后运行:

```
native/typst_bridge/build.sh
tools/release/build.sh --release --typst-bridge
```

在 Windows 上, 请使用 Git Bash, 并安装 Visual Studio 的 C++ 工具和 LLVM。Graphviz 没有 Windows 静态构建, 因此运行 `native/graphviz/windows.sh` 代替 `build.sh`: 它会安装官方的 Graphviz 16.1.0 DLL, `tools/release/build.sh` 再把它们复制到 `build/guidedog.exe` 旁边。 `tools/release/suites.sh` 运行测试套件, [RELEASING.md](#) 介绍发布流程。

## 用 Guidedog 翻译

```
mkdir -p docs/manual/_editions
build/guidedog build -b gettext docs/manual docs/manual/locales/templates -W
build/guidedog intl update -p docs/manual/locales/templates \
 -l zh_CN -l ja -l ko -l hi -l es -l de docs/manual
build/guidedog intl stat -l zh_CN -l ja -l ko -l hi -l es -l de docs/manual
build/guidedog build -b html docs/manual docs/manual/_editions/de \
 -D language=de -D 'html_extra_path=[]' -W
```

在 PO 文件中填写并审阅 `msgstr`。Guidedog 负责提取 POT、合并目录、报告进度和构建各语言版本, 不会自动编写翻译。命令、路径、代码、API 名称和引用目标保持不变。只翻译手册和仓库概览, API 与 GDS 保持英文。汇总所有网站版本和 PDF 的方法见[手册构建说明](#)。

GitHub 上的读者用上方链接选择 README 语言。自动语言选择由生成的文档网站提供。

## 阅读与嵌入

- **手册**：教程、专题、操作指南、参考和内部原理。
- **API 参考**：从 Odin 源码生成，并作为附录收入手册 PDF。
- **库指南**：调用方拥有的工作区、内存和转换示例。
- **模板**：用 Jinja 自定义 HTML，用 Typst 自定义图书。
- **GDS**：编号的设计记录和实现决策。
- **Beta 评审**：修复、验证、支持范围和发布前的剩余事项。

Guidedoc 使用调用方提供的存储，返回结构化诊断。宿主管理文件、预算、缓存、发布和原生编译器。`--untrusted` 只限制能力，不是进程沙箱，也不限制全部原生资源消耗。Beta 面向 macOS 和 Linux 上的可信文档项目。

## 致谢与许可

由 **Vikrant Rathore** 创建，**Ronak Rathore** 和 **Kanak Rathore** 协助开发。欢迎通过 issue 和 pull request 参与。文档署名使用姓名或账号名，不公开个人邮箱；Git 身份设置单独管理。

感谢 Sphinx 和 Georg Brandl、Docutils 和 David Goodger、Typst、Tree-sitter 和 Max Brunsfeld、Graphviz，以及 Ginger Bill 和 Odin 社区。

Guidedog 以 **GNU Affero 通用公共许可证 3.0 或更高版本**（AGPL-3.0-or-later）发布。如需其他许可条款，请联系作者。**NOTICE** 说明了联系方式，并列出保留各自许可证的第三方组件，例如捆绑字体。

附加许可允许将 Guidedog 与 Graphviz 链接并一同分发，详见 **NOTICE**。



---

# API Reference

---

- `autodoc` – Package autodoc generates the reStructuredText that Sphinx’s sphinx.ext.autodoc generates (Sphinx 9.1), from Python source analysed by lib/pyscan instead of imported modules: the same directives (“..
- `cli` – Package cli is the guidedog command: run parses a command line, does what it asks, and returns the exit status.
- `core` – Generated by tools/casefold from tools/data/CaseFolding-17.0.0.txt ; do not edit.
- `defaults` – Package defaults assembles the built-in readers and renderers into an explicit, immutable registry.
- `fetch`
- `gds` – Package gds is the GDS process of docs/gds (GDS 0001) as a library: it loads the discussion records of a directory into a Catalog, plans changes to them (a new draft, a promotion, a state or label change, the regenerated index) as a Plan, applies a plan as one recoverable transaction under a lock.
- `gettext` – Package gettext reads, writes, merges, and looks up GNU gettext message catalogs: PO and POT text files, and compiled MO files.
- `graphdog` – Package graphdog turns Graphviz DOT source into SVG (or PNG, or laid-out DOT) by calling the Graphviz C library in process.
- `highlight` – Generated by tools/pygments\_styles from pygments-2.19.2’s style modules and sphinx-9.1.0’s pygments\_styles.py; do not edit.
- `highlight/guidedoc` – Package highlight\_guidedoc connects the highlight library to Guidedoc’s renderers.
- `host` – Package host composes the allocation-free core with memory, files, and retries.
- `i18n` – Package i18n translates Guidedoc documents the way Sphinx does with gettext: it extracts every translatable message of a document into a POT template (Sphinx’s gettext builder), and it replaces each message with its translation from a catalog (Sphinx’s Locale transform), re-parsing the translation as inline markup of the document’s own language.
- `inventory`
- `jinja`
- `odindoc` – Package odindoc reads Odin packages by static analysis, with Odin’s own parser (core:odin/parser), and writes the reStructuredText of the Guidedoc odin domain that documents them, as sphinx.ext.autodoc writes the Python domain’s: packages, procedures, procedure groups, types and

their fields or enumerators, constants, variables, and foreign imports, each with its doc comment turned into reST.

- `project` – Package `project` builds a documentation project the way Sphinx does: a source directory with `conf.toml`, builders writing `_build/<builder>`, and incremental builds.
- `pyscan` – Package `pyscan` analyses Python source statically, through tree-sitter: modules and their docstrings, `__all__`, and imports; classes with their bases, decorators, and members in source order; functions with their parameters as written; and the attribute documentation Sphinx’s `ModuleAnalyzer` finds (`#:` comments and docstrings after assignments).
- `readers/commonmark` – Package `commonmark` reads Markdown into a Guidedoc document without calling an allocator, as two readers with distinct ids: “commonmark” (`reader`, `read`, `read_inline`) is pure CommonMark 0.31.2 and takes no settings; “myst” (`myst_reader`, `read_myst`, `read_myst_inline`) is the MyST Markdown that Sphinx projects write (`myst-parser` 0.16.1): directives, roles, targets, comments, front matter, footnotes, GFM tables, and the optional syntax that `Settings.enable_extensions` turns on.
- `readers/rst` – Package `rst` reads reStructuredText as the Docutils markup specification defines it, implemented independently.
- `readers/rst/inline` – Package `rst_inline` parses reStructuredText inline markup and resolves references.
- `readers/sphinx` – Package `sphinx` reads reStructuredText with the Sphinx layer: Sphinx’s directives, roles, and domains (GDS “Guidedog: everything Sphinx does, the Odin way”), implemented independently on the base reader’s extension interface.
- `render/html` – Package `html` renders a Guidedoc document as HTML without calling an allocator.
- `render/text` – Package `text_render` renders a Guidedoc document as plain text without calling an allocator, following the conventions of Sphinx’s text builder: paragraphs wrapped at 70 columns, titles underlined with `=` `~` and the other section characters, `“* “` and `“1.`
- `render/typst` – Package `typst_render` encodes a validated Guidedoc document as one Typst source file.
- `tree-sitter` – Package `tree-sitter` is a thin binding of the tree-sitter C library and its Python grammar: parsers, trees, nodes, and cursors, and nothing more.
- `typst` – Package `typst` adapts the Typst toolchain for the host.

## autodoc

Package `autodoc` generates the reStructuredText that Sphinx’s `sphinx.ext.autodoc` generates (Sphinx 9.1), from Python source analysed by `lib/pyscan` instead of imported modules: the same directives (`“.. py:class:: Name(signature)”` with `:module;`, `:canonical;`, `“Bases: ...”`), docstrings prepared as `prepare_docstring` prepares them, and members chosen, filtered, and ordered as `autodoc`’s options say. What static analysis cannot know (compiled modules, objects made at run time, members of classes outside the source roots) is reported as a note naming the object, never guessed silently.

It is a host library: it reads files through pyscan and allocates with the context allocator, freeing none of it, so a host runs it in an arena, the same as pyscan's: a Session and every Result live until the arena is freed.

## Types

**autodoc.Config** :: struct

Config holds the conf.toml settings autodoc reads, with Sphinx's names and defaults (see `default_config`).

**autoclass\_content**: string

“class”, “init”, “both”.

**autodoc\_class\_signature**: string

“mixed”, “separated”.

**autodoc\_default\_options**: []Default\_Option

**autodoc\_docstring\_signature**: bool

**autodoc\_inherit\_docstrings**: bool

**autodoc\_member\_order**: string

“alphabetical”, “bysource”, “groupwise”.

**autodoc\_mock\_imports**: []string

**autodoc\_preserve\_defaults**: bool

**autodoc\_typehints**: string

“signature”, “description”, “none”, “both”.

**autodoc\_typehints\_description\_target**: string

“all”, “documented”, “documented\_params”.

**autodoc\_typehints\_format**: string

“short”, “fully-qualified”.

**autodoc\_type\_aliases**: []Type\_Alias

---

**autodoc\_use\_type\_comments:** bool

---

**python\_display\_short\_literal\_types:** bool

---

**strip\_signature\_backslash:** bool

---

**tab\_width:** int

---

**typehint\_fields:** bool

---

typehint\_fields writes the fields of autodoc\_typehints = “description” into the text (true by default); Sphinx adds them to the parsed field lists instead, so comparisons with its generated text turn it off.

---

autodoc.**Default\_Option** :: struct

---

Default\_Option is one autodoc\_default\_options entry: a value, or true for a flag.

---

**name:** string

---

**value:** string

---

**flag:** bool

---

the setting is true: the option is given without a value.

---

autodoc.**Message** :: struct

---

Message is a problem to report where the directive stands, Elm-style.

---

**severity:** Severity

---

**code:** string

---

**title:** string

---

**text:** string

---

**hint:** string

---



```
autodoc.Option :: struct
```

Option is a directive option as written; value is “” for a flag.

```
name: string
```

```
value: string
```

```
autodoc.Request :: struct
```

Request is one autodoc directive: its name (autoclass, automodule, ...), argument, options, content, and the reader’s context (the current py:module and py:class).

```
directive: string
```

```
argument: string
```

```
options: []Option
```

```
content: []string
```

```
module: string
```

```
class: string
```

```
autodoc.Result :: struct
```

Result is the generated reST (lines without line breaks; empty when there is nothing to add) and the messages about it.

```
lines: [dynamic]string
```

```
messages: [dynamic]Message
```

```
autodoc.Session :: struct
```

Session is autodoc’s state across the directives of one build: the analysed project, and the modules whose analyser annotations autodoc has merged into their classes (as `_ensure_annotations_from_type_comments` does the first time it documents a data or attribute from them), which changes the members found afterwards.

```
project: ^py.Project
```

```
merged: map[string]bool
```

```
autodoc.Severity :: enum u8
```

```
Note
```

```
Warning
```

```
Error
```

```
autodoc.Type_Alias :: struct
```

```
name: string
```

```
target: string
```

### Procedures

```
autodoc.default_config :: proc() -> Config
```

default\_config is Sphinx 9.1's defaults for the settings autodoc reads.

```
autodoc.document :: proc(session: ^Session, config: ^Config,
request: Request) -> Result
```

document generates the reST of one autodoc directive, as AutodocDirective.run does before parsing it: the lines, and messages about them (an unknown directive or a bad option is an Error and no lines; what analysis cannot know is a note naming the object; never a panic). Modules are analysed through the session's project as needed; the result lives in the context arena.

```
autodoc.session_init :: proc(s: ^Session, project:
^py.Project)
```

session\_init starts a session over an analysed project, which it borrows for the session's life; its map lives in the context arena.

## cli

Package cli is the guidedog command: run parses a command line, does what it asks, and returns the exit status. It is the command's whole surface; everything

else here is private. It allocates with `context allocator` and never frees, so the command runs it in an arena (`cmd/guidedog`), which frees everything at once.

### Procedures

```
cli.run :: proc(args: []string) -> int
```

`run` runs the command line args (without the program name) and returns the exit status: 0 on success, and on failure the status the problem names, after reporting it on standard error. It allocates with `context allocator`, an arena the caller frees, through a `host.Guard`: when that allocator refuses memory, the command changes no record and ends with `host.memory` (exit 3), reported without allocating.

```
cli.version_text :: proc(buffer: []u8) -> string
```

`version_text` writes into buffer what `guidedog --version` prints before the Typst compiler it finds, the versions built into the program, and returns it. It allocates nothing, so the crash report is prepared with it at start (`cmd/guidedog`).

### Constants

```
cli.FULL_VERSION :: VERSION + VERSION_SUFFIX
```

`FULL_VERSION` is what the command prints as its version: `VERSION` and `VERSION_SUFFIX`.

```
cli.LICENSE_NOTICE
```

`LICENSE_NOTICE` is what `guidedog --licenses` prints before the notices of the libraries linked in.

```
cli.VERSION :: "0.2.0"
```

`VERSION` is the Guidedog release this program belongs to, the one version ``guidedog --version``, `guidedog --help`, and the crash report name. The file `VERSION` at the repository root holds the same text for release tooling, and `version_test.odin` fails when the two differ. The Guidedoc engine's own version (`core.VERSION`) and document schema are reported beside it; they are separate because programs embed the engine without this command.

```
cli.VERSION_SUFFIX :: #config(GUIDEDOG_VERSION_SUFFIX, "")
```

`VERSION_SUFFIX` is build metadata appended to `VERSION`, for development builds such as `-define:GUIDEDOG_VERSION_SUFFIX=+3fe1d8a` (Semantic Versioning's "+" form). A release build has none.

## core

Generated by tools/casfold from tools/data/CaseFolding-17.0.0.txt ; do not edit.

### *Types*

```
core.Admonition_Kind :: enum i32
```

**Generic**

**Attention**

**Caution**

**Danger**

**Error**

**Hint**

**Important**

**Note**

**Tip**

**Warning**

**Seealso**

**Todo**

```
core.Align :: enum u8
```

**None**

**Left**

**Center**

**Right**

```
core.Anchor_Extra :: struct
```

Anchor\_Extra is the id of an element a renderer generates for a node:

- an entry of a table of contents: owner is the Contents, item the Section;
- a footnote of target-notes: owner is the directive, item the first target (or anonymous reference) with the URI it lists, number its footnote number;
- the footnote reference target-notes adds after a reference: owner is the reference, item the directive, note the item of its footnote, number the same.

```
owner: Node_Id
```

```
item: Node_Id
```

```
note: Node_Id
```

```
span: u32
```

```
number: i32
```

```
core.Anchor_Ids :: struct
```

```
span: u32
```

```
core.Anchors :: struct
```

Anchors gives nodes stable, unique identifiers: HTML ids and Typst labels. Every renderer and the project host build the same table, so internal links agree across targets and the labels a project records are the ids its pages publish. The ids are the ones Docutils gives a document read as Sphinx reads it (auto\_id\_prefix “id”), so a project that moves from Sphinx keeps its links and bookmarks:

- An explicit Id attribute is used verbatim (terms, equations, domain objects).
- Every element Docutils names gets one id when Docutils’ parser or transforms name it, in the same order (anchors\_order.odin): the id made from its name by make\_id, or, when the name gives none or only an id already used, the next “idN”. Elements without a name (footnote references, anonymous targets) always take the next “idN”.
- Targets and names Docutils does not write (external and indirect targets, names of embedded URIs) still take their id, so later ids are numbered as Docutils numbers them, but no node carries it.
- A label (an internal block Target) gives its id to the element it names, after the element’s own ids.

- Elements a renderer generates have ids too (anchor\_extras): the entries of a table of contents and the footnotes of target-notes with their references.

A node may have several ids: anchor\_of returns the primary one, and anchor\_ids iterates all of them for renderers that emit the others as aliases.

**index:** []u32

per node: its first span, or NONE.

**spans:** []Text

ranges of bytes.

**next:** []u32

per span: the node's next span, or NONE.

**count:** int

**bytes:** Buffer

**prefix:** string

**prefix\_length:** int

the bytes the prefix takes before an id.

**names:** Symbol\_Table

every id: the node that carries it, or NONE.

**counter:** int

**labels:** []u32

per node: the first id labels give it, until they follow its own.

**own:** []u8

room for the longest name a node carries by itself.

**extras:** []Anchor\_Extra

sorted by owner, then in the order the ids were given.

**extra\_count:** int

```
core.Arg_Kind :: enum u8
```

```
None
```

```
Int
```

```
Span
```

```
Static
```

```
core.Attribute :: struct
```

```
key: Attribute_Key
```

```
name: Text
```

the key text when key is Custom.

```
value: Text
```

```
core.Attribute_Key :: enum u8
```

```
Custom
```

```
Class
```

```
Id
```

```
Name
```

```
Language
```

```
Alt
```

```
Width
```

```
Height
```

```
Scale
```

**Align****Format****Title****Target****Start****Widths****Source****core.Attribute\_Range** :: struct**first**: u32**count**: u32**core.Attributes** :: struct**next**: u32**end**: u32**core.Buffer** :: struct

Buffer is a bounded byte accumulator over caller-provided bytes. The first write that does not fit sets overflow, and every later write is dropped, so the bytes are always a prefix of what was written; a caller that sees overflow must report it. wanted counts every byte asked for, stored or not: writing into a zero Buffer measures, and scratch\_buffer then takes exactly that much.

**bytes**: []u8**used**: int**wanted**: int**overflow**: bool



```
core.Builder :: struct
```

Builder appends nodes in preorder. Open nodes form a bounded stack; closing a node patches its subtree end. The first failure poisons the builder: later calls do nothing, so an ignored capacity failure cannot produce a truncated document. The engine gives a reader or pass a builder over one bank for one stage: text is copied into the bank's pool, ids and Text values index that bank, and nothing a builder returns outlives the stage unless the stage succeeds. Adapters change nodes through these procedures; b.snap is visible so a reader can decode text straight into the pool after text\_used, adding what it wrote to text\_used before text\_close.

```
snap: ^Snapshot
```

```
open: Stack(Node_Id)
```

```
reports: ^Reports
```

```
limits: Limits
```

```
status: Status
```

```
need: Capacity_Need
```

```
work: u64
```

```
cancel: Cancel_Probe
```

```
core.Cancel_Probe :: struct
```

```
procedure: proc(user: rawptr) -> bool
```

```
user: rawptr
```

```
core.Capacity_Need :: struct
```

```
region: Region
```

```
minimum: u64
```

```
exact: bool
```

```
core.Cell_Info :: struct
```

```
row_span: u16
```

```
col_span: u16
```

```
align: Align
```

```
core.Children :: struct
```

```
next: u32
```

```
end: u32
```

```
core.Conformance :: enum u8
```

```
Planned
```

```
Partial
```

```
Conformant
```

```
core.Content :: enum u8
```

```
Nothing
```

```
Blocks
```

```
Inlines
```

```
Structure
```

children are restricted to specific kinds; see `child_allowed`.

```
core.Content_Policy :: enum u8
```

```
Refuse
```

an error that names the construct and the policy.

```
Omit
```

a warning; the content is left out.

**Allow**

trusted content is emitted.

```
core.Diagnostic :: struct
```

```
using msg: Message
```

```
span: Source_Span
```

```
args: [2]Diagnostic_Arg
```

```
core.Diagnostic_Arg :: struct
```

Diagnostic\_Arg is an integer, a quoted source span, or static text.

```
kind: Arg_Kind
```

```
int: i64
```

```
span: Source_Span
```

```
static: string
```

```
core.Document :: struct
```

Document is a checked handle to one build of a snapshot bank of a workspace. It borrows the workspace and is valid until the next conversion, reset, or rebuild of its bank, whichever comes first; document\_valid tells.

```
ws: ^Workspace
```

```
generation: u32
```

```
revision: u32
```

the bank's revision when the document was built.

```
bank: u8
```

```
core.Document_Reading :: enum i32
```

Document\_Reading is a Document's `value`: who read it, which decides the ids the transforms after reading give (anchors\_order.odin). The Sphinx layer marks what it reads, as Sphinx adds transforms of its own to Docutils'.

**Docutils**

**Sphinx**

```
core.Emit_Mode :: enum u8
```

A renderer has one encoding procedure. The engine runs it in Measure mode, with reports enabled, to check support and count bytes; then in Write mode to store them.

**Measure**

**Write**

```
core.Emitter :: struct
```

Emitter is where a renderer writes. It never fails: Measure mode counts bytes, and Write mode stores them in the caller's output, which the engine sized by measuring.

**mode:** Emit\_Mode

**out:** []u8

**written:** int

**source\_map:** []Map\_Entry

**mapped:** int

```
core.Highlighter :: struct
```

Highlighter writes highlighted HTML for a code block. It belongs to the caller's trust boundary and may allocate, but it must write the same bytes for the same input every time, because the renderer measures before it writes. It returns false to let the renderer write plain escaped code instead.

```
procedure: proc(user: rawptr, language, code: string, e: ^Emitter) -> bool
```

```
user: rawptr
```

```
core.INVALID_NODE :: Node_Id(max(u32))
```

```
core.Label :: enum u8
```

Label names a word or phrase a renderer writes by itself, not taken from the document, or that a reader writes into it (the group names of Sphinx’s info fields). A host translates them (Sphinx ships them in sphinx.po) and passes them in Render\_Options.labels, and to a reader in its own options. Labels holding “%s” take a value there (a version number).

**Attention**

**Caution**

**Danger**

**Error**

**Hint**

**Important**

**Note**

**Tip**

**Warning**

**See\_Also**

**Todo**

**Contents**

**Version\_Added**

---

**Version\_Changed**

---

**Version\_Deprecated**

---

**Version\_Removed**

---

**Section\_Author**

---

**Module\_Author**

---

**Code\_Author**

---

**Link\_Heading**

---

**Link\_Definition**

---

**Link\_Image**

---

**Link\_Table**

---

**Link\_Code**

---

**Link\_Equation**

---

**Link\_Term**

---

**Parameters**

---

**Variables**

---

**Raises**

---

**Returns**

---

**Return\_Type**

---

**Return\_Values**

---

**Template\_Parameters****Throws****Arguments****Availability**

```
core.Labels :: [Label]string
```

```
core.Limits :: struct
```

```
max_depth: int
```

nested blocks and inline markup.

```
max_nodes: int
```

```
max_work: u64
```

reader and resolution steps.

```
max_include_depth: int
```

```
max_included_bytes: int
```

```
max_expansion: int
```

substitution nesting.

```
stack_bytes: int
```

stack\_bytes bounds the machine stack a recursive reader or renderer uses; o means DEFAULT\_STACK\_BYTES. The conversion thread needs this much more stack (stack.odin).

```
core.Line_View :: struct
```

Line\_View is one line of a source as readers hand it to each other: its text without the line break, and the source offset of text[o]. The reST and Markdown readers share it, so nested content passes between them as it is, without converting or casting.

```
text: string
```

**offset:** u32

**core.Link\_Info** :: struct

**destination:** Text

**title:** Text

**name:** Text

normalized reference name, when named.

**resolved:** Node\_Id

meaningful only when kind is Internal.

**kind:** Link\_Kind

**core.Link\_Kind** :: enum u8

**External**

destination is a URL or path.

**Email**

destination is an address; HTML adds mailto:.

**Internal**

resolved is a node in this document.

**Unresolved**

name awaits resolution; an error, or a warning and problematic text

**Project**

under Policy.keep\_documents\_with\_errors.

destination is a project anchor key in another document (books).

**core.List\_Info** :: struct

**style:** List\_Style



**delimiter:** u8

‘, ‘), or ‘( for “(1)”; bullet character for bullets.

**start:** i32

**core.List\_Style** :: *enum* u8

**Bullet**

**Arabic**

**Lower\_Alpha**

**Upper\_Alpha**

**Lower\_Roman**

**Upper\_Roman**

**core.Map\_Entry** :: *struct*

Map\_Entry records that generated output from offset on came from span.

**offset:** u32

**span:** Source\_Span

**core.Math\_Style** :: *enum* u8

Math\_Style is how HTML writes math for a script (MathJax, KaTeX) to typeset.

**Plain**

class “math”, the text in  $\backslash( \backslash)$  or  $\backslash[ \backslash]$ , as Docutils writes it.

**Sphinx**

as Sphinx’s mathjax extension: its classes, and displayed math in parts

```
core.Media_Kind :: enum u8
```

Media\_Kind is what an image node shows. As in Docutils, an image directive naming a video file plays it; Guidedoc treats audio files the same way. Renderers decide from the path.

**Image**

**Video**

**Audio**

```
core.Message :: struct
```

Message is a static description of one problem. Packages declare them as constants, so a report never formats or copies text.

**code:** string

**severity:** Severity

**status:** Status

**title:** string

**message:** string

“{0}” and “{1}” name the arguments.

**hint:** string

```
core.Node :: struct
```

**kind:** Node\_Kind

**flags:** Node\_Flags

**subtree\_end:** u32

**value:** i32

**text:** Text

**payload:** u32

**span:** Source\_Span

**attributes:** Attribute\_Range

**core.Node\_Flag** :: *enum* u8

### **Tight**

List: items render without paragraph spacing.

### **Anonymous**

Reference or Target: anonymous hyperlink.

### **Auto**

Footnote or reference: auto-numbered.

### **Symbol**

Footnote or reference: auto-symbol.

### **Local**

Contents: only the enclosing section.

### **Header**

Row or Cell: belongs to a header.

### **Referenced**

Target or Footnote: something refers to it.

### **Generated**

Created by resolution rather than markup.

**core.Node\_Flags** :: *bit\_set*[Node\_Flag; u8]

**core.Node\_Id** :: *distinct* u32

```
core.Node_Kind :: enum u8
```

**Document**

**Title**

Block structure.

**Subtitle**

**Section**

**Heading**

**Paragraph**

**Block\_Quote**

**Attribution**

**List**

**List\_Item**

**Definition\_List**

**Definition\_Item**

**Term**

**Classifier**

**Definition**

**Field\_List**

**Field**

**Field\_Name**

**Field\_Body****Option\_List****Option\_Item****Option\_Group****Option****Option\_Argument****Option\_Description****Line\_Block****Line****Code\_Block****Preformatted****Doctest\_Block****Math\_Block****Raw\_Block****Thematic\_Break****Table****Table\_Head****Table\_Body****Row**

**Cell****Figure****Caption****Legend****Admonition****Topic****Sidebar****Rubric****Container****Compound****Footnote****Citation****Target****Comment****Substitution\_Definition****Contents****Directive****Toctree**

Sphinx layer (GDS “Guidedog: everything Sphinx does, the Odin way”).

**Toctree\_Entry**

**Only****Glossary****Version\_Modified****Desc****Desc\_Signature****Desc\_Content****Hlist****Production\_List****Production****Index****Text**

Inline content.

**Emphasis****Strong****Code****Link****Reference****Image****Footnote\_Reference****Citation\_Reference**

**Substitution\_Reference****Inline\_Target****Math****Raw\_Inline****Soft\_Break****Hard\_Break****Subscript****Superscript****Title\_Reference****Abbreviation****Role****Xref**

Sphinx layer inlines.

**Desc\_Name****Desc\_Addname****Desc\_Annotation****Desc\_Parameterlist****Desc\_Parameter****Desc\_Optional****Desc\_Returns**



```
core.Pass :: struct
```

A pass reads an immutable document and may rebuild it into the destination builder.

```
name: string
```

```
procedure: proc(user: rawptr, doc: Document, b: ^Builder)
-> Pass_Outcome
```

```
user: rawptr
```

```
core.Pass_Outcome :: enum u8
```

```
Unchanged
```

```
Rebuilt
```

```
Failed
```

```
core.Policy :: struct
```

```
raw: Content_Policy
```

raw HTML, raw Typst, raw LaTeX.

```
unsafe_urls: Content_Policy
```

javascript:, vbscript:, and non-image data: URLs.

```
math: Content_Policy
```

math the target cannot translate.

```
includes: bool
```

whether include directives may request resources.

```
strict: bool
```

warnings fail the conversion.

```
keep_documents_with_errors: bool
```

keep\_documents\_with\_errors keeps a document whose content has errors, as Docutils does below its halt level: an unknown directive or role,

an unresolved reference, or a bad option is still reported as an error, but the construct is left out (or rendered as problematic text) and the status is Ok. Renderers report such problems as warnings. Errors that leave no usable tree still fail: capacity, limits, cancellation, invalid storage, and validation. See `content_error`.

```
core.ROOT :: Node_Id(0)
```

```
core.Read_Context :: struct
```

```
ws: ^Workspace
```

```
b: ^Builder
```

```
source: Source_Id
```

```
options: Read_Options
```

```
policy: Policy
```

```
limits: Limits
```

```
resources: Resource_Provider
```

```
reports: ^Reports
```

```
scratch: ^Scratch
```

```
included: int
```

bytes of included sources, charged against `max_included_bytes`.

```
core.Read_Extension :: struct
```

`Read_Extension` carries a reader's own settings (`rst.Settings`, `sphinx.Config`, `commonmark.Settings`, or a third-party reader's type): a pointer the caller owns and the type it points to. `read_extension` makes one, and a reader takes it back with `reader_settings`, which refuses settings of another type, so settings meant for one reader are never read as another's. The zero value means the reader's defaults.

```
data: rawptr
```

**id:** typeid

```
core.Read_Inline_Proc :: #type proc(ctx: ^Read_Context, span:
Source_Span) -> Status
```

`Read_Inline_Proc` parses a span of `ctx.source` as inline content of the builder's open node, with the reader's settings (`ctx.options.extension`), as a translated message is parsed. It neither resolves references nor reports unresolved ones: the caller owns the surrounding document. `span.source` must be `ctx.source`.

```
core.Read_Options :: struct
```

**promote\_title:** bool

RST: a lone top-level section becomes the title.

**tab\_width:** int

RST tab stops; zero means 8.

**today:** string

RST date substitutions, e.g. 2026-09-29.

**extension:** Read\_Extension

the reader's own settings; zero: its defaults.

```
core.Read_Proc :: #type proc(ctx: ^Read_Context) -> Status
```

`Read_Proc` reads `ctx.source` into `ctx.b` and returns its status; see `Reader`.

```
core.Read_Result :: struct
```

**using** result: Result

**document:** Document

```
core.Reader :: struct
```

**id:** string

**name:** string

**version:** string

<b>conformance:</b> Conformance
---------------------------------

<b>read:</b> Read_Proc
------------------------

<b>read_inline:</b> Read_Inline_Proc
--------------------------------------

optional; see Read\_Inline\_Proc.

<code>core.Region :: enum u8</code>
-------------------------------------

<b>None</b>
-------------

<b>Nodes</b>
--------------

<b>Text</b>
-------------

<b>Attributes</b>
-------------------

<b>Lists</b>
--------------

<b>Links</b>
--------------

<b>Cells</b>
--------------

<b>Sources</b>
----------------

<b>Scratch</b>
----------------

<b>Reports</b>
----------------

<b>Output</b>
---------------

<b>Source_Map</b>
-------------------

<b>Second_Bank</b>
--------------------

the workspace has no bank B, which passes that rebuild write.

<code>core.Registry :: struct</code>
--------------------------------------

<b>readers:</b> []Reader
--------------------------

```
renderers: []Renderer
```

```
core.Render_Context :: struct
```

```
view: View
```

```
doc: Document
```

```
out: ^Emitter
```

```
reports: ^Reports
```

```
options: Render_Options
```

```
policy: Policy
```

```
limits: Limits
```

```
scratch: ^Scratch
```

```
core.Render_Options :: struct
```

```
standalone: bool
```

a complete document rather than a body fragment.

```
reference_profile: bool
```

CommonMark reference HTML, for conformance tests.

```
title: string
```

overrides the document's own title.

```
language: string
```

BCP 47 tag; "en" when empty.

```
stylesheet: string
```

standalone HTML: CSS emitted in a style element.

```
section_numbers: bool
```

**anchor\_prefix:** string

prepended to every anchor, e.g. a document ID in a book.

**permalinks:** bool

HTML: a link to each heading, caption, and signature.

**permalink\_icon:** string

“¶” when empty.

**secnumber\_suffix:** string

after section numbers; “. “ when empty.

**highlighter:** Highlighter

**labels:** ^Labels

the words renderers write, translated; nil: English.

**math:** Math\_Style

**math\_inline:** [2]string

HTML math delimiters for the script that typesets it, with Math\_Style.Sphinx; empty:  $(\ )$  inline and  $[ \ ]$  displayed, as Sphinx’s mathjax\_inline and mathjax\_display default to.

**math\_display:** [2]string

**core.Render\_Proc** :: #type proc(ctx: ^Render\_Context) -> Status

Render\_Proc writes ctx.view with ctx.out and returns its status; see Renderer.

**core.Renderer** :: struct

**id:** string

**name:** string

**version:** string

**media\_type:** string

**conformance:** Conformance

**render:** Render\_Proc

**core.Reports** :: struct

**items:** []Diagnostic

**used:** int

**suppressed:** int

**truncated:** bool

**muted:** bool

write mode repeats the checked encoding without reporting.

**primary:** Diagnostic

**has\_primary:** bool

**errors:** int

**warnings:** int

**fatal:** Status

the first error that is not a content error; see content\_error.

**core.Request** :: struct

**source:** Source

**reader:** string

**renderer:** string

**read:** Read\_Options

**render:** Render\_Options

**policy:** Policy**limits:** Limits**passes:** []Pass**resources:** Resource\_Provider**cancel:** Cancel\_Probe**source\_map:** []Map\_Entry

optional; filled by renderers that support it.

```
core.Resource_Provider :: struct
```

Resource\_Provider belongs to the caller's trust boundary and may allocate. The core passes the requesting source and the name as written; the provider returns borrowed UTF-8 text valid for the whole conversion, and a canonical name for cycle checks.

```
procedure: proc(user: rawptr, from: Source, name:
string) -> (text: string, canonical: string, status:
Resource_Status)
```

**user:** rawptr

```
core.Resource_Status :: enum u8
```

**Ok****Missing****Refused****Limit**

the host's memory budget cannot hold it.

```
core.Result :: struct
```

Result is the outcome of a stage. reports borrows the workspace's reports region, valid until the next conversion or reset; primary is the first error by value, so it survives both. On Capacity, need names the region to enlarge.



```
status: Status
```

```
written: int
```

```
reports: []Diagnostic
```

```
primary: Diagnostic
```

```
need: Capacity_Need
```

```
mapped: int
```

source\_map entries written.

```
suppressed: int
```

```
truncated: bool
```

```
core.Scratch :: struct
```

```
bytes: []u8
```

```
used: int
```

```
high: int
```

high-water mark, for sizing evidence.

```
core.Scratch_Mark :: distinct int
```

Scratch is stage-local storage carved from the caller's byte slice. Taking scratch advances a used-count; it is not an allocator call. A stage records a mark and releases to it when it finishes.

```
core.Severity :: enum u8
```

```
Note
```

```
Warning
```

```
Error
```

```
core.Slab_Plan :: struct
```

```
nodes: int
```

```
attributes: int
```

```
lists: int
```

```
links: int
```

```
cells: int
```

```
text: int
```

```
scratch: int
```

```
sources: int
```

```
reports: int
```

```
output: int
```

```
second_bank: bool
```

```
core.Snapshot :: struct
```

```
nodes: Table(Node)
```

```
attributes: Table(Attribute)
```

```
lists: Table(List_Info)
```

```
links: Table(Link_Info)
```

```
cells: Table(Cell_Info)
```

```
text: []u8
```

```
text_used: int
```

```
core.Snapshot_Storage :: struct
```

```
nodes: []Node
```

```
attributes: []Attribute
```

```
lists: []List_Info
```

```
links: []Link_Info
```

```
cells: []Cell_Info
```

```
text: []u8
```

```
core.Source :: struct
```

```
name: string
```

```
text: string
```

```
core.Source_Entry :: struct
```

```
using source: Source
```

```
parent: Source_Id
```

the including source; itself for the main source.

```
included_at: Source_Span
```

the include directive, when included.

```
use: Source_Use
```

how an included source is read.

```
core.Source_Id :: distinct u16
```

```
core.Source_Span :: struct
```

Source\_Span is a half-open byte range in a source of the workspace source table.

<b>source:</b> Source_Id
--------------------------

<b>start:</b> u32
-------------------

<b>end:</b> u32
-----------------

core. <b>Source_Use</b> :: <i>enum</i> u8
-------------------------------------------

Source\_Use tells how the reader uses an included source: parsed as markup (include), or shown as it is (literalinclude, a graph file). Hosts can tell a document's parts from files it only quotes, as Sphinx's env.included does.

<b>Parsed</b>
---------------

<b>Literal</b>
----------------

core. <b>Stack</b> :: <i>struct</i> (\$T: <i>typeid</i> )
-----------------------------------------------------------

Stack is a bounded LIFO over caller-provided items.

<b>items:</b> []T
-------------------

<b>used:</b> int
------------------

core. <b>Stack_Guard</b> :: <i>struct</i>
-------------------------------------------

<b>base:</b> uintptr
----------------------

<b>limit:</b> uintptr
-----------------------

core. <b>Status</b> :: <i>enum</i> u8
---------------------------------------

<b>Ok</b>
-----------

<b>Invalid_Input</b>
----------------------

<b>Unsupported</b>
--------------------

<b>Policy</b>
---------------

<b>Capacity</b>
-----------------

**Limit****Cancelled****Internal****core.Storage** :: struct

Storage lists every region a workspace borrows. All lengths are capacities.

**banks:** [2]Snapshot\_Storage**scratch:** []u8**sources:** []Source\_Entry**reports:** []Diagnostic**core.Symbol** :: struct

Symbol\_Table is open addressing over caller-provided slots. It never grows. The load factor stays at or below one half, and probes are bounded, so hostile keys cost a bounded amount of work and end in a limit report rather than a slow scan.

**hash:** u64**key:** string

borrowed; must outlive the table.

**value:** u32**used:** bool**core.Symbol\_Status** :: enum u8**Ok****Duplicate****Full**

**Probe\_Limit****core.Symbol\_Table** :: *struct***slots**: []Symbol**count**: int**max\_probe**: int**core.Table** :: *struct*(\$T: typeid)

Table is a caller-owned SoA slice with a used-count. Appending never allocates.

**rows**: []T**used**: int**core.Text** :: *struct*

Text is a byte range in the active snapshot's text pool.

**start**: u32**length**: u32**core.View** :: *struct*

View is the read-only form renderers and passes traverse. Obtaining a view checks the handle once; accessors are then plain bounds-checked indexing. Strings they return borrow the snapshot's text pool or the sources, valid while the document is valid. An id out of range is a programming error (a bounds-check panic), not a data error: ids come from the view itself.

**snap**: ^Snapshot**sources**: []Source\_Entry**core.Workspace** :: *struct*

Workspace is one conversion's borrowed storage. A document handle names a generation (one conversion, from read to render) and a bank's revision (one

build of that bank); both only grow, so a handle to a cleared or rebuilt bank never becomes valid again.

**banks:** [2]Snapshot

**revisions:** [2]u32

raised whenever a bank is cleared to be built again.

**active:** int

**generation:** u32

**scratch:** Scratch

**sources:** []Source\_Entry

**sources\_used:** int

**reports:** Reports

**keep\_errors:** bool

the read's Policy.keep\_documents\_with\_errors; passes honour it.

**output:** []u8

optional; set by init\_workspace\_from\_bytes for convert\_text.

**ready:** bool

### Procedures

```
core.add_attribute :: proc(b: ^Builder, id: Node_Id, key:
Attribute_Key, value: Text, name := Text{})
```

add\_attribute requires that id is the most recently begun node, so each node's attributes stay contiguous.

```
core.add_attribute_string :: proc(b: ^Builder, id: Node_Id,
key: Attribute_Key, value: string, name := "")
```

add\_attribute\_string copies value (and name, for a Custom key) into the text pool and adds the attribute as add\_attribute does.

```
core.add_flags :: proc(b: ^Builder, id: Node_Id, f:
Node_Flags)
```

add\_flags adds flags to node id; nothing happens once the builder has failed.

```
core.add_leaf :: proc(b: ^Builder, kind: Node_Kind, span:
Source_Span, text := Text{}) -> (Node_Id, bool) #optional_ok
```

add\_leaf adds a closed node of kind with text (borrowed from the bank's pool) and no children; it fails as begin\_node does.

```
core.add_source :: proc(ctx: ^Read_Context, name, text:
string) -> (Source_Id, bool)
```

add\_source registers text that is not a file, such as a configured prologue, so nodes parsed from it carry spans into it. The text must outlive the conversion.

```
core.add_text :: proc(b: ^Builder, s: string) -> Text
```

add\_text copies s into the text pool.

```
core.admonition_label :: proc(o: Render_Options, k:
Admonition_Kind) -> string
```

admonition\_label is the title of an admonition kind; "" for a generic admonition.

```
core.admonition_name :: proc(k: Admonition_Kind) -> string
```

admonition\_name returns the static class name of an admonition kind ("note", "tip").

```
core.anchor_extras :: proc(a: ^Anchors, owner: Node_Id) -
> []Anchor_Extra
```

anchor\_extras returns the ids generated for a node's sake (see Anchor\_Extra), in the order they were given.

```
core.anchor_ids :: proc(a: ^Anchors, id: Node_Id) ->
Anchor_Ids
```

anchor\_ids iterates every id of a node, the primary one first.

```
core.anchor_of :: proc(a: ^Anchors, id: Node_Id) -> (string,
bool)
```

anchor\_of returns the primary id of a node, borrowed from the anchor table, or false when the node has none.



```
core.arg_int :: proc(v: i64) -> Diagnostic_Arg
```

arg\_int makes an integer diagnostic argument.

```
core.arg_span :: proc(s: Source_Span) -> Diagnostic_Arg
```

arg\_span makes a diagnostic argument that quotes the source text of s.

```
core.arg_static :: proc(s: string) -> Diagnostic_Arg
```

arg\_static makes a diagnostic argument of static text, which must outlive the reports.

```
core.attribute :: proc(v: View, id: Node_Id, key:
Attribute_Key) -> (string, bool)
```

attribute returns the value of node id's first attribute with key, borrowed from the view, or false when it has none.

```
core.attributes :: proc(v: View, id: Node_Id) -> Attributes
```

attributes starts an iteration over every attribute of node id, for next\_attribute.

```
core.begin_node :: proc(b: ^Builder, kind: Node_Kind, span:
Source_Span) -> (Node_Id, bool) #optional_ok
```

begin\_node opens a node of kind as the last child of the current node, to be closed with end\_node. It fails, poisoning the builder and returning INVALID\_NODE, when the nesting is illegal (child\_allowed), a limit is reached, or the nodes region is full.

```
core.buffer_string :: proc(b: ^Buffer) -> string
```

buffer\_string returns the bytes written, borrowed from the buffer's storage.

```
core.buffer_write :: proc(b: ^Buffer, s: string)
```

buffer\_write appends s, or sets overflow and drops it, and every later write, when it does not fit; wanted counts it either way.

```
core.buffer_write_byte :: proc(b: ^Buffer, c: u8)
```

buffer\_write\_byte appends one byte, as buffer\_write does.

```
core.build_anchors :: proc(ctx: ^Render_Context, a: ^Anchors,
headings: bool) -> Status
```

build\_anchors takes its storage from the render context's scratch. Identifiers are made from text in the snapshot's pool, and folding never more than

doubles its length, so twice the pool plus a fixed overhead per anchor always suffices. A node's own name is gathered in room measured for the longest one.

```
core.builder_capacity :: proc(b: ^Builder, region: Region,
minimum: u64)
```

`builder_capacity` poisons the builder with a Capacity status whose need names region and the minimum it lacks, reporting CORE\_CAPACITY; nothing happens if it already failed.

```
core.builder_fail :: proc(b: ^Builder, d: Diagnostic)
```

`builder_fail` reports `d` and poisons the builder with its status (the first failure wins); later builder calls do nothing.

```
core.builder_finish :: proc(b: ^Builder) -> Status
```

`builder_finish` closes the root. Every other node must already be closed.

```
core.builder_init :: proc(b: ^Builder, snap: ^Snapshot,
scratch: ^Scratch, reports: ^Reports, limits: Limits,
root_span: Source_Span) -> Status
```

`builder_init` starts a builder over `snap` with its open-node stack taken from `scratch`, and opens the root Document node at `root_span`. The engine calls it for every read and pass; tests use it to drive a reader's pieces. It returns Capacity, with `b.need` set, when `scratch` cannot hold the stack. The builder borrows every argument for one stage.

```
core.cell_info :: proc(v: View, id: Node_Id) -> Cell_Info
```

`cell_info` returns the cell payload of node `id`, or a cell spanning one row and one column when it has none.

```
core.check_render :: proc(ws: ^Workspace, doc: Document,
renderer: Renderer, request: Request) -> Result
```

`check_render` runs the renderer in measure mode and reports its exact output size, as need, and every support problem, without writing. `doc` must be a valid document of `ws`.

```
core.child_count :: proc(v: View, id: Node_Id) -> int
```

`child_count` counts the children of node `id`, in time linear in their number.

```
core.children :: proc(v: View, id: Node_Id) -> Children
```

`children` starts an iteration over the children of node `id`, for `next_child`.

---

```
core.code_caption :: proc(v: View, id: Node_Id) -> Node_Id
```

---

code\_caption is a code block's Caption child (its parsed Custom caption), or INVALID\_NODE when it has none.

---

```
core.content_error :: proc(status: Status) -> bool
```

---

content\_error tells whether an error's status describes the content (an unknown directive or role, a bad option, an unresolved reference, content refused by policy) rather than a tree that cannot be used: capacity, limits, cancellation, or a bug. Policy.keep\_documents\_with\_errors keeps documents whose errors are all content errors.

---

```
core.content_model :: proc(k: Node_Kind) -> Content
```

---

content\_model tells what children nodes of kind k may have; see child\_allowed. The switch is exhaustive, so a new kind does not compile until it says.

---

```
core.contents_scope :: proc(v: View, contents: Node_Id) ->
(first, end: Node_Id, base: int)
```

---

contents\_scope is the range of nodes a table of contents lists and the number of sections around that range.

---

```
core.convert :: proc(ws: ^Workspace, registry: Registry,
request: Request, output: [u8]) -> Result
```

---

convert reads, optionally transforms, and renders one source into output, which the caller owns; on success output[:written] holds it, and on failure written is 0. It starts a new workspace generation, which invalidates every earlier document, even on failure. It checks the registry first (validate\_registry). The result borrows the workspace: its reports are valid until the next conversion or reset. It never allocates; every failure is a status with a primary diagnostic.

---

```
core.convert_text :: proc(ws: ^Workspace, registry: Registry,
reader, renderer, text: string, name := "input") -> (string,
Result)
```

---

convert\_text converts text with default\_request and the workspace's own output region (set by the slab initializers; convert with an output slice otherwise). The returned string borrows that region and is valid until the next conversion or reset of the workspace; on failure it is "" and the result says why.

```
core.copy_payload :: proc(b: ^Builder, v: View, from, to: Node_Id)
```

copy\_payload copies a node's list, cell, or link row, with its texts, to the node to of b's document; passes that rebuild a document share it.

```
core.copy_subtree :: proc(b: ^Builder, v: View, id: Node_Id)
-> Node_Id
```

copy\_subtree copies a node and its descendants from a document view into a builder, including text, attributes, and payload rows, and returns the copy of id. Copies are appended in order, one per node, so the copy of a descendant d is copy + (d - id): a pass can adjust copies with set\_value, set\_text, add\_flags, and set\_kind. Internal link targets keep their numbering only when the pass copies the whole document in order. On failure the builder is poisoned and the result is INVALID\_NODE.

```
core.current :: proc(b: ^Builder) -> Node_Id
```

current returns the innermost open node, or INVALID\_NODE when none is open.

```
core.current_kind :: proc(b: ^Builder) -> Node_Kind
```

current\_kind returns the kind of the innermost open node; Document when none is open.

```
core.custom_attribute :: proc(v: View, id: Node_Id, name: string) -> (string, bool)
```

custom\_attribute returns the value of node id's Custom attribute named name, borrowed from the view, or false when it has none.

```
core.decode_last_rune :: proc(s: string, end: int) -> (rune, int)
```

decode\_last\_rune decodes the UTF-8 character ending at byte end of s, as decode\_rune.

```
core.decode_rune :: proc(s: string, i: int) -> (rune, int)
```

decode\_rune decodes the UTF-8 character starting at byte i of s and returns it with its width; an invalid byte decodes as the replacement character, width 1.

```
core.default_limits :: proc() -> Limits
```

default\_limits returns the documented limits by value; zero limits never mean unlimited.

```
core.default_policy :: proc() -> Policy
```

default\_policy returns the safe policy by value: raw content, unsafe URLs, and math the target cannot translate are refused; includes may be requested.

```
core.default_read_options :: proc() -> Read_Options
```

default\_read\_options returns the read options by value: RST title promotion on, tab width 8, and no reader settings.

```
core.default_request :: proc(name, text, reader, renderer:
string) -> Request
```

default\_request assembles a complete Request with the documented defaults: default\_read\_options, a standalone render, default\_policy, and default\_limits, with no passes, resource provider, or cancellation probe. It borrows name and text. Change fields afterwards; lib/defaults has typed read options for the built-in readers.

```
core.diagnostic :: proc(m: Message, span: Source_Span, a :=
Diagnostic_Arg{}, b := Diagnostic_Arg{}) -> Diagnostic
```

diagnostic builds a Diagnostic by value from a static message, a span, and up to two arguments; it copies nothing.

```
core.document_valid :: proc(doc: Document) -> bool
```

document\_valid reports whether doc still names the snapshot it was built as: the same generation, the active bank, and the bank's revision at that time.

```
core.dump_tree :: proc(v: View, e: ^Emitter)
```

dump\_tree writes a readable, deterministic outline of a document. Tests use it as a renderer-independent golden format:

```
document
 section value=1
 heading value=1
 text "Title"
```

```
core.emit :: proc(e: ^Emitter, s: string)
```

emit writes s: Measure mode counts it, Write mode also stores it in the output.

```
core.emit_byte :: proc(e: ^Emitter, c: u8)
```

emit\_byte writes one byte, as emit does.

```
core.emit_int :: proc(e: ^Emitter, value: i64)
```

emit\_int writes value in decimal, as emit does.

```
core.emit_mark :: proc(e: ^Emitter, span: Source_Span)
```

emit\_mark records that output from here on originates at span.

```
core.emit_rune :: proc(e: ^Emitter, r: rune)
```

emit\_rune writes r UTF-8 encoded, as emit does.

```
core.encode_rune :: proc(r: rune) -> ([4]u8, int)
```

encode\_rune returns r encoded as UTF-8 and the number of bytes used.

```
core.end_node :: proc(b: ^Builder, id: Node_Id)
```

end\_node closes id, which must be the innermost open node, and records its subtree end; closing out of order poisons the builder.

```
core.extension_as :: proc(e: Read_Extension, $T: typeid) ->
(settings: ^T, ok: bool)
```

extension\_as returns the settings e carries as ^T: nil and true when e is empty, nil and false when it carries another type. It neither reports nor allocates.

```
core.extra_anchor :: proc(a: ^Anchors, owner, item: Node_Id)
-> (string, bool)
```

extra\_anchor returns the id of an element generated for owner and item.

```
core.extra_text :: proc(a: ^Anchors, e: Anchor_Extra) ->
string
```

extra\_text is the id of a generated element.

```
core.find_reader :: proc(r: Registry, id: string) -> (Reader,
bool)
```

find\_reader returns the reader of r with id, by value, or false.

```
core.find_renderer :: proc(r: Registry, id: string) ->
(Renderer, bool)
```

find\_renderer returns the renderer of r with id, by value, or false.

```
core.first_child :: proc(v: View, id: Node_Id) -> Node_Id
```

first\_child returns the first child of node id, or INVALID\_NODE when it has none.

```
core.first_problem :: proc(r: ^Reports) -> Diagnostic
```

first\_problem returns the primary error, or the first warning when there is none.

```
core.has_children :: proc(v: View, id: Node_Id) -> bool
```

has\_children tells whether node id has any child.

```
core.has_class :: proc(v: View, id: Node_Id, class: string) -> bool
```

has\_class looks through every class attribute; a value may hold several classes.

```
core.id_fold :: proc(r: rune) -> string
```

id\_fold is the ASCII transliteration make\_id gives a lowercase rune (“é” gives “e”, “2” gives “2”), from Unicode’s decompositions, or “” when it has none. The string is static.

```
core.include_source :: proc(ctx: ^Read_Context, from: Source_Id, name: string, span: Source_Span, use := Source_Use.Parsed) -> (Source_Id, bool)
```

include\_source asks the host for a resource named by a directive in source from, records it in the source table, and returns its ID. Every refusal is reported at span. use says whether the reader parses the text or shows it as it is.

```
core.init_workspace :: proc(ws: ^Workspace, storage: Storage) -> Result
```

init\_workspace borrows storage. It checks alignment-free invariants: every region is disjoint, and bank A can hold at least a root node. Initializing a workspace again invalidates every document of its earlier storage.

```
core.init_workspace_from_bytes :: proc(ws: ^Workspace, slab: []u8) -> Result
```

init\_workspace\_from\_bytes carves the regions of slab\_plan(estimate) from slab, one bank, for the largest input estimate whose plan fits, and initializes the workspace. The workspace borrows slab, which must outlive it; a slab too small for any plan is a capacity result. For passes that rebuild, use init\_workspace\_with\_plan.

```
core.init_workspace_with_plan :: proc(ws: ^Workspace, slab: []u8, p: Slab_Plan) -> Result
```

init\_workspace\_with\_plan carves exactly the regions of p from slab, and the rest of the slab becomes scratch. It is the authoritative path:

`init_workspace_from_bytes` only chooses a plan. A plan that cannot be sized is `Invalid_Input`; a slab smaller than `plan_bytes(p)` is a capacity result.

---

```
core.is_ascii_alnum :: proc(c: u8) -> bool
```

---

`is_ascii_alnum` tells whether `c` is an ASCII letter or digit.

---

```
core.is_ascii_digit :: proc(c: u8) -> bool
```

---

`is_ascii_digit` tells whether `c` is an ASCII digit.

---

```
core.is_ascii_punct :: proc(c: u8) -> bool
```

---

`is_ascii_punct` tells whether `c` is ASCII punctuation, as `CommonMark` defines it.

---

```
core.is_ascii_space :: proc(c: u8) -> bool
```

---

`is_ascii_space` tells whether `c` is ASCII white space: space, tab, LF, CR, FF, or VT.

---

```
core.is_unicode_punctuation :: proc(r: rune) -> bool
```

---

`is_unicode_punctuation` follows `CommonMark 0.31`: general categories P and S.

---

```
core.is_unicode_whitespace :: proc(r: rune) -> bool
```

---

`is_unicode_whitespace` follows `CommonMark`: Zs characters, tab, LF, FF, and CR.

---

```
core.kept_message :: proc(policy: Policy, m: Message) -> Message
```

---

`kept_message` is `m` as a renderer reports it: a content error becomes a warning when the policy keeps documents with errors, so the page is still written.

---

```
core.kind :: proc(v: View, id: Node_Id) -> Node_Kind
```

---

`kind` returns the kind of node `id`.

---

```
core.kind_name :: proc(k: Node_Kind) -> string
```

---

`kind_name` returns the static name of a node kind, as dumps and diagnostics write it.

