nameauth — Name authority mechanism for consistency in text and index*

Charles P. Schaum†

Released 2021/02/27

Abstract

The nameauth package automates the correct formatting and indexing of names for professional writing. This aids the use of a **name authority** and the editing process without needing to retype name references.

Contents

*This file describes version 3.6, last revised 2021/02/27.

†E-mail: charles[dot]schaum@comcast.net

1 Quick Start

Disclaimer

Names are about real people; examples should be too. This manual mentions notable figures both living and deceased. All names herein are meant to be used respectfully, for teaching purposes only. At no time is any disrespect or bias intended.

1.1 Introduction

A **name authority** is a canonical, scholarly list of name forms to which all variant name forms and aliases must refer. The task dashboard (Section 1.3) guides one to various areas of interest. To load the defaults, simply type:

```
\usepackage{nameauth}
```

The nameauth macros permit ambiguity because name forms are ambiguous unless they are put into a cultural context. Therefore, keep it simple. Use the quick interface. Use the fewest number of nameauth macros for one's use case.

Package Design and Features

The editorial process for book-length projects may require one to add, delete, or relocate text. Several issues emerge from this:

- Professional writing needs a full name form to introduce a person, using shorter forms thereafter. Moving text may require re-checking names.
- If a name is keyed to another name or narrative event, moving text may require checking for anachronistic references.
- Including special information in the index, such as including non-Latin script name forms with Latin script forms, can be complex and tedious.
- Unless one is familiar with professional indexing, one might create incorrect index entries.
- One must check if any names straddle page breaks and index them.

The nameauth package provides automated solutions for all points above at the time of writing. Names become abstractions; they are verbs that alter state and nouns that have state. That improves accuracy and consistency:

- **Automate** name forms. First uses of names have long forms. Later uses are short by default. Names vary in the text, but not in the index.
- Implement **cross-cultural, multilingual naming conventions**.
- Implement **complex name formatting** using conditional elements.
- Improve indexing with **automatic sorting and tagging, and cross-reference control.** Indexing rules are based on Nancy C. Mulvany, *Indexing Books* (Chicago: University of Chicago Press, 1994). All references [Mulvany] refer to this edition.
- **Associate and retrieve information** bound to names.

Basic Index:
Douglass, Frederick
Bailey, Betsey

For example, from a biography written a century ago, we show reordered paragraphs that require no subsequent changes. We use the "quick interface" and no name formatting (the package default). We "forget" names at the top of the right-hand column to simulate not using them yet (Section 2.8.1):

`\Doug\` Frederick Douglass rose to eminence by sheer force of character and talents that neither slavery nor caste proscription could crush. Circumstances could not prevent him from becoming a freeman and a leader.

`\Doug's` Douglass's early life is perhaps the most complete indictment of the slave system ever presented at the bar of public opinion.

`\Doug\` Douglass was born in February, l8l7. His earliest memories centered around the private cabin of his grandmother, `\Bailey,` Betsey Bailey, who was charged with only the duty of looking after young children.

`\Doug's` Frederick Douglass's early life is perhaps the most complete indictment of the slave system ever presented at the bar of public opinion.

`\Doug\` Douglass was born in February, l8l7. His earliest memories centered around the private cabin of his grandmother, `\Bailey,` Betsey Bailey, who was charged with only the duty of looking after young children.

`\Doug\` Douglass rose to eminence by sheer force of character and talents that neither slavery nor caste proscription could crush. Circumstances could not prevent him from becoming a freeman and a leader.

1.2 How to Use the Manual

Topics in this manual that are more basic or frequent in use are toward the front. Topics that are more complex or less-used are toward the back. As topics get more advanced, various sections mutually inform each other.

For reference, throughout this manual we show simplified and complete **name patterns** in the margins (Section 2.11.5). These patterns control name behavior. In the early pages of the manual we also show **basic index entries** in the margins.

Special Signs

This manual uses signs and illustrative typesetting that are not built-in defaults of nameauth, but in some cases are implemented using it:

We often highlight first and later uses of names (Sections 2.6, 2.8.1).

† A dagger indicates "non-native" Eastern forms (Section 2.3.3).

‡ A double dagger shows usage of the obsolete syntax (Section 2.11.4).

§ A section mark denotes index entries of fictional names.

3.0 ← Major changes have package version numbers in the margin.

 ← The "dangerous bend" shows where caution is needed.

Thanks

Thanks to Marc van Dongen, Enrico Gregorio, Philipp Stephani, Heiko Oberdiek, Uwe Lueck, Dan Luecking and Robert Schlicht for assistance in early versions of this package. Thanks also to users for valuable feedback.

1.3 Task Dashboard

Here we link to sections by task in order to get things done quickly. Many sections have return links at their end that bring the reader back to this page.

Where do you want to go today?

Quick Start

Basic concepts: 1.4
Macros: 1.5, 1.6, 1.7
Various hints: 1.8

Basics

Package options: 2.1
Name macros: 2.2.1, 2.2.2
Simple Variants (text/index): 2.2.3, 2.4

Language

Western names: 2.3.1, 2.3.2
Eastern names: 2.3.1, 2.3.3
Particles: 2.3.4, 2.7, 2.10.3

Language

Medieval/Ancient: 2.3.5, 2.10.1
"Continental" typesetting:
2.7, 2.7.1, 2.7.2, 2.10.3

Index

Page entries, index control, & xrefs: 2.4.1

Index

Setting up automatic sorting: 2.4.2
Auto-add info to index entries: 2.4.3

Intermediate

Name info database: 2.5
Test for the presence of names:
2.8, 2.8.1, 2.8.2

Advanced

Various discussions about errors: 2.3.4, 2.11.2, 2.11.3, 2.11.4, 2.11.5, 2.11.6

Advanced

Formatting: 2.6, 2.7, 2.7.1, 2.7.2, 2.10.1, 2.10.2, 2.10.3
Customizing: 2.10.4

Advanced

Link names & text to sequences of time or ideas: 2.5, 2.8.2 (history/game books)

Advanced

Use different formats to call out information: 2.5, 2.6, 2.7, 2.8.2, 2.10.1, 2.10.2, 2.10.3 (history/game books)

Advanced

Use nameauth with beamer overlays to get correct name forms: Sections 2.6, 2.8, 2.8.1, 2.8.2

For building the nameauth package, see `README.md`, located with this manual, and Section 2.11.

1.4 Basic Name Concepts

We encode names in macro arguments to address multiple naming systems. Required name elements are shown in **black**; optional parts are in **red**.[1] The arguments appear in the order ⟨*FNN*⟩ ⟨*SNN*⟩ ⟨*Affix*⟩ ⟨*Alternate*⟩. Section 2.11.4 shows the obsolete syntax, which is usable but discouraged. Basic syntactic forms are:

Western Name and "Non-native" Eastern Name

Forename(s): ⟨*FNN*⟩	Surname(s): ⟨*SNN*⟩	Descriptor: ⟨*Affix*⟩
Personal name(s): *baptismal name Christian name multiple names praenomen*[2]	Family name: *of father, mother ancestor, vocation origin, region nomen, cognomen patronym*	Sobriquet/title: *Sr., Jr., III. . . notable attribute origin, region*

Alternate Name(s): ⟨*Alternate*⟩

In the body text, not the index, ⟨*Alternate*⟩ swaps with ⟨*FNN*⟩ for Western names and ⟨*Affix*⟩ for all other name categories.

"Native" Eastern Name

Family name: ⟨*SNN*⟩	Personal name: ⟨*Affix*⟩	Descriptor: ⟨*Alternate*⟩
Family/clan name	Few multiple names; multi-character okay.	Replaces ⟨*Affix*⟩ (new); personal name (obsolete)

Royal/Medieval/Ancient Name

Personal name: ⟨*SNN*⟩	Descriptor: ⟨*Affix*⟩	Descriptor: ⟨*Alternate*⟩
Given name(s)	Sobriquet/title: *Sr., Jr., III. . . notable attribute origin, region patronym*	Alternate name Replaces ⟨*Affix*⟩ (new); titles, etc. (obsolete)

[1] Compare [Mulvany, 152–82] and the *Chicago Manual of Style*.

[2] How one handles Roman names depends on index entry form; some possible suggestions are given above. See more on page 31 and following.

1.5 Basic Interface

Name color and typeface are only illustrative, not package defaults.

Using nameauth allows one to use names according to one's culture of origin. The name arguments in this section are used in many nameauth macros.

- The name form's required arguments are shown below in **black**, with optional elements shown in **red**.
- If the required argument ⟨*SNN*⟩ expands to the empty string, nameauth will generate a package error.
- Extra spaces around each argument are stripped.
- Always include all name arguments to have consistent index entries.
- `\Name` prints first uses of names long, then short thereafter. `\Name*` always creates a long form. `\FName` prints a short form in later uses.

Western Name

	Required forename(s)	Required surname(s), optional ⟨*Affix*⟩	Optional, in text only
`\Name` `\Name*` `\FName`	`[`⟨***FNN***⟩`]`	`{`⟨***SNN, Affix***⟩`}`	`[`⟨***Alternate***⟩`]`

Add braces `{}` after `{`⟨***SNN, Affix***⟩`}` if other text in brackets `[]` follows.

Within nameauth, Western names have distinct features:

- Western names must use the first optional ⟨*FNN*⟩ argument.
- They require a comma to delimit any affixes (Section 2.3.1).
- Western index entries have two general forms: ⟨*SNN*⟩, ⟨*FNN*⟩ and ⟨*SNN*⟩, ⟨*FNN*⟩, ⟨*Affix*⟩.
- They do not share control patterns (Section 2.11.5) and index entry forms with non-Western names.

Simplified Name Pattern(s):
`George!Washington`
`GeorgeS.!Patton,Jr.`

Basic Index:
Washington, George
Patton, George S., Jr.

```
\Name [George]{Washington} ..... George Washington
\Name*[George]{Washington} ..... George Washington
\Name [George]{Washington} ..... Washington
\FName[George]{Washington} ..... George

\Name [George S.]{Patton, Jr.} ..... George S. Patton Jr.
\Name*[George S.]{Patton, Jr.} ..... George S. Patton Jr.
\Name [George S.]{Patton, Jr.} ..... Patton
\FName[George S.]{Patton, Jr.} ..... George S.
```

Below, ⟨*Alternate*⟩ swaps with ⟨*FNN*⟩ only in the text. Alternate forenames print in the text with a first use, `\Name*`, or `\FName`. Use `\DropAffix`, a prefix macro (Section 1.7), to drop affixes in long name forms, but only in the text.

Simplified Name Pattern(s):
GeorgeS.!Patton,Jr.
J.D.!Rockefeller,IV
CliveStaples!Lewis

Basic Index:
Patton, George S., Jr.
Rockefeller, J.D., IV
Lewis, Clive Staples

`\Name*[George S.]{Patton, Jr.}` . . . . . George S. Patton Jr.
`\DropAffix\Name*[George S.]{Patton, Jr.}[George]` . . . . . George Patton

`\Name [J.D.]{Rockefeller, IV}` . . . . . J.D. Rockefeller IV
`\Name*[J.D.]{Rockefeller, IV}[John David]` . . . . . John David Rockefeller IV
`\DropAffix\Name*[J.D.]{Rockefeller, IV}[Jay]` . . . . . Jay Rockefeller
`\Name [J.D.]{Rockefeller, IV}[Jay]` . . . . . Rockefeller

`\Name [Clive Staples]{Lewis}` . . . . . Clive Staples Lewis
`\Name*[Clive Staples]{Lewis}[C.S.]` . . . . . C.S. Lewis
`\FName[Clive Staples]{Lewis}[Jack]` . . . . . Jack

Use the first name for sorting names instead of the initials, as with J.D. Rockefeller: `\PretagName[J.D.]{Rockefeller, IV}{Rockefeller, John D 4}` (Section 2.4.2). For alternate surnames see Sections 2.2.3, 2.3.5, 2.7.2, 2.10.3.

"Non-native" Eastern Name

	Required forename(s)	Required surname(s), no ⟨*Affix*⟩	Optional, in text only
`\Name` `\Name*` `\FName`	`[`⟨***FNN***⟩`]`	`{`⟨***SNN***⟩`}`	`[`⟨***Alternate***⟩`]`

Add braces `{}` after `{`⟨***SNN***⟩`}` if other text in brackets `[]` follows.

"Non-native" Eastern names (Section 2.3.3) have these features:

- They must use the first optional ⟨*FNN*⟩ argument.
- They cannot use affixes; one would get ⟨*FNN*⟩ ⟨*Affix*⟩ ⟨*SNN*⟩.
- Index entries have the Western form with no affix: ⟨*SNN*⟩, ⟨*FNN*⟩.
- They do not share control patterns and index entry forms with non-Western names.
- They do not work with the obsolete syntax (Section 2.11.4).

Below we start with Western name forms:

Simplified Name Pattern(s):
Hideyo!Noguchi
Frenec!MolnÃąr

Basic Index:
Noguchi, Hideyo
Molnár, Frenec

`\Name [Hideyo]{Noguchi}` . . . . . Hideyo Noguchi
`\Name*[Hideyo]{Noguchi}[Doctor]` . . . . . Doctor Noguchi
`\Name [Frenec]{Molnár}` . . . . . Frenec Molnár[3]

We use the prefix macros `\RevName` and optionally `\CapName` (Section 1.7) to print an Eastern or Hungarian name order in the text [Mulvany, 166]. We see above that these macros work in context, not arbitrarily:

Same name patterns and index entries as above.

`\CapName\RevName\Name*[Hideyo]{Noguchi}[Sensei]` . . . . NOGUCHI Sensei†
`\CapName\RevName\Name [Hideyo]{Noguchi}[Sensei]` . . . . . NOGUCHI†
`\RevName\Name*[Frenec]{Molnár}` . . . . . Molnár Frenec†
`\RevName\Name [Frenec]{Molnár}` . . . . . Molnár†

[3] With `pdflatex` and `latex`, in `Frenec!MolnÃąr` the glyphs `Ãą` correspond to `\IeC{\'a}`.

"Native" Eastern Name

	Required surname and forename	Optional, in text only

`\Name`
`\Name*`
`\FName`
	`{`⟨***SNN, Affix***⟩`}`	`[`⟨***Alternate***⟩`]`

Add braces `{}` after `{`⟨***SNN, Affix***⟩`}` if other text in brackets `[]` follows.

These features denote "native" Eastern names in nameauth:

- They must **leave empty** the ⟨*FNN*⟩ argument.
- They use instead the ⟨*SNN, Affix*⟩ arguments.
- Their index entries take the non-Western form: ⟨*SNN Affix*⟩.
- They do not share control patterns and index entry forms with both Western names and "non-native" Eastern names.

The basic forms of "native" Eastern names are shown below. Notice that the macro `\FName` does not show the personal name by default. This design choice helps to prevent one from naively causing offense:

Simplified Name Pattern(s):
`Miyazaki,Hayao`
Basic Index:
Miyazaki Hayao

```
\Name {Miyazaki, Hayao} ..... Miyazaki Hayao
\Name {Miyazaki, Hayao} ..... Miyazaki
\FName{Miyazaki, Hayao} ..... Miyazaki
```

If "Native" Eastern names are reversed, they will have Western name order in the text, but **they will retain Eastern-form index entries.**

One must use the prefix macro `\ForceFN` (Section 1.7) with `\FName` to get a personal name. ⟨*Alternate*⟩ swaps with ⟨*FNN*⟩ (in long forms and in short forms using `\ForceFN`) in the text only. ⟨*Alternate*⟩ does not work with the obsolete syntax (Section 2.11.4):

Same name patterns and index entries as above.

```
\ForceFN\FName{Miyazaki, Hayao} ..... Hayao
\CapName\Name*{Miyazaki, Hayao}[Sensei] ..... MIYAZAKI Sensei
\ForceFN\FName{Miyazaki, Hayao}[Sensei] ..... Sensei
\RevName\Name*{Miyazaki, Hayao}[Mr.] ..... Mr. Miyazaki
```

"Native" Eastern names have the same kind of macro parameters as do royal, medieval, and ancient names from Europe and the Near East (below). From the standpoint of how the macros work, one can identify "non-western" names with the form ⟨*SNN, Affix*⟩.

Yet one should not confuse similarity in form with similarity in what the names themselves mean. Even though the syntactic form of non-Western names works the same, that form has different meanings in different contexts. For one context, ⟨*SNN, Affix*⟩ refers to a family name and a personal name. For another context, ⟨*SNN, Affix*⟩ refers to a person's name and any affixes thereto.

Royal/Medieval/Ancient Name

	Required name optional ⟨*Affix*⟩	Optional, in text only
`\Name` `\Name*` `\FName`	`{`⟨***SNN, Affix***⟩`}`	`[`⟨***Alternate***⟩`]`

Add braces `{}` after `{`⟨***SNN, Affix***⟩`}` if other text in brackets `[]` follows.

These features denote royal, medieval, and ancient names in nameauth, grouped under the general rubric of "non-Western" name forms:

- They must **leave empty** the ⟨*FNN*⟩ argument.
- They use instead the ⟨*SNN, Affix*⟩ arguments or just ⟨*SNN*⟩.
- Their index entries take the non-Western forms: ⟨*SNN Affix*⟩ or ⟨*SNN*⟩.

3.0

- Names with the form ⟨*SNN, Affix*⟩ can use the ⟨*Alternate*⟩ argument.
- Names with the form ⟨*SNN*⟩ cannot use ⟨*Alternate*⟩.
- They do not share control patterns and index entry forms with both Western names and "non-native" Eastern names.

As with "native" Eastern names, `\FName` prints ⟨*SNN*⟩ unless forced otherwise by `\ForceFN`. This guards against nonsense names in the text:

Simplified Name Pattern(s):
`Elizabeth,I`
`John,Eriugena`
`Aristotle`

Basic Index:
Elizabeth I
John Eriugena
Aristotle

```
\Name {Elizabeth, I} ..... Elizabeth I
\Name {Elizabeth, I} ..... Elizabeth
\FName{Elizabeth, I} ..... Elizabeth
\ForceFN\FName{Elizabeth, I} ..... I
\ForceFN\FName{Elizabeth, I}[Good Queen Bess] ..... Good Queen Bess

\Name{John, Eriugena} ..... John Eriugena
\Name{John, Eriugena} ..... John
\ForceFN\FName{John, Eriugena} ..... Eriugena

\Name{Aristotle} ..... Aristotle
\Name{Aristotle} ..... Aristotle
```

1.6 Quick Interface

`nameauth`

To reduce typing, we replace frequently-used macros with the shorthand forms of the quick interface. Using the `nameauth` environment in the preamble guards against undefined macros. It defines a delimited macro `\<`, recalling a `tabular`:

```
\begin{nameauth}
    \< ⟨arg1⟩ & ⟨arg2⟩ & ⟨arg3⟩ & ⟨arg4⟩ >
\end{nameauth}
```

The macro `\<` uses ⟨*arg1*⟩ as a basis to create three new macros per name:

```
\⟨arg1⟩  same as: \Name [⟨arg2⟩]{⟨arg3⟩}[⟨arg4⟩]
\L⟨arg1⟩ same as: \Name*[⟨arg2⟩]{⟨arg3⟩}[⟨arg4⟩]   % L for long
\S⟨arg1⟩ same as: \FName[⟨arg2⟩]{⟨arg3⟩}[⟨arg4⟩]   % S for short
```

If either ⟨*arg1*⟩ or ⟨*arg3*⟩ are empty, or ⟨*SNN*⟩ is empty, `nameauth` will generate a package error. Forgetting the backslash, any ampersand, argument, or angle bracket will cause errors. The ⟨*Alternate*⟩ field is ⟨*arg4*⟩ (see below).

Here we do not show the obsolete syntax (Section 2.11.4). Comments are not part of the `nameauth` environment. Extra spaces around each argument are stripped. Put trailing braces `{}` after the shorthand macros if text in brackets `[]` follows. We introduce Western name forms with particles (followed by *W. part.*).

```
\begin{nameauth}
%    ⟨arg1⟩     ⟨arg2⟩          ⟨arg3⟩              ⟨arg4⟩
  \< Wash     & George        & Washington        &        >% Western
  \< Lewis    & Clive Staples & Lewis             &        >% Western
  \< Pat      & George S.     & Patton, Jr.       &        >% W. affix
  \< JRIV     & J.D.          & Rockefeller, IV   &        >% W. affix
  \< Soto     & Hernando      & de Soto           &        >% W. part.
  \< JWG      & J.W. von      & Goethe            &        >% W. part.
  \< VBuren   & Martin        & Van Buren         &        >% W. part.
  \< Noguchi  & Hideyo        & Noguchi           &        >% W. as E.
  \< Molnar   & Frenec        & Molnár            &        >% W. as E.
  \< Miyaz    &               & Miyazaki, Hayao   &        >% Eastern
  \< Yamt     &               & Yamamoto, Isoroku &        >% Eastern
  \< Eliz     &               & Elizabeth, I      &        >% Royal
  \< Aeth     &               & Æthelred, II      &        >% Royal
  \< Eriugena &               & John, Eriugena    &        >% Medieval
  \< Aris     &               & Aristotle         &        >% Mono
  \< CSL      & Clive Staples & Lewis             & C.S.   >% W. alt.
  \< MSens    &               & Miyazaki, Hayao   & Sensei >% E. alt.
\end{nameauth}
```

⟨*Alternate*⟩ Tips

Two shorthands above use ⟨*arg4*⟩: `\CSL` and `\MSens`. Their similar forms `\Lewis` and `\Miyaz` leave ⟨*arg4*⟩ empty. Here is how they are related:

Simplified Name Pattern(s):
`CliveStaples!Lewis`
`Miyazaki,Hayao`

Basic Index:
Lewis, Clive Staples
Miyazaki Hayao

- They share name control patterns (Section 2.11.5). Therefore, they have the same "first-use" and "later-use" conditions.
- Usually, one adds alternate names to shorthands with an empty ⟨*arg4*⟩:

`\LLewis[C.S.]`	C.S. Lewis	`\LCSL`	C.S. Lewis
`\LMiyaz[Sensei]`	Miyazaki Sensei	`\LMSens`	Miyazaki Sensei

- The field ⟨*arg4*⟩ contains either ⟨*Alternate*⟩ or uses the obsolete syntax. Trying to add "alternate names" to shorthands that use ⟨*arg4*⟩ fails:

`\LCSL[Jack]`	C.S. Lewis[Jack]
`\LMSens[Sensei]`	Miyazaki Sensei[Sensei]

How Quick Is Quick?

Prefix macros (Section 1.7) work with both interfaces. Here we show just a few examples showing how much typing we save with common macros:

Output	Short Form	Long Form
Washington	`\Wash`	`\Name[George]{Washington}`
George Washington	`\LWash`	`\Name*[George]{Washington}`
George	`\SWash`	`\FName[George]{Washington}`
George Washington	`\ForgetThis\Wash`	`\ForgetName[George]{Washington}` `\Name[George]{Washington}`
Washington	`\SubvertThis\Wash`	`\SubvertName[George]{Washington}` `\Name[George]{Washington}`
	`\JustIndex\Wash`	`\IndexName[George]{Washington}`

Name Variant Overview

Name color and typeface are only illustrative, not package defaults.

