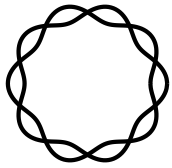

Simple System TEX Manual

Content

S^{imple}_{ystem}TEX Overview

S^{imple}_{ystem}TEX is a LATEX package built on LATEX3, specifically designed for large-scale, integrated typesetting tasks. It provides an automated file & block management system, while also featuring convenient tools for index generation and cross-referencing.

The core design philosophy of S^{imple}_{ystem}TEX is to maximize the **separation of content and structure**. This mechanism frees content creators from constantly managing global document structure and stylistic details, allowing them to focus on writing. Consequently, it significantly reduces the maintenance burden of large documents. This approach is particularly beneficial for boosting efficiency when utilizing AI tools for text polishing or assisting.

File Management System

In S^{imple}_{ystem}TEX , a *Section* serves as the fundamental content unit of the document. Each section corresponds to an independent subfile (i.e. a *Section File*) that stores the specific text code. The directory containing these subfiles is referred to as a *Group*. Groups support up to four levels of nesting; however, to minimize the complexity of the styling code, the nesting depth should be kept as shallow as possible.

Main File

The document body of a S^{imple}_{ystem}TEX main file serves a dual purpose: typesetting both the text and the table of contents (TOC). When a specific section file is imported using the `\ImportSection` command, the system not only renders the corresponding content within the text but also automatically adds it to the TOC. By simply adjusting the order of the `\ImportSection` commands within the main file, you can easily alter the sequence in which sections appear in both the text and the TOC.

If you need to use content typesetting commands directly within the main file, please wrap them inside the following command:

Content Typesetting Command

`\TextCommand {⟨LaTeX Command⟩¶}`

Mandatory Arguments

- ⟨*LaTeX Command*⟩¶ *: The LATEX code to be executed during content typesetting.*

Although this command allows you to write text directly in the main file, we strongly advise against extensive use of this practice. Doing so can easily lead to a disorganized

document structure, increase subsequent maintenance efforts, and contradict the core design philosophy of S^{imple}_{ystem}TEX .

Please note that any non–S^{imple}_{ystem}TEX commands in the main file that are not wrapped inside a `\TextCommand` will be executed at the very beginning of the document. Therefore, LATEX code for the cover page does not require additional wrapping.

Blocks

Blocks are the core structural elements used for organizing and presenting content in S^{imple}_{ystem}TEX . By creating various types of blocks, one can clearly distinguish different forms of content, such as definitions, theorems, and proofs. Furthermore, blocks facilitate seamless navigation and information retrieval throughout the document via built-in cross-reference and indexing functionalities.

An *Anonymous Block* is a special type of block, typically utilized for explanatory content or as supplement to other blocks. S^{imple}_{ystem}TEX will not generate cross-reference anchors for anonymous blocks; consequently, they cannot serve as targets for hyperlinks, nor will they appear in the block index.

To enhance the readability and consistency of the document, users should structure their text and apply appropriate block types based on the content's nature. We strongly advise against nesting regular blocks within one another.

Cross-Reference

In S^{imple}_{ystem}TEX , the cross-reference mechanism is deeply integrated with the file & block management system. Importation of section files or creation of blocks will automatically generate corresponding anchors. Users can insert hyperlinks by simply providing the identifier of the target section or block, instead of managing anchors manually. This ensures that cross-references dynamically adapt to content modifications, thereby significantly reducing maintenance overhead.

S^{imple}_{ystem}TEX Manual

Consistent with any document authored using S^{imple}_{ystem}TEX , this manual utilizes sections as its fundamental content units, which are categorized into top-level groups based on the usage contexts of the commands they describe. All commands are presented using a uniformly styled command block type, and the **S^{imple}_{ystem}TEX** Command Index is provided in the appendix for quick reference.

When invoking S^{imple}_{ystem}TEX commands, you must supply all required mandatory arguments. If optional arguments are omitted, their corresponding enclosing delimiters (e.g. `()` , `<>`) must also be omitted. The pilcrow symbol ¶ indicates that the argument accepts

multi-paragraph LaTeX code, allowing the use of the `\par` command or consecutive blank lines.

The vast majority of S^{imple}_{ystem}TeX commands have been actively utilized in the typesetting of this very manual and other demonstration projects. We encourage you to review their source code to familiarize yourself with their practical applications.

Last Revised: 2026 – 03 – 24.