---

```
core.label :: proc(o: Render_Options, l: Label) -> string
```

---

`label` returns a label in the render's language, or in English when it has none.



```
core.line_bounds :: proc(text: string, offset: int) ->
(start, end: int)
```

line\_bounds returns the byte range of the line containing offset, without the break.

```
core.line_column :: proc(text: string, offset: int) -> (line,
column: int)
```

line\_column converts a byte offset into a 1-based line and byte column.

```
core.link_info :: proc(v: View, id: Node_Id) -> (Link_Info,
bool)
```

link\_info returns the link payload of node id, or false and an unresolved link when it has none.

```
core.list_info :: proc(v: View, id: Node_Id) -> List_Info
```

list\_info returns the list payload of node id, or a bullet list starting at 1 when it has none.

```
core.load_object :: proc(ws: ^Workspace, bytes: []u8, limits:
Limits) -> Read_Result
```

load\_object rebuilds a document from bytes written by write\_object into bank A of a new workspace generation. bytes must outlive the document: source names and texts are views into them.

```
core.make_id :: proc(out: ^Buffer, name: string)
```

make\_id turns a name into an identifier matching `[a-z](-?[a-z0-9]+)*`: letters are lowercased and transliterated to ASCII (ID\_FOLDS, derived from Unicode data), every other run of characters becomes one hyphen, and leading digits and hyphens are dropped, as published Sphinx URLs expect. The result may be empty.

```
core.media_kind :: proc(path: string) -> Media_Kind
```

media\_kind reads a path's extension, ignoring case, a query, and a fragment.

```
core.next_anchor_id :: proc(a: ^Anchors, it: ^Anchor_Ids) ->
(string, bool)
```

next\_anchor\_id returns the next id of an anchor\_ids iteration, borrowed from the anchor table; false ends it.

```
core.next_attribute :: proc(v: View, it: ^Attributes) ->
(Attribute, bool)
```

next\_attribute returns the next attribute of an attributes iteration, whose Text values index the view's pool; false ends it.

```
core.next_child :: proc(v: View, it: ^Children) -> (Node_Id,
bool)
```

next\_child returns the next child of a children iteration; false ends it.

```
core.nfd_first :: proc(s: string) -> rune
```

nfd\_first returns the first character of s in NFD, as Sphinx takes an index entry's letter: `unicodedata.normalize("NFD", s[o])[o]`.

```
core.no_reader_settings :: proc(ctx: ^Read_Context) -> bool
```

no\_reader\_settings is reader\_settings for a reader that takes no settings: any extension is reported (CORE\_SETTINGS\_NONE) and the result is false.

```
core.node :: proc(v: View, id: Node_Id) -> Node
```

node returns node id by value.

```
core.node_count :: proc(v: View) -> int
```

node\_count returns the number of nodes in the document, the root included.

```
core.node_text :: proc(v: View, id: Node_Id) -> string
```

node\_text returns the text of node id, borrowed from the view.

```
core.normalize_label :: proc(out: ^Buffer, label: string)
```

normalize\_label writes the case-folded label with runs of white space collapsed to one space and the ends trimmed. It is used for CommonMark labels and RST names.

```
core.normalize_nfd_into :: proc(dst: []u8, s: string) -> (n:
int, ok: bool) #optional_ok
```

normalize\_nfd\_into writes s in Normalization Form D into dst, the caller's storage, and returns the length of the whole normal form. It never allocates and never borrows s: the result is always `dst[:n]`. When dst is too small, ok is false, n is the size dst needs, and dst holds only a prefix; a nil dst measures.

```
core.offset_in :: proc "contextless" (outer, inner: string) -> (int, bool)
```

offset\_in returns where inner starts in outer when inner is a slice of outer's bytes, and false otherwise (an empty inner included). It compares addresses only and reads neither string, so it is safe for any two strings.

```
core.parent_of :: proc(v: View, id: Node_Id) -> Node_Id
```

parent\_of scans backwards; it is linear and meant for diagnostics, not traversal.

```
core.plain_text :: proc(v: View, id: Node_Id, out: ^Buffer)
```

plain\_text writes the concatenated text of a subtree, as used for alt text and ids.

```
core.plan_bytes :: proc(p: Slab_Plan) -> (bytes: int, ok: bool) #optional_ok
```

plan\_bytes returns the slab size a plan needs. ok is false when a capacity is negative or the size overflows, which init\_workspace\_with\_plan reports.

```
core.pool_string :: proc(b: ^Builder, t: Text) -> string
```

pool\_string returns the text t of the builder's pool; it borrows the bank for the stage.

```
core.pop :: proc(s: ^Stack($T)) -> (v: T, ok: bool)
```

pop removes and returns the top item; false when the stack is empty.

```
core.push :: proc(s: ^Stack($T), v: T) -> bool
```

push appends v; false, changing nothing, when the stack is full.

```
core.read_document :: proc(ws: ^Workspace, reader: Reader, request: Request) -> Read_Result
```

read\_document parses the request's source into bank A of a new generation and validates it. The document borrows the workspace and the request's source text, which must outlive it; it is valid until the next conversion, reset, or rebuild of bank A, and is returned only on success.

```
core.read_extension :: proc(settings: ^$T) -> Read_Extension
```

read\_extension wraps a pointer to a reader's settings with their type. It borrows settings, which must outlive the read.

```
core.reader_settings :: proc(ctx: ^Read_Context, $T: typeid)
-> (settings: ^T, ok: bool)
```

reader\_settings is extension\_as at a reader's entry: settings of another type are reported (CORE\_SETTINGS\_TYPE, naming both types' packages) and ok is false; the reader then returns Invalid\_Input before reading anything. nil and true mean the defaults.

```
core.render_document :: proc(ws: ^Workspace, doc: Document,
renderer: Renderer, request: Request, output: [u8] -> Result
```

render\_document checks and measures, then writes into the caller's output only if it fits; otherwise the result is Capacity with the exact size needed, and output is untouched. doc must be a valid document of ws, and stays valid.

```
core.report :: proc(r: ^Reports, d: Diagnostic)
```

report records a diagnostic. The first error is kept by value in primary even when storage is full, so capacity failures remain explainable.

```
core.report_message :: proc(r: ^Reports, m: Message, span:
Source_Span, a := Diagnostic_Arg{}, b := Diagnostic_Arg{})
```

report\_message reports the message m at span with its arguments, as report does.

```
core.reset_workspace :: proc(ws: ^Workspace)
```

reset\_workspace invalidates every document and clears all used-counts.

```
core.resolve_link :: proc(b: ^Builder, row: u32, target:
Node_Id, kind: Link_Kind, destination := Text{})
```

resolve\_link records the resolution of a link row during a reader's resolve stage.

```
core.scratch_buffer :: proc(s: ^Scratch, n: int) -> (b:
Buffer, missing: u64, ok: bool)
```

scratch\_buffer takes a Buffer of n bytes from scratch, usually the wanted count of a measuring pass into a zero Buffer; ok is false, with the bytes missing, when scratch cannot hold it. The caller releases the scratch.

```
core.scratch_mark :: proc(s: ^Scratch) -> Scratch_Mark
```

scratch\_mark records how much scratch is taken, for scratch\_release.

```
core.scratch_release :: proc(s: ^Scratch, m: Scratch_Mark)
```

scratch\_release gives back every scratch item taken since m; slices taken after m must not be used afterwards.

```
core.scratch_take :: proc(s: ^Scratch, $T: typeid, n: int) ->
(items: []T, missing: u64, ok: bool)
```

scratch\_take returns n zeroed elements of T, or ok = false with the bytes missing.

```
core.set_cell :: proc(b: ^Builder, id: Node_Id, info:
Cell_Info)
```

set\_cell stores info as node id's cell payload; a full cells region poisons the builder with Capacity.

```
core.set_kind :: proc(b: ^Builder, id: Node_Id, k: Node_Kind)
```

set\_kind changes the kind of node id, as a pass does to a copy; validation checks the result. Nothing happens once the builder has failed.

```
core.set_link :: proc(b: ^Builder, id: Node_Id, info:
Link_Info) -> u32
```

set\_link stores info as node id's link payload and returns its row, for resolve\_link, or NONE when the builder has failed or the links region is full (Capacity).

```
core.set_list :: proc(b: ^Builder, id: Node_Id, info:
List_Info)
```

set\_list stores info as node id's list payload; a full lists region poisons the builder with Capacity.

```
core.set_span_end :: proc(b: ^Builder, id: Node_Id, end: u32)
```

set\_span\_end moves the end of node id's source span, as a reader learns where a construct ends; nothing happens once the builder has failed.

```
core.set_text :: proc(b: ^Builder, id: Node_Id, t: Text)
```

set\_text sets the text of node id; nothing happens once the builder has failed.

```
core.set_value :: proc(b: ^Builder, id: Node_Id, v: i32)
```

set\_value sets the value of node id (a heading level, a list count, and so on); nothing happens once the builder has failed.

```
core.slab_plan :: proc(input_bytes: int, second_bank :=
false) -> (p: Slab_Plan, ok: bool) #optional_ok
```

slab\_plan estimates capacities for an input of the given size. The estimate is generous for prose and typical technical documents; tables and heavy markup may need more, which a capacity result reports. ok is false when a capacity would overflow.

```
core.source_entry :: proc(v: View, id: Source_Id) ->
Source_Entry
```

source\_entry returns the source-table entry of id (the main source is o), borrowed from the view, or a zero entry when id is out of range.

```
core.source_of :: proc(ctx: ^Read_Context, id: Source_Id) -
> string
```

source\_of returns the text of source id, borrowed from the workspace's source table for the read.

```
core.source_text :: proc(v: View, span: Source_Span) ->
string
```

source\_text returns the source bytes a span covers, or "" for an invalid span.

```
core.spend :: proc(b: ^Builder, steps: u64 = 1) -> bool
```

spend charges work against limits.max\_work and polls cancellation periodically.

```
core.split_label :: proc(s: string) -> (before, after:
string)
```

split\_label splits a label at its "%s": the text before and after the value.

```
core.stack_diagnostic :: proc(g: Stack_Guard, span:
Source_Span) -> Diagnostic
```

stack\_diagnostic is the Limit a guard's recursion reports.

```
@(require_results) core.stack_exhausted :: proc "contextless"
(g: Stack_Guard) -> bool
```

stack\_exhausted reports whether the next frame would take the stack past the budget. A zero guard (never started) is never exhausted.

```
@(require_results, no_sanitize_address) core.stack_guard ::
proc "contextless" (limits: Limits) -> Stack_Guard
```

stack\_guard starts measuring at the caller's frame, with the limits' stack budget.

```
core.stack_slice :: proc(s: ^Stack($T)) -> []T
```

stack\_slice returns the items in the stack, bottom first, borrowed from its storage.

```
core.subtree_end :: proc(v: View, id: Node_Id) -> Node_Id
```

subtree\_end returns the id after the last descendant of node id.

```
core.symbol_get :: proc(t: ^Symbol_Table, key: string) ->
(value: u32, found: bool)
```

symbol\_get returns the value stored for key, or NONE and false when it is absent.

```
core.symbol_put :: proc(t: ^Symbol_Table, key: string, value:
u32) -> (existing: u32, status: Symbol_Status)
```

symbol\_put inserts key. For a duplicate it leaves the table unchanged and returns the existing value.

```
core.symbol_set :: proc(t: ^Symbol_Table, key: string, value:
u32) -> bool
```

symbol\_set replaces the value of an existing key.

```
core.symbol_table_init :: proc(t: ^Symbol_Table, s: ^Scratch,
capacity: int) -> (missing: u64, ok: bool)
```

symbol\_table\_init takes  $2 * \text{capacity}$  slots (rounded to a power of two) from scratch.

```
core.text :: proc(v: View, t: Text) -> string
```

text returns the string t names in the snapshot's text pool, borrowed from the view.

```
core.text_close :: proc(b: ^Builder, start: u32) -> Text
```

text\_close returns the Text pushed since start (text\_open), or an empty Text when the builder has failed.

```
core.text_open :: proc(b: ^Builder) -> u32
```

text\_open starts a run of pushed text: the offset text\_close measures from.

```
core.text_push :: proc(b: ^Builder, s: string)
```

text\_push copies s into the text pool; a full pool poisons the builder with Capacity.

```
core.text_push_byte :: proc(b: ^Builder, c: u8)
```

text\_push\_byte copies one byte into the text pool, as text\_push does.

```
core.text_push_rune :: proc(b: ^Builder, r: rune)
```

text\_push\_rune copies r, UTF-8 encoded, into the text pool, as text\_push does.

```
core.text_rewind :: proc(b: ^Builder, start: u32)
```

text\_rewind drops text pushed since start; used when a tentative parse is abandoned.

```
core.top :: proc(s: ^Stack($T)) -> ^T
```

top returns a pointer to the top item, valid until the next push or pop, or nil when the stack is empty.

```
core.transform_document :: proc(ws: ^Workspace, doc: Document, passes: []Pass, limits := Limits{}) -> Read_Result
```

transform\_document runs passes in order. A rebuilding pass writes the inactive bank, which is validated before it becomes active; on failure the old bank stays valid and is returned as the document. A workspace without a second bank fails before a pass runs, with Capacity and need.region Second\_Bank. doc must be a valid document of ws. Zero limits mean default\_limits.

```
core.trusted_policy :: proc() -> Policy
```

trusted\_policy suits conformance tests and content the caller wrote.

```
core.unwrap_node :: proc(b: ^Builder, id: Node_Id)
```

unwrap\_node removes a closed node and keeps its descendants in place, as when an RST section becomes the document title. Links that resolved to the removed node resolve to the node that takes its index. Validation still runs after the reader finishes.



```
core.validate :: proc(snap: ^Snapshot, sources:
[]Source_Entry, scratch: ^Scratch, limits: Limits) ->
(Status, Diagnostic)
```

validate checks every structural invariant of a finished snapshot. It needs a stack of open subtree ends, taken from scratch and released before returning.

```
core.validate_registry :: proc(r: Registry) -> Result
```

validate\_registry checks that every adapter has an ID, a version, and a procedure, and that no ID is registered twice. convert checks its registry the same way before using it, so a bad registry fails with Invalid\_Input instead of the first match silently winning; a host may call this once where it builds a registry.

```
core.view :: proc(doc: Document) -> (View, bool)
```

view checks doc once (document\_valid) and returns its read-only View, which borrows the workspace until doc becomes invalid; false for a stale or foreign handle.

```
core.workspace_size :: proc(input_bytes: int) -> (bytes: int,
ok: bool) #optional_ok
```

workspace\_size returns the slab size, in bytes, of slab\_plan(input\_bytes): one bank, for conversions without rebuilding passes. ok is false when the size cannot be represented. Size a workspace for passes with a plan (see above).

```
core.write_id :: proc(out: ^Buffer, name: string)
```

write\_id lowercases letters, keeps letters and digits, and joins every other run of characters with one hyphen. It names documents and chapters, whose names may start with digits; make\_id names anchors.

```
core.write_object :: proc(v: View, e: ^Emitter)
```

write\_object serializes the document in view v. Use Measure mode first for its size.

### Constants

```
core.BUILDER_OPEN :: max(u32) - 1
```

```
core.CORE_CAPACITY
```

```
core.CORE_DEPTH
```

```
core.CORE_NODES
```

---

`core.CORE_STACK`


---



---

`core.CORE_UNHANDLED_KIND`


---



---

`core.CORE_WORK`


---



---

`core.DEFAULT_STACK_BYTES :: (512 << 10) * ASAN_FRAME_SCALE`


---

DEFAULT\_STACK\_BYTES is the stack a reader or renderer may use when Limits.stack\_bytes is 0. It holds the default max\_depth of every recursive path with room to spare, and with STACK\_RESERVE it fits a 1 MiB thread, the smallest default stack of the supported platforms' threads (Windows; Odin's threads on Unix take RLIMIT\_STACK, usually 8 MiB).

---

`core.INCLUDE_MISSING`


---



---

`core.NONE :: max(u32)`


---



---

`core.SCHEMA_VERSION :: 1`


---



---

`core.STACK_RESERVE :: (128 << 10) * ASAN_FRAME_SCALE`


---

STACK\_RESERVE is the stack a conversion needs below its deepest guard.

---

`core.VERSION :: "0.2.0"`


---

VERSION is the engine's version. It is released with Guidedog and so far follows its version (cli.VERSION), but is kept apart because programs embed the engine without the command. SCHEMA\_VERSION counts the incompatible changes to documents saved as objects.

### Variables

---

`@(rodata) core.ENGLISH_LABELS`


---

ENGLISH\_LABELS are the labels in English, Sphinx's msgids; renderers fall back to them.

---

`@(rodata) core.LIST_STYLE_NAMES`


---

## defaults

Package defaults assembles the built-in readers and renderers into an explicit, immutable registry. Embedders may build their own registry from any subset.

## Procedures

```
defaults.myst_options :: proc(settings: ^commonmark.Settings,
base := gd.Read_Options{}) -> gd.Read_Options
```

myst\_options reads with the “myst” reader’s settings, from commonmark.default\_settings or sphinx.markdown\_settings. It borrows settings, as rst\_options does.

```
defaults.registry :: proc() -> gd.Registry
```

registry returns the built-in registry by value; it borrows static tables, valid for the whole program. Readers: “rst”, “sphinx”, “commonmark” (pure CommonMark), and “myst” (MyST Markdown). Renderers: “html”, “typst”, and “text”.

```
defaults.rst_options :: proc(settings: ^rst.Settings, base :=
gd.Read_Options{}) -> gd.Read_Options
```

rst\_options reads with the “rst” reader’s settings (rst.default\_settings() to start). It borrows settings; base, when not zero, supplies the other read options.

```
defaults.sphinx_options :: proc(config: ^sphinx.Config,
base := gd.Read_Options{}) -> gd.Read_Options
```

sphinx\_options reads with the “sphinx” reader’s configuration (sphinx.default\_config() to start). It borrows config, as rst\_options does.

## fetch

### Types

```
fetch.Error :: enum u8
```

**None**

**Unsupported**

not http, https, or a file; or no libcurl on this platform.

**Missing**

no such file.

**Network**

the connection or transfer failed.

**Timeout****Status**

the server answered with a status other than 2xx.

**Too\_Large**

`fetch.Options` :: *struct*

Options bound one get. The zero value is the defaults.

**timeout\_ms**: int

the whole transfer; 0: 30 seconds; below 0: none.

**max\_bytes**: int

0: 64 MiB.

**user\_agent**: string

`fetch.Result` :: *struct*

Result is what get read, or why it could not. Its data, url, and message are allocated with the allocator given to get, and the caller owns them: destroy frees them.

**data**: []u8

**url**: string

where the data came from, after redirects.

**status**: int

the HTTP status, or 0 for files.

**error**: Error

**message**: string

what went wrong, for a report.

## Procedures

```
fetch.destroy :: proc(r: ^Result, allocator :=
context.allocator)
```

destroy frees what get allocated for r, with the same allocator, and clears r.

```
fetch.get :: proc(location: string, options: Options,
allocator := context.allocator) -> Result
```

get reads a URL or path, allocating the result with allocator; the caller owns it and frees it with destroy. It fails with an Error in the result, never a panic: a file that is missing, a transfer that fails, times out, answers with a status other than 2xx, or grows beyond max\_bytes. It blocks until the transfer ends; threads may call it at once.

## Foreign imports

```
foreign import fetch.curl "system:curl"
```

可用平台: Darwin, Linux, FreeBSD, OpenBSD, NetBSD, WASI, JS, Orca, Freestanding.

## gds

Package gds is the GDS process of docs/gds (GDS 0001) as a library: it loads the discussion records of a directory into a Catalog, plans changes to them (a new draft, a promotion, a state or label change, the regenerated index) as a Plan, applies a plan as one recoverable transaction under a lock. RST publication uses the project engine; Typst typesets PDF. Failure is a Problem that carries the exit status, never a panic. It allocates with context.allocator and frees nothing, so a caller runs it in an arena, as `guidedog gds` does, and frees the arena when done.

## Types

```
gds.Catalog :: struct
```

```
records: [dynamic]Record
```

```
reserved: int
```

```
gds.Change :: struct
```

```
path: string
```

```
before: string
```

---

**after:** string

---

---

**existed:** bool

---

---

**present:** bool

---

---

**gds.Field** *:: struct*

---

---

**name:** string

---

---

**literal:** string

---

---

**gds.Options** *:: struct*

---

---

**dir:** string

---

---

**command:** string

---

---

**identifier:** string

---

---

**author:** string

---

---

**slug:** string

---

---

**labels:** string

---

---

**state:** string

---

---

**label:** string

---

---

**to:** string

---

---

**reason:** string

---

---

**evidence:** string

---

---

**discussion:** string

---

---

**record:** string

---

<b>target:</b> string
-----------------------

<b>number:</b> int
--------------------

<b>dry_run:</b> bool
----------------------

<b>check_only:</b> bool
-------------------------

<b>historical:</b> bool
-------------------------

<b>render:</b> bool
---------------------

<b>rollback:</b> bool
-----------------------

<b>gds.Plan</b> :: struct
---------------------------

<b>changes:</b> [dynamic]Change
---------------------------------

<b>summary:</b> string
------------------------

<b>gds.Problem</b> :: struct
------------------------------

<b>code:</b> string
---------------------

<b>title:</b> string
----------------------

<b>path:</b> string
---------------------

<b>message:</b> string
------------------------

<b>hint:</b> string
---------------------

<b>exit:</b> int
------------------

<b>gds.Record</b> :: struct
-----------------------------

<b>number:</b> string
-----------------------

<b>slug:</b> string
---------------------

**title:** string**state:** string**kind:** string**created:** string**updated:** string**discussion:** string**authors:** []string**labels:** []string**path:** string**text:** string**historical:** bool**fields:** [dynamic]Field

### *Procedures*

**gds.apply** :: *proc*(*dir*: string, *plan*: Plan) -> Problem

apply writes a plan's changes as one transaction under *dir*: each file must still have the contents the plan was made from, a journal records the originals first, and every file is replaced atomically, so an interrupted apply can be rolled back by recover. Hold lock while planning and applying.

**gds.build** :: *proc*(*o*: Options, *c*: Catalog, *engine*: string) -> (string, Problem)

build publishes *o.target* "pdf", "html", or "both" (the default). An RST catalog uses the project engine: HTML is a complete site and *o.record* selects separate PDF books. A legacy Typst catalog compiles records directly into docs/build. It returns the output paths, comma-separated. Typst is needed only for RST PDF or legacy publication; engine names its program, unless the compiler is embedded. Publication is staged so a compiler failure cannot replace a previously good output.



```
gds.check_index :: proc(dir: string, c: Catalog) -> Problem
```

check\_index is a Problem when the registry or bundle of dir differs from what index\_plan would write.

```
gds.check_sources :: proc(c: Catalog) -> Problem
```

check\_sources checks that every record other than a historical one has the sections the discussion template requires; the first missing one is a Problem.

```
gds.edit_plan :: proc(o: Options, c: Catalog) -> (plan: Plan, p: Problem)
```

edit\_plan plans the change o.command asks of the record o.identifier names: promote (a draft takes the next number), state (a move the process allows, with its reason and evidence), or label; with the index refreshed. It changes nothing; apply writes the plan. A historical record, or a move the process forbids, is a Problem.

```
gds.index_plan :: proc(dir: string, c: Catalog) -> (plan: Plan, p: Problem)
```

index\_plan plans the regenerated registry.typ and bundle.typ of dir for the catalog; the plan has no changes when they are current.

```
gds.load :: proc(dir: string) -> (catalog: Catalog, p: Problem)
```

load reads the discussion directory dir (records/\*.rst or legacy \*.typ), and the numbers it reserves, into a catalog allocated in the context arena. It fails with a Problem, and changes nothing, when an update awaits recovery, a record is unreadable, too large, or malformed, or two records claim one number or slug.

```
gds.lock :: proc(dir: string) -> (^os.File, Problem)
```

lock takes the lock of the discussion directory dir and returns its file, which the caller holds while it plans and applies, and closes to release; another command holding it is a Problem. A kernel lock is released even if the process crashes. The lock file is kept: unlinking it would allow two processes to lock different inodes at the same path.

```
gds.new_plan :: proc(o: Options, c: Catalog) -> (plan: Plan, p: Problem)
```

new\_plan plans a new draft record for o.identifier: its file, from the template, and the refreshed index. It changes nothing; apply writes the plan. A bad or taken slug, or no author (–author, or git’s user.name), is a Problem.

```
gds.next_number :: proc(o: Options, c: Catalog) -> (int, Problem)
```

`next_number` is the number a promoted draft takes: `o.number` when it is free, else one past the highest that a record, the registry's reservation, or a `gds/` branch of the repository in `o.dir` holds. A number already reserved, or above 9999, is a `Problem`.

```
gds.now :: proc() -> datetime.DateTime
```

`now` is the moment a build is stamped with. As in Sphinx and other reproducible builds, `SOURCE_DATE_EPOCH` (seconds since 1970, in UTC) sets it; otherwise it is the local time, in the local time zone, which stays loaded for the life of the process.

```
gds.parse_record :: proc(path, source: string) -> (r: Record, p: Problem)
```

`parse_record` reads RST metadata or legacy `#let gds-...` lines from `source`; the record borrows `path` and `source`. A missing, repeated, unknown, or invalid field, a bad slug, or dates out of order is a `Problem`.

```
gds.problem :: proc(code, title, path, message, hint: string, exit := 1) -> Problem
```

`problem` makes a `Problem`: its code ("`gds.slug`"), title, the path it concerns, what went wrong, a hint, and the exit status the command returns for it.

```
gds.recover :: proc(dir: string, dry_run: bool) -> Problem
```

`recover` finishes an interrupted apply: it restores every file the journal names to its original contents and removes the journal; `dry_run` only checks that it can. No journal is no work.

```
gds.replace_field :: proc(source, key, literal: string) -> string
```

`replace_field` returns `source` with the value of the metadata field key replaced by the Typst literal (quoted, as "`\`"accepted`\`"), allocated in the context arena.

```
gds.resolve :: proc(c: Catalog, id: string) -> (Record, Problem)
```

`resolve` finds the one record an identifier names: its number (7 or "`0007`"), slug, title, or file name. The record borrows the catalog; no match, or more than one, is a `Problem` that lists the candidates.

```
gds.today :: proc() -> string
```

`today` is now's date, as YYYY-MM-DD.

## Constants

```
gds.LABELS
```

```
gds.STATES :: []string{"prediscussion", "discussion",
"accepted", "published", "committed", "abandoned"}
```

## gettext

Package `gettext` reads, writes, merges, and looks up GNU `gettext` message catalogs: PO and POT text files, and compiled MO files. Its lookups follow the C library's `gettext`, `ngettext`, and `pgettext`; state is explicit, a `Catalog` or a `Translator`, instead of the C library's process-wide domain table. It imports only Odin's core packages.

### Types

```
gettext.Bound_Catalog :: struct
```

`Bound_Catalog` is a catalog a translator serves for a domain.

```
domain: string
```

```
catalog: Catalog
```

```
gettext.Catalog :: struct
```

`Catalog` is one message catalog. Everything it holds lives in its own arena, so `destroy` frees it at once. Lookups use an index built on first use.

```
name: string
```

file name, or the name given to `parse_po`.

```
header: Message
```

the msgid “” entry; its `strs[o]` holds the header fields.

```
has_header: bool
```

```
messages: [dynamic]Message
```

```
plural: Plural_Rule
```

**allow\_fuzzy:** bool

look fuzzy translations up (gettext\_allow\_fuzzy\_translations).

**index:** Index

**arena:** ^Catalog\_Arena

nil when even it could not be allocated.

**allocator:** mem.Allocator

the arena's allocator; everything above lives in it.

**owner:** mem.Allocator

allocated the arena itself.

gettext.**Catalog\_Arena** :: struct

Catalog\_Arena is what a catalog owns: a growing arena on the allocator make\_catalog was given. Once refused memory it refuses everything (refused), so a catalog is never built from a mix of what was and was not allocated; catalog\_refused tells.

**using** arena: runtime.Arena

**refused:** bool

gettext.**Error** :: struct

Error explains a problem in the style of Elm's compiler: where, what, and what to do. kind == .None means success. Strings are allocated with the allocator the failing procedure was given; destroy\_error frees them.

**kind:** Error\_Kind

**file:** string

**line:** int

1-based; 0 when the problem has no line.

**column:** int

1-based byte column of the problem in excerpt; 0 for none.

**excerpt:** string

the offending line.

**message:** string

**hint:** string

gettext.**Error\_Kind** :: *enum* u8

**None**

**Syntax**

the PO text breaks the format.

**Duplicate**

a message is defined twice.

**Charset**

the catalog is not UTF-8.

**Encoding**

bytes that are not valid UTF-8.

**Plural**

a Plural-Forms header that cannot be used.

**Mo\_Format**

a truncated or inconsistent MO file.

**Io**

the file could not be read or written.

**Memory**

there was not enough memory to hold the catalog.

gettext.**Header\_Field** :: *struct*

Header\_Field is one “Name: value” line of the header entry.

**name:** string

**value:** string

**gettext.Index** :: struct

Index is a catalog's lookup table, built once from its messages. The translations it serves are copied into one pool, each form followed by “\n”, so every string a lookup returns is a slice of pool: a reader can parse them as one source (see `pool_of`).

**built:** bool

**fuzzy:** bool

built with `allow_fuzzy`.

**slots:** []i32

open addressing over entries; -1 is empty.

**entries:** []Index\_Entry

**forms:** []string

slices of pool.

**pool:** string

**gettext.Index\_Entry** :: struct

**message:** i32

in messages, or -1 for the header.

**first:** i32

first form in forms.

**count:** i32

**gettext.Merge\_Options** :: struct

**fuzzy\_matching:** bool

look for similar messages (msgmerge without -N).

**keep\_obsolete:** bool

keep translated messages that left the template as “#~”.

**stats:** ^Merge\_Stats

when set, what the fuzzy search did (fuzzy.odin).

**gettext.Merge\_Stats** :: struct

Merge\_Stats counts what a merge’s fuzzy search did; set Merge\_Options.stats to read it.

**searches:** int

msgids searched for (each distinct msgid once).

**bounds:** int

candidates whose byte counts were compared (similarity\_bound).

**comparisons:** int

candidates compared in full (similarity).

**work:** int

bytes those comparisons read.

**cut\_short:** int

searches the work bound stopped before they had looked everywhere.

**gettext.Message** :: struct

Message is one catalog entry. strs holds the msgstr, or msgstr[o..n) of a plural message; an empty string is an untranslated form.

**ctx:** string

**has\_ctx:** bool

msgctxt “” is a context too, different from none.

**id:** string

**id\_plural:** string

**has\_plural:** bool

**strs:** [dynamic]string

**comments:** [dynamic]string

“#” translator comments.

**notes:** [dynamic]string

“#.” comments for translators, from the extractor.

**references:** [dynamic]string

“#:” locations, file:line each.

**flags:** [dynamic]string

“#,” flags other than fuzzy, e.g. “c-format”.

**fuzzy:** bool

**obsolete:** bool

“#~”: kept for reuse, never looked up.

**previous:** Previous

**has\_previous:** bool

**line:** int

where the entry starts in its file; 0 when built in code.

gettext.**Plural\_Node** :: struct

**op:** Plural\_Op

**a:** i16

**b:** i16

**c:** i16



**value:** u64

gettext.Plural\_Op :: *enum* u8

**Number**

**N**

**Not**

**Mul**

**Div**

**Mod**

**Add**

**Sub**

**Less**

**Greater**

**Less\_Equal**

**Greater\_Equal**

**Equal**

**Not\_Equal**

**And**

**Or**

**Choose**

a ? b : c

```
gettext.Plural_Problem :: struct
```

Plural\_Problem is where and why an expression was refused; column is 1-based.

```
message: string
```

static text.

```
column: int
```

```
gettext.Plural_Rule :: struct
```

Plural\_Rule is a compiled Plural-Forms header: nplurals and the plural expression.

```
nplurals: int
```

```
nodes: [MAX_PLURAL_NODES]Plural_Node
```

```
count: int
```

```
root: int
```

```
gettext.Previous :: struct
```

Previous holds a fuzzy message's earlier msgctxt, msgid, and msgid\_plural (the “#” comments msgmerge writes), so a translator sees what changed.

```
ctx: string
```

```
has_ctx: bool
```

```
id: string
```

```
id_plural: string
```

```
has_plural: bool
```

```
gettext.Reference_Style :: enum u8
```

```
Wrapped
```

“#: a.rst:1 b.rst:2”, wrapped at the width, as msgmerge writes.

**One\_Per\_Line**

one “#:” line per location, as Sphinx’s gettext builder writes.

**None**

no locations (gettext\_location = False).

gettext.**Stats** :: *struct*

**translated**: int

**fuzzy**: int

**untranslated**: int

**obsolete**: int

gettext.**Translator** :: *struct*

Translator serves several text domains, each from a chain of catalogs: the first bound catalog that translates a message wins, as Sphinx searches locale\_dirs in order before its own catalogs. It stands in for the C library’s global state (bindtextdomain, textdomain, setlocale).

**language**: string

**domain**: string

the current domain (textdomain); “messages” by default.

**bound**: [dynamic]Bound\_Catalog

**allocator**: mem.Allocator

gettext.**Write\_Options** :: *struct*

**width**: int

lines are wrapped to this many characters; 0 or less: no wrapping.

**references**: Reference\_Style

---

**previous:** bool

write “#|” previous msgids of fuzzy messages.

---

**obsolete:** bool

write “#~” obsolete messages.

### *Procedures*

---

**gettext.add\_catalog** :: *proc*(tr: ^Translator, domain: string, cat: Catalog)

add\_catalog binds a catalog to a domain after those already bound; the translator owns it from now on.

---

**gettext.add\_message** :: *proc*(cat: ^Catalog, m: Message) -> ^Message

add\_message appends a copy of m, cloning its strings into the catalog’s arena, and returns the copy, which the catalog owns, or nil when the arena refused the memory. The pointer is valid until the next message is added (the list may then move). The index is rebuilt on the next lookup.

---

**gettext.bind\_text\_domain** :: *proc*(tr: ^Translator, domain, dir: string, allow\_fuzzy := false) -> (found: bool, err: Error)

bind\_text\_domain looks for dir/<language>/LC\_MESSAGES/<domain>.po, then .mo, trying the language, then its base language (pt\_BR, then pt), and binds the first catalog found, which the translator then owns. found is false, with no error, when there is none; an error is allocated with the translator’s allocator (destroy\_error).

---

**gettext.build\_index** :: *proc*(cat: ^Catalog)

build\_index (re)builds the lookup table, in the catalog’s arena. Lookups call it when the catalog changed or allow\_fuzzy was toggled; call it yourself to pay the cost up front.

---

**gettext.catalog\_bytes** :: *proc*(cat: ^Catalog) -> int

catalog\_bytes is what cat’s arena holds, in bytes: what a host charges for keeping it.

---

**gettext.catalog\_refused** :: *proc*(cat: ^Catalog) -> bool

catalog\_refused tells whether cat’s arena was refused memory, so that what it holds may be incomplete. The readers then fail with an error of kind Memory.

```
gettext.charset :: proc(cat: ^Catalog) -> string
```

charset returns the charset the header's Content-Type names, or "" without one.

```
gettext.clone_message :: proc(cat: ^Catalog, m: Message) -> Message
```

clone\_message copies m and its strings into the catalog's arena.

```
gettext.compile_plural :: proc(expr: string, nplurals: int) -> (Plural_Rule, Plural_Problem)
```

compile\_plural compiles an expression for nplurals forms.

```
gettext.default_merge_options :: proc() -> Merge_Options
```

default\_merge\_options is msgmerge's default: fuzzy matching on, obsolete messages kept.

```
gettext.default_write_options :: proc() -> Write_Options
```

default\_write\_options wraps at 76 columns, as sphinx-intl (through Babel) does.

```
gettext.destroy :: proc(cat: ^Catalog)
```

destroy frees a catalog and everything it holds.

```
gettext.destroy_error :: proc(err: Error, allocator := context.allocator)
```

destroy\_error frees the strings of an error; an error of kind Memory has static text, which it leaves alone.

```
gettext.destroy_translator :: proc(tr: ^Translator)
```

destroy\_translator frees the translator and every catalog bound to it.

```
gettext.domain_plural :: proc(tr: ^Translator, domain, id, id_plural: string, n: int) -> string
```

domain\_plural is dngettext.

```
gettext.domain_plural_with_context :: proc(tr: ^Translator, domain: string, ctx: Maybe(string), id, id_plural: string, n: int) -> string
```

domain\_plural\_with\_context is dnpgettext.

```
gettext.domain_text :: proc(tr: ^Translator, domain, id:
string) -> string
```

domain\_text is dgettext.

```
gettext.domain_with_context :: proc(tr: ^Translator, domain,
ctx, id: string) -> string
```

domain\_with\_context is dpgettext.

```
gettext.escape_string :: proc(s: string, allocator :=
context.allocator) -> string
```

escape\_string escapes a value for a PO string: backslash, quote, and the control characters C names. The result is allocated with allocator; the caller owns it.

```
gettext.find :: proc(cat: ^Catalog, id: string, ctx:
Maybe(string) = nil) -> ([string, bool])
```

find returns the translated forms of a message, slices of the index pool. ctx nil means no context. A fuzzy message is found only when allow\_fuzzy is set.

```
gettext.find_catalog_file :: proc(dir, language, domain:
string, allocator := context.allocator) -> (string, bool)
```

find\_catalog\_file returns the path of a domain's catalog for a language, if any, allocated with allocator; the caller owns it.

```
gettext.format_error :: proc(err: Error, allocator :=
context.allocator) -> string
```

format\_error renders an error as a friendly report, in a string allocated with allocator that the caller owns (" " for no error):

```
-- PO SYNTAX ERROR ----- de/index.po:12

This string has no closing quote.

12| msgstr "Hallo
 | ^
Hint: End the string with " on the same line; continue on the next line.
```

```
gettext.format_stats :: proc(s: Stats, allocator :=
context.allocator) -> string
```

format\_stats writes sphinx-intl's summary line, allocated with allocator; the caller owns it.

```
gettext.header_field :: proc(cat: ^Catalog, name: string) ->
(string, bool)
```

header\_field returns the value of a header field, compared case-insensitively.

```
gettext.language_candidates :: proc(language: string,
allocator := context.allocator) -> []string
```

language\_candidates lists the catalog directory names to try for a language, most specific first: “pt-BR” gives pt-BR, pt\_BR, and pt; “sr@latin” gives it, then sr. The list and each of its strings are allocated with allocator, apart from language and from each other; the caller owns them. A candidate whose copy is refused is left out.

```
gettext.load_mo :: proc(path: string, allocator :=
context.allocator) -> (Catalog, Error)
```

load\_mo reads a compiled catalog. Its messages have no comments or references. The catalog owns its arena, made with allocator (destroy); an error is allocated with allocator too (destroy\_error).

```
gettext.load_po :: proc(path: string, allow_fuzzy := false,
allocator := context.allocator) -> (Catalog, Error)
```

load\_po reads a PO or POT file. Errors are allocated with allocator.

```
gettext.make_catalog :: proc(name := "", allocator :=
context.allocator) -> Catalog
```

make\_catalog returns an empty catalog with a germanic plural rule (n != 1). The catalog owns an arena, made with allocator, that holds everything added to it; destroy frees it whole.

```
gettext.make_translator :: proc(language: string,
allocator := context.allocator) -> Translator
```

make\_translator returns a translator for language with no catalogs bound, in the “messages” domain. It allocates with allocator; destroy\_translator frees it and the catalogs bound to it.

```
gettext.merge :: proc(def, ref: ^Catalog, options :=
Merge_Options{fuzzy_matching = true, keep_obsolete = true},
allocator := context.allocator) -> Catalog
```

merge updates a translation (def) to a new template (ref), as msgmerge and sphinx-intl update do. The result has the template’s messages in the template’s order, with its locations, notes, and flags:

- a message found in def keeps its translation, translator comments, and fuzzy flag;

- a new message similar to an old translated one gets that translation, marked fuzzy, with the old msgid kept as its previous msgid (“#|”);
- old translated messages the template lost become obsolete (“#~”).

The header is def’s, with the template’s POT-Creation-Date. The result is a new catalog that owns its arena, made with allocator (destroy); def and ref are borrowed.

```
gettext.new_message :: proc(cat: ^Catalog) -> Message
```

new\_message returns an empty message whose lists use the catalog’s arena.

```
gettext.parse_mo :: proc(data: []u8, name := "<mo>",
allocator := context.allocator) -> (Catalog, Error)
```

parse\_mo decodes MO bytes into a catalog that copies what it keeps into its own arena, made with allocator (destroy), so data may be freed afterwards. Every offset is checked, so a damaged file gives an error, allocated with allocator (destroy\_error).

```
gettext.parse_plural_forms :: proc(value: string,
allocator := context.allocator) -> (Plural_Rule, Error)
```

parse\_plural\_forms compiles a Plural-Forms header value such as “nplurals=2; plural=(n != 1);”. The rule is a value; an error, allocated with allocator (destroy\_error), carries the value as its excerpt.

```
gettext.parse_po :: proc(text: string, name := "<string>",
allow_fuzzy := false, allocator := context.allocator) ->
(Catalog, Error)
```

parse\_po parses PO or POT text. name labels errors and the catalog. The catalog copies what it keeps, so text may be freed afterwards.

```
gettext.plural_forms_for :: proc(language: string) -> string
```

plural\_forms\_for returns a Plural-Forms header value for a language, for a new catalog: the rules the GNU gettext manual lists for common languages, and the germanic rule (n != 1) for others. “pt\_BR” and “pt-BR” find “pt\_BR”, then “pt”. The value is static; the candidates are tried in the temp allocator.

```
gettext.plural_index :: proc(rule: ^Plural_Rule, n: u64) -
> int
```

plural\_index returns the msgstr index for n; out-of-range results select form 0, as the C library does.



```
gettext.pool_of :: proc(cat: ^Catalog) -> string
```

pool\_of returns the text every translation of the catalog is a slice of, so a caller can register it once as a source and give parsed translations exact spans.

```
gettext.save_po :: proc(cat: ^Catalog, path: string,
options := Write_Options{width = 76, previous = true,
obsolete = true}, allocator := context.allocator) -> Error
```

save\_po writes a catalog to a file, formatting it in the temp allocator; an error is allocated with allocator (destroy\_error).

```
gettext.set_header_field :: proc(cat: ^Catalog, name, value:
string)
```

set\_header\_field replaces a header field, or appends it, keeping the others in order; the catalog's arena holds the copy.

```
gettext.similarity :: proc(a, b: string) -> f64
```

similarity is  $2 \cdot M / (|a| + |b|)$ , where M is the length of the longest common subsequence of bytes: 1 for equal strings, 0 for strings with nothing in common. It is the measure msgmerge's fstrcmp approximates. Long strings are compared by their first FUZZY\_PREFIX bytes.

```
gettext.sort_by_reference :: proc(cat: ^Catalog)
```

sort\_by\_reference orders messages by their first location: file name, then line number as a number, then msgid; messages without locations go last. Obsolete messages keep their place after the live ones. Each message's own locations are sorted the same way, as Sphinx sorts them.

```
gettext.stats :: proc(cat: ^Catalog) -> Stats
```

stats counts messages as sphinx-intl stat does; the header does not count.

```
gettext.text_domain :: proc(tr: ^Translator, domain: string)
```

text\_domain sets the current domain, as textdomain does; the translator owns its copy.

```
gettext.translated :: proc(m: Message) -> bool
```

translated reports whether every form of m has a translation.

```
gettext.translation :: proc(cat: ^Catalog, id: string, ctx:
Maybe(string) = nil) -> (string, bool)
```

translation returns a message's singular translation when there is one.

```
gettext.write_mo :: proc(cat: ^Catalog, include_fuzzy :=
false, allocator := context.allocator) -> []u8
```

write\_mo compiles a catalog as msgfmt does: the header and every translated message, sorted by key, with a hash table. Fuzzy messages are left out unless include\_fuzzy. The bytes are allocated with allocator; the caller owns them. They are nil when the memory was refused: a compiled catalog is never shorter than its 28-byte header.

```
gettext.write_po :: proc(cat: ^Catalog, options :=
Write_Options{width = 76, previous = true, obsolete = true},
allocator := context.allocator) -> string
```

write\_po writes a catalog as PO text: the header, then the messages in order, then the obsolete ones, each entry laid out as msgmerge lays it out. The output depends on nothing but the catalog and the options, so files diff cleanly. The text is allocated with allocator; the caller owns it.

### Procedure groups

```
gettext.plural :: proc{catalog_plural, translator_plural}
```

plural is ngettext.

```
gettext.plural_with_context ::
proc{catalog_plural_with_context,
translator_plural_with_context}
```

plural\_with\_context is npgettext.

```
gettext.text :: proc{catalog_text, translator_text}
```

text is gettext, on a catalog or on a translator's current domain.

```
gettext.with_context :: proc{catalog_with_context,
translator_with_context}
```

with\_context is pgettext.

### Constants

```
gettext.FUZZY_MIN_WORK :: 1 << 26
```

```
gettext.FUZZY_THRESHOLD :: 0.6
```

FUZZY\_THRESHOLD is msgmerge's: a changed message inherits the translation of the most similar old message when their similarity is at least this.

```
gettext.FUZZY_WORK_PER_BYTE :: 256
```

## Variables

```
@(rodata) gettext.ERROR_TITLES
```

## graphdog

Package graphdog turns Graphviz DOT source into SVG (or PNG, or laid-out DOT) by calling the Graphviz C library in process. It depends only on Odin's core packages and Graphviz.

Example:

```
import "core:fmt"
import "lib/graphdog"
source := "digraph { a -> b }"
ctx := graphdog.init()
defer graphdog.close(&ctx)
svg, err := graphdog.render(&ctx, source, {engine = .Dot}) // svg: the
caller's
if err.kind != .None do fmt.eprintln(graphdog.format_error(err, source,
"g.dot"))
defer delete(svg)
```

Threads. Graphviz keeps process-wide state (the parser, the error handler, plugin tables), so it is not thread-safe even across separate contexts. graphdog therefore holds one process-wide mutex around every call into Graphviz: render from as many threads as you like, but renders run one at a time.

## Types

```
graphdog.Attribute :: struct
```

One -G/-N/-E default. Like dot, it overrides the graph's own defaults (node [shape=box]) but not attributes written on a single node or edge.

```
kind: Attribute_Kind
```

```
name: string
```

```
value: string
```

```
graphdog.Attribute_Kind :: enum u8
```

What a default attribute applies to: -G, -N and -E on the dot command line.

```
Graph
```

```
Node
```

**Edge**

```
graphdog.Context :: struct
```

A Graphviz context: loaded plugins and nothing else. Make one with `init` and keep it; loading plugins is the expensive part.

```
gvc: ^GVC
```

```
graphdog.Engine :: enum u8
```

Layout engines. Each one is a Graphviz layout plugin: `.Dot` comes from `dot_layout`, the rest from `neato_layout`. The zero value is `.Dot`, like the `dot` command.

**Dot**

hierarchical, for directed graphs

**Neato**

spring model (stress majorization)

**Fdp**

spring model (force directed)

**Sfdp**

multiscale force directed, for large graphs

**Circo**

circular

**Twopi**

radial

**Osage**

clustered array packing

**Patchwork**

squarified treemap

```
graphdog.Error :: struct
```

Error says what went wrong and how to fix it. line and column are 1-based positions in the source (column counts characters), or 0 when Graphviz does not say. message is one sentence, in Graphviz's own words when it gave any; hint is advice for the author. render allocates near, message and hint with its allocator; delete\_error frees them.

```
kind: Error_Kind
```

```
line: int
```

```
column: int
```

```
near: string
```

the text Graphviz stopped at, for .Syntax and .Html\_Label

```
message: string
```

```
hint: string
```

```
graphdog.Error_Kind :: enum u8
```

```
None
```

```
Syntax
```

the DOT source does not parse

```
Html_Label
```

an HTML-like label (label=<...>) is malformed

```
Empty_Input
```

the source holds no graph

```
Missing_Engine
```

this Graphviz has no plugin for the layout engine

```
Missing_Format
```

this Graphviz has no plugin for the output format

**Layout**

the layout engine reported an error

**Render**

the renderer reported an error

**Out\_Of\_Memory****File\_Refused**

the caller refused a file the graph names (Options.files)

**Not\_Initialized**

render was called with a context that init did not make

```
graphdog.File_Access :: struct
```

`File_Access` decides each file name a graph gives through attribute (“src” for an `<IMG>`): it returns the path Graphviz may open, “” to leave the attribute out, or ok false to refuse the graph, which render reports as a `.File_Refused` error.

```
procedure: proc(user: rawptr, attribute, name: string) ->
(path: string, ok: bool)
```

```
user: rawptr
```

```
graphdog.Format :: enum u8
```

Output formats. The zero value is `.Svg`, which the core plugin always provides.

**Svg**

standalone SVG 1.1 document, UTF-8

**Png**

needs the cairo (pango) or gd plugin; the static build has neither

**Dot**

the input with layout positions attached, like `dot -Tdot`

```
graphdog.Options :: struct
```

```
engine: Engine
```

```
format: Format
```

```
attributes: []Attribute
```

```
warnings: ^[dynamic]string
```

When set, each Graphviz warning (“using box for unknown shape boxx”) is appended, allocated with the allocator given to render.

```
files: File_Access
```

When set, decides every file the graph names before Graphviz may open it (see files.odin); otherwise Graphviz opens what the graph names, as dot does.

```
reserve: Reserve
```

When set, counts the output before render allocates it.

```
graphdog.Reserve :: struct
```

Reserve lets the caller count the output before render allocates it: false refuses the bytes, and render reports .Out\_Of\_Memory.

```
procedure: proc(user: rawptr, bytes: int) -> bool
```

```
user: rawptr
```

## Procedures

```
graphdog.close :: proc(ctx: ^Context)
```

close frees the context. Output and errors returned by render stay valid.

```
graphdog.delete_error :: proc(err: Error, allocator := context.allocator)
```

delete\_error frees the strings of an error that render returned, with the allocator given to render.

```
graphdog.engine_name :: proc(engine: Engine) -> string
```

engine\_name is the Graphviz name of an engine (“dot”, “neato”, ...).

```
graphdog.error_title :: proc(kind: Error_Kind) -> string
```

`error_title` is a short upper-case title for the kind of error, such as “DOT SYNTAX ERROR”.

```
graphdog.format_error :: proc(err: Error, source, name: string, allocator := context.allocator) -> string
```

`format_error` writes `err` the way Elm writes compiler errors: a title rule naming the file, the source line with a caret under the problem, what went wrong, and a hint. `name` labels the source (“`diagram.dot`”, “`<stdin>`”); `source` is the text given to render. The text is allocated with `allocator`, and the caller owns it.

```
-- DOT SYNTAX ERROR -----
diagram.dot

3 | b -> -> c
 ^ ^
Graphviz stopped at `->`.

Hint: An edge needs a node on each side, like `a -> b`. ...
```

```
graphdog.format_message :: proc(title, name, message, hint: string, allocator := context.allocator) -> string
```

`format_message` writes an error that has no source to point at, such as a bad command line option, in the same shape as `format_error`. The text is allocated with `allocator`, and the caller owns it.

```
graphdog.init :: proc() -> (ctx: Context, ok: bool)
#optional_ok
```

`init` creates a Graphviz context and loads its plugins. `ok` is false only when Graphviz cannot allocate one.

```
graphdog.parse_engine :: proc(name: string) -> (Engine, bool)
```

`parse_engine` maps a Graphviz engine name to an `Engine`.

```
graphdog.parse_format :: proc(name: string) -> (Format, bool)
```

`parse_format` maps a -T format name to a `Format`.

```
graphdog.render :: proc(ctx: ^Context, source: string,
options := Options{}, allocator := context.allocator) ->
(output: []u8, err: Error)
```

`render` lays out one graph from DOT source and renders it. On success `output` holds the rendered bytes, allocated with `allocator` (`string(output)` for SVG); Graphviz’s own buffers are already freed. On failure `output` is nil and `err` says what went wrong and how to fix it; its strings are allocated with `allocator`.



too (see `delete_error`). Only the first graph in source is rendered, as with `dot` without `-O`.

```
graphdog.version :: proc(ctx: ^Context) -> string
```

version is the linked Graphviz version, such as “16.1.0”. It is owned by Graphviz and valid until close.

### Constants

```
graphdog.FILE_ATTRIBUTES :: [?]string{"image", "shapefile",
"imagepath", "fontpath"}
```

FILE\_ATTRIBUTES are the attributes through which Graphviz reads files.

```
graphdog.STATIC :: #config(GRAPHDOG_STATIC, false)
```

The raw Graphviz C API that graphdog uses, and nothing more. Every signature was checked against the headers of Graphviz 16.1 (Homebrew) and 2.42 (Debian); where they differ, the note says how this binding stays correct for both.

Linking. By default graphdog links the shared libraries that a package manager installs:

macOS	brew install graphviz	Odin already passes -L/opt/homebrew/ lib and
		-L/usr/local/lib, and Homebrew's
	dylibs carry	absolute install names, so no rpath
		is needed.
Linux	apt install libgraphviz-dev	the libraries are on the default search path.

GRAPHDOG\_LINK\_FLAGS adds linker flags for any other prefix, for example

```
-define:GRAPHDOG_LINK_FLAGS="-L/opt/gv/lib -Wl,-rpath,/opt/gv/lib"
```

`-define:GRAPHDOG_STATIC=true` links the static libraries that native/graphviz/build.sh builds instead, and preloads the dot, neato and core plugins (see `static.odin`), so the program needs no Graphviz installation at run time.

Windows has no static build: native/graphviz/windows.sh installs the official Graphviz release for Windows (the same version, checked against its SHA-256) into native/graphviz/out, and graphdog links its import libraries there. The program then needs the DLLs and the plugin list (`config8`) from native/graphviz/out/bin beside it, which is how the Windows release archive ships them.

### Foreign imports

```
@(extra_linker_flags = #config(GRAPHDOG_LINK_FLAGS, ""))
foreign import graphdog.graphviz {"../..../native/graphviz/
out/lib/libgvplugin_dot_layout.a", "../..../native/graphviz/
out/lib/libgvplugin_neato_layout.a", "../..../native/graphviz/
out/lib/libgvplugin_core.a", "../..../native/graphviz/out/lib/
libgvc.a", "../..../native/graphviz/out/lib/libpathplan.a",
"../..../native/graphviz/out/lib/libxdot.a", "../..../native/
graphviz/out/lib/libcgraph.a", "../..../native/graphviz/out/
lib/libcdt.a", "../..../native/graphviz/out/lib/libutil.a",
"../..../native/graphviz/out/lib/libvpasc.a", "system:expat",
"system:z", "system:c++"}
```

可用平台: Darwin.

The attribute takes only literals, so the vendored directory is spelled out.

```
@(extra_linker_flags = #config(GRAPHDOG_LINK_FLAGS, ""))
foreign import graphdog.graphviz {"../..../native/graphviz/
out/lib/libgvplugin_dot_layout.a", "../..../native/graphviz/
out/lib/libgvplugin_neato_layout.a", "../..../native/graphviz/
out/lib/libgvplugin_core.a", "../..../native/graphviz/out/lib/
libgvc.a", "../..../native/graphviz/out/lib/libpathplan.a",
"../..../native/graphviz/out/lib/libxdot.a", "../..../native/
graphviz/out/lib/libcgraph.a", "../..../native/graphviz/out/
lib/libcdt.a", "../..../native/graphviz/out/lib/libutil.a",
"../..../native/graphviz/out/lib/libvpasc.a", "../..../native/
graphviz/out/lib/libexpat.a", "../..../native/graphviz/out/lib/
libz.a", "system:stdc++", "system:m"}
```

可用平台: Windows, Linux, FreeBSD, OpenBSD, NetBSD, WASI, JS, Orca, Freestanding.

```
@(extra_linker_flags = #config(GRAPHDOG_LINK_FLAGS, ""))
foreign import graphdog.graphviz {"../..../native/graphviz/out/
lib/gvc.lib", "../..../native/graphviz/out/lib/cgraph.lib"}
```

可用平台: Windows.

```
@(extra_linker_flags = #config(GRAPHDOG_LINK_FLAGS, ""))
foreign import graphdog.graphviz {"system:gvc",
"system:cgraph"}
```

可用平台: Darwin, Linux, FreeBSD, OpenBSD, NetBSD, WASI, JS, Orca, Freestanding.

## highlight

Generated by tools/pygments\_styles from pygments-2.19.2's style modules and sphinx-9.1.0's pygments\_styles.py; do not edit.

*Types*

```
highlight.Color :: struct
```

Color is a CSS colour as Pygments normalizes it: six hex digits, case kept.

```
hex: [6]u8
```

```
set: bool
```

```
highlight.Html_Lines :: struct
```

Html\_Lines numbers and emphasizes lines as Pygments does for Sphinx's :linenos: and :emphasize-lines: (linenos="inline", hl\_lines): each line starts with <span class="linenos"> 7</span>, and an emphasized line, number included, is wrapped in <span class="hll">...\n</span>.

```
first: int
```

the first line's number; 0 leaves lines unnumbered.

```
emphasized: proc(user: rawptr, line: int) -> bool
```

by 1-based line; nil for none.

```
user: rawptr
```

```
highlight.Lexer :: enum u8
```

Lexer names a language. .Default and .Guess are Sphinx's pseudo languages: tokenize resolves them against the code (see resolve).

```
Text
```

```
Python
```

```
Pycon
```

```
Pytb
```

```
C
```

```
Cpp
```

**Rust****Go****Java****Kotlin****Csharp****Swift****Zig****Odin****Javascript****Typescript****Json****Yaml****Toml****Ini****Xml****Html****Css****Bash****Console**

**Powershell****Sql****Diff****Makefile****Dockerfile****Rst****Markdown****Latex****Typst****Dot****Default****Guess****highlight.Lexer\_Info** :: *struct***name:** string

Pygments' lexer name, e.g. "Python".

**aliases:** []string

the names code blocks use; the first is the canonical one.

**filenames:** []string

glob patterns: "\*.py", "Makefile".

**mimetypes:** []string**step:** Step

**spec:** ^Code\_Spec

the rule tables of lexers built on the code engine.

**highlight.Look** :: struct

Look is a token kind's resolved style.

**color:** Color

**background:** Color

**border:** Color

**bold:** bool

**italic:** bool

**underline:** bool

**highlight.Output** :: struct

Output is caller storage for the writers. They never allocate: bytes that fit go into buf, and written counts every byte the output needs, so a nil buf measures the output and a second call with a buffer of that size writes it. The same input always gives the same bytes.

**buf:** []u8

**written:** int

**highlight.Problem** :: struct

**kind:** Problem\_Kind

**name:** string

the language or style as written.

**lexer:** Lexer

**suggestion:** string

a close name, for "Did you mean".

**line:** int

Lexing\_Error: 1-based line and column (in characters) of

**column:** int

the first Error token,

**excerpt:** string

the line it is on,

**near:** string

and its text.

highlight.**Problem\_Kind** :: *enum* u8

**None**

**Unknown\_Language**

Sphinx: “Pygments lexer name ‘x’ is not known”.

**Unknown\_Style**

pygments\_style names no style.

**Lexing\_Error**

Sphinx: “Lexing literal\_block ... resulted in an error”.

highlight.**Scanner** :: *struct*

Scanner tokenizes one input. Create it with scanner\_init and read tokens with next\_token, or let tokenize and highlight\_html drive it.

**code:** string

**pos:** int

**limit:** int

steps see code[:limit]; an inner lexer sees its region only.

**state:** State

**inner:** State

a nested lexer: Python after a pycon prompt, CSS in <style>.

**region:** int

the inner lexer runs while pos < region.

**region\_start:** int

**nested:** bool

**queue:** [QUEUE\_SIZE]Token

**head:** int

**count:** int

highlight.**Style** :: *enum* u8

Style is a pygments\_style: every style Pygments ships, and Sphinx's sphinx and none.

**Abap**

**Algol**

**Algol\_Nu**

**Arduino**

**Autumn**

**Borland**

**Bw**

**Coffee**

**Colorful**



**Default****Dracula****Emacs****Friendly****Friendly\_Grayscale****Fruity****Github\_Dark****Gruvbox\_Dark****Gruvbox\_Light****Igor****Inkpot****Lightbulb****Lilypond****Lovelace****Manni****Material****Monokai****Murphy****Native**

None
------

Nord
------

Nord_Darker
-------------

One_Dark
----------

Paraiso_Dark
--------------

Paraiso_Light
---------------

Pastie
--------

Perldoc
---------

Rainbow_Dash
--------------

Rrt
-----

Sas
-----

Solarized_Dark
----------------

Solarized_Light
-----------------

Sphinx
--------

Staroffice
------------

Stata_Dark
------------

Stata_Light
-------------

Tango
-------

Trac
------

Vim

Vs

Xcode

Zenburn

```
highlight.Style_Info :: struct
```

**name:** string

**dark:** bool

the background is dark.

**background:** string

the .highlight background.

**highlight:** string

emphasized lines (.hll).

**line\_number:** string

**line\_number\_background:** string

**special\_line\_number:** string

**special\_background:** string

**base:** string

the root token's definition, inherited by all.

**rules:** []Style\_Rule

```
highlight.Style_Rule :: struct
```

Style\_Rule is one entry of a style's dict; a style has at most one per kind.

**kind:** Token\_Kind

```
definition: string
```

```
highlight.Token :: struct
```

Token is a run of the input: `code[start:end]` has kind. The tokens tokenize returns are contiguous, cover every byte exactly once, and never hold two neighbours of one kind.

```
start: u32
```

```
end: u32
```

```
kind: Token_Kind
```

```
highlight.Token_Kind :: enum u8
```

Token\_Kind mirrors Pygments' token hierarchy. Each kind has Pygments' dotted name (`kind_name`), its short CSS class (`kind_class`) and its parent (`kind_parent`), so the stylesheets Sphinx themes already ship style Guidedog's output unchanged.

```
Text
```

```
Whitespace
```

```
Escape
```

```
Error
```

```
Other
```

```
Keyword
```

```
Keyword_Constant
```

```
Keyword_Declaration
```

```
Keyword_Namespace
```

```
Keyword_Pseudo
```

<b>Keyword_Reserved</b>
-------------------------

<b>Keyword_Type</b>
---------------------

<b>Name</b>
-------------

<b>Name_Attribute</b>
-----------------------

<b>Name_Builtin</b>
---------------------

<b>Name_Builtin_Pseudo</b>
----------------------------

<b>Name_Class</b>
-------------------

<b>Name_Constant</b>
----------------------

<b>Name_Decorator</b>
-----------------------

<b>Name_Entity</b>
--------------------

<b>Name_Exception</b>
-----------------------

<b>Name_Function</b>
----------------------

<b>Name_Function_Magic</b>
----------------------------

<b>Name_Property</b>
----------------------

<b>Name_Label</b>
-------------------

<b>Name_Namespace</b>
-----------------------

<b>Name_Other</b>
-------------------

<b>Name_Tag</b>
-----------------

<b>Name_Variable</b>
----------------------

---

**Name\_Variable\_Class**

---

---

**Name\_Variable\_Global**

---

---

**Name\_Variable\_Instance**

---

---

**Name\_Variable\_Magic**

---

---

**Literal**

---

---

**Literal\_Date**

---

---

**String**

---

---

**String\_Affix**

---

---

**String\_Backtick**

---

---

**String\_Char**

---

---

**String\_Delimiter**

---

---

**String\_Doc**

---

---

**String\_Double**

---

---

**String\_Escape**

---

---

**String\_Heredoc**

---

---

**String\_Interpol**

---

---

**String\_Other**

---

---

**String\_Regex**

---

---

**String\_Single**

---

**String\_Symbol****Number****Number\_Bin****Number\_Float****Number\_Hex****Number\_Integer****Number\_Integer\_Long****Number\_Oct****Operator****Operator\_Word****Punctuation****Punctuation\_Marker****Comment****Comment\_Hashbang****Comment\_Multiline****Comment\_Preproc****Comment\_Preproc\_File****Comment\_Single****Comment\_Special**

**Generic****Generic\_Deleted****Generic\_Emph****Generic\_Emph\_Strong****Generic\_Error****Generic\_Heading****Generic\_Inserted****Generic\_Output****Generic\_Prompt****Generic\_Strong****Generic\_Subheading****Generic\_Traceback****Punctuation\_Indicator**

Pygments' YAML lexer uses two types outside its standard set; HTML gives them the classes of their ancestors too, as Pygments does.