Below we use `\ForgetThis` (Section 2.8.1) to simulate first uses of names as needed, then proceed with subsequent uses:[4]

Simplified Name Pattern(s):
`George!Washington`
`Hernando!de~Soto`
`GeorgeS.!Patton,Jr.`
`J.D.!Rockefeller,IV`
`CliveStaples!Lewis`
`Aristotle`
`ÃĘthelred,II`
`John,Eriugena`
`Hideyo!Noguchi`
`Yamamoto,Isoroku`

Basic Index:
Washington, George
de Soto, Hernando
Patton, George S., Jr.
Rockefeller, J.D., IV
Lewis, Clive Staples
Aristotle
Æthelred II
John Eriugena
Noguchi, Hideyo
Yamamoto Isoroku

Western: (Sections 2.2.1, 2.2.2)
`\Wash` . . . George Washington
`\LWash` . . . George Washington
`\Wash` . . . Washington
`\SWash` . . . George
`\RevComma\LWash` . . . Washington, George

Particles: (Section 2.3.4)
`\Soto` . . . Hernando de Soto
`\Soto` . . . de Soto
`\CapThis\Soto` . . . De Soto

Affixes: (Section 2.3.1)
`\Pat` . . . George S. Patton Jr.
`\LPat` . . . George S. Patton Jr.
`\DropAffix\LPat` . . . George S. Patton
`\Pat` . . . Patton

Nicknames: (Section 2.2.2)
`\DropAffix\LPat[George]` George Patton
`\SPat[George]` . . . George
`\JRIV[John D.]` . . John D. Rockefeller IV
`\DropAffix\LJRIV[Jay]` . . Jay Rockefeller
`\SJRIV[Jay]` . . . Jay
`\Lewis` . . . Clive Staples Lewis
`\LLewis[Jack]` . . . Jack Lewis
`\SLewis[Jack]` . . . Jack
`\LCSL` . . . C.S. Lewis
`\SCSL` . . . C.S.

Ancient Mononym (trivial case)
`\Aris` . . . Aristotle
`\Aris` . . . Aristotle
(This is the trivial case.)

Royal and Medieval: (Sections 2.3.3, 2.3.5)
`\Aeth` . . . Æthelred II
`\Aeth` . . . Æthelred
`\LAeth[Unrædig]` . . . Æthelred Unrædig
`\Eriugena` . . . John Eriugena
`\Eriugena` . . . John

"Non-native" Eastern: (Section 2.3.3)
`\Noguchi` . . . Hideyo Noguchi
`\LNoguchi` . . . Hideyo Noguchi
`\LNoguchi[Doctor]` . . . Doctor Noguchi
`\SNoguchi` . . . Hideyo
`\RevName\LNoguchi` . . . Noguchi Hideyo†
`\CapName\RevName\LNoguchi` . . NOGUCHI Hideyo†
`\CapName\Noguchi` . . . NOGUCHI†

"Native" Eastern: (Section 2.3.3)
`\CapName\Yamt` . . . YAMAMOTO Isoroku
`\CapName\LYamt` . . . YAMAMOTO Isoroku
`\CapName\Yamt` . . . YAMAMOTO
`\RevName\LYamt` . . . Isoroku Yamamoto
`\RevName\LYamt[Admiral]` . . . Admiral Yamamoto
`\SYamt` . . . Yamamoto
`\ForceFN\SYamt` . . . Isoroku

[4] With `pdflatex` and `latex`, in `ÃĘthelred,II` the glyphs `ÃĘ` correspond to `\IeC{\AE}`.

1.7 Select Macro Overview

Macros Taking Name Arguments

Naming	⟨*prefix macros*⟩	\Name	⟨*optional **⟩	⟨*name args*⟩	
	⟨*prefix macros*⟩	\FName	⟨*optional **⟩	⟨*name args*⟩	
Page ref	\SeeAlso	\IndexName		⟨*name args*⟩	
Only cross-ref	\SeeAlso	\IndexRef		⟨*xref args*⟩	⟨*target*⟩
Prevent page ref		\ExcludeName		⟨*name args*⟩	
Enable page ref		\IncludeName	⟨*optional **⟩	⟨*name args*⟩	
Sort index		\PretagName		⟨*name args*⟩	⟨*sort key*⟩
Append idx tag		\TagName		⟨*name args*⟩	⟨*tag*⟩
Delete idx tag		\UntagName		⟨*name args*⟩	
Make text tag		\NameAddInfo		⟨*name args*⟩	⟨*tag*⟩
Show text tag		\NameQueryInfo		⟨*name args*⟩	
Delete text tag		\NameClearInfo		⟨*name args*⟩	
Delete name cs		\ForgetName		⟨*name args*⟩	
Create name cs		\SubvertName		⟨*name args*⟩	
Name cs tests		\IfMainName		⟨*name args*⟩	{⟨*y*⟩}{⟨*n*⟩}
		\IfFrontName		⟨*name args*⟩	{⟨*y*⟩}{⟨*n*⟩}
		\IfAKA		⟨*name args*⟩	{⟨*y*⟩}{⟨*n*⟩}{⟨*x*⟩}
Debugging		\ShowPattern		⟨*name args*⟩	
		\ShowIdxPageref	⟨*optional **⟩	⟨*name args*⟩	

Not shown above are \AKA, \AKA*, \PName, and \PName* (Section 2.9). These macros from the early days of nameauth have specialized arguments and issues.

Prefix Macros (One Use Per Name)

	Capitalization in the Text
\CapName	Cap entire ⟨*SNN*⟩ in body text. Overrides \CapThis.
\CapThis	Capitalize first letter of all name components in body text.
\AccentCapThis	Fallback when Unicode detection cannot be done.
	Reversing in the Text
\RevName	Reverse order of any name in body text. Overrides \RevComma
\RevComma	Reverse only Western names to ⟨*SNN*⟩, ⟨*FNN*⟩.
	Commas in the Text
\ShowComma	Add comma between ⟨*SNN*⟩ and ⟨*Affix*⟩.
\NoComma	No comma between ⟨*SNN*⟩ and ⟨*Affix*⟩. Overrides \ShowComma.
	Name Breaks in the Text
\DropAffix	Drop affix only for a long Western name reference.
\KeepAffix	Insert non-breaking space (NBSP) between ⟨*SNN*⟩, ⟨*FNN/Affix*⟩.
\KeepName	Insert NBSP between all name elements. Overrides \KeepAffix.
	Forcing Name Forms via Control Sequence
\ForgetThis	Force a first-time name use. Negates \SubvertThis.
\SubvertThis	Force a subsequent use.
	Forcing Name Forms via Boolean Flags
\ForceName	Force first-use formatting hooks.
\ForceFN	Force printing of ⟨*Affix*⟩ in non-Western short forms.
	Indexing
\SeeAlso	For \IndexName, \AKA, and \PName; make a *see also* xref.
\SkipIndex	For naming macros; do not create an index entry.
\JustIndex	For naming macros; index only (once); negated by \AKA, \PName.

More on Prefix Macros

- Prefix macros stack:
  `\CapThis\RevName\SkipIndex\Name[bar]{foo}` Foo Bar.
- The Boolean flags governed by the prefix macros are reverted after the appropriate macros produce output in the text (or index) unless the output of the naming macros is suppressed.
- Except for `\SeeAlso`, use prefix macros only before a naming macro that is designed to print output in the text.

3.5
- Use `\SeeAlso` only with `\IndexRef`, `\AKA`, and `\PName`. Otherwise it will be reset by `\IndexName` and the naming macros.

3.5
- Using `\JustIndex` will cause name form modifiers to be reset.

Macros that do not take name arguments include:

- State-changing macros with broad effects (document, section, scope).
- State-changing macros with single-use effects (prefix macros).
- Macros that alter general nameauth package behavior.
- Formatting macros.

1.8 Various Hints

In this section we make a brief foray into some technical issues that are good to keep in mind, but not overwhelming at this point. Sections 2.11.2 and 2.11.3 go into greater detail on the things that one can do to diagnose missteps and avoid errors. The point here is to keep the quick start quick.

Automatic Stripping of Spaces

Simplified Name Pattern(s):
`MartinLuther!King,Jr.`
Basic Index:
King, Martin Luther, Jr.

The nameauth package trims extra spaces **around** name arguments to prevent errors. Here, name arguments include ⟨*FNN*⟩, ⟨*SNN*⟩, ⟨*Affix*⟩, and ⟨*Alternate*⟩. For example, instead of being two different names, below we have the same name in a first, then subsequent use. We use no name formatting below in order to show this:

```
1 \Name*[Martin␣Luther]
2 {King,Jr.}\\
3 \Name*[␣␣Martin␣␣Luther␣␣]
4 {␣␣King,␣␣Jr.␣␣}
```

No spaces: |Martin Luther||King Jr.|

Spaces: |Martin Luther||King Jr.|

Using macros that expand to spaces will produce a totally different name:

```
1 \Name*[␣Martin␣Luther␣]
2 {␣King,␣Jr.␣}\\
3 \Name*[\␣Martin␣Luther\␣]
4 {\␣King,␣Jr.\␣}
```

Spaces: |Martin Luther||King Jr.|

Macros: | Martin Luther || King Jr. |

Simple Pattern:
`\MartinLuther\!\King,Jr.\`

Index: King, Martin Luther , Jr.

Yet one may have to include a non-breaking space (active character ~) after a name particle like *de* to keep the name from breaking badly (Section 2.3.4). One must use that non-breaking space consistently to avoid errors.

Full Stop Detection

Full stops appear in one's initials and in affixes like "Jr". (junior), "Sr". (senior), "d. J". (*der Jüngere*), and "d. Ä". (*der Ältere*). The naming macros and some alternate name macros (Section 2.9) check if the printed name ends with a full stop and is followed by one. They gobble the extra full stop. Below we resume formatting and pretend that we have not seen Dr. King's name yet:

Simplified Name Pattern(s):
`MartinLuther!King,Jr.`

```
This is Rev. Dr. \Name[Martin Luther]{King, Jr.}.
```
This is Rev. Dr. Martin Luther King Jr. Full stop is gobbled.

```
This is Rev. Dr. \Name[Martin Luther]{King, Jr.}.
```
This is Rev. Dr. King. Full stop is not gobbled.

```
Again we speak fully of \Name*[Martin Luther]{King, Jr.}.
```
Again we speak fully of Martin Luther King Jr. Full stop is gobbled.

```
We drop the affix: \DropAffix\Name*[Martin Luther]{King, Jr.}.
```
We drop the affix: Martin Luther King. Full stop is not gobbled.

```
His initials are \FName[Martin Luther]{King, Jr.}[M.L.].
```
His initials are M.L. Full stop is gobbled.

Caveats with Grouping

Take care when using braces and spaces with a name at the end of a sentence. Braces will change the control sequence patterns generated by name arguments. Put simply, this means that both the names and their index entries will be different and behave differently — even though they look the same (Sections 2.4.2, 2.11.5). We disable indexing for the three points below:

Simplified Name Pattern(s):
`MartinLuther!King,Jr.`

- If one encapsulates a name in braces, the punctuation detection fails:
  ```
  This is Rev. Dr. {\Name*[Martin Luther]{King, Jr.}}.
  ```
  This is Rev. Dr. Martin Luther King Jr.. Full stop is not gobbled.

  A solution encapsulates both the name and the full stop:
  ```
  This is Rev. Dr. {\Name*[Martin Luther]{King, Jr.}.}
  ```
  This is Rev. Dr. Martin Luther King Jr. Full stop is gobbled.

Simplified Name Pattern(s):
`MartinLuther!King,{Jr.}`
`MartinLuther!King,{Jr}.`
Basic Index:
King, Martin Luther, Jr.
King, Martin Luther, Jr.
(Looks identical, but not.)

- If one encapsulates ⟨*Affix*⟩ in braces, the punctuation detection fails:
  ```
  This is Rev. Dr. \Name*[Martin Luther]{King, {Jr.}}.
  ```
  This is Rev. Dr. Martin Luther King Jr.. Full stop is not gobbled.

  A solution leaves the full stop in ⟨*Affix*⟩ outside the braces:
  ```
  This is Rev. Dr. \Name*[Martin Luther]{King, {Jr}.}.
  ```
  This is Rev. Dr. Martin Luther King Jr. Full stop is gobbled.

  Yet the name patterns (Section 2.11.5) are different, creating two different names and two different index entries.

Simplified Name Pattern(s):
`MartinLuther!King,Jr.`

- A space between a name and full stop hinders punctuation detection, except with the quick interface:
  ```
  This is Rev. Dr. \Name*[Martin Luther]{King, Jr.}␣.
  ```
  This is Rev. Dr. Martin Luther King Jr. . Full stop is not gobbled.

  The solution removes the extra space:
  ```
  This is Rev. Dr. \Name*[Martin Luther]{King, Jr.}.
  ```
  This is Rev. Dr. Martin Luther King Jr. Full stop is gobbled.

Caveats with Active Characters

Variations in the use of active characters and control sequences also change name arguments, name control patterns, and index sorting. These changes can depend on the LaTeX engine being used, but often different names are just different, even if they appear the same (Section 2.4.2; cf. 2.11.6 and 2.11.7):

Simplified Name Pattern(s):
`ÃĘthelred,II` (1)
`\AEthelred,II` (2)
`Bo\"ethius` (3)
`BoÃńthius` (4)
`Bo{\"e}thius` (5)

1. `\Name*{Æthelred, II}` . . . . . Æthelred II[5]
   We have seen this name earlier.
2. `\SkipIndex\Name{\AE thelred, II}` . . . . . Æthelred II
   This is a new name that looks the same.
3. `\Name{Bo\"ethius}` . . . . . Boëthius
   We introduce this new name.
4. `\SkipIndex\Name{Boëthius}` . . . . . Boëthius[6]
   This is a different name that looks the same.
5. `\SkipIndex\Name{Bo{\"e}thius}` . . . . . Boëthius
   This is a third, different name that looks the same.

Formatting Initials

This is a thorny topic. Some publishers are dead-set on having a space between initials. Many designers find that practice to be inelegant at best. Robert Bringhurst wisely advises one to omit spaces between initials.[7]

Yet fighting with one's editor will be a lost cause unless one already has sufficient *gravitas*. If a style guide requires spaces, try thin spaces. Use `\PretagName` to sort those names (Section 2.4.2). Below we use no formatting:

```
\PretagName[E.\,B.]{White}%
  {White, Elwyn}
\begin{nameauth}
  \< White & E.\,B. & White & >
\end{nameauth}
```

`\White`	E. B. White
Normal text:	E. B. White

Multicultural Hyphenation

Names can be hyphenated to reflect their cultural and linguistic origins. With nameauth, one can use either optional hyphens or the babel/polyglossia packages to handle such names. Below we offer a simplified example without alternate formatting (Section 2.7):

```
\newcommand\de[1]{\foreignlanguage{ngerman}{#1}}
 % or polyglossia: \newcommand\de[1]{\textgerman{#1}}
\NameAddInfo[John]{\de{Strietelmeier}}%
  {a professor at Valparaiso University}
\begin{nameauth}
  \< Striet & John & \de{Strietelmeier} & >
\end{nameauth}
\PretagName[John]{\de{Strietelmeier}}{Strietelmeier, John}
```

[5] With `pdflatex` and `latex`, in `ÃĘthelred,II` the glyphs `ÃĘ` correspond to `\IeC{\AE}`.
[6] With `pdflatex` and `latex`, in `BoÃńthius` the glyphs `Ãń` correspond to `\IeC{\"e}`.
[7] Robert Bringhurst, *Elements of Typographic Style* 3.2 ed. (Point Roberts, Washington: Hartley & Marks, 2008.)

Now we demonstrate three different ways of engaging this problem. In the first example we use the default hyphenation. We omit this version from the index. One might think that the name were pronounced "stree-et-el-mai-er":

Simplified Name Pattern(s):
`John!Strietelmeier`

Not fixed:
In English, some names come from other cultures. These names, like John Strietelmeier, `\SkipIndex\Name[John]{Strietelmeier},` can break badly.

The next example uses discretionary hyphens. It is a different name from the one above and one must be consistent with the discretionary hyphens. We also omit this version from the index:

Simplified Name Pattern(s):
`John!Strie\-tel\-meier`

Fixed with discretionary hyphens:

In English, some names come from other cultures. These names, like John Strietelmeier, `\SkipIndex\Name[John]{Strie\-tel\-meier},` could break badly.

Finally we use what may be the best general solution, the babel or polyglossia packages. Since the leading element of ⟨*SNN*⟩ is a macro, using `\CapThis` would halt LaTeX with errors unless we used alternate formatting (Section 2.7):

Simplified Name Pattern(s):
`John!\de{Strietelmeier}`

Fixed with language packages:

In English, some names come from other cultures. These names, like John Strietelmeier, `\Striet,` could break badly. Strietelmeier was at Valparaiso University.

Obsolete Syntax Caution

We moved the discussion of the obsolete syntax to Section 2.11.4 because, as this package matures, we do not expect people to use it much anymore. There are more advantages to using the current syntax.

Simplified Name Pattern(s):
`Henry,VIII` (1–2)
`Henry!VIII` (3)
Basic Index:
Henry VIII (1–2)
VIII, Henry (3)

1. Only the newer syntax permits variants: `\Name*{Henry, VIII}[Tudor]` Henry Tudor. The new syntax is preferred.
2. A proper form for the old syntax is `\Name*{Henry}[VIII]`: Henry VIII. Both old and new share name patterns (Section 2.11.5).
3. `\Name[Henry]{VIII}` is a malformed Western name: "Henry VIII" and "VIII". Likewise `\Name[Henry]{VIII}[Tudor]`: "Tudor VIII" and "VIII". Both have the incorrect index entry "VIII, Henry".

Back to Section 1.3

'Tis but thy name that is my enemy;...
What's in a name? That which we call a rose
By any other name would smell as sweet;
So Romeo would, were he not Romeo call'd,
Retain that dear perfection which he owes
Without that title. Romeo, doff thy name,
And for that name which is no part of thee
Take all myself.

—William Shakespeare, *Romeo and Juliet*, Act II, Scene II

2 Detailed Usage

2.1 Package Options

One includes the nameauth package thus:

```
\usepackage[⟨option1⟩,⟨option2⟩,...,⟨optionn⟩]{nameauth}
```

The options have no required order. Still, we discuss them from the general to the specific, as the headings below indicate. In the listings below, **implicit default options are boldface and need not be invoked by the user.** Non-default options are in red and must be invoked explicitly.

Choosing Features

Choose Formatting System

mainmatter	**Start with "main-matter names" and formatting hooks** (see also page 19).
frontmatter	Start with "front-matter names" and hooks until \NamesActive starts the main system.
alwaysformat	Use only respective "first use" formatting hooks.
formatAKA	Format the first use of a name with \AKA like the first use of a name with \Name.

The mainmatter and frontmatter options enable two respectively independent systems of name use and formatting. See Section 2.6.

The alwaysformat option forces "first use" hooks globally in both naming systems. Its use is limited in current versions of nameauth.

3.1 The formatAKA option permits \AKA to use the "first use" formatting hooks. This enables \ForceName to trigger those hooks at will (Section 2.9). Otherwise \AKA only uses "subsequent use" formatting hooks.

Enable/Disable Indexing

index	**Create index entries in place with names.**
noindex	Suppress indexing of names.

These options and related macros apply only to the nameauth package macros. The default index option enables name indexing right away. The noindex option disables the indexing of names until \IndexActive enables it. **Caution:** using noindex and \IndexInactive prevents index tags until you call \IndexActive, as explained also in Section 2.4.1. For indexing feature priority, see page 20.

Enable/Disable Index Sorting

pretag	**Create sort keys used with makeindex.**
nopretag	Do not create sort keys.

The default allows \PretagName to create sort keys used with makeindex. The nopretag option disables the sorting mechanism and causes \PretagName only to emit warnings. That is designed for cases that use different sorting methods, such as xindy. See Section 2.4.2.

Enable "Global" Decision Paths

`globaltest` Do not put name decision paths in a local scope.

The default puts the decision paths of `\IfMainName`, etc., into groups with local scope (Section 2.8.2). This option removes that scoping.

Enable Package Warnings

`verbose` Show more diagnostic warnings.

The default suppresses all but the most essential package warnings. Increasing the warnings may help to debug index page entries, cross-references, and exclusions.

Choose Version Compatibility

Using these options will increase the chance of undocumented behavior. They are included only for the sake of backward compatibility.

`oldAKA` Force `\AKA*` to act like it did before version 3.0, instead of like `\FName`.

`oldreset` Reset per-use name flags locally; let `\ForgetThis` and `\SubvertThis` pass through `\AKA` (pre-v3.3). Let `\SeeAlso` pass through `\IndexName` and other macros. Keep `\IndexName` and `\IndexRef` from resetting `\SkipIndex` (pre-version 3.5).

`oldpass` When `\Justindex` is called, allow long/short flags to pass through, as before version 3.3.

`oldtoks` Token registers holding the arguments of the last-used name are set locally, as before version 3.5.

`oldsee` Allow lax handling of *see* references that are extant names, as before version 3.5.

Previously, local scope for Boolean flags related to the prefix macros and long/short name forms could produce unexpected results, but that could hide the problems with some flags not being reset by `\AKA`, `\AKA*`, and the use of `\JustIndex`. Global name token registers are preferable, as is the newer, stricter control over *see* references related to index page entries.

nameauth version	compatibility options to approximate:
2.6	`oldAKA,oldpass,oldreset,oldtoks,oldsee`
3.0–3.2	`oldpass,oldreset,oldtoks,oldsee`
3.3–3.4	`oldreset,oldtoks,oldsee`

Affect the Syntax of Names

Show/Hide Affix Commas

`nocomma` **Suppress commas between surnames and affixes, following the *Chicago Manual of Style* and other conventions.**

`comma` Retain commas between surnames and affixes.

These options do not affect the index. On comma macro priority, see page 20. If you use **modern standards**, choose the default `nocomma` option to get, e.g., James Earl Carter Jr. If you need to adopt **older standards** that use commas between surnames and affixes, you have two choices:

1. The `comma` option globally produces, e.g., James Earl Carter, Jr.
2. Section 2.3.1 shows how one can use `\ShowComma` with the `nocomma` option and `\NoComma` with the `comma` option to get per-name results.

Capitalize Entire Surnames

`normalcaps`	**Do not perform any special capitalization.**
`allcaps`	Capitalize entire surnames, e.g., romanized Eastern names, throughout the document.

These options do not affect the index. See Section 2.3.3 for finer control. To capitalize names in the index, use all caps or alternate formatting (Section 2.7). On capitalization feature priority, see page 20.