- S^{imple}_{ystem}TeX Repository: https://github.com/JokerXin2025/SimpleSystemTeX.
- Author's Email: jokerxin2025@163.com.

MainFile | Importing Section Files

To avoid repeatedly typing complex file paths when importing section files, one should predefine a *Section Path Prefix*. When importing a section file or adding a section link, you only need to provide the portion of the file path excluding the prefix and the file extension (i.e. the *Section Path*). The section path minus the section file name is referred to as the *Group Path* of that section. Please note that none of the commands provided by S^{imple}_{ystem} TEX require the full section file path. Providing the full path will prevent the group structure from being parsed correctly.

The system defaults to `./sec/` as the section path prefix. Therefore, please ensure your project adheres to the following directory structure:

```
⟨Project⟩/
├─ ⟨Project⟩.tex ..................................... Main File
├─ simplesystemtex.sty ............................... SimpleSystemTeX Package
├─ sec/
│  └─ ⟨Group⟩/ ....................................... First-Level Group
│     ├─ ⟨Subgroup⟩/ ................................ Second-Level Group
│     │  └─ ⟨Section1⟩.tex
│     └─ ⟨Section2⟩.tex
```

Taking above file structure as an example, the respective paths for ⟨*Section1*⟩ are:

- Section File Path: `./sec/`⟨*Group*⟩`/`⟨*Subgroup*⟩`/`⟨*Section1*⟩`.tex`
- Section Path: ⟨*Group*⟩`/`⟨*Subgroup*⟩`/`⟨*Section1*⟩
- Group Path: ⟨*Group*⟩`/`⟨*Subgroup*⟩`/`

You can modify the section path prefix using the following command to accommodate existing project structures:

Modify Section Path Prefix

`\SetSectionPath` ⟨*delimiter*⟩⟨*path prefix*⟩⟨*delimiter*⟩

Mandatory Arguments

- ⟨*path prefix*⟩ : *The new section path prefix. It must begin with* `./` *(current directory) or* `../` *(parent directory) and end with* `/` *. The path is relative to the directory containing the main file, with directories separated by* `/` *.*

You may use a matched pair of braces `{` and `}`, or any two identical characters as delimiters. Please ensure that the path prefix does not contain the delimiter character.

After you modify the section path prefix, SimpleTeX will parse the group structure according to the new setting. For instance, if the prefix is changed to `./sec/⟨Group⟩/` , the top-level group for ⟨ *Section1* ⟩ becomes ⟨ *Subgroup* ⟩ . Please ensure the new section path prefix matches your intended file structure.

Use the following command in the document body of the main file to import sections:

Import Section File

`\ImportSection {⟨section path⟩} (⟨display name⟩) <⟨custom parameter⟩>`

Mandatory Arguments

• ⟨ *section path* ⟩ : *The path of the section to be imported, which must not contain special characters* `\` , `%` , `{` , `}` , *or consecutive spaces.*

Optional Arguments

• ⟨ *display name* ⟩ : *The name of the section to be displayed. If this argument is empty, the section file name will be used by default.*

• ⟨ *custom parameter* ⟩ : *Used in* Preamble | Style Configuration *to achieve various complex typesetting logic.*

When the section file name contains unprintable special characters, you must provide an escaped display name to ensure the section name renders correctly in the table of contents, section headings, and cross-references.

Example

```
\ImportSection {Linear Space/Hamel Basis}
\ImportSection {Topology Space/Interior&Closure} (Interior\& Closure)
\ImportSection {Measure Space/Lp Convergence} ($L^p$ Convergence)
```

SimpleTeX determines the rendering order of sections in the document body, as well as their entry sequence in the table of contents, based on the order in which the section files are imported. You can also add ”parts” to group sections together. Please note that grouping by parts depends entirely on the import order of the section files and is independent of the group structure in which the sections reside.