**Literal\_Scalar\_Plain****Generic\_Underline**

Types only Pygments' styles name: github-dark's `Generic.Underline`, gruvbox's `Comment.PreProc` (not `Comment.Preproc`), and the LilyPond lexer's, which lilypond styles. `Token.String` and `Token.Number` are not `Literal.String` and `Literal.Number`.

**Comment\_PreProc****Name\_Lvalue**



Name_Backslash_Reference
--------------------------

Name_Builtin_Articulation
---------------------------

Name_Builtin_Clef
-------------------

Name_Builtin_Context
----------------------

Name_Builtin_Context_Property
-------------------------------

Name_Builtin_Dynamic
----------------------

Name_Builtin_Grob
-------------------

Name_Builtin_Grob_Property
----------------------------

Name_Builtin_Header_Variable
------------------------------

Name_Builtin_Markup_Command
-----------------------------

Name_Builtin_Music_Command
----------------------------

Name_Builtin_Music_Function
-----------------------------

Name_Builtin_Paper_Variable
-----------------------------

Name_Builtin_Repeat_Type
--------------------------

Name_Builtin_Scale
--------------------

Name_Builtin_Scheme_Builtin
-----------------------------

Name_Builtin_Scheme_Function
------------------------------

Name_Builtin_Translator
-------------------------

Token_String
--------------

**Token\_String\_Escape****Token\_String\_Symbol****Token\_Number****Pitch****Chord\_Modifier*****Procedures***

```
highlight.check_code :: proc(language, code: string) ->
Problem
```

`check_code` finds what Sphinx would warn about when it highlights code as language: an unknown language, or code that does not lex as the language it is marked as. Blocks in the “default” language never warn; they fall back to plain text.

```
highlight.check_style :: proc(name: string) -> Problem
```

`check_style` reports a `pygments_style` that names no style.

```
highlight.count_tokens :: proc(lexer: Lexer, code: string) -
> int
```

`count_tokens` is how many tokens tokenize needs for code.

```
highlight.find_lexer :: proc(name: string) -> (Lexer, bool)
```

`find_lexer` finds a lexer by alias (“py”, “c++”), as Pygments’ `get_lexer_by_name` does, ignoring case; failing that by file name (“setup.py”, “Makefile”) or MIME type.

```
highlight.find_lexer_for_filename :: proc(path: string) ->
(Lexer, bool)
```

`find_lexer_for_filename` matches the base name of a path against the lexers’ patterns.

```
highlight.find_style :: proc(name: string) -> (Style, bool)
```

`find_style` finds a style by its Pygments name, ignoring case.

```
highlight.guess_lexer :: proc(code: string) -> Lexer
```

`guess_lexer` picks a lexer from the code alone, as Sphinx’s “guess” language does. JSON that parses cleanly is JSON; otherwise the rule table decides.

```
highlight.has_errors :: proc(lexer: Lexer, code: string) -> bool
```

`has_errors` reports whether lexing code produces an Error token, which is when Sphinx warns that a block does not lex as its language.

```
highlight.highlight_html :: proc(lexer: Lexer, code: string, out: ^Output)
```

`highlight_html` lexes code and writes its HTML without a token buffer.

```
highlight.highlight_html_lines :: proc(lexer: Lexer, code: string, lines: Html_Lines, out: ^Output)
```

`highlight_html_lines` lexes code and writes it as `write_html_lines` does.

```
highlight.highlight_typst :: proc(lexer: Lexer, code: string, out: ^Output)
```

`highlight_typst` lexes code and writes its Typst array without a token buffer.

```
highlight.kind_class :: proc(kind: Token_Kind) -> string
```

`kind_class` is the CSS class Pygments’ `HtmlFormatter` writes: “k”, “s2”, “c1”; “” for Text, which HTML writes without a span.

```
highlight.kind_is :: proc(kind, ancestor: Token_Kind) -> bool
```

`kind_is` reports whether `kind` is ancestor or one of its descendants, as Pygments’ `ttype` in `Token.String` does.

```
highlight.kind_name :: proc(kind: Token_Kind) -> string
```

`kind_name` is Pygments’ dotted name without the “Token.” root: “Literal.String.Double”.

```
highlight.kind_parent :: proc(kind: Token_Kind) -> (Token_Kind, bool)
```

`kind_parent` is the kind’s parent; false for kinds directly below the root.

```
highlight.lexer_alias :: proc(lexer: Lexer) -> string
```

`lexer_alias` is the lexer’s canonical alias: “python”, “cpp”.

```
highlight.lexer_info :: proc(lexer: Lexer) -> Lexer_Info
```

lexer\_info is the lexer's static description: name, aliases, file patterns, and MIME types.

```
highlight.lexer_name :: proc(lexer: Lexer) -> string
```

lexer\_name is Pygments' name for the lexer: "Python", "C++".

```
highlight.next_token :: proc(s: ^Scanner) -> (Token, bool)
```

next\_token returns the next token; adjacent tokens of one kind come back merged.

```
highlight.output_fits :: proc(out: ^Output) -> bool
```

output\_fits reports whether everything written so far is in the buffer.

```
highlight.output_string :: proc(out: ^Output) -> string
```

output\_string is what fits in the buffer; it is the whole output when output\_fits.

```
highlight.put :: proc(out: ^Output, s: string)
```

put writes s to out: what fits into the buffer, and counts all of it.

```
highlight.put_byte :: proc(out: ^Output, c: u8)
```

put\_byte writes one byte to out, as put does.

```
highlight.put_html :: proc(out: ^Output, text: string)
```

put\_html writes text escaped as Pygments escapes it: & < > " and '.

```
highlight.resolve :: proc(lexer: Lexer, code: string) -> Lexer
```

resolve makes Sphinx's choices for a code block. .Default is Python, or a Python console session when the code starts with ">>>", and falls back to .Text when that does not lex cleanly. Python blocks that start with ">>>" are console sessions too. .Guess guesses from the code. Every other lexer is itself.

```
highlight.resolve_style :: proc(style: Style) -> (looks: [Token_Kind]Look, base: Look)
```

resolve\_style resolves every kind's look: its parent's, changed by its own rules.

```
highlight.scanner_init :: proc(s: ^Scanner, lexer: Lexer,
code: string)
```

scanner\_init prepares s to tokenize code. The pseudo lexers .Default and .Guess are resolved against the code first. Inputs are limited to 4 GiB.

```
highlight.style_name :: proc(style: Style) -> string
```

style\_name is the style's pygments\_style name, as find\_style takes it: "github-dark".

```
highlight.suggest_lexer :: proc(name: string) -> string
```

suggest\_lexer is the alias closest to a name that find\_lexer does not know, for "Did you mean" hints; "" when nothing is close.

```
highlight.token_text :: proc(t: Token, code: string) ->
string
```

token\_text is the input a token covers.

```
highlight.tokenize :: proc(lexer: Lexer, code: string,
buffer: []Token) -> (tokens: []Token, need: int)
```

tokenize fills buffer with the tokens of code and returns them. need is how many the whole input has; when it exceeds len(buffer), tokens holds only the first ones, and a buffer of need tokens holds them all. tokenize never allocates.

```
highlight.write_css :: proc(style: Style, selector: string,
out: ^Output)
```

write\_css writes the stylesheet Pygments writes for the style with the selector, as Sphinx writes \_static/pygments.css with ".highlight". It ends without a newline.

```
highlight.write_html :: proc(tokens: []Token, code: string,
out: ^Output)
```

write\_html writes tokens of code as Pygments' HTML.

```
highlight.write_html_lines :: proc(tokens: []Token, code:
string, lines: Html_Lines, out: ^Output)
```

write\_html\_lines writes tokens of code as Pygments' HTML with numbered or emphasized lines. The whole block is lexed at once, so a string or comment across lines keeps its colour on every line.

```
highlight.write_problem :: proc(p: Problem, location: string,
out: ^Output)
```

write\_problem writes an Elm-style report; location, such as “index.rst:12”, heads it.

```
-- UNKNOWN LANGUAGE ----- index.rst:12

I do not know how to highlight "pyhton", so this block is shown as plain
text.

Hint: Did you mean "python"?
```

```
highlight.write_typst :: proc(tokens: []Token, code: string,
out: ^Output)
```

write\_typst writes tokens of code as a Typst array of lines. A final newline does not start another line.

```
highlight.write_typst_theme :: proc(style: Style, out:
^Output)
```

write\_typst\_theme writes a style as a Typst dictionary: “background” is the block’s fill, “text” the default text settings, and each class maps to its text settings (fill, weight, style), with “underline”, “background” and “border” where set.

### Variables

```
highlight.STYLES
```

## highlight/guidedoc

Package highlight\_guidedoc connects the highlight library to Guidedoc’s renderers. It is separate so that the highlight library itself depends on nothing but Odin’s core.

### Procedures

```
guidedoc.highlighter :: proc() -> gd.Highlighter
```

highlighter is the value for Render\_Options.highlighter: code blocks become the HTML Pygments writes for Sphinx, with its short CSS classes.

```
guidedoc.typst_highlighter :: proc() -> gd.Highlighter
```

typst\_highlighter is for a Typst renderer that draws code with Guidedog’s colours rather than Typst’s: it writes code as write\_typst’s array of lines, for the book template’s gd-highlight function (see the README). It returns false for plain text and languages it does not know, so the renderer can keep Typst’s own raw block for those.

## host

Package `host` composes the allocation-free core with memory, files, and retries. Everything here may allocate; the core's zero-allocation contract ends at `gd` calls.

Ownership, unless a procedure says otherwise: what a procedure returns (a string, a slice, a report, a problem's text) is allocated in `context allocator` and belongs to the caller, which frees it with that allocator's arena rather than piece by piece; its arguments are borrowed for the call only. A failure is a `Problem` (`exit ! = 0`) or a false result, never a panic. The surface is listed in `lib/host/README.md`.

### Types

**host.Arena** :: *struct*

Arena hands out memory from blocks it takes from a backing allocator (a `Meter`, usually) and frees them all at once. It is `core:mem`'s `Dynamic_Arena` made safe for a backing that refuses: a block the backing refuses leaves the arena as it was, and the request fails with `.Out_Of_Memory`, where `Dynamic_Arena` would go on handing out memory at address `0`. The blocks are linked through a header at their start, so the arena allocates nothing besides them; each block is used through bounds-checked slices of itself. It is not safe for use by several threads at once.

**backing:** `mem.Allocator`

**block:** `int`

bytes of a block; `0`: `ARENA_BLOCK_BYTES`.

**blocks:** `^Arena_Block`

the newest first; the first is the one being filled.

**last:** `[]byte`

the latest allocation from the first block, which can grow.

**bytes:** `int`

the bytes of every block held.

**host.Artifact\_Request** :: *struct*

`Artifact_Request` is one conversion of `guidedog convert`. Its strings and slices are borrowed for the call.

**input:** `string`

path, or `"-"` for standard input.

**from:** Format

**to:** Target

**output:** string

path, or “-” for standard output.

**emit\_typst:** string

optional path for the intermediate Typst source.

**force:** bool

**policy:** gd.Policy

**limits:** gd.Limits

**read:** gd.Read\_Options

**render:** gd.Render\_Options

**typst\_program:** string

Typst program name or path.

**typst\_options:** typst.Options

packages and fonts.

**roots:** []string

approved resource roots; the input’s directory by default.

**budget:** int

**allocator:** mem.Allocator

session regions; unset: context.allocator. Must reclaim frees.

**host.Artifact\_Result** :: struct

Artifact\_Result is what convert\_artifact did: exit 0 and the published path (or the bytes for standard output), or exit != 0 and the problem; reports either way. All of it is allocated in the caller’s context.allocator.



---

**exit:** int

---

**reports:** []Report

---

**problem:** Problem

---

**written:** string

the published path, or "" for standard output.

---

**stdout:** []u8

bytes to print when output is "-".

---

**host.Budget** :: struct

Budget bounds the memory sessions hold at once: one session's alone, or, shared, the total of a parallel build's threads. A session reserves what it will allocate before allocating it, and releases it when it frees it.

Units of work that run at once (the threads of a parallel read) join the budget while they run. A reservation the budget cannot hold now waits while another joined unit is still running, since that one will release what it holds when it finishes; when every other unit is waiting too, nothing would be released, and it is refused. So a reservation never waits for itself, and waiting units never wait for each other: a build whose units together need more than the budget refuses some of them.

A reservation the budget cannot hold, and no unit will make room for, goes to the budget's policy (Budget\_Policy): it is refused, unless the policy raises the limit or spills working memory to disk.

---

**limit:** int

---

**used:** int

---

**active:** int

units joined: see budget\_join.

---

**waiting:** int

joined units waiting in reserve.

---

**peak:** int

the most used at once.

**mutex:** `sync.Mutex`

**freed:** `sync.Cond`

**policy:** `Budget_Policy`

**stopped:** `bool`

the policy chose to stop: later refusals do not ask again.

**refused\_used:** `int`

What the budget held, and was asked for, when it last refused (budget\_hint).

**refused\_need:** `int`

**host.Budget\_Policy** *:: struct*

`Budget_Policy` is what a budget does when it cannot hold a reservation (`Budget`). With no `on_refusal`, and `spill` not set, the reservation is refused: the step that asked fails with `host.budget`. `on_refusal` can raise the limit, or turn on spilling; `spill` set from the start spills without asking. Spilling sends working memory past the limit (sessions' storage, and what Meters allocate) to disk: slower, but bounded by the Disk's own limit and free space. Other reservations, such as a file a build keeps in memory, are still refused.

**on\_refusal:** `On_Refusal`

**user:** `rawptr`

**disk:** `^Disk`

borrowed; the owner destroys it after the build is released.

**spill:** `bool`

**locate:** `proc(user: rawptr) -> Report`

`locate` says where the work that asked for memory is, for `Refusal.place`; it is called, with `locate_user`, from inside the allocator that asked.

**locate\_user:** `rawptr`

```
host.Confinement :: enum u8
```

Confinement is where a path lies with respect to the roots a project reads from.

#### Inside

confined: below a root once every link is followed.

#### Outside

the path as written is below no root.

#### Through\_Link

written below a root, but a link on the way leads out of the roots.

```
host.Disk :: struct
```

Disk is working memory on disk: memory-mapped temporary files, used when a build's budget is exhausted and the build may go on slowly rather than stop (Budget\_Policy). Each file is written out in full before it is mapped, so the disk space is taken before the memory is handed out and writing to it never meets a full disk. On Linux, macOS, and the BSDs a file is removed as soon as it is mapped, so the system reclaims it however the process ends, a crash or Ctrl+C included; on Windows it is created to be deleted when closed.

A Disk hands out memory from its files, as an arena does from its blocks, and counts the allocations each file holds: a file is unmapped when its last allocation is freed, or by disk\_destroy, and nothing reads it after. It is safe for use by several threads at once. Its zero value, with dir set, is ready to use; the owner calls disk\_destroy once nothing allocated from it is used any more.

```
dir: string
```

the folder of the temporary files; "" for the system's.

```
limit: int
```

the most bytes mapped at once; 0: the free space, less DISK\_MARGIN.

```
used: int
```

bytes mapped now.

```
peak: int
```

the most bytes mapped at once.

**refused:** int

bytes a refused request asked for, or 0.

**failure:** Disk\_Failure

why the latest request was refused.

**chunks:** ^Disk\_Chunk

the newest first; the first is the one being filled.

**mutex:** sync.Mutex

host.**Disk\_Failure** :: *enum* u8

Disk\_Failure is why a Disk refused a request.

**None**

**Limit**

the disk budget (Disk.limit) cannot hold it.

**Space**

the disk lacks the free space, less DISK\_MARGIN.

**System**

the system could not create, write, or map the file.

host.**Files** :: *struct*

Files provides included resources from approved roots. It refuses Docutils' "<standard>" includes, names that leave the roots, and names that pass through a symbolic link. Links are checked on the path as written, component by component with lstat, before it is resolved; then the resolved path must still lie in a root. The rule is therefore the same on Windows, macOS, and Linux.

**roots:** []string

**max\_bytes:** int

**project:** string

When set, a name starting with / is relative to this directory, as Sphinx reads /path in include and literalinclude; otherwise such a name is absolute.

**canonical:** []string

the roots, resolved; filled by files\_provider.

**ledger:** ^Ledger

When set, each file's size is charged here before it is read (see read\_file); a file the budget cannot hold is refused. The caller settles it when the files go.

host.**Folder\_Entry** :: struct

Folder\_Entry is a name in a folder, what it names (a symbolic link is .Symlink, whatever it leads to), and its inode, as the listing gives it: two names with the same inode on the same file system are the same file. inode is 0 where the system gives none.

**name:** string

**type:** os.File\_Type

**inode:** u64

host.**Format** :: enum u8

Format is the markup convert\_artifact reads.

**Rst**

**Commonmark**

**Typst**

host.**Guard** :: struct

Guard is an allocator that watches another for refusals. Guidedog's libraries do not all stop at a failed allocation: they go on with what they have, never touching memory they were refused (tests/allocation\_failure\_test.odin holds them to that), so their result may be incomplete. A caller that must know its

work is whole allocates through a Guard and asks `guard_problem` when the work ends.

The first refusal trips the guard, and from then on it refuses every allocation itself: memory that ran out once is not relied on again, so the work that follows only shrinks, and nothing is built from a mix of what was and was not allocated. Frees still reach the backing allocator. A guard may be shared by threads.

---

**backing:** `mem.Allocator`

---



---

**tripped:** `i32`

---

atomic: 1 once an allocation was refused.

---



---

**refused:** `int`

---

atomic: bytes the first refused allocation asked for.

---



---

**host.Identity** *:: struct*

---

Identity is a file's or a folder's device and inode, which a rename keeps. The zero value is none: the path does not exist, or the system gives no inode (Windows may not).

---

**device:** `u64`

---



---

**inode:** `u64`

---



---

**host.Ledger** *:: struct*

---

Ledger counts what one unit of work (reading a document, rendering a page, composing a book) holds of a budget besides its sessions: the files it reads, the graphs it draws. Each charge is made before the memory is allocated, and settle releases them all when the unit frees its memory. A ledger without a budget counts nothing.

---

**budget:** `^Budget`

---



---

**held:** `int`

---



---

**refused:** `int`

---

bytes a refused charge asked for, or 0.

```
host.Meter :: struct
```

Meter is an allocator that charges a ledger for each allocation before passing it to its backing allocator. It is for work whose memory is not known in advance, such as rendering a page's layout or parsing a saved environment. A charge the budget refuses is refused before anything is allocated: the allocation fails with `.Out_Of_Memory`, the ledger records the refused bytes (`ledger.refused`), and the step that asked stops with `metered_refusal`'s problem. Code behind a meter must therefore handle a failed allocation; see "Memory" in `lib/host/README.md` for what the build's steps do.

When the budget spills (`Budget_Policy`), an allocation it cannot hold comes from the policy's `Disk` instead, and is freed there. Freed memory from the backing allocator are not discharged one by one (an arena's allocator is told no sizes); `settle` returns the step's charges when it frees its memory.

`prepaid` holds bytes charged ahead (`meter_prepay`) that allocations use before charging anew.

```
backing: mem.Allocator
```

```
ledger: ^Ledger
```

```
prepaid: int
```

```
host.On_Refusal :: proc(user: rawptr, r: Refusal) ->
Refusal_Decision
```

`On_Refusal` decides what a budget does with a reservation it cannot hold. It is called at most once at a time, while the budget's other reservations wait; it must not reserve from the budget itself.

```
host.Problem :: struct
```

Problem is a host failure: files, the Typst backend, or command usage.

```
code: string
```

```
title: string
```

```
path: string
```

```
message: string
```

```
hint: string
```

```
exit: int
```

```
host.Refusal :: struct
```

Refusal is what a budget tells its policy when it cannot hold a reservation: the bytes asked for, what the budget holds, and its limit; and where the work that asked is, as the policy's locate says (a report's path, line, and excerpt; zero when unknown), so that a question names the place a report would.

```
need: int
```

```
used: int
```

```
limit: int
```

```
disk: bool
```

the policy has a Disk to go on with.

```
place: Report
```

```
host.Refusal_Choice :: enum u8
```

Refusal\_Choice is what a build does when its budget cannot hold a reservation: stop with the budget's problem, raise the budget, or go on with working memory on disk.

```
Stop
```

```
Raise
```

to Refusal\_Decision.limit.

```
Disk
```

```
host.Refusal_Decision :: struct
```

```
choice: Refusal_Choice
```

```
limit: int
```

Raise: the new limit in bytes, more than Refusal.limit.



```
host.Report :: struct
```

Report is a diagnostic with its template filled in and its position resolved.

```
schema: int
```

```
code: string
```

```
severity: string
```

```
title: string
```

```
message: string
```

```
hint: string
```

```
path: string
```

```
line: int
```

```
column: int
```

```
end_line: int
```

```
end_column: int
```

```
excerpt: string `json:"- "`
```

```
marker: string `json:"- "`
```

```
source: string
```

the complete source line, for JSON consumers; columns count runes.

```
category: string
```

The warning's category as "type.subtype", such as "ref.term", which a host may let users silence (Sphinx's `suppress_warnings`); "" for none.

```
host.Session :: struct
```

Session owns the storage of one conversion so reports stay valid while it lives. Its storage, output, and source map are reserved from shared when that is set (the threads of a parallel build share one), and otherwise from a budget of its own of limit bytes. allocator supplies its regions (context.allocator when unset); it must outlive the session and reclaim freed regions to bound actual memory across retries. A shared budget's policy may raise its limit; all reservations use that same limit.

```
allocator: mem.Allocator
```

```
ws: gd.Workspace
```

```
storage: gd.Storage
```

```
sizes: Sizes
```

```
output: []u8
```

```
result: gd.Result
```

```
source_map: []gd.Map_Entry
```

```
map_size: int
```

source map entries; 0 for none.

```
retries: int
```

```
ready: bool
```

```
limit: int
```

0: DEFAULT\_BUDGET.

```
shared: ^Budget
```

nil: own.

```
own: Budget
```

**held:** int

bytes reserved for the storage, output, and source map.

**inputs:** ^Kept\_File

owned input buffers; freed before releasing kept.

**kept:** int

bytes reserved for owned input buffers and their records.

**refused:** int

bytes a refused reservation asked for, or 0; max(int): too many.

**denied:** bool

the system refused an allocation of refused bytes the budget allowed.

**disk:** ^Disk

where the regions are when the budget spilled them; nil: memory.

**allocated:** mem.Allocator

the allocator that owns the current regions.

**host.Sizes** :: struct

Sizes are region capacities in elements (bytes for text and scratch).

**nodes:** int

**attributes:** int

**lists:** int

**links:** int

**cells:** int

**text:** int

**scratch:** int

```
sources: int
```

```
reports: int
```

```
output: int
```

```
second_bank: bool
```

needed only for passes.

```
host.Target :: enum u8
```

Target is the artifact convert\_artifact publishes.

```
Html
```

```
Pdf
```

### Procedures

```
host.arena_allocator :: proc(a: ^Arena) -> mem.Allocator
```

arena\_allocator returns an allocator that allocates from a; it borrows a. Free does nothing; Free\_All frees every block.

```
host.arena_destroy :: proc(a: ^Arena)
```

arena\_destroy frees every block of a, which is empty after and may be used again. Nothing allocated from a may be used after it.

```
host.arena_init :: proc(a: ^Arena, backing: mem.Allocator,
block := 0)
```

arena\_init starts an empty arena over backing, which must outlive it, taking blocks of block bytes (0: ARENA\_BLOCK\_BYTES; tests use small ones to reach every allocation).

```
host.available_memory :: proc() -> (bytes: int, ok: bool)
```

available\_memory is the memory the system could give the process now without swapping, as it reports it: MemAvailable on Linux, lowered to what the process's cgroup still allows, the available physical memory on Windows, and the free and inactive pages on macOS; or GUIDEDOG\_AVAILABLE\_MIB when it is set to a number. ok is false when it cannot say. It allocates nothing.

---

```
host.backend_problem :: proc(program: string) -> Problem
```

---

backend\_problem is the problem of a Typst program that cannot be found; program is borrowed into it.

---

```
host.budget_advice :: proc(used, need, limit, available: int,
disk: bool, other: string, available_known := false) ->
string
```

---

budget\_advice is the hint budget\_hint gives, for a step that asked for need bytes while the budget held used of its limit, on a machine with available bytes free (0 when unknown unless available\_known is true). It names the budget to build with (suggest\_budget); when that is more than the free memory, it says so, and names keeping working memory on disk first, when disk is there to offer. It is allocated.

---

```
host.budget_hint :: proc(b: ^Budget, need: int, other:
string) -> string
```

---

budget\_hint is the hint of a budget's problem (budget\_advice): the budget the refused step fits in, from what b held and was asked for when it last refused (need, the bytes the step needed, when b refused nothing itself), capped by the memory the machine has free; keeping working memory on disk, when the caller gave b a Disk; and the other way out the caller names. It is allocated.

---

```
host.budget_join :: proc(b: ^Budget)
```

---

budget\_join counts a unit of work that runs at once with others until budget\_leave.

---

```
host.budget_leave :: proc(b: ^Budget)
```

---

budget\_leave ends a unit budget\_join counted, waking reservations that waited for it.

---

```
host.budget_limit :: proc(b: ^Budget) -> int
```

---

budget\_limit is b's limit now: on\_refusal may have raised it since it was set.

---

```
host.budget_problem :: proc(what, path: string, bytes: int,
b: ^Budget) -> Problem
```

---

budget\_problem explains a charge the budget refused: what needed the bytes, and where. what and path are borrowed into it; the message is allocated.

---

```
host.budget_report :: proc(s: ^Session) -> (Report, bool)
```

---

budget\_report explains a session that stopped at its budget, or at an allocation the system refused, or returns false. The message is allocated.

```
host.cancel_probe :: proc() -> gd.Cancel_Probe
```

`cancel_probe` returns a probe with which the core polls the Ctrl+C flag (`catch_interrupts`) during long conversions; it holds no state of its own.

```
host.canonical_roots :: proc(roots: []string) -> []string
```

`canonical_roots` resolves roots to absolute paths, as `confined` compares paths with them; a root that cannot be resolved is kept as written. The slice is allocated.

```
host.catch_interrupts :: proc()
```

`catch_interrupts` makes Ctrl+C (SIGINT) set a flag instead of ending the process, for the whole process: `was_interrupted` reads it, and `cancel_probe` gives it to the core. A program calls it once, at the start.

```
host.charge :: proc(l: ^Ledger, bytes: int) -> bool
```

`charge` reserves bytes for the unit, waiting as `Budget` describes, or returns false, reserving nothing and recording the refused bytes (`l.refused`). A nil ledger, or one without a budget, accepts every charge.

```
host.confined :: proc(roots: []string, path: string) -> bool
```

`confined` reports whether a file a project reads (a document, an image, a template, a static file) lies in one of the canonical roots once every link on its way is followed. Where the system resolves paths (POSIX) the resolved path decides; on Windows, where it does not, a path through a link is refused.

```
host.confinement :: proc(roots: []string, path: string) -
> Confinement
```

`confinement` tells whether `confined` accepts a path, and why not when it does not, so a refusal can say whether the path itself or a link on its way leads outside. A refused path has a link leading out when one of its folders is itself confined: the path is written below a readable folder, and something below that folder leads away.

```
host.containing_root :: proc(roots: []string, path: string) -
> (string, bool)
```

`containing_root` returns the root of roots that path is written below, or false. It compares the paths as written (case-insensitively on Windows) and follows no link; `confined` is the check that does.

```
host.convert_artifact :: proc(registry: gd.Registry, req:
Artifact_Request) -> Artifact_Result
```

convert\_artifact is the host route from a file to a published HTML or PDF artifact: read the input, convert it in a session of req.budget bytes, run Typst for PDF, and publish the output atomically (publish), or return it in stdout for “-“. Guidedoc conversion inside it is allocation-free; reading, Typst, and publishing are not. The result, its reports, and stdout are allocated in context.allocator and owned by the caller; the session is destroyed before it returns. A failure sets exit and problem and leaves any existing output unchanged.

```
host.crash_on_assertion :: proc(prefix, message: string, loc:
runtime.Source_Code_Location) -> !
```

crash\_on\_assertion reports a panic or a failed assertion in Odin code, with its message and where it was raised, and exits with CRASH\_EXIT.

```
host.default_stack_budget :: proc(stack: int) -> int
```

default\_stack\_budget is the budget to use when none is asked for: the default, or less on a thread too small for it.

```
host.describe :: proc(d: gd.Diagnostic, sources:
[]gd.Source_Entry) -> Report
```

describe fills a diagnostic’s templates and resolves its span against sources into a Report: path, line, column, excerpt and marker. It borrows d and sources for the call; the report’s strings are allocated and do not refer to the sources afterwards.

```
host.describe_all :: proc(s: ^Session) -> []Report
```

describe\_all describes what the session’s conversion reported, and the budget it reached if any (budget\_report). The reports are allocated and do not borrow the session, so they outlive session\_destroy.

```
host.destroy_storage :: proc(st: gd.Storage, allocator :=
context.allocator)
```

destroy\_storage frees every region of storage make\_storage allocated with allocator; a region left nil is skipped, so it also frees a partly made storage.

```
host.discharge :: proc(l: ^Ledger, bytes: int)
```

discharge returns bytes the unit has freed before it ends.

---

```
host.disk_allocator :: proc(d: ^Disk) -> mem.Allocator
```

---

disk\_allocator returns an allocator that allocates from d's files; it borrows d. Free returns an allocation, and unmaps its file once the file holds none; Free\_All is not supported (disk\_destroy unmaps everything).

---

```
host.disk_destroy :: proc(d: ^Disk)
```

---

disk\_destroy unmaps every file of d, freed or not; nothing allocated from d may be used after it. d may be used again.

---

```
host.disk_owns :: proc(d: ^Disk, p: rawptr) -> bool
```

---

disk\_owns reports whether p lies in one of d's files.

---

```
host.display_path :: proc(name: string) -> string
```

---

display\_path returns name relative to the working directory when that is shorter to read (not above it by three folders or more), and name itself otherwise or for "" and "-". The result may borrow name; a relative one is allocated.

---

```
host.dump :: proc(s: ^Session, doc: gd.Document) -> string
```

---

dump returns the tree outline of a successfully read document, allocated, or "" when the document is no longer valid; tests use it.

---

```
host.each_chunk :: proc(path: string, user: rawptr, each:
proc(user: rawptr, chunk: [ju8) -> bool) -> bool
```

---

each\_chunk reads a file through a CHUNK\_BYTES buffer, giving each chunk to each until it returns false. It returns whether the whole file was read and accepted.

---

```
host.excerpt_at :: proc(r: ^Report, text: string, line,
column: int)
```

---

excerpt\_at fills a report's excerpt and marker at a 1-based line and Unicode scalar column, for positions in sources Guidedoc generated, such as a composed book.

---

```
host.exchange :: proc(a, b: string) -> (swapped: bool, err:
os.Error)
```

---

可用平台: Windows, Darwin, Linux, FreeBSD, OpenBSD, NetBSD.

exchange: see switch.odin. macOS swaps with rename\_np and RENAME\_SWAP. It allocates nothing (fixed.odin).



```
host.exit_code :: proc(status: gd.Status) -> int
```

exit\_code maps a conversion status to the process exit status Guidedog's commands use: 0 ok, 1 invalid input or policy, 3 capacity or limits, 5 internal, 130 cancelled.

```
host.files_provider :: proc(f: ^Files) ->
gd.Resource_Provider
```

files\_provider resolves f.roots and returns a gd.Resource\_Provider that serves includes through f, which it borrows: f and its roots must outlive every conversion the provider is given to. Each file it serves is allocated in the context.allocator of the conversion's caller and charged to f.ledger when that is set.

```
host.fitting_budget :: proc(available: int) -> int
```

fitting\_budget is a budget that fits in the free memory: nine tenths of it, in steps of 16 MiB; 0 when there is no room for even that minimum. Callers must not offer a zero budget to the CLI, where it means the default budget.

```
host.folders_of :: proc(paths: []string, top: string) ->
[]string
```

folders\_of lists, once each and in order, the folders that hold paths and the folders between them and top (top included): the folders whose names a publication adds or changes. A path not below top gives only its own folder.

```
host.grow_session :: proc(s: ^Session, need:
gd.Capacity_Need) -> bool
```

grow\_session enlarges the region need names, for the next prepare\_session; false when the enlarged sizes cannot be represented, which counts as the budget reached.

```
host.guard_allocator :: proc(g: ^Guard) -> mem.Allocator
```

guard\_allocator returns an allocator that allocates from g.backing until it refuses once, and then refuses everything (see Guard); it borrows g, which must outlive every allocation made through it.

```
host.guard_problem :: proc(g: ^Guard, what: string) ->
(Problem, bool)
```

guard\_problem is host.memory (exit 3) when an allocation through g was refused, with what, a static sentence naming the work, as its message; or false. Its text is static, since the memory to write more has run out.

```
host.guard_tripped :: proc(g: ^Guard) -> bool
```

guard\_tripped tells whether an allocation through g was refused. A caller may poll it to stop long work early; guard\_problem explains it.

```
host.identity :: proc(path: string) -> (id: Identity, err: os.Error)
```

identity is the identity of what is at path; {} with no error when nothing is. It allocates nothing.

```
host.install_crash_reporter :: proc(args: []string, version: string)
```

install\_crash\_reporter prepares the crash report for the command line args (the program name first) and version, what `guidedog --version` prints (one or more lines), and installs the handlers that print it: for faults and traps: signals on POSIX, an unhandled-exception filter on Windows. It allocates nothing. The caller also sets `context.assertion_failure_proc` to `crash_on_assertion`, so a panic or failed assertion is reported with its message.

```
host.interrupted_problem :: proc() -> Problem
```

interrupted\_problem is the problem, with exit status 130, of a command stopped by Ctrl+C.

```
host.io_problem :: proc(path: string, err: os.Error) -> Problem
```

io\_problem explains an operating-system error on path; path is borrowed into it.

```
host.is_stage_name :: proc(name: string) -> bool
```

is\_stage\_name is whether name is one stage\_file gives a temporary file: STAGE\_PREFIX, STAGE\_DIGITS digits, then nothing or one extension. Any other name, however close, is not Guidedog's.

```
host.link_file :: proc(from, to: string, type: os.File_Type) -> (copied: bool, err: os.Error)
```

link\_file makes to name the same file as from: a hard link, or, where the file system cannot link it (FAT, some network shares), a copy, which copied reports and the caller must flush. A symbolic link is made again, to the same target. An existing to is an error, never replaced. A regular file is linked or copied without allocating (fixed.odin).

```
host.list_folder :: proc(dir: string, allocator :=
context.allocator) -> (entries: []Folder_Entry, err:
os.Error)
```

**可用平台:** Windows, Darwin, Linux, FreeBSD, OpenBSD, NetBSD.

`list_folder` lists a folder's entries, "." and ".." left out, in the order the system gives them, with each entry's type from the listing itself (`d_type`): unlike `os.read_all_directory_by_path` it opens no entry, which matters for a folder of thousands of files. An entry whose type the file system does not give is read with `lstat`. The names are allocated in `allocator`; an allocation that fails is an error.

```
host.main_stack_bytes :: proc() -> int
```

**可用平台:** Windows, Darwin, Linux, FreeBSD, OpenBSD, NetBSD.

`main_stack_bytes` is the stack of the main thread.

```
host.make_storage :: proc(s: Sizes, allocator :=
context.allocator) -> (st: gd.Storage, ok: bool) #optional_ok
```

`make_storage` allocates storage of the sizes with `allocator`; the caller owns it and frees it with `destroy_storage`. `ok` is false, and nothing is kept allocated, when any region cannot be allocated: a region is never left nil. It reserves nothing from a Budget; a Session does that before calling it.

```
host.meter_allocator :: proc(m: ^Meter) -> mem.Allocator
```

`meter_allocator` returns an allocator that charges `m.ledger` and allocates from `m.backing`; it borrows `m`, which must outlive every allocation made through it.

```
host.meter_prepay :: proc(m: ^Meter, bytes: int) -> bool
```

`meter_prepay` charges bytes ahead, for work whose memory can be bounded before it starts: the work is refused at once when the budget cannot hold the bound, before any of it is done. Allocations then use the prepaid bytes before charging more; `meter_refund` returns what they did not use.

```
host.meter_refund :: proc(m: ^Meter)
```

`meter_refund` returns the prepaid bytes no allocation used to the budget.

```
host.metered_refusal :: proc(l: ^Ledger, what, path: string)
-> (Problem, bool)
```

`metered_refusal` reports a charge a Meter's ledger could not make: what the step needed, as `budget_problem` explains it.

```
host.out_of_memory :: proc(what: string) -> Problem
```

out\_of\_memory is host.memory with what, a static sentence naming the work that could not get memory, as its message; nothing in it is allocated.

```
host.output_text :: proc(s: ^Session) -> string
```

output\_text is what the session's last successful run rendered. It borrows the session's output buffer: it is valid until the next run, prepare\_session, resize\_output, or session\_destroy.

```
host.path_type :: proc(path: string) -> (type: os.File_Type,
found: bool, err: os.Error)
```

path\_type is what is at path, not following a symbolic link; found is false, with no error, when nothing is. It allocates nothing.

```
host.plan_session :: proc(s: ^Session, input_bytes: int,
second_bank: bool) -> bool
```

plan\_session sizes the session for an input of input\_bytes; false when the sizes cannot be represented, which counts as the budget reached.

```
host.prepare_session :: proc(s: ^Session) -> bool
```

prepare\_session allocates storage of the session's current sizes, releasing any older storage first. It reserves the bytes from the session's budget before allocating, and returns false, allocating nothing, when the budget cannot hold them. run and read call it; callers that load objects call it directly.

```
host.publish :: proc(path: string, data: []u8, force: bool) -
> Problem
```

publish writes bytes through a sibling temporary file, flushes it, and renames it over the destination, making the folders on its way: path holds its old bytes or the new ones, never a part. An existing destination is refused (host.output.exists) unless force is set; any failure leaves the destination unchanged and returns its Problem.

```
host.read :: proc(s: ^Session, reader: gd.Reader, request:
gd.Request, budget := DEFAULT_BUDGET) -> gd.Read_Result
```

read parses without rendering, then runs request.passes (a translation, for example) over the document; storage grows and the whole read retries when a region fills. The document and reports borrow the session, as run's do, and the source text.

```
host.read_file :: proc(path: string, ledger: ^Ledger,
limit := MAX_SOURCE_BYTES, allocator := context.allocator) ->
([]u8, Problem)
```

`read_file` reads a regular file of at most `limit` bytes whole, charging its size to `ledger` before allocating; the caller discharges the bytes when it frees them. A file that grows while it is read is refused. If it shrinks, its original allocation stays charged until freed: fewer bytes read do not mean fewer bytes held.

```
host.read_kept :: proc(s: ^Session, path: string, limit: int)
-> ([]u8, Problem)
```

`read_kept` reads a file whose bytes the caller keeps while the session's document refers to them. The bytes belong to the caller, allocated in `context.allocator`; the caller frees them no later than `session_destroy`, which returns their budget charge. Use `read_owned` when the session should own and free the bytes itself.

```
host.read_owned :: proc(s: ^Session, path: string, limit:
int) -> ([]u8, Problem)
```

`read_owned` reads a file the session's document refers to while it lives, such as its cached object. The session owns the returned bytes until `session_destroy`, which frees them before releasing their charge. Its allocator supplies both the bytes and their ownership record; each allocation is charged before it is made.

```
host.read_source :: proc(path: string, limit :=
MAX_SOURCE_BYTES) -> (text, canonical: string, p: Problem)
```

`read_source` reads a named input, or standard input for “-“, within `limit` bytes. It returns the text and the absolute path used as the source's canonical name.

```
host.remove_path :: proc(path: string) -> os.Error
```

**可用平台:** Windows, Darwin, Linux, FreeBSD, OpenBSD, NetBSD.

`remove_path` removes a file, a link, or an empty folder; see `fixed.odin`.

```
host.remove_stages :: proc(dir: string)
```

`remove_stages` removes, in `dir` and every folder below it, the temporary files that a process stopped (killed, or the machine lost power) left before renaming or removing them: files, never folders or links, whose names `is_stage_name` accepts. Every other name is left as it is. A folder it cannot list, or has no memory to name, is left as it is: a stage file is clutter that nothing reads. The caller makes sure no stage file below `dir` is still in use.

```
host.rename_path :: proc(from, to: string) -> os.Error
```

可用平台: Windows, Darwin, Linux, FreeBSD, OpenBSD, NetBSD.  
 rename\_path renames from to to, replacing a file at to; see fixed.odin.

```
host.render_json :: proc(r: Report) -> string
```

render\_json is a report as one JSON object (schema DIAGNOSTICS\_SCHEMA), without its terminal excerpt and marker; “{}” when it cannot be encoded. The text is allocated.

```
host.render_problem_json :: proc(p: Problem) -> string
```

render\_problem\_json is a host problem as a JSON report of severity “error”.

```
host.render_problem_text :: proc(p: Problem, color: bool) -> string
```

render\_problem\_text lays out a host problem as render\_text lays out an error report.

```
host.render_text :: proc(report: Report, consequence: string, color: bool) -> string
```

render\_text lays out a report the Elm way: what happened, where, what it means for the result, and what to do next. Color supplements words; it never replaces them.

```
host.resize_output :: proc(s: ^Session, bytes: int) -> bool
```

resize\_output replaces the session’s output buffer with one of bytes, reserving the whole new buffer while the old one still lives. It returns false, keeping the old buffer, when the budget cannot hold the new one.

```
host.resize_source_map :: proc(s: ^Session, entries: int) -> bool
```

resize\_source\_map gives the session a source map of entries, reserved as resize\_output reserves its buffer.

```
host.run :: proc(s: ^Session, registry: gd.Registry, request: gd.Request, budget := DEFAULT_BUDGET, map_source := false) -> gd.Result
```

run converts with storage sized for the input, growing the exhausted region and retrying until it fits or the budget (bytes; see Session) would be exceeded, which ends it with status Limit and a report that says so (budget\_report). With map\_source set, the session also collects a source map from the renderer. The session owns everything the result refers to: its reports and output\_text are

valid until the next run or read on `s`, or `session_destroy`. request is borrowed for the call, and its source text for as long as the reports are read.

```
host.run_workers :: proc(count: int, data: ^$T, work:
proc(data: ^T))
```

`run_workers` runs `work(data)` on `count` threads at once and returns when every one has ended; work shares data between them, and takes its share of the work itself (an index taken atomically, for one). A thread that cannot start is left out; when none can, work runs once on the calling thread, so the work is always done. It allocates the threads with `context allocator`, which the workers may share (`read_pending` gives them the build's, through a mutex this procedure does not take): so every thread is made before any starts, and freed only once every one has ended, and nothing is allocated while a worker runs. A refusal starts fewer.

```
host.same_contents :: proc(path: string, data: []u8) -> bool
```

`same_contents` reports whether the file at `path` holds exactly data, reading it in chunks and stopping at the first difference.

```
host.session_destroy :: proc(s: ^Session)
```

`session_destroy` frees the session's storage, output, source map, and kept inputs, and releases what it reserved and kept from its budget. Reports, `output_text`, and documents the session returned are invalid after it. The session may be run again.

```
host.settle :: proc(l: ^Ledger)
```

`settle` releases everything `l` holds of its budget; the unit calls it when it frees the memory it charged. A nil ledger or one without a budget is a no-op.

```
host.sizes_for :: proc(input_bytes: int) -> (s: Sizes, ok:
bool) #optional_ok
```

`sizes_for` estimates capacities from the input length. The estimate need not be tight: a capacity result names the region, and the session grows it and retries. `ok` is false when a capacity cannot be represented, which a session reports as its budget reached.

```
host.spill_disk :: proc(b: ^Budget) -> ^Disk
```

`spill_disk` is the Disk that working memory `b` cannot hold goes to, or nil when `b` does not spill.

```
host.stack_budget_limit :: proc(stack: int) -> int
```

`stack_budget_limit` is the largest stack budget a thread with stack bytes can hold: all but `STACK_HEADROOM`, and never more than half the stack. An 8 MiB thread (the main thread's and `core:thread`'s usual stack on Linux and macOS) holds 6 MiB; a 1 MiB one (Windows) holds `core.DEFAULT_STACK_BYTES`, which `stack.odin` sizes to fit it.

```
host.suggest_budget :: proc(used, need, limit, available:
int, available_known := false) -> (budget: int, fits: bool)
```

`suggest_budget` is the budget, in bytes, to suggest for a step that asked for need bytes while the budget held used of its limit, so that one change is enough: what is held, and room for the request `STEP_COPIES` times over (a step that builds a value of that size also holds its output and the copy it writes), or twice the old limit, whichever is larger, in steps of 64 MiB. When the machine has less memory free (available; 0 when unknown), the budget is capped to what fits there (`fitting_budget`), or, when what is needed does not fit, is what is needed, in steps of 16 MiB: then `fits` is false. `available_known` distinguishes measured zero from the default, unknown zero.

```
host.sync_all :: proc(paths: []string) -> Problem
```

`sync_all` flushes files and folders to storage, `SYNC_WORKERS` at a time, and returns the first that failed. A folder is flushed so the names in it are durable. Windows keeps its folders' names durable itself and flushes no folder, so they are skipped there.

```
host.sync_each :: proc(paths: []string) -> Problem
```

`sync_each` flushes each path in turn on the calling thread, as `sync_all` does with threads, and returns the first that failed; it allocates nothing.

```
host.thread_stack_bytes :: proc() -> int
```

可用平台: Windows, Darwin, Linux, FreeBSD, OpenBSD, NetBSD.  
`thread_stack_bytes` is the stack of a thread `core:thread` starts.

```
host.was_interrupted :: proc() -> bool
```

`was_interrupted` reports whether Ctrl+C was pressed since `catch_interrupts`.

```
host.write_unsynced :: proc(path: string, data: []u8) ->
Problem
```

`write_unsynced` writes bytes as `publish` does, through a sibling temporary file renamed over `path`, so `path` holds either its old bytes or the new ones, but does not flush them: a caller that needs them durable syncs them later (`sync_all`).



*Constants*


---

```
host.ARENA_BLOCK_BYTES :: 64 << 10
```

---

ARENA\_BLOCK\_BYTES is the size of the blocks an Arena takes from its backing allocator; a request larger than a quarter of it gets a block of its own.

---

```
host.AVAILABLE_OVERRIDE :: "GUIDEDOG_AVAILABLE_MIB"
```

---

AVAILABLE\_OVERRIDE names the environment variable that states the memory available, in MiB, in place of what the system reports: for machines whose report misleads, and for tests that need the same advice on every machine.

---

```
host.CRASH_EXIT :: 70
```

---

CRASH\_EXIT is the exit status of a command that crashed: EX\_SOFTWARE, an internal software error, which no other failure uses.

---

```
host.DEFAULT_BUDGET :: 256 << 20
```

---

DEFAULT\_BUDGET is a session's budget in bytes when none is given (256 MiB).

---

```
host.DIAGNOSTICS_SCHEMA :: 1
```

---

DIAGNOSTICS\_SCHEMA is the version of the JSON form of Report (render\_json), which a consumer checks before reading the fields; it changes only when a field does.

---

```
host.DISK_CHUNK_BYTES :: 64 << 20
```

---

DISK\_CHUNK\_BYTES is the size of the temporary files a Disk maps; a larger request gets a file of its own size.

---

```
host.DISK_MARGIN :: 256 << 20
```

---

DISK\_MARGIN is the free space a Disk leaves on the disk it maps files from, so that spilling working memory never fills the disk the rest of the system writes to.

---

```
host.ISSUE_TRACKER :: #config(GUIDEDOG_ISSUE_TRACKER, "")
```

---

ISSUE\_TRACKER is where a crash is to be reported: the address of the issue tracker of whoever distributes the build, set when it is built (-define:GUIDEDOG\_ISSUE\_TRACKER=https://...). Without it, the report asks the user to send the details to whoever provided their build, and names no address.

```
host.MAX_RETRIES :: 64
```

MAX\_RETRIES only guards against a bug: the budget is what bounds a session. Each retry at least doubles the region that filled, so growth reaches the budget in a few dozen steps however the regions fill in turn, as a document made of includes fills them.

```
host.MAX_SOURCE_BYTES :: 32 << 20
```

MAX\_SOURCE\_BYTES is the default size limit of one input file (32 MiB).

```
host.MEMORY_HINT
```

MEMORY\_HINT is what to do when the system refuses memory.

```
host.PATH_BYTES :: 4096
```

PATH\_BYTES is the longest path, with its terminator, these operations take: the longest that Linux takes (PATH\_MAX), and four times macOS's.

```
host.STAGE_PREFIX :: ".guidedog-stage-"
```

STAGE\_PREFIX begins the name of every temporary file stage\_file makes. It is followed by the STAGE\_DIGITS digits os.create\_temp\_file draws, then the destination's extension.

### *Foreign imports*

```
foreign import host.kernel32 "system:Kernel32.lib"
```

可用平台: Windows.

```
foreign import host.system "system:System"
```

可用平台: Darwin.

## **i18n**

Package i18n translates Guidedoc documents the way Sphinx does with gettext: it extracts every translatable message of a document into a POT template (Sphinx's gettext builder), and it replaces each message with its translation from a catalog (Sphinx's Locale transform), re-parsing the translation as inline markup of the document's own language. It imports the Guidedoc core and the gettext library only; the reader that parses translations is passed in (gd.Reader.read\_inline).

## Types

```
il18n.Compact :: struct
```

Compact is gettext\_compact: per top-level directory (true, Sphinx’s default), per document (false), or one catalog of the given name.

```
mode: enum u8 {Per_Directory, Per_Document, Single}
```

```
name: string
```

for Single.

```
il18n.Extract_Options :: struct
```

Extract\_Options are the gettext builder’s settings for one document.

```
targets: Targets
```

gettext\_additional\_targets.

```
location: bool
```

gettext\_location: “#: path:line” for every occurrence.

```
uuid: bool
```

gettext\_uuid: a “#: uuid” line for every occurrence.

```
path: string
```

how locations name the main source, e.g. “../index.rst”.

```
docname: string
```

the document’s name, e.g. “guide/usage”; seeds the uuids.

```
base: string
```

included files are named relative to this directory, if set.

```
il18n.Image_Exists :: struct
```

Image\_Exists reports whether an image path, as a document would write it, names a file. The host resolves it against the document’s directory.

```
procedure: proc(user: rawptr, docname, path: string) -
> bool
```

---

**user:** rawptr

---



---

**i18n.Image\_Localization** :: *struct*

---

Image\_Localization is figure\_language\_filename for the translation pass.

---

**format:** string

---

“” is DEFAULT\_FIGURE\_FILENAME.

---

**docname:** string

---



---

**language:** string

---



---

**exists:** Image\_Exists

---



---

**i18n.Index\_Part** :: *struct*

---

Index\_Part is one translatable piece of an index entry line, e.g. “b” of “single: a; b”. first and length locate it in the line.

---

**text:** string

---



---

**first:** int

---



---

**length:** int

---



---

**i18n.Pot\_Header** :: *struct*

---

Pot\_Header holds the values of a template’s header, as Sphinx fills them.

---

**project:** string

---



---

**version:** string

---



---

**copyright:** string

---



---

**last\_translator:** string

---

gettext\_last\_translator.

---

**language\_team:** string

---

gettext\_language\_team.

```
creation_date: string
```

e.g. “2026-09-29 12:00+0000”.

```
i18n.Progress :: enum u8
```

Progress is translation\_progress\_classes: which translatable nodes get the class “translated” or “untranslated”, so a stylesheet can show what is left to translate.

```
Translated
```

```
Untranslated
```

```
i18n.Progress_Classes :: bit_set[Progress; u8]
```

```
i18n.Target :: enum u8
```

Target names the extra kinds of content gettext\_additional\_targets makes translatable.

```
Index
```

“index”: index entries.

```
Literal_Block
```

“literal-block”: literal and code blocks, parsed literals.

```
Doctest_Block
```

“doctest-block”

```
Raw
```

“raw”

```
Image
```

“image”: image alt text.

```
i18n.Targets :: bit_set[Target; u8]
```

```
i18n.Template :: struct
```

Template collects the messages of one or more documents into a POT catalog, as Sphinx’s gettext builder does: a message found again gains a

location instead of a second entry, and messages keep the order of their first occurrence.

**catalog:** gettext.Catalog

**seen:** map[string]int

msgid -> message index.

**uuids:** map[string]int

uuids per message, for their positions.

**il8n.Translation** :: struct

Translation configures the translation pass, Sphinx's Locale transform: every translatable node whose message the catalog translates gets the translation. Inline content is re-parsed with `read_inline`, in the markup the document was written in (read carries the reader's settings, e.g. a `sphinx.Config`), and keeps the original's references; titles keep their anchors. The pass allocates nothing: it takes scratch memory, and registers the catalog's translation pool as one source, so nodes parsed from a translation have exact spans and diagnostics quote the translation.

**catalog:** ^gettext.Catalog

**read\_inline:** gd.Read\_Inline\_Proc

**read:** gd.Read\_Options

**policy:** gd.Policy

**limits:** gd.Limits

zero: `gd.default_limits()`.

**targets:** Targets

`gettext_additional_targets`.

**source\_name:** string

names the translations in diagnostics, e.g. "de/index.po".

**progress:** Progress\_Classes

`translation_progress_classes`.

**images:** Image\_Localization

figure\_language\_filename; applies without a catalog.

**translated:** int

Counts of the last run.  
messages replaced.

**kept:** int

translations refused (bad markup or references); warned.

**i18n.UI\_Problem** :: *struct*

UI\_Problem is a project catalog that could not be read.

**path:** string

**error:** gettext.Error

**i18n.Unit** :: *enum* u8

Unit says how a node carries a message.

**None**

**Inline**

inline content: the source text with its markup; translations re-parse.

**Literal**

the node's text, verbatim: code, doctest, raw, and math blocks.

**Parsed**

a parsed literal: inline content, only with the literal-block target.

**Alt**

an image's Alt attribute.

**Graphviz**

diagram alt text and, when requested, its literal source.

**Index**

each part of each index entry.

**Caption**

a toctree’s caption attribute.

**Entry**

a toctree entry’s explicit title.

**Meta**

the values of a meta directive.

**Contents**

a contents directive’s title.

***Procedures***

```
i18n.add :: proc(t: ^Template, msgid, location, uuid: string)
```

add adds one occurrence of a message: its location (“path:line”, or “”) and uuid. The template copies what it keeps, with its own allocator, so the arguments are borrowed.

```
i18n.destroy_template :: proc(t: ^Template)
```

destroy\_template frees a template and everything added to it, and clears it.

```
i18n.domain_for :: proc(docname: string, compact: Compact) -
> string
```

domain\_for returns the catalog (text domain) a document’s messages belong to, as Sphinx’s docname\_to\_domain does: “guide/usage” is “guide” when compact, else “guide/usage”; “index” is “index” either way.

```
i18n.embedded_catalog :: proc(language: string) -> (string,
bool)
```

embedded\_catalog returns the built-in catalog text for a language, trying it, then its base language (pt\_BR, then pt). The text is static, borrowed from the binary; the candidates are made in the temp allocator.



```
i18n.extract :: proc(t: ^Template, v: gd.View, options:
Extract_Options) -> int
```

`extract` adds every translatable message of a document to the template and returns how many occurrences it found. The template copies the messages with its allocator; the view is borrowed, and scratch strings go to the temp allocator.

```
i18n.figure_filename :: proc(format, filename, docname,
language: string, allocator := context.allocator) -> (string,
bool, string)
```

`figure_filename` is `write_figure_filename` into a new string allocated with allocator, which the caller owns; on failure the third result explains the format's problem, in the temp allocator.

```
i18n.figure_for_language :: proc(format, filename, docname,
language: string, exists: Image_Exists, allocator :=
context.allocator) -> string
```

`figure_for_language` returns the localized image when it exists, else the image as written, as Sphinx's `search_image_for_language` does; with no exists.procedure, no localized image exists. Either way the path is a new string allocated with allocator, which the caller owns.

```
i18n.fill_labels :: proc(tr: ^gettext.Translator, labels:
^gd.Labels)
```

`fill_labels` translates every renderer label into labels, for `Render_Options.labels`.

```
i18n.format_date :: proc(format: string, moment:
datetime.DateTime, language: string, allocator :=
context.allocator) -> string
```

`format_date` formats a moment with a strftime format, as Sphinx's `format_date` does, with the month and day names of language. The text is allocated with allocator, and the caller owns it.

```
i18n.index_parts :: proc(line: string, parts: []Index_Part) -
> int
```

`index_parts` splits an index entry line into its translatable pieces, as Sphinx's `split_index_msg` does: the entry type and "!" main marker are kept out of them.

```
i18n.make_template :: proc(name := "", allocator :=
context.allocator) -> Template
```

`make_template` starts an empty POT catalog named `name`. It and everything added to it are allocated with `allocator`; the caller frees them with `destroy_template`.

```
i18n.parse_target :: proc(name: string) -> (Target, bool)
```

`parse_target` returns the target a `gettext_additional_targets` name stands for.

```
i18n.set_header :: proc(t: ^Template, h: Pot_Header)
```

`set_header` gives the template the header Sphinx's `gettext` builder writes, copied with the template's allocator.

```
i18n.translation_pass :: proc(t: ^Translation) -> gd.Pass
```

`translation_pass` returns the pass; `t` must outlive the transform. The workspace keeps views of the catalog's pool and of `source_name` (spans, and reports quoting them), so the catalog and `source_name` must outlive every use of the document and its reports.

```
i18n.ui_text :: proc(tr: ^gettext.Translator, msgid: string)
-> string
```

`ui_text` translates one of Guidedog's words.

```
i18n.ui_translator :: proc(language: string, dirs: []string,
allow_fuzzy := false, allocator := context.allocator) ->
(gettext.Translator, []UI_Problem)
```

`ui_translator` makes the translator of Guidedog's own words for a language. `dirs` are the project's locale directories (absolute). A catalog that fails to parse is left out and reported in problems; the others still apply. The translator and its catalogs are allocated with `allocator`, and the caller owns them (`gettext.destroy_translator`); the problems too.

```
i18n.unit_of :: proc(v: gd.View, id: gd.Node_Id, targets:
Targets) -> Unit
```

`unit_of` classifies a node as Sphinx's `is_translatable` does, with the extra targets. The switch is exhaustive, so a new kind does not compile until it says whether it is translated.

```
i18n.write_figure_filename :: proc(out: ^gd.Buffer, format,
filename, docname, language: string) -> (ok: bool, bad_field:
string)
```

`write_figure_filename` fills a `figure_language_filename` format for an image path as written in a document, as Sphinx’s `get_image_filename_for_language` does:

- `{root}`: the path without its extension (“img/logo” of “img/logo.png”);
- `{path}`: its directory with a trailing slash, or “” (“img/”);
- `{basename}`: the file name without directory or extension (“logo”);
- `{ext}`: the extension with its dot (“.png”);
- `{docpath}`: the document’s directory with a trailing slash, or “”;
- `{language}`: the language.