Reverse Name Order

`notreversed`	**Print names in the order specified by `\Name` and the other macros.**
`allreversed`	Print all name forms in "smart" reverse order.
`allrevcomma`	Print all names in "Surname, Forenames" order, meant for Western names.

These options do not affect the index and are mutually exclusive. See also Sections 2.3.2 and 2.3.3. Regarding which of these features overrides the other, see page 20. So-called "last-comma-first" lists of names via `allrevcomma` and the reversing macros `\ReverseCommaActive` and `\RevComma` (Section 2.3.2) are **not** the same as the `comma` option. They only affect Western names.

Typographic Post-Processing

Formatting Attributes

`noformat`	**Do not define a default format.**
`smallcaps`	First use of a main-matter name in small caps.
`italic`	First use of a main-matter name in italic.
`boldface`	First use of a main-matter name in boldface.

The options above are "quick" definitions of `\NamesFormat` based on English typography.[8] The default is no formatting, the overwhelming user preference.

The following macros are formatting hooks that do "typographic post-processing" of names in the text. Originally, `\NamesFormat` was the only such hook, which resulted in the organic development of the names of these macros. This development reflects the use of two naming systems, one for main-matter text (default) and one for front-matter text.

Unlike alternate formatting, the hooks do not affect the index. Sections 2.6, 2.10.1, 2.10.2, and 2.10.3 explain these hooks and their redefinition in greater detail. Changes to the formatting hooks apply within the scope where they are made:

[8] For the old default, use the `smallcaps` option. See also Robert Bringhurst, *The Elements of Typographic Style*, version 3.2 (Point Roberts, Washington: Hartley & Marks, 2008), 53–60.

- `\NamesFormat` formats first uses of main-matter names.
- `\MainNameHook` formats subsequent uses of main-matter names.
- `\FrontNamesFormat` formats first uses of front-matter names.
- `\FrontNameHook` formats subsequent uses of front-matter names.

Section 2.9 discusses how `\AKA` does not respect these formatting systems and uses the hooks differently. To avoid using the `formatAKA` option and `\ForceName` with `\AKA`, Section 2.4.1 shows how to use `\IndexRef` and `\Name` instead.

Alternate or Continental Formatting

Alternate Formatting

`altformat` Make available the alternate formatting framework from the start of the document. Activate formatting by default.

3.1 A built-in framework provides an alternate formatting mechanism that can be used for "Continental" formatting that one sees in German, French, and so on. Continental standards often format surnames only, both in the text and in the index. Section 2.7 introduces the topic and should be sufficient for most users, while Section 2.10.3 goes into greater detail.

Previous methods that produced Continental formatting were more complex than the current ones. Yet these older solutions still should work, as long as one uses the `altformat` option and related macros.

Feature Priority

Below we see the relative priority of package options and macros, with darker rows showing lower priority. Within a column, high priority can override low priority. Thus, `\IndexInactive` overrides `\JustIndex`, which overrides `\SkipIndex`.

Indexing	Capitalization	Reversing	Name Forms, Commas, Breaks
index	normalcaps	notreversed	\ForgetThis
noindex	allcaps	allreversed	\DropAffix
\IndexActive	\AllCapsInactive	\ReverseActive	
\IndexInactive	\AllCapsActive	\ReverseInactive	
\JustIndex	\CapName	\RevName	\SubvertThis \ForceName \NoComma
\SkipIndex	\AccentCapThis	allrevcomma \RevCommaActive \RevCommaInactive	\KeepName \ForceFN \ShowComma
\SeeAlso	\CapThis	\RevComma	\KeepAffix

Back to Section 1.3

2.2 Naming Macros

Name color and typeface are only illustrative, not package defaults.

In this manual we modify the formatting hooks to show first and later name uses, forcing such uses as needed (Sections 2.6–2.8.1). All naming macros create index entries before and after a name for when a name straddles a page break.

2.2.1 \Name and \Name*

\Name
\Name*

\Name displays and indexes names. It always prints the ⟨*SNN*⟩ argument. \Name prints the full name at the first occurrence, then usually just the ⟨*SNN*⟩ argument thereafter. \Name* always prints the full name:

```
\Name [⟨FNN⟩]{⟨SNN, Affix⟩}[⟨Alternate⟩]
\Name*[⟨FNN⟩]{⟨SNN, Affix⟩}[⟨Alternate⟩]
```

In the body text, not the index, the ⟨*Alternate*⟩ argument replaces either ⟨*FNN*⟩ or, if ⟨*FNN*⟩ is absent, ⟨*Affix*⟩. If both ⟨*FNN*⟩ and ⟨*Affix*⟩ are absent when ⟨*Alternate*⟩ is present, then the obsolete syntax is used (Section 2.11.4).

```
1 \begin{nameauth}
2   \< Einstein  & Albert & Einstein        & >
3   \< Cicero    & M.T.   & Cicero          & >
4   \< Confucius &        & Confucius       & >
5   \< Miyaz     &        & Miyazaki, Hayao & >
6   \< Eliz      &        & Elizabeth, I    & >
7 \end{nameauth}
```

Simplified Name Pattern(s):
Albert!Einstein
M.T.!Cicero
Confucius
Miyazaki,Hayao
Elizabeth,I

Basic Index:
Einstein, Albert
Cicero, M.T.
Confucius
Miyazaki Hayao
Elizabeth I

\Name [Albert]{Einstein} or \Einstein	Albert Einstein
\Name*[Albert]{Einstein} or \LEinstein	Albert Einstein
\Name [Albert]{Einstein} or \Einstein	Einstein
\Name [M.T.]{Cicero} or \Cicero	M.T. Cicero
\Name*[M.T.]{Cicero}[Marcus Tullius]	Marcus Tullius Cicero
\Name [M.T.]{Cicero} or \Cicero	Cicero
\Name {Confucius}, \Confucius	Confucius
Same for all variants; no ⟨*Affix*⟩ or ⟨*Alternate*⟩.	Confucius
\Name {Miyazaki, Hayao} or \Miyaz	Miyazaki Hayao
\Name*{Miyazaki, Hayao}[Sensei]	Miyazaki Sensei
\Name {Miyazaki, Hayao} or \Miyaz	Miyazaki
\Name {Elizabeth, I} or \Eliz	Elizabeth I
\Name*{Elizabeth, I} or \LEliz	Elizabeth I
\Name {Elizabeth, I} or \Eliz	Elizabeth

When using the quick interface, the preferred way to get alternate names is \LCicero[Marcus Tullius] and \LMiyaz[Sensei]: Marcus Tullius Cicero and Miyazaki Sensei. The alternate forename is not shown in subsequent short name references e.g., \Cicero[Marcus Tullius] Cicero. Remember the following:

No:	\LEinstein	[said]...	said Einstein...
No:	\Einstein	[said]...	Einstein...
Yes:	\LEinstein{}	[said]...	Albert Einstein [said]...
Yes:	\Einstein{}	[said]...	Einstein [said]...

2.2.2 Forenames: \FName

\FName
\FName*

\FName and its synonym \FName* print personal names only in subsequent name uses. They print full names for first uses. These synonyms let one add an F either to \Name or \Name* to get the same effect:

```
\FName [⟨FNN⟩]{⟨SNN, Affix⟩}[⟨Alternate⟩]
\FName*[⟨FNN⟩]{⟨SNN, Affix⟩}[⟨Alternate⟩]
```

\ForceFN
3.0

These macros work with both Eastern and Western names, but to get an Eastern personal name, one must precede these macros with \ForceFN. This was designed to discourage one from being too familiar and causing offense. See also Sections 2.3.4 and 2.8.1 on how to vary some of the forms below:

Simplified Name Pattern(s):
Albert!Einstein
M.T.!Cicero
Confucius
Miyazaki,Hayao
Elizabeth,I

\FName[Albert]{Einstein} or \SEinstein	Albert
\FName[M.T.]{Cicero}[Marcus Tullius] or \SCicero[Marcus Tullius]	Marcus Tullius
\FName{Confucius} or \SConfucius	Confucius
\FName{Miyazaki, Hayao} or \SMiyaz	Miyazaki
\ForceFN\FName{Miyazaki, Hayao} or \ForceFN\SMiyaz	Hayao
\ForceFN\FName{Miyazaki, Hayao}[Sensei] or \ForceFN\SMiyaz[Sensei]	Sensei
\FName{Elizabeth, I} or \SEliz	Elizabeth
\ForceFN\SEliz[Good Queen Bess]	Good Queen Bess

The ⟨*Alternate*⟩ argument replaces forenames in the text, which strongly shapes the use of \FName. We apply page 10 to forenames:

```
1 \begin{nameauth}
2   \< Lewis & Clive Staples & Lewis             &        >
3   \< CSL   & Clive Staples & Lewis             & C.S.   >
4   \< Miyaz &               & Miyazaki, Hayao   &        >
5   \< MSens &               & Miyazaki, Hayao   & Sensei >
6 \end{nameauth}
```

Simplified Name Pattern(s):
CliveStaples!Lewis
Miyazaki,Hayao
Basic Index:
Lewis, Clive Staples
Miyazaki Hayao

- They share name control patterns (Section 2.11.5). Therefore, they have the same "first-use" and "later-use" conditions.
- Usually, one adds alternate names to shorthands with an empty ⟨*arg4*⟩:

\SLewis[C.S.]	C.S.	\SCSL	C.S.
\SMiyaz[Sensei]	Miyazaki	\SMSens	Miyazaki
\ForceFN\SMiyaz[Sensei]	Sensei	\ForceFN\SMSens	Sensei

- Trying to add "alternate names" to shorthands that use ⟨*arg4*⟩ fails:

\SCSL[Jack]	C.S.[Jack]
\ForceFN\SMSens[Sensei]	Sensei[Sensei]

Back to Section 1.3

2.2.3 Variant Names

3.1 This section explains how to manage more complicated variants, which gives one the skills needed to implement a name authority. We draw from Sections 2.4.1, 2.4.2, 2.6, and 2.8.1. One might want to consult those sections also.

Variant forenames

We begin with the easier kind of variant names, namely, variant forenames indexed under a canonical name entry:

```
\begin{nameauth}
  \< Tyson & Mike & Tyson & >
  \< Iron & Mike & Tyson & Iron Mike >
\end{nameauth}
```

Simplified Name Pattern(s):
`Mike!Tyson`
Basic Index:
Tyson, Mike

Below, all variants have the same pattern and index entry because they are based on the same name form:

\Iron	Iron Mike Tyson	\LTyson[Iron Mike]	Iron Mike Tyson
\LIron	Iron Mike Tyson	\LTyson	Mike Tyson
\SIron	Iron Mike	\STyson	Mike

Since Iron Mike Tyson is indexed as "Tyson, Mike" throughout the document, we can use `\IndexRef{Iron Mike}{Tyson, Mike}`, which produces no output in the text. Thus we get "Iron Mike *see* Tyson, Mike" as a cross-reference in the index.

Variant surnames

Variant family names are more complicated. The following method avoids using macros in name arguments (cf. Page 31) to get decent results:

```
\begin{nameauth}
  \< DuBois    & W.E.B. & Du~Bois & >
  \< AltDuBois & W.E.B. & DuBois  & >
\end{nameauth}
\PretagName[W.E.B.]{Du~Bois}{Dubois, William}%
\PretagName[W.E.B.]{DuBois}{Dubois, William}
```

Simplified Name Pattern(s):
`W.E.B.!Du~Bois`
`W.E.B.!DuBois`
Basic Index:
Du Bois, W.E.B.
DuBois, W.E.B.

- We decide the canonical name form: `\DuBois` W.E.B. Du Bois.
- Both `\Name[W.E.B.]{Du Bois}` and `\Name[W.E.B.]{DuBois}` have the same pattern: `W.E.B.!DuBois` (Section 2.11.5).
- Here we use `\Name[W.E.B.]{Du~Bois}` to avoid bad breaks.
- The sort key for both names is `{Dubois, William}`. Had we kept the space, the name would be sorted before `dual` (Section 2.4.2). One may have to spell out a name when sorting its initials.
- Instead of using `\SkipIndex\AltDuBois` many times, we create a cross-reference in the preamble so that no page entry for the alternate form will occur in the index:

  ```
  \IndexRef[W.E.B.]{DuBois}{Du Bois, W.E.B.}
  ```

- We can use `\JustIndex\DuBois\AltDuBois` W.E.B. DuBois, keep full stop detection, and check if the name straddles a page break in order to append `\JustIndex\DuBois` if needed.
- If we create a macro like the one below, we lose full stop detection but then we do not have to check if the name straddles a page break. Normally, the name macros create two index entries each in order to handle this issue automatically:

  ```
  \newcommand\NewDuBois%
    {\JustIndex\DuBois\AltDubois\JustIndex\DuBois}
  ```

Example Name Authority

Below are a couple of names from a name authority created for a translation of *De Diaconis et Diaconissis Veteris Ecclesiae Liber Commentarius* by Caspar Ziegler, of which the present author was the editor.[9]

Constructing that name authority was a challenge. In order to get the names right — the deceased translator unfortunately had left them in abbreviated Latin, as well as leaving many place names in Latin or translating them incorrectly — the present author used the following sources, among several others:

- CERL Thesaurus: https://data.cerl.org/thesaurus/_search
- Virtual International Authority File: http://viaf.org/
- EDIT16: http://edit16.iccu.sbn.it/web_iccu/ehome.htm
- WorldCat: https://www.worldcat.org/
- An older version of Graesse, *Orbis Latinus*: http://www.columbia.edu/acis/ets/Graesse/contents.html

This author used the vernacular forms as canonical, with the Latin versions as alternates. I translated all the place-names.

Below we have candidates for sorting with `\PretagName` (Section 2.4.2) and potential use of `\CapThis` (Section 2.3.4). **After** using `\IndexRef` with a particular name, using `\Name` with that same name will not create a page reference from that point onward (Section 2.4.1). If one were to use the alternate name **before** using `\IndexRef`, then `\SeeAlso\IndexRef` would be used after all name references.

Simplified Name Pattern(s):
`Jacques!De~Pamele`
`Jacobus!Pamelius`
`Giovanni!d'Andrea`
`Ioannes!Andreae`

Basic Index:
De Pamele, Jacques
Pamelius, Jacobus
d'Andrea, Giovanni
Andreae, Ioannes

```
\PretagName[Jacques]{De~Pamele}{Depamele, Jacques}
\Name[Jacques]{De~Pamele}[Jacques de~Joigny]
\IndexRef[Jacobus]{Pamelius}{De~Pamele, Jacques}
\Name[Jacobus]{Pamelius}

\PretagName[Giovanni]{d'Andrea}{Dandrea, Giovanni}
\Name[Giovanni]{d'Andrea}
\IndexRef[Ioannes]{Andreae}{d'Andrea, Giovanni}
\Name[Ioannes]{Andreae}
```

Canonical Name	Alternate Name
Jacques de Joigny De Pamele	Jacobus Pamelius
Giovanni d'Andrea	Ioannes Andreae

D'Andrea `\CapThis\Name[Giovanni]{d'Andrea}` can be used at the beginning of a sentence. `\Name[Jacques]{De~Pamele}` gives De Pamele.

Back to Section 1.3

[9] The book, *The Diaconate of the Ancient and Medieval Church*, originally was typeset using LaTeX, but had to be converted to a different format. Using LaTeX, the present author has published Charles P. Schaum and Albert B. Collver III, *Breath of God, Yet Work of Man: Scripture, Philosophy, Dialogue, and Conflict* (St. Louis: Concordia Publishing House, 2019).

2.3 Language Topics

Here we focus on specific issues that are related to parts of names used differently in various cultures and kinds of names related to specific cultures. Comma-delimited affixes ⟨*SNN, Affix*⟩ are a key concept here. Advanced topics in this section draw on Sections 2.4, 2.6, 2.7, and 2.8.

2.3.1 Affixes Require Commas

Simplified Name Pattern(s):
`Oskar!Hammerstein,II`
`Louis,XIV`
`Sun,Yat-sen`
Basic Index:
Hammerstein, Oskar, II
Louis XIV
Sun Yat-sen

A comma separates a Western surname and affix; a "native" Eastern family name and personal name; and a royal, medieval, or ancient name and affix. To avoid errors, one must treat ⟨*SNN, Affix*⟩ as two separate arguments (Section 2.7). Spaces around the comma are ignored.

`\Name[Oskar]{Hammerstein, II}`	Oskar Hammerstein II
`\Name[Oskar]{Hammerstein, II}`	Hammerstein
`\Name{Louis, XIV}`	Louis XIV
`\Name{Louis, XIV}`	Louis
`\Name{Sun, Yat-sen}`	Sun Yat-sen
`\Name{Sun, Yat-sen}`	Sun

Simplified Name Pattern(s):
`Oskar!Hammerstein`

Western names with affixes must use ⟨*SNN, Affix*⟩, never the obsolete syntax, which is meant for non-Western names and is discouraged. We get II Hammerstein and a bad index entry from, e.g., `\SkipIndex\Name[Oskar]{Hammerstein}[II]`.

`\KeepAffix`

In the text only, `\KeepAffix` turns the space **between** ⟨*SNN*⟩ and ⟨*Affix*⟩ into a non-breaking space if both ⟨*SNN*⟩ and ⟨*Affix*⟩ are displayed. This macro works with all name types, even with the obsolete syntax.

`\KeepName`
3.1

In the text only, `\KeepName` turns all spaces **between** name elements ⟨*FNN*⟩, ⟨*SNN*⟩, and ⟨*Affix*⟩ into non-breaking spaces if those elements are displayed. This macro does not alter spaces **within** name elements that have multiple names like French or German forenames and Spanish surnames. As above, this macro works with all name types, even with the obsolete syntax.

`\DropAffix`
3.0

Preceding the naming macros with `\DropAffix` will suppress an affix only in a Western name. `\DropAffix\Name*[Oskar]{Hammerstein, II}` produces "Oskar Hammerstein". This macro does not affect non-Western names.

With non-Western names, the ⟨*Affix*⟩ in the ⟨*SNN, Affix*⟩ pair drops automatically in the text for subsequent uses, making `\DropAffix` redundant. We see that above in the case of Louis XIV, who becomes Louis.

`\ShowComma`
`\NoComma`

`\ShowComma` forces a comma between a Western name and its affix. It works like the `comma` option on a per-name basis, and only in the text. One uses `\ShowComma` with older publication styles that separate a Western name and affix with a comma. `\NoComma` works like the `nocomma` option in the body text on a per-name basis. Neither of these macros affect the use of `\RevComma`, which always prints a comma.

`\ShowComma\LPat`	George S. Patton, Jr.
`\NoComma\LPat`	George S. Patton Jr.

Back to Section 1.3

2.3.2 Listing Western names by Surname

\ReverseCommaActive
\ReverseCommaInactive
\RevComma
3.0

In addition to the options for reversed comma listing (Section 2.1), the macros `\ReverseCommaActive` and `\ReverseCommaInactive` function the same way with blocks of text. They all override `\RevComma`. These all reorder only long Western and "non-native" Eastern name forms. The first two are broad toggles, while the third works on a per-name basis.

Simplified Name Pattern(s):
`Oskar!Hammerstein,II`
`Hideyo!Noguchi`
`ÃĘthelred,II`
`Sun,Yat-sen`
`Confucius`

Martin Van Buren	Van Buren, Martin	change
Oskar Hammerstein II	Hammerstein II, Oskar	change
Hideyo Noguchi	Noguchi, Hideyo†	change
Æthelred II	Æthelred II	no change
Sun Yat-sen	Sun Yat-sen	no change
Confucius	Confucius	no change

\global

Both `\ReverseCommaActive` and `\ReverseCommaInactive` can be used either as a pair or singly within a local scope. Use `\global` to force a global effect.

Back to Section 1.3

2.3.3 Eastern Names

All non-Western name forms using the nameauth macros omit the first optional argument. Yet the reversing macros can make Western names have Eastern name order, but only in the text, not in the index.

"Non-native"

One produces a "non-native" Eastern name in the text by reversing a Western name without ⟨*Affix*⟩ using `\RevName`, e.g.:

`\RevName\Name[`⟨*FNN*⟩`]{`⟨*SNN*⟩`}[`⟨*Alternate*⟩`]`

The index entry of this name form looks like ⟨*SNN*⟩, ⟨*FNN*⟩ (including the comma). This is a Western index entry. This form is used also for Hungarian names, e.g.: `\RevName\Name[Frenec]{Molnár}` Molnár Frenec†, Molnár†.

"Native"

In contrast, "native" Eastern names use either comma-delimited syntax or the obsolete syntax (Section 2.11.4). They have index entries appropriate to Eastern names: ⟨*SNN*⟩ ⟨*Affix*⟩ (no comma). The current syntax permits alternate names; the obsolete does not. These forms work also with ancient and medieval names:

`\Name{`⟨*SNN, Affix*⟩`}[`⟨*Alternate*⟩`]` % *new syntax*

Avoid error

People can make mistakes that these forms help one to avoid. For example, in an otherwise excellent German-language multi-volume history text, one finds the incorrect, Western-form index entry "Yat-Sen, Sun". It should be "Sun Yat-sen".[10] The macro `\Name*{Sun, Yat-sen}` Sun Yat-sen ensures the correct entry by using the correct form. The goal is to promote cross-cultural sensitivity.

[10] See Immanuel Geiss, *Personen: Die biographische Dimension der Weltgeschichte*, Geschichte Griffbereit vol. 2 (Munich: Wissen Media Verlag, 2002), 720.

\ReverseActive
\ReverseInactive
\RevName

In addition to the options for reversing (Section 2.1), \ReverseActive and \ReverseInactive reverse name order for blocks of text. These all override the use of \RevName, which reverses once per name. These macros do not affect the index. They work also with \AKA and friends. Reversing only affects long name forms. "Non-native" forms are shown with a dagger (†):

Simplified Name Pattern(s):
Hideyo!Noguchi
Miyazaki,Hayao
Basic Index:
Noguchi, Hideyo
Miyazaki Hayao

	unchanged	\RevName
\LNoguchi	Hideyo Noguchi	Noguchi Hideyo†
\LNoguchi[Doctor]	Doctor Noguchi	———
\LNoguchi[Sensei]	———	Noguchi Sensei†
\Noguchi	Noguchi	Noguchi†
\SNoguchi	Hideyo	Hideyo†
\LMiyaz	Miyazaki Hayao	Hayao Miyazaki
\LMiyaz[Mr.]	———	Mr. Miyazaki
\LMiyaz[Sensei]	Miyazaki Sensei	———
\Miyaz	Miyazaki	Miyazaki
\SMiyaz	Miyazaki	Miyazaki
\ForceFN\SMiyaz	Hayao	Hayao

\global

Both \ReverseActive and \ReverseInactive can be used either as a pair or singly within an explicitly local scope. Use \global to force a global effect.