Create Part

`\AddPart {⟨part⟩}`

Mandatory Arguments

- ⟨ *part* ⟩ : *The name of the part to be displayed in the table of contents.*

As an example, the following shows part of the project $S^{imple}_{ystem}TEX$ Manual's structure alongside its corresponding import commands:

```
├─ simplesystemtex.doc.tex
└─ sec/
   ├─ SimpleSystemTeX Command Index.tex
   ├─ SimpleSystemTeX Overview.tex
   ├─ Style Code Preset Values.tex
   ├─ Main File/
   │  ├─ Generating TOC.tex
   │  └─ Importing Section Files.tex
   ├─ Preamble/
   │  ├─ Declaring Block Types.tex
   │  └─ Style Configuration.tex
   └─ Text/
      ├─ Adding Links.tex
      ├─ Creating Blocks.tex
      └─ Generating Block Indexes.tex
```

Example

```
\AddPart {\SimpleSystemTeX[bf]}
\ImportSection {SimpleSystemTeX Overview}(\SimpleSystemTeX[bf] Overview)
\AddPart {File Management}
\ImportSection {Main File/Importing Section Files}
\ImportSection {Main File/Generating TOC}
\AddPart {Blocks\& Cross-Reference}
\ImportSection {Preamble/Declaring Block Types}
\ImportSection {Text/Creating Blocks}
\ImportSection {Text/Generating Block Indexes}
\ImportSection {Text/Adding Links}
\AddPart {Others}
\ImportSection {Preamble/Style Configuration}
\AddPart {Appendix}
\ImportSection {Style Code Preset Values}
\ImportSection {SimpleSystemTeX Command Index} (\SimpleSystemTeX[bf]Command Index)
```

MainFile | Generating TOC

Use the following command in the document body of the main file to generate TOC:

Generate Table of Contents

```
\TableofContents
```

The table of contents generated by `\TableofContents` will automatically include all section files imported via `\ImportSection`. The table of contents itself will not be listed as an entry within it. After the first compilation following any document modifications, all section entries and part titles will update automatically. However, due to the multi-pass nature of TEX's cross-reference mechanism, you will need to compile a second time to ensure page numbers and hyperlinks are displayed correctly.

If you need to execute typesetting commands between two section entries within the table of contents, please place them between the corresponding `\ImportSection` commands and wrap them in the following command:

TOC Typesetting Command

```
\TableCommand {⟨LaTeX Command⟩¶}
```

Mandatory Arguments

- *⟨LaTeX Command⟩¶ : The LaTeX code to be executed during TOC typesetting.*

Example

```
\TableofContents
\AddPart {Part A}
\ImportSection {Group/Section A1}
\ImportSection {Group/Section A2}
\AddPart {Part B}
\ImportSection {Group/Section B1}
\TableCommand {\bigskip\bigskip}
\TextCommand {\newpage}
\AddPart {Appendix}
\ImportSection {Appendix 1}
\ImportSection {Appendix 2}
```

Preamble | Declaring Block Types

Declare Block Type

`\NewBlockType {⟨regular block type⟩}`

`\NewBlockType* {⟨anonymous block type⟩}`

Mandatory Arguments

• ⟨*regular block type*⟩ , ⟨*anonymous block type*⟩ : *The name of the block type to be declared, which can only contain letters and is case-sensitive.*

You cannot use `\NewBlockType` or `\NewBlockType*` commands on a block type that has already been created, but you can redefine their styles. S^{imple}_{ystem} TEX provides the following predefined basic block types:

Predefined Regular Block Types

- `Definition` : Mathematical definition.
- `Theorem` : Mathematical theorem.
- `Corollary` : Mathematical corollary.

Predefined Anonymous Block Types

- `Proof` : Mathematical proof.

After declaring a new block type, you need to perform Style Configuration for it.

Text | Creating Blocks

You can use the following commands in the text or Style Code to create a block of an already defined block type:

Create Block

\⟨*regular block type*⟩ {⟨*label*⟩} (⟨*display name*⟩) <⟨*custom parameter*⟩> {⟨*content*⟩¶}

\⟨*anonymous block type*⟩ (⟨*display name*⟩) <⟨*custom parameter*⟩> {⟨*content*⟩¶}

Mandatory Arguments

- ⟨*regular block type*⟩ , ⟨*anonymous block type*⟩ : *The name of the block type to be created.*
- ⟨*label*⟩ : *The unique identifier of the block. Do not start with a* **.** .
- ⟨*content*⟩¶ : *The specific content of the block.*

Optional Arguments

- ⟨*display name*⟩ : *The name of the block used for display. If this argument is empty, the* ⟨*label*⟩ *will be used by default.*
- ⟨*custom parameter*⟩ : *Used in* Preamble | Style Configuration *to achieve various complex typesetting logic.*