“{” and “}” stand for braces. On an unknown field it returns the field’s name.

```
i18n.write_message :: proc(v: gd.View, id: gd.Node_Id, out:
^gd.Buffer) -> bool
```

`write_message` writes a node’s msgid as Sphinx makes it from the node’s source: line breaks and the indentation after them become one space, and the ends are trimmed.

```
i18n.write_template :: proc(t: ^Template, allocator :=
context.allocator) -> string
```

`write_template` writes the template as Sphinx lays out a POT file: every location on its own “#:” line, sorted, then the uuids, and strings broken only after newlines. The text is allocated with `allocator`, and the caller owns it.

## Constants

```
i18n.DEFAULT_FIGURE_FILENAME :: "{root}.{language}{ext}"
```

`DEFAULT_FIGURE_FILENAME` is Sphinx’s default `figure_language_filename`.

```
i18n.UI_DOMAIN :: "guidedog"
```

## inventory

### Types

```
inventory.Error :: enum u8
```

```
None
```

**Unknown\_Version**

the header names no inventory version Sphinx writes.

**Truncated**

the header ends early.

**Bad\_Compression**

the zlib data does not inflate.

**Out\_Of\_Memory**

the allocator refused memory: the inventory may be incomplete.

```
inventory.Inventory :: struct
```

```
project: string
```

```
version: string
```

```
items: [dynamic]Item
```

```
by_type: map[string]map[string]int
```

by\_type[type][name] is the index of the item in items.

```
folded: map[string]map[string]string
```

folded[type][lowercase name] is the first name of that spelling, for the types Sphinx matches ignoring case (labels and terms).

```
inventory.Item :: struct
```

```
project: string
```

the inventory's project and version, for "(in Python v3.14)".

```
version: string
```

```
name: string
```

```
type: string
```

domain:objtype, such as "py:function".

---

**priority:** int

---

**uri:** string

absolute, or relative to the inventory's base.

---

**display:** string

“-“ when it is the name itself.

### *Procedures*

---

```
inventory.add :: proc(inv: ^Inventory, item: Item) -> bool
```

add appends an item and indexes it; a later item of the same type and name replaces the earlier one in the index, as a later line overwrites in Sphinx's dictionaries. It allocates with context allocator, the arena the inventory lives in, and keeps item's strings, which must live as long. It is false, and the item may be left out, when the allocator refuses memory.

---

```
inventory.find :: proc(inv: ^Inventory, type, name: string) -> (Item, bool)
```

find returns the item of a type and name, which borrows the inventory.

---

```
inventory.find_folded :: proc(inv: ^Inventory, type, name: string) -> (Item, bool)
```

find\_folded returns the item of a type whose name matches ignoring case: the first such name listed, as Sphinx matches labels and terms. The item borrows the inventory; the lowercase key is made in the temp allocator.

---

```
inventory.load :: proc(data: []u8, base: string, allocator := context.allocator) -> (inv: Inventory, err: Error)
```

load reads an inventory. base joins relative locations, as Sphinx joins them with the URL or folder the inventory came from; "" leaves them as written. It allocates the inventory and its temporary work with allocator, an arena the caller frees; data is only borrowed. A header it does not know, or data that does not inflate, is an Error.

---

```
inventory.sort_items :: proc(items: []Item)
```

sort\_items orders items as Sphinx writes them: by domain, then by the name, display, type, and location of each object.

```
inventory.write :: proc(project, version: string, items:
[]Item, allocator := context.allocator) -> []u8
```

write returns an inventory in version 2 format, lines in the order given, as Sphinx writes them sorted. Locations ending in the name are shortened with \$, and a display equal to the name is written as -. It allocates the bytes, and its temporary work, with allocator, an arena the caller frees.

## jinja

### Types

```
jinja.Call :: struct
```

Call is what a native procedure receives: the positional and keyword arguments (for a filter or test, args[o] is the value being filtered or tested), the user pointer given at registration, and the receiver of a bound method.

```
args: []Value
```

```
kwargs: []Keyword
```

```
user: rawptr
```

```
self: Value
```

```
name: string
```

```
state: ^State
```

```
jinja.Callable :: struct
```

Callable is a function value: a native procedure (optionally bound to self) or a macro.

```
name: string
```

```
procedure: Native
```

```
user: rawptr
```

```
self: Value
```

```
macro: ^Macro
```

```
jinja.Cycler :: struct
```

Cycler is the object made by the `cycler()` global.

```
items: []Value
```

```
pos: int
```

```
jinja.Dict :: struct
```

Dict is an insertion-ordered dictionary. String keys are indexed; other keys are compared with Python equality (`1 == 1.0 == True`). allocator owns the dict as it owns a List.

```
keys: [dynamic]Value
```

```
values: [dynamic]Value
```

```
index: map[string]int
```

```
kind: Dict_Kind
```

```
name: string
```

the template name of a module

```
allocator: mem.Allocator
```

```
jinja.Dict_Kind :: enum u8
```

```
Dict
```

a plain dict

```
Namespace
```

`namespace()`: the only object whose attributes templates may assign

```
Module
```

the result of `{% import %}`: exported macros and variables

```
jinja.Environment :: struct
```

Environment holds options, filters, tests, globals and the template cache. Its own memory (parsed templates, tables) comes from an arena freed by `destroy_environment`.

```
options: Options
```

```
filters: map[string]Registered
```

```
tests: map[string]Registered
```

```
globals: map[string]Value
```

```
cache: map[string]^Compiled
```

```
arena: ^Arena
```

```
allocator: mem.Allocator
```

```
backing: mem.Allocator
```

```
stack: Stack_Guard
```

started by the outermost call on the environment

```
out_of_memory: bool
```

Set when the environment's memory could not hold its tables: every template it parses then fails with a Memory error.

```
jinja.Error :: struct
```

Error describes a failure. `kind == .None` means success. `excerpt` is the template line the error points at; `hint`, when set, suggests a fix.

```
kind: Error_Kind
```

```
template: string
```

```
line: int
```

```
message: string
```



<b>excerpt:</b> string
------------------------

<b>hint:</b> string
---------------------

jinja. <b>Error_Kind</b> :: <i>enum</i> u8
--------------------------------------------

<b>None</b>
-------------

<b>Syntax</b>
---------------

the template does not parse (TemplateSyntaxError)

<b>Undefined</b>
------------------

an undefined value was used (UndefinedError)

<b>Type</b>
-------------

an operation got the wrong kind of value (TypeError)

<b>Not_Found</b>
------------------

a template could not be loaded (TemplateNotFound)

<b>Limit</b>
--------------

a recursion, iteration or output limit was reached

<b>Memory</b>
---------------

the memory the parse or render may use could not hold it (MemoryError)

<b>Runtime</b>
----------------

any other failure while rendering

jinja. <b>Keyword</b> :: <i>struct</i>
----------------------------------------

Keyword is one keyword argument of a call.

<b>name:</b> string
---------------------

<b>value:</b> Value
---------------------

**jinja.List** :: struct

List is a list, or a tuple when tuple is set. fields names the items of a named tuple (the groups made by groupby), which can then also be read as attributes. allocator owns the list: its items grow there, and what a template stores in the list is copied there when the render ends (see Memory in the README).

**items**: [dynamic]Value

**tuple**: bool

**fields**: []string

**allocator**: mem.Allocator

**jinja.Loader** :: struct

Loader is a procedure plus its data. destroy, when set, is called by destroy\_environment.

**procedure**: Loader\_Proc

**user**: rawptr

**destroy**: proc(user: rawptr)

**jinja.Loader\_Proc** :: #type proc(user: rawptr, name: string, allocator: mem.Allocator) -> (source: string, found: bool)

Loader\_Proc returns the source of a template, allocated with allocator.

**jinja.Loop** :: struct

Loop is the special loop variable of a for loop.

**items**: []Value

**index0**: int

**depth0**: int

**last\_seen**: Value

for loop.changed()

**changed:** bool

whether `changed()` was called before

**recurse:** ^Recursion

set in a recursive loop

**jinja.Markup** :: *distinct* string

Markup is a string that is already safe HTML: autoescaping leaves it alone.

**jinja.Native** :: #type *proc*(call: ^Call) -> Value

Native is the one signature of every filter, test, global function and method written in Odin. Report errors with `fail(call, ...)` and return anything.

**jinja.None** :: struct

None is Jinja's none / None.

**jinja.Options** :: struct

**autoescape:** bool

**autoescape\_extensions:** []string

When set, templates whose names end with one of these (".html") autoescape and others do not, like `select_autoescape()`; `autoescape` is then ignored.

**trim\_blocks:** bool

**lstrip\_blocks:** bool

**keep\_trailing\_newline:** bool

**undefined:** Undefined\_Mode

**loader:** Loader

**max\_recursion:** int

nested macros, includes and recursive loops; 0 means 200

**max\_iterations:** int

loop iterations per render; 0 means 10,000,000

**max\_output:** int

Bytes per render, and per value a render builds (a repetition, padding, a join), checked before the value is allocated; 0 means unlimited.

**max\_nesting:** int

how deep template syntax nests; 0 means 100

**stack\_bytes:** int

machine stack parsing and rendering may use; 0 means 512 KiB

**random\_seed:** u64

seeds |random and lipsum(); renders are deterministic

**jinja.State** :: struct

State is one render in progress. Natives reach it through Call.state.

**env:** ^Environment**out:** ^strings.Builder**err:** Error**failed:** bool**autoescape:** bool**depth:** int**iterations:** int**ctx:** ^Context**current:** ^Compiled**line:** int

**rng:** u64

**guard:** ^Guard

the render's allocator; see `memory.odin`.

**scratch:** ^Arena

the render's own memory, or nil; see `ownership.odin`.

**caller:** mem.Allocator

context.allocator of the call that renders.

**stored:** map[rawptr]Value

containers not the render's own it stored values in.

**jinja.Template** :: struct

Template is a handle to a parsed template.

**env:** ^Environment

**compiled:** ^Compiled

**name:** string

**jinja.Undefined** :: struct

Undefined is what a missing variable, attribute or item evaluates to. The fields only feed the error message produced when the value is used in a way Jinja forbids.

**name:** string

the missing variable or attribute

**owner:** string

“dict object” when an attribute or item was missing

**hint:** string

a complete message that replaces the generated one

```
jinja.Undefined_Mode :: enum u8
```

#### Default

renders as “”, fails on attribute access, calls and arithmetic

#### Strict

fails on every use except `is defined` and `|default`

#### Chainable

like Default, but attributes of undefined are undefined too

#### Debug

like Default, but renders as `{{ name }}`

```
jinja.Value :: union #no_nil {Undefined, None, bool, i64,
f64, string, Markup, ^List, ^Dict, ^Callable, ^Loop, ^Cycler}
```

Value is every value a template can see. The zero value is Undefined.

### Procedures

```
jinja.add_filter :: proc(env: ^Environment, name: string,
procedure: Native, user: rawptr = nil)
```

`add_filter` registers a filter: `call.args[0]` is the filtered value. The environment copies name into its arena; user is borrowed until `destroy_environment`.

```
jinja.add_function :: proc(env: ^Environment, name: string,
procedure: Native, user: rawptr = nil)
```

`add_function` registers a native procedure as a global function, allocated in the environment’s arena; user is borrowed until `destroy_environment`.

```
jinja.add_global :: proc(env: ^Environment, name: string,
value: Value)
```

`add_global` makes a value visible to every template. The environment copies name into its arena; the value is borrowed and must outlive renders.

```
jinja.add_template :: proc(env: ^Environment, name, source:
string) -> Error
```

`add_template` parses source and caches it under name, so that `extends`, `include` and `import` find it without a loader. An error is allocated with `context allocator` (`destroy_error`).

```
jinja.add_test :: proc(env: ^Environment, name: string,
 procedure: Native, user: rawptr = nil)
```

add\_test registers a test: call.args[o] is the tested value; return a bool. The environment copies name into its arena; user is borrowed until destroy\_environment.

```
jinja.append_value :: proc(l: Value, items: ..Value)
```

append\_value adds values to the end of a list value, growing it with the list's own allocator; nothing happens when l is not a list or that allocator refuses.

```
jinja.arg :: proc(c: ^Call, index: int, name: string,
 default: Value = Undefined{}) -> Value
```

arg returns positional argument index, else the keyword argument name, else default.

```
jinja.arg_int :: proc(c: ^Call, index: int, name: string,
 default: i64) -> (i64, bool)
```

arg\_int reads an integer argument (bools and integral floats accepted).

```
jinja.autoescaping :: proc(c: ^Call) -> bool
```

autoescaping reports whether the call happens while autoescape is on.

```
jinja.boolean :: proc(b: bool) -> Value
```

boolean makes a boolean value.

```
jinja.call_filter :: proc(s: ^State, name: string, args:
 []Value, kwargs: []Keyword) -> Value
```

call\_filter runs a registered filter by name (used by map, and by hosts).

```
jinja.call_test :: proc(s: ^State, name: string, args:
 []Value, kwargs: []Keyword) -> bool
```

call\_test runs a registered test by name (used by select and reject).

```
jinja.check_args :: proc(c: ^Call, who: string, names:
 []string, required := 0) -> bool
```

check\_args enforces a Python signature: at most len(names) positional arguments and only the named keywords, each at most once. who names the callable in errors.

```
jinja.clear_cache :: proc(env: ^Environment)
```

clear\_cache forgets parsed templates so the next load reads them again.

```
jinja.create_environment :: proc(options: Options = {}) -
> Environment
```

create\_environment makes an environment with options, zero fields taking their defaults. It keeps its own arena, backed by context allocator, for parsed templates and registrations; destroy\_environment frees it. When that memory cannot hold the environment's tables, out\_of\_memory is set and every parse fails with a Memory error.

```
jinja.destroy_environment :: proc(env: ^Environment)
```

destroy\_environment frees the environment's arena, with every template parsed in it, and its loader's state (Loader.destroy).

```
jinja.destroy_error :: proc(err: Error, allocator :=
context.allocator)
```

destroy\_error frees an error returned by this package.

```
jinja.dict :: proc() -> Value
```

dict makes an empty dict, allocated with context.allocator (Undefined when it refuses the memory); fill it with set().

```
jinja.escape :: proc(v: Value, allocator :=
context.allocator) -> Markup
```

escape converts a value to Markup: Markup stays as it is, everything else is converted to a string and escaped (MarkupSafe's escape()), allocating with allocator when it must; the result may borrow v, so it belongs in an arena, as values do.

```
jinja.escape_string :: proc(s: string, allocator :=
context.allocator) -> string
```

escape\_string replaces the five HTML special characters as MarkupSafe does. When s has none it is returned as it is, borrowed; otherwise the result is allocated with allocator, which the caller owns (or, as usual, an arena holds).

```
jinja.evaluate :: proc(env: ^Environment, expression: string,
ctx: Value = {}) -> (Value, Error)
```

evaluate evaluates one expression against ctx (Jinja's compile\_expression). The value and an error's message are allocated with context.allocator, the caller's; the lists and dicts of the value grow through a small allocator



record kept there too. When that allocator refuses memory the error is a Memory error.

```
jinja.fail :: proc(c: ^Call, format: string, args: ..any) -> Value
```

fail records a runtime error from a native procedure; return its result. The message is formatted with context.allocator, the render's arena.

```
jinja.filesystem_loader :: proc(directories: []string) -> Loader
```

filesystem\_loader serves templates from directories, searched in order; the first match wins. Names use "/" and may not climb out of a directory with "...". The loader copies directories with context.allocator and frees them when its environment is destroyed (Loader.destroy). When that allocator refuses the copy, the loader finds nothing.

```
jinja.float :: proc(f: f64) -> Value
```

float makes a floating-point value.

```
jinja.format_error :: proc(err: Error, allocator := context.allocator) -> string
```

format\_error renders an error as a friendly report in the style of Elm's compiler, in a string allocated with allocator that the caller owns (" for no error):

```
-- UNDEFINED VALUE ----- page.html:3
'user' is undefined
3| <h1>{{ user.name }}</h1>
Hint: Pass it in the render context, or test it with `is defined`.
```

```
jinja.function :: proc(name: string, procedure: Native, user: rawptr = nil) -> Value
```

function wraps a native procedure as a callable value for globals or context data, allocated with context.allocator (Undefined when it refuses the memory); name and user are borrowed.

```
jinja.get :: proc(d: Value, key: string) -> Value
```

get reads a string key from a dict value; missing keys give Undefined.

```
jinja.has_arg :: proc(c: ^Call, index: int, name: string) -> bool
```

has\_arg reports whether the argument was passed at all.

```
jinja.integer :: proc(i: i64) -> Value
```

integer makes an integer value (int would shadow Odin's type).

```
jinja.is_markup :: proc(v: Value) -> bool
```

is\_markup reports a safe-HTML string.

```
jinja.is_none :: proc(v: Value) -> bool
```

is\_none reports None.

```
jinja.is_undefined :: proc(v: Value) -> bool
```

is\_undefined reports an undefined value.

```
jinja.keys :: proc(d: Value, allocator := context.allocator)
-> []string
```

keys returns the string keys of a dict value, in insertion order, in a slice allocated with allocator that the caller owns; the keys borrow the dict.

```
jinja.length :: proc(v: Value) -> int
```

length is the number of items of a list or dict, or of runes of a string.

```
jinja.list :: proc(items: ..Value) -> Value
```

list makes a list holding the given values, allocated with context.allocator; when it refuses the memory, the result is Undefined.

```
jinja.load_template :: proc(env: ^Environment, name: string)
-> (Template, Error)
```

load\_template loads a template through the environment's loader, cached in the environment's arena until destroy\_environment (or clear\_cache). An error is allocated with context.allocator; the caller frees it with destroy\_error.

```
jinja.markup :: proc(s: string) -> Value
```

markup makes a string value that is safe HTML, which autoescaping leaves alone; the value borrows s.

```
jinja.memory_loader :: proc(sources: ^map[string]string) -
> Loader
```

memory\_loader serves templates from a map the host owns (name -> source).

```
jinja.none :: proc() -> Value
```

none makes Python's None.

```
jinja.parse_template :: proc(env: ^Environment, name, source:
string) -> (Template, Error)
```

parse\_template parses source directly (not cached; not visible to includes). The template lives in the environment's arena, which copies source; an error is allocated with context allocator (destroy\_error).

```
jinja.render :: proc(t: ^Template, ctx: Value = {}) ->
(string, Error)
```

render renders a template with ctx (a dict value, or Undefined for none). The result and any error message are allocated with context allocator; everything else lives in a scratch arena freed before returning. When that allocator refuses memory the render stops with a Memory error at the line it was rendering.

```
jinja.render_position :: proc() -> (template: string, line:
int, excerpt: string, ok: bool)
```

render\_position is where the render running on this thread is: its template's name, the line being rendered, and that line's text; false when none runs. A host asked about a render while it runs, as when the render's allocator asks what to do about memory, names the place with it. The strings borrow the template.

```
jinja.render_source :: proc(env: ^Environment, source:
string, ctx: Value = {}, name := "<template>") -> (string,
Error)
```

render\_source parses and renders source in one call (nothing is cached), in a scratch arena it frees before returning. The output, and an error, are allocated with context allocator; the caller owns them.

```
jinja.repr :: proc(v: Value, allocator := context.allocator)
-> string
```

repr converts a value as Python's repr() does, in a string allocated with allocator, which the caller owns.

```
jinja.set :: proc(d: Value, key: string, value: Value)
```

set stores value under the string key in a dict value, growing it with the dict's own allocator; nothing happens when d is not a dict or that allocator refuses.

```
jinja.set_key :: proc(d: Value, key: Value, value: Value)
```

set\_key stores value under any key in a dict value.

```
jinja.str :: proc(s: string) -> Value
```

str makes a string value; the value borrows s.

```
jinja.to_string :: proc(v: Value, allocator :=
context.allocator) -> string
```

to\_string converts a value as Python's str() does (Undefined gives ""). A string or Markup value is returned as it is, borrowed; anything else is written into a string allocated with allocator, so the result belongs in an arena, as values do.

```
jinja.truthy :: proc(v: Value) -> bool
```

truthy is Python truthiness (Undefined is false).

```
jinja.tuple :: proc(items: ..Value) -> Value
```

tuple makes a tuple holding the given values, allocated as list allocates.

```
jinja.type_name :: proc(v: Value) -> string
```

type\_name is the Python type name used in Jinja's error messages.

```
jinja.undefined :: proc(name := "") -> Value
```

undefined makes an undefined value; name, borrowed, is what error messages call it.

```
jinja.values_equal :: proc(a, b: Value) -> bool
```

values\_equal compares as Python's == does: lists and dicts by content, 200 levels deep.

## odindoc

Package odindoc reads Odin packages by static analysis, with Odin's own parser (core:odin/parser), and writes the reStructuredText of the Guidedoc `odin` domain that documents them, as sphinx.ext.autodoc writes the Python domain's: packages, procedures, procedure groups, types and their fields or enumerators, constants, variables, and foreign imports, each with its doc comment turned into reST.

It never runs the compiler: types are shown as written. A package's files are read the way Odin builds them for every target at once: file-name suffixes (`_windows`, `_linux_arm64`) and `#+build` tags choose the targets a file builds for, and file-scope `when ODIN_OS == ...` blocks are evaluated when their conditions are that simple. A declaration made for every target the package builds for is documented once; one made for some targets only says which (see `platform.odin`).

It is a host library: it reads files through a `File_Reader` and allocates with the context allocator, freeing nothing itself, so a host runs it in an arena: a Project, the packages it loads, and the lines document returns live until the arena is freed.

### *Types*

<code>odindoc.Arch :: runtime.Odin_Arch_Type</code>
-----------------------------------------------------

<code>odindoc.Archs :: runtime.Odin_Arch_Types</code>
-------------------------------------------------------

<code>odindoc.Config :: struct</code>
---------------------------------------

<code>default_options: []Option</code>
----------------------------------------

<code>member_order: string</code>
-----------------------------------

“source” (the default), “alphabetical”, “groupwise”.

<code>odindoc.Decl :: struct</code>
-------------------------------------

Decl is a package-level declaration. signature is one line, without attributes, as the odin domain’s directive takes it (“clone :: proc(s: string) -> string”).

<code>name: string</code>
---------------------------

<code>kind: Kind</code>
-------------------------

<code>signature: string</code>
--------------------------------

<code>attributes: []string</code>
-----------------------------------

as written, in order: “@(require\_results)”.

<code>docs: string</code>
---------------------------

<code>private: Private</code>
-------------------------------

<code>members: []Member</code>
--------------------------------

<code>foreign_lib: string</code>
----------------------------------

the library of a foreign block it is declared in.

<code>targets: Targets</code>
-------------------------------

**availability:** string

its targets for people, when not the package's all.

**variant:** int

0; 1, 2, ... for later declarations of the name that differ.

**file:** string

base name.

**line:** int

**order:** int

in the package: files by name, then position.

`odindoc.File_Read :: struct`

**path:** string

**text:** string

`odindoc.File_Reader :: struct`

`File_Reader` reads the files the analysis needs. `procedure` reads a file; `list` names the entries of a folder (folders end with `"/`). Either may be `nil`, which uses the file system; a host confines both to its readable roots.

`reserve`, when set, is asked for the memory parsing a file may take (`parse_bound`) before the file is parsed; a file it refuses is not parsed (`Package.too_large`). Odin's parser does not survive a failed allocation, so a host whose allocator can refuse reserves the bound ahead, and its allocator then serves the parse from what was reserved.

**procedure:** `proc(user: rawptr, path: string) -> (string, bool)`

**list:** `proc(user: rawptr, dir: string) -> ([string, bool])`

**reserve:** `proc(user: rawptr, bytes: int) -> bool`

**user:** rawptr

```
odindoc.Import :: struct
```

Import is a package another package imports: the name it is used by in the source (its alias) and the id of the package it names.

```
alias: string
```

```
id: string
```

```
odindoc.Kind :: enum u8
```

```
Proc
```

```
Proc_Group
```

```
Struct
```

```
Union
```

```
Enum
```

```
Bit_Set
```

```
Bit_Field
```

```
Type
```

```
Const
```

```
Var
```

```
Foreign_Import
```

```
odindoc.Member :: struct
```

```
name: string
```

```
kind: Member_Kind
```

```
signature: string
```

**docs:** string

the comment above it and the one after it on its line.

`odindoc.Member_Kind :: enum u8`

**Field**

of a struct: “[using ]name: T[ tag]”.

**Enumerator**

of an enum: “Name[ = value]”.

**Bit\_Field**

of a bit\_field: “name: T | bits”.

`odindoc.Message :: struct`

**severity:** Severity

**code:** string

**title:** string

**text:** string

**hint:** string

`odindoc.OS :: runtime.Odin_OS_Type`

`odindoc.Option :: struct`

**name:** string

**value:** string

`odindoc.Package :: struct`

Package is an analysed package. doc is its package comment (from doc.odin first, then the first file by name that has one) with comment markers stripped; synopsis is that comment’s first sentence.



**id:** string

**name:** string

**dir:** string

**doc:** string

**synopsis:** string

**decls:** []Decl

**imports:** []Import

**targets:** Targets

every target some file of the package builds for.

**platform:** string

targets shown for people; “” when it builds for all.

**files:** []string

the package’s files that were read, sorted.

**errors:** int

syntax errors the parser reported.

**too\_deep:** [dynamic]string

files nested too deeply to parse (nesting.odin).

**too\_large:** [dynamic]string

files whose parse\_bound the reader refused.

odindoc.**Package\_Ref** :: struct

**id:** string

**dir:** string

```
odindoc.Private :: enum u8
```

**Public**

**Package**

@(private), or a #+private file.

**File**

@(private="file"), or a #+private file file.

```
odindoc.Project :: struct
```

**roots:** []Root

**reader:** File\_Reader

**files:** [dynamic]File\_Read

every file read, for the host's dependencies.

**packages:** map[string]^Package

loaded packages by id; nil for one that failed.

```
odindoc.Request :: struct
```

**directive:** string

"odin:autopackage", "odin:autoproc", ...

**argument:** string

**options:** []Option

**current:** string

the reader's current odin package, or "".

```
odindoc.Result :: struct
```

**lines:** [dynamic]string

**messages:** [dynamic]Message

```
odindoc.Root :: struct
```

Root is a folder holding packages. Without a collection, a package's id is its path relative to dir ("readers/sphinx"), and dir's own package is named by dir's base name; with one, ids are import paths in that collection ("core:fmt").

```
dir: string
```

```
collection: string
```

```
odindoc.Severity :: enum u8
```

```
Note
```

```
Warning
```

```
Error
```

```
odindoc.Targets :: [OS]Archs
```

Targets is a set of (os, arch) pairs: per system, its architectures.

### Procedures

```
odindoc.document :: proc(p: ^Project, config: ^Config,
request: Request) -> Result
```

document runs one autodoc directive (request.directive, "odin:autopackage" and the others, with its argument and options over config's defaults) and returns the reStructuredText lines it generates, with any messages: an unknown directive, option, package, or name is an Error message and no lines, never a panic. Packages are loaded through p as needed. The result lives in the context arena.

```
odindoc.init :: proc(p: ^Project, roots: []Root, reader :=
File_Reader{})
```

init starts a project over roots, reading through reader (the file system when it is zero). The project and everything it loads live in the context arena.

```
odindoc.list_packages :: proc(p: ^Project) -> []Package_Ref
```

list\_packages lists every folder below the roots that holds a package, by id.

```
odindoc.load :: proc(p: ^Project, id: string) -> (^Package,
bool)
```

load analyses the package with an id, once; later calls return the same package, which the project keeps. It is false when no root holds a package with that id.

```
odindoc.parse_bound :: proc(n: int) -> int
```

parse\_bound is the memory parsing a file of n bytes may take.

### Constants

```
odindoc.PARSE_BASE_BYTES :: 256 << 10
```

PARSE\_BASE\_BYTES is what the parser allocates for any file, and an arena block's room.

```
odindoc.PARSE_BYTES_PER_BYTE :: 256
```

PARSE\_BYTES\_PER\_BYTE bounds what Odin's parser allocates, with the build tags read after it, for each byte of a file it parses. Its tree takes 20 to 30 bytes for each byte of ordinary code, and up to about 170 for the densest input measured: long operator chains, runs of semicolons, and one-character list items (parse\_bounds\_hold). The bound counts every growth of a list as a new allocation, as an arena that cannot grow it in place makes it.

## project

Package project builds a documentation project the way Sphinx does: a source directory with conf.toml, builders writing \_build/<builder>, and incremental builds. It is host code and may allocate; documents are read and rendered by the Guidedoc engine.

Its surface is the verbs on a Project (project.odin says who owns what they allocate) and the types they take and return; every other file is #+private. lib/project/README.md lists the surface.

### Types

```
project.Build_Options :: struct
```

Build\_Options are one build's command-line options; build copies them.

```
all: bool
```

-a: write every output, not only outdated ones.

```
fresh: bool
```

-E: ignore the saved environment.

**nitpicky:** bool`-n`**warnings\_as\_errors:** bool`-W`**keep\_going:** bool`--keep-going`**untrusted:** bool`--untrusted:` see `untrusted_policy`.**jobs:** int`-j:` reading threads; 0 or 1 is one, -1 one per core.**budget:** int`--budget:` bytes; 0 is `BUILD_BUDGET`.**memory:** `host.Budget_Policy`when the budget is reached: `-memory`, or `ask`.**heap:** `mem.Allocator`

Where steps' scratch memory comes from (a document read, a page rendered); zero: the heap. Reading threads use it at once, so it must be thread-safe.

**max\_depth:** int`--max-depth:` how deep content may nest; 0 is the default.**max\_nodes:** int`--max-nodes:` nodes per document; 0 is the default.**stack\_bytes:** int`--stack-kib:` stack budget; 0 is the default. See `build_limits`.**tags:** `[]string``-t`

**output\_dir:** string

sphinx-build form; otherwise build\_dir/<builder>.

**arena\_block:** int

bytes the build's arenas ask for at a time; 0 is 64 KiB.

**now:** datetime.DateTime

the build's moment; zero is now, in UTC.

**year:** string

project.**Build\_Result** :: struct

Build\_Result is what a build did: exit 0, or exit != 0 with a problem; the reports of the whole build either way, and its counters. It owns everything it refers to, in its arena, until release.

**exit:** int

**problem:** host.Problem

**reports:** []host.Report

**read:** int

documents parsed.

**unread:** int

documents that could not be read.

**reused:** int

documents taken from the environment.

**written:** int

outputs written: published, or only prepared when not published.

**unchanged:** int

outputs already up to date.

**output:** string

**reached:** `Build_Stage`

how far the build got.

**published:** `bool`

the outputs are in place; otherwise the previous ones stay,

**unsettled:** `bool`

unless this is set: a publication neither finished nor undone.

**peak:** `int`

the most memory the budget held at once, in bytes.

**budget:** `int`

the budget at the end, in bytes: `Budget_Policy` may raise it.

**memory:** `^Memory`

everything the build allocated; see `release`.

**owner:** `mem.Allocator`

the allocator memory came from.

`project.Build_Stage :: enum`

`Build_Stage` is how far a build got, so its summary says what it did and no more.

### Preparing

settings, the documents to read, the readers: no document read yet.

### Reading

reading the documents and resolving their references.

### Writing

writing the outputs beside the output folder.

### Written

every output written; publishing them is what is left.

```
project.Builder_Kind :: enum u8
```

Builder\_Kind is the output a build writes; BUILDER\_NAMES are the names the command line gives them, and builder\_named reads a name.

Html

Dirhtml

Singlehtml

Pdf

Text

Gettext

Dummy

```
project.Config :: struct
```

Config holds conf.toml. Names follow conf.py so a Sphinx user recognizes them.

**source\_dir:** string

the directory holding conf.toml (absolute).

**build\_dir:** string

\_build beside it, or build/ in the separated layout.

**project:** string

Project.

**author:** string

Project.

**copyright:** string

Project.



**version:** string

Project.

**release:** string

Project.

**today:** string

Project.

**today\_fmt:** string

Project.

**language:** string**root\_doc:** string

General.

**source\_suffix:** []string**exclude\_patterns:** []string**include\_roots:** []string

folders beyond the source folder files may come from.

**include\_patterns:** []string**templates\_path:** []string**extensions:** []string**rst\_prolog:** string**rst\_epilog:** string**default\_role:** string**primary\_domain:** string

---

**highlight\_language:** string

---

**pygments\_style:** string

Pygments style for code; “” is the theme’s.

---

**pygments\_dark\_style:** string

for dark pages; “” is github-dark.

---

**numfig:** bool

---

**numfig\_format:** map[string]string

---

**numfig\_secnum\_depth:** int

---

**math\_number\_all:** bool

---

**math\_eqref\_format:** string

---

**math\_numfig:** bool

---

**nitpicky:** bool

---

**nitpick\_ignore:** [][]string

---

**nitpick\_ignore\_regex:** [][]string

---

**nitpick\_patterns:** [][]regex.Regular\_Expression

nitpick\_ignore\_regex, compiled.

---

**show\_authors:** bool

---

**add\_function\_parentheses:** bool

---

**add\_module\_names:** bool

---

**toc\_object\_entries:** bool

---

**toc\_object\_entries\_show\_parents:** string

“domain”, “hide”, or “all”.

**trim\_footnote\_reference\_space:** bool

**smartquotes:** bool

**keep\_warnings:** bool

**suppress\_warnings:** []string

**todo\_include\_todos:** bool

**autosectionlabel\_prefix\_document:** bool

labels are “docname:Title”.

**autosectionlabel\_maxdepth:** int

sections this deep or deeper get no label; 0: all.

**trim\_doctest\_flags:** bool

doctest flags and <BLANKLINE> hidden in sessions.

**manpages\_url:** string

**option\_emphasise\_placeholders:** bool

**extlinks:** map[string][2]string

**extlinks\_detect\_hardcoded\_links:** bool

**object\_types:** [dynamic]sphinx.Object\_Type

see object\_types.odin.

**directive\_aliases:** []sphinx.Directive\_Alias

**mathjax\_path:** string

sphinx.ext.mathjax, Sphinx’s HTML math; see mathjax.odin.

**mathjax\_options:** map[string]string

**mathjax\_inline:** []string

each an opening and a closing delimiter.

**mathjax\_display:** []string

each an opening and a closing delimiter.

**mathjax3\_config:** string

JSON, as json.dumps writes the table.

**mathjax4\_config:** string

JSON, as json.dumps writes the table.

**mathjax\_config\_path:** string

**intersphinx\_mapping:** []Intersphinx\_Project

sphinx.ext.intersphinx; see intersphinx.odin.

**intersphinx\_cache\_limit:** int

days an inventory is reused; below 0: always.

**intersphinx\_timeout:** int

seconds; 0: no limit, as None in conf.py.

**intersphinx\_disabled\_reftypes:** []string

**intersphinx\_resolve\_self:** string

**autodoc\_source\_paths:** []string

sphinx.ext.autodoc, which reads Python source without running it;  
see autodoc.odin.

folders holding the packages, like sys.path.

**autoclass\_content:** string

**autodoc\_class\_signature:** string

**autodoc\_default\_options:** []ad.Default\_Option

**autodoc\_docstring\_signature:** bool

**autodoc\_inherit\_docstrings:** bool

**autodoc\_member\_order:** string

**autodoc\_mock\_imports:** []string

accepted: nothing is imported, so nothing is mocked.

**autodoc\_preserve\_defaults:** bool

**autodoc\_typehints:** string

**autodoc\_typehints\_description\_target:** string

**autodoc\_typehints\_format:** string

**autodoc\_type\_aliases:** map[string]string

**autodoc\_use\_type\_comments:** bool

**autodoc\_warningiserror:** bool

**python\_display\_short\_literal\_types:** bool

**strip\_signature\_backslash:** bool

**odin\_autoapi\_dirs:** []string

The odin domain's generating directives and API pages; see `odin.odin`.  
folders whose packages get API pages.

**odin\_collections:** map[string]string

collection name to folder ("core").

**odin\_autoapi\_root:** string

the folder of the generated pages: "api".

**odin\_autoapi\_options:** []string

members, undoc-members, private-members.

**odin\_autoapi\_member\_order:** string

source, alphabetical, groupwise.

**odin\_autoapi\_add\_tocentry\_entry:** bool

the API index joins the root's toctree.

**odin\_autoapi\_generate\_api\_docs:** bool

false: only the directives read the dirs.

**html\_theme:** string

HTML.

**html\_theme\_options:** map[string]string

**html\_title:** string

**html\_short\_title:** string

**html\_logo:** string

**html\_favicon:** string

**html\_static\_path:** []string

**html\_extra\_path:** []string

**html\_css\_files:** []string

**html\_js\_files:** []string

**html\_last\_updated\_fmt:** Maybe(string)

unset: no date; “”: the default format.

**html\_permalink:** bool

---

```
html_permlinks_icon: string
```

---

```
html_use_index: bool
```

---

```
html_domain_indices: bool
```

---

the Python module index, py-modindex.html.

---

```
modindex_common_prefix: []string
```

---

prefixes the module index leaves out.

---

```
html_split_index: bool
```

---



---

```
html_copy_source: bool
```

---



---

```
html_show_sourcelink: bool
```

---



---

```
html_sourcelink_suffix: string
```

---



---

```
html_show_copyright: bool
```

---



---

```
html_show_sphinx: bool
```

---



---

```
html_baseurl: string
```

---



---

```
html_file_suffix: string
```

---



---

```
html_link_suffix: string
```

---



---

```
html_secnumber_suffix: string
```

---



---

```
html_context: Toml_Value
```

---

a table, as written: the pages' variables.

---

```
html_additional_pages: map[string]string
```

---

page name: the template it is made from.

---

```
pdf_documents: []Pdf_Document
```

---

PDF.

**pdf\_paper\_size:** string

**pdf\_logo:** string

**pdf\_toplevel\_sectioning:** string

**pdf\_show\_urls:** string

**pdf\_preamble:** string

**pdf\_packages:** string

**pdf\_fonts:** string

**pdf\_font\_paths:** []string

**typst:** string

**graphviz\_output\_format:** string

Graphviz, linked in through graphdog.  
svg; png is read as svg.

**graphviz\_dot\_args:** []string

-Gname=value, -Nname=value, -Ename=value.

**graphviz\_dot:** string

accepted from conf.py; Graphviz is built in.

**myst\_enable\_extensions:** []string

Markdown, with myst-parser 0.16.1's names; the reader has the support built in.  
optional syntax, such as colon\_fence.

**myst\_heading\_anchors:** int

heading depth that gets slug anchors; 0: none.

**myst\_dmath\_allow\_labels:** bool



---

**myst\_dmath\_allow\_space:** bool

---

**myst\_dmath\_allow\_digits:** bool

---

**myst\_dmath\_double\_inline:** bool

---

**myst\_linkify\_fuzzy\_links:** bool

---

**myst\_substitutions:** map[string]string

---

**myst\_sub\_delimiters:** []string

two characters, each doubled: {{ name }}.

---

**myst\_url\_schemes:** []string

schemes that make external links; empty: all.

---

**myst\_footnote\_transition:** bool

---

**myst\_html\_meta:** map[string]string

---

**myst\_commonmark\_only:** bool

CommonMark syntax only; headings still make sections.

---

**myst\_disable\_syntax:** []string

markdown-it rule names not parsed.

---

**myst\_words\_per\_minute:** int

---

**myst\_dmath\_enable:** bool

deprecated spellings of the extensions.

---

**myst\_amsmath\_enable:** bool

deprecated spellings of the extensions.

---

**locale\_dirs:** []string

Internationalization, with Sphinx's names (see lib/i18n).  
relative to the source directory.

**gettext\_compact:** Flag\_Or\_Name

true, false, or one catalog's name.

**gettext\_uuid:** bool

**gettext\_location:** bool

**gettext\_auto\_build:** bool

compile .po files to .mo beside them.

**gettext\_additional\_targets:** []string

**gettext\_exclude\_patterns:** []string

document paths excluded from extraction only.

**gettext\_allow\_fuzzy\_translations:** bool

**gettext\_last\_translator:** string

**gettext\_language\_team:** string

**figure\_language\_filename:** string

**translation\_progress\_classes:** Flag\_Or\_Name

true, false, translated, untranslated.

**html\_title\_derived:** bool

html\_title was made from the project name.

**html\_short\_title\_derived:** bool

**project.Flag\_Or\_Name** *:: struct*

Flag\_Or\_Name is a setting that is true, false, or a name, as gettext\_compact is.

**on:** bool

**name:** string

the name, when one was given; on is then true.

```
project.Intersphinx_Project :: struct
```

Intersphinx\_Project is one intersphinx\_mapping entry: the name references use as a prefix, the base URL of the other project's pages, and where its inventory is, tried in order; "" stands for None, the base URL's objects.inv.

```
name: string
```

```
uri: string
```

```
locations: []string
```

```
project.Intl_Action :: enum u8
```

Intl\_Action is what intl\_update did to one catalog.

```
Create
```

```
Update
```

```
Not_Changed
```

```
project.Intl_File :: struct
```

Intl\_File is one catalog intl\_update or intl\_stat looked at, with its statistics.

```
action: Intl_Action
```

```
path: string
```

```
stats: gettext.Stats
```

```
project.Intl_Options :: struct
```

Intl\_Options are sphinx-intl's options for update and stat.

```
pot_dir: string
```

-p: the templates; default: the gettext builder's output.

```
locale_dir: string
```

-d: default: the first of locale\_dirs.

**languages:** []string

-l, repeatable; default: the project's language.

**width:** int

-w: line width; 0 means 76.

**obsolete:** bool

keep obsolete messages (sphinx-intl's default).

**project.Load\_Options :: struct**

Load\_Options are the command line's say in where a project is and how it is set; load borrows them for the call.

**conf\_dir:** string

-c: where conf.toml is, when not in the source directory.

**no\_config:** bool

-C: build with defaults only.

**overrides:** []string

-D name=value, applied after conf.toml.

**source:** string

sphinx-build form: explicit source directory.

**output:** string

sphinx-build form: explicit output directory.

**project.Migrate\_Options :: struct**

**input:** string

conf.py, guidedog.toml, or a folder holding one.

**output:** string

where conf.toml goes; "" is beside the input, "-" writes nothing.

**force:** bool

replace an existing conf.toml.

```
year: string
```

what datetime's current year evaluates to.

```
project.Migrate_Result :: struct
```

Migrate\_Result is what migrate wrote (toml, and output unless that was "-"), with notes on what it could not convert as written, and how many settings it converted.

```
input: string
```

```
output: string
```

```
toml: string
```

```
notes: []host.Report
```

```
converted: int
```

settings written to conf.toml.

```
attention: int
```

settings left out that may matter.

```
project.Pdf_Document :: struct
```

Pdf\_Document is one book of pdf\_documents: which document it starts from, and the file, title, and author it gets.

```
root: string
```

the root document of the book.

```
file: string
```

the PDF name in \_build/pdf.

```
title: string
```

```
author: string
```

**project.Project** :: struct

Project is a loaded documentation project. The API is a handful of verbs over it:

```
p, problem := project.load("docs", {}, defaults.registry())
defer project.unload(&p)
if problem.exit != 0 do return problem.exit
result := project.build(&p, .Html, {})
defer project.release(&result)
project.clean(&p)
```

Who owns what each entry point allocates, all of it taken from context.allocator:

- **load**: the project owns everything, in an arena of its own; **unload** releases it, including the text of a problem load returned, so it is called whether load succeeded or not.
- **build**: the result owns everything the build allocated, in an arena of its own; **release** frees it. The project is never changed by a build (see **build**), so it may be built any number of times, and must outlive every result of its builds only while their reports are used. **load**, **build**, **release**, **unload** leaves nothing live.
- **serve**: each request works in an arena it frees; the rebuild callback owns (and releases) the results of the builds it runs.
- **clean**, **intl\_update**, **intl\_stat**, **migrate**, **quickstart**: their results and problems are allocated in context.allocator and not freed one by one; a caller on the heap runs them with an arena (or the temporary allocator) as context.allocator and frees it once it has used them.
- **builder\_named**, **absolute**, **join**: plain values; **absolute** and **join** allocate the path.

**config**: Config

**warnings**: [dynamic]host.Report

from configuration; builds add their own.

**registry**: gd.Registry

**settings**: string

digest of conf.toml and the -D overrides: what every output uses.

**memory**: ^Memory

everything load allocated; see **unload**.

**owner**: mem.Allocator

the allocator memory came from.

```
project.Quickstart_Options :: struct
```

Quickstart\_Options are guidedog quickstart's answers; "" takes each default.

```
dir: string
```

```
project: string
```

```
author: string
```

```
version: string
```

```
release: string
```

```
language: string
```

```
suffix: string
```

“.rst” by default.

```
root_doc: string
```

“index” by default.

```
separate: bool
```

source/ and build/ instead of one folder with \_build.

```
year: string
```

for the copyright line.

```
date: string
```

for the root document's comment.

```
project.Server :: struct
```

Server previews the built html output on the loopback interface. Before serving a page it checks whether any source changed and rebuilds incrementally if so.

```
project: ^Project
```

```
rebuild: proc(s: ^Server) -> bool
```

returns false when the rebuild failed.

```
user: rawptr
```

```
stamp: i64
```

fingerprint of watched paths, sizes, and modification times.

```
project.Toml_Kind :: enum u8
```

TOML 1.0: tables, arrays of tables, dotted and quoted keys, inline tables, arrays, every string form, integers in all bases, floats, booleans, and date-times (kept as text). Errors carry the line, so a mistake in `conf.toml` points at itself.

```
String
```

```
Integer
```

```
Float
```

```
Boolean
```

```
Datetime
```

```
Array
```

```
Table
```

```
project.Toml_Value :: struct
```

`Toml_Value` is one parsed TOML value; `Config.html_context` keeps a table of them as written. Its strings and lists belong to the project that loaded it.

```
kind: Toml_Kind
```

```
text: string
```

String and Datetime.

```
integer: i64
```



**float:** f64

**boolean:** bool

**items:** [dynamic]Toml\_Value

Array.

**keys:** [dynamic]string

Table, in definition order.

**values:** [dynamic]Toml\_Value

Table, parallel to keys.

**line:** int

**defined:** bool

Table: defined by a header or as a value, not only implied.

**inline:** bool

written inline, so it may not be extended later.

## Procedures

**project.absolute** :: *proc(path: string) -> string*

absolute makes a path absolute against the working directory without resolving it, so it works for folders a build has not created yet. The path is allocated in the context allocator and owned by the caller.

**project.build** :: *proc(p: ^Project, kind: Builder\_Kind, o: Build\_Options) -> Build\_Result*

build reads, resolves, and writes the project with one builder. Outputs are written only for documents whose inputs changed, unless o.all is set.

Ownership: the build allocates everything, its result included, in an arena of its own taken from context.allocator, and release frees it: a caller releases each result once it is done with its reports. The build never changes the project: it works on its own copy of the configuration (build\_config), so an –untrusted build confines only itself, and any number of builds may follow on the same project.

The memory of each step (a document read, a page rendered) comes from o.heap, the heap by default, and is freed when the step ends. When

context.allocator or o.heap refuses memory, the build stops with host.memory (exit 3) and publishes nothing.

```
project.build_limits :: proc(o: Build_Options) -> (gd.Limits,
host.Problem)
```

build\_limits are the limits every session of a build reads and renders with: the documented defaults, with `-max-depth`, `-max-nodes`, and `-stack-kib`. Reading runs on the main thread, or on core:thread's threads with `-j`, and pages render on the main thread, so the stack budget may be at most what the smaller of those stacks holds (host.stack\_budget\_limit); without `-stack-kib` it is the default, or less on a thread too small for it. A `-stack-kib` above that is the problem build.stack (exit 2); a caller may call it before build to refuse the options early. Only the problem's message is allocated.

```
project.builder_named :: proc(name: string) -> (Builder_Kind,
bool)
```

builder\_named accepts Guidedog's builder names and the Sphinx names that map onto them: the LaTeX builders become pdf, because Guidedog typesets PDF with Typst.

```
project.clean :: proc(p: ^Project) -> host.Problem
```

clean removes everything in the project's build directory, as `make clean` does for Sphinx, and leaves the directory itself; a missing one is nothing to clean. It stops at the first file it cannot remove and returns its problem (host.io), whose text is allocated in the caller's context allocator and owned by the caller.

```
project.intl_stat :: proc(p: ^Project, options: Intl_Options)
-> ([Intl_File, host.Problem)
```

intl\_stat is sphinx-intl stat: the translated, fuzzy, and untranslated messages of every catalog of the languages. It writes nothing; the files are allocated in context.allocator (see Project).

```
project.intl_update :: proc(p: ^Project, options:
Intl_Options) -> ([Intl_File, host.Problem)
```

intl\_update is sphinx-intl update: for each template of pot\_dir and each language, it creates the translation catalog, or merges the template into the existing one (gettext.merge: kept, fuzzy, and obsolete messages, as msgmerge does). It returns what it did to each catalog, allocated in context.allocator (see Project), or the first problem, with the catalogs written before it kept.

```
project.join :: proc(parts: ..string) -> string
```

join joins path parts with the system's separator and cleans the result, as filepath.join does; the path is allocated in context.allocator.

```
project.load :: proc(dir: string, options: Load_Options,
registry: gd.Registry) -> (p: Project, problem: host.Problem)
```

load finds the project from dir (see find\_project) or the explicit directories in options, and reads its configuration. The registry is checked first: an adapter with no ID, version, or procedure, or an ID registered twice, is a problem, as it is for gd.convert. Whether it has the readers the documents need is known once a build has found them (find\_readers). The project owns everything load allocates, in an arena taken from context allocator, which unload frees; a problem's text lives in that arena too. When that allocator refuses memory, load stops with host.memory (exit 3), and unload is still what frees the rest.

```
project.migrate :: proc(o: Migrate_Options) -> (r:
Migrate_Result, problem: host.Problem)
```

migrate turns the Sphinx conf.py (or a guidedog.toml) that o.input names into conf.toml, and writes it unless o.output is "-": literal assignments become settings, and what Python would have computed is reported in r.notes. The written conf.toml is loaded first, so it always builds; an existing one is refused unless o.force is set. Everything is allocated in context allocator (see Project); a failure is the problem, with nothing written.

```
project.quickstart :: proc(o: Quickstart_Options) ->
(created: []string, p: host.Problem)
```

quickstart creates a documentation project in o.dir laid out as Sphinx lays one out, with conf.toml, a root document, and the default templates copied in so they can be edited, and returns the paths it created. It refuses to overwrite a project. The paths and the problem are allocated in context allocator (see Project).

```
project.release :: proc(r: ^Build_Result)
```

release frees everything a build allocated, its result included, which is empty after.

```
project.serve :: proc(s: ^Server, port: int) -> host.Problem
```

serve previews the project's built html output on 127.0.0.1:port, rebuilding through s.rebuild before a page is served when a source changed since the last build. It borrows s and the project for as long as it runs, which is until the process ends: it returns only the problem of a port it cannot listen on. Each request works in an arena of its own on the heap, freed when it is answered.

```
project.unload :: proc(p: ^Project)
```

unload releases everything load allocated.

### Constants

```
project.BUILD_BUDGET :: 1 << 30
```

BUILD\_BUDGET bounds what the build holds at once: its sessions, and the files it reads: reading threads wait rather than exceed it (see `host.Budget`). – budget sets another.

```
project.CONFIG_FILE :: "conf.toml"
```

CONFIG\_FILE is the name of a project's configuration, in its source directory.

### Variables

```
@(rodata) project.BUILDER_NAMES
```

## pyscan

Package `pyscan` analyses Python source statically, through tree-sitter: modules and their docstrings, `__all__`, and imports; classes with their bases, decorators, and members in source order; functions with their parameters as written; and the attribute documentation Sphinx's `ModuleAnalyzer` finds (`#`: comments and docstrings after assignments). Names resolve across the modules of the configured source roots, so a class re-exported by a package is found where it is defined. Nothing is executed.

It is a host library: it reads files (through a reader the host may confine) and allocates with the context allocator, freeing none of it, so a host runs it in an arena: the project and every module, class, and string it returns live until the arena is freed. `destroy` frees what tree-sitter holds, which no arena can.

### Types

```
pyscan.All :: struct
```

All is a module's `__all__` when it can be read without running code.

```
names: [dynamic]string
```

```
defined: bool
```

the module assigns `__all__`.

```
known: bool
```

and its value was read: literal lists, `+=`, `append`, `extend`.

**pyscan.Analysis** :: struct

Analysis is what Sphinx's ModuleAnalyzer (sphinx.pycode.parser.Parser) finds in a module, keyed as it keys them: attribute documentation from #: comments and from string literals after assignments, annotations, definition order, @final, @overload.

**attr\_docs:** map[Attr\_Key][dynamic]string

lines, ending with "".

**attr\_order:** [dynamic]Attr\_Key

first-insertion order of attr\_docs.

**annotations:** map[Attr\_Key]string

**annotation\_order:** [dynamic]Attr\_Key

first-insertion order of annotations.

**tagorder:** map[string]int

dotted qualname -> order of its last definition.

**counter:** int

**finals:** map[string]bool

**overloads:** map[string][dynamic]^Function

**pyscan.Attr\_Key** :: struct

Attr\_Key is (scope, name): scope is "" at module level, else the class's qualname.

**scope:** string

**name:** string

**pyscan.Class** :: struct

**module:** ^Module

**name:** string

**qualname:** string

**node:** ts.Node

**bases:** [dynamic]ts.Node

positional arguments of the class statement.

**keywords:** [dynamic]Keyword

metaclass=..., and others.

**decorators:** [dynamic]ts.Node

the decorator expressions.

**doc:** Doc

**scope:** Scope

**outer:** ^Class

the class it is nested in, or nil.

**mro\_done:** bool

**mro\_complete:** bool

**mro\_order:** [dynamic]Mro\_Entry

pyscan.**Def** :: struct

Def is one binding of a name in a module or class body, in source order.

**name:** string

**kind:** Def\_Kind

**node:** ts.Node

the statement, or the definition.

**runtime:** bool

false under if TYPE\_CHECKING: .

**class:** ^Class

**function:** ^Function

**value:** ts.Node

right-hand side; null when there is none.

**annotation:** ts.Node

the type node of an annotated assignment.

**target:** string

imports: the absolute module imported from, or imported.

**member:** string

Import\_From: the name imported; “\*” for a wildcard.

**tuple:** bool

bound by unpacking: its value is one part of value.

**line:** int

1-based line of the statement.

pyscan.**Def\_Kind** :: *enum* u8

**Class**

**Function**

**Assign**

name = value, name: type = value.

**Annotation**

name: type, with no value.

**Import**

import a.b [as c].

**Import\_From**

from m import name [as alias].

**Type\_Alias**

type Name = value (PEP 695).

pyscan.**Doc** :: struct

Doc is a docstring as Python evaluates it (escapes processed), or none.

**text**: string

**ok**: bool

pyscan.**File\_Read** :: struct

**path**: string

**text**: string

pyscan.**File\_Reader** :: struct

File\_Reader reads a file for the analyser. The host confines reads to what it may read; ok is false for a file that is missing or refused.

**procedure**: proc(user: rawptr, path: string) -> (text: string, ok: bool)

**user**: rawptr

pyscan.**Function** :: struct

**module**: ^Module

**name**: string

**qualname**: string

**node**: ts.Node

the function\_definition, or the lambda.



**is\_async:** bool

**is\_lambda:** bool

**decorators:** [dynamic]ts.Node

**params:** [dynamic]Param

**returns:** ts.Node

the return type node, or null.

**doc:** Doc

**class:** ^Class

the class whose body defines it, or nil.

**type\_comment:** bool

A PEP 484 type comment after the header, “# type: (int, str) -> bool”: the parameter types as written (“...” alone when elided) and the return type.

**type\_args:** [dynamic]string

**type\_returns:** string

**pyscan.Keyword** :: struct

**name:** string

**value:** ts.Node

**pyscan.Module** :: struct

**project:** ^Project

**name:** string

dotted.

**kind:** Module\_Kind

**path:** string

the file read; the folder of a namespace package.

**source:** string

with line breaks normalised to \n.

**tree:** `^ts.Tree`

**root:** `ts.Node`

**doc:** `Doc`

**scope:** `Scope`

**all:** `All`

**info:** `Analysis`

what Sphinx's `ModuleAnalyzer` finds.

**broken:** bool

tree-sitter found syntax errors.

**too\_deep:** bool

it nests past `MAX_SYNTAX_DEPTH`, so it is not analysed (kind `.Native`).

**origin:** `^Module`

A parsed snippet (`parse_expression`) names things in the namespace of origin.

`pyscan.Module_Kind` *:: enum u8*

**Source**

a .py file.

**Package**

a folder with `__init__.py`.

**Namespace**

a folder without `__init__.py` (PEP 420).

**Native**

a compiled extension (`.so`, `.pyd`): it cannot be read.

`pyscan.Mro_Entry :: struct`

`Mro_Entry` is one class of a method resolution order: a class in the roots, or one outside them known by path.

**class:** `^Class`

**path:** `string`

for a class outside the roots: “`builtins.object`”, “`werkzeug.X`”.

`pyscan.Object :: struct`

**kind:** `Object_Kind`

**module:** `^Module`

.Module: the module itself; else the module that defines it.

**class:** `^Class`

**function:** `^Function`

**value:** `ts.Node`

.Value: the expression assigned.

**tuple:** `bool`

.Value: bound by unpacking, so value is only its source.

**path:** `string`

.External, .Builtin: the dotted name.

**wrapper:** `Wrapper`

**abstract:** booldecorated with `abstractmethod`.**owner:** ^Class

the class whose body defines it, when found in one.

**def:** ^Def

the binding it was found through.

**decorator:** string

.Unknown wrapper: the decorator as written.

**doc:** Doca docstring given explicitly: `property(fget, doc="...")`.**class\_property:** boola property under `classmethod`: a property of the class.**partial\_args:** int`functools.partial` and `partialmethod`: how many leading parameters (after `self`, for `partialmethod`) the call already gave, which the signature leaves out.**partial:** bool**partial\_method:** bool`partialmethod`: as got from the class, a function with no docstring.`pyscan.Object_Kind :: enum u8`**None**

not found.