\AllCapsActive
\AllCapsInactive
\CapName

In addition to the options for capitalizing (Section 2.1), \AllCapsActive and \AllCapsInactive work for blocks of text. All override \CapName, which works once per name. These capitalize ⟨*SNN*⟩ in the body text only. They also work with \AKA and friends. For caps in the text and index see Sections 2.7 and 2.10.3. We show "non-native" Eastern forms with a dagger (†):

	\CapName only	\CapName\RevName
\LNoguchi	Hideyo NOGUCHI	NOGUCHI Hideyo†
\LMiyaz	MIYAZAKI Hayao	Hayao MIYAZAKI

\global

Both \AllCapsActive and \AllCapsInactive can be used either as a pair or singly within an explicitly local scope. Use \global to force a global effect.

Back to Section 1.3

2.3.4 Particles in Names

Particles in names have specific rules:

- English use of *de*, *de la*, *d'*, *von*, *van*, and *ten* often keeps them with the surname with varied capitalization.
- *Le*, *La*, and *L'* always are capitalized unless preceded by *de*.[11]
- Modern Romance languages keep particles with the surname.
- German and medieval Romance languages put particles with forenames.

[11] According to [Mulvany, 152–82] and the *Chicago Manual of Style*.

Simplified Name Pattern(s):
Martin!VanBuren
Hernando!de~Soto
J.W.von!Goethe
Basic Index:
Van Buren, Martin
de Soto, Hernando
Goethe, J.W. von

Macro	Body Text	Index
\ForgetThis\VBuren	Martin Van Buren	Van Buren, Martin
\VBuren	Van Buren	Van Buren, Martin
\ForgetThis\Soto	Hernando de Soto	de Soto, Hernando
\CapThis\Soto	De Soto	de Soto, Hernando
\ForgetThis\JWG	J.W. von Goethe	Goethe, J.W. von
\JWG	Goethe	Goethe, J.W. von

A few tips

We recommend inserting a tilde (active character for a non-breaking space) or \nobreakspace between some particles and names to prevent bad breaks, sorting them with \PretagName (Section 2.4.2). Some particles look similar: *L'* (L+apostrophe) and *d'* (d+apostrophe) are two separate glyphs each. In contrast, *Ľ* (L+caron) and *ď* (d+caron) are one Unicode glyph each (Section 2.11.6).

\CapThis

In English and modern Romance languages, e.g., Hernando de Soto shows that these particles go in the ⟨*SNN*⟩ argument of \Name: de Soto. When the particle appears at the beginning of a sentence, one must capitalize it:

```
\CapThis\Soto\ De Soto was a famous Spanish explorer in North America.
```

3.2

\CapName overrides the ⟨*SNN*⟩ created by \CapThis. \CapThis should work with all of the Unicode characters available in the T1 encoding (its mechanism is explained in Section 2.11.6 and on page 96). For a broader set of Unicode characters, consider using xelatex and lualatex.

Surname variants

For another example, we mention poet e.e. cummings. One can have formatted name caps and inflections. The easiest way to do that is from Section 2.4.1:

Simplified Name Pattern(s):
e.e.!cummings
e.e.!cummings's
Basic Index:
cummings, e.e.

```
1 \ExcludeName[e.e.]{cummings's}\IndexName[e.e.]{cummings}
2 \SubvertThis\CapThis\Name[e.e.]{cummings's} motif of the
3 goat-footed balloon man has underlying sexual themes that
4 nevertheless have a childish facade''.
```

E.e. Cummings's motif of the goat-footed balloon man has underlying sexual themes that nevertheless have a childish facade".

One must use \SubvertThis only for the first use to avoid "E.e. Cummings's"; all name elements are capped with \CapThis. Using \ExcludeName keeps one from having to use \SkipIndex every time. See also Section 2.2.3.

Section 2.7 explains how to use \CapThis with alternate formatting when using macros in name arguments. Page 56 describes how automation lends itself to Continental formats (French, German, etc.) and grammatical inflections.

\AccentCapThis
3.0

If one uses this package on a system that does not support Unicode, one can use \AccentCapThis instead of \CapThis to handle active initial characters. Otherwise, one should not need to use \AccentCapThis.

Back to Section 1.3

"Rose is a rose is a rose is a rose"
—Gertrude Stein, "Sacred Emily" in *Geography and Plays*

2.3.5 Medieval, Ancient, and Roman Names

Medieval Names

Medieval names present some interesting difficulties, often based on the expected standards of the context in which they are used. Some publications use them like Western names while others do not. In the following preamble snippet we have:

Simplified Name Pattern(s):
`Thomas,Ãã~Kempis` (1–4, 7)
`Thomas,\‘a~Kempis` (5–6)
`Thomas!Ãã~Kempis` (8–10)
Basic Index:
Thomas à Kempis (1–4, 7)
Thomas à Kempis (5–6)
à Kempis, Thomas (8–10)

```
1 \PretagName{Thomas, à~Kempis}{Thomas Akempis}  % medieval
2 \PretagName[Thomas]{à~Kempis}{Akempis, Thomas} % Western
3 \IndexRef[Thomas]{à~Kempis}{Thomas à~Kempis}   % xref
4 \ExcludeName{Thomas,\‘a~Kempis} % alternate form excluded
5 \begin{nameauth}
6   \< KempMed & & Thomas,   à~Kempis & >        % medieval
7   \< KempW     & Thomas  & à~Kempis & >        % Western
8 \end{nameauth}
```

3.1

1. Thomas à Kempis is indexed as “Thomas à Kempis”.
2. Later uses display Thomas because “à Kempis” `\ForceFN\SKempMed` is a place name, not a surname. It is Latin for *von Kempen*.
3. À Kempis `\CapThis\ForceFN\SKempMed` starts a sentence.
4. We use `\PretagName` (Section 2.4.2) to sort the name.
5. Thomas à Kempis `\Name{Thomas,\‘a~Kempis}` is a different name. As above, we would sort this name with `\PretagName`.
6. We used `\ExcludeName` (Section 2.4.1) before using the alternate name to keep it out of the index.
7. We index the canonical form here with `\JustIndex\KempMed`.
8. Thomas à Kempis `\KempW` is a Western form with the index entry: “à Kempis, Thomas”.
9. À Kempis appears via `\CapThis\KempW`.
10. We created a cross-reference from the Western form to the medieval form, before we used the Western form, thus preventing any spurious page entries (Section 2.4.1). We index with the medieval form (7).

Spaces count when sorting index entries (Section 2.4.2). Sorting the cross-reference with `\PretagName[Thomas]{à~Kempis}{a Kempis, Thomas}`, would put it before `aardvark`. `\PretagName[Thomas]{à~Kempis}{Akempis, Thomas}` sorts the cross-reference between `ajar` and `alkaline`.

From this point forward, we shall change what we show in the margin. We shall show full name patterns that reflect naming systems, index sorting tags, index entry tags, cross-references, and name info (Section 2.11.5).

No longer shall we show index entries in the margins because they shall become too complex for such display to work well.

Ancient Names

Ancient contexts may or may not bind particles or other name elements to surnames. One must handle these cases not only in the text, but also in the index. In the rest of this section the examples do not use the formatting conventions of this manual and present themselves as if they were in an ordinary LaTeX document.

- For name entries in the index, we can use `\PretagName` and `\TagName` to ensure that any "long form" information is displayed without using macros in the name arguments. See Sections 2.4.2, 2.4.3.
- In the text, we can use the ⟨*Alternate*⟩ argument or the name information database (Section 2.5) to add "long form" information as needed. Beyond that, we would have to use macros in the name arguments.

First we explore the easiest way to handle royal or ancient variants with extra "long form" information using the ⟨*Alternate*⟩ argument. We use macros introduced in Sections 2.4.2 and 2.4.3.[12]

Name Pattern(s):
`Demetrius,I!PRE`
`Demetrius,I!TAG`
`Demetrius,I!MN`

```
\PretagName{Demetrius, I}{Demetrius 1}
\TagName{Demetrius, I}{ Soter, king}
\begin{nameauth}
  \< Dem & & Demetrius, I & >
\end{nameauth}
```

`\Dem[I Soter]`	Demetrius I Soter
`\LDem`	Demetrius I
`\Dem`	Demetrius

Index (normal LaTeX document): `Demetrius 1@Demetrius I Soter, king`

Using the name information database ("text tags") with the formatting macros (Sections 2.5, 2.10.2), we can provide a more automatic approach:

Name Pattern(s):
`Demetrius,I!DB`
`Demetrius,I!MN`

```
\NameAddInfo{Demetrius, I}{ Soter}
\makeatletter
\renewcommand*\NamesFormat[1]{%
  \begingroup%
  \protected@edef\temp{\endgroup%
    {#1\noexpand\NameQueryInfo
      [\unexpanded\expandafter{\the\@nameauth@toksa}]
      {\unexpanded\expandafter{\the\@nameauth@toksb}}
      [\unexpanded\expandafter{\the\@nameauth@toksc}]%
    }%
  }%
  \temp%
}
\makeatother
```

`\ForgetThis\Dem`	Demetrius I Soter
`\LDem`	Demetrius I
`\Dem`	Demetrius

The index entry is the same as above.

[12] Copies of examples in this section are in `examples.tex`, located with this manual.

Roman Names

Earlier we treated Marcus Tullius Cicero as a Western name. Now we handle Roman names properly. The examples below do not use this manual's standard formatting. Roman names have the following format:

- A personal name: *praenomen*
- A clan name: *nomen*
- A nickname, often hereditary to denote clan branches: *cognomen*
- Affixed names: *agnomina*

Popular works

Popular sources tend to treat the *cognomen* as if it were a Western surname.[13] Using this approach, Roman names have the indexed form:

⟨*cognomen*⟩ ⟨*agnomen*⟩, ⟨*praenomen*⟩ ⟨*nomen*⟩

Using nameauth, one can drop both *praenomen* and *nomen* automatically in subsequent uses in the text. We accomplish this by designing names using macros in their arguments. When doing so, here are a few tips:

- Use alternate formatting (Sections 2.7) if the macros in the name arguments will be "segmented" in some way, as `\CapThis` does by separating the first letter from the rest.
- Use `\noexpand` before the macros in the name arguments if they contain conditional statements. Otherwise one will get spurious index entries.
- Ensure that the default state of any Boolean flags (`\if`⟨*flag*⟩) trigger expansion so that all the desired names appear in the index entry.

We define all macros and conditionals used in naming macro arguments in the preamble. We use `\noexpand` in the naming macro arguments to prevent error. Since we do not use `\CapThis` in the examples below, we skip alternate formatting for simplicity, yet we still recommend it.

Since we have four name components, we need two Boolean flags to reflect local changes and two global flags to trigger the local changes without affecting the index. We define macros in ⟨*FNN*⟩ and ⟨*SNN*⟩ that expand one or two components: *praenomen* and *nomen*, *cognomen* and *agnomen*.

```
\newif\ifSkipGens
\newif\ifNoGens
\newif\ifSkipAgnomen
\newif\ifNoAgnomen
\newcommand*\SCIPi{\ifNoGens
                     Publius\else Publius Cornelius\fi}
\newcommand*\SCIPii{\ifNoAgnomen
                      Scipio\else Scipio Africanus\fi}
\newcommand*\ScipioOnly{\SkipAgnomentrue\Scipio}
\begin{nameauth}
  \< Scipio & \noexpand\SCIPi & \noexpand\SCIPii & >
\end{nameauth}
\PretagName[\noexpand\SCIPi]{\noexpand\SCIPii}{Scipio Africanus}
```

[13] See Geiss, *Geschichte Griffbereit*; Kinder and Hilgemann, *dtv-Atlas zur Weltgeschichte*, 2 vols., 29th printing (1964; Munich: Deutscher Taschenbuch Verlag, 1993). See also this page on indexing and Wikipedia on Roman names.

We begin a new scope below, redefining the formatting hooks (Section 2.6), which affect only names printed in the text. If the local Boolean flags are false, one gets longer name forms. If the flags are true, one gets shorter forms. This approach allows the global state of the flags to be false by default, meaning that one need not remember to set any of them true in the preamble. That results in one less thing to remember, and one less problem to fix.

```
\renewcommand*\NamesFormat[1]
  {\ifSkipGens\NoGenstrue\fi\ifSkipAgnomen\NoAgnomentrue\fi#1%
  \global\SkipGensfalse\global\SkipAgnomenfalse}
\renewcommand*\MainNameHook[1]
  {\ifSkipGens\NoGenstrue\fi\ifSkipAgnomen\NoAgnomentrue\fi#1%
  \global\SkipGensfalse\global\SkipAgnomenfalse}
```

The index always shows the name determined by the global state of `\NoGens` and `\NoAgnomen`, which we set up as false, meaning a maximally long name form. In the body text we have:

> Publius Cornelius Scipio `\ScipioOnly` was born around 236 BC into the Scipio branch of the Cornelius clan, one of six large patrician clans. Scipio `\ScipioOnly` rose to military fame during the Second Punic War. Thereafter he was known as Scipio Africanus `\Scipio`.

Below we show more information about popular name forms by way of comparison with scholarly name forms.

Scholarly works

The *Oxford Classical Dictionary* and other scholarly sources index according to the *nomen*. That approach moves the *nomen* from ⟨*FNN*⟩ to ⟨*SNN*⟩. They have the the indexed form:

> ⟨*nomen*⟩ ⟨*cognomen*⟩ ⟨*agnomen*⟩, ⟨*praenomen*⟩

The two methods do not clash *per se* in the text, but they make incompatible index entries. In this case, since we have indexed Scipio under the popular form above, we use `\ExcludeName` to exclude the scholarly form below.

In the document preamble we define the following Boolean flags and macros. We use a nested conditional in ⟨*SNN*⟩. The default still is to show all names so that they can be indexed that way:

```
\newif\ifSkipGens % These flags remain the same as above.
\newif\ifNoGens
\newif\ifSkipAgnomen
\newif\ifNoAgnomen
\global\def\CSA{\ifNoGens\ifNoAgnomen
                  Scipio\else
                  Scipio Africanus\fi
                \else\ifNoAgnomen
                  Cornelius Scipio\else
                  Cornelius Scipio Africanus\fi\fi}
\ExcludeName[Publius]{\noexpand\CSA}
\begin{nameauth}
  \< OScipio & Publius & \noexpand\CSA & > % O for Oxford
\end{nameauth}
\PretagName[Publius]{\noexpand\CSA}{Cornelius Scipio Africanus}
```

We keep the same formatting macros that we defined above. By the way, these formatting macros could work with regular names as well as Roman names, e.g., Demetrius I, because they have no side effects.

The scholarly form of Roman names has a different name pattern, so it is not compatible with the popular version. Nevertheless, we show what the index entries would be in a normal LaTeX document without hyperlinks. Since we have excluded the scholarly form in order to suppress any spurious index entries, we only index the popular form below:

Simplified Name Patterns:
Scholarly: `Publius!\noexpand\CSA`
Popular: `\noexpand\SCIPi!\noexpand\SCIPii`

Full Index Entries:
Scholarly: `Cornelius Scipio Africanus@Cornelius Scipio Africanus, Publius`
Popular: `Scipio Africanus@Scipio Africanus, Publius Cornelius`

Basic Index Entries:
Scholarly: Cornelius Scipio Africanus, Publius
Popular: Scipio Africanus, Publius Cornelius

Below we compare some differences between the scholarly and popular forms, and how to get equivalent forms in the text while understanding that they would be two separate forms in the index.

First use:
scholarly: . . . . . `\OScipio` Publius Cornelius Scipio Africanus
popular: . . . . . `\Scipio` Publius Cornelius Scipio Africanus

Subsequent use:
scholarly: . . . . . `\OScipio` Cornelius Scipio Africanus
scholarly: . . . . . `\SkipGenstrue\OScipio` Scipio Africanus
popular: . . . . . `\Scipio` Scipio Africanus

Subsequent use, full, no *agnomen*:
scholarly: . . . . . `\SkipAgnomentrue\LOScipio` Publius Cornelius Scipio
popular: . . . . . `\SkipAgnomentrue\LScipio` Publius Cornelius Scipio

Subsequent use, shortest forms:
scholarly: . . . . . `\SkipAgnomentrue\OScipio` Cornelius Scipio
`\SkipGenstrue\SkipAgnomentrue\OScipio` Scipio
popular: . . . . . `\SkipAgnomentrue\Scipio` Scipio

Subsequent use, personal name:
scholarly: . . . . . `\SOScipio` Publius
popular: . . . . . `\SScipio` Publius Cornelius
popular: . . . . . `\SkipGenstrue\SScipio` Publius

Back to Section 1.3

Oft fühl ich jetzt . . . [und] je tiefer ich einsehe, dass
Schicksal und Gemüt Namen eines Begriffes sind.

—Novalis, *Heinrich von Ofterdingen*

2.4 Indexing Macros

3.5 The strictness of the `nameauth` indexing macros and the detail of index-related warnings, especially with the `verbose` option, are comparable to professional indexing software. In addition to the macros below, please see also:

Here also are the general protection rules for the indexing macros:

Permit	Ignore	Attempted Action
☐	■	Use `\IndexRef` with an extant name.
■	☐	Use `\SeeAlso\IndexRef` with an extant name.
☐	■	Create page references to a name after `\IndexRef` created a cross-reference using that name.
☐	■	Try to make the same cross-reference multiple times.
☐	■	Use `\IndexName` with a cross-reference.
☐	■	Use `\ExcludeName` with a cross-reference.
☐	■	Use `\IncludeName` with a cross-reference.
■	☐	Use `\IncludeName*` with a cross-reference.
■	☐	Use `\PretagName` to sort a cross-reference.
☐	■	Use `\TagName` and `\UntagName` with a cross-reference.

We test `\ExcludeName{Gregory, I}` here. See page 37.

2.4.1 General Indexing Macros

General Control

`\IndexInactive` `\IndexActive`

`\IndexInactive` deactivates the indexing functions of the naming macros, `\IndexName`, and `\IndexRef`. `\IndexActive` enables indexing. These can be used throughout the document.

- `\IndexInactive` broadly suppresses `\IndexName`, `\IndexRef`, the page entry indexing components of the naming macros, and the cross-referencing components of `\AKA` and `\PName`.
- For a fine degree of control, use `\ExcludeName` and `\IncludeName`.

`\global`

`\IndexActive` and `\IndexInactive` can be used as a pair or singly within a group. They have top priority (page 20). Use `\global` to force a global effect.

`\IndexProtect`

3.3 This macro causes all naming macros to do nothing. `\IndexProtect` is local in scope, e.g., `{\IndexProtect\Name{Error}}` is isolated. The naming macros and `\AKA` have locks that prevent them from being used in their own arguments to prevent errors. To prevent unlikely errors in the index, one can use `\IndexProtect` right before `\printindex` to eliminate spurious output.[14]

[14] This manual uses the tag § for `\Name{foo\Name{bar}}`, not shown in the example.

Macro	Text	.ind file	Index
\Name{foo\Name{bar}}	foo	\item foo\Name {bar}	foobar
\printindex (next iteration adds) →		\item bar	bar
\Name{foo\Name{bar}}	foo	\item foo\Name {bar}	foo
\IndexProtect\printindex		(no further output)	

\NameauthIndex **3.5** LaTeX has various ways to produce multiple indexes. \NameauthIndex, which is defined as \index, can be redefined to implement multiple indexes of names. Below we use the index package to do this, but other alternatives also are possible.[15]

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{index}
\usepackage{nameauth}

\makeindex % Default index
\newindex{per}{rdx}{rnd}{Index of Persons} % Other index
\renewcommand\NameauthIndex{\index[per]}

\begin{document}
  Electric Boogaloo\index{Boogaloo, Electric}% to main index
  by \Name{Ollie~\& Jerry}%                    to name index

  \printindex[per]% Shows the entry: Ollie & Jerry, 1

  \renewcommand\indexname{Index of Subjects}
  \printindex     % Shows the entry: Boogaloo, Electric, 1
\end{document}
```

Page Entries

\IndexName Both package users and the naming macros themselves use this macro to create index entries. It prints nothing in the body text:

\IndexName[⟨*FNN*⟩]{⟨*SNN, Affix*⟩}[⟨*Alternate*⟩]

If ⟨*FNN*⟩ is present, it ignores ⟨*Alternate*⟩ for Western and "native" Eastern name forms. If ⟨*FNN*⟩ is absent, \IndexName can use either the current or the obsolete non-Western syntax (Section 2.11.4). Indexing follows [Mulvany, 152–82].

3.5 Currently, we assume that if one wants to use \IndexName, one really wants to index something. That means \IndexName will not respect \SkipIndex. Using \IndexInactive will still suppress indexing. The naming macros have used the following for some time, which causes \IndexName to obey \SkipIndex:

```
\unless\if@nameauth@SkipIndex \IndexName...\fi.
```

3.5 The stricter indexing control in place means that:

- \IndexName will not index names excluded by \ExcludeName, as well as cross-references. This has been true for quite a while.
- \IndexName resets the effects of both \SeeAlso and \SkipIndex unless one uses the oldreset option.

\SkipIndex **3.1** The prefix macro \SkipIndex will suppress indexing for just one instance of a naming macro. See also page 38. \SkipIndex\Name[Monty]{Python} produces Monty Python and Python in the text, but with no index entry.

\JustIndex **3.5** This prefix macro makes \Name, \Name*, \FName, and the quick interface shorthand macros act similar to a one-time call to \IndexName. \JustIndex suppresses name output in the text, but it resets flags for long and first name forms as if the naming macro had produced output. Using the oldreset option prevents these flags from being reset. See also page 38.

Cross-References

\IndexRef By default, \IndexRef creates a *see* reference from the name defined by its first three arguments to the target in its final argument:

\IndexRef[⟨*FNN*⟩]{⟨*SNN, Affix*⟩}[⟨*Alternate*⟩]{⟨*reference target*⟩}

3.5 The stricter indexing control in place means that:

- \IndexRef will not cross-reference names excluded by \ExcludeName, as well as cross-references. This has been true for quite a while.
- \IndexRef will not index any extant names used with \Name and friends, as well as the quick interface, unless one uses the oldsee option.
- \IndexName resets the effects of both \SeeAlso and \SkipIndex unless one uses the oldreset option.
- To have multiple names and cross-references interact, see page 39.

\IndexRef prints nothing in the text. The name parsing is like \IndexName. The final argument is not checked in any way. For example:

Name Pattern(s):
SunKing!PN

source: \IndexRef{Sun King}{Louis XIV}
index: Sun King *see* Louis XIV

\SeeAlso **3.5** Put \SeeAlso before \IndexRef, \AKA, and \PName to make a *see also* reference for a name that has appeared already in the index. If enabled before invoking \PName, \SeeAlso will be disabled when the regular name is generated, then enabled when the cross-reference is generated. Currently \IndexName and any macros that use it will reset the Boolean flag governed by \SeeAlso unless one uses the oldreset option. This does not change the intended behavior of \SeeAlso. Rather, it prevents a stray use of the macro from affecting the index.

\ExcludeName **3.0** This macro prevents a name from being used as either an index entry or as an index cross-reference. It will not exclude extant cross-references:

\ExcludeName[⟨*FNN*⟩]{⟨*SNN, Affix*⟩}[⟨*Alternate*⟩]

Unlike \IndexInactive and \IndexActive, which inhibit indexing altogether, this macro only excludes a specific name from being printed as a page reference or cross-reference in the index. See the following example, as well as examples in Sections 2.2.3 and 2.3.4:

[15] See also this on multiple indexes and the test in examples.tex, located with this manual.

Name Pattern(s):
Kris!Kringle!PN
Santa!Claus!PN
Grinch!PN
Kris!Kringle!MN
Santa!Claus!MN
Grinch!MN