Example

```
\Theorem {Orthonormality Lemma} {
   Let $(\bm e_k)_{k=1}^n$ be an orthonormal system of vectors, then
   \begin{align*}
      \forall (a_k)_{k=1}^n \in \mathbf{F}^n
      \left|\left| \sum_{k=1}^n a_k \bm e_k \right|\right|^2
      = \sum_{k=1}^n \left| a_k \right|^2 \ .
   \end{align*}
}
\Proof {Orthonormality Lemma-proof1} (Proof I) {
   \begin{align*}
      \left|\left| \sum_{k=1}^n a_k \bm e_k \right|\right|^2
      = \left\langle \sum_{k=1}^n a_k \bm e_k\ ,\ \sum_{k=1}^n a_k \bm e_k
      \right\rangle
      = \sum_{i=1}^n \sum_{j=1}^n a_i \overline{a_{j}} \delta_{i,j}
      = \sum_{k=1}^n | a_k |^2 \ .
   \end{align*}
}
```

```
\Proof {Orthonormality Lemma-proof2} (Proof II) <Induction> {
   \par For $n=1$ , we have
   \begin{align*}
      || a_1 \bm e_1 || = | a_1 | || \bm e_1 || = | a_1 | \ .
   \end{align*}
   \par Suppose the proposition holds for $n=m$ . By the \blclink{Pythagorean
   Theorem}, we have
   \begin{align*}
      \left|\left| \sum_{k=1}^{m+1} a_k \bm e_k \right|\right|^2
      = \left|\left| \sum_{k=1}^m a_k \bm e_k \right|\right|^2
      + \left|\left| a_{m+1} \bm e_{m+1} \right|\right|^2
      = \sum_{k=1}^m | a_k |^2 + | a_{m+1} |^2
      = \sum_{k=1}^{m+1} | a_k |^2 \ .
   \end{align*}
   Thus, the proposition holds for $n=m+1$ , and consequently, it holds for all
   positive integers $n$ .
}
```

When creating a block, please ensure that this block type has already been declared via the `\NewBlockType` command. The label of a regular block serves as the unique identifier for cross-referencing and cannot duplicate that of other blocks. You can use the following command to quickly create an anonymous block under a regular block type:

Quickly Create Anonymous Block

```
\⟨regular block type⟩* {⟨label⟩} (⟨display name⟩) <⟨parameter⟩> {⟨content⟩¶}
```

If the ⟨*content*⟩ is redundant for a specific block type, you can write macros in the preamble to wrap the block creation command. This allows you to omit the content argument when creating the block. The following example demonstrates a simplified command for creating a `MyBlockType` block:

Example

```
\NewBlockType {MyBlockType}
\NewDocumentCommand \myblocktype {m d() d<>} {
   \MyBlockType {#1} (#2) <#3> {}
}
```

A similar wrapping technique can also be employed for the synchronized creation of sub-blocks. Please refer to the following example:

Example

```
\NewDocumentCommand \TheoremWithProof {m +m +d[] +d[]} {
   \ParentBlockType {#1-parent} (#1) {#2}
   \IfVauleT {#3} {
      \Proof {#1-proofA} (#1) {#3}
   }
   \IfVauleT {#4} {
      \Proof {#1-proofB} (#1) {#4}
   }
}
```

To easily distinguish between the block types themselves and your wrapped macros, we recommend using words with capitalized first letters (PascalCase) for block type names, and words starting with a lowercase letter (camelCase) to name your wrapper commands.

Text | Generating Block Indexes

The primary advantage of encapsulating content components within blocks is the ability to use the `\MakeBlockIndex` command anywhere in the document body to rapidly generate an index of all blocks of the same type in a specific group. Furthermore, block indices are equipped with cross-reference anchors, allowing you to add block index links elsewhere in the document to help readers navigate directly to the index.

Thanks to the hierarchical tree structure of groups, an index generated for a specific group will automatically encompass the block entries from all its descendant subgroups. Cross-reference anchors for these subgroup indices are also generated simultaneously.