**Module****Class****Function**

**Value**

anything an assignment made: value is its expression.

**External**

from a module outside the source roots: only path is known.

**Builtin**

a builtin such as str or Exception: path is “builtins.<name>”.

`pyscan.Param :: struct`

**name:** string

**kind:** Param\_Kind

**annotation:** ts.Node

the type node, or null.

**default:** ts.Node

the default expression, or null.

**type\_comment:** string

“a, # type: str”: a per-parameter type comment, or “”.

`pyscan.Param_Kind :: enum u8`

**Positional\_Only**

**Positional\_Or\_Keyword**

**Var\_Positional**

**Keyword\_Only**

**Var\_Keyword**

`pyscan.Project :: struct`

Project finds and analyses modules below its source roots, each once.

**roots:** []string

**reader:** File\_Reader

a nil procedure reads with core:os.

**parser:** ^ts.Parser

**modules:** map[string]^Module

by dotted name; nil records a module not found.

**files:** [dynamic]File\_Read

every source read, in order: a build's dependencies.

**trees:** [dynamic]^ts.Tree

**steps:** int

The lookup in progress: its steps (MAX\_STEPS), where its stack starts (LOOKUP\_STACK\_BYTES), and how deep lookups and MRO computations are nested.

**stack\_base:** uintptr

**resolving:** int

**mro\_depth:** int

pyscan.**Scope** :: struct

**defs:** [dynamic]Def

pyscan.**Slot** :: struct

Slot is one entry of a class's `__slots__`, with the docstring a dictionary gives it.

**name:** string

**doc:** Doc

`pyscan.String_Kind :: enum u8`

**Str**

**Bytes**

**Formatted**

an f-string or t-string: not a constant.

`pyscan.Unparse_Mode :: enum u8`

Unparse\_Mode chooses whose unparser to follow.

**Sphinx**

Sphinx's `sphinx.pycode.ast.unparse`, which `preserve_defaults` and `@overload` signatures use: numbers as written when source is given, lambda bodies as "...", and no parentheses added for precedence.

**Sphinx\_Repr**

Sphinx's unparser without the source, as for overload signatures: numbers as repr.

**Python**

Python's own `ast.unparse`, which makes the annotation strings of `from __future__ import annotations`: parentheses kept, every expression.

`pyscan Wrapper :: enum u8`

Wrapper is what the decorators of a function made of it.

**None**

**Staticmethod**

**Classmethod**

**Property**

**Cached\_Property**

**Unknown**

a decorator from outside the source roots that is not known to keep functions.

**Procedures**

```
pyscan.base_objects :: proc(c: ^Class) -> [dynamic]Object
```

base\_objects resolves the bases a class statement lists; subscripted generics give their origin.

```
pyscan.class_attr :: proc(c: ^Class, name: string, depth := 0) -> Object
```

class\_attr is getattr(cls, name): the first class along the MRO whose body binds name.

```
pyscan.def_object :: proc(m: ^Module, owner: ^Class, d: ^Def, depth: int) -> Object
```

def\_object turns a binding into the object it holds.

```
pyscan.destroy :: proc(p: ^Project)
```

destroy frees the trees and the parser. Everything else is in the caller's allocator.

```
pyscan.exported :: proc(m: ^Module, name: string) -> bool
```

exported is whether `from m import *` binds name: `__all__` when m has one, else the public names.

```
pyscan.find_def :: proc(s: ^Scope, name: string) -> ^Def
```

find\_def returns the binding of name that is in force when the body has run: the last runtime one; nil when there is none.

```
pyscan.get_attr :: proc(obj: Object, name: string, depth := 0) -> Object
```

get\_attr is getattr(obj, name) as far as the source tells.

```
pyscan.init :: proc(p: ^Project, roots: []string, reader := File_Reader{}) -> bool
```

init prepares a project over roots (folders, absolute or relative to the working directory). It returns false when tree-sitter cannot read the Python grammar.



```
pyscan.is_builtin :: proc(name: string) -> bool
```

is\_builtin lists the names of Python's builtins module that matter to documentation.

```
pyscan.is_builtin_exception :: proc(name: string) -> bool
```

is\_builtin\_exception lists the builtin exception classes.

```
pyscan.is_exception :: proc(c: ^Class) -> bool
```

is\_exception is whether a class derives from BaseException, as far as its MRO is known.

```
pyscan.lookup_name :: proc(code: ^Module, scope: ^Class,
name: string, depth := 0) -> Object
```

lookup\_name resolves a name where a class body (scope) or the module runs: the class body first, then the module, then builtins.

```
pyscan.module :: proc(p: ^Project, name: string) -> ^Module
```

module finds, reads, and analyses a module by dotted name, once; nil when no root has it.

```
pyscan.module_attr :: proc(m: ^Module, name: string, depth :=
0) -> Object
```

module\_attr looks a name up in a module's namespace, then among its submodules.

```
pyscan.module_object :: proc(m: ^Module) -> Object
```

module\_object is the object of a module.

```
pyscan.mro :: proc(c: ^Class) -> (order: [dynamic]Mro_Entry,
complete: bool)
```

mro is the class's method resolution order, by C3 linearisation, ending with object. complete is false when a base could not be resolved or lies outside the roots (it then appears by path, and what it defines is unknown).

```
pyscan.number_repr :: proc(literal: string) -> string
```

number\_repr is Python's repr of a numeric literal: integers in decimal, floats in their shortest round-trip form, imaginary numbers with j.

```
pyscan.parse_expression :: proc(m: ^Module, s: string) ->
(^Module, ts.Node, bool)
```

parse\_expression parses text, such as a string annotation, as an expression whose names resolve in m. The module returned holds the snippet's source.

```
pyscan.prepare_docstring :: proc(s: string, tab_width := 8) -
> [dynamic]string
```

prepare\_docstring is Sphinx's sphinx.util.docstrings.prepare\_docstring: lines of reST with the common indentation of the lines after the first removed, the first line stripped, leading blank lines dropped, and a blank line at the end.

```
pyscan.reference :: proc(m: ^Module, scope: ^Class, n:
ts.Node, depth := 0) -> Object
```

reference evaluates an expression that names an object: a name, or attributes of one, or a subscripted generic (its origin). Other expressions give None.

```
pyscan.repr_bytes :: proc(s: string) -> string
```

repr\_bytes is Python's repr of bytes.

```
pyscan.repr_str :: proc(s: string) -> string
```

repr\_str is Python's repr of a str: single quotes unless the text has one and no double quote, and escapes for what is not printable.

```
pyscan.slots :: proc(c: ^Class) -> (out: [dynamic]Slot, ok:
bool)
```

slots is inspect's getslots: the \_\_slots\_\_ the class has (its own, or the nearest base's), as a list, tuple, dictionary, or single string of names. ok is false when there are none, or they are computed.

```
pyscan.split_lines :: proc(s: string) -> [dynamic]string
```

split\_lines is str.splitlines for \n, \r\n, \r and the other line boundaries Python knows; a final line break does not start another line.

```
pyscan.str_value :: proc(m: ^Module, n: ts.Node) -> (string,
bool)
```

str\_value is string\_value for str constants only.

```
pyscan.string_value :: proc(m: ^Module, n: ts.Node) ->
(value: string, kind: String_Kind, ok: bool)
```

string\_value evaluates a string or concatenated\_string node as Python does: prefixes, escape sequences, and adjacent literals joined. ok is false for other nodes.

```
pyscan.text :: proc(m: ^Module, n: ts.Node) -> string
```

text is the source a node spans.

```
pyscan.try_unparse :: proc(m: ^Module, n: ts.Node, mode :=
Unparse_Mode.Sphinx) -> (string, bool)
```

try\_unparse is unparse that reports expressions Sphinx’s unparser refuses (comparisons, comprehensions, f-strings, conditional expressions...).

```
pyscan.unparse :: proc(m: ^Module, n: ts.Node, mode :=
Unparse_Mode.Sphinx) -> string
```

unparse writes an expression back as Sphinx’s unparser does (Sphinx mode).

## readers/commonmark

Package commonmark reads Markdown into a Guidedoc document without calling an allocator, as two readers with distinct ids: “commonmark” (reader, read, read\_inline) is pure CommonMark 0.31.2 and takes no settings; “myst” (myst\_reader, read\_myst, read\_myst\_inline) is the MyST Markdown that Sphinx projects write (myst-parser 0.16.1): directives, roles, targets, comments, front matter, footnotes, GFM tables, and the optional syntax that Settings.enable\_extensions turns on.

Block structure is parsed line by line into a skeleton in scratch; link reference definitions are collected over the whole document; then the skeleton is emitted in preorder, with inline content parsed per paragraph and heading. Directive content is parsed the same way when the directive asks for it (see extension.odin), so a parse works on line views of any source.

### Types

```
commonmark.Directive :: struct
```

Directive is a directive the reader has split into its parts. Lines are views of source.

```
name: string
```

```
name_span: gd.Source_Span
```

**span:** `gd.Source_Span`

**source:** `gd.Source_Id`

**arg:** `[]Line`

the argument; empty when there is none.

**options:** `[]Option`

**content:** `[]Line`

**sections:** `bool`

the directive stands where sections are allowed.

**classes:** `gd.Text`

classes a handler leaves for the next element (Docutils' class directive without content); the reader gives them to the next block it emits.

`commonmark.Directive_Handler` *:: struct*

**name:** `string`

compared case-insensitively.

**run:** `Directive_Proc`

`commonmark.Directive_Proc` *:: #type proc(h: ^Host, d: ^Directive, user: rawptr) -> Directive\_Result*

`Directive_Proc` runs a parsed directive; `user` is `Extension.user`. The directive and its lines are borrowed for the call.

`commonmark.Directive_Result` *:: enum u8*

**Ran**

**Unknown**

the handler does not know the name; the reader reports it.

**Failed**

the handler refused the directive and reported why.

```
commonmark.Directive_Shape :: struct
```

Directive\_Shape tells the reader how a directive takes its first line: as an argument, or, for a directive without arguments, as the first line of content; and whether it takes options at all.

```
argument: bool
```

```
options: bool
```

it has options; without, an option block stays content.

```
commonmark.Extension :: struct
```

```
directives: []Directive_Handler
```

```
directive: Directive_Proc
```

directive is offered every directive the table does not list.

```
shape: proc(h: ^Host, name: string, user: rawptr) ->
(Directive_Shape, bool)
```

shape describes a directive by name; found is false for an unknown directive.

```
roles: []Role_Handler
```

```
role: Role_Proc
```

role is offered every role the table does not list.

```
table: proc(h: ^Host, id: gd.Node_Id, span:
gd.Source_Span, columns: int, user: rawptr) -> bool
```

table runs after the reader opens a Table node, before its rows, so a table directive around it can add its title and attributes; it returns true when it did, and the reader then adds no attributes of its own.

```
begin: proc(h: ^Host, user: rawptr)
```

before the document's blocks.

```
end: proc(h: ^Host, user: rawptr)
```

after them.

---

**resolve:** `proc(h: ^Host, user: rawptr) -> gd.Status`

resolve, when set, links references, targets, and footnotes instead of the reader's own resolution (resolve.odin); the Sphinx layer resolves as for reStructuredText.

---

**user:** `rawptr`

---

`commonmark.Host` *:: struct*

Host is the reader's document-wide state, shared by the parse of the document and the parses of nested content. Directive and role handlers receive it and treat it as opaque: they reach the read through context\_of and the procedures below.

---

**ctx:** `^gd.Read_Context`

---

**b:** `^gd.Builder`

---

**settings:** `Settings`

---

**ext:** `^Extension`

---

**myst:** `bool`

the MyST syntax is on: the "myst" reader, not "commonmark".

---

**disabled:** `Rules`

myst\_disable\_syntax.

---

**syntax:** `Syntax_Extensions`

myst\_enable\_extensions in force.

---

**refs:** `[]Ref`

Link reference definitions, collected over the whole document.

---

**nrefs:** `int`

---

**names:** `gd.Symbol_Table`

---

**labels:** `gd.Buffer`

```
label_tmp: []u8
```

```
html_format: gd.Text
```

```
rel_images: bool
```

MyST include options in force: rewrite image paths, and link paths starting with `rel_docs`, of included sources to be relative to the main source.

```
rel_docs: string
```

```
using state: Myst_State
```

```
commonmark.Line :: gd.Line_View
```

Line is a view of one input line: its text without the line break, and the source offset of `text[o]`. Directive handlers pass nested content as lines.

```
commonmark.Meta :: struct
```

Meta is one HTML meta tag: key as MyST writes it (“description lang=en”, “http-equiv=Content-Type”, “keywords”) and its content.

```
key: string
```

```
content: string
```

```
commonmark.Option :: struct
```

Option is one directive option: its name and value lines. A value written as a YAML scalar has its quoting and folding processed; its lines may then be copies in scratch, whose offset is where the value starts in the source.

```
name: string
```

```
value: []Line
```

```
span: gd.Source_Span
```

```
commonmark.Role :: struct
```

Role is one role, `{name}`content``. lines hold the whole construct, split at line breaks; content is the range of the content in the lines joined by “\n”.

<b>name:</b> string
---------------------

<b>lines:</b> []Line
----------------------

<b>content:</b> [2]int
------------------------

<b>span:</b> gd.Source_Span
-----------------------------

<b>source:</b> gd.Source_Id
-----------------------------

commonmark. <b>Role_Handler</b> :: struct
-------------------------------------------

<b>name:</b> string
---------------------

compared case-insensitively.

<b>run:</b> Role_Proc
-----------------------

commonmark. <b>Role_Proc</b> :: #type proc(h: ^Host, r: ^Role, user: rawptr) -> bool
--------------------------------------------------------------------------------------

Role\_Proc handles a role; it returns false for a role it does not know.

commonmark. <b>Rule</b> :: enum u8
------------------------------------

Rule names a syntax element that `myst_disable_syntax` can turn off, by markdown-it's rule name.

<b>Table</b>
--------------

<b>Code</b>
-------------

<b>Fence</b>
--------------

<b>Blockquote</b>
-------------------

<b>Hr</b>
-----------

<b>List</b>
-------------

<b>Reference</b>
------------------



---

**Html\_Block**

---

---

**Heading**

---

---

**Lheading**

---

---

**Emphasis**

---

---

**Backticks**

---

---

**Link**

---

---

**Image**

---

---

**Autolink**

---

---

**Html\_Inline**

---

---

**Entity**

---

---

**Escape**

---

---

**Newline**

---

---

**Front\_Matter**

---

---

**Myst\_Target**

---

---

**Myst\_Line\_Comment**

---

---

**Myst\_Block\_Break**

---

---

**Myst\_Role**

---

---

**Math\_Block**

---

---

**Math\_Inline**

---

**Footnote\_Def****Footnote\_Ref****Myst\_Directive**

not a markdown-it rule: {name} fences (off with `commonmark_only`).

```
commonmark.Settings :: struct
```

Settings are the “myst” reader’s settings: myst-parser 0.16.1’s `myst_*` configuration values, named without the prefix. They reach the reader through `gd.Read_Options.extension`, made with `gd.read_extension(&settings)`; an empty extension means `default_settings()`. The “commonmark” reader (pure CommonMark) takes none. A caller that supplies Settings starts from `default_settings()`, because several defaults are not zero.

```
extension: ^Extension
```

Directive and role handlers (the Sphinx layer provides them); nil means none, and directives and roles are then reported as unknown.

```
commonmark_only: bool
```

`myst_commonmark_only`: CommonMark syntax only (no directives, roles, or other MyST syntax), but a document’s tree as for MyST: headings make sections.

```
enable_extensions: Syntax_Extensions
```

`myst_enable_extensions`: the optional syntax. Default `{.Dollarmath}`.

```
disable_syntax: []string
```

`myst_disable_syntax`: markdown-it rule names whose syntax is not parsed, e.g. “emphasis”, “table”, “html\_block” (see `syntax_rule`).

```
url_schemes: []string
```

`myst_url_schemes`: schemes of `[text](scheme:...)` links that are external; other destinations are cross-references. nil recognizes every scheme. Default `http, https, mailto, and ftp`.

```
linkify_fuzzy_links: bool
```

`myst_linkify_fuzzy_links`: linkify recognizes URLs without a scheme. Default `true`.

**heading\_anchors:** int

myst\_heading\_anchors: headings up to this level get a GitHub-style slug anchor that `[](#slug)` and `[](file.md#slug)` reach. 0 means none.

**heading\_slug\_func:** string

myst\_heading\_slug\_func is a Python callable and cannot be supported; a non-empty value is reported and the built-in slugs are used.

**substitutions:** []Substitution

myst\_substitutions: global substitutions; front matter overrides them.

**sub\_delimiters:** [2]u8

myst\_sub\_delimiters: the characters doubled around substitution references. Default `{‘, ’}`, giving `{{ name }}`.

**html\_meta:** []Meta

myst\_html\_meta: HTML meta tags for every document; front matter html\_meta overrides entries with the same key.

**footnote\_transition:** bool

myst\_footnote\_transition: a transition of class “footnotes” precedes the footnotes at the end of the document. Default true.

**dmath\_allow\_labels:** bool

myst\_dmath\_allow\_labels: `$$...$$ (label)`. Default true.

**dmath\_allow\_space:** bool

myst\_dmath\_allow\_space: inline math may start or end with a space. Default true.

**dmath\_allow\_digits:** bool

myst\_dmath\_allow\_digits: inline math may follow or precede a digit. Default true.

**dmath\_double\_inline:** bool

myst\_dmath\_double\_inline: `$$...$$` inside a paragraph. Default false.

**words\_per\_minute:** int

myst\_words\_per\_minute: for the wordcount-minutes substitution. Default 200.

**docname:** string

The document’s name (“usage/intro”) and configuration values, which substitutions reach as env.docname and env.config.<name>.

**env\_config:** []Substitution

**today:** string

The {sub-ref} today: the project’s formatted date; empty is Read\_Options.today.

commonmark.**Substitution** :: struct

A substitution’s value is Markdown, or a mapping written as nested YAML that dotted names reach (“key.sub”).

**name:** string

**value:** string

commonmark.**Syntax\_Extension** :: enum u8

Syntax\_Extension names an entry of myst\_enable\_extensions.

**Amsmath**

**Colon\_Fence**

**Deflist**

**Dollarmath**

**Fieldlist**

**Html\_Admonition**

**Html\_Image**

**Linkify****Replacements****Smartquotes****Substitution****Tasklist**

```
commonmark.Syntax_Extensions :: bit_set[Syntax_Extension;
u16]
```

**Procedures**

```
commonmark.context_of :: proc(h: ^Host) -> ^gd.Read_Context
```

context\_of returns the read context, for handlers that build nodes.

```
commonmark.default_settings :: proc() -> Settings
```

default\_settings returns myst-parser 0.16.1's defaults. The value borrows only static data (the default URL schemes); fields a caller sets must outlive the read.

```
commonmark.include_paths :: proc(h: ^Host, images: bool,
docs: string) -> (bool, string)
```

include\_paths sets MyST's include options :relative-images: and :relative-docs: for the content of the include directive being run; it returns the previous values.

```
commonmark.list_items :: proc(h: ^Host, lines: []Line,
source: gd.Source_Id) -> (items: [][]Line, starts: []int,
ok: bool)
```

list\_items splits lines that hold one bullet list into the content lines of its items, and where each item starts in lines (for list-table). ok is false when the lines are anything but one bullet list. The results live in scratch.

```
commonmark.myst_reader :: proc() -> gd.Reader
```

myst\_reader returns the “myst” reader by value: MyST Markdown with ^Settings.

```
commonmark.parse_body :: proc(h: ^Host, lines: []Line,
source: gd.Source_Id, sections := false, classes :=
gd.Text{})
```

`parse_body` parses lines of a source as Markdown body content of the builder's open node. `sections` lets headings open sections, as at the top of a document; `classes`, when set, go to every top-level element.

```
commonmark.parse_inline :: proc(h: ^Host, lines: []Line,
source: gd.Source_Id)
```

`parse_inline` parses lines of a source as Markdown inline content of the builder's open node.

```
commonmark.read :: proc(ctx: ^gd.Read_Context) -> gd.Status
```

`read` reads `ctx.source` as pure CommonMark 0.31.2 (`gd.Read_Proc`): headings, no sections. Any settings in the read options are refused with `Invalid_Input`. It borrows `ctx` for the call; its skeleton lives in scratch, released before it returns. Problems go to `ctx.reports`; the result is the builder's status (`.Capacity` when a table is full).

```
commonmark.read_inline :: proc(ctx: ^gd.Read_Context, span:
gd.Source_Span) -> gd.Status
```

`read_inline` parses a span of `ctx.source` as CommonMark inline content of the builder's open node, for a translated message (`gd.Read_Inline_Proc`). Link reference definitions and footnote definitions of the surrounding document are not known to it.

```
commonmark.read_myst :: proc(ctx: ^gd.Read_Context) ->
gd.Status
```

`read_myst` reads `ctx.source` as MyST Markdown (`gd.Read_Proc`), with the `^Settings` the read options carry (`gd.read_extension`), or `default_settings` when they carry none. Settings of another type are refused with `Invalid_Input`. Ownership and failures are read's; the settings are borrowed for the call.

```
commonmark.read_myst_inline :: proc(ctx: ^gd.Read_Context,
span: gd.Source_Span) -> gd.Status
```

`read_myst_inline` is `read_inline` for MyST, with the settings of `ctx.options.extension` (roles included), as `read_myst` takes them.

```
commonmark.reader :: proc() -> gd.Reader
```

`reader` returns the “commonmark” reader by value: pure CommonMark 0.31.2, no settings.

```
commonmark.syntax_extension_named :: proc(name: string) ->
(Syntax_Extension, bool)
```

syntax\_extension\_named finds a myst\_enable\_extensions entry by its name.

```
commonmark.syntax_rule :: proc(name: string) -> (Rule, bool)
```

syntax\_rule finds a myst\_disable\_syntax entry by its markdown-it rule name.

### Constants

```
commonmark.CONFORMANCE :: gd.Conformance.Conformant
```

```
commonmark.MYST_CONFORMANCE :: gd.Conformance.Partial
```

```
commonmark.MYST_VERSION :: "0.16.1"
```

MyST follows myst-parser 0.16.1, with the differences CONFORMANCE.md documents.

```
commonmark.VERSION :: "0.31.2"
```

## readers/rst

Package rst reads reStructuredText as the Docutils markup specification defines it, implemented independently. It owns lines, block structure, and directives; inline markup and reference resolution live in the inline subpackage.

### Types

```
rst.Arg_Rule :: enum u8
```

None

Optional

Required

```
rst.Content_Rule :: enum u8
```

None

Optional

Required

```
rst.Dir_Opt :: struct
```

```
spec: Opt_Spec
```

```
lines: []Line
```

the value: the rest of the marker line, then continuation lines.

```
span: gd.Source_Span
```

```
rst.Dir_Spec :: struct
```

```
name: string
```

```
kind: Dir_Kind
```

```
arg: Arg_Rule
```

```
final_ws: bool
```

```
content: Content_Rule
```

```
opts: Opt_Set
```

```
value: gd.Admonition_Kind
```

```
rst.Directive :: struct
```

```
spec: ^Dir_Spec
```

```
span: gd.Source_Span
```

```
name_span: gd.Source_Span
```

```
arg: []Line
```

argument lines, trimmed; empty when there is no argument.

```
opts: [MAX_OPTIONS]Dir_Opt
```

```
nopts: int
```



**content:** []Line

**sections:** bool

**subst:** bool

inside a substitution definition.

**ext:** ^Directive\_Handler

a dialect's handler, or nil for a standard directive.

**arg\_block:** []Line

the argument lines with their relative indentation.

**subst\_name:** string

the name of the substitution definition it is in, as written.

**rst.Directive\_Handler** :: struct

**name:** string

compared case-insensitively, as directive names are.

**arg:** Arg\_Rule

**final\_ws:** bool

the last argument may contain white space.

**content:** Content\_Rule

**options:** []Opt\_Spec

**run:** Directive\_Proc

**rst.Directive\_Proc** :: #type proc(p: ^Parser, d: ^Directive, user: rawptr)

Directive\_Proc runs a parsed directive. user is Extension.user.

**rst.Directive\_Shape** :: struct

Directive\_Shape tells a host how a directive takes its parts.

---

**arg:** Arg\_Rule

---

**final\_ws:** bool

---

the last argument may contain white space.

---

**content:** Content\_Rule

---

**options:** []Opt\_Spec

---



---

**rst.Extension** :: struct

---

**directives:** []Directive\_Handler

---

**lookup:** proc(name: string, user: rawptr) ->  
^Directive\_Handler

---

lookup, when set, finds handlers the table does not list, such as names that depend on reader state (a default domain). It returns nil for unknown names.

---

**roles:** []Role\_Handler

---

**role:** Role\_Proc

---

role, when set, is offered every role the table does not list.

---

**begin:** Hook\_Proc

---

before the main source's blocks (reader state).

---

**prologue:** Hook\_Proc

---

prologue runs after the main source's leading field lines, where Sphinx inserts rst\_prolog: after a document's bibliographic fields, else before everything.

---

**end:** Hook\_Proc

---

after them, before sections close (e.g. an epilogue).

---

**substitution:** proc(name: string, user: rawptr) -> (text:  
string, ok: bool)

---

substitution, when set, supplies default substitutions: resolution offers it each reference whose name no definition in the document has (see

`rst_inline.Substitution_Hook`). It runs after parsing, with the extension's user.

```
user: rawptr
```

```
rst.Frame :: struct
```

```
src: gd.Source_Id
```

```
text: string
```

```
tab: int
```

```
rst.Hook_Proc :: #type proc(p: ^Parser, user: rawptr)
```

`Hook_Proc` runs at a fixed point of the read, with the reader's state.

```
rst.Host :: struct
```

Host procedures receive `Host.user`. Lines are views of the current source (`p.src`), as the block parser's are, so every node keeps exact spans.

```
body: proc(p: ^Parser, lines: []Line, sections: bool,
each: gd.Text, user: rawptr)
```

`body` parses nested body content of the open node; `sections` allows headings to make sections, and `each`, when set, gives these classes to every top-level element (the class directive).

```
inline: proc(p: ^Parser, lines: []Line, user: rawptr)
```

`inline` parses inline content of the open node.

```
items: proc(p: ^Parser, lines: []Line, user: rawptr) ->
(items: [][]Line, starts: []int, ok: bool)
```

`items` splits lines that hold one bullet list into its items' content, with where each item starts in lines (list-table, with the host's list syntax); `ok` is false when the lines are not one bullet list.

```
user: rawptr
```

```
rst.Host_Directive :: struct
```

`Host_Directive` is a directive the host has split into its parts.

**name:** string

**name\_span:** gd.Source\_Span

**span:** gd.Source\_Span

**arg:** []Line

the argument lines; empty when there is none.

**options:** []Host\_Option

**content:** []Line

**sections:** bool

the directive stands where sections are allowed.

**rst.Host\_Option** :: struct

Host\_Option is a directive option as the host wrote it: its name and value lines.

**name:** string

**value:** []Line

**span:** gd.Source\_Span

**rst.Hosted\_Result** :: enum u8

**Ran**

**Unknown**

no directive of that name; nothing was reported.

**Failed**

the arguments, options, or content were refused and reported.

**rst.Line** :: struct

Line is a view of one input line in columns: tabs are expanded to tab stops and trailing white space is stripped. A line that held a tab, form feed, or vertical tab is a copy in scratch; raw then keeps the source line so columns map back

to source bytes. Nested blocks use cut views of their parent's lines, so every node keeps exact spans.

**text:** string

**raw:** string

the source line of an expanded copy, else "".

**offset:** u32

source offset of text[o]; of raw[o] when raw is set.

**col:** i32

column of text[o] in the expanded line when raw is set.

**rst.Opt\_Spec** :: struct

**name:** string

**type:** Opt\_Type

**choices:** string

space-separated, for Choice.

**rst.Opt\_Type** :: enum u8

**Flag**

**Text**

**Text\_Required**

**Int**

**Nonneg\_Int**

**Pos\_Int**

**Opt\_Int**

**Length****Length\_Percent****Figwidth****Percentage****Class****Name****Choice****Encoding****Char****Widths****Path****rst.Parser** *:: struct***ctx:** ^gd.Read\_Context**b:** ^gd.Builder**inl:** rst\_inline.Inline\_State**src:** gd.Source\_Id

the source the current lines come from.

**text:** string**tab:** int**depth:** int

**stack:** `gd.Stack_Guard`

bounds the machine stack of the recursion (enter).

**styles:** `[MAX_LEVELS]Style`

**styles\_used:** `int`

**sections:** `[MAX_LEVELS]Section_Entry`

**sections\_used:** `int`

**section\_base:** `int`

sections open before titled content (`parse_titled`) began.

**level\_base:** `int`

the level of the last of them: titled content's sections nest in it.

**top\_count:** `int`

top-level elements since the current section began.

**top\_transition:** `bool`

the last top-level element was a transition.

**transition\_span:** `gd.Source_Span`

**pending:** `gd.Text`

classes from a class directive, for the next element.

**pending\_span:** `gd.Source_Span`

**has\_pending:** `bool`

**in\_sidebar:** `int`

**ext:** `^Extension`

a dialect's directives and roles; see `extension.odin`.

---

**host:** `^Host`

another markup that uses the directives; see `host.odin`.

---

**hosted:** `bool`

the running directive was written in the host markup.

---

**table\_extra:** `^Table_Extra`

a table directive's title and options for the host.

---

**`rst.Role_Handler`** `:: struct`


---

**name:** `string`

compared case-insensitively.

---

**run:** `Role_Proc`


---

**`rst.Role_Proc`** `:: #type proc(p: ^Parser, call: ^rst_inline.Role_Call, user: rawptr) -> bool`

`Role_Proc` handles interpreted text; it returns false for a role it does not know.

---

**`rst.Settings`** `:: struct`

Settings are the reader's dialect settings. They reach the reader through `gd.Read_Options.extension`, made with `gd.read_extension(&settings)`; settings of another type are refused. A dialect's own configuration embeds `Settings` and passes a pointer to that embedded field when it runs this reader. An empty extension means defaults (`default_settings`); a caller that supplies `Settings` starts from `default_settings`, because the zero value turns bibliographic fields off.

---

**extension:** `^Extension`

Directive and role handlers consulted before the standard ones.

---

**host:** `^Host`

Another markup (MyST Markdown) that reads the main source and nested directive content with these directives and roles; see `host.odin`.

---

**pep\_references:** `bool`

Implicit "PEP nnn" and "RFC nnn" references in running text become links.



**rfc\_references:** bool

**pep\_base\_url:** string

“” means <https://peps.python.org/>

**rfc\_base\_url:** string

“” means <https://datatracker.ietf.org/doc/html/>

**trim\_footnote\_reference\_space:** bool

White space before a footnote reference is removed.

**smart\_quotes:** Smart\_Quotes

Typographic quotation marks, apostrophes, dashes, and ellipses in text.

**smart\_quotes\_locales:** string

Quotation marks per language, overriding the built-in ones, as “tag: “”,tag2:«»<>”: primary open and close, then secondary open and close.

**language:** string

BCP 47 tag of the document language: bibliographic field names, generated titles, and quotation marks. “” means “en”.

**docinfo:** bool

A field list first in the document becomes bibliographic data (docinfo).

**sectsubtitle:** bool

A lone subsection title becomes the section’s subtitle.

**tab\_width:** int

Tab stops; zero means `gd.Read_Options.tab_width`, which zero makes 8.

**character\_level\_inline\_markup:** bool

Inline markup is recognized inside words (Docutils’ `character_level_inline_markup`).

`rst.Smart_Quotes :: rst_inline.Smart_Quotes`

```
rst.Table_Extra :: struct
```

Table\_Extra carries what table directives add to a table: a title and attributes.

```
title: []Line
```

```
classes: gd.Text
```

```
name: gd.Text
```

```
align: string
```

```
width: string
```

```
widths: []int
```

from :widths:: nil keeps the table's own widths.

```
auto_widths: bool
```

:widths: auto leaves widths to the renderer.

```
width_class: string
```

“colwidths-auto” or “colwidths-given” from :widths:.

```
found: bool
```

set by the table parser when it emits a table.

### Procedures

```
rst.add_common :: proc(p: ^Parser, id: gd.Node_Id, d:
^Directive, class_opt := "class")
```

add\_common adds the :class: and :name: options as attributes. Directive nodes keep their Name attribute for the directive name, so :name: becomes Custom “name”.

```
rst.blank :: proc(l: Line) -> bool
```

blank tells whether l has no text (white space is stripped when lines are split).

```
rst.bullet_width :: proc(t: string) -> int
```

bullet\_width returns the byte length of a bullet that starts t, or 0.

---

```
rst.class_directive :: proc(p: ^Parser, d: ^Directive)
```

---

class\_directive runs the class directive: its classes go to the content's elements, or, without content, to the next element; invalid names are reported.

---

```
rst.close :: proc(p: ^Parser, id: gd.Node_Id)
```

---

close ends the element open returned id for.

---

```
rst.code_text :: proc(p: ^Parser, lines: []Line) -> gd.Text
```

---

code\_text writes lines to the text pool, each ending in “\n”.

---

```
rst.custom :: proc(p: ^Parser, id: gd.Node_Id, name, value: string)
```

---

custom adds a Custom attribute name=value to node id; both strings are copied to the text pool. A full pool poisons the builder, as every builder write does.

---

```
rst.cut :: proc(l: Line, k: int) -> Line
```

---

cut drops the first k columns.

---

```
rst.dedent :: proc(lines: []Line)
```

---

dedent strips the least indentation of the non-blank lines, in place.

---

```
rst.default_settings :: proc() -> Settings
```

---

default\_settings returns Docutils' defaults: bibliographic fields on, everything else off or empty. The value owns nothing.

---

```
rst.directive_node :: proc(p: ^Parser, d: ^Directive, name: string) -> gd.Node_Id
```

---

directive\_node opens a generic Directive node named after the directive.

---

```
rst.directive_shape :: proc(p: ^Parser, name: string) -> (shape: Directive_Shape, found: bool)
```

---

directive\_shape looks a directive up as the reader would, a dialect's handlers first.

---

```
rst.embed_begin :: proc(p: ^Parser, ctx: ^gd.Read_Context)
```

---

embed\_begin prepares p for a host that reads the main source itself: the settings of ctx.options.extension apply as for read, and the dialect's begin hook runs.

```
rst.embed_end :: proc(p: ^Parser) -> gd.Status
```

`embed_end` runs the dialect's end hook and resolves the document as `read` does.

```
rst.enter_source :: proc(p: ^Parser, id: gd.Source_Id, tab: int) -> Frame
```

`enter_source` makes lines of another source current, for spans and inline parsing.

```
rst.find_role :: proc(p: ^Parser, name: string) -> int
```

`find_role` returns the index of the role the document defined with the role directive under `name`, in `p.inl.roles`, or -1. The search spends work budget.

```
rst.get_opt :: proc(d: ^Directive, name: string) -> ^Dir_Opt
```

`get_opt` returns the option called `name` that the directive was given, or nil. The result points into `d` and lives as long as it.

```
rst.has_option :: proc(d: ^Directive, name: string) -> bool
```

`has_option` reports whether a directive was given an option, flags included.

```
rst.head :: proc(l: Line, n: int) -> Line
```

`head` keeps the first `n` columns, without trailing spaces.

```
rst.host_lines :: proc(p: ^Parser, views: []rst_inline.Line) -> ([]Line, bool)
```

`host_lines` makes lines of views of source text (a host's lines): tabs are expanded to the parser's tab stops, as `split_lines` does.

```
rst.host_views :: proc(p: ^Parser, lines: []Line) -> ([]rst_inline.Line, bool)
```

`host_views` gives the host the source text of lines, tabs unexpanded (`inline_view`).

```
rst.indent_of :: proc(s: string) -> int
```

`indent_of` counts the spaces `s` starts with.

```
rst.inline :: proc(p: ^Parser, lines: []Line)
```

`inline` parses lines as inline content of the currently open node.

```
rst.inline_node :: proc(p: ^Parser, kind: gd.Node_Kind,
lines: []Line)
```

`inline_node` adds a node of kind holding lines parsed as inline content (`inline`); nothing for no lines.

```
rst.is_indented :: proc(l: Line) -> bool
```

`is_indented` tells whether `l` starts with a space.

```
rst.join_lines :: proc(p: ^Parser, lines: []Line, sep:
string) -> string
```

`join_lines` joins lines into a scratch copy, or returns the single line's text.

```
rst.leave_source :: proc(p: ^Parser, f: Frame)
```

`leave_source` makes the source that `enter_source` returned `f` for current again.

```
rst.line_span :: proc(p: ^Parser, l: Line) -> gd.Source_Span
```

`line_span` is the source span of one line (`span_of`).

```
rst.lines_span :: proc(p: ^Parser, lines: []Line) ->
gd.Source_Span
```

`lines_span` is the source span of lines without trailing blank lines; empty lines give an empty span at the start of the source.

```
rst.mark :: proc(p: ^Parser) -> gd.Scratch_Mark
```

`mark` records the scratch position; `release` returns scratch to it. Pair them with `defer` around work that takes scratch.

```
rst.open :: proc(p: ^Parser, kind: gd.Node_Kind, span:
gd.Source_Span) -> gd.Node_Id
```

`open` begins an element and gives it the classes of a preceding class directive.

```
rst.opt_value :: proc(p: ^Parser, d: ^Directive, name:
string) -> (string, bool)
```

`opt_value` returns an option's value lines joined by spaces and trimmed; `ok` is false when the option was not given. The text is a view of the source or a scratch copy, valid until the caller's scratch mark is released.

```
rst.option_lines :: proc(d: ^Directive, name: string) ->
([]Line, bool)
```

`option_lines` returns the value lines of a directive option, or `nil`.

```
rst.parse_blocks :: proc(p: ^Parser, lines: []Line,
sections := false, each := gd.Text{})
```

`parse_blocks` parses body elements. `sections` allows titles and transitions; each gives classes to every element parsed at this level (class directive content).

```
rst.parse_body :: proc(p: ^Parser, lines: []Line, sections :=
false, each := gd.Text{})
```

`parse_body` parses nested body content of the open node in the markup of the running directive (see `Host`). Outside hosted directives it is `parse_blocks`.

```
rst.parse_source :: proc(p: ^Parser, id: gd.Source_Id,
sections: bool)
```

`parse_source` parses a whole registered source (`gd.add_source`) as body content at the current position, as if its lines stood there. `sections` allows titles.

```
rst.parse_titled :: proc(p: ^Parser, lines: []Line)
```

`parse_titled` parses content in which titles begin sections, in a title context of its own (Sphinx's nested parse with titles): adornments take levels afresh, the first nesting in the innermost open section, and the sections close when the content ends. The surrounding adornments are kept for what follows.

```
rst.path_of :: proc(p: ^Parser, lines: []Line) -> string
```

`path_of` joins path lines without white space, as the specification treats paths.

```
rst.place_titled :: proc(p: ^Parser, lines: []Line)
```

`place_titled` prepares for titled content (`parse_titled`) as Sphinx's only directive does: when the content's first title has an adornment the document already uses, the content stands where a title of that level would, so deeper sections close first.

```
rst.pool_classes :: proc(p: ^Parser, s: string) -> (gd.Text,
bool)
```

`pool_classes` writes white-space separated class names as identifiers, one space apart. It fails when a name has no letters.

```
rst.pool_lines :: proc(p: ^Parser, lines: []Line, sep:
string) -> gd.Text
```

`pool_lines` writes the lines joined by `sep` into the text pool and returns the text, which lives as long as the document; a full pool poisons the builder.

---

```
rst.pool_lower :: proc(p: ^Parser, s: string) -> string
```

---

pool\_lower writes s in lower case to the text pool and returns the pooled string.

---

```
rst.push_lines :: proc(p: ^Parser, lines: []Line, sep: string)
```

---

push\_lines writes the lines joined by sep into the text pool.

---

```
rst.read :: proc(ctx: ^gd.Read_Context) -> gd.Status
```

---

read reads ctx.source as reStructuredText into ctx.b (gd.Read\_Proc). Settings of another type than ^Settings are refused with Invalid\_Input before anything is read. It borrows ctx for the call and keeps nothing; its tables live in scratch, released before it returns. Problems go to ctx.reports; the result is the builder's status (.Capacity when a table is full, .Limit past the nesting or stack limits), or that of reference resolution.

---

```
rst.read_inline :: proc(ctx: ^gd.Read_Context, span: gd.Source_Span) -> gd.Status
```

---

read\_inline parses a span of ctx.source as inline content of the builder's open node, with the settings of ctx.options.extension, as Sphinx parses a translated message (gd.Read\_Inline\_Proc). Each line's leading white space is dropped, as the lines of a paragraph lose their indentation. References are left for the caller to resolve.

---

```
rst.reader :: proc() -> gd.Reader
```

---

reader returns the "rst" reader descriptor, for a gd.Registry. It is a value with no state: registering it twice or copying it is harmless.

---

```
rst.release :: proc(p: ^Parser, m: gd.Scratch_Mark)
```

---

release frees the scratch taken since mark m; views into it become invalid.

---

```
rst.report :: proc(p: ^Parser, m: gd.Message, span: gd.Source_Span, a := gd.Diagnostic_Arg{}, b := gd.Diagnostic_Arg{})
```

---

report adds diagnostic m at span, with its arguments, to the read's reports; a full reports region counts it as suppressed (gd.report) rather than failing the read.

```
rst.run_hosted :: proc(p: ^Parser, h: Host_Directive) ->
Hosted_Result
```

run\_hosted checks and runs a directive written in the host markup. Options are checked as in reStructuredText; a host with other rules checks them first (directive\_shape, option\_problem) and passes only the ones it keeps.

```
rst.span_of :: proc(p: ^Parser, first, last: Line) ->
gd.Source_Span
```

span\_of is the source span from the start of first to the end of last, in the current source, clamped to its text.

```
rst.standard_role :: proc(name: string) ->
(rst_inline.Role_Base, bool)
```

standard\_role finds a standard role or alias by name, compared case-insensitively; ok is false (base .Custom) for another name.

```
rst.table_decorate :: proc(p: ^Parser, id: gd.Node_Id,
span: gd.Source_Span, cols: int, widths: []int, extra:
^Table_Extra)
```

table\_decorate gives an open Table, before its rows, its widths and a table directive's attributes and title. A host calls it for the table it emits inside a table directive (Parser.table\_extra).

```
rst.take :: proc(p: ^Parser, $T: typeid, n: int) -> ([]T,
bool)
```

take returns n zeroed items of T from scratch, valid until the enclosing mark is released. When scratch is short it records the capacity need on the builder (poisoning it) and returns false.

```
rst.trim_blank_end :: proc(lines: []Line) -> []Line
```

trim\_blank\_end returns lines without its trailing blank lines (a subslice).

### Constants

```
rst.CONFORMANCE :: gd.Conformance.Partial
```

```
rst.DIRECTIVE_UNKNOWN
```

DIRECTIVE\_UNKNOWN is what the reader reports for a directive no handler knows; a handler that finds it cannot serve a directive it was given reports it too.

```
rst.VERSION :: "0.1.0"
```



## readers/rst/inline

Package `rst_inline` parses reStructuredText inline markup and resolves references. The block reader in `lib/readers/rst` owns lines, directives, and document structure; it calls `parse_inline` for every inline context and resolve once parsing ends.

### Types

```
inline.Inline_State :: struct
```

`Inline_State` is shared, long-lived reader state for inline parsing.

```
ctx: ^gd.Read_Context
```

```
source: gd.Source_Id
```

the source the current lines come from.

```
roles: []Role
```

custom roles; storage owned by the block reader.

```
roles_used: int
```

```
default_role: string
```

“” means title-reference.

```
hook: Role_Hook
```

a dialect's roles, consulted first; see `extension.odin`.

```
options: Options
```

settings of the block reader; see `options.odin`.

```
inline.Line :: gd.Line_View
```

`Line` is a view of one line of text. `offset` is the source byte offset of `text[o]`; for a line copied during tab expansion it is the offset of the original line start.

```
inline.Options :: struct
```

`Options` are the settings of the block reader (`rst.Settings`) that inline parsing and resolution need.

```
pep_references: bool
```

**rfc\_references:** bool

**pep\_base\_url:** string

**rfc\_base\_url:** string

**trim\_footnote\_reference\_space:** bool

**smart\_quotes:** Smart\_Quotes

**smart\_quotes\_locales:** string

**language:** string

**docinfo:** bool

**sectsubtitle:** bool

**character\_level:** bool

**tab\_width:** int

tab stops of the main source; 0 means 8.

**default\_substitution:** Substitution\_Hook

**inline.Role** :: struct

Role is a role defined by the role directive, or a standard role.

**name:** string

normalized (lowercase) role name.

**base:** Role\_Base

**classes:** string

space-separated, from :class:.

**format:** string

raw roles: the output format, e.g. "html".

**language:** string

code-derived roles.

`inline.Role_Base :: enum u8`

**Emphasis**

**Strong**

**Literal**

**Code**

**Math**

**Subscript**

**Superscript**

**Title\_Reference**

**Pep\_Reference**

**Rfc\_Reference**

**Abbreviation**

**Acronym**

**Raw**

**Custom**

a Role node carrying the role name and classes.

`inline.Role_Call :: struct`

Role\_Call describes one interpreted text for an extension’s role handler. The strings live in scratch and are valid only during the call.

**state:** ^Inline\_State

**b:** `^gd.Builder`

**name:** `string`

the role name as written, or the default role's name.

**marked:** `string`

the content with every escaping backslash as NUL; breaks as “\n”.

**text:** `string`

the content with escapes processed; line breaks become spaces.

**span:** `gd.Source_Span`

the whole interpreted text, role included.

**content:** `gd.Source_Span`

the content between the backquotes.

**inline.Role\_Hook** *:: struct*

Role\_Hook lets a dialect handle interpreted text before the document's and the standard roles are consulted. The procedure returns false for a role it does not know, and the standard lookup continues.

**procedure:** `proc(user: rawptr, call: ^Role_Call) -> bool`

**user:** `rawptr`

**inline.Smart\_Quotes** *:: enum u8*

Smart\_Quotes selects typographic quotation marks, apostrophes, dashes, and ellipses.

**Off**

**On**

the language's primary quotation marks outside, secondary inside.

**Alt**

the language's alternative marks, where it has them.

```
inline.Substitution_Hook :: struct
```

Substitution\_Hook supplies substitutions a dialect defines by default, such as Sphinx’s `|version|`, `|release|`, and `|today|`. Resolution offers it every reference whose name no definition in the document has exactly (the case-insensitive match comes after it); a procedure returning true replaces the reference with that text.

```
procedure: proc(user: rawptr, name: string) -> (text:
string, ok: bool)
```

```
user: rawptr
```

### Procedures

```
inline.apply_host_role :: proc(s: ^Inline_State, name:
string, lines: []Line, q, j: int) -> gd.Status
```

`apply_host_role` runs the role called `name` on interpreted text written in another markup, such as MyST’s `{name}`content``, as the dialect’s hook and the standard roles would for `reStructuredText`. `lines` hold the whole construct; the content is the joined text’s range `[q, j]` (lines joined by “\n”) and is taken literally: a backslash is an ordinary character. An unknown role is reported, and the construct stays as text.

```
inline.footnote_symbol :: proc(n: int, out: ^gd.Buffer)
```

`footnote_symbol` writes the label of symbol footnote `n` (1-based) into `out`; nothing for `n < 1`. A label that does not fit sets `out`’s overflow, as every `Buffer` write does.

```
inline.parse_inline :: proc(s: ^Inline_State, lines: []Line)
-> gd.Status
```

`parse_inline` emits inline nodes for the lines as children of the builder’s currently open node. Markup may span lines; a line break outside inline literals and roles becomes a `Soft_Break` node, as in `CommonMark`. Markup that cannot be completed is kept as text with a warning; unknown or misused roles are errors. It borrows `s` and `lines` for the call; its working memory is scratch it releases. Problems go to the reports; the result is the builder’s status, which a full table or spent work budget sets.

```
inline.python_space :: proc(r: rune) -> bool
```

`python_space` reports whether `r` is white space as Python’s `str.isspace` and the `\s` of Python’s regular expressions define it, for readers that follow Python patterns.

```
inline.resolve :: proc(ctx: ^gd.Read_Context, options :=
Options{docinfo = true}) -> gd.Status
```

resolve links references to targets, numbers footnotes, checks substitutions, and promotes the document title when ctx.options.promote\_title is set. It edits the builder's nodes in place, with scratch it releases; problems go to ctx.reports; the result is the builder's status.

```
inline.unescape :: proc(out: ^gd.Buffer, marked: string)
```

unescape writes marked text (escapes as NUL, see Role\_Call.marked) with escapes processed: an escaped space or line break disappears, any other escaped character stays, and an unescaped line break becomes a space.

## readers/sphinx

Package sphinx reads reStructuredText with the Sphinx layer: Sphinx's directives, roles, and domains (GDS "Guidedog: everything Sphinx does, the Odin way"), implemented independently on the base reader's extension interface. It produces the shapes of lib/core/AST.md, "Sphinx layer"; cross-document work (toctrees, cross-references, numbering) is left to the project host. The same handlers serve Markdown (MyST) read by the commonmark reader: see markdown.odin.

### Types

```
sphinx.Autodoc_Message :: struct
```

Autodoc\_Message is a problem the host found, reported at the directive.

```
severity: gd.Severity
```

```
code: string
```

```
title: string
```

```
message: string
```

```
hint: string
```

```
sphinx.Autodoc_Option :: struct
```

```
name: string
```

```
value: string
```

"" for a flag.

```
sphinx.Autodoc_Provider :: struct
```

Autodoc\_Provider belongs to the caller’s trust boundary and may allocate.

```
procedure: proc(user: rawptr, request: ^Autodoc_Request) -
> Autodoc_Result
```

```
user: rawptr
```

```
sphinx.Autodoc_Request :: struct
```

Autodoc\_Request is one autodoc directive and the context it needs: the current py:module and py:class (ref\_context) and the document.

```
directive: string
```

“autoclass”

```
argument: string
```

```
options: []Autodoc_Option
```

```
content: string
```

the directive’s content, lines joined with “\n”.

```
module: string
```

```
class: string
```

```
docname: string
```

```
sphinx.Autodoc_Result :: struct
```

Autodoc\_Result borrows host memory valid for the whole conversion. name names the generated text as a source (for diagnostics), such as “<autodoc flask.Flask>”.

```
name: string
```

```
text: string
```

```
messages: []Autodoc_Message
```

```
sphinx.Config :: struct
```

Config holds the `conf.toml` settings that reading needs, with Sphinx's names. It reaches the reader through `gd.Read_Options.extension`. Start from `default_config`: several defaults are true. Settings that the base reader owns (smart quotes, PEP and RFC base URLs, trimmed footnote reference space, language) live in settings.

```
using settings: rst.Settings
```

```
docname: string
```

the document's name, e.g. "usage/intro".

```
default_role: string
```

" " means Docutils' title-reference.

```
primary_domain: string
```

"py"; " " means none.

```
highlight_language: string
```

"default".

```
extlinks: []Extlink
```

```
rst_prolog: string
```

parsed before every document.

```
rst_epilog: string
```

parsed after every document.

```
numfig: bool
```

```
math_number_all: bool
```

```
todo_include_todos: bool
```

```
show_authors: bool
```



**add\_function\_parentheses:** bool

true.

**add\_module\_names:** bool

true.

**toc\_object\_entries:** bool

true: objects have table of contents entries.

**toc\_object\_entries\_show\_parents:** Toc\_Parents

**manpages\_url:** string

“{page}”, “{section}”, “{path}” substituted.

**option\_emphasise\_placeholders:** bool

**trim\_doctest\_flags:** bool

true: doctest flags and <BLANKLINE> are hidden.

**version:** string

The default substitutions |version|, |release|, and |today|, for documents that do not define them: the short version, the full release, and today’s date already formatted (Sphinx’s today, or today\_fmt applied).

**release:** string

**today:** string

**intersphinx:** bool

sphinx.ext.intersphinx: the :external: roles are read when it is on, and an external role naming intersphinx\_resolve\_self is an ordinary reference.

**intersphinx\_resolve\_self:** string

**labels:** ^gd.Labels

The group names of info fields (“Parameters”, “Returns”) in the document’s language, as a host translates its labels; nil for English.

**object\_types:** `[]Object_Type`

Directives and roles the project declares (see `object_types.odin`), and other names for directives.

**directive\_aliases:** `[]Directive_Alias`

**autodoc:** `Autodoc_Provider`

`sphinx.ext.autodoc`: the host that generates what autodoc directives stand for; `nil` when the extension is off. See `autodoc.odin`.

**odin\_autodoc:** `Autodoc_Provider`

The `odin` domain's generating directives (`odin:autopackage`, ...): the host that reads the `Odin` source; `nil` leaves them unknown.

`sphinx.Directive_Alias :: struct`

**name:** `string`

"django-admin-option" names "option"; a target may name

**target:** `string`

"django-admin-option" names "option"; a target may name

`sphinx.Extlink :: struct`

`Extlink` is one `sphinx.ext.extlinks` entry: `:name:part` links to `url` with `%s` replaced by `part`; `caption`, when set, is the link text with `%s` replaced likewise.

**name:** `string`

**url:** `string`

**caption:** `string`

`sphinx.Markdown :: struct`

`Markdown` serves the `commonmark` reader with this layer's directives and roles, the same handlers that serve `reStructuredText`: the reader parses `MyST Markdown` and hands each directive and role here (`commonmark.Extension`); they run on an embedded `reStructuredText` parser whose host is the `Markdown` reader (`rst.Host`), so their nested content is parsed as `Markdown` again. `eval-rst` parses its content as `reStructuredText`. The document is

resolved as reStructuredText is (references, footnotes, targets, bibliographic fields, smart quotes).

Use: `markdown_settings(&m, config, myst)` gives the reader settings to pass as `gd.Read_Options.extension` to the “commonmark” reader; `m` must outlive the read.

<b>config:</b> Config
-----------------------

<b>settings:</b> cm.Settings
------------------------------

<b>ext:</b> cm.Extension
--------------------------

<b>state:</b> State
---------------------

<b>parser:</b> rst.Parser
---------------------------

<b>host:</b> rst.Host
-----------------------

<b>reader:</b> ^cm.Host
-------------------------

sphinx. <b>Object_Type</b> :: struct
--------------------------------------

<b>directive:</b> string
--------------------------

“setting”

<b>role:</b> string
---------------------

“setting”; “” for none.

<b>index:</b> string
----------------------

Sphinx’s `indextemplate`, “pair: %s; setting”; “” for none.

<b>target:</b> bool
---------------------

`add_crossref_type`: a target and an index entry, no description.

<b>first_word:</b> bool
-------------------------

the object’s name is the signature up to its first space.

<b>display:</b> string
------------------------

how the signature is shown, “%s” for it; “” shows it as written.

```
program: bool
```

the name becomes the program options that follow belong to.

```
sphinx.Toc_Parents :: enum u8
```

Toc\_Parents is toc\_object\_entries\_show\_parents: how much of an object's hierarchy its table of contents entry shows.

```
Domain
```

as the domain names it: Python's Class.method, C++'s enclosing objects.

```
Hide
```

the object's own name.

```
All
```

every part: module.Class.method, ns::Class::f.

### Procedures

```
sphinx.default_config :: proc() -> Config
```

default\_config returns Sphinx's defaults, with the base reader's default\_settings and smart quotes on. The value owns nothing; fields a caller sets must outlive the read.

```
sphinx.markdown_settings :: proc(m: ^Markdown, config: Config, myst: cm.Settings) -> ^cm.Settings
```

markdown\_settings prepares m for one read of a Markdown document with the Sphinx configuration config and the MyST settings myst, and returns the reader settings. It copies both into m; the result points into m, so m must outlive the read and serves one read at a time. It cannot fail.

```
sphinx.read :: proc(ctx: ^gd.Read_Context) -> gd.Status
```

read reads reStructuredText with the Sphinx layer. ctx.options.extension is a ^Config (gd.read\_extension), or empty for default\_config; another type is refused with Invalid\_Input. The read copies the Config and keeps nothing of it afterwards; its tables live in scratch, released before it returns. Otherwise it fails as rst.read.

```
sphinx.read_inline :: proc(ctx: ^gd.Read_Context, span: gd.Source_Span) -> gd.Status
```

read\_inline parses a span of ctx.source as reStructuredText inline content with Sphinx's roles, for a translated message (gd.Read\_Inline\_Proc). rst\_prolog

and `rst_epilog` do not apply to a message. Settings, ownership, and failures are read's.

```
sphinx.reader :: proc() -> gd.Reader
```

`reader` returns the “sphinx” reader descriptor, for a `gd.Registry`. It is a value with no state: registering it twice or copying it is harmless.

### Constants

```
sphinx.CONFORMANCE :: gd.Conformance.Partial
```

```
sphinx.MAX_OBJECT_TYPES :: 64
```

```
sphinx.VERSION :: "0.1.0"
```

## render/html

Package `html` renders a Guidedoc document as HTML without calling an allocator. The default output is semantic HTML with anchors, notes, and tables; the reference profile reproduces the CommonMark reference renderer byte for byte for conformance tests. Raw content and unsafe URLs are governed by the conversion policy.

### Procedures

```
html.render :: proc(ctx: ^gd.Render_Context) -> gd.Status
```

`render` writes `ctx.view` as HTML through `ctx.out`: a whole page when `ctx.options.standalone`, else a fragment, and the CommonMark reference renderer's bytes when `ctx.options.reference_profile`. It borrows the context for the call and keeps nothing; its scratch is released before it returns. Problems go to `ctx.reports`; the result is `.Ok`, the status of the first error (a raw block or URL the policy refuses, an unresolved reference, nesting past the limits), or `.Capacity` when scratch runs out.

```
html.renderer :: proc() -> gd.Renderer
```

`renderer` returns the “html” renderer descriptor, for a `gd.Registry`. It is a value with no state: registering it twice or copying it is harmless.

### Constants

```
html.CONFORMANCE :: gd.Conformance.Partial
```

```
html.VERSION :: "0.1.0"
```

## render/text

Package `text_render` renders a Guidedoc document as plain text without calling an allocator, following the conventions of Sphinx’s text builder: paragraphs wrapped at 70 columns, titles underlined with `= ~` and the other section characters, `“* “` and `“1. “` list markers with hanging indents, definitions and literal blocks indented by three spaces, grid tables drawn with `+--+` borders, and `“Note:”` admonitions.

Inline content is laid out twice, once to measure and once into scratch, and then wrapped; table cells are laid out into scratch at trial widths before the table is drawn. Output goes only through the emitter, so the engine’s measure and write runs produce identical bytes.

### Procedures

```
text.render :: proc(ctx: ^gd.Render_Context) -> gd.Status
```

`render` writes `ctx.view` as plain text through `ctx.out`. It borrows the context for the call and keeps nothing; its scratch is released before it returns. Problems go to `ctx.reports`; the result is `.Ok`, the status of the first error (raw content the policy refuses, an unresolved reference, nesting past the limits), or `.Capacity` when scratch cannot hold the indentation frames or a laid-out paragraph.

```
text.renderer :: proc() -> gd.Renderer
```

`renderer` returns the “text” renderer descriptor, for a `gd.Registry`. It is a value with no state: registering it twice or copying it is harmless.