```
\ExcludeName[Kris]{Kringle}
\ExcludeName[Santa]{Claus}
\ExcludeName{Grinch}
\Name[Kris]{Kringle}, a.k.a. \Name[Santa]{Claus}
even likes the \Name{Grinch}.
```

Kris Kringle, a.k.a. Santa Claus even likes the Grinch.

\IncludeName
\IncludeName*

Use these macros to break a few indexing rules. They remove the protections used for exclusion and cross-referencing. They have the same syntax as \ExcludeName:

\IncludeName [⟨*FNN*⟩]{⟨*SNN, Affix*⟩}[⟨*Alternate*⟩]
\IncludeName*[⟨*FNN*⟩]{⟨*SNN, Affix*⟩}[⟨*Alternate*⟩]

\IncludeName only voids an exclusion created by \ExcludeName. The more extreme \IncludeName* completely un-protects a cross-reference. Thereafter, one may create page entries for it as a name. For example:

Name Pattern(s):
Gregory,I!PN
Gregory,I!MN

- As an excluded name, \Name*{Gregory, I} Gregory I does not create an index entry.
- \IfAKA{Gregory, I}{⟨*an xref*⟩}{⟨*a name*⟩}{⟨*excluded*⟩} tells us that Gregory is ⟨*excluded*⟩ (cf. Section 2.8.2).

After using \IncludeName{Gregory, I}, the following points are true:

- \Name*{Gregory, I} displays Gregory I and creates an index entry.
- \IfAKA{Gregory, I}{⟨*an xref*⟩}{⟨*a name*⟩}{⟨*excluded*⟩} now tells us that Gregory has become ⟨*a name*⟩.

Name Pattern(s):
J.D.!Rockefeller,IV!MN
Jay!Rockefeller!PN
Jay!Rockefeller!MN

Cross-references get more protection. We have seen Jay Rockefeller indexed under "Rockefeller, J.D., IV" with \DropAffix\LJRIV[Jay]. We create the cross-index \IndexRef[Jay]{Rockefeller}{Rockefeller, J.D., IV}.

- \IfAKA[Jay]{Rockefeller}{⟨*xref*⟩}{⟨*name*⟩}{} calls Jay an ⟨*xref*⟩.
- After \IncludeName[Jay]{Rockefeller} he still is an ⟨*xref*⟩.
- After \IncludeName*[Jay]{Rockefeller} he becomes a ⟨*name*⟩.
- Now \Name[Jay]{Rockefeller} will create page entries.

Analogous to how \IncludeName* allows one to turn an xref into a name with page references, \ForgetName and possibly \ForgetThis (Section 2.8.1) allow one to assign a *see* reference to an extant name. This is bad when using one index, but desirable when using multiple indexes.

Error Prevention

3.5 \IndexName and \IndexRef strictly enforce professional indexing practices. Now they are more sensitive to order of use. Both \ForgetName and \ForgetThis (Section 2.8.1) can affect this mechanism. The verbose option aids debugging.

Page	Macro	Index Result
1	`\Name{bar}`	bar, 1
2	`\Name{bar}`	bar, 1, 2

- A *see also* reference to a certain name must follow all page references to that name. The use of `\SeeAlso\IndexRef{bar}{foo}` on page 3 prevents `\Name{bar}` from creating an index page reference on page 4:

3	`\SeeAlso\IndexRef{bar}{foo}`	bar, 1, 2, *see also* foo
4	`\Name{bar}`	bar, 1, 2, *see also* foo

- Targets of cross-references are not affected by this. `\Name{foo}` creates index page entries because it is the target, not the xref:

5	`\Name{foo}`	bar, 1, 2, *see also* foo foo, 5

3.5

- A *see* reference is supposed to have no page references; a *see also* reference does have page references, but only before it is created. Trying to use an extant name as a *see* reference is strictly ignored:

6	`\Name{baz}`	bar, 1, 2, *see also* foo baz, 6 foo, 5
7	`\IndexRef{baz}{meschugge}` . .	bar, 1, 2, *see also* foo baz, 6 foo, 5

Prefix Macros and Indexing Macros

Indexing macros ignore Boolean flags meant for naming macros. Yet there are three prefix macros that affect indexing: `\SeeAlso`, `\SkipIndex`, and `\JustIndex`.

- `\SeeAlso` works with and is reset by `\IndexRef`, `\AKA`, and `\PName` (see also Section 2.9). There are few, if any, side effects.

- `\SkipIndex` works with the naming macros. Side effects include:

3.5

  - Unless the `oldreset` option is used, both `\IndexName` and `\IndexRef` issue warnings if `\SkipIndex` precedes them, ignore `\SkipIndex`, and reset its flag.

3.5

  - Only when the `oldreset` option is used, both `\PName` and `\PName*` issue warnings when `\if@nameauth@SkipIndex` is true on exit.

- `\JustIndex` makes the naming macros act like `\IndexName`. That affects both the printing and indexing flags governed by prefix macros.

There are many potential side effects related to `\JustIndex`:

- Both `\AKA` and `\PName` ignore `\JustIndex` and go on about their business. They also set `\@nameauth@JustIndexfalse`.
- `\JustIndex` causes the naming macros to imitate `\IndexName` and ignore most flags, except for those discussed in the next bullet point. That makes the following two lines equivalent:

```
\SkipIndex \JustIndex \Name{A} \Name{B}
\JustIndex \Name{A} \SkipIndex \Name{B}
```

According to the table on page 20, `\JustIndex` takes priority with `\Name{A}` and passes `\@nameauth@SkipIndextrue` to `\Name{B}`.

3.3

- Currently, the naming macros always reset `\@nameauth@FullNamefalse` and `\@nameauth@FirstNamefalse`. For example:

Name Pattern(s):
`George!Washington!MN`

Source	Output
`\JustIndex\FName[George]{Washington}%`	
`  \Name[George]{Washington}`	Washington
`\JustIndex\SWash \Wash`	Washington
`\JustIndex\Name*[George]{Washington}%`	
`  \Name[George]{Washington}`	Washington
`\JustIndex\LWash \Wash`	Washington

- The `oldpass` option restores the old behavior, which did not always reset the name length modifier. For example:

Name Pattern(s):
`George!Washington!MN`

Source	Output
`\JustIndex\FName[George]{Washington}%`	
`  \Name[George]{Washington}`	George
`\JustIndex\SWash \Wash`	George
`\JustIndex\Name*[George]{Washington}%`	
`  \Name[George]{Washington}`	George Washington
`\JustIndex\LWash \Wash`	George Washington

Variant Names and Cross-References

Here we show differences among variants and cross-references. We can choose to index variants under the canonical name or we can set up cross-references with variants. The order in which we do that is significant:

Name Pattern(s):
`J.E.!Carter,Jr.!MN` (1–2)
`Jimmy!Carter!PN` (3, 6)
`Jimmy!Carter!MN` (4)
`J.E.!Carter,Jr.!PN` (5)

1. We use the canonical name to create page references:

   `\Name*[J.E.]{Carter, Jr.}` . . . . . J.E. Carter Jr.

2. Variants that use ⟨*Alternate*⟩ in the text create page entries under the canonical form, not the variant form:

   `\DropAffix\Name*[J.E.]{Carter, Jr.}[Jimmy]` . . Jimmy Carter
   `\ShowIdxPageref*[J.E.]{Carter, Jr.}[Jimmy]` Carter, J.E., Jr.

3.5

3. We must create a *see* reference from an alternate form to a canonical form **before** using the alternate form in a naming macro, or it will be ignored and a warning will result:

   `\IndexRef[Jimmy]{Carter}{Carter, J.E., Jr.}`

4. No page references will occur below because we made the *see* reference first. Note how the alternate form is an independent name:

   `\Name[Jimmy]{Carter}` . . . . . Jimmy Carter

5. If we want to index the alternate name, we have to use the canonical name instead of the alternate name:

   `\IndexName[J.E.]{Carter, Jr.}`

6. If instead we wanted to make a *see also* reference, we would use both the canonical name and the alternate name, then create the cross-reference **after** all uses of the alternate name (at the end of the document), e.g.:

   ```
   \SeeAlso\IndexRef[Jimmy]{Carter}{Carter, J.E., Jr.}
   ```

Multiple connections

Below, two names are indexed with page numbers. They have *see also* cross-references to each other. One of those names also has a *see* reference to it:

Name Pattern(s):
`Maimonides!MN` (1)
`Moses,ben-Maimon!PN` (2)
`Moses,ben-Maimon!MN` (3)
`Rambam!MN` (4)
`Rambam!PN` (5)

1. We use the canonical name to set up page references:

   `\Name{Maimonides}` . . . . . Maimonides

2. Maimonides has two other names, one more used than the other. We set up his least-used name as the *see* reference:

   ```
   \IndexRef{Moses, ben-Maimon}{Maimonides}
   ```

3. We now have a main name with a page entry and a "*see* reference" to that name. No page references will occur below because we made the *see* reference first:

   `\Name{Moses, ben-Maimon}` . . . . . Moses ben-Maimon

4. Before creating *see also* cross-references, we use the other alternate name so that all the page entries precede the cross-references:

   `\Name{Rambam}` . . . . . Rambam

5. All *see also* references must come after all page references. For example, one could put both of these macros at the end of the document:

   ```
   \SeeAlso\IndexRef{Maimonides}{Rambam}
   \SeeAlso\IndexRef{Rambam}{Maimonides}
   ```

Combining xrefs

`\IndexRef` will not merge multiple cross-references and it will not allow more than one cross-reference. For multiple cross-references one must use something like:

source: `\IndexRef{bar}{baz; foo}`
index: bar, *see* baz; foo

Multiple targets

There is a special case where one cross-reference can point to multiple targets, such as demonstrated in the example below:

Name Pattern(s):
`\textit{Snellius}!PRE`
`\textit{Snellius}!PN`
`W.!SnelvanRoyen!MN`
`R.!SnelvanRoyen!MN`

```
1 \PretagName{\textit{Snellius}}{Snellius}
2 \IndexRef{\textit{Snellius}}
3   {Snel van Royen, R.; Snel van Royen, W.}
4 Both \Name[W.]{Snel van Royen}[Willebrord] and
5 his son \Name[R.]{Snel van Royen}[Rudolph] were known
6 by the Latin moniker \Name{\textit{Snellius}}.
```

Both Willebrord Snel van Royen and his son Rudolph Snel van Royen were known by the Latin moniker *Snellius*.

Location matters

`\IndexRef` prevents page numbers in cross-references, so one must plan how to set up complex cross-references. Above, `\Name{\textit{Snellius}}` produces no index entry because `\IndexRef` comes first.

Back to Section 1.3

2.4.2 Index Sorting

Here we introduce the index sorting macros, with examples of error prevention:

\IndexActual

Using \index{⟨*sort key*⟩@⟨*actual*⟩} works with both makeindex and texindy.[16] By default, the "actual" character is @. If one needs to change the "actual" character, such as when using gind.ist with .dtx files, one would put \IndexActual{=} in the preamble (or driver section) before using \PretagName.

\global

Effects of \IndexActual are local in scope. Use \global to make it otherwise, but that will affect every use of \PretagName thereafter. We demonstrate this scoping below as it pertains to gind.ist in a dtx file:

Name Pattern(s):
ÃĘgidius!PRE

```
\PretagName{Ægidius}{Aegidius}
\begingroup
  \IndexActual{@}
  \ShowIdxPageref{Ægidius}\qquad
\endgroup
\ShowIdxPageref{Ægidius}
```

Aegidius@Ægidius Aegidius=Ægidius

\PretagName

The nameauth package enables automatic index sorting using a "pretag" (cf. Section 2.11.5). Unless the nopretag option is used (which results in warnings), \PretagName creates a sort key terminated with the "actual" character. Do not put the "actual" character in the "pretag":

\PretagName[⟨*FNN*⟩]{⟨*SNN, Affix*⟩}[⟨*Alternate*⟩]{⟨*tag*⟩}

One can "pretag" any name, any cross-reference, and even excluded names. Once made, sorting tags cannot be unmade. If one uses \PretagName in the preamble, those names will be sorted automatically. For example:

Name Pattern(s):
ÃĘthelred,II!PRE
W.E.B.!Du~Bois!PRE
ÃĘthelred,II!MN
W.E.B.!Du~Bois!MN

```
\PretagName{Æthelred, II}{Aethelred 2}
\PretagName[W.E.B.]{Du~Bois}{Dubois, William}
```

Every reference to Æthelred II and W.E.B. Du Bois is automatically tagged and sorted. One should "pretag" all names that contain active characters or macros. That can differ when using xindy and Unicode-based LaTeX. We keep this example simple and do not use alternate formatting (cf. Sections 2.3.5, 2.7). The name patterns are: \textit{Doctorangelicus}!PRE \textit{Doctorangelicus}!PN Thomas,Aquinas!MN \textit{Doctorangelicus}!MN:

```
\PretagName{\textit{Doctor angelicus}}{Doctor angelicus}
\IndexRef{\textit{Doctor angelicus}}{Thomas Aquinas}

Perhaps the greatest medieval theologian was
\Name{Thomas, Aquinas}, later known as
\Name{\textit{Doctor angelicus}}.
```

Perhaps the greatest medieval theologian was Thomas Aquinas, later known as *Doctor angelicus*.

Particles and languages

Spaces change sorting. For example, the sort tag `De␣Soto` precedes `deal` due to the space therein. The sort tag `DeSoto` falls between `derp` and `determinism`. German ä ö ü ß map to English `ae oe ue ss`. Yet Norwegian æ ø å follow `z` in that order. Check a style guide regarding collating sequences, spaces, and sorting. This is where using `xindy` can be very helpful. See also Section 2.3.4.

Sub-entries

One can sort names by creating sub-entries, which depends on the index style and formatting files: `\PretagName[Some]{Name}{`⟨*category*⟩`!Name, Some}`. See also the documentation for `xindy` and `makeindex`.

Potential Sorting Problems

If an entry is sorted incorrectly in the index, check the following:

- Are there any active characters, internal spaces, or control sequences in the name arguments? Use `\PretagName`.
- Is alternate formatting used consistently? Are any names used within sections of alternate formatting ever used outside of them?
- Are macros in the name arguments that can expand differently under different conditions preceded by `\noexpand`?

Since 2018, changes in the way that `pdflatex` and `latex` handle Unicode characters have made indexing simpler and more intuitive, e.g.

pre-2018 text		index	post-2018 text		index
ä	→	`\IeC␣{\"a}`	ä	→	`ä`
æ	→	`\IeC␣{\ae␣}`	æ	→	`æ`

One can test for this change and take different approaches with:

`\IfFileExists{utf8-2018.def}{`⟨*yes*⟩`}{`⟨*no*⟩`}`

One also should look at the entries in the `.idx` or `.ind` files to see how the name arguments and other index entry components are turned into index entries. If there are entries that do not work, one can find the corresponding page numbers in order to identify the problem.

Extra spaces are significant when sorting index entries, yet usually are not significant in the body text. Hidden spaces, tokens, macros, and control sequences create unique index entries that look similar, yet expand and sort differently. Some macros can add spaces to index entries.

This is not an issue with the `nameauth` package as such. Rather, it stems from the use of `\protected@edef`. This package requires `\protected@edef` in any situation where the macros that generate index entries are written to the `aux` file for execution there, such as in the `memoir` class. Without `\protected@edef`, any active Unicode characters would start expanding, thus making different index entries than those one might make by hand. Since we do not want a "package way" to index names and a "regular way" that is different and hard to integrate with the "package way", this is an unavoidable necessity.

[16] The general practice for sorting with `makeindex -s` involves creating an `.ist` file (pages 659–65 in *The Latex Companion*).

Below we show the general minimal working example:

```
\newcommand\Idx[1]{%
  \protected@edef\arg{#1}%
  \index{\arg}}
```

The macro `\Idx{\textsc{football}}` produces:
`\indexentry{\textsc␣␣{football}}{`⟨*page*⟩`}`

The macro `\index{\textsc{football}}` produces:
`\indexentry{\textsc{football}}{`⟨*page*⟩`}`

The debugging macros will not help at this point. We must inspect the `idx` file. The problem with the debugging macros is that they show how an index entry will appear on the page, more or less, but not how it will appear in the `idx` file, which determines sorting.

Back to Section 1.3

2.4.3 Index Tags

`\TagName` `\UntagName`

This macro creates a tag appended to all index entries for a corresponding `\Name`. The tag persists until one changes it with `\TagName` or destroys it with `\UntagName`. Tags can include life dates, regnal dates, and other information. Both `\TagName` and `\UntagName` have **global scope** and handle arguments like `\IndexName`:

`\TagName[`⟨*FNN*⟩`]{`⟨*SNN, Affix*⟩`}[`⟨*Alternate*⟩`]{`⟨*tag*⟩`}`
`\UntagName[`⟨*FNN*⟩`]{`⟨*SNN, Affix*⟩`}[`⟨*Alternate*⟩`]`

All the indexing macros are keyed to the name patterns. `\PretagName` generates the leading sort key. `\TagName` and `\UntagName` affect the trailing content. The following graphic illustrates the "segments" of an index entry and the nameauth macros that affect the respective segments:

	\PretagName	\IndexName	
\index{	**Aethelred 2@**	**Æthelred II**	**, king}**
			\TagName
			\UntagName

Scholarly helps

Tags created by `\TagName` can be helpful in the indexes of academic texts by adding dates, titles, etc. `\TagName` causes the nameauth indexing macros to append "`,␣pope`" to the index entries created below:

Name Pattern(s):
`Gregory,I!TAG`
`Gregory,I!MN`
`Gregory,theGreat!PN`

```
\TagName{Gregory, I}{, pope}
Pope \Name*{Gregory, I} was known as \Name{Gregory, I}
``the Great''.\IndexRef{Gregory, the Great}{Gregory I}
```

Pope Gregory I was known as Gregory "the Great".

`\TagName` works with all names, but not with cross-references from `\IndexRef`, `\AKA`, etc. (cf. Sections 2.4.1, 2.9). Tags also can be daggers, asterisks, and so on. For example, all fictional names in the index of this manual are tagged with §. One must add any desired spaces to the start of the tag.

Same name game

We can format and index one name as two different people with `\TagName` and `\ForgetThis` (Section 2.8.1). The index tags group together their respective entries. In a normal LaTeX document one would write, e.g.:

Name Pattern(s):
`E.!Humperdinck!TAG`
`E.!Humperdinck!MN`

```
\TagName[E.]{Humperdinck}{ (composer)}
This refers to the classical composer:
\Name[E.]{Humperdinck}[Engelbert].

\TagName[E.]{Humperdinck}{ (singer)}
This refers to the pop singer from the 60s and 70s:
\ForgetThis\Name[E.]{Humperdinck}[Engelbert].
```

This refers to the classical composer: Engelbert Humperdinck.

This refers to the pop singer from the 60s and 70s: Engelbert Humperdinck.

Special tags

3.3

One can use `\TagName` to create "special" index entries for names with the general form `\TagName`⟨*name args*⟩`{|`⟨*Macro*⟩`}`, when `\def\`⟨*Macro*⟩`#1{#1}` exists. These tags are compatible with hyperref.[17]

For example, using the ltxdoc class with hypdoc does not create hyperlinked page entries with nameauth. This behavior does not affect normal LaTeX documents that use nameauth and hyperref. In this manual we had to tag every name with `\TagName`⟨*name args*⟩`{|hyperpage}` in the driver section of the `dtx` file.

In the "commented" package documentation part of a `dtx` file, the vertical bar is active. This adds an extra layer of complexity. Index tags in the documentation part must use the form: `\TagName`⟨*name args*⟩`{\string|hyperpage}`.

Below we use the conventions of this manual to create a special tag:

```
\newcommand\Orphan[2]{#1(\hyperpage{#2})}
\TagName[Lost]{Name}{\,\S|Orphan{perdit}}
\Name[Lost]{Name}
```

Lost Name
`idx` file: `\indexentry{Name, Lost\,\S  |Orphan{perdit}}{`⟨*page*⟩`}`
`ind` file: `\item Name, Lost\,\S  \pfill \Orphan{perdit}{`⟨*page*⟩`}`

Error prevention

When `\IndexRef` calls `\@nameauth@Index`, a tag of the form ⟨*some text*⟩`|`⟨*some macro*⟩ is reduced to ⟨*some text*⟩, allowing a new `|`⟨*cross-reference*⟩ macro to be added. This keeps cross-references from breaking.

Manual breaks and entries

The microtype package and its `Spacing` environment may be the best solution to fix index entries and sub-entries that break badly across columns or pages. Suppose, however, that we wanted to insert manual breaks into an index at the very end of the editing and proofreading passes. That is fairly easy to do.

We cannot just insert something like `\newpage` or `\columnbreak` directly into an index. Instead, we create a helper macro that takes an argument and adds a

[17] Before version 3.3 these special tags did not work with hyperref. The fix was inspired by Heiko Oberdiek in this question.

break after that argument. That is, for example, how macros like `\textbf` use implied page references: `\index{Doe, John|textbf}`.

Below we use `\newpage`, but if we instead make use of the multicol or idxlayout packages we can replace that with `\columnbreak`. On line 1 we define the `\Endbreak` macro that will break the index page or column at the end of an entry. On line 3 we do the same for somewhere in the middle of an index entry. In the latter case there will be a comma that we must `\@gobble` after a page reference:

```
1 \newcommand*{\EndBreak}[1]{#1\newpage}
2 \makeatletter
3 \newcommand*{\MidBreak}[1]{#1\newpage\@gobble}
4 \makeatother
```

Using `\MidBreak` to insert a break into the middle of an index entry does work to some extent, but what it does is quite sketchy and error-prone. We avoid using it for these reasons, but we just wanted to show that it can be done.

Instead, we use `\EndBreak` after the last page in a given entry. This method works for manual index entries and for the nameauth macros. If all instances of `\Name{Some, Name}` on page 18 have the same tag, there will be no duplicate page entries, hyperlinks will work, and the index will break as indicated:

Page	Macro	Index Result
10	`\Name{Some, Name}`	Some Name, 10
	`\index{Topic}`	Topic, 10
15	`\Name{Some, Name}`	Some Name, 10, 15
	`\index{Topic}`	Topic, 10, 15
18	`\TagName{Some, Name|EndBreak}%`	
	`\Name{Some, Name}`	Some Name, 10, 15, 18⟨*break*⟩
	`\index{Topic|EndBreak}`	Topic, 10, 15, 18⟨*break*⟩

We do not have to supply an argument to `\EndBreak` because, as with the font switching example above, the page reference is implied.

We can intermix nameauth macros with manual index entries. We may need to look at the `idx` or `ind` files to craft matching entries on the page where the break occurs. Instead of using `\TagName`, we can do this:

18	`\SkipIndex\Name{Some, Name}%`	
	`\index{Some Name|EndBreak}`	Some Name, 10, 15, 18⟨*break*⟩

Results for manual entries may vary, depending on what distribution of LaTeX is being used and how old it is. As we saw in the previous section, any name with active characters needs to be handled differently before 2018 than after 2018. All instances of `\index{Some Name|EndBreak}` must fall on the same page.

Back to Section 1.3

2.5 "Text Tags"

\NameAddInfo

Unlike index tags, "text tags" are not printed automatically with every name managed by nameauth. Sections 2.8.2 and 2.10.2 have more examples. The macro is \long, allowing for some complexity in the ⟨*tag*⟩ argument:

\NameAddInfo[⟨*FNN*⟩]{⟨*SNN, Affix*⟩}[⟨*Alternate*⟩]{⟨*tag*⟩}

Name Pattern(s):
George!Washington!DB
George!Washington!MN

For example, \NameAddInfo[George]{Washington}{(1732--99)} makes a text tag but does not print whenever \Wash Washington is used.

\NameQueryInfo

To print the text tag macro associated with a name, we use \NameQueryInfo, which calls the appropriate macro in the name info data set:

\NameQueryInfo[⟨*FNN*⟩]{⟨*SNN, Affix*⟩}[⟨*Alternate*⟩]

Name Pattern(s):
George!Washington!DB
UlyssesS.!Grant!DB
Schuyler!Colfax!DB
Schuyler!Colfax!MN
UlyssesS.!Grant!MN

\NameQueryInfo[George]{Washington} expands to (1732–99). One can insert a space at its start or use signs like asterisks, daggers, and even footnotes, such as one for Schuyler Colfax.[18] Below is the source for footnote 18:

```
1 \NameAddInfo[Ulysses S.]{Grant}{(president from 1869 to 1877)}%
2 \NameAddInfo[Schuyler]{Colfax}{\footnote{He was the seventeenth
3 US vice-president, holding office during the first term (1869--73)
4 of \Name[Ulysses S.]{Grant} \NameQueryInfo[Ulysses S.]{Grant}.}}
  ...\Name[Schuyler]{Colfax}.\NameQueryInfo[Schuyler]{Colfax}
```

One can nest "text tags" and have them call each other. Therefore, one must protect against a stack overflow by using Boolean flags to stop the recursion:

```
1 \newif\ifA
2 \newif\ifB
3 \NameAddInfo{A}{%
4   \Atrue A \ifB Stop \else \NameQueryInfo{B} \fi \Afalse}
5 \NameAddInfo{B}{%
6   \Btrue B \ifA Stop \else \NameQueryInfo{A} \fi \Bfalse}

\NameQueryInfo{A} → A B Stop
\NameQueryInfo{B} → B A Stop
```

\NameClearInfo

\NameAddInfo will replace one text tag with another text tag, but it does not delete a text tag. That is the role of \NameClearInfo. The syntax is:

\NameClearInfo[⟨*FNN*⟩]{⟨*SNN, Affix*⟩}[⟨*Alternate*⟩]

Name Pattern(s):
George!Washington!DB

After using \NameClearInfo[George]{Washington}, the next attempt to query the tag \NameQueryInfo[George]{Washington} will produce nothing.

Back to Section 1.3

[18] He was the seventeenth US vice-president, holding office during the first term (1869–73) of Ulysses S. Grant (president from 1869 to 1877).

2.6 Basic Formatting

Below are color-coded forms and formats of names, showing most (but not all) of the variations that can occur. The next several sections will explain such variations (and quite a few more) in detail:

Full Forms, Front Matter

`\NamesInactive`

First Use (Default)

`\Name`	George S. Patton Jr. Elizabeth I Yamamoto Isoroku (cf. Section 2.8.1)

Later Use (* or \L⟨*macro*⟩)

`\Name*`	George S. Patton Jr. Elizabeth I Yamamoto Isoroku

Long, with `\DropAffix`

`\DropAffix\L`⟨*macro*⟩
George S. Patton

Short Forms, Front matter

`\NamesInactive`

Later Use (Default)

`\Name`	Patton; Elizabeth Yamamoto
`\FName`, `\S`⟨*macro*⟩	George S.; Elizabeth Yamamoto

Later Use (`\ForceName`)

`\Name`	Patton; Elizabeth Yamamoto
`\FName`, `\S`⟨*macro*⟩	George S.; Elizabeth Yamamoto

Later Use (`\ForceFN`)

`\FName`, `\S`⟨*macro*⟩	Isoroku

Full Forms, Main Matter

`\NamesActive`

First Use (Default)

`\Name`	George S. Patton Jr. Elizabeth I Yamamoto Isoroku (cf. Section 2.8.1)

Later Use (* or \L⟨*macro*⟩)

`\Name*`	George S. Patton Jr. Elizabeth I Yamamoto Isoroku

Long, with `\DropAffix`

`\DropAffix\L`⟨*macro*⟩
George S. Patton

Short Forms, Main Matter

`\NamesActive`

Later Use (Default)

`\Name`	Patton; Elizabeth Yamamoto
`\FName`, `\S`⟨*macro*⟩	George S.; Elizabeth Yamamoto

Later Use (`\ForceName`)

`\Name`	Patton; Elizabeth Yamamoto
`\FName`, `\S`⟨*macro*⟩	George S.; Elizabeth Yamamoto

Later Use (`\ForceFN`)

`\FName`, `\S`⟨*macro*⟩	Isoroku

Naming system behavior

Since formatting is related closely to name usage (the existence of a name control sequence), we use similar descriptors for these distinct topics. A name's classification, elements, and control sequence are determined in the syntactic element layer. A name's displayed form is determined in the name display layer. Finally, a name's typographic form is determined by the format hook dispatcher and formatting hooks. Here is the general layout:

Syntactic Element Layer

`\@nameauth@Parse`: Determine name category, capitalization, punctuation, elements to be used, and pass on to...

Name Display Layer

`\@nameauth@NonWest`: Consult rules, construct displayed form.
`\@nameauth@West`: Consult rules, construct displayed form.
`\@nameauth@Form`: "Magic Eight Ball" of rules for name forms. Includes checking if a name control sequence exists, checking which is the calling macro, and so on. Pass the information to...

Format Hook Dispatcher

`\@nameauth@Hook`: Check the name to be printed for a final full stop. Check which naming system we are using. Call a formatting hook to print the name in the text.

`\@nameauth@Parse`: Instantiate name control sequence.

The different forms and formats seen on the previous page are a result of all three layers interacting. Sections 2.6 and 2.7 (with its subsections) deal mostly with the format hook dispatcher and formatting hooks, but they also interact with the name display layer. Section 2.8 (with its subsections) deals mostly with the syntactic element and name display layers, but also in some cases with the format hook dispatcher and formatting hooks. The following concepts flow from this layout and affect both form and formatting:

- **Name first use**
  - No name control sequence exists.
  - A name is printed with its long form (default).
  - The "first-use" formatting hook is used (default).
  - When the name is printed, a name control sequence is created.
- **Name subsequent use**
  - A name control sequence already exists.
  - A name is printed using a shorter form (default).
  - The "subsequent-use" formatting hook is used (default).

The parser and related macros create name forms and formats only in the text. Macros in name arguments affect both text and index (Section 2.7).

\NamesActive
\NamesInactive

Independent "main-matter" and "front-matter" systems are used to format first and subsequent name uses. `\NamesInactive` and the `frontmatter` option enable the front-matter system. `\NamesActive` switches names to the main-matter system. The `mainmatter` option is the default setting for names.

\global

These two macros can be used explicitly as a pair or singly within an explicit local scope. Use `\global` to force a global effect.

\NamesFormat
\MainNameHook
\FrontNamesFormat
\FrontNameHook

The main-matter system uses `\NamesFormat` to post-process first occurrences of names and `\MainNameHook` for subsequent uses. The front-matter system uses `\FrontNamesFormat` for first uses and `\FrontNameHook` for subsequent uses. The `alwaysformat` option causes only `\NamesFormat` and `\FrontNamesFormat` to be used.[19] Section 2.11.5 shows how name control sequences are keyed either to the main-matter system or to the front-matter system. The two formatting systems are distinct, useful for separate document elements. We color-code them below:

Name Pattern(s):
front-matter
`Rudolph!Carnap!NF`
`Nicolas!Malebranche!NF`
main-matter
`Rudolph!Carnap!MN`
`Nicolas!Malebranche!MN`

Front-matter system:	`\NamesInactive`
`\Name[Rudolph]{Carnap}`	Rudolph Carnap
`\Name[Rudolph]{Carnap}`	Carnap
`\Name[Nicolas]{Malebranche}`	Nicolas Malebranche
`\Name[Nicolas]{Malebranche}`	Malebranche

Main-matter system:	`\NamesActive`
`\Name[Rudolph]{Carnap}`	Rudolph Carnap
`\Name[Rudolph]{Carnap}`	Carnap
`\Name[Nicolas]{Malebranche}`	Nicolas Malebranche
`\Name[Nicolas]{Malebranche}`	Malebranche

We achieved that color coding using the following macros and the xcolor package:

```
\renewcommand*\FrontNamesFormat[1]{\color{red}\sffamily #1}
\renewcommand*\FrontNameHook[1]{\color{darkgray}\sffamily #1}
\renewcommand*\NamesFormat[1]{\color{blue}\sffamily #1}
\renewcommand*\MainNameHook[1]{\sffamily #1}
```

\ForceName
3.1

We show examples of `\ForceName` in Sections 2.8.1, 2.9, and 2.10.2. Use this prefix macro to force "first use" formatting for the next `\Name`, etc., but without changing any name control sequences. Thus:

Name Pattern(s):
`Rudolph!Carnap!MN`

`\Name[Rudolph]{Carnap}` Carnap
`\ForceName\Name[Rudolph]{Carnap}` Carnap

alwaysformat

Below we simulate `alwaysformat` via package internals:

- Front matter: Albert Einstein, Einstein; Confucius, Confucius.
  Patterns: `Albert!Einstein!NF Confucius!NF`
- Main matter: M.T. Cicero, Cicero; Elizabeth I, Elizabeth.
  Patterns: `M.T.!Cicero!MN Elizabeth,I!MN`

[19] The names of these macros grew from `\NamesFormat`, originally the only formatting hook.

Hook caveats

The name parser determines what syntactic name elements exist and how they are constituted. It passes that information to macros that determine the form of non-Western or Western names to be displayed. They in turn call the format hook dispatcher for post-processing, which calls the formatting hooks using the pattern:

`\bgroup`⟨*Hook*⟩`{#1}\egroup`.

Thus, one can create hooks that take either no arguments or one argument, e.g.:

```
\renewcommand*\NamesFormat{⟨content⟩}
\renewcommand*\NamesFormat[1]{⟨content⟩}
```

A hook that takes one argument can discard it and invoke `\NameParser` (Page 73). Due to the dispatcher design, both the following achieve the same effect, giving the choice of non-robust and robust forms:

```
\renewcommand*\NamesFormat{\itshape}
\renewcommand*\NamesFormat{\textit}
```

Applied to footnotes

The independent systems of names work with footnotes. Names in the body text, such as Adolf Harnack, normally affect name forms in the footnotes.[20] In footnote 20 `\MainNameHook` is called instead of `\NamesFormat` because Harnack already had occurred above. We can use the front-matter system to change that:

Name Pattern(s):
`Adolf!Harnack!MN`
`Adolf!Harnack!NF`

```
\begingroup
\makeatletter
\let\@oldfntext\@makefntext
\long\def\@makefntext#1{\NamesInactive\@oldfntext{#1}\NamesActive}
\makeatother
```

When we create another footnote, we see very different results.[21] Footnote 21 shows a different result. One can synchronize the two systems with `\ForgetThis` and `\SubvertThis` (Section 2.8.1). Below we revert footnotes with:

```
\makeatletter
\let\@makefntext\@oldfntext
\makeatother
```

Back to Section 1.3

Eyn Chriſten menſch iſt eyn freyer herr / über alle ding /
und niemande unterthan.
Eyn Chriſten menſch iſt eyn dienſtpar knecht aller ding
und yderman unterthan.

—Martin Luther, *Von der Freiheit eines Christenmenschen*

[20] We have Harnack from `\Harnack` instead of Adolf Harnack.
[21] We have Adolf Harnack from `\Harnack`, then Harnack.

2.7 Alternate Formatting

In this section, the name patterns do not fit in the margin.

The formatting hooks only affect names in the body text. Continental formatting occurs in both the text and in the index. One needs to format those names with macros in the name arguments.

We already saw the use of macros in name arguments for Roman names (page 31). We did not use alternate formatting there because we knew that we would not use an ⟨*SNN*⟩,⟨*Affix*⟩ pair; we would not use `\CapThis`; and we used `\noexpand` before the macros in the arguments.

In larger contexts, however, these constraints might not hold. The following scenarios, especially where there is segmentation of input, can be problematic:

- Using a comma-delimited required name argument pair as the argument of a robust macro like `\textsc` will halt LaTeX with errors. The nameauth macros will split that pair, which will break the robust macro:

  Bad `\Name{\textsc{`⟨*SNN*⟩`,`⟨*Affix*⟩`}}`
  Good `\Name{\textsc{`⟨*SNN*⟩`},\textsc{`⟨*Affix*⟩`}}`

  Not even alternate formatting can fix this issue; one simply must avoid the problem when encoding names.
- Using `\CapThis` with a name whose leading element in any one argument is neither a letter nor an active Unicode character, such as a macro. This may work in normal formatting or it may fail, depending on the macro. Use alternate formatting to have `\CapThis` activate the alternate capitalization mechanism.
- In a name argument, using a macro that contains a conditional statement can cause the macro to expand to different results. When such conditional changes occur, spurious results follow. Use `\noexpand` in the name arguments to fix this.

2.7.1 Basic Features

What we call "basic" alternate formatting is meant to be a temporary stop on the road to "advanced" usage, which is more robust. For reasons shown above, it is quite helpful to use `\noexpand` before macros in name arguments:

```
\Name[Martin]{\textSC{Luther}}            % basic; good
\Name[Martin]{\noexpand\textSC{Luther}}   % advanced; better
```

The reason why the basic features work is because the built-in formatting macros discussed below have their effects set when alternate formatting is enabled. If one does not use the macros in names apart from alternate formatting and one does not change the Boolean flags that govern them, the basic features may suffice.

Enabling and Disabling

The first thing we need to know is how to enable and disable alternate formatting. The macros that accomplish this are global in scope and cannot be isolated in a local scope. Normally, they are used in pairs, except with the `altformat` option.

At the start of this section we used `\AltFormatActive` to enable alternate formatting and activate the formatting macros (see below). At the end of this section we must use `\AltFormatInactive` to deactivate and disable them.

\AltFormatActive

Both the `altformat` option and `\AltFormatActive` enable and activate alternate formatting. Both cause `\CapThis` to work via `\AltCaps` instead of the normal way. `\AltFormatActive` countermands `\AltFormatActive*`.

- **Enabled** means that the alternate formatting mechanism inhibits the normal behavior of `\CapThis`.
- **Disabled** means that the normal behavior of `\CapThis` is again in force and alternate formatting is inhibited.
- **Activated** means that `\textSC` and other alternate formatting macros below format their arguments.
- **Deactivated** means that `\textSC` and other macros below do not format their arguments.

\AltFormatActive*

The starred form `\AltFormatActive*` enables alternate formatting but deactivates the special formatting macros, preventing them from changing their arguments. It countermands both the `altformat` option and `\AltFormatActive`. It causes `\CapThis` only to work via `\AltCaps`.

\AltFormatInactive

This macro both disables and deactivates alternate formatting. This reverts globally to standard formatting and the normal function of `\CapThis`.

Macro	Enabled	Activated
`\AltFormatActive`	■	■
`\AltFormatActive*`	■	
`\AltFormatInactive`		

All three macros **always** make global changes.
This differs from other macros in nameauth.

Basic Alternate Formatting

\textSC
\textIT
\textBF
\textUC

Continental formatting can be as simple as using the short macro `\textSC`. Three other macros also implement alternate formatting. These macros make changes only when alternate formatting is active. We sort the index entry with `\PretagName` (Section 2.4.2):

```
1 \PretagName[Greta]{\textSC{Garbo}}{Garbo, Greta}
2 \PretagName[Ada]{\textIT{Lovelace}}{Lovelace, Ada}
3 \PretagName[Charles]{\textBF{Babbage}}{Babbage, Charles}
4 \PretagName{\textUC{Tokugawa}, Ieyasu}{Tokugawa Ieyasu}
```

```
\Name[Greta]{\textSC{Garbo}} . . . . . Greta Garbo; Garbo
\Name[Ada]{\textIT{Lovelace}} . . . . . Ada Lovelace; Lovelace
\Name[Charles]{\textBF{Babbage}} . . . . . Charles Babbage; Babbage
\Name{\textUC{Tokugawa}, Ieyasu} . . . TOKUGAWA Ieyasu; TOKUGAWA
```

Font substitutions might occur with these macros.[22] They **always format their arguments** when using the `altformat` option or after `\AltFormatActive`.

[22] Since we switch to Latin Modern Sans in the formatting hooks, the switch to small caps forces a substitution to Latin Modern Roman. This action varies with the font being used.

Likewise, they **never format their arguments** when `\AltFormatActive*` is used. Still, `\CapName`, `\RevComma`, and `\RevName` can modify the effects of alternate formatting, but only in the text, not the index:

`\CapName\Name*[Greta]{\textSC{Garbo}}`	Greta GARBO
`\RevComma\Name*[Ada]{\textIT{Lovelace}}`	*Lovelace*, Ada
`\RevName\Name*{\textUC{Tokugawa}, Ieyasu}`	Ieyasu TOKUGAWA

Here is a more practical example on how to format the required argument {⟨*SNN, Affix*⟩} as two separate arguments:

```
\PretagName[J.D.]{\textSC{Rockefeller},\textSC{III}}
  {Rockefeller, John D 3}
\PretagName{\textUC{Fukuyama}, Takeshi}{Fukuyama Takeshi}
\begin{nameauth}
  \< JRIII & J.D. & \textSC{Rockefeller},\textSC{III} & >
  \< Fukuyama  & & \textUC{Fukuyama}, Takeshi &         >
\end{nameauth}
```

From above we get J.D. Rockefeller III, then Rockefeller. This works also for non-Western names. Applying the macros above, we get:

`\Fukuyama`	FUKUYAMA Takeshi
`\Fukuyama`	FUKUYAMA

Only the new syntax allows one to use alternate names in the text. For example, "`\LFukuyama[Sensei]` FUKUYAMA Sensei wrote *Nihon Fukuin Rūteru Kyōkai Shi* in 1954, after studying in the US in the 1930s".

Using names designed for alternate formatting also with regular formatting will produce inconsistent formatting and spurious index entries.

Example Reference Work

This example uses basic Continental formatting. We build on our knowledge so far to construct head-words and articles. Below we use alternate formatting, sort index entries, set up a cross-reference, and define an article macro:

```
\PretagName[Greta]{\textSC{Garbo}}{Garbo, Greta}
\PretagName{\textSC{Misora}, Hibari}{Misora Hibari}
\PretagName[Heinz]{\textSC{Rühmann}}{Ruehmann, Heinz}
\PretagName[Heinrich Wilhelm]{\textSC{Rühmann}}%
  {Ruehmann, Heinrich Wilhelm}
\IndexRef[Heinrich Wilhelm]{\textSC{Rühmann}}%
  {\textSC{Rühmann}, Heinz}%

\newcommand{\RefArticle}[3]{%
  \def\check{#2}%
  \ifx\check\empty
    \noindent\ForgetThis#1\ #3
  \else
    \noindent\ForgetThis#1\ #2\ #3
  \fi
}
```

`\RefArticle` either formats the name from the first argument and appends the third argument, ignoring the the second if it is empty, or it formats the first two arguments and appends the third. We determine what those arguments mean by including specific naming macros. That includes using `\RevComma` for Western names, but not for Eastern ones.