Generate Block Index

```
\MakeBlockIndex {⟨scope path⟩} [⟨block type list⟩] <⟨custom parameter⟩>
```

Mandatory Arguments

- ⟨ `scope path` ⟩ : *Specifies the group path for the blocks to be included in the index.*

Optional Arguments

- ⟨ `block type list` ⟩ : *Specifies the list of regular block types to be included in the index, separated by* `,` *. If this argument is empty, the index will include all block types by default.*
- ⟨ `custom parameter` ⟩ : *Used in Preamble | Style Configuration to achieve various complex typesetting logic.*

The scope path serves as the unique identifier for cross-referencing a group index. So you cannot generate two block indices with overlapping scopes, even if they specify different block type lists. If generating overlapping block indices is strictly necessary, you can use the following command to create a temporary block index that does not generate anchors:

Generate Temporary Block Index

```
\MakeBlockIndex* {⟨scope path⟩} [⟨block type list⟩] <⟨custom parameter⟩>
```

Please do not use block index generation commands within the arguments of blocks or other commands.

Text | Adding Links

Add Section Link

`\seclink {⟨section path⟩} (⟨link text⟩)`

Mandatory Arguments

- ⟨ *section path* ⟩ : *The path of the target section.*

Optional Arguments

- ⟨ *link text* ⟩ : *The text to be displayed for the link.*

Add Block Link

`\blclink {⟨label⟩} (⟨link text⟩)`

Mandatory Arguments

- ⟨ *label* ⟩ : *The label of the target block.*

Optional Arguments

- ⟨ *link text* ⟩ : *The text to be displayed for the link.*

Add Block Index Link

`\indlink {⟨scope path⟩} (⟨link text⟩)`

Mandatory Arguments

- ⟨ *scope path* ⟩ : *The scope path of the target block index.*

Optional Arguments

- ⟨ *link text* ⟩ : *The text to be displayed for the link.*

The link will be rendered using ⟨ ***link text*** ⟩ as its display content. If this argument is omitted, S^{imple}_{ystem}TEX will apply a default link display style. You can modify the relevant display logic by altering the style code in the Style Configuration .

Preamble | Style Configuration

S^{imple}_{ystem}TEX allows users to customize *Style Code* by using the `\SetStyle` command in the preamble, thereby setting or modifying the styling of various document components. In addition to standard LaTeX commands, users can also utilize *Display Arguments* prefixed with `@` within the style code to dynamically extract document content. The values of these arguments can similarly be evaluated in conditional statements to achieve highly flexible typesetting effects.

Configure Style

`\SetStyle {⟨style⟩} {⟨style code⟩¶}`

Mandatory Arguments

- ⟨*style*⟩ : *The name of the style to be set or modified.*

Optional Arguments

- ⟨*style code*⟩¶ : *The LaTeX code used to define the appearance of the style. You may use standard LaTeX commands for typesetting, display arguments prefixed with* `@` *, and the following conditional statements prefixed with* `#` *:*
  - `#ifempty` *: Evaluates whether a display argument is empty.*

    `#ifempty @⟨argument⟩ {⟨true code⟩¶} {⟨false code⟩¶}`
  - `#ifis` *: Evaluates whether a display argument, after one level of expansion, equals a specific value.*

    `#ifis @⟨argument⟩ {⟨value⟩} {⟨true code⟩¶} {⟨false code⟩¶}`
  - `#ifequal` *: Evaluates whether two display arguments, after one level of expansion, are equal to each other.*

    `#ifequal @⟨argument1⟩ @⟨argument2⟩ {⟨true code⟩¶} {⟨false code⟩¶}`

When writing style code, please ensure that a display argument does not immediately follow any macro command, as this may lead to syntax parsing errors. You must insert a space before the `@` to prevent this issue, for example: `\par @⟨argument⟩` .

Customizable styles and their respective available display arguments include:

Table of Contents

- **`TOCTitle` : Table of the table of contents**.
- **`TOCPartTitle` : Part title in the table of contents**.
  - `@part` : Part.
- **`TOCSection` : Section entry in the table of contents**.
  - `@section` : Entry section file name.
  - `@sectionname` : Entry section's display name.
  - `@part` : Part containing the entry section.
  - `@group` , `@subgroup` ... : Top-level group, second-level group... containing the entry section.
  - `@sectionpath` : Entry section path.
  - `@sectionpage` : Page number where the entry section heading is located.
  - `@sectionparameter` : Custom parameter of the entry section.

Section

- **`SectionHeading` : Section Heading**.
  - `@section` : Section file name.
  - `@sectionname` : Section's display name.
  - `@part` : Part containing the section.
  - `@group` , `@subgroup` ... : Top-level group, second-level group... containing the section.
  - `@sectionpath` : Section path.
  - `@sectionpage` : Page number where the section heading is located.
  - `@sectionparameter` : Custom parameter of the section.
- ⟨***`block type`***⟩ **: Block**.
  - `@section` , `@sectionname` ... : (Display arguments of the containing section)
  - `@block` : Block's label, not available for anonymous blocks.
  - `@blockname` : Block's display name.
  - `@blockparameter` : Custom parameter of the block.
  - `@blockcontent` : Block's content.