### Constants

```
text.CONFORMANCE :: gd.Conformance.Partial
```

```
text.VERSION :: "0.1.0"
```

## render/typst

Package `typst_render` encodes a validated Guidedoc document as one Typst source file.

The output is a fixed, trusted preamble of `gd-` helper definitions followed by the body. Source text never becomes executable Typst: markup text escapes every ASCII punctuation character, and code, URLs, labels, and paths are string literals. Working memory comes only from the render context’s scratch; output goes only through the emitter, so the Measure and Write runs produce identical bytes.

## Procedures

```
typst.emit_preamble :: proc(e: ^gd.Emitter)
```

emit\_preamble writes the helper definitions every body relies on. A host composing a book from body-only chapters writes it once, before the chapters. The text is a constant: it cannot fail, and the emitter counts what does not fit.

```
typst.render :: proc(ctx: ^gd.Render_Context) -> gd.Status
```

render writes ctx.view as Typst source through ctx.out: the preamble, page settings, and body when ctx.options.standalone, else the body alone (see emit\_preamble). It borrows the context for the call and keeps nothing but scratch taken from ctx, which the caller's scratch mark releases. Problems go to ctx.reports; the result is .Ok, the status of the first error (raw content the policy refuses, nesting past the limits), or .Capacity when scratch cannot hold the per-node marks.

```
typst.renderer :: proc() -> gd.Renderer
```

renderer returns the “typst” renderer descriptor, for a gd.Registry. It is a value with no state: registering it twice or copying it is harmless.

## Constants

```
typst.CONFORMANCE :: gd.Conformance.Partial
```

```
typst.VERSION :: "0.1.1"
```

## treesitter

Package treesitter is a thin binding of the tree-sitter C library and its Python grammar: parsers, trees, nodes, and cursors, and nothing more. It imports only Odin's own packages, so any Odin program can use it; Guidedog's autodoc reads Python source through it.

## Types

```
treesitter.Cursor :: struct
```

Cursor is TSTreeCursor; cursor\_delete frees it.

```
tree: rawptr
```

```
id: rawptr
```

```
ctx: [3]u32
```

```
tree sitter.Field_Id :: u16
```

```
tree sitter.Iterator :: struct
```

Iterator visits a node's children in order, named or all, for for child in ....

```
node: Node
```

```
index: int
```

```
count: int
```

```
named: bool
```

```
tree sitter.Language :: struct
```

```
tree sitter.Node :: struct
```

Node is TSNODE, a value that is valid while its tree lives.

```
ctx: [4]u32
```

```
id: rawptr
```

```
tree: ^Tree
```

```
tree sitter.Parser :: struct
```

Opaque handles.

```
tree sitter.Point :: struct
```

```
row: u32
```

```
column: u32
```

in bytes

```
tree sitter.Symbol :: u16
```

```
tree sitter.Tree :: struct
```



## Procedures

```
tree-sitter.abi_version :: proc(l: ^Language) -> int
```

abi\_version is the grammar's ABI, which the runtime must support.

```
tree-sitter.child :: proc(n: Node, i: int) -> Node
```

child is the node's child i of child\_count, or a null node.

```
tree-sitter.child_count :: proc(n: Node) -> int
```

child\_count counts the node's children, named and anonymous.

```
tree-sitter.children :: proc(n: Node, named := true) ->
Iterator
```

children iterates n's children, named ones only unless named is false, with next\_child.

```
tree-sitter.cursor_delete :: proc(c: ^Cursor)
```

cursor\_delete frees what cursor\_new allocated.

```
tree-sitter.cursor_field_name :: proc(c: ^Cursor) -> string
```

cursor\_field\_name names the field the cursor's node fills in its parent, or "".

```
tree-sitter.cursor_first_child :: proc(c: ^Cursor) -> bool
```

cursor\_first\_child moves to the first child; false, and no move, when there is none.

```
tree-sitter.cursor_new :: proc(n: Node) -> Cursor
```

cursor\_new starts a cursor at n, for walks that allocate nothing per step. tree-sitter allocates the cursor's stack; the caller frees it with cursor\_delete.

```
tree-sitter.cursor_next_sibling :: proc(c: ^Cursor) -> bool
```

cursor\_next\_sibling moves to the next sibling; false, and no move, when there is none.

```
tree-sitter.cursor_node :: proc(c: ^Cursor) -> Node
```

cursor\_node is the node the cursor is at.

```
treesitter.cursor_parent :: proc(c: ^Cursor) -> bool
```

`cursor_parent` moves to the parent; false, and no move, at the node the cursor started at.

```
treesitter.deeper_than :: proc(n: Node, limit: int) -> bool
```

`deeper_than` reports whether the tree below `n` nests more than `limit` levels (`n`'s children are level 1). Parsing needs no recursion however deep the source nests, but recursive walks over the tree do: a program checks first, and refuses trees deeper than its walks can hold. It walks with a cursor, so it needs no recursion itself.

```
treesitter.end_byte :: proc(n: Node) -> int
```

`end_byte` is the byte offset just past the node in the parsed text.

```
treesitter.end_point :: proc(n: Node) -> Point
```

`end_point` is the 0-based row and byte column where the node ends; see `start_point`.

```
treesitter.equal :: proc(a, b: Node) -> bool
```

`equal` reports whether `a` and `b` are the same node of the same tree.

```
treesitter.field :: proc(n: Node, name: string) -> Node
```

`field` is the child the grammar names, such as “name” or “body”; a null node when absent.

```
treesitter.field_name_for_child :: proc(n: Node, i: int) -> string
```

`field_name_for_child` names the field child `i` fills, or “”.

```
treesitter.has_error :: proc(n: Node) -> bool
```

`has_error` reports whether the node or any node below it is an error or missing node.

```
treesitter.is_error :: proc(n: Node) -> bool
```

`is_error` reports a node the parser could not fit into the grammar.

```
treesitter.is_extra :: proc(n: Node) -> bool
```

`is_extra`: comments and other tokens that may appear anywhere.

```
tree_sitter.is_missing :: proc(n: Node) -> bool
```

`is_missing` reports a node the parser inserted to recover from a syntax error.

```
tree_sitter.is_named :: proc(n: Node) -> bool
```

`is_named` reports a node the grammar names, rather than an anonymous token such as “(“.

```
tree_sitter.is_null :: proc(n: Node) -> bool
```

`is_null` reports a null node: an absent field, child, sibling, or parent.

```
tree_sitter.kind :: proc(n: Node) -> string
```

`kind` is the node’s type name, such as “function\_definition”, or the token itself for anonymous nodes such as “(“ or “def”.

```
tree_sitter.named_child :: proc(n: Node, i: int) -> Node
```

`named_child` is the node’s named child `i` of `named_child_count`, or a null node.

```
tree_sitter.named_child_count :: proc(n: Node) -> int
```

`named_child_count` counts the node’s named children.

```
tree_sitter.next_child :: proc(it: ^Iterator) -> (n: Node, i: int, ok: bool)
```

`next_child` returns the next child and its index, for  
for child, i in `next_child(&it)`.

```
tree_sitter.next_named_sibling :: proc(n: Node) -> Node
```

`next_named_sibling` is the next named node under `n`’s parent, or a null node.

```
tree_sitter.next_sibling :: proc(n: Node) -> Node
```

`next_sibling` is the node after `n` under its parent, or a null node.

```
tree_sitter.parent :: proc(n: Node) -> Node
```

`parent` is the node’s parent, or a null node for the root.

```
tree_sitter.parse :: proc(p: ^Parser, text: string) -> ^Tree
```

`parse` parses UTF-8 text into a tree, which tree-sitter allocates and the caller frees with `tree_delete`. The tree refers to text by byte offsets only, so text need not outlive it, but text slices whatever text it is given. Nil when parsing was

impossible (text over 4 GiB, or no language set); syntax errors give a tree with error nodes instead (`has_error`).

```
tree sitter.parser_delete :: proc(p: ^Parser)
```

`parser_delete` frees a parser from `parser_new`; nil is ignored. Trees it parsed stay valid.

```
tree sitter.parser_new :: proc(language: ^Language) -> ^Parser
```

`parser_new` returns a parser for `language`, which tree-sitter allocates and the caller frees with `parser_delete`, or nil when this runtime cannot read the grammar (a shared runtime older than the grammar's ABI).

```
tree sitter.prev_named_sibling :: proc(n: Node) -> Node
```

`prev_named_sibling` is the previous named node under `n`'s parent, or a null node.

```
tree sitter.prev_sibling :: proc(n: Node) -> Node
```

`prev_sibling` is the node before `n` under its parent, or a null node.

```
tree sitter.python :: proc() -> ^Language
```

`python` returns the Python grammar, static data that is never freed.

```
tree sitter.root :: proc(t: ^Tree) -> Node
```

`root` is the tree's root node, valid while the tree lives.

```
tree sitter.start_byte :: proc(n: Node) -> int
```

`start_byte` is the byte offset where the node starts in the parsed text.

```
tree sitter.start_point :: proc(n: Node) -> Point
```

`start_point` and `end_point` are 0-based rows and byte columns.

```
tree sitter.symbol :: proc(n: Node) -> Symbol
```

`symbol` is the grammar's number for the node's type.

```
tree sitter.text :: proc(n: Node, source: string) -> string
```

`text` is the source the node spans, a slice of the parsed text.

```
tree_sitter.tree_delete :: proc(t: ^Tree)
```

`tree_delete` frees a tree from parse, and with it every node of the tree; nil is ignored.

### Constants

```
tree_sitter.NOTICES :: #load("licenses/tree-sitter.txt",
string) + "\n" + #load("licenses/tree-sitter-python.txt",
string)
```

The MIT licence notices of the two libraries, which a program that links them ships.

```
tree_sitter.PYTHON_VERSION :: "0.25.0"
```

```
tree_sitter.RUNTIME_VERSION :: "0.26.13"
```

The versions native/tree-sitter/build.sh pins; a shared build links whatever is installed.

```
tree_sitter.STATIC :: #config(TREESITTER_STATIC, BUILT)
```

### Foreign imports

```
@(extra_linker_flags = #config(TREESITTER_LINK_FLAGS, ""))
foreign import tree_sitter.ts {"../..../native/tree-sitter/out/
lib/tree-sitter-python.lib", "../..../native/tree-sitter/out/
lib/tree-sitter.lib"}
```

```
@(extra_linker_flags = #config(TREESITTER_LINK_FLAGS, ""))
foreign import tree_sitter.ts {"../..../native/tree-sitter/out/
lib/libtree-sitter-python.a", "../..../native/tree-sitter/out/
lib/libtree-sitter.a"}
```

```
@(extra_linker_flags = #config(TREESITTER_LINK_FLAGS, ""))
foreign import tree_sitter.ts {"system:tree-sitter-python",
"system:tree-sitter"}
```

## typst

Package `typst` adapts the Typst toolchain for the host. The development backend runs an installed, pinned `typst` executable with an argument vector, never a shell; the embedded backend links the compiler in (bridge.odin). It is outside Guidedoc's allocation-free core.

Typst is not a sandbox of its own; what it may do is what the caller gives it:

- Files: Typst reads only paths below the root it is given, and refuses a path that leaves it; a link below the root is followed. Fonts come from `font_paths` and, unless `embedded_fonts_only` is set, from the system.
- Packages: an `@preview` import downloads the package, unless `offline` is set (the embedded backend only; the typst program has no such switch).
- Memory and time: Typst's own, in the process (embedded) or in the typst program; no budget of the host bounds them. Only the output the embedded backend copies back is counted, through `reserve`.

So Typst code reaches everything below its root: a caller that compiles Typst it does not trust gives a root holding only what that code may read. Guidedog's builds do (project/typst\_root.odin), and an `-untrusted` build compiles only Typst it generated.

### Types

`typst.Backend` *:: struct*

Backend is the compiler a compile runs: an installed typst program, or the embedded one. `locate` makes it; its strings are the caller's.

**program:** string

**version:** string

**embedded:** bool

the statically linked compiler; no external program runs.

`typst.Diagnostic` *:: struct*

Diagnostic is one problem Typst reported.

**file:** string

**line:** int

1-based; 0 when the compiler gave no position.

**column:** int

**severity:** string

**message:** string

**stdin:** bool

the position is in the generated source fed on `stdin`.

```
typst.Options :: struct
```

Options mirror the typst CLI. Zero values mean: download missing packages into the system cache, and use system fonts before the embedded ones.

```
offline: bool
```

use only packages already on disk.

```
package_cache: string
```

package cache directory; “” is the system cache.

```
embedded_fonts_only: bool
```

ignore system fonts, for reproducible output.

```
font_paths: []string
```

extra font directories, searched first.

```
reserve: Reserve
```

When set, the embedded compiler’s output is counted before it is copied into host memory; a refusal fails the compile. The typst program writes its output itself.

```
typst.Outcome :: struct
```

Outcome is what a compile did. Its diagnostics and text are allocated in context allocator and owned by the caller; launch\_error is set when Typst could not run.

```
ok: bool
```

```
exit_code: int
```

```
diagnostics: []Diagnostic
```

```
detail: string
```

unparsed compiler output, bounded.

```
launch_error: string
```

```
typst.Reserve :: struct
```

Reserve asks the caller whether bytes may be allocated, before the embedded compiler's output is copied into host memory; false refuses the compile.

```
procedure: proc(user: rawptr, bytes: int) -> bool
```

```
user: rawptr
```

```
typst.Target :: enum u8
```

Target is the artifact a compile writes.

```
Pdf
```

```
Html
```

### Procedures

```
typst.compile_file :: proc(b: Backend, source, root, output: string, target: Target, options := Options{}) -> Outcome
```

compile\_file compiles a native Typst source file into output; paths resolve against root, and Typst reads nothing outside it. The Outcome is allocated in context.allocator and owned by the caller. Write to a staged path and rename it on success when a failed compile must not disturb an earlier artifact.

```
typst.compile_text :: proc(b: Backend, text, root, output: string, target: Target, options := Options{}) -> Outcome
```

compile\_text compiles generated Typst fed on stdin, so no file is written beside the user's sources. Relative paths in the text resolve against root. The Outcome is allocated in context.allocator and owned by the caller.

```
typst.find_program :: proc(name: string) -> (string, bool)
```

find\_program looks a program up on PATH, handling Windows' ";" and ".exe". The path is allocated in context.allocator and owned by the caller; false when none is found.

```
typst.locate :: proc(program: string) -> (Backend, bool)
```

locate returns the embedded compiler when this build links one and the caller did not name another program; otherwise it finds the program and asks for its version. The backend's strings are allocated in context.allocator and owned by the caller; false when no program is found. A program that cannot tell its version has version "".



```
typst.offset_of :: proc(text: string, line, column: int) -> int
```

offset\_of converts a 1-based line and column (in characters) into a byte offset of text, clamped to its end; it allocates nothing.

```
typst.pinned :: proc(b: Backend) -> bool
```

pinned reports whether the backend is the Typst release Guidedog is tested with.

### Constants

```
typst.EMBEDDED :: #config(GUIDEDOG_TYPST_BRIDGE, false)
```

EMBEDDED selects the statically linked Typst compiler. Build the bridge first:

```
cargo build --release --offline --manifest-path native/typst_bridge/
Cargo.toml
odin build cmd/guidedog -define:GUIDEDOG_TYPST_BRIDGE=true -out:build/
guidedog
```

Without it, Guidedog uses the development backend: an installed typst program.

```
typst.PINNED_VERSION :: "0.15.1"
```

PINNED\_VERSION is the Typst release Guidedog is built and tested with.

### Foreign imports

```
foreign import typst.bridge {"../../native/typst_bridge/
target/release/guidedog_typst_bridge.lib",
"system:ws2_32.lib", "system:userenv.lib",
"system:ntdll.lib", "system:bcrypt.lib",
"system:advapi32.lib", "system:kernel32.lib"}
```

可用平台: Windows.

```
foreign import typst.bridge {"../../native/typst_bridge/
target/release/
libguidedog_typst_bridge.a", "system:Security.framework",
"system:CoreFoundation.framework",
"system:System", "system:iconv"}
```

可用平台: Darwin.

```
foreign import typst.bridge {"../../native/typst_bridge/
target/release/libguidedog_typst_bridge.a", "system:gcc_s",
"system:util", "system:rt", "system:pthread", "system:m",
"system:dl", "system:c"}
```

可用平台: Linux, FreeBSD, OpenBSD, NetBSD, WASI, JS, Orca, Freestanding.



---

# 索引

---

## A

- abi\_version (procedure in treesitter), 333
- absolute (procedure in project), 269
- add (procedure in i18n), 220
  - (procedure in inventory), 225
- add\_attribute (procedure in core), 115
- add\_attribute\_string (procedure in core), 115
- add\_catalog (procedure in gettext), 152
- add\_common (procedure in readers/rst), 310
- add\_filter (procedure in jinja), 234
- add\_flags (procedure in core), 116
- add\_function (procedure in jinja), 234
- add\_global (procedure in jinja), 234
- add\_leaf (procedure in core), 116
- add\_message (procedure in gettext), 152
- add\_source (procedure in core), 116
- add\_template (procedure in jinja), 234
- add\_test (procedure in jinja), 235
- add\_text (procedure in core), 116
- Admonition\_Kind (enum in core), 80
- Admonition\_Kind.Attention (enumerator in core), 80
- Admonition\_Kind.Caution (enumerator in core), 80
- Admonition\_Kind.Danger (enumerator in core), 80
- Admonition\_Kind.Error (enumerator in core), 80
- Admonition\_Kind.Generic (enumerator in core), 80
- Admonition\_Kind.Hint (enumerator in core), 80
- Admonition\_Kind.Important (enumerator in core), 80
- Admonition\_Kind.Note (enumerator in core), 80
- Admonition\_Kind.Seealso (enumerator in core), 80
- Admonition\_Kind.Tip (enumerator in core), 80
- Admonition\_Kind.TODO (enumerator in core), 80
- Admonition\_Kind.Warning (enumerator in core), 80
- admonition\_label (procedure in core), 116
- admonition\_name (procedure in core), 116
- Align (enum in core), 80
- Align.Center (enumerator in core), 80
- Align.Left (enumerator in core), 80
- Align.None (enumerator in core), 80
- Align.Right (enumerator in core), 80
- All (struct in pyscan), 272
- All.defined (field in pyscan), 272
- All.known (field in pyscan), 272
- All.names (field in pyscan), 272
- Analysis (struct in pyscan), 273
- Analysis.annotation\_order (field in pyscan), 273
- Analysis.annotations (field in pyscan), 273
- Analysis.attr\_docs (field in pyscan), 273
- Analysis.attr\_order (field in pyscan), 273
- Analysis.counter (field in pyscan), 273
- Analysis.finals (field in pyscan), 273
- Analysis.overloads (field in pyscan), 273
- Analysis.tagorder (field in pyscan), 273
- Anchor\_Extra (struct in core), 81
- Anchor\_Extra.item (field in core), 81
- Anchor\_Extra.note (field in core), 81
- Anchor\_Extra.number (field in core), 81
- Anchor\_Extra.owner (field in core), 81
- Anchor\_Extra.span (field in core), 81
- anchor\_extras (procedure in core), 116
- anchor\_ids (procedure in core), 116
- Anchor\_Ids (struct in core), 81
- Anchor\_Ids.span (field in core), 81
- anchor\_of (procedure in core), 116
- anchors (struct in core), 81
- anchors.bytes (field in core), 82
- anchors.count (field in core), 82
- anchors.counter (field in core), 82
- anchors.extra\_count (field in core), 82
- anchors.extras (field in core), 82
- anchors.index (field in core), 82
- anchors.labels (field in core), 82
- anchors.names (field in core), 82
- anchors.next (field in core), 82
- anchors.own (field in core), 82
- anchors.prefix (field in core), 82
- anchors.prefix\_length (field in core), 82
- anchors.spans (field in core), 82
- append\_value (procedure in jinja), 235
- apply (procedure in gds), 140
- apply\_host\_role (procedure in readers/rst/inline), 321
- Arch (type in odindoc), 241
- Archs (type in odindoc), 241

- Arena (struct in host), 187
- Arena.backing (field in host), 187
- Arena.block (field in host), 187
- Arena.blocks (field in host), 187
- Arena.bytes (field in host), 187
- Arena.last (field in host), 187
- arena\_allocator (procedure in host), 200
- ARENA\_BLOCK\_BYTES (constant in host), 213
- arena\_destroy (procedure in host), 200
- arena\_init (procedure in host), 200
- arg (procedure in jinja), 235
- arg\_int (procedure in core), 117
  - (procedure in jinja), 235
- Arg\_Kind (enum in core), 83
- Arg\_Kind.Int (enumerator in core), 83
- Arg\_Kind.None (enumerator in core), 83
- Arg\_Kind.Span (enumerator in core), 83
- Arg\_Kind.Static (enumerator in core), 83
- Arg\_Rule (enum in readers/rst), 299
- Arg\_Rule.None (enumerator in readers/rst), 299
- Arg\_Rule.Optional (enumerator in readers/rst), 299
- Arg\_Rule.Required (enumerator in readers/rst), 299
- arg\_span (procedure in core), 117
- arg\_static (procedure in core), 117
- Artifact\_Request (struct in host), 187
- Artifact\_Request.allocator (field in host), 188
- Artifact\_Request.budget (field in host), 188
- Artifact\_Request.emit\_typst (field in host), 188
- Artifact\_Request.force (field in host), 188
- Artifact\_Request.from (field in host), 188
- Artifact\_Request.input (field in host), 187
- Artifact\_Request.limits (field in host), 188
- Artifact\_Request.output (field in host), 188
- Artifact\_Request.policy (field in host), 188
- Artifact\_Request.read (field in host), 188
- Artifact\_Request.render (field in host), 188
- Artifact\_Request.roots (field in host), 188
- Artifact\_Request.to (field in host), 188
- Artifact\_Request.typst\_options (field in host), 188
- Artifact\_Request.typst\_program (field in host), 188
- Artifact\_Result (struct in host), 188
- Artifact\_Result.exit (field in host), 189
- Artifact\_Result.problem (field in host), 189
- Artifact\_Result.reports (field in host), 189
- Artifact\_Result.stdout (field in host), 189
- Artifact\_Result.written (field in host), 189
- Attr\_Key (struct in pyscan), 273
- Attr\_Key.name (field in pyscan), 273
- Attr\_Key.scope (field in pyscan), 273
- attribute (procedure in core), 117
- Attribute (struct in core), 83
  - (struct in graphdog), 159
- Attribute.key (field in core), 83
- Attribute.kind (field in graphdog), 159
- Attribute.name (field in core), 83
  - (field in graphdog), 159
- Attribute.value (field in core), 83
  - (field in graphdog), 159
- Attribute\_Key (enum in core), 83
- Attribute\_Key.Align (enumerator in core), 84
- Attribute\_Key.Alt (enumerator in core), 83
- Attribute\_Key.Class (enumerator in core), 83
- Attribute\_Key.Custom (enumerator in core), 83
- Attribute\_Key.Format (enumerator in core), 84
- Attribute\_Key.Height (enumerator in core), 83
- Attribute\_Key.Id (enumerator in core), 83
- Attribute\_Key.Language (enumerator in core), 83
- Attribute\_Key.Name (enumerator in core), 83
- Attribute\_Key.Scale (enumerator in core), 83
- Attribute\_Key.Source (enumerator in core), 84
- Attribute\_Key.Start (enumerator in core), 84
- Attribute\_Key.Target (enumerator in core), 84
- Attribute\_Key.Title (enumerator in core), 84
- Attribute\_Key.Width (enumerator in core), 83
- Attribute\_Key.Widths (enumerator in core), 84
- Attribute\_Kind (enum in graphdog), 159
- Attribute\_Kind.Edge (enumerator in graphdog), 160
- Attribute\_Kind.Graph (enumerator in graphdog), 159
- Attribute\_Kind.Node (enumerator in graphdog), 159
- Attribute\_Range (struct in core), 84
- Attribute\_Range.count (field in core), 84
- Attribute\_Range.first (field in core), 84
- attributes (procedure in core), 117
- Attributes (struct in core), 84
- Attributes.end (field in core), 84
- Attributes.next (field in core), 84
- autodoc (package), 74
- Autodoc\_Message (struct in readers/sphinx), 322
- Autodoc\_Message.code (field in readers/sphinx), 322
- Autodoc\_Message.hint (field in readers/sphinx), 322
- Autodoc\_Message.message (field in readers/sphinx), 322
- Autodoc\_Message.severity (field in readers/sphinx), 322
- Autodoc\_Message.title (field in readers/sphinx), 322
- Autodoc\_Option (struct in readers/sphinx), 322
- Autodoc\_Option.name (field in readers/sphinx), 322
- Autodoc\_Option.value (field in readers/sphinx), 322

Autodoc\_Provider (struct in readers/  
sphinx), 323  
Autodoc\_Provider.procedure (field in  
readers/sphinx), 323  
Autodoc\_Provider.user (field in readers/  
sphinx), 323  
Autodoc\_Request (struct in readers/  
sphinx), 323  
Autodoc\_Request.argument (field in  
readers/sphinx), 323  
Autodoc\_Request.class (field in readers/  
sphinx), 323  
Autodoc\_Request.content (field in readers/  
sphinx), 323  
Autodoc\_Request.directive (field in  
readers/sphinx), 323  
Autodoc\_Request.docname (field in  
readers/sphinx), 323  
Autodoc\_Request.module (field in readers/  
sphinx), 323  
Autodoc\_Request.options (field in readers/  
sphinx), 323  
Autodoc\_Result (struct in readers/sphinx),  
323  
Autodoc\_Result.messages (field in readers/  
sphinx), 323  
Autodoc\_Result.name (field in readers/  
sphinx), 323  
Autodoc\_Result.text (field in readers/  
sphinx), 323  
autoescaping (procedure in jinja), 235  
available\_memory (procedure in host), 200  
AVAILABLE\_OVERRIDE (constant in  
host), 213

## B

Backend (struct in typst), 338  
Backend.embedded (field in typst), 338  
Backend.program (field in typst), 338  
Backend.version (field in typst), 338  
backend\_problem (procedure in host), 201  
base\_objects (procedure in pyscan), 284  
begin\_node (procedure in core), 117  
bind\_text\_domain (procedure in gettext),  
152  
blank (procedure in readers/rst), 310  
boolean (procedure in jinja), 235  
Bound\_Catalog (struct in gettext), 143  
Bound\_Catalog.catalog (field in gettext),  
143  
Bound\_Catalog.domain (field in gettext),  
143  
bridge (foreign import in typst), 341  
Budget (struct in host), 189  
Budget.active (field in host), 189  
Budget.freed (field in host), 190  
Budget.limit (field in host), 189  
Budget.mutex (field in host), 190  
Budget.peak (field in host), 189  
Budget.policy (field in host), 190  
Budget.refused\_need (field in host), 190  
Budget.refused\_used (field in host), 190  
Budget.stopped (field in host), 190  
Budget.used (field in host), 189

Budget.waiting (field in host), 189  
budget\_advice (procedure in host), 201  
budget\_hint (procedure in host), 201  
budget\_join (procedure in host), 201  
budget\_leave (procedure in host), 201  
budget\_limit (procedure in host), 201  
Budget\_Policy (struct in host), 190  
Budget\_Policy.disk (field in host), 190  
Budget\_Policy.locate (field in host), 190  
Budget\_Policy.locate\_user (field in host),  
190  
Budget\_Policy.on\_refusal (field in host),  
190  
Budget\_Policy.spill (field in host), 190  
Budget\_Policy.user (field in host), 190  
budget\_problem (procedure in host), 201  
budget\_report (procedure in host), 201  
Buffer (struct in core), 84  
Buffer.bytes (field in core), 84  
Buffer.overflow (field in core), 84  
Buffer.used (field in core), 84  
Buffer.wanted (field in core), 84  
buffer\_string (procedure in core), 117  
buffer\_write (procedure in core), 117  
buffer\_write\_byte (procedure in core), 117  
build (procedure in gds), 140  
(procedure in project), 269  
build\_anchors (procedure in core), 117  
BUILD\_BUDGET (constant in project), 272  
build\_index (procedure in gettext), 152  
build\_limits (procedure in project), 270  
Build\_Options (struct in project), 248  
Build\_Options.all (field in project), 248  
Build\_Options.arena\_block (field in  
project), 250  
Build\_Options.budget (field in project), 249  
Build\_Options.fresh (field in project), 248  
Build\_Options.heap (field in project), 249  
Build\_Options.jobs (field in project), 249  
Build\_Options.keep\_going (field in  
project), 249  
Build\_Options.max\_depth (field in  
project), 249  
Build\_Options.max\_nodes (field in  
project), 249  
Build\_Options.memory (field in project),  
249  
Build\_Options.nitpicky (field in project),  
249  
Build\_Options.now (field in project), 250  
Build\_Options.output\_dir (field in project),  
250  
Build\_Options.stack\_bytes (field in  
project), 249  
Build\_Options.tags (field in project), 249  
Build\_Options.untrusted (field in project),  
249  
Build\_Options.warnings\_as\_errors (field in  
project), 249  
Build\_Options.year (field in project), 250  
Build\_Result (struct in project), 250  
Build\_Result.budget (field in project), 251  
Build\_Result.exit (field in project), 250  
Build\_Result.memory (field in project), 251  
Build\_Result.output (field in project), 250  
Build\_Result.owner (field in project), 251  
Build\_Result.peak (field in project), 251

Build\_Result.problem (field in project), 250  
 Build\_Result.published (field in project), 251  
 Build\_Result.reached (field in project), 251  
 Build\_Result.read (field in project), 250  
 Build\_Result.reports (field in project), 250  
 Build\_Result.reused (field in project), 250  
 Build\_Result.unchanged (field in project), 250  
 Build\_Result.unread (field in project), 250  
 Build\_Result.unsettled (field in project), 251  
 Build\_Result.written (field in project), 250  
 Build\_Stage (enum in project), 251  
 Build\_Stage.Preparing (enumerator in project), 251  
 Build\_Stage.Reading (enumerator in project), 251  
 Build\_Stage.Writing (enumerator in project), 251  
 Build\_Stage.Written (enumerator in project), 251  
 Builder (struct in core), 85  
 Builder.cancel (field in core), 85  
 Builder.limits (field in core), 85  
 Builder.need (field in core), 85  
 Builder.open (field in core), 85  
 Builder.reports (field in core), 85  
 Builder.snap (field in core), 85  
 Builder.status (field in core), 85  
 Builder.work (field in core), 85  
 builder\_capacity (procedure in core), 118  
 builder\_fail (procedure in core), 118  
 builder\_finish (procedure in core), 118  
 builder\_init (procedure in core), 118  
 Builder\_Kind (enum in project), 252  
 Builder\_Kind.Dirhtml (enumerator in project), 252  
 Builder\_Kind.Dummy (enumerator in project), 252  
 Builder\_Kind.Gettext (enumerator in project), 252  
 Builder\_Kind.Html (enumerator in project), 252  
 Builder\_Kind.Pdf (enumerator in project), 252  
 Builder\_Kind.Singlehtml (enumerator in project), 252  
 Builder\_Kind.Text (enumerator in project), 252  
 builder\_named (procedure in project), 270  
 BUILDER\_NAMES (variable in project), 272  
 BUILDER\_OPEN (constant in core), 133  
 bullet\_width (procedure in readers/rst), 310

## C

Call (struct in jinja), 226  
 Call.args (field in jinja), 226  
 Call.kwargs (field in jinja), 226  
 Call.name (field in jinja), 226  
 Call.self (field in jinja), 226  
 Call.state (field in jinja), 226  
 Call.user (field in jinja), 226

call\_filter (procedure in jinja), 235  
 call\_test (procedure in jinja), 235  
 Callable (struct in jinja), 226  
 Callable.macro (field in jinja), 227  
 Callable.name (field in jinja), 226  
 Callable.procedure (field in jinja), 226  
 Callable.self (field in jinja), 226  
 Callable.user (field in jinja), 226  
 cancel\_probe (procedure in host), 202  
 Cancel\_Probe (struct in core), 85  
 Cancel\_Probe.procedure (field in core), 85  
 Cancel\_Probe.user (field in core), 85  
 canonical\_roots (procedure in host), 202  
 Capacity\_Need (struct in core), 85  
 Capacity\_Need.exact (field in core), 85  
 Capacity\_Need.minimum (field in core), 85  
 Capacity\_Need.region (field in core), 85  
 Catalog (struct in gds), 137  
 (struct in gettext), 143  
 Catalog allocator (field in gettext), 144  
 Catalog.allow\_fuzzy (field in gettext), 144  
 Catalog.arena (field in gettext), 144  
 Catalog.has\_header (field in gettext), 143  
 Catalog.header (field in gettext), 143  
 Catalog.index (field in gettext), 144  
 Catalog.messages (field in gettext), 143  
 Catalog.name (field in gettext), 143  
 Catalog.owner (field in gettext), 144  
 Catalog.plural (field in gettext), 143  
 Catalog.records (field in gds), 137  
 Catalog.reserved (field in gds), 137  
 Catalog\_Arena (struct in gettext), 144  
 Catalog\_Arena.refused (field in gettext), 144  
 Catalog\_Arena.using (field in gettext), 144  
 catalog\_bytes (procedure in gettext), 152  
 catalog\_refused (procedure in gettext), 152  
 catch\_interrupts (procedure in host), 202  
 cell\_info (procedure in core), 118  
 Cell\_Info (struct in core), 86  
 Cell\_Info.align (field in core), 86  
 Cell\_Info.col\_span (field in core), 86  
 Cell\_Info.row\_span (field in core), 86  
 Change (struct in gds), 137  
 Change.after (field in gds), 138  
 Change.before (field in gds), 137  
 Change.existed (field in gds), 138  
 Change.path (field in gds), 137  
 Change.present (field in gds), 138  
 charge (procedure in host), 202  
 charset (procedure in gettext), 153  
 check\_args (procedure in jinja), 235  
 check\_code (procedure in highlight), 182  
 check\_index (procedure in gds), 141  
 check\_render (procedure in core), 118  
 check\_sources (procedure in gds), 141  
 check\_style (procedure in highlight), 182  
 child (procedure in treesitter), 333  
 child\_count (procedure in core), 118  
 (procedure in treesitter), 333  
 children (procedure in core), 118  
 (procedure in treesitter), 333  
 Children (struct in core), 86  
 Children.end (field in core), 86  
 Children.next (field in core), 86  
 Class (struct in pyscan), 273  
 Class.bases (field in pyscan), 274  
 Class.decorators (field in pyscan), 274  
 Class.doc (field in pyscan), 274  
 Class.keywords (field in pyscan), 274



- Class.module (field in pyscan), 273
- Class.mro\_complete (field in pyscan), 274
- Class.mro\_done (field in pyscan), 274
- Class.mro\_order (field in pyscan), 274
- Class.name (field in pyscan), 273
- Class.node (field in pyscan), 274
- Class.outer (field in pyscan), 274
- Class.qualname (field in pyscan), 274
- Class.scope (field in pyscan), 274
- class\_attr (procedure in pyscan), 284
- class\_directive (procedure in readers/rst), 311
- clean (procedure in project), 270
- clear\_cache (procedure in jinja), 236
- cli (package), 78
- clone\_message (procedure in gettext), 153
- close (procedure in graphdog), 163
  - (procedure in readers/rst), 311
- code\_caption (procedure in core), 119
- code\_text (procedure in readers/rst), 311
- Color (struct in highlight), 167
- Color.hex (field in highlight), 167
- Color.set (field in highlight), 167
- Compact (struct in i18n), 215
- Compact.mode (field in i18n), 215
- Compact.name (field in i18n), 215
- compile\_file (procedure in typst), 340
- compile\_plural (procedure in gettext), 153
- compile\_text (procedure in typst), 340
- Config (struct in autodoc), 75
  - (struct in odindoc), 241
  - (struct in project), 252
  - (struct in readers/sphinx), 324
- Config.add\_function\_parentheses (field in project), 254
  - (field in readers/sphinx), 325
- Config.add\_module\_names (field in project), 254
  - (field in readers/sphinx), 325
- Config.author (field in project), 252
- Config.autoclass\_content (field in autodoc), 75
  - (field in project), 256
- Config.autodoc (field in readers/sphinx), 326
- Config.autodoc\_class\_signature (field in autodoc), 75
  - (field in project), 256
- Config.autodoc\_default\_options (field in autodoc), 75
  - (field in project), 257
- Config.autodoc\_docstring\_signature (field in autodoc), 75
  - (field in project), 257
- Config.autodoc\_inherit\_docstrings (field in autodoc), 75
  - (field in project), 257
- Config.autodoc\_member\_order (field in autodoc), 75
  - (field in project), 257
- Config.autodoc\_mock\_imports (field in autodoc), 75
  - (field in project), 257
- Config.autodoc\_preserve\_defaults (field in autodoc), 75
  - (field in project), 257
- Config.autodoc\_source\_paths (field in project), 256
- Config.autodoc\_type\_aliases (field in autodoc), 75
  - (field in project), 257
- Config.autodoc\_typehints (field in autodoc), 75
  - (field in project), 257
- Config.autodoc\_typehints\_description\_target (field in autodoc), 75
  - (field in project), 257
- Config.autodoc\_typehints\_format (field in autodoc), 75
  - (field in project), 257
- Config.autodoc\_use\_type\_comments (field in autodoc), 76
  - (field in project), 257
- Config.autodoc\_warningiserror (field in project), 257
- Config.autosectionlabel\_maxdepth (field in project), 255
- Config.autosectionlabel\_prefix\_document (field in project), 255
- Config.build\_dir (field in project), 252
- Config.copyright (field in project), 252
- Config.default\_options (field in odindoc), 241
- Config.default\_role (field in project), 253
  - (field in readers/sphinx), 324
- Config.directive\_aliases (field in project), 255
  - (field in readers/sphinx), 326
- Config.docname (field in readers/sphinx), 324
- Config.exclude\_patterns (field in project), 253
- Config.extensions (field in project), 253
- Config.extlinks (field in project), 255
  - (field in readers/sphinx), 324
- Config.extlinks\_detect\_hardcoded\_links (field in project), 255
- Config.figure\_language\_filename (field in project), 262
- Config.gettext\_additional\_targets (field in project), 262
- Config.gettext\_allow\_fuzzy\_translations (field in project), 262
- Config.gettext\_auto\_build (field in project), 262
- Config.gettext\_compact (field in project), 262
- Config.gettext\_exclude\_patterns (field in project), 262
- Config.gettext\_language\_team (field in project), 262
- Config.gettext\_last\_translator (field in project), 262
- Config.gettext\_location (field in project), 262
- Config.gettext\_uuid (field in project), 262
- Config.graphviz\_dot (field in project), 260
- Config.graphviz\_dot\_args (field in project), 260

- Config.graphviz\_output\_format (field in project), 260
- Config.highlight\_language (field in project), 254
- Config.html\_additional\_pages (field in project), 259
- Config.html\_baseurl (field in project), 259
- Config.html\_context (field in project), 259
- Config.html\_copy\_source (field in project), 259
- Config.html\_css\_files (field in project), 258
- Config.html\_domain\_indices (field in project), 259
- Config.html\_extra\_path (field in project), 258
- Config.html\_favicon (field in project), 258
- Config.html\_file\_suffix (field in project), 259
- Config.html\_js\_files (field in project), 258
- Config.html\_last\_updated\_fmt (field in project), 258
- Config.html\_link\_suffix (field in project), 259
- Config.html\_logo (field in project), 258
- Config.html\_permalinks (field in project), 258
- Config.html\_permalinks\_icon (field in project), 259
- Config.html\_secnumber\_suffix (field in project), 259
- Config.html\_short\_title (field in project), 258
- Config.html\_short\_title\_derived (field in project), 262
- Config.html\_show\_copyright (field in project), 259
- Config.html\_show\_sourcelink (field in project), 259
- Config.html\_show\_sphinx (field in project), 259
- Config.html\_sourcelink\_suffix (field in project), 259
- Config.html\_split\_index (field in project), 259
- Config.html\_static\_path (field in project), 258
- Config.html\_theme (field in project), 258
- Config.html\_theme\_options (field in project), 258
- Config.html\_title (field in project), 258
- Config.html\_title\_derived (field in project), 262
- Config.html\_use\_index (field in project), 259
- Config.include\_patterns (field in project), 253
- Config.include\_roots (field in project), 253
- Config.intersphinx (field in readers/sphinx), 325
- Config.intersphinx\_cache\_limit (field in project), 256
- Config.intersphinx\_disabled\_reftypes (field in project), 256
- Config.intersphinx\_mapping (field in project), 256
- Config.intersphinx\_resolve\_self (field in project), 256
- Config.intersphinx\_sphinx (field in readers/sphinx), 325
- Config.intersphinx\_timeout (field in project), 256
- Config.keep\_warnings (field in project), 255
- Config.labels (field in readers/sphinx), 325
- Config.language (field in project), 253
- Config.locale\_dirs (field in project), 261
- Config.manpages\_url (field in project), 255
- Config.math\_eqref\_format (field in project), 254
- Config.math\_number\_all (field in project), 254
- Config.math\_number\_all (field in readers/sphinx), 324
- Config.math\_numfig (field in project), 254
- Config.mathjax3\_config (field in project), 256
- Config.mathjax4\_config (field in project), 256
- Config.mathjax\_config\_path (field in project), 256
- Config.mathjax\_display (field in project), 256
- Config.mathjax\_inline (field in project), 256
- Config.mathjax\_options (field in project), 256
- Config.mathjax\_path (field in project), 255
- Config.member\_order (field in odindoc), 241
- Config.modindex\_common\_prefix (field in project), 259
- Config.myst\_amsmath\_enable (field in project), 261
- Config.myst\_commonmark\_only (field in project), 261
- Config.myst\_disable\_syntax (field in project), 261
- Config.myst\_dmath\_allow\_digits (field in project), 261
- Config.myst\_dmath\_allow\_labels (field in project), 260
- Config.myst\_dmath\_allow\_space (field in project), 261
- Config.myst\_dmath\_double\_inline (field in project), 261
- Config.myst\_dmath\_enable (field in project), 261
- Config.myst\_enable\_extensions (field in project), 260
- Config.myst\_footnote\_transition (field in project), 261
- Config.myst\_heading\_anchors (field in project), 260



- Config.myst\_html\_meta (field in project), 261
- Config.myst\_linkify\_fuzzy\_links (field in project), 261
- Config.myst\_sub\_delimiters (field in project), 261
- Config.myst\_substitutions (field in project), 261
- Config.myst\_url\_schemes (field in project), 261
- Config.myst\_words\_per\_minute (field in project), 261
- Config.nitpick\_ignore (field in project), 254
- Config.nitpick\_ignore\_regex (field in project), 254
- Config.nitpick\_patterns (field in project), 254
- Config.nitpicky (field in project), 254
- Config.numfig (field in project), 254 (field in readers/sphinx), 324
- Config.numfig\_format (field in project), 254
- Config.numfig\_secnum\_depth (field in project), 254
- Config.object\_types (field in project), 255 (field in readers/sphinx), 326
- Config.odin\_autoapi\_add\_toctree\_entry (field in project), 258
- Config.odin\_autoapi\_dirs (field in project), 257
- Config.odin\_autoapi\_generate\_api\_docs (field in project), 258
- Config.odin\_autoapi\_member\_order (field in project), 258
- Config.odin\_autoapi\_options (field in project), 258
- Config.odin\_autoapi\_root (field in project), 257
- Config.odin\_autodoc (field in readers/sphinx), 326
- Config.odin\_collections (field in project), 257
- Config.option\_emphasise\_placeholders (field in project), 255 (field in readers/sphinx), 325
- Config.pdf\_documents (field in project), 259
- Config.pdf\_font\_paths (field in project), 260
- Config.pdf\_fonts (field in project), 260
- Config.pdf\_logo (field in project), 260
- Config.pdf\_packages (field in project), 260
- Config.pdf\_paper\_size (field in project), 260
- Config.pdf\_preamble (field in project), 260
- Config.pdf\_show\_urls (field in project), 260
- Config.pdf\_toplevel\_sectioning (field in project), 260
- Config.primary\_domain (field in project), 253 (field in readers/sphinx), 324
- Config.project (field in project), 252
- Config.pygments\_dark\_style (field in project), 254
- Config.pygments\_style (field in project), 254
- Config.python\_display\_short\_literal\_types (field in autodoc), 76 (field in project), 257
- Config.release (field in project), 253 (field in readers/sphinx), 325
- Config.root\_doc (field in project), 253
- Config.rst\_epilog (field in project), 253 (field in readers/sphinx), 324
- Config.rst\_prolog (field in project), 253 (field in readers/sphinx), 324
- Config.show\_authors (field in project), 254 (field in readers/sphinx), 324
- Config.smartquotes (field in project), 255
- Config.source\_dir (field in project), 252
- Config.source\_suffix (field in project), 253
- Config.strip\_signature\_backslash (field in autodoc), 76 (field in project), 257
- Config.suppress\_warnings (field in project), 255
- Config.tab\_width (field in autodoc), 76
- Config.templates\_path (field in project), 253
- Config.toc\_object\_entries (field in project), 254 (field in readers/sphinx), 325
- Config.toc\_object\_entries\_show\_parents (field in project), 255 (field in readers/sphinx), 325
- Config.today (field in project), 253 (field in readers/sphinx), 325
- Config.today\_fmt (field in project), 253
- Config.todo\_include\_todos (field in project), 255 (field in readers/sphinx), 324
- Config.translation\_progress\_classes (field in project), 262
- Config.trim\_doctest\_flags (field in project), 255 (field in readers/sphinx), 325
- Config.trim\_footnote\_reference\_space (field in project), 255
- Config.typehint\_fields (field in autodoc), 76
- Config.typst (field in project), 260
- Config.using (field in readers/sphinx), 324
- Config.version (field in project), 253 (field in readers/sphinx), 325
- CONFIG\_FILE (constant in project), 272
- confined (procedure in host), 202
- Confinement (enum in host), 191
- confinement (procedure in host), 202
- Confinement.Outside (enumerator in host), 191
- Confinement.Through\_Link (enumerator in host), 191
- CONFORMANCE (constant in readers/commonmark), 299 (constant in readers/rst), 316

- (constant in readers/sphinx), 329
- (constant in render/html), 329
- (constant in render/text), 330
- (constant in render/typst), 331
- Conformance (enum in core), 86
- Conformance.Conformant (enumerator in core), 86
- Conformance.Partial (enumerator in core), 86
- Conformance.Planned (enumerator in core), 86
- containing\_root (procedure in host), 202
- Content (enum in core), 86
- Content.Blocks (enumerator in core), 86
- Content.Inlines (enumerator in core), 86
- Content.Nothing (enumerator in core), 86
- Content.Structure (enumerator in core), 86
- content\_error (procedure in core), 119
- content\_model (procedure in core), 119
- Content\_Policy (enum in core), 86
- Content\_Policy.Allow (enumerator in core), 87
- Content\_Policy.Omit (enumerator in core), 86
- Content\_Policy.Refuse (enumerator in core), 86
- Content\_Rule (enum in readers/rst), 299
- Content\_Rule.None (enumerator in readers/rst), 299
- Content\_Rule.Optional (enumerator in readers/rst), 299
- Content\_Rule.Required (enumerator in readers/rst), 299
- contents\_scope (procedure in core), 119
- Context (struct in graphdog), 160
- Context.gvc (field in graphdog), 160
- context\_of (procedure in readers/commonmark), 297
- convert (procedure in core), 119
- convert\_artifact (procedure in host), 203
- convert\_text (procedure in core), 119
- copy\_payload (procedure in core), 120
- copy\_subtree (procedure in core), 120
- core (package), 80
- CORE\_CAPACITY (constant in core), 133
- CORE\_DEPTH (constant in core), 133
- CORE\_NODES (constant in core), 133
- CORE\_STACK (constant in core), 134
- CORE\_UNHANDLED\_KIND (constant in core), 134
- CORE\_WORK (constant in core), 134
- count\_tokens (procedure in highlight), 182
- CRASH\_EXIT (constant in host), 213
- crash\_on\_assertion (procedure in host), 203
- create\_environment (procedure in jinja), 236
- curl (foreign import in fetch), 137
- current (procedure in core), 120
- current\_kind (procedure in core), 120
- Cursor (struct in treesitter), 331
- Cursor.ctx (field in treesitter), 331
- Cursor.id (field in treesitter), 331
- Cursor.tree (field in treesitter), 331
- cursor\_delete (procedure in treesitter), 333

- cursor\_field\_name (procedure in treesitter), 333
- cursor\_first\_child (procedure in treesitter), 333
- cursor\_new (procedure in treesitter), 333
- cursor\_next\_sibling (procedure in treesitter), 333
- cursor\_node (procedure in treesitter), 333
- cursor\_parent (procedure in treesitter), 334
- custom (procedure in readers/rst), 311
- custom\_attribute (procedure in core), 120
- cut (procedure in readers/rst), 311
- Cycler (struct in jinja), 227
- Cycler.items (field in jinja), 227
- Cycler.pos (field in jinja), 227

## D

- Decl (struct in odindoc), 241
- Decl.attributes (field in odindoc), 241
- Decl.availability (field in odindoc), 242
- Decl.docs (field in odindoc), 241
- Decl.file (field in odindoc), 242
- Decl.foreign\_lib (field in odindoc), 241
- Decl.kind (field in odindoc), 241
- Decl.line (field in odindoc), 242
- Decl.members (field in odindoc), 241
- Decl.name (field in odindoc), 241
- Decl.order (field in odindoc), 242
- Decl.private (field in odindoc), 241
- Decl.signature (field in odindoc), 241
- Decl.targets (field in odindoc), 241
- Decl.variant (field in odindoc), 242
- decode\_last\_rune (procedure in core), 120
- decode\_rune (procedure in core), 120
- dedent (procedure in readers/rst), 311
- deeper\_than (procedure in treesitter), 334
- Def (struct in pyscan), 274
- Def.annotation (field in pyscan), 275
- Def.class (field in pyscan), 275
- Def.function (field in pyscan), 275
- Def.kind (field in pyscan), 274
- Def.line (field in pyscan), 275
- Def.member (field in pyscan), 275
- Def.name (field in pyscan), 274
- Def.node (field in pyscan), 274
- Def.runtime (field in pyscan), 274
- Def.target (field in pyscan), 275
- Def.tuple (field in pyscan), 275
- Def.value (field in pyscan), 275
- Def\_Kind (enum in pyscan), 275
- Def\_Kind.Annotation (enumerator in pyscan), 275
- Def\_Kind.Assign (enumerator in pyscan), 275
- Def\_Kind.Class (enumerator in pyscan), 275
- Def\_Kind.Function (enumerator in pyscan), 275
- Def\_Kind.Import (enumerator in pyscan), 275
- Def\_Kind.Import\_From (enumerator in pyscan), 276
- Def\_Kind.Type\_Alias (enumerator in pyscan), 276
- def\_object (procedure in pyscan), 284
- DEFAULT\_BUDGET (constant in host), 213

- default\_config (procedure in autodoc), 78
- (procedure in readers/sphinx), 328
- DEFAULT\_FIGURE\_FILENAME (constant in i18n), 223
- default\_limits (procedure in core), 120
- default\_merge\_options (procedure in gettext), 153
- Default\_Option (struct in autodoc), 76
- Default\_Option.flag (field in autodoc), 76
- Default\_Option.name (field in autodoc), 76
- Default\_Option.value (field in autodoc), 76
- default\_policy (procedure in core), 121
- default\_read\_options (procedure in core), 121
- default\_request (procedure in core), 121
- default\_settings (procedure in readers/commonmark), 297
- (procedure in readers/rst), 311
- default\_stack\_budget (procedure in host), 203
- DEFAULT\_STACK\_BYTES (constant in core), 134
- default\_write\_options (procedure in gettext), 153
- defaults (package), 134
- delete\_error (procedure in graphdog), 163
- describe (procedure in host), 203
- describe\_all (procedure in host), 203
- destroy (procedure in fetch), 137
- (procedure in gettext), 153
- (procedure in pyscan), 284
- destroy\_environment (procedure in jinja), 236
- destroy\_error (procedure in gettext), 153
- (procedure in jinja), 236
- destroy\_storage (procedure in host), 203
- destroy\_template (procedure in i18n), 220
- destroy\_translator (procedure in gettext), 153
- diagnostic (procedure in core), 121
- Diagnostic (struct in core), 87
- (struct in typst), 338
- Diagnostic.args (field in core), 87
- Diagnostic.column (field in typst), 338
- Diagnostic.file (field in typst), 338
- Diagnostic.line (field in typst), 338
- Diagnostic.message (field in typst), 338
- Diagnostic.severity (field in typst), 338
- Diagnostic.span (field in core), 87
- Diagnostic.stdin (field in typst), 338
- Diagnostic.using (field in core), 87
- Diagnostic\_Arg (struct in core), 87
- Diagnostic\_Arg.int (field in core), 87
- Diagnostic\_Arg.kind (field in core), 87
- Diagnostic\_Arg.span (field in core), 87
- Diagnostic\_Arg.static (field in core), 87
- DIAGNOSTICS\_SCHEMA (constant in host), 213
- dict (procedure in jinja), 236
- Dict (struct in jinja), 227
- Dict allocator (field in jinja), 227
- Dict.index (field in jinja), 227
- Dict.keys (field in jinja), 227
- Dict.kind (field in jinja), 227
- Dict.name (field in jinja), 227
- Dict.values (field in jinja), 227
- Dict\_Kind (enum in jinja), 227
- Dict\_Kind.Dict (enumerator in jinja), 227
- Dict\_Kind.Module (enumerator in jinja), 227
- Dict\_Kind.Namespace (enumerator in jinja), 227
- Dir\_Opt (struct in readers/rst), 300
- Dir\_Opt.lines (field in readers/rst), 300
- Dir\_Opt.span (field in readers/rst), 300
- Dir\_Opt.spec (field in readers/rst), 300
- Dir\_Spec (struct in readers/rst), 300
- Dir\_Spec.arg (field in readers/rst), 300
- Dir\_Spec.content (field in readers/rst), 300
- Dir\_Spec.final\_ws (field in readers/rst), 300
- Dir\_Spec.kind (field in readers/rst), 300
- Dir\_Spec.name (field in readers/rst), 300
- Dir\_Spec.opts (field in readers/rst), 300
- Dir\_Spec.value (field in readers/rst), 300
- Directive (struct in readers/commonmark), 287
- (struct in readers/rst), 300
- Directive.arg (field in readers/commonmark), 288
- (field in readers/rst), 300
- Directive.arg\_block (field in readers/rst), 301
- Directive.classes (field in readers/commonmark), 288
- Directive.content (field in readers/commonmark), 288
- (field in readers/rst), 301
- Directive.ext (field in readers/rst), 301
- Directive.name (field in readers/commonmark), 287
- Directive.name\_span (field in readers/commonmark), 287
- (field in readers/rst), 300
- Directive.nopts (field in readers/rst), 300
- Directive.options (field in readers/commonmark), 288
- Directive.opts (field in readers/rst), 300
- Directive.sections (field in readers/commonmark), 288
- (field in readers/rst), 301
- Directive.source (field in readers/commonmark), 288
- Directive.span (field in readers/commonmark), 288
- (field in readers/rst), 300
- Directive.spec (field in readers/rst), 300
- Directive.subst (field in readers/rst), 301
- Directive.subst\_name (field in readers/rst), 301
- Directive\_Alias (struct in readers/sphinx), 326
- Directive\_Alias.name (field in readers/sphinx), 326
- Directive\_Alias.target (field in readers/sphinx), 326
- Directive\_Handler (struct in readers/commonmark), 288
- (struct in readers/rst), 301
- Directive\_Handler.arg (field in readers/rst), 301

- Directive\_Handler.content (field in readers/rst), 301
  - Directive\_Handler.final\_ws (field in readers/rst), 301
  - Directive\_Handler.name (field in readers/commonmark), 288 (field in readers/rst), 301
  - Directive\_Handler.options (field in readers/rst), 301
  - Directive\_Handler.run (field in readers/commonmark), 288 (field in readers/rst), 301
  - directive\_node (procedure in readers/rst), 311
  - Directive\_Proc (type in readers/commonmark), 288 (type in readers/rst), 301
  - Directive\_Result (enum in readers/commonmark), 288
  - Directive\_Result.Failed (enumerator in readers/commonmark), 288
  - Directive\_Result.Ran (enumerator in readers/commonmark), 288
  - Directive\_Result.Unknown (enumerator in readers/commonmark), 288
  - directive\_shape (procedure in readers/rst), 311
  - Directive\_Shape (struct in readers/commonmark), 289 (struct in readers/rst), 301
  - Directive\_Shape.arg (field in readers/rst), 302
  - Directive\_Shape.argument (field in readers/commonmark), 289
  - Directive\_Shape.content (field in readers/rst), 302
  - Directive\_Shape.final\_ws (field in readers/rst), 302
  - Directive\_Shape.options (field in readers/commonmark), 289 (field in readers/rst), 302
  - DIRECTIVE\_UNKNOWN (constant in readers/rst), 316
  - discharge (procedure in host), 203
  - Disk (struct in host), 191
  - Disk.chunks (field in host), 192
  - Disk.dir (field in host), 191
  - Disk.failure (field in host), 192
  - Disk.limit (field in host), 191
  - Disk.mutex (field in host), 192
  - Disk.peak (field in host), 191
  - Disk.refused (field in host), 192
  - Disk.used (field in host), 191
  - disk\_allocator (procedure in host), 204
  - DISK\_CHUNK\_BYTES (constant in host), 213
  - disk\_destroy (procedure in host), 204
  - Disk\_Failure (enum in host), 192
  - Disk\_Failure.Limit (enumerator in host), 192
  - Disk\_Failure.None (enumerator in host), 192
  - Disk\_Failure.Space (enumerator in host), 192
  - Disk\_Failure.System (enumerator in host), 192
  - DISK\_MARGIN (constant in host), 213
  - disk\_owns (procedure in host), 204
  - display\_path (procedure in host), 204
  - Doc (struct in pyscan), 276
  - Doc.ok (field in pyscan), 276
  - Doc.text (field in pyscan), 276
  - document (procedure in autodoc), 78 (procedure in odindoc), 247
  - Document (struct in core), 87
  - Document.bank (field in core), 87
  - Document.generation (field in core), 87
  - Document.revision (field in core), 87
  - Document.ws (field in core), 87
  - Document\_Reading (enum in core), 88
  - Document\_Reading.Docutils (enumerator in core), 88
  - Document\_Reading.Sphinx (enumerator in core), 88
  - document\_valid (procedure in core), 121
  - domain\_for (procedure in i18n), 220
  - domain\_plural (procedure in gettext), 153
  - domain\_plural\_with\_context (procedure in gettext), 153
  - domain\_text (procedure in gettext), 154
  - domain\_with\_context (procedure in gettext), 154
  - dump (procedure in host), 204
  - dump\_tree (procedure in core), 121
- ## E
- each\_chunk (procedure in host), 204
  - edit\_plan (procedure in gds), 141
  - embed\_begin (procedure in readers/rst), 311
  - embed\_end (procedure in readers/rst), 312
  - EMBEDDED (constant in typst), 341
  - embedded\_catalog (procedure in i18n), 220
  - emit (procedure in core), 121
  - emit\_byte (procedure in core), 121
  - emit\_int (procedure in core), 121
  - emit\_mark (procedure in core), 122
  - Emit\_Mode (enum in core), 88
  - Emit\_Mode.Measure (enumerator in core), 88
  - Emit\_Mode.Write (enumerator in core), 88
  - emit\_preamble (procedure in render/typst), 331
  - emit\_rune (procedure in core), 122
  - Emitter (struct in core), 88
  - Emitter.mapped (field in core), 88
  - Emitter.mode (field in core), 88
  - Emitter.out (field in core), 88
  - Emitter.source\_map (field in core), 88
  - Emitter.written (field in core), 88
  - encode\_rune (procedure in core), 122
  - end\_byte (procedure in treesitter), 334
  - end\_node (procedure in core), 122
  - end\_point (procedure in treesitter), 334
  - Engine (enum in graphdog), 160
  - Engine.Circo (enumerator in graphdog), 160
  - Engine.Dot (enumerator in graphdog), 160
  - Engine.Fdp (enumerator in graphdog), 160
  - Engine.Neato (enumerator in graphdog), 160