```
\RefArticle
  {\RevComma\Name[Greta]{\textSC{Garbo}}}
  {}
  {(18 September 1905\,--\,15 April 1990) was a Swedish
   film actress during the 1920s and 1930s.}

\RefArticle
  {\Name{\textSC{Misora}, Hibari}}
  {}
  {(W: ``\RevName\Name*{\textSC{Misora}, Hibari}'';
   29 May 1937\,--\,24 June 1989) was a Japanese singer
   and actress noted for her positive message.}

\RefArticle
  {\RevComma\Name[Heinrich Wilhelm]{\textSC{Rühmann}}}
  {``\SubvertThis\ForceName\FName[Heinz]{\textSC{Rühmann}}''}
  {(7 March 1902\,--\,3 October 1994) was a German actor
   in over 100 films.}
\AltFormatInactive
```

GARBO, Greta (18 September 1905 – 15 April 1990) was a Swedish film actress during the 1920s and 1930s.

MISORA Hibari (W: "Hibari MISORA"; 29 May 1937 – 24 June 1989) was a Japanese singer and actress noted for her positive message.

RÜHMANN, Heinrich Wilhelm "Heinz" (7 March 1902 – 3 October 1994) was a German actor in over 100 films.

2.7.2 Advanced Formatting Features

Advanced features involve a dance of sorts between the name argument macros and the formatting hooks. We are moving from immutability to changeability. Yet so far, we have heard that this could change conditional statements and create spurious index entries.

Using `\noexpand` befor macros in name arguments that expand conditionally will solve this problem for us:

- The use of `\noexpand` isolates the global state of `\textSC` and the other formatting macros from local changes in the formatting hooks.
- Indexing never occurs within the formatting hooks.
- Special triggering macros in the formatting hooks isolate local changes.

As a result of these points, advanced alternate formatting works as expected. It uses a similar approach as did the Boolean flags for Roman names (page 31).

Using `\noexpand` is key to consistent index entries.

Alternate Capitalization

\AltCaps When alternate formatting is enabled, \CapThis causes \AltCaps to format its argument only in a formatting hook. It is enabled whenever alternate formatting is enabled, but it works independently of \AltOn and \AltOff below:

\noexpand\AltCaps{⟨*Arg*⟩}

We introduce this macro with a silly example, disabling indexing in the process:

```
\IndexInactive
What's in \Name[\noexpand\AltCaps{a}]{Name}?
\CapThis\Name*[\noexpand\AltCaps{a}]{Name} smells not,
but a rose does, even if it has a
\Name[\noexpand\AltCaps{a}]{Name}.
```

What's in a Name? A Name smells not, but a rose does, even if it has a Name.

Advanced Alternate Formatting

Advanced features come with "some assembly required". An author must put one of the trigger macros below into one or more of the nameauth formatting hooks. These macros change the state of the formatting macros.

\AltOff Vaguely reminiscent of an automobile's manual clutch and gearbox, \AltOff deactivates \textSC, \textBF, \textIT, and \textUC only in a formatting hook.

\AltOn Likewise, \AltOn activates \textSC, \textBF, \textIT, and \textUC only in a formatting hook.

Continuing the example (indexing suppressed), we redefine \MainNameHook to suppress small caps. Note the copious use of \noexpand:

```
\renewcommand*\MainNameHook%
  {\sffamily\AltOff}% we match the manual

\CapThis\Name
  [\noexpand\textSC{\noexpand\AltCaps{a} Name}]
  {\noexpand\textSC{Problem}}
will not become a \Name
  [\noexpand\textSC{\noexpand\AltCaps{a} Name}]
  {\noexpand\textSC{Problem}},
even if it smells like a rose.
\IndexActive
```

A Name Problem will not become a Problem, even if it smells like a rose.

Now indexing is active again. Using the same \MainNameHook defined above, consider the following in a document preamble:

```
\begin{nameauth}
  \< Luth & Martin & \noexpand\textSC{Luther} & >
  \< Cath & Catherine \noexpand\AltCaps{d}e'
          & \noexpand\textSC{Medici} & >
\end{nameauth}
\PretagName[Martin]{\noexpand\textSC{Luther}}{Luther, Martin}
\PretagName[Catherine \noexpand\AltCaps{d}e']
            {\noexpand\textSC{Medici}}{Medici, Catherine de}
```

In the body text, we see \ForgetThis\Luth Martin Luther and \Luth Luther. Likewise \Cath Catherine de' Medici, then \Cath Medici. Medieval Italian differs from modern Italian with respect to particles. To get de' Medici in the text, use \LCath[\noexpand\AltCaps{d}e'], with which \CapThis also works. The index entry should be "Medici, Catherine de' ", instead of "de' Medici, Catherine".

Name inflections

We can design grammatical inflections either with or without alternate formatting. \DoGentrue occurs only in the formatting hook, keeping the index entries consistent via \noexpand. Highly inflected languages require two Boolean flags per case and nested conditional statements.[23] In the example below we do not use the formatting in this manual and we hide the special tags used herein. In the preamble of a document we would have:

```
\newif\ifGenitive
\newif\ifDoGen
\newcommand*\JEFF{\ifDoGen\textSC{Jefferson's}\else
  \textSC{Jefferson}\fi}
\begin{nameauth}
  \< Jeff & Thomas & \noexpand\JEFF & >
\end{nameauth}
\PretagName[Thomas]{\noexpand\JEFF}{Jefferson, Thomas}
\TagName[Thomas]{\noexpand\JEFF}{, pres.}
```

In the preamble or document text, we can have:

```
\renewcommand*\NamesFormat[1]
  {\ifGenitive\DoGentrue\fi#1\global\Genitivefalse}
\renewcommand*\MainNameHook[1]
  {\ifGenitive\DoGentrue\fi\AltOff#1\global\Genitivefalse}

Consider \Genitivetrue\Jeff\ legacy. More on \Jeff\ later.
\Genitivetrue\Jeff\ reputation has declined in recent decades.
```

Consider Thomas Jefferson's legacy. More on Jefferson later. Jefferson's reputation has declined in recent decades.

Now we end the scope, resume normal formatting, and do not use the names in this section outside of it.[24]

Back to Section 1.3

Perhaps one can use the nameauth package...

> There in the ring where name and image meet
>
> —W.H. Auden, "Perhaps"

[23] We hide the same information that we did on page 30. A copy of this example is in examples.tex, located with this manual.

[24] In a dtx file it is best to put the nameauth environment, \PretagName, and \TagName macros in the driver section, especially when names contain macros.

2.8 Name Decisions

The macros in this section force and detect name states. Below we keep names consistent with beamer overlays using some of the macros explained in this section. Otherwise, name forms will change as one advances the slides:[25]

```
\documentclass{beamer}
\usepackage{nameauth}
\mode<presentation>
\beamerdefaultoverlayspecification{<+->}

\begin{document}

\begin{frame}{Move Text Without Retyping Names}
  \begin{itemize}\footnotesize
  \item<1-> Original\ForgetName[George]{Washington}%
                    \ForgetName[George]{Washington's}\\
            This version of \Name[Ulysses S.]{Grant} changes.
  \begin{enumerate}
  \item<2-> \IfMainName[George]{Washington's}{He}%
            {\Name[George]{Washington}}
            became the first president
            of the United States.
  \item<3-> \IfMainName[George]{Washington}{His}%
            {\SkipIndex\Name*[George]{Washington's}}
            military successes during the Seven Years War
            readied him to command the army
            of the Continental Congress.
  \end{enumerate}
  \item<1-> Reordered\ForgetName[George]{Washington}%
                     \ForgetName[George]{Washington's}\\
            This version of \ForgetThis\Name[Ulysses S.]{Grant}
            does not change.
  \begin{enumerate}
  \item<3-> \IfMainName[George]{Washington}{His}%
            {\SkipIndex\Name*[George]{Washington's}}
            military successes during the Seven Years War
            readied him to command the army
            of the Continental Congress.
  \item<2-> \IfMainName[George]{Washington's}{He}%
            {\Name[George]{Washington}}
            became the first president
            of the United States.
  \end{enumerate}
  \end{itemize}
\end{frame}

\end{document}
```

The overlays, numbered progressively from one to three, begin by deleting name control sequence patterns. Uncontrolled names will change. Name conditionals ensure specific, context-dependent forms based on what name has appeared. These conditionals allow the text to be order-independent.

[25] A copy of this example is in `examples.tex`, located with this manual.

2.8.1 Making Decisions

By default, the macros below produce global effects. They change both the `!MN` and `!NF` data sets (Section 2.11.5). With `\ForceName` (Section 2.6), they change formatting. Apart from `\ForceName`, they also change long or short name forms and the outcome of both indexing error protection (Section 2.4.1) and the name testing macros (Section 2.8.2).

`\ForgetName` This macro takes the same arguments as `\Name`, but it prints no output. It "forgets" a name, forcing a "pre-first use" state that will trigger a first-time name use the next time a naming macro makes reference to the name:

`\ForgetName[`⟨*FNN*⟩`]{`⟨*SNN, Affix*⟩`}[`⟨*Alternate*⟩`]`

This macro "unprotects" names like `\IncludeName*` "unprotects" cross-references, allowing one to make a *see* reference to a name, even if that name already has index page references. If one is using a single name index, that could be an error. If one is using multiple indexes for names, however, that could be necessary.

`\SubvertName` This macro takes the same arguments as `\Name`, but it produces no output in the text. It "subverts" a name by creating a name pattern control sequence, forcing a "subsequent use", and "protecting" a name from being used as a *see* reference (similar to `\ExcludeName` and cross-references):

`\SubvertName[`⟨*FNN*⟩`]{`⟨*SNN, Affix*⟩`}[`⟨*Alternate*⟩`]`

`\ForgetThis` **3.1** This prefix macro causes the next instance of a naming macro or shorthand to "forget" a name before printing it. After knowing `\Einstein` "Einstein" we forget him and again have a first reference: `\ForgetThis\Einstein` "Albert Einstein".

This macro does not affect the index unless one uses `\ForgetThis` in a situation where a naming macro produces no output in the text. That results in the same outcome as a careless use of `\ForgetName`:

Page	Macro	Index Result
10	`\ForgetThis\JustIndex\Einstein`	Einstein, Albert, 10
12	`\IndexRef[Albert]{Einstein}%` `{Smart Dude}`	Einstein, Albert, 10, *see* Smart Dude

`\SubvertThis` **3.1** This prefix macro causes the next instance of a naming macro or shorthand to "subvert" a name before printing it. As indicated in the table on page 20, `\ForgetThis` has a higher priority than `\SubvertThis` and negates it. The caveat regarding naming macro output applies to both `\SubvertThis` and `\ForgetThis`.

We advise one to avoid using `\ForgetThis` and `\SubvertThis` before any naming macro that produces no output in the text.

Name Decision and Formatting Summary

First Use: `\Bailey` . . . . . Betsey Bailey
Name control sequence (cseq) undefined, then created with text output. Index *see* protection set. Default name form (long). Default first-use format hooks.

Later Use: `\Bailey` . . . . . Bailey
No change to extant name cseq. No change to index *see* protection. Default name form (short). Default later-use format hooks.

Forgotten: `\ForgetName[Betsey]{Bailey}` . . . . . (no output)
Name cseq deleted. Index *see* protection removed. Force first use. Next use of `\Bailey` will be a first use.

Subverted: `\SubvertName[Betsey]{Bailey}` . . . . . (no output)
Name cseq created. Index *see* protection set. Force later use. Next use of `\Bailey` will be a later use.

`\ForgetThis`: `\ForgetThis\Bailey` . . . . . Betsey Bailey
Name cseq deleted, then created. Index *see* protection removed, then set. Force first use. Default form (long). Default first-use format hooks. Next use of `\Bailey` will be a later use if output occurs.

`\SubvertThis`: `\SubvertThis\Bailey` . . . . . Bailey
Name cseq created if undefined. Index *see* protection set. Force later use. Default form (short). Default later-use format hooks. Next use of `\Bailey` will be a later use if output occurs.

Long Use: `\LBailey` . . . . . Betsey Bailey
No change to extant name cseq. No change to index *see* protection. **Force long name form.** No change to format hooks.

"Short", first: `\ForgetThis\SBailey` . . . . . Betsey Bailey
Name cseq deleted, then created. Index *see* protection removed, then set. First use forces long form. Default first-use format hooks.

Short, later: `\SBailey` . . . . . Betsey
No change to extant name cseq. No change to index *see* protection. **Force first name or short name.** No change to format hooks.

Forced, later: `\ForceName\Bailey` . . . . . Bailey
No change to extant name cseq. No change to index *see* protection. Default name form. **Force first-use hooks.**

Forced, long: `\ForceName\LBailey` . . . . . Betsey Bailey
No change to extant name cseq. No change to index *see* protection. **Force long name form. Force first-use hooks.**

Forced, short: `\ForceName\SBailey` . . . . . Betsey
No change to extant name cseq. No change to index *see* protection. **Force first name or short name if subsequent use. Force first-use hooks.**

`\LocalNames`
`\GlobalNames`

By default, `\ForgetName`, `\SubvertName`, `\ForgetThis`, and `\SubvertThis` are not limited either by scope or by the active naming system. `\LocalNames` restricts the effects of these macros to the current naming system, but not to scope. `\GlobalNames` restores the default behavior that affects both systems. Both macros always have global scope.

To see how these two macros work, in the following example we define a macro that reports whether or not `\Name[Charlie]{Chaplin}` exists. This macro gives four possible results: the name exists in the main matter, it exists in the front matter, it exists in both systems, or it does not exist (see Section 2.8.2):

```
\def\CheckChuck{{\bfseries\IfFrontName[Charlie]{Chaplin}%
  {\IfMainName[Charlie]{Chaplin}{both}{front}}%
  {\IfMainName[Charlie]{Chaplin}{main}{none}}}}
```

We start in the "main-matter" system with no extant name:

```
\CheckChuck . . . . . none
```

We create a name in the "main matter":

```
\Name[Charlie]{Chaplin} . . . . . Charlie Chaplin
\CheckChuck . . . . . main
```

We switch to the "front-matter" system and create a name, but since we are using the `quote` environment, we add `\global`:

```
\global\NamesInactive
\Name[Charlie]{Chaplin} . . . . . Charlie Chaplin
\CheckChuck . . . . . both
```

We now have two names. Their patterns are: `Charlie!Chaplin!MN` for the main matter and `Charlie!Chaplin!NF` for the front matter (Section 2.11.5).

We use `\LocalNames` to make both `\ForgetName` and `\SubvertName` work with only the current system. When we "forget" the name, the current system is front matter, so we forget the front-matter name:

```
\LocalNames
\ForgetName[Charlie]{Chaplin}
\CheckChuck . . . . . main
```

Next we "subvert" the front-matter name to "remember" it again and switch to main matter, again using `\global` to ignore scoping.

```
\SubvertName[Charlie]{Chaplin}
\global\NamesActive
\CheckChuck . . . . . both
```

Now the current system is main matter. We then forget the main-matter name only. Additionally, we use `\GlobalNames` to reset the default behavior so that `\ForgetName` and `\SubvertName` work with both systems again:

```
\ForgetName[Charlie]{Chaplin}
\GlobalNames
\CheckChuck . . . . . front
```

Finally, we forget everything. Even though we are in a main-matter section, the front-matter name also goes away:

```
\ForgetName[Charlie]{Chaplin}
\CheckChuck . . . . . none
```

Back to Section 1.3

2.8.2 Testing Decisions

The macros in this section test for the presence or absence of a name, then expand to a result based on the outcome of the test.

\GlobalNameTest
\LocalNameTest
3.5

The default behavior encapsulates the decision paths in a local scope, insulating any changes therein. If this is not desired, use the `globaltest` option or `\GlobalNameTest`. `\LocalNameTest` will enable it again. These commands affect assignment statements in test paths. By default, one must explicitly use `\global` when desired. See also the example below.

\IfMainName

In order to test whether or not a "main matter" name control sequence exists, use this long macro that can accommodate paragraph breaks:

`\IfMainName[`⟨*FNN*⟩`]{`⟨*SNN, Affix*⟩`}[`⟨*Alternate*⟩`]{`⟨*yes*⟩`}{`⟨*no*⟩`}`

For example we have not encountered `\Name[Bob]{Hope}` yet. Using `\IndexName` does not affect the tests in this section. We could do the following test that will reflect whether or not the name is present:

```
I heard someone say: \IfMainName[Bob]{Hope}
  {Bob here!}
  {No Bob here.}\IndexName[Bob]{Hope}
```

I heard someone say: No Bob.

Now we test for `\Name{Elizabeth,I}`, a name that has occurred, and we also show the difference between local and global test paths:

```
\GlobalNameTest
\def\msg{We are unsure about \LEliz}
\IfMainName{Elizabeth,I}
  {\def\msg{We really do know of \LEliz}}
  {\def\msg{We do not know of \LEliz}}
  \msg\quad (\cmd{\GlobalNameTest}).

\LocalNameTest
\def\msg{We are unsure about \LEliz}
\IfMainName{Elizabeth,I}
  {\def\msg{We really do know of \LEliz}}
  {\def\msg{We do not know of \LEliz}}
  \msg\quad (\cmd{\LocalNameTest}).
```

We really do know of Elizabeth I (`\GlobalNameTest`).

We are unsure about Elizabeth I (`\LocalNameTest`).

We see that the default keeps local any assignments made in the test paths.

\IfFrontName

In order to test whether or not a "front matter" name control sequence exists, use this long macro that can accommodate paragraph breaks. Its syntax is:

`\IfFrontName[`⟨*FNN*⟩`]{`⟨*SNN, Affix*⟩`}[`⟨*Alternate*⟩`]{`⟨*yes*⟩`}{`⟨*no*⟩`}`

This macro works just like `\IfMainName`, except using the "front matter" name control sequences as the test subject. These testing macros prove their worth especially through combination. For example, on the next page we do a test based on Section 2.6.

```
\IfFrontName[Rudolph]{Carnap}
{%
  \IfMainName[Rudolph]{Carnap}
    {\Name[Rudolph]{Carnap} is in both main- and front-matter text.}
    {\Name[Rudolph]{Carnap} is only in front-matter text.}
}%
{%
  \IfMainName[Rudolph]{Carnap}
    {\Name[Rudolph]{Carnap} is only in main-matter text.}
    {\Name[Rudolph]{Carnap} has not been mentioned.}
}
```

Carnap is in both main- and front-matter text.

`\IfAKA` This macro tests whether or not a regular or excluded form of cross-reference control sequence exists. The syntax is:

`\IfAKA[`⟨*FNN*⟩`]{`⟨*SNN, Affix*⟩`}[`⟨*Alternate*⟩`]{`⟨*y*⟩`}{`⟨*n*⟩`}{`⟨*excl*⟩`}`

This macro also works like `\IfMainName`, except that it has an additional ⟨*excl*⟩ branch in order to detect the activity of `\ExcludeName` (Section 2.4.1).

3.5 Cross-references are governed by name control sequences ending in `!PN` (Section 2.11.5).

- Excluded control sequences (the ⟨*excl*⟩ path) expand to the value of `\@nameauth@Exclude`.
- Regular cross-references (the ⟨*y*⟩ path) do not expand to that value. At present, they are empty.
- `\ExcludeName` creates excluded xrefs. `\IncludeName` destroys them.
- Regular xrefs are created by `\IndexRef`, `\AKA`, `\PName` and their starred forms. Regular xrefs are destroyed by `\IncludeName*`.

Based on the known facts above, here we offer some examples about how to use this logic:

Name Pattern(s):
`Jesse!Ventura!MN`
`James!Janos!PN`
`James!Janos!MN`

1. In the text we refer to former pro-wrestler and Minnesota governor Jesse Ventura, `\Name[Jesse]{Ventura}`.
2. We establish his lesser-known legal name as an alias: "James Janos", `\IndexRef[James]{Janos}{Ventura, Jesse}\Name[James]{Janos}`.
3. We get the result: "Jesse Ventura is a stage name". If we do not use `\ExcludeName`, we can leave the ⟨*excl*⟩ branch empty:

```
\IfAKA[James]{Janos}%
  {\Name*[Jesse]{Ventura} is a stage name}%
  {\Name*[Jesse]{Ventura} is a regular name}%
  {}
```

Otherwise, based on Section 2.4.1, we get: "Grinch is excluded":

Name Pattern(s):
`Grinch!PN`

```
``\IfAKA{Grinch}%
  {\Name{Grinch} is an alias}%
  {\Name{Grinch} is not an alias}%
  {\Name{Grinch} is excluded}''
```

We can combine all these macros create a complete test:

```
\IfAKA[FNN]{SNN, Affix}[Alternate]%
  {true; it is a pseudonym}%
  {% if not a pseudonym:
    \IfFrontName[FNN]{SNN, Affix}[Alternate]% yes path
    {\IfMainName[FNN]{SNN, Affix}[Alternate]%
      {both}%
      {front}%
    }%
    {\IfMainName[FNN]{SNN, Affix}[Alternate]% no path
      {main}%
      {does not exist}%
    }%
  }%
  {excluded path}
```

Sync with floats

If one uses names in floats and in the text, the testing macros can synchronize a float with the text. Assume that we want to print a full name for whichever appears first, like Johann Wolfgang von Goethe instead of J.W. von Goethe. In both the text and the float we use the following:

Name Pattern(s):
J.W.von!Goethe!MN

```
\IfMainName[J.W. von]{Goethe}
  {\Name*[J.W. von]{Goethe}}
  {\Name*[J.W. von]{Goethe}[Johann Wolfgang von]}
```

Sync with events

Text tags can work with the conditional macros to prevent anachronistic references. This aids working with history texts, game books, and so on. One must avoid unbounded recursion that results in a stack overflow (Section 2.5):

Name Pattern(s):
Paul!PN
Saul,ofTarsus!DB
Jesus,Christ!MN
Lucius!SergiusPaulus!MN
Paul!MN
Saul,ofTarsus!MN