Block Index

• **`IndexTitle` , `IndexSubtitle` ... : Titles of top-level group, second-level group... in the block index**.

◦ `@section` , `@sectionname` ... : (Display arguments of the containing section)

◦ `@indexgroup` , `@indexsubgroup` ... : Top-level group, second-level group... in the block index's scope path.

◦ `@indexparameter` : Custom parameter of the block index.

• **`IndexBlock` : Entry in the block index**.

◦ `@section` , `@sectionname` ... : (Display arguments of the containing section)

◦ `@indexparameter` : Custom parameter of the block index.

◦ `@indexblock` : Entry block's label, not available for anonymous blocks.

◦ `@indexblockname` : Entry block's display name.

◦ `@indexblockparameter` : Custom parameter of the entry block.

◦ `@indexsection` : Section file name containing the entry block.

◦ `@indexsectionname` : Display name of the section containing the entry block.

◦ `@indexpart` : Part containing the entry block.

◦ `@indexgroup` , `@indexsubgroup` ... : Top-level group, second-level group... containing the entry block.

◦ `@indexsectionpath` : Section path containing the entry block.

◦ `@indexsectionparameter` : Custom parameter of the section containing the entry block.

Cross-Reference

• **`SectionLink` : Section Link**.

◦ `@section` , `@sectionname` ... : (Display arguments of the containing section)

◦ `@block` , `@blockname` ... : (Display arguments of the containing block)

◦ `@linktext` : Link text.

◦ `@targetsection` : Target section file name.

◦ `@targetsectionname` : Target section's display name.

◦ `@targetpart` : Part containing the target section.

◦ `@targetgroup` , `@targetsubgroup` ... : Top-level group, second-level group... containing the target section.

◦ `@targetsectionpath` : Target section path.

◦ `@targetsectionpage` : Page number where the target section heading is located.

◦ `@targetsectionparameter` : Custom parameter of the target section.

- **`BlockLink` : Block Link.**
  - `@section` , `@sectionname` ... : (Display arguments of the containing section)
  - `@block` , `@blockname` ... : (Display arguments of the containing block)
  - `@linktext` : Link text.
  - `@targetblock` : Target block's label, not available for anonymous blocks.
  - `@targetblockname` : Target block's display name.
  - `@targetblockparameter` : Custom parameter of the target block.
- **`IndexLink` : Index Link.**
  - `@section` , `@sectionname` ... : (Display arguments of the containing section)
  - `@block` , `@blockname` ... : (Display arguments of the containing block)
  - `@linktext` : Link text.
  - `@targetgroup` , `@targetsubgroup` ... : Top-level group, second-level group... in the target index's scope path.
  - `@targetindexparameter` : Custom parameter of the target index.

Not all display arguments are intended for rendering content. For instance, if `@section` contains special characters, you might use `@sectionname` as a substitute. In such scenarios, `@section` should be used to provide the section file name for conditional evaluation, whereas `@sectionname` should be used for display in the document. Please select the appropriate display arguments according to your specific requirements when authoring style code.

Please avoid the following practices within your style code, as they may lead to errors such as scope conflicts among display arguments:

Bad Practice

- Do not leave label-type arguments containing special characters, like `@section`, directly exposed in the style code.
- Do not nest blocks in the style code of ⟨*block type*⟩ .
- Do not nest links in the style code of **`SectionLink`**, **`BlockLink`**, or **`IndexLink`**.
- Do not use `@blockcontent` in the style code of **`SectionLink`**, **`BlockLink`**, or **`IndexLink`**.
- Do not compare path-type arguments like `@sectionpath` with other arguments like `@section` or their concatenated values, because they possess different TEX catcodes.

Before authoring style code, please review the Style Code Preset Values . It is highly recommended to copy the specific portions you intend to modify into the preamble first, and then base your further adjustments upon them.