- Engine.Osage (enumerator in graphdog), 160
- Engine.Patchwork (enumerator in graphdog), 160
- Engine.Sfdp (enumerator in graphdog), 160
- Engine.Twopi (enumerator in graphdog), 160
- engine\_name (procedure in graphdog), 163
- ENGLISH\_LABELS (variable in core), 134
- enter\_source (procedure in readers/rst), 312
- Environment (struct in jinja), 228
- Environment allocator (field in jinja), 228
- Environment.arena (field in jinja), 228
- Environment.backing (field in jinja), 228
- Environment.cache (field in jinja), 228
- Environment.filters (field in jinja), 228
- Environment.globals (field in jinja), 228
- Environment.options (field in jinja), 228
- Environment.out\_of\_memory (field in jinja), 228
- Environment.stack (field in jinja), 228
- Environment.tests (field in jinja), 228
- equal (procedure in treesitter), 334
- Error (enum in fetch), 135
  - (enum in inventory), 223
  - (struct in gettext), 144
  - (struct in graphdog), 161
  - (struct in jinja), 228
- Error.Bad\_Compression (enumerator in inventory), 224
- Error.column (field in gettext), 144
  - (field in graphdog), 161
- Error.excerpt (field in gettext), 145
  - (field in jinja), 229
- Error.file (field in gettext), 144
- Error.hint (field in gettext), 145
  - (field in graphdog), 161
  - (field in jinja), 229
- Error.kind (field in gettext), 144
  - (field in graphdog), 161
  - (field in jinja), 228
- Error.line (field in gettext), 144
  - (field in graphdog), 161
  - (field in jinja), 228
- Error.message (field in gettext), 145
  - (field in graphdog), 161
  - (field in jinja), 228
- Error.Missing (enumerator in fetch), 135
- Error.near (field in graphdog), 161
- Error.Network (enumerator in fetch), 135
- Error.None (enumerator in fetch), 135
  - (enumerator in inventory), 223
- Error.Out\_Of\_Memory (enumerator in inventory), 224
- Error.Status (enumerator in fetch), 136
- Error.template (field in jinja), 228
- Error.Timeout (enumerator in fetch), 136
- Error.Too\_Large (enumerator in fetch), 136
- Error.Truncated (enumerator in inventory), 224
- Error.Unknown\_Version (enumerator in inventory), 224
- Error.Unsupported (enumerator in fetch), 135
- Error\_Kind (enum in gettext), 145
  - (enum in graphdog), 161
  - (enum in jinja), 229
- Error\_Kind.Charset (enumerator in gettext), 145
- Error\_Kind.Duplicate (enumerator in gettext), 145
- Error\_Kind.Empty\_Input (enumerator in graphdog), 161
- Error\_Kind.Encoding (enumerator in gettext), 145
- Error\_Kind.File\_Refused (enumerator in graphdog), 162
- Error\_Kind.Html\_Label (enumerator in graphdog), 161
- Error\_Kind.Io (enumerator in gettext), 145
- Error\_Kind.Layout (enumerator in graphdog), 162
- Error\_Kind.Limit (enumerator in jinja), 229
- Error\_Kind.Memory (enumerator in gettext), 145
  - (enumerator in jinja), 229
- Error\_Kind.Missing\_Engine (enumerator in graphdog), 161
- Error\_Kind.Missing\_Format (enumerator in graphdog), 161
- Error\_Kind.Mo\_Format (enumerator in gettext), 145
- Error\_Kind.None (enumerator in gettext), 145
  - (enumerator in graphdog), 161
  - (enumerator in jinja), 229
- Error\_Kind.Not\_Found (enumerator in jinja), 229
- Error\_Kind.Not\_Initialized (enumerator in graphdog), 162
- Error\_Kind.Out\_Of\_Memory (enumerator in graphdog), 162
- Error\_Kind.Plural (enumerator in gettext), 145
- Error\_Kind.Render (enumerator in graphdog), 162
- Error\_Kind.Runtime (enumerator in jinja), 229
- Error\_Kind.Syntax (enumerator in gettext), 145
  - (enumerator in graphdog), 161
  - (enumerator in jinja), 229
- Error\_Kind.Type (enumerator in jinja), 229
- Error\_Kind.Undefined (enumerator in jinja), 229
- error\_title (procedure in graphdog), 164
- ERROR\_TITLES (variable in gettext), 159
- escape (procedure in jinja), 236
- escape\_string (procedure in gettext), 154
  - (procedure in jinja), 236
- evaluate (procedure in jinja), 236
- excerpt\_at (procedure in host), 204
- exchange (procedure in host), 204
- exit\_code (procedure in host), 205
- exported (procedure in pycan), 284
- Extension (struct in readers/
  - commonmark), 289
  - (struct in readers/rst), 302
- Extension.begin (field in readers/
  - commonmark), 289

- (field in readers/rst), 302
- Extension.directive (field in readers/  
commonmark), 289
- Extension.directives (field in readers/  
commonmark), 289
- (field in readers/rst), 302
- Extension.end (field in readers/  
commonmark), 289
- (field in readers/rst), 302
- Extension.lookup (field in readers/rst), 302
- Extension.prologue (field in readers/rst),  
302
- Extension.resolve (field in readers/  
commonmark), 290
- Extension.role (field in readers/  
commonmark), 289
- (field in readers/rst), 302
- Extension.roles (field in readers/  
commonmark), 289
- (field in readers/rst), 302
- Extension.shape (field in readers/  
commonmark), 289
- Extension.substitution (field in readers/  
rst), 302
- Extension.table (field in readers/  
commonmark), 289
- Extension.user (field in readers/  
commonmark), 290
- (field in readers/rst), 303
- extension\_as (procedure in core), 122
- Extlink (struct in readers/sphinx), 326
- Extlink.caption (field in readers/sphinx),  
326
- Extlink.name (field in readers/sphinx), 326
- Extlink.url (field in readers/sphinx), 326
- extra\_anchor (procedure in core), 122
- extra\_text (procedure in core), 122
- extract (procedure in i18n), 221
- Extract\_Options (struct in i18n), 215
- Extract\_Options.base (field in i18n), 215
- Extract\_Options.docname (field in i18n),  
215
- Extract\_Options.location (field in i18n),  
215
- Extract\_Options.path (field in i18n), 215
- Extract\_Options.targets (field in i18n), 215
- Extract\_Options.uuid (field in i18n), 215

## F

- fail (procedure in jinja), 237
- fetch (package), 135
- field (procedure in treesitter), 334
- Field (struct in gds), 138
- Field.literal (field in gds), 138
- Field.name (field in gds), 138
- Field\_Id (type in treesitter), 332
- field\_name\_for\_child (procedure in  
treesitter), 334
- figure\_filename (procedure in i18n), 221
- figure\_for\_language (procedure in i18n),  
221
- File\_Access (struct in graphdog), 162
- File\_Access.procedure (field in graphdog),  
162
- File\_Access.user (field in graphdog), 162

- FILE\_ATTRIBUTES (constant in  
graphdog), 165
- File\_Read (struct in odindoc), 242
- (struct in pyscan), 276
- File\_Read.path (field in odindoc), 242
- (field in pyscan), 276
- File\_Read.text (field in odindoc), 242
- (field in pyscan), 276
- File\_Reader (struct in odindoc), 242
- (struct in pyscan), 276
- File\_Reader.list (field in odindoc), 242
- File\_Reader.procedure (field in odindoc),  
242
- (field in pyscan), 276
- File\_Reader.reserve (field in odindoc), 242
- File\_Reader.user (field in odindoc), 242
- (field in pyscan), 276
- Files (struct in host), 192
- Files.canonical (field in host), 193
- Files.ledger (field in host), 193
- Files.max\_bytes (field in host), 192
- Files.project (field in host), 193
- Files.roots (field in host), 192
- files\_provider (procedure in host), 205
- filesystem\_loader (procedure in jinja), 237
- fill\_labels (procedure in i18n), 221
- find (procedure in gettext), 154
- (procedure in inventory), 225
- find\_catalog\_file (procedure in gettext),  
154
- find\_def (procedure in pyscan), 284
- find\_folded (procedure in inventory), 225
- find\_lexer (procedure in highlight), 182
- find\_lexer\_for\_filename (procedure in  
highlight), 182
- find\_program (procedure in typst), 340
- find\_reader (procedure in core), 122
- find\_renderer (procedure in core), 122
- find\_role (procedure in readers/rst), 312
- find\_style (procedure in highlight), 182
- first\_child (procedure in core), 122
- first\_problem (procedure in core), 123
- fitting\_budget (procedure in host), 205
- Flag\_Or\_Name (struct in project), 262
- Flag\_Or\_Name.name (field in project), 262
- Flag\_Or\_Name.on (field in project), 262
- float (procedure in jinja), 237
- Folder\_Entry (struct in host), 193
- Folder\_Entry.inode (field in host), 193
- Folder\_Entry.name (field in host), 193
- Folder\_Entry.type (field in host), 193
- folders\_of (procedure in host), 205
- footnote\_symbol (procedure in readers/rst/  
inline), 321
- Format (enum in graphdog), 162
- (enum in host), 193
- Format.Commonmark (enumerator in  
host), 193
- Format.Dot (enumerator in graphdog), 162
- Format.Png (enumerator in graphdog), 162
- Format.Rst (enumerator in host), 193
- Format.Svg (enumerator in graphdog), 162
- Format.Typst (enumerator in host), 193
- format\_date (procedure in i18n), 221
- format\_error (procedure in gettext), 154
- (procedure in graphdog), 164
- (procedure in jinja), 237

format\_message (procedure in graphdog), 164  
 format\_stats (procedure in gettext), 154  
 Frame (struct in readers/rst), 303  
 Frame.src (field in readers/rst), 303  
 Frame.tab (field in readers/rst), 303  
 Frame.text (field in readers/rst), 303  
 FULL\_VERSION (constant in cli), 79  
 function (procedure in jinja), 237  
 Function (struct in pyscan), 276  
 Function.class (field in pyscan), 277  
 Function.decorators (field in pyscan), 277  
 Function.doc (field in pyscan), 277  
 Function.is\_async (field in pyscan), 277  
 Function.is\_lambda (field in pyscan), 277  
 Function.module (field in pyscan), 276  
 Function.name (field in pyscan), 276  
 Function.node (field in pyscan), 276  
 Function.params (field in pyscan), 277  
 Function.qualname (field in pyscan), 276  
 Function.returns (field in pyscan), 277  
 Function.type\_args (field in pyscan), 277  
 Function.type\_comment (field in pyscan), 277  
 function.type\_returns (field in pyscan), 277  
 FUZZY\_MIN\_WORK (constant in gettext), 158  
 FUZZY\_THRESHOLD (constant in gettext), 158  
 FUZZY\_WORK\_PER\_BYTE (constant in gettext), 159

## G

gds (package), 137  
 get (procedure in fetch), 137  
   (procedure in jinja), 237  
 get\_attr (procedure in pyscan), 284  
 get\_opt (procedure in readers/rst), 312  
 gettext (package), 143  
 graphdog (package), 159  
 graphviz (foreign import in graphdog), 166  
 grow\_session (procedure in host), 205  
 Guard (struct in host), 193  
 Guard.backing (field in host), 194  
 Guard.refused (field in host), 194  
 Guard.tripped (field in host), 194  
 guard\_allocator (procedure in host), 205  
 guard\_problem (procedure in host), 205  
 guard\_tripped (procedure in host), 206  
 guess\_lexer (procedure in highlight), 183

## H

has\_arg (procedure in jinja), 237  
 has\_children (procedure in core), 123  
 has\_class (procedure in core), 123  
 has\_error (procedure in treesitter), 334  
 has\_errors (procedure in highlight), 183  
 has\_option (procedure in readers/rst), 312  
 head (procedure in readers/rst), 312  
 header\_field (procedure in gettext), 155  
 Header\_Field (struct in gettext), 145  
 Header\_Field.name (field in gettext), 146  
 Header\_Field.value (field in gettext), 146  
 highlight (package), 166  
 highlight/guidedoc (package), 186  
 highlight\_html (procedure in highlight), 183

highlight\_html\_lines (procedure in highlight), 183  
 highlight\_typst (procedure in highlight), 183  
 highlighter (procedure in highlight/guidedoc), 186  
 Highlighter (struct in core), 88  
 Highlighter.procedure (field in core), 89  
 Highlighter.user (field in core), 89  
 Hook\_Proc (type in readers/rst), 303  
 host (package), 187  
 Host (struct in readers/commonmark), 290  
   (struct in readers/rst), 303  
 Host.b (field in readers/commonmark), 290  
 Host.body (field in readers/rst), 303  
 Host.ctx (field in readers/commonmark), 290  
 Host.disabled (field in readers/commonmark), 290  
 Host.ext (field in readers/commonmark), 290  
 Host.html\_format (field in readers/commonmark), 291  
 Host.inline (field in readers/rst), 303  
 Host.items (field in readers/rst), 303  
 Host.label\_tmp (field in readers/commonmark), 291  
 Host.labels (field in readers/commonmark), 290  
 Host.myst (field in readers/commonmark), 290  
 Host.names (field in readers/commonmark), 290  
 Host.nrefs (field in readers/commonmark), 290  
 Host.refs (field in readers/commonmark), 290  
 Host.rel\_docs (field in readers/commonmark), 291  
 Host.rel\_images (field in readers/commonmark), 291  
 Host.settings (field in readers/commonmark), 290  
 Host.syntax (field in readers/commonmark), 290  
 Host.user (field in readers/rst), 303  
 Host.using (field in readers/commonmark), 291  
 Host\_Directive (struct in readers/rst), 303  
 Host\_Directive.arg (field in readers/rst), 304  
 Host\_Directive.content (field in readers/rst), 304  
 Host\_Directive.name (field in readers/rst), 304  
 Host\_Directive.name\_span (field in readers/rst), 304  
 Host\_Directive.options (field in readers/rst), 304  
 Host\_Directive.sections (field in readers/rst), 304  
 Host\_Directive.span (field in readers/rst), 304

host\_lines (procedure in readers/rst), 312  
 Host\_Option (struct in readers/rst), 304  
 Host\_Option.name (field in readers/rst), 304  
 Host\_Option.span (field in readers/rst), 304  
 Host\_Option.value (field in readers/rst), 304  
 host\_views (procedure in readers/rst), 312  
 Hosted\_Result (enum in readers/rst), 304  
 Hosted\_Result.Failed (enumerator in readers/rst), 304  
 Hosted\_Result.Ran (enumerator in readers/rst), 304  
 Hosted\_Result.Unknown (enumerator in readers/rst), 304  
 Html\_Lines (struct in highlight), 167  
 Html\_Lines.emphasized (field in highlight), 167  
 Html\_Lines.first (field in highlight), 167  
 Html\_Lines.user (field in highlight), 167

## I

i18n (package), 214  
 id\_fold (procedure in core), 123  
 identity (procedure in host), 206  
 Identity (struct in host), 194  
 Identity.device (field in host), 194  
 Identity.inode (field in host), 194  
 Image\_Exists (struct in i18n), 215  
 Image\_Exists.procedure (field in i18n), 215  
 Image\_Exists.user (field in i18n), 216  
 Image\_Localization (struct in i18n), 216  
 Image\_Localization.docname (field in i18n), 216  
 Image\_Localization.exists (field in i18n), 216  
 Image\_Localization.format (field in i18n), 216  
 Image\_Localization.language (field in i18n), 216  
 Import (struct in odindoc), 243  
 Import.alias (field in odindoc), 243  
 Import.id (field in odindoc), 243  
 INCLUDE\_MISSING (constant in core), 134  
 include\_paths (procedure in readers/commonmark), 297  
 include\_source (procedure in core), 123  
 indent\_of (procedure in readers/rst), 312  
 Index (struct in gettext), 146  
 Index.built (field in gettext), 146  
 Index.entries (field in gettext), 146  
 Index.forms (field in gettext), 146  
 Index.fuzzy (field in gettext), 146  
 Index.pool (field in gettext), 146  
 Index.slots (field in gettext), 146  
 Index\_Entry (struct in gettext), 146  
 Index\_Entry.count (field in gettext), 146  
 Index\_Entry.first (field in gettext), 146  
 Index\_Entry.message (field in gettext), 146  
 Index\_Part (struct in i18n), 216  
 Index\_Part.first (field in i18n), 216  
 Index\_Part.length (field in i18n), 216  
 Index\_Part.text (field in i18n), 216  
 index\_parts (procedure in i18n), 221  
 index\_plan (procedure in gds), 141  
 init (procedure in graphdog), 164  
   (procedure in odindoc), 247  
   (procedure in pyscan), 284  
 init\_workspace (procedure in core), 123  
 init\_workspace\_from\_bytes (procedure in core), 123  
 init\_workspace\_with\_plan (procedure in core), 123  
 inline (procedure in readers/rst), 312  
 inline\_node (procedure in readers/rst), 313  
 Inline\_State (struct in readers/rst/inline), 317  
 Inline\_State.ctx (field in readers/rst/inline), 317  
 Inline\_State.default\_role (field in readers/rst/inline), 317  
 Inline\_State.hook (field in readers/rst/inline), 317  
 Inline\_State.options (field in readers/rst/inline), 317  
 Inline\_State.roles (field in readers/rst/inline), 317  
 Inline\_State.roles\_used (field in readers/rst/inline), 317  
 Inline\_State.source (field in readers/rst/inline), 317  
 install\_crash\_reporter (procedure in host), 206  
 integer (procedure in jinja), 238  
 interrupted\_problem (procedure in host), 206  
 Intersphinx\_Project (struct in project), 263  
 Intersphinx\_Project.locations (field in project), 263  
 Intersphinx\_Project.name (field in project), 263  
 Intersphinx\_Project.uri (field in project), 263  
 Intl\_Action (enum in project), 263  
 Intl\_Action.Create (enumerator in project), 263  
 Intl\_Action.Not\_Changed (enumerator in project), 263  
 Intl\_Action.Update (enumerator in project), 263  
 Intl\_File (struct in project), 263  
 Intl\_File.action (field in project), 263  
 Intl\_File.path (field in project), 263  
 Intl\_File.stats (field in project), 263  
 Intl\_Options (struct in project), 263  
 Intl\_Options.languages (field in project), 264  
 Intl\_Options.locale\_dir (field in project), 263  
 Intl\_Options.obsolete (field in project), 264  
 Intl\_Options.pot\_dir (field in project), 263  
 Intl\_Options.width (field in project), 264  
 intl\_stat (procedure in project), 270  
 intl\_update (procedure in project), 270  
 INVALID\_NODE (type in core), 89  
 inventory (package), 223  
 Inventory (struct in inventory), 224  
 Inventory.by\_type (field in inventory), 224  
 Inventory.folded (field in inventory), 224



Inventory.items (field in inventory), 224  
 Inventory.project (field in inventory), 224  
 Inventory.version (field in inventory), 224  
 io\_problem (procedure in host), 206  
 is\_ascii\_alnum (procedure in core), 124  
 is\_ascii\_digit (procedure in core), 124  
 is\_ascii\_punct (procedure in core), 124  
 is\_ascii\_space (procedure in core), 124  
 is\_builtin (procedure in pyscan), 285  
 is\_builtin\_exception (procedure in  
   pyscan), 285  
 is\_error (procedure in treesitter), 334  
 is\_exception (procedure in pyscan), 285  
 is\_extra (procedure in treesitter), 334  
 is\_indented (procedure in readers/rst), 313  
 is\_markup (procedure in jinja), 238  
 is\_missing (procedure in treesitter), 335  
 is\_named (procedure in treesitter), 335  
 is\_none (procedure in jinja), 238  
 is\_null (procedure in treesitter), 335  
 is\_stage\_name (procedure in host), 206  
 is\_undefined (procedure in jinja), 238  
 is\_unicode\_punctuation (procedure in  
   core), 124  
 is\_unicode\_whitespace (procedure in  
   core), 124  
 ISSUE\_TRACKER (constant in host), 213  
 Item (struct in inventory), 224  
 Item.display (field in inventory), 225  
 Item.name (field in inventory), 224  
 Item.priority (field in inventory), 225  
 Item.project (field in inventory), 224  
 Item.type (field in inventory), 224  
 Item.uri (field in inventory), 225  
 Item.version (field in inventory), 224  
 Iterator (struct in treesitter), 332  
 Iterator.count (field in treesitter), 332  
 Iterator.index (field in treesitter), 332  
 Iterator.named (field in treesitter), 332  
 Iterator.node (field in treesitter), 332

## J

jinja (package), 226  
 join (procedure in project), 270  
 join\_lines (procedure in readers/rst), 313

## K

kept\_message (procedure in core), 124  
 kernel32 (foreign import in host), 214  
 keys (procedure in jinja), 238  
 Keyword (struct in jinja), 229  
   (struct in pyscan), 277  
 Keyword.name (field in jinja), 229  
   (field in pyscan), 277  
 Keyword.value (field in jinja), 229  
   (field in pyscan), 277  
 Kind (enum in odindoc), 243  
 kind (procedure in core), 124  
   (procedure in treesitter), 335  
 Kind.Bit\_Field (enumerator in odindoc),  
   243  
 Kind.Bit\_Set (enumerator in odindoc), 243  
 Kind.Const (enumerator in odindoc), 243  
 Kind.Enum (enumerator in odindoc), 243  
 Kind.Foreign\_Import (enumerator in  
   odindoc), 243  
 Kind.Proc (enumerator in odindoc), 243

Kind.Proc\_Group (enumerator in odindoc),  
   243  
 Kind.Struct (enumerator in odindoc), 243  
 Kind.Type (enumerator in odindoc), 243  
 Kind.Union (enumerator in odindoc), 243  
 Kind.Var (enumerator in odindoc), 243  
 kind\_class (procedure in highlight), 183  
 kind\_is (procedure in highlight), 183  
 kind\_name (procedure in core), 124  
   (procedure in highlight), 183  
 kind\_parent (procedure in highlight), 183

## L

Label (enum in core), 89  
 label (procedure in core), 124  
 Label.Arguments (enumerator in core), 91  
 Label.Attention (enumerator in core), 89  
 Label.Availability (enumerator in core), 91  
 Label.Caution (enumerator in core), 89  
 Label.Code\_Author (enumerator in core),  
   90  
 Label.Contents (enumerator in core), 89  
 Label.Danger (enumerator in core), 89  
 Label.Error (enumerator in core), 89  
 Label.Hint (enumerator in core), 89  
 Label.Important (enumerator in core), 89  
 Label.Link\_Code (enumerator in core), 90  
 Label.Link\_Definition (enumerator in  
   core), 90  
 Label.Link\_Equation (enumerator in core),  
   90  
 Label.Link\_Heading (enumerator in core),  
   90  
 Label.Link\_Image (enumerator in core), 90  
 Label.Link\_Table (enumerator in core), 90  
 Label.Link\_Term (enumerator in core), 90  
 Label.Module\_Author (enumerator in  
   core), 90  
 Label.Note (enumerator in core), 89  
 Label.Parameters (enumerator in core), 90  
 Label.Raisees (enumerator in core), 90  
 Label.Return\_Type (enumerator in core),  
   90  
 Label.Return\_Values (enumerator in core),  
   90  
 Label>Returns (enumerator in core), 90  
 Label.Section\_Author (enumerator in  
   core), 90  
 Label.See\_Also (enumerator in core), 89  
 Label.Template\_Parameters (enumerator in  
   core), 91  
 Label.Throws (enumerator in core), 91  
 Label.Tip (enumerator in core), 89  
 Label.TODO (enumerator in core), 89  
 Label.Variables (enumerator in core), 90  
 Label.Version\_Added (enumerator in core),  
   89  
 Label.Version\_Changed (enumerator in  
   core), 90  
 Label.Version\_Deprecated (enumerator in  
   core), 90  
 Label.Version\_Removed (enumerator in  
   core), 90  
 Label.Warning (enumerator in core), 89  
 LABELS (constant in gds), 143  
 Labels (type in core), 91

- Language (struct in treesitter), 332
- language\_candidates (procedure in gettext), 155
- leave\_source (procedure in readers/rst), 313
- Ledger (struct in host), 194
- Ledger.budget (field in host), 194
- Ledger.held (field in host), 194
- Ledger.refused (field in host), 194
- length (procedure in jinja), 238
- Lexer (enum in highlight), 167
- Lexer.Bash (enumerator in highlight), 168
- Lexer.C (enumerator in highlight), 167
- Lexer.Console (enumerator in highlight), 168
- Lexer.Cpp (enumerator in highlight), 167
- Lexer.Csharp (enumerator in highlight), 168
- Lexer.Css (enumerator in highlight), 168
- Lexer.Default (enumerator in highlight), 169
- Lexer.Diff (enumerator in highlight), 169
- Lexer.Dockerfile (enumerator in highlight), 169
- Lexer.Dot (enumerator in highlight), 169
- Lexer.Go (enumerator in highlight), 168
- Lexer.Guess (enumerator in highlight), 169
- Lexer.Html (enumerator in highlight), 168
- Lexer.Ini (enumerator in highlight), 168
- Lexer.Java (enumerator in highlight), 168
- Lexer.Javascript (enumerator in highlight), 168
- Lexer.Json (enumerator in highlight), 168
- Lexer.Kotlin (enumerator in highlight), 168
- Lexer.Latex (enumerator in highlight), 169
- Lexer.Makefile (enumerator in highlight), 169
- Lexer.Markdown (enumerator in highlight), 169
- Lexer.Odin (enumerator in highlight), 168
- Lexer.Powershell (enumerator in highlight), 169
- Lexer.Pycon (enumerator in highlight), 167
- Lexer.Pytb (enumerator in highlight), 167
- Lexer.Python (enumerator in highlight), 167
- Lexer.Rst (enumerator in highlight), 169
- Lexer.Rust (enumerator in highlight), 168
- Lexer.Sql (enumerator in highlight), 169
- Lexer.Swift (enumerator in highlight), 168
- Lexer.Text (enumerator in highlight), 167
- Lexer.Toml (enumerator in highlight), 168
- Lexer.Typescript (enumerator in highlight), 168
- Lexer.Typst (enumerator in highlight), 169
- Lexer.Xml (enumerator in highlight), 168
- Lexer.Yaml (enumerator in highlight), 168
- Lexer.Zig (enumerator in highlight), 168
- lexer\_alias (procedure in highlight), 183
- lexer\_info (procedure in highlight), 184
- Lexer\_Info (struct in highlight), 169
- Lexer\_Info.aliases (field in highlight), 169
- Lexer\_Info.fileNames (field in highlight), 169
- Lexer\_Info.mimetypes (field in highlight), 169
- Lexer\_Info.name (field in highlight), 169
- Lexer\_Info.spec (field in highlight), 170
- Lexer\_Info.step (field in highlight), 169
- lexer\_name (procedure in highlight), 184
- LICENSE\_NOTICE (constant in cli), 79
- Limits (struct in core), 91
- Limits.max\_depth (field in core), 91
- Limits.max\_expansion (field in core), 91
- Limits.max\_include\_depth (field in core), 91
- Limits.max\_included\_bytes (field in core), 91
- Limits.max\_nodes (field in core), 91
- Limits.max\_work (field in core), 91
- Limits.stack\_bytes (field in core), 91
- Line (struct in readers/rst), 304
  - (type in readers/commonmark), 291
  - (type in readers/rst/inline), 317
- Line.col (field in readers/rst), 305
- Line.offset (field in readers/rst), 305
- Line.raw (field in readers/rst), 305
- Line.text (field in readers/rst), 305
- line\_bounds (procedure in core), 125
- line\_column (procedure in core), 125
- line\_span (procedure in readers/rst), 313
- Line\_View (struct in core), 91
- Line\_View.offset (field in core), 92
- Line\_View.text (field in core), 91
- lines\_span (procedure in readers/rst), 313
- link\_file (procedure in host), 206
- link\_info (procedure in core), 125
- Link\_Info (struct in core), 92
- Link\_Info.destination (field in core), 92
- Link\_Info.kind (field in core), 92
- Link\_Info.name (field in core), 92
- Link\_Info.resolved (field in core), 92
- Link\_Info.title (field in core), 92
- Link\_Kind (enum in core), 92
- Link\_Kind.Email (enumerator in core), 92
- Link\_Kind.External (enumerator in core), 92
- Link\_Kind.Internal (enumerator in core), 92
- Link\_Kind.Project (enumerator in core), 92
- Link\_Kind.Unresolved (enumerator in core), 92
- list (procedure in jinja), 238
- List (struct in jinja), 230
- List allocator (field in jinja), 230
- List.fields (field in jinja), 230
- List.items (field in jinja), 230
- List.tuple (field in jinja), 230
- list\_folder (procedure in host), 207
- list\_info (procedure in core), 125
- List\_Info (struct in core), 92
- List\_Info.delimiter (field in core), 93
- List\_Info.start (field in core), 93
- List\_Info.style (field in core), 92
- list\_items (procedure in readers/commonmark), 297
- list\_packages (procedure in odindoc), 247
- List\_Style (enum in core), 93
- List\_Style.Arabic (enumerator in core), 93
- List\_Style.Bullet (enumerator in core), 93
- List\_Style.Lower\_Alpha (enumerator in core), 93
- List\_Style.Lower\_Roman (enumerator in core), 93

List\_Style.Upper\_Alpha (enumerator in core), 93  
 List\_Style.Upper\_Roman (enumerator in core), 93  
 LIST\_STYLE\_NAMES (variable in core), 134  
 load (procedure in gds), 141  
   (procedure in inventory), 225  
   (procedure in odindoc), 248  
   (procedure in project), 271  
 load\_mo (procedure in gettext), 155  
 load\_object (procedure in core), 125  
 Load\_Options (struct in project), 264  
 Load\_Options.conf\_dir (field in project), 264  
 Load\_Options.no\_config (field in project), 264  
 Load\_Options.output (field in project), 264  
 Load\_Options.overrides (field in project), 264  
 Load\_Options.source (field in project), 264  
 load\_po (procedure in gettext), 155  
 load\_template (procedure in jinja), 238  
 Loader (struct in jinja), 230  
 Loader.destroy (field in jinja), 230  
 Loader.procedure (field in jinja), 230  
 Loader.user (field in jinja), 230  
 Loader\_Proc (type in jinja), 230  
 locate (procedure in typst), 340  
 lock (procedure in gds), 141  
 Look (struct in highlight), 170  
 Look.background (field in highlight), 170  
 Look.bold (field in highlight), 170  
 Look.border (field in highlight), 170  
 Look.color (field in highlight), 170  
 Look.italic (field in highlight), 170  
 Look.underline (field in highlight), 170  
 lookup\_name (procedure in pyscan), 285  
 Loop (struct in jinja), 230  
 Loop.changed (field in jinja), 231  
 Loop.depth0 (field in jinja), 230  
 Loop.index0 (field in jinja), 230  
 Loop.items (field in jinja), 230  
 Loop.last\_seen (field in jinja), 230  
 Loop.recurse (field in jinja), 231

## M

main\_stack\_bytes (procedure in host), 207  
 make\_catalog (procedure in gettext), 155  
 make\_id (procedure in core), 125  
 make\_storage (procedure in host), 207  
 make\_template (procedure in i18n), 222  
 make\_translator (procedure in gettext), 155  
 Map\_Entry (struct in core), 93  
 Map\_Entry.offset (field in core), 93  
 Map\_Entry.span (field in core), 93  
 mark (procedure in readers/rst), 313  
 Markdown (struct in readers/sphinx), 326  
 Markdown.config (field in readers/sphinx), 327  
 Markdown.ext (field in readers/sphinx), 327  
 Markdown.host (field in readers/sphinx), 327  
 Markdown.parser (field in readers/sphinx), 327

Markdown.reader (field in readers/sphinx), 327  
 Markdown.settings (field in readers/sphinx), 327  
 Markdown.state (field in readers/sphinx), 327  
 markdown\_settings (procedure in readers/sphinx), 328  
 markup (procedure in jinja), 238  
 Markup (type in jinja), 231  
 Math\_Style (enum in core), 93  
 Math\_Style.Plain (enumerator in core), 93  
 Math\_Style.Sphinx (enumerator in core), 93  
 MAX\_OBJECT\_TYPES (constant in readers/sphinx), 329  
 MAX\_RETRIES (constant in host), 214  
 MAX\_SOURCE\_BYTES (constant in host), 214  
 Media\_Kind (enum in core), 94  
 media\_kind (procedure in core), 125  
 Media\_Kind.Audio (enumerator in core), 94  
 Media\_Kind.Image (enumerator in core), 94  
 Media\_Kind.Video (enumerator in core), 94  
 Member (struct in odindoc), 243  
 Member.docs (field in odindoc), 244  
 Member.kind (field in odindoc), 243  
 Member.name (field in odindoc), 243  
 Member.signature (field in odindoc), 243  
 Member\_Kind (enum in odindoc), 244  
 Member\_Kind.Bit\_Field (enumerator in odindoc), 244  
 Member\_Kind.Enumerator (enumerator in odindoc), 244  
 Member\_Kind.Field (enumerator in odindoc), 244  
 MEMORY\_HINT (constant in host), 214  
 memory\_loader (procedure in jinja), 238  
 merge (procedure in gettext), 155  
 Merge\_Options (struct in gettext), 146  
 Merge\_Options.fuzzy\_matching (field in gettext), 146  
 Merge\_Options.keep\_obsolete (field in gettext), 147  
 Merge\_Options.stats (field in gettext), 147  
 Merge\_Stats (struct in gettext), 147  
 Merge\_Stats.bounds (field in gettext), 147  
 Merge\_Stats.comparisons (field in gettext), 147  
 Merge\_Stats.cut\_short (field in gettext), 147  
 Merge\_Stats.searches (field in gettext), 147  
 Merge\_Stats.work (field in gettext), 147  
 Message (struct in autodoc), 76  
   (struct in core), 94  
   (struct in gettext), 147  
   (struct in odindoc), 244  
 Message.code (field in autodoc), 76  
   (field in core), 94  
   (field in odindoc), 244  
 Message.comments (field in gettext), 148  
 Message.ctx (field in gettext), 147  
 Message.flags (field in gettext), 148  
 Message.fuzzy (field in gettext), 148

- Message.has\_ctx (field in gettext), 147
  - Message.has\_plural (field in gettext), 148
  - Message.has\_previous (field in gettext), 148
  - Message.hint (field in autodoc), 76
    - (field in core), 94
    - (field in odindoc), 244
  - Message.id (field in gettext), 147
  - Message.id\_plural (field in gettext), 147
  - Message.line (field in gettext), 148
  - Message.message (field in core), 94
  - Message.notes (field in gettext), 148
  - Message.obsolete (field in gettext), 148
  - Message.previous (field in gettext), 148
  - Message.references (field in gettext), 148
  - Message.severity (field in autodoc), 76
    - (field in core), 94
    - (field in odindoc), 244
  - Message.status (field in core), 94
  - Message.strs (field in gettext), 148
  - Message.text (field in autodoc), 76
    - (field in odindoc), 244
  - Message.title (field in autodoc), 76
    - (field in core), 94
    - (field in odindoc), 244
  - Meta (struct in readers/commonmark), 291
  - Meta.content (field in readers/commonmark), 291
  - Meta.key (field in readers/commonmark), 291
  - Meter (struct in host), 195
  - Meter.backing (field in host), 195
  - Meter.ledger (field in host), 195
  - Meter.prepaid (field in host), 195
  - meter\_allocator (procedure in host), 207
  - meter\_prepay (procedure in host), 207
  - meter\_refund (procedure in host), 207
  - metered\_refusal (procedure in host), 207
  - migrate (procedure in project), 271
  - Migrate\_Options (struct in project), 264
  - Migrate\_Options.force (field in project), 264
  - Migrate\_Options.input (field in project), 264
  - Migrate\_Options.output (field in project), 264
  - Migrate\_Options.year (field in project), 265
  - Migrate\_Result (struct in project), 265
  - Migrate\_Result.attention (field in project), 265
  - Migrate\_Result.converted (field in project), 265
  - Migrate\_Result.input (field in project), 265
  - Migrate\_Result.notes (field in project), 265
  - Migrate\_Result.output (field in project), 265
  - Migrate\_Result.toml (field in project), 265
  - module (procedure in pyscan), 285
  - Module (struct in pyscan), 277
  - Module.all (field in pyscan), 278
  - Module.broken (field in pyscan), 278
  - Module.doc (field in pyscan), 278
  - Module.info (field in pyscan), 278
  - Module.kind (field in pyscan), 277
  - Module.name (field in pyscan), 277
  - Module.origin (field in pyscan), 278
  - Module.path (field in pyscan), 278
  - Module.project (field in pyscan), 277
  - Module.root (field in pyscan), 278
  - Module.scope (field in pyscan), 278
  - Module.source (field in pyscan), 278
  - Module.too\_deep (field in pyscan), 278
  - Module.tree (field in pyscan), 278
  - module\_attr (procedure in pyscan), 285
  - Module\_Kind (enum in pyscan), 278
  - Module\_Kind.Namespace (enumerator in pyscan), 279
  - Module\_Kind.Native (enumerator in pyscan), 279
  - Module\_Kind.Package (enumerator in pyscan), 278
  - Module\_Kind.Source (enumerator in pyscan), 278
  - module\_object (procedure in pyscan), 285
  - mro (procedure in pyscan), 285
  - Mro\_Entry (struct in pyscan), 279
  - Mro\_Entry.class (field in pyscan), 279
  - Mro\_Entry.path (field in pyscan), 279
  - MYST\_CONFORMANCE (constant in readers/commonmark), 299
  - myst\_options (procedure in defaults), 135
  - myst\_reader (procedure in readers/commonmark), 297
  - MYST\_VERSION (constant in readers/commonmark), 299
- ## N
- named\_child (procedure in treesitter), 335
  - named\_child\_count (procedure in treesitter), 335
  - Native (type in jinja), 231
  - new\_message (procedure in gettext), 156
  - new\_plan (procedure in gds), 141
  - next\_anchor\_id (procedure in core), 125
  - next\_attribute (procedure in core), 126
  - next\_child (procedure in core), 126
    - (procedure in treesitter), 335
  - next\_named\_sibling (procedure in treesitter), 335
  - next\_number (procedure in gds), 142
  - next\_sibling (procedure in treesitter), 335
  - next\_token (procedure in highlight), 184
  - nfd\_first (procedure in core), 126
  - no\_reader\_settings (procedure in core), 126
  - node (procedure in core), 126
  - Node (struct in core), 94
    - (struct in treesitter), 332
  - Node.attributes (field in core), 95
  - Node.ctx (field in treesitter), 332
  - Node.flags (field in core), 94
  - Node.id (field in treesitter), 332
  - Node.kind (field in core), 94
  - Node.payload (field in core), 95
  - Node.span (field in core), 95
  - Node.subtree\_end (field in core), 94
  - Node.text (field in core), 94
  - Node.tree (field in treesitter), 332
  - Node.value (field in core), 94
  - node\_count (procedure in core), 126
  - Node\_Flag (enum in core), 95
  - Node\_Flag.Anonymous (enumerator in core), 95
  - Node\_Flag.Auto (enumerator in core), 95



- Node\_Flag.Generated (enumerator in core), 95
- Node\_Flag.Header (enumerator in core), 95
- Node\_Flag.Local (enumerator in core), 95
- Node\_Flag.Referenced (enumerator in core), 95
- Node\_Flag.Symbol (enumerator in core), 95
- Node\_Flag.Tight (enumerator in core), 95
- Node\_Flags (bit set in core), 95
- Node\_Id (type in core), 95
- Node\_Kind (enum in core), 96
- Node\_Kind.Abbreviation (enumerator in core), 100
- Node\_Kind.Admonition (enumerator in core), 98
- Node\_Kind.Attribution (enumerator in core), 96
- Node\_Kind.Block\_Quote (enumerator in core), 96
- Node\_Kind.Caption (enumerator in core), 98
- Node\_Kind.Cell (enumerator in core), 98
- Node\_Kind.Citation (enumerator in core), 98
- Node\_Kind.Citation\_Reference (enumerator in core), 99
- Node\_Kind.Classifier (enumerator in core), 96
- Node\_Kind.Code (enumerator in core), 99
- Node\_Kind.Code\_Block (enumerator in core), 97
- Node\_Kind.Comment (enumerator in core), 98
- Node\_Kind.Compound (enumerator in core), 98
- Node\_Kind.Container (enumerator in core), 98
- Node\_Kind.Contents (enumerator in core), 98
- Node\_Kind.Definition (enumerator in core), 96
- Node\_Kind.Definition\_Item (enumerator in core), 96
- Node\_Kind.Definition\_List (enumerator in core), 96
- Node\_Kind.Desc (enumerator in core), 99
- Node\_Kind.Desc\_Addname (enumerator in core), 100
- Node\_Kind.Desc\_Annotation (enumerator in core), 100
- Node\_Kind.Desc\_Content (enumerator in core), 99
- Node\_Kind.Desc\_Name (enumerator in core), 100
- Node\_Kind.Desc\_Optional (enumerator in core), 100
- Node\_Kind.Desc\_Parameter (enumerator in core), 100
- Node\_Kind.Desc\_Parameterlist (enumerator in core), 100
- Node\_Kind.Desc\_Returns (enumerator in core), 100
- Node\_Kind.Desc\_Signature (enumerator in core), 99
- Node\_Kind.Directive (enumerator in core), 98
- Node\_Kind.Doctest\_Block (enumerator in core), 97
- Node\_Kind.Document (enumerator in core), 96
- Node\_Kind.Emphasis (enumerator in core), 99
- Node\_Kind.Field (enumerator in core), 96
- Node\_Kind.Field\_Body (enumerator in core), 97
- Node\_Kind.Field\_List (enumerator in core), 96
- Node\_Kind.Field\_Name (enumerator in core), 96
- Node\_Kind.Figure (enumerator in core), 98
- Node\_Kind.Footnote (enumerator in core), 98
- Node\_Kind.Footnote\_Reference (enumerator in core), 99
- Node\_Kind.Glossary (enumerator in core), 99
- Node\_Kind.Hard\_Break (enumerator in core), 100
- Node\_Kind.Heading (enumerator in core), 96
- Node\_Kind.Hlist (enumerator in core), 99
- Node\_Kind.Image (enumerator in core), 99
- Node\_Kind.Index (enumerator in core), 99
- Node\_Kind.Inline\_Target (enumerator in core), 100
- Node\_Kind.Legend (enumerator in core), 98
- Node\_Kind.Line (enumerator in core), 97
- Node\_Kind.Line\_Block (enumerator in core), 97
- Node\_Kind.Link (enumerator in core), 99
- Node\_Kind.List (enumerator in core), 96
- Node\_Kind.List\_Item (enumerator in core), 96
- Node\_Kind.Math (enumerator in core), 100
- Node\_Kind.Math\_Block (enumerator in core), 97
- Node\_Kind.Only (enumerator in core), 99
- Node\_Kind.Option (enumerator in core), 97
- Node\_Kind.Option\_Argument (enumerator in core), 97
- Node\_Kind.Option\_Description (enumerator in core), 97
- Node\_Kind.Option\_Group (enumerator in core), 97
- Node\_Kind.Option\_Item (enumerator in core), 97
- Node\_Kind.Option\_List (enumerator in core), 97
- Node\_Kind.Paragraph (enumerator in core), 96

Node\_Kind.Preformatted (enumerator in core), 97  
 Node\_Kind.Production (enumerator in core), 99  
 Node\_Kind.Production\_List (enumerator in core), 99  
 Node\_Kind.Raw\_Block (enumerator in core), 97  
 Node\_Kind.Raw\_Inline (enumerator in core), 100  
 Node\_Kind.Reference (enumerator in core), 99  
 Node\_Kind.Role (enumerator in core), 100  
 Node\_Kind.Row (enumerator in core), 97  
 Node\_Kind.Rubric (enumerator in core), 98  
 Node\_Kind.Section (enumerator in core), 96  
 Node\_Kind.Sidebar (enumerator in core), 98  
 Node\_Kind.Soft\_Break (enumerator in core), 100  
 Node\_Kind.Strong (enumerator in core), 99  
 Node\_Kind.Subscript (enumerator in core), 100  
 Node\_Kind.Substitution\_Definition (enumerator in core), 98  
 Node\_Kind.Substitution\_Reference (enumerator in core), 100  
 Node\_Kind.Subtitle (enumerator in core), 96  
 Node\_Kind.Superscript (enumerator in core), 100  
 Node\_Kind.Table (enumerator in core), 97  
 Node\_Kind.Table\_Body (enumerator in core), 97  
 Node\_Kind.Table\_Head (enumerator in core), 97  
 Node\_Kind.Target (enumerator in core), 98  
 Node\_Kind.Term (enumerator in core), 96  
 Node\_Kind.Text (enumerator in core), 99  
 Node\_Kind.Thematic\_Break (enumerator in core), 97  
 Node\_Kind.Title (enumerator in core), 96  
 Node\_Kind.Title\_Reference (enumerator in core), 100  
 Node\_Kind.Toctree (enumerator in core), 98  
 Node\_Kind.Toctree\_Entry (enumerator in core), 98  
 Node\_Kind.Topic (enumerator in core), 98  
 Node\_Kind.Version\_Modified (enumerator in core), 99  
 Node\_Kind.Xref (enumerator in core), 100  
 node\_text (procedure in core), 126  
 NONE (constant in core), 134  
 none (procedure in jinja), 239  
 None (struct in jinja), 231  
 normalize\_label (procedure in core), 126  
 normalize\_nfd\_into (procedure in core), 126  
 NOTICES (constant in treesitter), 337  
 now (procedure in gds), 142  
 number\_repr (procedure in pyscan), 285

## O

Object (struct in pyscan), 279  
 Object.abstract (field in pyscan), 280  
 Object.class (field in pyscan), 279  
 Object.class\_property (field in pyscan), 280  
 Object.decorator (field in pyscan), 280  
 Object.def (field in pyscan), 280  
 Object.doc (field in pyscan), 280  
 Object.function (field in pyscan), 279  
 Object.kind (field in pyscan), 279  
 Object.module (field in pyscan), 279  
 Object.owner (field in pyscan), 280  
 Object.partial (field in pyscan), 280  
 Object.partial\_args (field in pyscan), 280  
 Object.partial\_method (field in pyscan), 280  
 Object.path (field in pyscan), 279  
 Object.tuple (field in pyscan), 279  
 Object.value (field in pyscan), 279  
 Object.wrapper (field in pyscan), 279  
 Object\_Kind (enum in pyscan), 280  
 Object\_Kind.Builtin (enumerator in pyscan), 281  
 Object\_Kind.Class (enumerator in pyscan), 280  
 Object\_Kind.External (enumerator in pyscan), 281  
 Object\_Kind.Function (enumerator in pyscan), 280  
 Object\_Kind.Module (enumerator in pyscan), 280  
 Object\_Kind.None (enumerator in pyscan), 280  
 Object\_Kind.Value (enumerator in pyscan), 281  
 Object\_Type (struct in readers/sphinx), 327  
 Object\_Type.directive (field in readers/sphinx), 327  
 Object\_Type.display (field in readers/sphinx), 327  
 Object\_Type.first\_word (field in readers/sphinx), 327  
 Object\_Type.index (field in readers/sphinx), 327  
 Object\_Type.program (field in readers/sphinx), 328  
 Object\_Type.role (field in readers/sphinx), 327  
 Object\_Type.target (field in readers/sphinx), 327  
 odindoc (package), 240  
 offset\_in (procedure in core), 127  
 offset\_of (procedure in typst), 341  
 On\_Refusal (type in host), 195  
 open (procedure in readers/rst), 313  
 Opt\_Spec (struct in readers/rst), 305  
 Opt\_Spec.choices (field in readers/rst), 305  
 Opt\_Spec.name (field in readers/rst), 305  
 Opt\_Spec.type (field in readers/rst), 305  
 Opt\_Type (enum in readers/rst), 305  
 Opt\_Type.Char (enumerator in readers/rst), 306  
 Opt\_Type.Choice (enumerator in readers/rst), 306

- Opt\_Type.Class (enumerator in readers/rst), 306
- Opt\_Type.Encoding (enumerator in readers/rst), 306
- Opt\_Type.Figwidth (enumerator in readers/rst), 306
- Opt\_Type.Flag (enumerator in readers/rst), 305
- Opt\_Type.Int (enumerator in readers/rst), 305
- Opt\_Type.Length (enumerator in readers/rst), 306
- Opt\_Type.Length\_Percent (enumerator in readers/rst), 306
- Opt\_Type.Name (enumerator in readers/rst), 306
- Opt\_Type.Nonneg\_Int (enumerator in readers/rst), 305
- Opt\_Type.Opt\_Int (enumerator in readers/rst), 305
- Opt\_Type.Path (enumerator in readers/rst), 306
- Opt\_Type.Percentage (enumerator in readers/rst), 306
- Opt\_Type.Pos\_Int (enumerator in readers/rst), 305
- Opt\_Type.Text (enumerator in readers/rst), 305
- Opt\_Type.Text\_Required (enumerator in readers/rst), 305
- Opt\_Type.Widths (enumerator in readers/rst), 306
- opt\_value (procedure in readers/rst), 313
- Option (struct in autodoc), 77
  - (struct in odindoc), 244
  - (struct in readers/commonmark), 291
- Option.name (field in autodoc), 77
  - (field in odindoc), 244
  - (field in readers/commonmark), 291
- Option.span (field in readers/commonmark), 291
- Option.value (field in autodoc), 77
  - (field in odindoc), 244
  - (field in readers/commonmark), 291
- option\_lines (procedure in readers/rst), 313
- Options (struct in fetch), 136
  - (struct in gds), 138
  - (struct in graphdog), 163
  - (struct in jinja), 231
  - (struct in readers/rst/inline), 317
  - (struct in typst), 339
- Options.attributes (field in graphdog), 163
- Options.author (field in gds), 138
- Options.autoescape (field in jinja), 231
- Options.autoescape\_extensions (field in jinja), 231
- Options.character\_level (field in readers/rst/inline), 318
- Options.check\_only (field in gds), 139
- Options.command (field in gds), 138
- Options.default\_substitution (field in readers/rst/inline), 318
- Options.dir (field in gds), 138
- Options.discussion (field in gds), 138
- Options.docinfo (field in readers/rst/inline), 318
- Options.dry\_run (field in gds), 139
- Options.embedded\_fonts\_only (field in typst), 339
- Options.engine (field in graphdog), 163
- Options.evidence (field in gds), 138
- Options.files (field in graphdog), 163
- Options.font\_paths (field in typst), 339
- Options.format (field in graphdog), 163
- Options.historical (field in gds), 139
- Options.identifier (field in gds), 138
- Options.keep\_trailing\_newline (field in jinja), 231
- Options.label (field in gds), 138
- Options.labels (field in gds), 138
- Options.language (field in readers/rst/inline), 318
- Options.loader (field in jinja), 231
- Options.lstrip\_blocks (field in jinja), 231
- Options.max\_bytes (field in fetch), 136
- Options.max\_iterations (field in jinja), 232
- Options.max\_nesting (field in jinja), 232
- Options.max\_output (field in jinja), 232
- Options.max\_recursion (field in jinja), 231
- Options.number (field in gds), 139
- Options.offline (field in typst), 339
- Options.package\_cache (field in typst), 339
- Options.pep\_base\_url (field in readers/rst/inline), 318
- Options.pep\_references (field in readers/rst/inline), 317
- Options.random\_seed (field in jinja), 232
- Options.reason (field in gds), 138
- Options.record (field in gds), 138
- Options.render (field in gds), 139
- Options.reserve (field in graphdog), 163
  - (field in typst), 339
- Options.rfc\_base\_url (field in readers/rst/inline), 318
- Options.rfc\_references (field in readers/rst/inline), 318
- Options.rollback (field in gds), 139
- Options.sectsubtitle (field in readers/rst/inline), 318
- Options.slug (field in gds), 138
- Options.smart\_quotes (field in readers/rst/inline), 318
- Options.smart\_quotes\_locales (field in readers/rst/inline), 318
- Options.stack\_bytes (field in jinja), 232
- Options.state (field in gds), 138
- Options.tab\_width (field in readers/rst/inline), 318
- Options.target (field in gds), 139
- Options.timeout\_ms (field in fetch), 136
- Options.to (field in gds), 138
- Options.trim\_blocks (field in jinja), 231
- Options.trim\_footnote\_reference\_space (field in readers/rst/inline), 318
- Options.undefined (field in jinja), 231
- Options.user\_agent (field in fetch), 136
- Options.warnings (field in graphdog), 163
- OS (type in odindoc), 244
- out\_of\_memory (procedure in host), 208
- Outcome (struct in typst), 339
- Outcome.detail (field in typst), 339

Outcome.diagnostics (field in typst), 339  
 Outcome.exit\_code (field in typst), 339  
 Outcome.launch\_error (field in typst), 339  
 Outcome.ok (field in typst), 339  
 Output (struct in highlight), 170  
 Output.buf (field in highlight), 170  
 Output.written (field in highlight), 170  
 output\_fits (procedure in highlight), 184  
 output\_string (procedure in highlight), 184  
 output\_text (procedure in host), 208

## P

Package (struct in odindoc), 244  
 Package.decls (field in odindoc), 245  
 Package.dir (field in odindoc), 245  
 Package.doc (field in odindoc), 245  
 Package.errors (field in odindoc), 245  
 Package.files (field in odindoc), 245  
 Package.id (field in odindoc), 245  
 Package.imports (field in odindoc), 245  
 Package.name (field in odindoc), 245  
 Package.platform (field in odindoc), 245  
 Package.synopsis (field in odindoc), 245  
 Package.targets (field in odindoc), 245  
 Package.too\_deep (field in odindoc), 245  
 Package.too\_large (field in odindoc), 245  
 Package\_Ref (struct in odindoc), 245  
 Package\_Ref.dir (field in odindoc), 245  
 Package\_Ref.id (field in odindoc), 245  
 Param (struct in pyscan), 281  
 Param.annotation (field in pyscan), 281  
 Param.default (field in pyscan), 281  
 Param.kind (field in pyscan), 281  
 Param.name (field in pyscan), 281  
 Param.type\_comment (field in pyscan), 281  
 Param\_Kind (enum in pyscan), 281  
 Param\_Kind.Keyword\_Only (enumerator in pyscan), 281  
 Param\_Kind.Positional\_Only (enumerator in pyscan), 281  
 Param\_Kind.Positional\_Or\_Keyword (enumerator in pyscan), 281  
 Param\_Kind.Var\_Keyword (enumerator in pyscan), 281  
 Param\_Kind.Var\_Positional (enumerator in pyscan), 281  
 parent (procedure in treesitter), 335  
 parent\_of (procedure in core), 127  
 parse (procedure in treesitter), 335  
 PARSE\_BASE\_BYTES (constant in odindoc), 248  
 parse\_blocks (procedure in readers/rst), 314  
 parse\_body (procedure in readers/commonmark), 298  
   (procedure in readers/rst), 314  
 parse\_bound (procedure in odindoc), 248  
 PARSE\_BYTES\_PER\_BYTE (constant in odindoc), 248  
 parse\_engine (procedure in graphdog), 164  
 parse\_expression (procedure in pyscan), 286  
 parse\_format (procedure in graphdog), 164  
 parse\_inline (procedure in readers/commonmark), 298  
   (procedure in readers/rst/inline), 321  
 parse\_mo (procedure in gettext), 156

parse\_plural\_forms (procedure in gettext), 156  
 parse\_po (procedure in gettext), 156  
 parse\_record (procedure in gds), 142  
 parse\_source (procedure in readers/rst), 314  
 parse\_target (procedure in i18n), 222  
 parse\_template (procedure in jinja), 239  
 parse\_titled (procedure in readers/rst), 314  
 Parser (struct in readers/rst), 306  
   (struct in treesitter), 332  
 Parser.b (field in readers/rst), 306  
 Parser.ctx (field in readers/rst), 306  
 Parser.depth (field in readers/rst), 306  
 Parser.ext (field in readers/rst), 307  
 Parser.has\_pending (field in readers/rst), 307  
 Parser.host (field in readers/rst), 308  
 Parser.hosted (field in readers/rst), 308  
 Parser.in\_sidebar (field in readers/rst), 307  
 Parser.inl (field in readers/rst), 306  
 Parser.level\_base (field in readers/rst), 307  
 Parser.pending (field in readers/rst), 307  
 Parser.pending\_span (field in readers/rst), 307  
 Parser.section\_base (field in readers/rst), 307  
 Parser.sections (field in readers/rst), 307  
 Parser.sections\_used (field in readers/rst), 307  
 Parser.src (field in readers/rst), 306  
 Parser.stack (field in readers/rst), 307  
 Parser.styles (field in readers/rst), 307  
 Parser.styles\_used (field in readers/rst), 307  
 Parser.tab (field in readers/rst), 306  
 Parser.table\_extra (field in readers/rst), 308  
 Parser.text (field in readers/rst), 306  
 Parser.top\_count (field in readers/rst), 307  
 Parser.top\_transition (field in readers/rst), 307  
 Parser.transition\_span (field in readers/rst), 307  
 parser\_delete (procedure in treesitter), 336  
 parser\_new (procedure in treesitter), 336  
 Pass (struct in core), 101  
 Pass.name (field in core), 101  
 Pass.procedure (field in core), 101  
 Pass.user (field in core), 101  
 Pass\_Outcome (enum in core), 101  
 Pass\_Outcome.Failed (enumerator in core), 101  
 Pass\_Outcome.Rebuilt (enumerator in core), 101  
 Pass\_Outcome.Unchanged (enumerator in core), 101  
 PATH\_BYTES (constant in host), 214  
 path\_of (procedure in readers/rst), 314  
 path\_type (procedure in host), 208  
 Pdf\_Document (struct in project), 265  
 Pdf\_Document.author (field in project), 265  
 Pdf\_Document.file (field in project), 265  
 Pdf\_Document.root (field in project), 265  
 Pdf\_Document.title (field in project), 265  
 pinned (procedure in typst), 341  
 PINNED\_VERSION (constant in typst), 341