Style Code Preset Values

```
\SetStyle {TOCTitle} {%
    \bfseries\LARGE
    \begin{center}\textbf{Content}\end{center}%
    \bigskip
}
\SetStyle {TOCPartTitle} {%
    \bfseries\Large
    \par\bigskip\noindent
    @part%
    \par\medskip
}
\SetStyle {TOCSection} {%
    \bfseries
    \par\noindent
    #ifempty @group {%
        \seclink {@sectionpath} (%
            \color{black}%
            \!@sectionname%
            \cftdotfill{\cftnodots}%
            \textbf{@sectionpage}%
        )%
    } {%
        #ifempty @subgroup {%
            \seclink {@sectionpath} (%
                \color{black}%
                \!{\footnotesize @group}%
                \:|\:@sectionname%
                \cftdotfill{\cftnodots}%
                \textbf{@sectionpage}%
            )%
        } {%
            \seclink{@sectionpath} (%
                \color{black}%
                \!{\footnotesize @group}%
                \;@subgroup%
                \:|\:@sectionname%
                \cftdotfill{\cftnodots}%
                \textbf{@sectionpage}%
            )%
        }%
    }%
    \par
}
```

```
\SetStyle {IndexGroup} {%
    \bfseries\Large
    \par\medskip\noindent
    \indlink {@indexgroup/} (%
        \color{black}%
        \!@indexgroup%
    )%
    \par\medskip
}
\SetStyle {IndexSubgroup} {%
    \bfseries
    \par\medskip\noindent
    \indlink {@indexgroup/@indexsubgroup/} (%
        \color{black}%
        \!{\small @indexgroup}%
        \ {\large @indexsubgroup}%
    )%
    \par\smallskip
}
\SetStyle {IndexSubsubgroup} {%
    \bfseries\normalsize
    \par\smallskip\noindent
    \indlink {@indexgroup/@indexsubgroup/@indexsubsubgroup/} (%
        \color{black}%
        \!@indexsubsubgroup%
    )%
    \par
}
\SetStyle {IndexBlock} {%
    \bfseries
    \par\noindent
    \blclink{@indexblock} (%
        \color{black}%
        \!@indexblockname\quad @indexblockparameter%
        \cftdotfill{\cftnodots}%
        \textbf{@indexsectionpage}%
    )%
    \par
}
```

```
\SetStyle {SectionHeading} {%
    \bfseries
    \par\noindent
    #ifempty @group {%
        {\huge @sectionname}%
    } {%
        #ifempty @subgroup {%
            {\Large @group}%
            {\huge\;|\;@sectionname}%
        } {%
            {\Large @group}%
            \;\;{\huge @subgroup\;|\;@sectionname}%
        }%
    }%
    \par\bigskip
}
\SetStyle {SectionLink} {%
    \color{blue}%
    #ifempty @linktext {%
        #ifempty @targetgroup {%
            \,@targetsectionname\,%
        } {%
            #ifempty @targetsubgroup {%
                \,{\footnotesize @targetgroup}%
                \:|\:@targetsectionname\,%
            } {%
                \,{\footnotesize @targetgroup}%
                \;@targetsubgroup%
                \:|\:@targetsectionname\,%
            }%
        }%
    } {%
        \,@linktext\,%
    }%
}
```

```
\SetStyle {BlockLink} {%
    \color{blue}%
    #ifempty @linktext {%
        \,@targetblockname\,%
    } {%
        \,@linktext\,%
    }%
}
\SetStyle {IndexLink} {%
    \color{blue}%
    #ifempty @linktext {%
        #ifempty @targetsubgroup {%
            \,@targetgroup\,%
        } {%
            \,{\footnotesize @targetgroup}%
            \;@targetsubgroup\,%
        }%
    } {%
        \,@linktext\,%
    }%
}

\SetStyle {Definition} {%
    \color{purple}%
    \par\noindent{\bfseries @blockname}%
    \quad{\itshape @blockcontent}%
    \par
}
\SetStyle {Theorem} {%
    \color{teal}%
    \par\noindent{\bfseries @blockname}%
    \quad{\itshape @blockcontent}%
    \par
}
\SetStyle {Corollary} {%
    \color{brown}%
    \par\noindent{\bfseries @blockname}%
    \quad{\itshape @blockcontent}%
    \par
}
\SetStyle {Proof} {%
    \color{gray}\itshape
    \par\noindent
    @blockname {\scriptsize (@blockparameter)}%
    \par @blockcontent%
    \par
}
```

$\mathbf{S}^{\text{imple}}_{\text{ystem}}\TeX$ Command Index

Preamble

Main File

Text