- place\_titled (procedure in readers/rst), 314
- plain\_text (procedure in core), 127
- Plan (struct in gds), 139
- Plan.changes (field in gds), 139
- Plan.summary (field in gds), 139
- plan\_bytes (procedure in core), 127
- plan\_session (procedure in host), 208
- plural (procedure group in gettext), 158
- plural\_forms\_for (procedure in gettext), 156
- plural\_index (procedure in gettext), 156
- Plural\_Node (struct in gettext), 148
- Plural\_Node.a (field in gettext), 148
- Plural\_Node.b (field in gettext), 148
- Plural\_Node.c (field in gettext), 148
- Plural\_Node.op (field in gettext), 148
- Plural\_Node.value (field in gettext), 149
- Plural\_Op (enum in gettext), 149
- Plural\_Op.Add (enumerator in gettext), 149
- Plural\_Op.And (enumerator in gettext), 149
- Plural\_Op.Choose (enumerator in gettext), 149
- Plural\_Op.Div (enumerator in gettext), 149
- Plural\_Op.Equal (enumerator in gettext), 149
- Plural\_Op.Greater (enumerator in gettext), 149
- Plural\_Op.Greater\_Equal (enumerator in gettext), 149
- Plural\_Op.Less (enumerator in gettext), 149
- Plural\_Op.Less\_Equal (enumerator in gettext), 149
- Plural\_Op.Mod (enumerator in gettext), 149
- Plural\_Op.Mul (enumerator in gettext), 149
- Plural\_Op.N (enumerator in gettext), 149
- Plural\_Op.Not (enumerator in gettext), 149
- Plural\_Op.Not\_Equal (enumerator in gettext), 149
- Plural\_Op.Number (enumerator in gettext), 149
- Plural\_Op.Or (enumerator in gettext), 149
- Plural\_Op.Sub (enumerator in gettext), 149
- Plural\_Problem (struct in gettext), 150
- Plural\_Problem.column (field in gettext), 150
- Plural\_Problem.message (field in gettext), 150
- Plural\_Rule (struct in gettext), 150
- Plural\_Rule.count (field in gettext), 150
- Plural\_Rule.nodes (field in gettext), 150
- Plural\_Rule.nplurals (field in gettext), 150
- Plural\_Rule.root (field in gettext), 150
- plural\_with\_context (procedure group in gettext), 158
- Point (struct in treesitter), 332
- Point.column (field in treesitter), 332
- Point.row (field in treesitter), 332
- Policy (struct in core), 101
- Policy.includes (field in core), 101
- Policy.keep\_documents\_with\_errors (field in core), 101
- Policy.math (field in core), 101
- Policy.raw (field in core), 101
- Policy.strict (field in core), 101
- Policy.unsafe\_urls (field in core), 101
- pool\_classes (procedure in readers/rst), 314
- pool\_lines (procedure in readers/rst), 314
- pool\_lower (procedure in readers/rst), 315
- pool\_of (procedure in gettext), 157
- pool\_string (procedure in core), 127
- pop (procedure in core), 127
- Pot\_Header (struct in i18n), 216
- Pot\_Header.copyright (field in i18n), 216
- Pot\_Header.creation\_date (field in i18n), 217
- Pot\_Header.language\_team (field in i18n), 216
- Pot\_Header.last\_translator (field in i18n), 216
- Pot\_Header.project (field in i18n), 216
- Pot\_Header.version (field in i18n), 216
- prepare\_docstring (procedure in pycan), 286
- prepare\_session (procedure in host), 208
- prev\_named\_sibling (procedure in treesitter), 336
- prev\_sibling (procedure in treesitter), 336
- Previous (struct in gettext), 150
- Previous.ctx (field in gettext), 150
- Previous.has\_ctx (field in gettext), 150
- Previous.has\_plural (field in gettext), 150
- Previous.id (field in gettext), 150
- Previous.id\_plural (field in gettext), 150
- Private (enum in odindoc), 246
- Private.File (enumerator in odindoc), 246
- Private.Package (enumerator in odindoc), 246
- Private.Public (enumerator in odindoc), 246
- problem (procedure in gds), 142
- Problem (struct in gds), 139
- (struct in highlight), 170
- (struct in host), 195
- Problem.code (field in gds), 139
- (field in host), 195
- Problem.column (field in highlight), 171
- Problem.excerpt (field in highlight), 171
- Problem.exit (field in gds), 139
- (field in host), 196
- Problem.hint (field in gds), 139
- (field in host), 195
- Problem.kind (field in highlight), 170
- Problem.lexer (field in highlight), 170
- Problem.line (field in highlight), 171
- Problem.message (field in gds), 139
- (field in host), 195
- Problem.name (field in highlight), 170
- Problem.near (field in highlight), 171
- Problem.path (field in gds), 139
- (field in host), 195
- Problem.suggestion (field in highlight), 170
- Problem.title (field in gds), 139
- (field in host), 195
- Problem\_Kind (enum in highlight), 171
- Problem\_Kind.Lexing\_Error (enumerator in highlight), 171
- Problem\_Kind.None (enumerator in highlight), 171

Problem\_Kind.Unknown\_Language  
     (enumerator in highlight), 171  
 Problem\_Kind.Unknown\_Style  
     (enumerator in highlight), 171  
 Progress (enum in i18n), 217  
 Progress.Translated (enumerator in i18n),  
     217  
 Progress.Untranslated (enumerator in  
     i18n), 217  
 Progress\_Classes (bit set in i18n), 217  
 project (package), 248  
 Project (struct in odindoc), 246  
     (struct in project), 266  
     (struct in pyscan), 281  
 Project.config (field in project), 266  
 Project.files (field in odindoc), 246  
     (field in pyscan), 282  
 Project.memory (field in project), 266  
 Project.modules (field in pyscan), 282  
 Project.mro\_depth (field in pyscan), 282  
 Project.owner (field in project), 266  
 Project.packages (field in odindoc), 246  
 Project.parser (field in pyscan), 282  
 Project.reader (field in odindoc), 246  
     (field in pyscan), 282  
 Project.registry (field in project), 266  
 Project.resolving (field in pyscan), 282  
 Project.roots (field in odindoc), 246  
     (field in pyscan), 282  
 Project.settings (field in project), 266  
 Project.stack\_base (field in pyscan), 282  
 Project.steps (field in pyscan), 282  
 Project.trees (field in pyscan), 282  
 Project.warnings (field in project), 266  
 publish (procedure in host), 208  
 push (procedure in core), 127  
 push\_lines (procedure in readers/rst), 315  
 put (procedure in highlight), 184  
 put\_byte (procedure in highlight), 184  
 put\_html (procedure in highlight), 184  
 pyscan (package), 272  
 python (procedure in treesitter), 336  
 python\_space (procedure in readers/rst/  
     inline), 321  
 PYTHON\_VERSION (constant in  
     treesitter), 337

## Q

quickstart (procedure in project), 271  
 Quickstart\_Options (struct in project), 267  
 Quickstart\_Options.author (field in  
     project), 267  
 Quickstart\_Options.date (field in project),  
     267  
 Quickstart\_Options.dir (field in project),  
     267  
 Quickstart\_Options.language (field in  
     project), 267  
 Quickstart\_Options.project (field in  
     project), 267  
 Quickstart\_Options.release (field in  
     project), 267  
 Quickstart\_Options.root\_doc (field in  
     project), 267  
 Quickstart\_Options.separate (field in  
     project), 267

Quickstart\_Options.suffix (field in project),  
     267  
 Quickstart\_Options.version (field in  
     project), 267  
 Quickstart\_Options.year (field in project),  
     267

## R

read (procedure in host), 208  
     (procedure in readers/commonmark),  
         298  
     (procedure in readers/rst), 315  
     (procedure in readers/sphinx), 328  
 Read\_Context (struct in core), 102  
 Read\_Context.b (field in core), 102  
 Read\_Context.included (field in core), 102  
 Read\_Context.limits (field in core), 102  
 Read\_Context.options (field in core), 102  
 Read\_Context.policy (field in core), 102  
 Read\_Context.reports (field in core), 102  
 Read\_Context.resources (field in core), 102  
 Read\_Context.scratch (field in core), 102  
 Read\_Context.source (field in core), 102  
 Read\_Context.ws (field in core), 102  
 read\_document (procedure in core), 127  
 read\_extension (procedure in core), 127  
 Read\_Extension (struct in core), 102  
 Read\_Extension.data (field in core), 102  
 Read\_Extension.id (field in core), 103  
 read\_file (procedure in host), 209  
 read\_inline (procedure in readers/  
     commonmark), 298  
     (procedure in readers/rst), 315  
     (procedure in readers/sphinx), 328  
 Read\_Inline\_Proc (type in core), 103  
 read\_kept (procedure in host), 209  
 read\_myst (procedure in readers/  
     commonmark), 298  
 read\_myst\_inline (procedure in readers/  
     commonmark), 298  
 Read\_Options (struct in core), 103  
 Read\_Options.extension (field in core), 103  
 Read\_Options.promote\_title (field in core),  
     103  
 Read\_Options.tab\_width (field in core),  
     103  
 Read\_Options.today (field in core), 103  
 read\_owned (procedure in host), 209  
 Read\_Proc (type in core), 103  
 Read\_Result (struct in core), 103  
 Read\_Result.document (field in core), 103  
 Read\_Result.using (field in core), 103  
 read\_source (procedure in host), 209  
 reader (procedure in readers/  
     commonmark), 298  
     (procedure in readers/rst), 315  
     (procedure in readers/sphinx), 329  
 Reader (struct in core), 103  
 Reader.conformance (field in core), 104  
 Reader.id (field in core), 103  
 Reader.name (field in core), 103  
 Reader.read (field in core), 104  
 Reader.read\_inline (field in core), 104  
 Reader.version (field in core), 103  
 reader\_settings (procedure in core), 128  
 readers/commonmark (package), 287  
 readers/rst (package), 299

- readers/rst/inline (package), 317
- readers/sphinx (package), 322
- Record (struct in gds), 139
- Record.authors (field in gds), 140
- Record.created (field in gds), 140
- Record.discussion (field in gds), 140
- Record.fields (field in gds), 140
- Record.historical (field in gds), 140
- Record.kind (field in gds), 140
- Record.labels (field in gds), 140
- Record.number (field in gds), 139
- Record.path (field in gds), 140
- Record.slug (field in gds), 139
- Record.state (field in gds), 140
- Record.text (field in gds), 140
- Record.title (field in gds), 140
- Record.updated (field in gds), 140
- recover (procedure in gds), 142
- reference (procedure in pyscan), 286
- Reference\_Style (enum in gettext), 150
- Reference\_Style.None (enumerator in gettext), 151
- Reference\_Style.One\_Per\_Line (enumerator in gettext), 151
- Reference\_Style.Wrapped (enumerator in gettext), 150
- Refusal (struct in host), 196
- Refusal.disk (field in host), 196
- Refusal.limit (field in host), 196
- Refusal.need (field in host), 196
- Refusal.place (field in host), 196
- Refusal.used (field in host), 196
- Refusal\_Choice (enum in host), 196
- Refusal\_Choice.Disk (enumerator in host), 196
- Refusal\_Choice.Raise (enumerator in host), 196
- Refusal\_Choice.Stop (enumerator in host), 196
- Refusal\_Decision (struct in host), 196
- Refusal\_Decision.choice (field in host), 196
- Refusal\_Decision.limit (field in host), 196
- Region (enum in core), 104
- Region.Attributes (enumerator in core), 104
- Region.Cells (enumerator in core), 104
- Region.Links (enumerator in core), 104
- Region.Lists (enumerator in core), 104
- Region.Nodes (enumerator in core), 104
- Region.None (enumerator in core), 104
- Region.Output (enumerator in core), 104
- Region.Reports (enumerator in core), 104
- Region.Scratch (enumerator in core), 104
- Region.Second\_Bank (enumerator in core), 104
- Region.Source\_Map (enumerator in core), 104
- Region.Sources (enumerator in core), 104
- Region.Text (enumerator in core), 104
- registry (procedure in defaults), 135
- Registry (struct in core), 104
- Registry.readers (field in core), 104
- Registry.renderers (field in core), 105
- release (procedure in project), 271
- (procedure in readers/rst), 315
- remove\_path (procedure in host), 209
- remove\_stages (procedure in host), 209
- rename\_path (procedure in host), 210
- render (procedure in graphdog), 164
- (procedure in jinja), 239
- (procedure in render/html), 329
- (procedure in render/text), 330
- (procedure in render/typst), 331
- render/html (package), 329
- render/text (package), 330
- render/typst (package), 330
- Render\_Context (struct in core), 105
- Render\_Context.doc (field in core), 105
- Render\_Context.limits (field in core), 105
- Render\_Context.options (field in core), 105
- Render\_Context.out (field in core), 105
- Render\_Context.policy (field in core), 105
- Render\_Context.reports (field in core), 105
- Render\_Context.scratch (field in core), 105
- Render\_Context.view (field in core), 105
- render\_document (procedure in core), 128
- render\_json (procedure in host), 210
- Render\_Options (struct in core), 105
- Render\_Options.anchor\_prefix (field in core), 106
- Render\_Options.highlighter (field in core), 106
- Render\_Options.labels (field in core), 106
- Render\_Options.language (field in core), 105
- Render\_Options.math (field in core), 106
- Render\_Options.math\_display (field in core), 106
- Render\_Options.math\_inline (field in core), 106
- Render\_Options.permalink\_icon (field in core), 106
- Render\_Options.permalinks (field in core), 106
- Render\_Options.reference\_profile (field in core), 105
- Render\_Options.secnumber\_suffix (field in core), 106
- Render\_Options.section\_numbers (field in core), 105
- Render\_Options.standalone (field in core), 105
- Render\_Options.stylesheet (field in core), 105
- Render\_Options.title (field in core), 105
- render\_position (procedure in jinja), 239
- render\_problem\_json (procedure in host), 210
- render\_problem\_text (procedure in host), 210
- Render\_Proc (type in core), 106
- render\_source (procedure in jinja), 239
- render\_text (procedure in host), 210
- renderer (procedure in render/html), 329
- (procedure in render/text), 330
- (procedure in render/typst), 331
- Renderer (struct in core), 106
- Renderer.conformance (field in core), 107
- Renderer.id (field in core), 106
- Renderer.media\_type (field in core), 106
- Renderer.name (field in core), 106
- Renderer.render (field in core), 107
- Renderer.version (field in core), 106
- replace\_field (procedure in gds), 142

- report (procedure in core), 128
  - (procedure in readers/rst), 315
- Report (struct in host), 197
- Report.category (field in host), 197
- Report.code (field in host), 197
- Report.column (field in host), 197
- Report.end\_column (field in host), 197
- Report.end\_line (field in host), 197
- Report.excerpt (field in host), 197
- Report.hint (field in host), 197
- Report.line (field in host), 197
- Report.marker (field in host), 197
- Report.message (field in host), 197
- Report.path (field in host), 197
- Report.schema (field in host), 197
- Report.severity (field in host), 197
- Report.source (field in host), 197
- Report.title (field in host), 197
- report\_message (procedure in core), 128
- Reports (struct in core), 107
- Reports.errors (field in core), 107
- Reports.fatal (field in core), 107
- Reports.has\_primary (field in core), 107
- Reports.items (field in core), 107
- Reports.muted (field in core), 107
- Reports.primary (field in core), 107
- Reports.suppressed (field in core), 107
- Reports.truncated (field in core), 107
- Reports.used (field in core), 107
- Reports.warnings (field in core), 107
- repr (procedure in jinja), 239
- repr\_bytes (procedure in pyscan), 286
- repr\_str (procedure in pyscan), 286
- Request (struct in autodoc), 77
  - (struct in core), 107
  - (struct in odindoc), 246
- Request.argument (field in autodoc), 77
  - (field in odindoc), 246
- Request.cancel (field in core), 108
- Request.class (field in autodoc), 77
- Request.content (field in autodoc), 77
- Request.current (field in odindoc), 246
- Request.directive (field in autodoc), 77
  - (field in odindoc), 246
- Request.limits (field in core), 108
- Request.module (field in autodoc), 77
- Request.options (field in autodoc), 77
  - (field in odindoc), 246
- Request.passes (field in core), 108
- Request.policy (field in core), 108
- Request.read (field in core), 107
- Request.reader (field in core), 107
- Request.render (field in core), 107
- Request.renderer (field in core), 107
- Request.resources (field in core), 108
- Request.source (field in core), 107
- Request.source\_map (field in core), 108
- Reserve (struct in graphdog), 163
  - (struct in typst), 340
- Reserve.procedure (field in graphdog), 163
  - (field in typst), 340
- Reserve.user (field in graphdog), 163
  - (field in typst), 340
- reset\_workspace (procedure in core), 128
- resize\_output (procedure in host), 210
- resize\_source\_map (procedure in host), 210
- resolve (procedure in gds), 142
  - (procedure in highlight), 184
  - (procedure in readers/rst/inline), 322
- resolve\_link (procedure in core), 128
- resolve\_style (procedure in highlight), 184
- Resource\_Provider (struct in core), 108
- Resource\_Provider.procedure (field in core), 108
- Resource\_Provider.user (field in core), 108
- Resource\_Status (enum in core), 108
- Resource\_Status.Limit (enumerator in core), 108
- Resource\_Status.Missing (enumerator in core), 108
- Resource\_Status.Ok (enumerator in core), 108
- Resource\_Status.Refused (enumerator in core), 108
- Result (struct in autodoc), 77
  - (struct in core), 108
  - (struct in fetch), 136
  - (struct in odindoc), 246
- Result.data (field in fetch), 136
- Result.error (field in fetch), 136
- Result.lines (field in autodoc), 77
  - (field in odindoc), 246
- Result.mapped (field in core), 109
- Result.message (field in fetch), 136
- Result.messages (field in autodoc), 77
  - (field in odindoc), 246
- Result.need (field in core), 109
- Result.primary (field in core), 109
- Result.reports (field in core), 109
- Result.status (field in core), 109
  - (field in fetch), 136
- Result.suppressed (field in core), 109
- Result.truncated (field in core), 109
- Result.url (field in fetch), 136
- Result.written (field in core), 109
- Role (struct in readers/commonmark), 291
  - (struct in readers/rst/inline), 318
- Role.base (field in readers/rst/inline), 318
- Role.classes (field in readers/rst/inline), 318
- Role.content (field in readers/commonmark), 292
- Role.format (field in readers/rst/inline), 318
- Role.language (field in readers/rst/inline), 319
- Role.lines (field in readers/commonmark), 292
- Role.name (field in readers/commonmark), 292
  - (field in readers/rst/inline), 318
- Role.source (field in readers/commonmark), 292
- Role.span (field in readers/commonmark), 292
- Role\_Base (enum in readers/rst/inline), 319
- Role\_Base.Abbreviation (enumerator in readers/rst/inline), 319
- Role\_Base.Acronym (enumerator in readers/rst/inline), 319
- Role\_Base.Code (enumerator in readers/rst/inline), 319
- Role\_Base.Custom (enumerator in readers/rst/inline), 319



- Role\_Base.Emphasis (enumerator in readers/rst/inline), 319
- Role\_Base.Literal (enumerator in readers/rst/inline), 319
- Role\_Base.Math (enumerator in readers/rst/inline), 319
- Role\_Base.Pep\_Reference (enumerator in readers/rst/inline), 319
- Role\_Base.Raw (enumerator in readers/rst/inline), 319
- Role\_Base.Rfc\_Reference (enumerator in readers/rst/inline), 319
- Role\_Base.Strong (enumerator in readers/rst/inline), 319
- Role\_Base.Subscript (enumerator in readers/rst/inline), 319
- Role\_Base.Superscript (enumerator in readers/rst/inline), 319
- Role\_Base.Title\_Reference (enumerator in readers/rst/inline), 319
- Role\_Call (struct in readers/rst/inline), 319
- Role\_Call.b (field in readers/rst/inline), 320
- Role\_Call.content (field in readers/rst/inline), 320
- Role\_Call.marked (field in readers/rst/inline), 320
- Role\_Call.name (field in readers/rst/inline), 320
- Role\_Call.span (field in readers/rst/inline), 320
- Role\_Call.state (field in readers/rst/inline), 319
- Role\_Call.text (field in readers/rst/inline), 320
- Role\_Handler (struct in readers/commonmark), 292
  - (struct in readers/rst), 308
- Role\_Handler.name (field in readers/commonmark), 292
  - (field in readers/rst), 308
- Role\_Handler.run (field in readers/commonmark), 292
  - (field in readers/rst), 308
- Role\_Hook (struct in readers/rst/inline), 320
- Role\_Hook.procedure (field in readers/rst/inline), 320
- Role\_Hook.user (field in readers/rst/inline), 320
- Role\_Proc (type in readers/commonmark), 292
  - (type in readers/rst), 308
- root (procedure in treesitter), 336
- Root (struct in odindoc), 247
- ROOT (type in core), 102
- Root.collection (field in odindoc), 247
- Root.dir (field in odindoc), 247
- rst\_options (procedure in defaults), 135
- Rule (enum in readers/commonmark), 292
- Rule.Autolink (enumerator in readers/commonmark), 293
- Rule.Backticks (enumerator in readers/commonmark), 293
- Rule.Blockquote (enumerator in readers/commonmark), 292
- Rule.Code (enumerator in readers/commonmark), 292
- Rule.Emphasis (enumerator in readers/commonmark), 293
- Rule.Entity (enumerator in readers/commonmark), 293
- Rule.Escape (enumerator in readers/commonmark), 293
- Rule.Fence (enumerator in readers/commonmark), 292
- Rule.Footnote\_Def (enumerator in readers/commonmark), 294
- Rule.Footnote\_Ref (enumerator in readers/commonmark), 294
- Rule.Front\_Matter (enumerator in readers/commonmark), 293
- Rule.Heading (enumerator in readers/commonmark), 293
- Rule.Hr (enumerator in readers/commonmark), 292
- Rule.Html\_Block (enumerator in readers/commonmark), 293
- Rule.Html\_Inline (enumerator in readers/commonmark), 293
- Rule.Image (enumerator in readers/commonmark), 293
- Rule.Lheading (enumerator in readers/commonmark), 293
- Rule.Link (enumerator in readers/commonmark), 293
- Rule.List (enumerator in readers/commonmark), 292
- Rule.Math\_Block (enumerator in readers/commonmark), 293
- Rule.Math\_Inline (enumerator in readers/commonmark), 293
- Rule.Myst\_Block\_Break (enumerator in readers/commonmark), 293
- Rule.Myst\_Directive (enumerator in readers/commonmark), 294
- Rule.Myst\_Line\_Comment (enumerator in readers/commonmark), 293
- Rule.Myst\_Role (enumerator in readers/commonmark), 293
- Rule.Myst\_Target (enumerator in readers/commonmark), 293
- Rule.Newline (enumerator in readers/commonmark), 293
- Rule.Reference (enumerator in readers/commonmark), 292
- Rule.Table (enumerator in readers/commonmark), 292
- run (procedure in cli), 79
  - (procedure in host), 210
- run\_hosted (procedure in readers/rst), 316
- run\_workers (procedure in host), 211
- RUNTIME\_VERSION (constant in treesitter), 337

## S

- same\_contents (procedure in host), 211
- save\_po (procedure in gettext), 157
- Scanner (struct in highlight), 171
- Scanner.code (field in highlight), 171
- Scanner.count (field in highlight), 172
- Scanner.head (field in highlight), 172
- Scanner.inner (field in highlight), 172
- Scanner.limit (field in highlight), 171
- Scanner.nested (field in highlight), 172
- Scanner.pos (field in highlight), 171
- Scanner.queue (field in highlight), 172
- Scanner.region (field in highlight), 172
- Scanner.region\_start (field in highlight), 172
- Scanner.state (field in highlight), 171
- scanner\_init (procedure in highlight), 185
- SCHEMA\_VERSION (constant in core), 134
- Scope (struct in pyscan), 282
- Scope.defs (field in pyscan), 282
- Scratch (struct in core), 109
- Scratch.bytes (field in core), 109
- Scratch.high (field in core), 109
- Scratch.used (field in core), 109
- scratch\_buffer (procedure in core), 128
- scratch\_mark (procedure in core), 128
- Scratch\_Mark (type in core), 109
- scratch\_release (procedure in core), 129
- scratch\_take (procedure in core), 129
- serve (procedure in project), 271
- Server (struct in project), 267
- Server.project (field in project), 267
- Server.rebuild (field in project), 268
- Server.stamp (field in project), 268
- Server.user (field in project), 268
- Session (struct in autodoc), 77
- (struct in host), 198
- Session.allocated (field in host), 199
- Session allocator (field in host), 198
- Session.denied (field in host), 199
- Session.disk (field in host), 199
- Session.held (field in host), 199
- Session.inputs (field in host), 199
- Session.kept (field in host), 199
- Session.limit (field in host), 198
- Session.map\_size (field in host), 198
- Session.merged (field in autodoc), 78
- Session.output (field in host), 198
- Session.own (field in host), 198
- Session.project (field in autodoc), 78
- Session.ready (field in host), 198
- Session.refused (field in host), 199
- Session.result (field in host), 198
- Session.retries (field in host), 198
- Session.shared (field in host), 198
- Session.sizes (field in host), 198
- Session.source\_map (field in host), 198
- Session.storage (field in host), 198
- Session.ws (field in host), 198
- session\_destroy (procedure in host), 211
- session\_init (procedure in autodoc), 78
- set (procedure in jinja), 239
- set\_cell (procedure in core), 129
- set\_header (procedure in i18n), 222
- set\_header\_field (procedure in gettext), 157
- set\_key (procedure in jinja), 239
- set\_kind (procedure in core), 129
- set\_link (procedure in core), 129
- set\_list (procedure in core), 129
- set\_span\_end (procedure in core), 129
- set\_text (procedure in core), 129
- set\_value (procedure in core), 129
- Settings (struct in readers/commonmark), 294
- (struct in readers/rst), 308
- Settings.character\_level\_inline\_markup (field in readers/rst), 309
- Settings.commonmark\_only (field in readers/commonmark), 294
- Settings.disable\_syntax (field in readers/commonmark), 294
- Settings.dmath\_allow\_digits (field in readers/commonmark), 295
- Settings.dmath\_allow\_labels (field in readers/commonmark), 295
- Settings.dmath\_allow\_space (field in readers/commonmark), 295
- Settings.dmath\_double\_inline (field in readers/commonmark), 295
- Settings.docinfo (field in readers/rst), 309
- Settings.docname (field in readers/commonmark), 296
- Settings.enable\_extensions (field in readers/commonmark), 294
- Settings.env\_config (field in readers/commonmark), 296
- Settings.extension (field in readers/commonmark), 294
- (field in readers/rst), 308
- Settings.footnote\_transition (field in readers/commonmark), 295
- Settings.heading\_anchors (field in readers/commonmark), 295
- Settings.heading\_slug\_func (field in readers/commonmark), 295
- Settings.host (field in readers/rst), 308
- Settings.html\_meta (field in readers/commonmark), 295
- Settings.language (field in readers/rst), 309
- Settings.linkify\_fuzzy\_links (field in readers/commonmark), 294
- Settings.pep\_base\_url (field in readers/rst), 309
- Settings.pep\_references (field in readers/rst), 308
- Settings.rfc\_base\_url (field in readers/rst), 309
- Settings.rfc\_references (field in readers/rst), 309
- Settings.sectsubtitle (field in readers/rst), 309
- Settings.smart\_quotes (field in readers/rst), 309
- Settings.smart\_quotes\_locales (field in readers/rst), 309
- Settings.sub\_delimiters (field in readers/commonmark), 295
- Settings.substitutions (field in readers/commonmark), 295

- Settings.tab\_width (field in readers/rst), 309
- Settings.today (field in readers/commonmark), 296
- Settings.trim\_footnote\_reference\_space (field in readers/rst), 309
- Settings.url\_schemes (field in readers/commonmark), 294
- Settings.words\_per\_minute (field in readers/commonmark), 296
- settle (procedure in host), 211
- Severity (enum in autodoc), 78
  - (enum in core), 109
  - (enum in odindoc), 247
- Severity.Error (enumerator in autodoc), 78
  - (enumerator in core), 109
  - (enumerator in odindoc), 247
- Severity.Note (enumerator in autodoc), 78
  - (enumerator in core), 109
  - (enumerator in odindoc), 247
- Severity.Warning (enumerator in autodoc), 78
  - (enumerator in core), 109
  - (enumerator in odindoc), 247
- similarity (procedure in gettext), 157
- Sizes (struct in host), 199
- Sizes.attributes (field in host), 199
- Sizes.cells (field in host), 199
- Sizes.links (field in host), 199
- Sizes.lists (field in host), 199
- Sizes.nodes (field in host), 199
- Sizes.output (field in host), 200
- Sizes.reports (field in host), 200
- Sizes.scratch (field in host), 199
- Sizes.second\_bank (field in host), 200
- Sizes.sources (field in host), 200
- Sizes.text (field in host), 199
- sizes\_for (procedure in host), 211
- slab\_plan (procedure in core), 130
- Slab\_Plan (struct in core), 110
- Slab\_Plan.attributes (field in core), 110
- Slab\_Plan.cells (field in core), 110
- Slab\_Plan.links (field in core), 110
- Slab\_Plan.lists (field in core), 110
- Slab\_Plan.nodes (field in core), 110
- Slab\_Plan.output (field in core), 110
- Slab\_Plan.reports (field in core), 110
- Slab\_Plan.scratch (field in core), 110
- Slab\_Plan.second\_bank (field in core), 110
- Slab\_Plan.sources (field in core), 110
- Slab\_Plan.text (field in core), 110
- Slot (struct in pyscan), 282
- Slot.doc (field in pyscan), 282
- Slot.name (field in pyscan), 282
- slots (procedure in pyscan), 286
- Smart\_Quotes (enum in readers/rst/inline), 320
  - (type in readers/rst), 309
- Smart\_Quotes.Alt (enumerator in readers/rst/inline), 320
- Smart\_Quotes.Off (enumerator in readers/rst/inline), 320
- Smart\_Quotes.On (enumerator in readers/rst/inline), 320
- Snapshot (struct in core), 110
- Snapshot.attributes (field in core), 110
- Snapshot.cells (field in core), 110
- Snapshot.links (field in core), 110
- Snapshot.lists (field in core), 110
- Snapshot.nodes (field in core), 110
- Snapshot.text (field in core), 110
- Snapshot.text\_used (field in core), 110
- Snapshot\_Storage (struct in core), 111
- Snapshot\_Storage.attributes (field in core), 111
- Snapshot\_Storage.cells (field in core), 111
- Snapshot\_Storage.links (field in core), 111
- Snapshot\_Storage.lists (field in core), 111
- Snapshot\_Storage.nodes (field in core), 111
- Snapshot\_Storage.text (field in core), 111
- sort\_by\_reference (procedure in gettext), 157
- sort\_items (procedure in inventory), 225
- Source (struct in core), 111
- Source.name (field in core), 111
- Source.text (field in core), 111
- source\_entry (procedure in core), 130
- Source\_Entry (struct in core), 111
- Source\_Entry.included\_at (field in core), 111
- Source\_Entry.parent (field in core), 111
- Source\_Entry.use (field in core), 111
- Source\_Entry.using (field in core), 111
- Source\_Id (type in core), 111
- source\_of (procedure in core), 130
- Source\_Span (struct in core), 111
- Source\_Span.end (field in core), 112
- Source\_Span.source (field in core), 112
- Source\_Span.start (field in core), 112
- source\_text (procedure in core), 130
- Source\_Use (enum in core), 112
- Source\_Use.Literal (enumerator in core), 112
- Source\_Use.Parsed (enumerator in core), 112
- span\_of (procedure in readers/rst), 316
- spend (procedure in core), 130
- sphinx\_options (procedure in defaults), 135
- spill\_disk (procedure in host), 211
- split\_label (procedure in core), 130
- split\_lines (procedure in pyscan), 286
- Stack (struct in core), 112
- Stack.items (field in core), 112
- Stack.used (field in core), 112
- stack\_budget\_limit (procedure in host), 212
- stack\_diagnostic (procedure in core), 130
- stack\_exhausted (procedure in core), 130
- stack\_guard (procedure in core), 131
- Stack\_Guard (struct in core), 112
- Stack\_Guard.base (field in core), 112
- Stack\_Guard.limit (field in core), 112
- STACK\_RESERVE (constant in core), 134
- stack\_slice (procedure in core), 131
- STAGE\_PREFIX (constant in host), 214
- standard\_role (procedure in readers/rst), 316
- start\_byte (procedure in treesitter), 336
- start\_point (procedure in treesitter), 336
- State (struct in jinja), 232
- State.autoescape (field in jinja), 232
- State.caller (field in jinja), 233
- State.ctx (field in jinja), 232
- State.current (field in jinja), 232

- State.depth (field in jinja), 232
- State.env (field in jinja), 232
- State.err (field in jinja), 232
- State.failed (field in jinja), 232
- State.guard (field in jinja), 233
- State.iterations (field in jinja), 232
- State.line (field in jinja), 232
- State.out (field in jinja), 232
- State.rng (field in jinja), 233
- State.scratch (field in jinja), 233
- State.stored (field in jinja), 233
- STATES (constant in gds), 143
- STATIC (constant in graphdog), 165
  - (constant in treesitter), 337
- stats (procedure in gettext), 157
- Stats (struct in gettext), 151
- Stats.fuzzy (field in gettext), 151
- Stats.obsolete (field in gettext), 151
- Stats.translated (field in gettext), 151
- Stats.untranslated (field in gettext), 151
- Status (enum in core), 112
- Status.Cancelled (enumerator in core), 113
- Status.Capacity (enumerator in core), 112
- Status.Internal (enumerator in core), 113
- Status.Invalid\_Input (enumerator in core), 112
- Status.Limit (enumerator in core), 113
- Status.Ok (enumerator in core), 112
- Status.Policy (enumerator in core), 112
- Status.Unsupported (enumerator in core), 112
- Storage (struct in core), 113
- Storage.banks (field in core), 113
- Storage.reports (field in core), 113
- Storage.scratch (field in core), 113
- Storage.sources (field in core), 113
- str (procedure in jinja), 240
- str\_value (procedure in pyscan), 286
- String\_Kind (enum in pyscan), 283
- String\_Kind.Bytes (enumerator in pyscan), 283
- String\_Kind.Formatted (enumerator in pyscan), 283
- String\_Kind.Str (enumerator in pyscan), 283
- string\_value (procedure in pyscan), 287
- Style (enum in highlight), 172
- Style.Abap (enumerator in highlight), 172
- Style.Algol (enumerator in highlight), 172
- Style.Algol\_Nu (enumerator in highlight), 172
- Style.Arduino (enumerator in highlight), 172
- Style.Autumn (enumerator in highlight), 172
- Style.Borland (enumerator in highlight), 172
- Style.Bw (enumerator in highlight), 172
- Style.Coffee (enumerator in highlight), 172
- Style.Colorful (enumerator in highlight), 172
- Style.Default (enumerator in highlight), 173
- Style.Dracula (enumerator in highlight), 173
- Style.Emacs (enumerator in highlight), 173
- Style.Friendly (enumerator in highlight), 173
- Style.Friendly\_Grayscale (enumerator in highlight), 173
- Style.Fruity (enumerator in highlight), 173
- Style.Github\_Dark (enumerator in highlight), 173
- Style.Gruvbox\_Dark (enumerator in highlight), 173
- Style.Gruvbox\_Light (enumerator in highlight), 173
- Style.Igor (enumerator in highlight), 173
- Style.Inkpot (enumerator in highlight), 173
- Style.Lightbulb (enumerator in highlight), 173
- Style.Lilypond (enumerator in highlight), 173
- Style.Lovelace (enumerator in highlight), 173
- Style.Manni (enumerator in highlight), 173
- Style.Material (enumerator in highlight), 173
- Style.Monokai (enumerator in highlight), 173
- Style.Murphy (enumerator in highlight), 173
- Style.Native (enumerator in highlight), 173
- Style.None (enumerator in highlight), 174
- Style.Nord (enumerator in highlight), 174
- Style.Nord\_Darker (enumerator in highlight), 174
- Style.One\_Dark (enumerator in highlight), 174
- Style.Paraíso\_Dark (enumerator in highlight), 174
- Style.Paraíso\_Light (enumerator in highlight), 174
- Style.Pastie (enumerator in highlight), 174
- Style.Perldoc (enumerator in highlight), 174
- Style.Rainbow\_Dash (enumerator in highlight), 174
- Style.Rrt (enumerator in highlight), 174
- Style.Sas (enumerator in highlight), 174
- Style.Solarized\_Dark (enumerator in highlight), 174
- Style.Solarized\_Light (enumerator in highlight), 174
- Style.Sphinx (enumerator in highlight), 174
- Style.Staroffice (enumerator in highlight), 174
- Style.Stata\_Dark (enumerator in highlight), 174
- Style.Stata\_Light (enumerator in highlight), 174
- Style.Tango (enumerator in highlight), 174
- Style.Trac (enumerator in highlight), 174
- Style.Vim (enumerator in highlight), 175
- Style.Vs (enumerator in highlight), 175
- Style.Xcode (enumerator in highlight), 175
- Style.Zenburn (enumerator in highlight), 175
- Style\_Info (struct in highlight), 175



Style\_Info.background (field in highlight), 175  
 Style\_Info.base (field in highlight), 175  
 Style\_Info.dark (field in highlight), 175  
 Style\_Info.highlight (field in highlight), 175  
 Style\_Info.line\_number (field in highlight), 175  
 Style\_Info.line\_number\_background (field in highlight), 175  
 Style\_Info.name (field in highlight), 175  
 Style\_Info.rules (field in highlight), 175  
 Style\_Info.special\_background (field in highlight), 175  
 Style\_Info.special\_line\_number (field in highlight), 175  
 style\_name (procedure in highlight), 185  
 Style\_Rule (struct in highlight), 175  
 Style\_Rule.definition (field in highlight), 176  
 Style\_Rule.kind (field in highlight), 175  
 STYLES (variable in highlight), 186  
 Substitution (struct in readers/commonmark), 296  
 Substitution.name (field in readers/commonmark), 296  
 Substitution.value (field in readers/commonmark), 296  
 Substitution\_Hook (struct in readers/rst/inline), 321  
 Substitution\_Hook.procedure (field in readers/rst/inline), 321  
 Substitution\_Hook.user (field in readers/rst/inline), 321  
 subtree\_end (procedure in core), 131  
 suggest\_budget (procedure in host), 212  
 suggest\_lexer (procedure in highlight), 185  
 symbol (procedure in treesitter), 336  
 Symbol (struct in core), 113  
 (type in treesitter), 332  
 Symbol.hash (field in core), 113  
 Symbol.key (field in core), 113  
 Symbol.used (field in core), 113  
 Symbol.value (field in core), 113  
 symbol\_get (procedure in core), 131  
 symbol\_put (procedure in core), 131  
 symbol\_set (procedure in core), 131  
 Symbol\_Status (enum in core), 113  
 Symbol\_Status.Duplicate (enumerator in core), 113  
 Symbol\_Status.Full (enumerator in core), 113  
 Symbol\_Status.Ok (enumerator in core), 113  
 Symbol\_Status.Probe\_Limit (enumerator in core), 114  
 Symbol\_Table (struct in core), 114  
 Symbol\_Table.count (field in core), 114  
 Symbol\_Table.max\_probe (field in core), 114  
 Symbol\_Table.slots (field in core), 114  
 symbol\_table\_init (procedure in core), 131  
 sync\_all (procedure in host), 212  
 sync\_each (procedure in host), 212

Syntax\_Extension (enum in readers/commonmark), 296  
 Syntax\_Extension.Amsmath (enumerator in readers/commonmark), 296  
 Syntax\_Extension.Colon\_Fence (enumerator in readers/commonmark), 296  
 Syntax\_Extension.Deflist (enumerator in readers/commonmark), 296  
 Syntax\_Extension.Dollarmath (enumerator in readers/commonmark), 296  
 Syntax\_Extension.Fieldlist (enumerator in readers/commonmark), 296  
 Syntax\_Extension.Html\_Admonition (enumerator in readers/commonmark), 296  
 Syntax\_Extension.Html\_Image (enumerator in readers/commonmark), 296  
 Syntax\_Extension.Linkify (enumerator in readers/commonmark), 297  
 Syntax\_Extension.Replacements (enumerator in readers/commonmark), 297  
 Syntax\_Extension.Smartquotes (enumerator in readers/commonmark), 297  
 Syntax\_Extension.Substitution (enumerator in readers/commonmark), 297  
 Syntax\_Extension.Tasklist (enumerator in readers/commonmark), 297  
 syntax\_extension\_named (procedure in readers/commonmark), 299  
 Syntax\_Extensions (bit set in readers/commonmark), 297  
 syntax\_rule (procedure in readers/commonmark), 299

## T

Table (struct in core), 114  
 Table.rows (field in core), 114  
 Table.used (field in core), 114  
 table\_decorate (procedure in readers/rst), 316  
 Table\_Extra (struct in readers/rst), 310  
 Table\_Extra.align (field in readers/rst), 310  
 Table\_Extra.auto\_widths (field in readers/rst), 310  
 Table\_Extra.classes (field in readers/rst), 310  
 Table\_Extra.found (field in readers/rst), 310  
 Table\_Extra.name (field in readers/rst), 310  
 Table\_Extra.title (field in readers/rst), 310  
 Table\_Extra.width (field in readers/rst), 310  
 Table\_Extra.width\_class (field in readers/rst), 310  
 Table\_Extra.widths (field in readers/rst), 310

- take (procedure in readers/rst), 316
- Target (enum in host), 200
  - (enum in i18n), 217
  - (enum in typst), 340
- Target.Doctest\_Block (enumerator in i18n), 217
- Target.Html (enumerator in host), 200
  - (enumerator in typst), 340
- Target.Image (enumerator in i18n), 217
- Target.Index (enumerator in i18n), 217
- Target.Literal\_Block (enumerator in i18n), 217
- Target.Pdf (enumerator in host), 200
  - (enumerator in typst), 340
- Target.Raw (enumerator in i18n), 217
- Targets (bit set in i18n), 217
  - (type in odindoc), 247
- Template (struct in i18n), 217
  - (struct in jinja), 233
- Template.catalog (field in i18n), 218
- Template.compiled (field in jinja), 233
- Template.env (field in jinja), 233
- Template.name (field in jinja), 233
- Template.seen (field in i18n), 218
- Template.uuids (field in i18n), 218
- text (procedure group in gettext), 158
  - (procedure in core), 131
  - (procedure in pyscan), 287
  - (procedure in treesitter), 336
- Text (struct in core), 114
- Text.length (field in core), 114
- Text.start (field in core), 114
- text\_close (procedure in core), 131
- text\_domain (procedure in gettext), 157
- text\_open (procedure in core), 132
- text\_push (procedure in core), 132
- text\_push\_byte (procedure in core), 132
- text\_push\_rune (procedure in core), 132
- text\_rewind (procedure in core), 132
- thread\_stack\_bytes (procedure in host), 212
- to\_string (procedure in jinja), 240
- Toc\_Parents (enum in readers/sphinx), 328
- Toc\_Parents.All (enumerator in readers/sphinx), 328
- Toc\_Parents.Domain (enumerator in readers/sphinx), 328
- Toc\_Parents.Hide (enumerator in readers/sphinx), 328
- today (procedure in gds), 142
- Token (struct in highlight), 176
- Token.end (field in highlight), 176
- Token.kind (field in highlight), 176
- Token.start (field in highlight), 176
- Token\_Kind (enum in highlight), 176
- Token\_Kind.Chord\_Modifier (enumerator in highlight), 182
- Token\_Kind.Comment (enumerator in highlight), 179
- Token\_Kind.Comment\_Hashbang (enumerator in highlight), 179
- Token\_Kind.Comment\_Multiline (enumerator in highlight), 179
- Token\_Kind.Comment\_PreProc (enumerator in highlight), 180
- Token\_Kind.Comment\_Preproc (enumerator in highlight), 179
- Token\_Kind.Comment\_Preproc\_File (enumerator in highlight), 179
- Token\_Kind.Comment\_Single (enumerator in highlight), 179
- Token\_Kind.Comment\_Special (enumerator in highlight), 179
- Token\_Kind.Error (enumerator in highlight), 176
- Token\_Kind.Escape (enumerator in highlight), 176
- Token\_Kind.Generic (enumerator in highlight), 180
- Token\_Kind.Generic\_Deleted (enumerator in highlight), 180
- Token\_Kind.Generic\_Emph (enumerator in highlight), 180
- Token\_Kind.Generic\_Emph\_Strong (enumerator in highlight), 180
- Token\_Kind.Generic\_Error (enumerator in highlight), 180
- Token\_Kind.Generic\_Heading (enumerator in highlight), 180
- Token\_Kind.Generic\_Inserted (enumerator in highlight), 180
- Token\_Kind.Generic\_Output (enumerator in highlight), 180
- Token\_Kind.Generic\_Prompt (enumerator in highlight), 180
- Token\_Kind.Generic\_Strong (enumerator in highlight), 180
- Token\_Kind.Generic\_Subheading (enumerator in highlight), 180
- Token\_Kind.Generic\_Traceback (enumerator in highlight), 180
- Token\_Kind.Generic\_Underline (enumerator in highlight), 180
- Token\_Kind.Keyword (enumerator in highlight), 176
- Token\_Kind.Keyword\_Constant (enumerator in highlight), 176
- Token\_Kind.Keyword\_Declaration (enumerator in highlight), 176
- Token\_Kind.Keyword\_Namespace (enumerator in highlight), 176
- Token\_Kind.Keyword\_Pseudo (enumerator in highlight), 176
- Token\_Kind.Keyword\_Reserved (enumerator in highlight), 177
- Token\_Kind.Keyword\_Type (enumerator in highlight), 177
- Token\_Kind.Literal (enumerator in highlight), 178
- Token\_Kind.Literal\_Date (enumerator in highlight), 178
- Token\_Kind.Literal\_Scalar\_Plain (enumerator in highlight), 180
- Token\_Kind.Name (enumerator in highlight), 177
- Token\_Kind.Name\_Attribute (enumerator in highlight), 177

- Token\_Kind.Name\_Backslash\_Reference  
(enumerator in highlight), 181
- Token\_Kind.Name\_Builtin (enumerator in highlight), 177
- Token\_Kind.Name\_Builtin\_Articulation  
(enumerator in highlight), 181
- Token\_Kind.Name\_Builtin\_Clef  
(enumerator in highlight), 181
- Token\_Kind.Name\_Builtin\_Context  
(enumerator in highlight), 181
- Token\_Kind.Name\_Builtin\_Context\_Property  
(enumerator in highlight), 181
- Token\_Kind.Name\_Builtin\_Dynamic  
(enumerator in highlight), 181
- Token\_Kind.Name\_Builtin\_Grob  
(enumerator in highlight), 181
- Token\_Kind.Name\_Builtin\_Grob\_Property  
(enumerator in highlight), 181
- Token\_Kind.Name\_Builtin\_Header\_Variable  
(enumerator in highlight), 181
- Token\_Kind.Name\_Builtin\_Markup\_Command  
(enumerator in highlight), 181
- Token\_Kind.Name\_Builtin\_Music\_Command  
(enumerator in highlight), 181
- Token\_Kind.Name\_Builtin\_Music\_Function  
(enumerator in highlight), 181
- Token\_Kind.Name\_Builtin\_Paper\_Variable  
(enumerator in highlight), 181
- Token\_Kind.Name\_Builtin\_Pseudo  
(enumerator in highlight), 177
- Token\_Kind.Name\_Builtin\_Repeat\_Type  
(enumerator in highlight), 181
- Token\_Kind.Name\_Builtin\_Scale  
(enumerator in highlight), 181
- Token\_Kind.Name\_Builtin\_Scheme\_Builtin  
(enumerator in highlight), 181
- Token\_Kind.Name\_Builtin\_Scheme\_Function  
(enumerator in highlight), 181
- Token\_Kind.Name\_Builtin\_Translator  
(enumerator in highlight), 181
- Token\_Kind.Name\_Class (enumerator in highlight), 177
- Token\_Kind.Name\_Constant (enumerator in highlight), 177
- Token\_Kind.Name\_Decorator (enumerator in highlight), 177
- Token\_Kind.Name\_Entity (enumerator in highlight), 177
- Token\_Kind.Name\_Exception (enumerator in highlight), 177
- Token\_Kind.Name\_Function (enumerator in highlight), 177
- Token\_Kind.Name\_Function\_Magic  
(enumerator in highlight), 177
- Token\_Kind.Name\_Label (enumerator in highlight), 177
- Token\_Kind.Name\_Lvalue (enumerator in highlight), 180
- Token\_Kind.Name\_Namespace  
(enumerator in highlight), 177
- Token\_Kind.Name\_Other (enumerator in highlight), 177
- Token\_Kind.Name\_Property (enumerator in highlight), 177
- Token\_Kind.Name\_Tag (enumerator in highlight), 177
- Token\_Kind.Name\_Variable (enumerator in highlight), 177
- Token\_Kind.Name\_Variable\_Class  
(enumerator in highlight), 178
- Token\_Kind.Name\_Variable\_Global  
(enumerator in highlight), 178
- Token\_Kind.Name\_Variable\_Instance  
(enumerator in highlight), 178
- Token\_Kind.Name\_Variable\_Magic  
(enumerator in highlight), 178
- Token\_Kind.Number (enumerator in highlight), 179
- Token\_Kind.Number\_Bin (enumerator in highlight), 179
- Token\_Kind.Number\_Float (enumerator in highlight), 179
- Token\_Kind.Number\_Hex (enumerator in highlight), 179
- Token\_Kind.Number\_Integer (enumerator in highlight), 179
- Token\_Kind.Number\_Integer\_Long  
(enumerator in highlight), 179
- Token\_Kind.Number\_Oct (enumerator in highlight), 179
- Token\_Kind.Operator (enumerator in highlight), 179
- Token\_Kind.Operator\_Word (enumerator in highlight), 179
- Token\_Kind.Other (enumerator in highlight), 176
- Token\_Kind.Pitch (enumerator in highlight), 182
- Token\_Kind.Punctuation (enumerator in highlight), 179
- Token\_Kind.Punctuation\_Indicator  
(enumerator in highlight), 180
- Token\_Kind.Punctuation\_Marker  
(enumerator in highlight), 179
- Token\_Kind.String (enumerator in highlight), 178
- Token\_Kind.String\_Affix (enumerator in highlight), 178
- Token\_Kind.String\_Backtick (enumerator in highlight), 178
- Token\_Kind.String\_Char (enumerator in highlight), 178
- Token\_Kind.String\_Delimiter (enumerator in highlight), 178
- Token\_Kind.String\_Doc (enumerator in highlight), 178
- Token\_Kind.String\_Double (enumerator in highlight), 178
- Token\_Kind.String\_Escape (enumerator in highlight), 178

- Token\_Kind.String\_Heredoc (enumerator in highlight), 178
  - Token\_Kind.String\_Interpol (enumerator in highlight), 178
  - Token\_Kind.String\_Other (enumerator in highlight), 178
  - Token\_Kind.String\_Regex (enumerator in highlight), 178
  - Token\_Kind.String\_Single (enumerator in highlight), 178
  - Token\_Kind.String\_Symbol (enumerator in highlight), 179
  - Token\_Kind.Text (enumerator in highlight), 176
  - Token\_Kind.Token\_Number (enumerator in highlight), 182
  - Token\_Kind.Token\_String (enumerator in highlight), 181
  - Token\_Kind.Token\_String\_Escape (enumerator in highlight), 182
  - Token\_Kind.Token\_String\_Symbol (enumerator in highlight), 182
  - Token\_Kind.Whitespace (enumerator in highlight), 176
  - token\_text (procedure in highlight), 185
  - tokenize (procedure in highlight), 185
  - Toml\_Kind (enum in project), 268
  - Toml\_Kind.Array (enumerator in project), 268
  - Toml\_Kind.Boolean (enumerator in project), 268
  - Toml\_Kind.Datetime (enumerator in project), 268
  - Toml\_Kind.Float (enumerator in project), 268
  - Toml\_Kind.Integer (enumerator in project), 268
  - Toml\_Kind.String (enumerator in project), 268
  - Toml\_Kind.Table (enumerator in project), 268
  - Toml\_Value (struct in project), 268
  - Toml\_Value.boolean (field in project), 269
  - Toml\_Value.defined (field in project), 269
  - Toml\_Value.float (field in project), 269
  - Toml\_Value.inline (field in project), 269
  - Toml\_Value.integer (field in project), 268
  - Toml\_Value.items (field in project), 269
  - Toml\_Value.keys (field in project), 269
  - Toml\_Value.kind (field in project), 268
  - Toml\_Value.line (field in project), 269
  - Toml\_Value.text (field in project), 268
  - Toml\_Value.values (field in project), 269
  - top (procedure in core), 132
  - transform\_document (procedure in core), 132
  - translated (procedure in gettext), 157
  - translation (procedure in gettext), 157
  - Translation (struct in i18n), 218
  - Translation.catalog (field in i18n), 218
  - Translation.images (field in i18n), 219
  - Translation.kept (field in i18n), 219
  - Translation.limits (field in i18n), 218
  - Translation.policy (field in i18n), 218
  - Translation.progress (field in i18n), 218
  - Translation.read (field in i18n), 218
  - Translation.read\_inline (field in i18n), 218
  - Translation.source\_name (field in i18n), 218
  - Translation.targets (field in i18n), 218
  - Translation.translated (field in i18n), 219
  - translation\_pass (procedure in i18n), 222
  - Translator (struct in gettext), 151
  - Translator allocator (field in gettext), 151
  - Translator.bound (field in gettext), 151
  - Translator.domain (field in gettext), 151
  - Translator.language (field in gettext), 151
  - Tree (struct in treesitter), 332
  - tree\_delete (procedure in treesitter), 337
  - treesitter (package), 331
  - trim\_blank\_end (procedure in readers/rst), 316
  - trusted\_policy (procedure in core), 132
  - truthy (procedure in jinja), 240
  - try\_unparse (procedure in pyscan), 287
  - ts (foreign import in treesitter), 337
  - tuple (procedure in jinja), 240
  - Type\_Alias (struct in autodoc), 78
  - Type\_Alias.name (field in autodoc), 78
  - Type\_Alias.target (field in autodoc), 78
  - type\_name (procedure in jinja), 240
  - typst (package), 337
  - typst\_highlighter (procedure in highlight/guidedoc), 186
- ## U
- UI\_DOMAIN (constant in i18n), 223
  - UI\_Problem (struct in i18n), 219
  - UI\_Problem.error (field in i18n), 219
  - UI\_Problem.path (field in i18n), 219
  - ui\_text (procedure in i18n), 222
  - ui\_translator (procedure in i18n), 222
  - undefined (procedure in jinja), 240
  - Undefined (struct in jinja), 233
  - Undefined.hint (field in jinja), 233
  - Undefined.name (field in jinja), 233
  - Undefined.owner (field in jinja), 233
  - Undefined\_Mode (enum in jinja), 234
  - Undefined\_Mode.Chainable (enumerator in jinja), 234
  - Undefined\_Mode.Debug (enumerator in jinja), 234
  - Undefined\_Mode.Default (enumerator in jinja), 234
  - Undefined\_Mode.Strict (enumerator in jinja), 234
  - unescape (procedure in readers/rst/inline), 322
  - Unit (enum in i18n), 219
  - Unit.Alt (enumerator in i18n), 219
  - Unit.Caption (enumerator in i18n), 220
  - Unit.Contents (enumerator in i18n), 220
  - Unit.Entry (enumerator in i18n), 220
  - Unit.Graphviz (enumerator in i18n), 219
  - Unit.Index (enumerator in i18n), 220
  - Unit.Inline (enumerator in i18n), 219
  - Unit.Literal (enumerator in i18n), 219
  - Unit.Meta (enumerator in i18n), 220
  - Unit.None (enumerator in i18n), 219
  - Unit.Parsed (enumerator in i18n), 219
  - unit\_of (procedure in i18n), 222
  - unload (procedure in project), 271

unparse (procedure in pyscan), 287  
 Unparse\_Mode (enum in pyscan), 283  
 Unparse\_Mode.Python (enumerator in  
   pyscan), 283  
 Unparse\_Mode.Sphinx (enumerator in  
   pyscan), 283  
 Unparse\_Mode.Sphinx\_Repr (enumerator  
   in pyscan), 283  
 unwrap\_node (procedure in core), 132

## V

validate (procedure in core), 133  
 validate\_registry (procedure in core), 133  
 Value (union in jinja), 234  
 values\_equal (procedure in jinja), 240  
 VERSION (constant in cli), 79  
   (constant in core), 134  
   (constant in readers/commonmark), 299  
   (constant in readers/rst), 316  
   (constant in readers/sphinx), 329  
   (constant in render/html), 329  
   (constant in render/text), 330  
   (constant in render/typst), 331  
 version (procedure in graphdog), 165  
 VERSION\_SUFFIX (constant in cli), 79  
 version\_text (procedure in cli), 79  
 view (procedure in core), 133  
 View (struct in core), 114  
 View.snap (field in core), 114  
 View.sources (field in core), 114

## W

was\_interrupted (procedure in host), 212  
 with\_context (procedure group in gettext),  
   158  
 Workspace (struct in core), 114  
 Workspace.active (field in core), 115  
 Workspace.banks (field in core), 115  
 Workspace.generation (field in core), 115  
 Workspace.keep\_errors (field in core), 115  
 Workspace.output (field in core), 115  
 Workspace.ready (field in core), 115  
 Workspace.reports (field in core), 115  
 Workspace.revisions (field in core), 115  
 Workspace.scratch (field in core), 115  
 Workspace.sources (field in core), 115  
 Workspace.sources\_used (field in core),  
   115  
 workspace\_size (procedure in core), 133  
 Wrapper (enum in pyscan), 283  
 Wrapper.Cached\_Property (enumerator in  
   pyscan), 283  
 Wrapper.Classmethod (enumerator in  
   pyscan), 283  
 Wrapper.None (enumerator in pyscan), 283  
 Wrapper.Property (enumerator in pyscan),  
   283  
 Wrapper.Staticmethod (enumerator in  
   pyscan), 283  
 Wrapper.Unknown (enumerator in  
   pyscan), 284  
 write (procedure in inventory), 226  
 write\_css (procedure in highlight), 185  
 write\_figure\_filename (procedure in i18n),  
   223  
 write\_html (procedure in highlight), 185

write\_html\_lines (procedure in highlight),  
   185  
 write\_id (procedure in core), 133  
 write\_message (procedure in i18n), 223  
 write\_mo (procedure in gettext), 158  
 write\_object (procedure in core), 133  
 Write\_Options (struct in gettext), 151  
 Write\_Options.obsolete (field in gettext),  
   152  
 Write\_Options.previous (field in gettext),  
   152  
 Write\_Options.references (field in gettext),  
   151  
 Write\_Options.width (field in gettext), 151  
 write\_po (procedure in gettext), 158  
 write\_problem (procedure in highlight),  
   186  
 write\_template (procedure in i18n), 223  
 write\_typst (procedure in highlight), 186  
 write\_typst\_theme (procedure in  
   highlight), 186  
 write\_unsynced (procedure in host), 212