```
\IndexRef{Paul}{Saul of Tarsus}
\NameAddInfo{Saul, of Tarsus}
  {\IfMainName{Jesus, Christ}
    {\IfMainName[Lucius]{Sergius Paulus}
      {renamed himself \Name{Paul}}
      {a preacher to the Gentiles}}
    {wrote that he persecuted Christians}}
\Name{Saul, of Tarsus} \NameQueryInfo{Saul, of Tarsus}.
He saw a vision of \Name{Jesus, Christ} on the road to Damascus.
\Name{Saul, of Tarsus} became \NameQueryInfo{Saul, of Tarsus}.
After converting \Name[Lucius]{Sergius Paulus},
\Name{Saul, of Tarsus} \NameQueryInfo{Saul, of Tarsus}.
```

Saul of Tarsus wrote that he persecuted Christians. He saw a vision of Jesus Christ on the road to Damascus. Saul became a preacher to the Gentiles. After converting Lucius Sergius Paulus, Saul renamed himself Paul.

Caveats

Using these tests inside other macros or passing control sequences to them may create false results (see *The TEXbook*, 212–15). That is why nameauth uses token registers to save name arguments (Section 2.10.2. Consider using `\noexpand` in macros passed as name arguments and see also Section 2.11.6. Using the trace package, `\show`, or `\meaning` can help one mitigate problems.

Back to Section 1.3

2.9 Alternate Name Macros

3.0 The macros in this section predate `\IndexRef` and are less flexible than just using `\IndexRef` with `\Name` (cf. page 39). We recommend the macros below only for backward compatibility, or if one likes using them. To save space, we show the syntax of these macros using ⟨*SAFX*⟩ as the equivalent of ⟨*SNN, Affix*⟩.

- These macros do not create page references.
- Opposite of `\IndexRef`, the target `[`⟨*FNN*⟩`]{`⟨*SAFX*⟩`}` comes first; then the cross-reference `[`⟨*xref FNN*⟩`]{`⟨*xref SAFX*⟩`}[`⟨*xref Alternate*⟩`]`.
- The obsolete syntax cannot be used with `[`⟨*FNN*⟩`]{`⟨*SAFX*⟩`}`; it can be used with `[`⟨*xref FNN*⟩`]{`⟨*xref SAFX*⟩`}[`⟨*xref Alternate*⟩`]`.
- Only ⟨*SAFX*⟩ and ⟨*xref SAFX*⟩ can have comma-delimited suffixes.
- One cannot use `\TagName` with a cross-reference, but one can sort it with `\PretagName[`⟨*xref FNN*⟩`]{`⟨*xref SAFX*⟩`}{`⟨*sort tag*⟩`}`.

`\AKA` `\AKA*` `\AKA` (*also known as*) and its starred form display an alias in the text and create a cross-reference in the index. They format names differently than `\Name`, etc.:

> `\AKA [`⟨*FNN*⟩`]{`⟨*SAFX*⟩`}[`⟨*xref FNN*⟩`]{`⟨*xref SAFX*⟩`}[`⟨*xref Alternate*⟩`]`
> `\AKA*[`⟨*FNN*⟩`]{`⟨*SAFX*⟩`}[`⟨*xref FNN*⟩`]{`⟨*xref SAFX*⟩`}[`⟨*xref Alternate*⟩`]`

- Both macros create a cross-reference in the index from the ⟨*xref FNN*⟩, ⟨*xref SAFX*⟩, and ⟨*xref Alternate*⟩ arguments to a target defined by ⟨*FNN*⟩ and ⟨*SAFX*⟩.
- The order of the name and cross-reference in `\AKA` is opposite that of `\IndexRef` in order to avoid ambiguity.
- `\AKA` prints a long form of the cross-reference name in the text. `\SeeAlso` works with `\AKA`, `\AKA*`, `\PName`, and `\PName*`.
- `\AKA` prints the ⟨*xref FNN*⟩ and ⟨*xref SAFX*⟩ arguments in the text.
- If ⟨*xref Alternate*⟩ is present with ⟨*xref FNN*⟩, or if ⟨*xref Affix*⟩ when ⟨*xref FNN*⟩ is absent, `\AKA` swaps them only in the text.
- If ⟨*xref Alternate*⟩ is present without ⟨*xref FNN*⟩ and ⟨*xref Affix*⟩, the obsolete syntax is used.

3.0
- `\AKA*` is analogous to `\FName` and `\ForceFN` works with it. The `oldAKA` option implies `\ForceFN` with every use of `\AKA*`.

3.5
- Neither `\AKA` nor its derivatives permit the effects of `\ForgetThis` and `\SubvertThis` to "pass through" because they produce output in the text. The `oldreset` option negates this.

We make cross-references to Bob Hope; all of the forms below will create the cross-reference "Hope, Leslie Townes *see* Hope, Bob":

Name Pattern(s):
`Bob!Hope!MN`
`LeslieTownes!Hope!PN`

`\AKA[Bob]{Hope}[Leslie Townes]{Hope}`	Leslie Townes Hope
`\RevComma\AKA[Bob]{Hope}[Leslie Townes]{Hope}`	Hope, Leslie Townes
`\AKA[Bob]{Hope}[Leslie Townes]{Hope}[Lester T.]`	Lester T. Hope
`\AKA*[Bob]{Hope}[Leslie Townes]{Hope}`	Leslie Townes
`\AKA*[Bob]{Hope}[Leslie Townes]{Hope}[Lester]`	Lester

Next we refer to Louis XIV, and Lao-tzu, as well as Lafcadio Hearn and Charles du Fresne. The caps and reversing macros work. Even with `\ForceName`, the formatting hooks do not change. `\AKA` and its derivatives use only `\MainNamesHook` and `\FrontNamesHook`.

Name Pattern(s):
```
Louis,XIV!MN
Lao-tzu!MN
Lafcadio!Hearn!MN
Charles!du~Fresne!MN
SunKing!PN
Li,Er!PN
du~Cange!PN
Koizumi,Yakumo!PN
```

`\AKA{Louis, XIV}{Sun King}`	Sun King
`\ForceName\AKA{Louis, XIV}{Sun King}`	Sun King
`\AKA*{Louis, XIV}{Sun King}`	Sun King
`\AKA{Lao-tzu}{Li, Er}`	Li Er
`\AKA*{Lao-tzu}{Li, Er}`	Li
`\AKA[Charles]{du~Fresne}{du~Cange}`	du Cange
`\CapThis\AKA[Charles]{du~Fresne}{du~Cange}`	Du Cange
`\CapName\AKA[Lafcadio]{Hearn}{Koizumi, Yakumo}`	KOIZUMI Yakumo
`\RevName\AKA[Lafcadio]{Hearn}{Koizumi, Yakumo}`	Yakumo Koizumi

`formatAKA`

In order to format cross-references like names, either avoid using these older macros or use the `formatAKA` option. That allows `\ForceName` to work properly, but cross-references use their own system for being "first" (Section 2.11.5). We simulate `formatAKA` and use `\AKA{Elizabeth,I}[Good Queen]{Bess}`:

Name Pattern(s):
```
Elizabeth,I!NF
GoodQueen!Bess!PN
Elizabeth,I!MN
```

Front Matter:	Elizabeth I	Elizabeth	Good Queen Bess
Main Matter:	Elizabeth I	Elizabeth	Good Queen Bess
Using `\ForceName`:			Good Queen Bess

The first appearance of Good Queen Bess uses either `\NamesFormat` or `\FrontNamesFormat`. After that, only `\mainNameHook` and `\FrontNameHook` can be used without `\ForceName`, which triggers the first-use hooks.

`alwaysformat`

Below we compare the behavior of the `alwaysformat` option, where all regular names use only `\NamesFormat` and `\FrontNamesFormat`:

> Elizabeth I was known as "Good Queen Bess". Again we mention Queen Elizabeth, "Good Queen Bess". Using `\ForceName`: Good Queen Bess.

`\PName`
`\PName*`

These convenience macros (an early feature) print a main name and a cross-reference in parentheses:

`\PName [`⟨*FNN*⟩`]{`⟨*SAFX*⟩`}[`⟨*xref FNN*⟩`]{`⟨*xref SAFX*⟩`}[`⟨*xref Alternate*⟩`]`
`\PName*[`⟨*FNN*⟩`]{`⟨*SAFX*⟩`}[`⟨*xref FNN*⟩`]{`⟨*xref SAFX*⟩`}[`⟨*xref Alternate*⟩`]`

The starred form `\PName*` is like the starred form `\Name*` to the extent that it prints a long form of ⟨*FNN*⟩⟨*SAFX*⟩. It does not affect the cross-reference arguments ⟨*xref FNN*⟩⟨*xref SAFX*⟩⟨*xref Alternate*⟩.

- Most prefix macros only affect ⟨*FNN*⟩⟨*SAFX*⟩, not the cross-reference, which always has a long form.
- `\SkipIndex` keeps both names out of the index.
- `\PName` allows the obsolete syntax only for the alternate name.
- Even though it is permitted, please avoid using the obsolete syntax with the xref part of `\PName`, such as `\PName{Lao-tzu}{Li}[Er]` and `\PName{William, I}{William}[the Conqueror]`.

Alternate names for the non-Western syntax do not work with the name portion of `\PName`; only the cross-reference portion can support it. If we attempted to use `\SkipIndex\PName*{William, I}[William]{the Conqueror}`, this macro would put "William I (William the Conqueror)" in the body text, but its index entry would be incorrect: "the Conqueror, William *see* William I". We use `\ForgetName{William, I}` to prepare for the example below that shows the preferred usage of these macros:[26]

Name Pattern(s):

```
Mark!Twain!MN
SamuelL.!Clemens!PN
Voltaire!MN
FranÃğois-Marie!Arouet!PN
William,I!MN
William,theConqueror!PN
Bernard,ofClairvaux!MN
Lao-tzu!MN
Li,Er!PN
```

Macro/Output	**Index**
`\PName[Mark]{Twain}[Samuel L.]{Clemens}`	
Mark Twain (Samuel L. Clemens)	Clemens, Samuel L. *see* Twain, Mark
Twain (Samuel L. Clemens)	Clemens, Samuel L. *see* Twain, Mark
`\PName*[Mark]{Twain}[Samuel L.]{Clemens}[Sam]`	
Mark Twain (Sam Clemens)	Clemens, Samuel L. *see* Twain, Mark
`\PName{Voltaire}[François-Marie]{Arouet}`	
Voltaire (François-Marie Arouet)	Arouet, François-Marie *see* Voltaire
Voltaire (François-Marie Arouet)	Arouet, François-Marie *see* Voltaire
`\PName{William, I}{William, the Conqueror}`	
William I (William the Conqueror)	William the Conqueror *see* William I
William (William the Conqueror)	William the Conqueror *see* William I
`\PretagName{\textit{Doctor mellifluus}}{Doctor mellifluus}`	
`\PName{Bernard, of Clairvaux}{\textit{Doctor mellifluus}`	
Bernard of Clairvaux (*Doctor mellifluus*)	*Doctor mellifluus see* Bernard of Clairvaux
Bernard (*Doctor mellifluus*)	*Doctor mellifluus see* Bernard of Clairvaux
`\ForgetThis\PName{Lao-tzu}{Li, Er}`	
Lao-tzu (Li Er)	Li Er *see* Lao-tzu
Lao-tzu (Li Er)	Li Er *see* Lao-tzu

Back to Section 1.3

This space is intentionally left blank.

[26] The xref pattern `\textit{Doctormellifluus}!PN` is too large for the margin note. With `pdflatex` and `latex`, in `FranÃğois-Marie!Arouet!PN` the glyphs `Ãğ` correspond to the Unicode encoding macro `\IeC{\c c}`.

2.10 Longer Examples

For the rest of this manual, many examples are in `examples.tex` with the nameauth documentation. Here we set the formatting hooks to the package default.

dtx files

A reminder: When creating package documentation, any name that has a macro in its argument should be set up in the driver section (the `nameauth` environment and tags from `\PretagName` and `\TagName`). Otherwise, errors can result.

2.10.1 Hooks: Intro

Before we get to the use of text tags and name conditionals in name formatting, we seek to illustrate that something more complex than a font switch can occur in `\NamesFormat`. Below we put the first mention of a name in boldface, along with a marginal notation if possible. Unlike the rest of this section, we do not change formatting macros within a scope. Instead, we illustrate a different approach using `\let` to save the old value, then restore it later.

```
\let\OldFormat\NamesFormat
\renewcommand*\NamesFormat[1]{\textbf{#1}\unless\ifinner
   \marginpar{\raggedleft\scriptsize #1}\fi}
\PretagName{Vlad, Ţepeş}{Vlad Tepes} % for accented names
\TagName{Vlad, II}{ Dracul}          % for index information
\TagName{Vlad, III}{ Dracula}
```

Within the document after the preamble:

```
Name{Vlad, III}[III Dracula], known as
\AKA{Vlad III}{Vlad, Ţepeş} (the Impaler)
after his death, was the son of \Name{Vlad, II}[II Dracul],
a member of the Order of the Dragon. Later references to
``\Name*{Vlad, III}'' and ``\Name{Vlad, III}'' appear thus.
```

Vlad III Dracula
Vlad II Dracul

Vlad III Dracula, known as Vlad Ţepeş (the Impaler) after his death, was the son of **Vlad II Dracul**, a member of the Order of the Dragon. Later references to “Vlad III” and “Vlad” appear thus.

```
let\NamesFormat\OldFormat
```

Now we have reverted to the default `\NamesFormat` and we get:

- `\ForgetThis\Name{Vlad, III}[III Dracula]` . . . Vlad III Dracula
- `\Name*{Vlad, III}` . . . . . Vlad III
- `\Name{Vlad, III}` . . . . . Vlad

We also set up the cross-reference `\IndexRef{Dracula}{Vlad III}`. Compare the examples for Demetrius I in Section 2.3.5.

Back to Section 1.3

2.10.2 Hooks: Life Dates

We can use name conditionals (Section 2.8.2) and text tags (Section 2.5) to add life information to names when desired. We begin a local scope to isolate any changes to the formatting hooks.

\if@nameauth@InName
\if@nameauth@InAKA

The example \NamesFormat below adds a text tag to the first occurrences of main-matter names. It uses internal macros of \@nameauth@Name. To prevent errors, the Boolean values \if@nameauth@InName and \if@nameauth@InAKA are true only within the scope of \@nameauth@Name and \AKA respectively.

\@nameauth@toksa
\@nameauth@toksb
\@nameauth@toksc

This package makes three token registers available to facilitate using the name conditional macros as we do below. These registers are necessary for names that contain accents anddiacritics.[27]

Below the first use of a name is in small caps. Text tags are in boldface with naming macros, and roman with \AKA. Just because we set up a cross-reference does not mean that we have to use \AKA. We use \ForceName as needed with \AKA. In the document preamble we set up the following:

```
\newif\ifNoTag
\makeatletter
\renewcommand*\NamesFormat[1]{\begingroup%
  \protected@edef\temp{\endgroup\textsc{#1}%
  \unless\ifNoTag
    \if@nameauth@InName
      {\bfseries\noexpand\NameQueryInfo
      [\unexpanded\expandafter{\the\@nameauth@toksa}]
      {\unexpanded\expandafter{\the\@nameauth@toksb}}
      [\unexpanded\expandafter{\the\@nameauth@toksc}]}\fi
    \if@nameauth@InAKA
      {\normalfont\noexpand\NameQueryInfo
      [\unexpanded\expandafter{\the\@nameauth@toksa}]
      {\unexpanded\expandafter{\the\@nameauth@toksb}}
      [\unexpanded\expandafter{\the\@nameauth@toksc}]}\fi
  \fi}\temp\global\NoTagfalse%
}
\makeatother
\let\FrontNamesFormat\NamesFormat
```

We print tags in the first use hooks unless \NoTag is set true. This method uses the two ϵ-TEX primitives \noexpand and \unexpanded to avoid repetition of \expandafter. Since nameauth depends on etoolbox, we assume ϵ-TEX.

Before we can refer to any text tags, we must create them. Using the approach above, we include a leading space in the text tags. The leading space is needed only when a text tag appears.[28] We also set up a cross-reference, which we will use regardless of whether we also use \AKA. The cross-reference will be created only once and skipped thereafter:

```
\NameAddInfo[George]{Washington}{ (1732--99)}
\NameAddInfo[Mustafa]{Kemal}{ (1881--1938)}
\NameAddInfo{Atatürk}{ (in 1934, a special surname)}
\IndexRef{Atatürk}{Kemal, Mustafa}
```

[27] In \AKA these registers correspond to the **last** three arguments, the xref.

[28] Another way to add that space is to put it in the conditional path of the formatting hook and leave it out of the text tags: . . . `{ }\noexpand\NameQueryInfo`. . .

Now we begin with the first example, which, after all the setup, looks deceptively simple, but highly reusable without extra work:

```
\ForgetThis\Wash held office 1789--97.
No tags: \Wash.\\
First use, dates suppressed: \NoTagtrue\ForgetThis\Wash.\\
Subsequent use with format and dates: \ForceName\Wash.
```

George Washington **(1732–99)** held office 1789–97.
No tags: Washington.
First use, dates suppressed: George Washington.
Subsequent use with format and dates: Washington **(1732–99)**.

Since we already set up a cross-reference with `\IndexRef`, we can use just the the naming macros with "Atatürk" and get the desired formatting without any page references in the index:

```
\Name[Mustafa]{Kemal} was granted the name
\Name{Atatürk}. We mention \Name[Mustafa]{Kemal}
and \Name{Atatürk} again.

First use, no tag:
\NoTagtrue\ForgetThis\Name{Atatürk}.
```

Mustafa Kemal **(1881–1938)** was granted the name Atatürk **(in 1934, a special surname)**. We mention Kemal and Atatürk again.

First use, no tag: Atatürk.

Since we set up distinct formatting for `\AKA` (`\normalfont` instead of boldface for text tags associated with cross-references), we now simulate the `formatAKA` package option and use `\ForceName` with `\AKA`:

```
\makeatletter\@nameauth@AKAFormattrue\makeatother
\ForgetThis\Name[Mustafa]{Kemal} was granted the name
\ForceName\AKA[Mustafa]{Kemal}{Atatürk}. We mention
\Name[Mustafa]{Kemal} and \AKA[Mustafa]{Kemal}{Atatürk} again.

First use, no tag:
\NoTagtrue\ForceName\AKA[Mustafa]{Kemal}{Atatürk}.
```

Mustafa Kemal **(1881–1938)** was granted the name Atatürk (in 1934, a special surname). We mention Kemal and Atatürk again.

First use, no tag: Atatürk.

Now we end the scope to revert any changes to formatting hooks.

Back to Section 1.3

2.10.3 Hooks: Advanced

In this section we invoke `\AltFormatActive` and create several scopes containing respective examples. Some macros in this section normally should be defined in a document preamble. We define them locally and ensure that names do not use them when they are undefined. This is not best practices, but it makes sense for this manual's need for multiple redefinitions.

Alternate Formatting: Details

Here we discuss the implementation details of alternate formatting, which will engage the rest of the section. This framework provides features that aid both error protection and ease of hook design. Names that use alternate formatting may cause spurious index entries if used also in the default formatting regime.

Both `\AltFormatActive` and `\AltFormatActive*` globally set the internal Boolean flag `\@nameauth@AltFormattrue`, enabling alternate formatting. `\AltFormatActive` globally sets `\@nameauth@DoAlttrue`, which activates formatting. Both flags are reset globally by `\AltFormatInactive` and normal formatting resumes.

`\AltFormatActive*` normally suppresses formatting changes but it still forces `\CapThis` to work through `\AltCaps`. One can leverage this to get the default look of nameauth while mitigating errors if many names use macros in their arguments.

Alternate formatting protects against errors created when `\@nameauth@Cap` (used by `\CapThis`) gets a failure result from `\@nameauthUTFtest`, but that result is neither a letter nor a macro that expands to a sequence of letters. Protected macros and other cases may create errors if `\MakeUppercase` is applied to them. `\AltCaps` and `\CapThis` work together to avoid this problem (Section 2.7).

Continental Format: Predefined

Here we look in greater detail at how nameauth implements the advanced version of Continental formatting. Font changes occur in the short macros `\textSC`, `\textIT`, `\textBF`, and `\textUC`. They all look similar to `\textSC`:

```
\newcommand*\textSC[1]{%
  \if@nameauth@DoAlt\textsc{#1}\else#1\fi
}
```

If the `altformat` option or `\AltFormatActive` is used, formatting occurs in both the text and in the index. `\AltOff` deactivates formatting only in the formatting hooks:

```
\newcommand*\AltOff{%
  \if@nameauth@InHook\@nameauth@DoAltfalse\fi
}
```

`\CapThis` triggers `\AltCaps` to capitalize its argument:

```
\newcommand*\AltCaps[1]{%
  \if@nameauth@InHook
    \if@nameauth@DoCaps\MakeUppercase{#1}\else#1\fi
  \else#1\fi
}
```

We must put `\noexpand` before `\textSC`, `\AltCaps`, and so on to prevent them from expanding outside of the formatting hooks.

Before we alter the formatting hooks, either we can `\let` the hook macros to recall them later or we can use `\begingroup` and `\endgroup` to create a new scope that localizes any changes. We use scoping in this section.

Continental Format: User-Defined

The user must implement this final step. We use `\AltFormatActive`, then redefine `\MainNameHook` to have small caps on by default in the index and first uses in the text, then off in subsequent uses in the text:

```
\renewcommand*\MainNameHook{\AltOff}
\let\FrontNameHook\MainNameHook
```

To suppress all formatting in the front-matter text, one need simply to use `\let\FrontNamesFormat\MainNameHook`. We do not do that here. Usually, we set up the names and any related macros in the preamble:

```
\begin{nameauth}
  \< Adams   & John  & \noexpand\textSC{Adams}        & >
  \< SDJR    & Sammy & \noexpand\textSC{Davis},
                       \noexpand\textSC{Jr}.          & >
  \< HAR     &       & Harun, \noexpand\textSC%
                       {\noexpand\AltCaps{a}l-Rashid} & >
  \< Mencius &       & \noexpand\textSC{Mencius}      & >
\end{nameauth}
```

Likewise in the preamble, we must ensure that these names are sorted properly in the index. When sorting names, be sure to use `\noexpand` as well:

```
\PretagName[John]{\noexpand\textSC{Adams}}{Adams, John}
\PretagName[Sammy]%
  {\noexpand\textSC{Davis}, \noexpand\textSC{Jr}.}%
  {Davis, Sammy, Jr.}
\PretagName{Harun, \noexpand\textSC%
  {\noexpand\AltCaps{a}l-Rashid}}{Harun al-Rashid}
\PretagName{\noexpand\textSC{Mencius}}{Mencius}
```

First	Next	Long	Short
`\Adams`	`\Adams`	`\LAdams`	`\SAdams`
John Adams	Adams	John Adams	John
`\SDJR`	`\SDJR`	`\LSDJR`	`\SSDJR`
Sammy Davis Jr.	Davis	Sammy Davis Jr.	Sammy
`\HAR`	`\HAR`	`\LHAR`	`\SHAR`
Harun al-Rashid	Harun	Harun al-Rashid	Harun
`\Mencius`	`\Mencius`	`\LMencius`	`\SMencius`
Mencius	Mencius	Mencius	Mencius

- Punctuation detection works: Sammy Davis Jr. Also Sammy Davis Jr. Then Davis. Now Davis. (We used `\ForceName` for formatting.)
- `\ForceName\DropAffix\LSDJR` gives Sammy Davis. Otherwise, using the macro `\DropAffix\LSDJR` gives Sammy Davis.
- `\RevComma\LAdams` yields Adams, John. All the reversing macros work.
- `\ForceName\ForceFN\SHAR` produces al-Rashid. `\ForceFN\SHAR` produces al-Rashid. If we add `\CapThis` we get Al-Rashid and Al-Rashid.[29]
- One must include all the macros in the name arguments.

[29] The way that Continental resources treat certain affixes relates to similar issues in [Mulvany, 168–73]. Handling non-Western names in Western sources can be a gray area. One ought take care to be culturally sensitive in these matters.

With the `formatAKA` option we refer to Mencius as Meng Ke and Meng Ke:

```
\PretagName{\noexpand\textSC{Meng}, Ke}{Meng Ke}
\AKA{\noexpand\textSC{Mencius}}{\noexpand\textSC{Meng}, Ke}
```

Rolling Your Own: Basic

Here we set out on the path to custom formatting by using package features that have been implemented already and look similar to the solutions in Section 2.7.

When redesigning formatting hooks, we recommend using `\AltFormatActive` or the `altformat` option to enable alternate formatting and prevent `\CapThis` from breaking custom formatting macros.

We recommend examining the internal package flag `\@nameauth@DoAlt`, which activates alternate formatting, `\@nameauth@DoCaps`, which handles capitalization, and `\@nameauth@InHook`, which is true when the formatting hooks are called. See page 111 and following. If you create your own macros, they will look similar.

Normally we define a macro for use in name arguments in the document preamble in order to ensure that it is always defined:

```
\makeatletter
\newcommand*\Fbox[1]{%
  \if@nameauth@DoAlt\protect\fbox{#1}\else#1\fi
}
\makeatother
```

Since `\AltCaps` is part of nameauth, you need not reinvent that wheel. Just use it. The final step is redefining the hooks, which can be as simple as:

```
\renewcommand*\MainNameHook{\AltOff}
\let\FrontNameHook\MainNameHook
```

When sorting names, be sure to use `\noexpand` as shown previously:

```
\begin{nameauth}
  \< deSmet & Pierre-Jean &
     \noexpand\Fbox{\noexpand\AltCaps{d}e~Smet} & >
\end{nameauth}

\PretagName[Pierre-Jean]%
  {\noexpand\Fbox{\noexpand\AltCaps{d}e~Smet}}%
  {de~Smet, Pierre-Jean}
```

Now we show how the formatting hooks work in the body text. One can check the index to see that it is formatted properly and consistently.

First	`\deSmet`	Pierre-Jean de Smet
Next	`\deSmet`	de Smet
Long	`\LdeSmet`	Pierre-Jean de Smet
Short	`\SdeSmet`	Pierre-Jean
	`\CapThis\deSmet`	De Smet
	`\ForceName\CapThis\deSmet`	De Smet

Some formatting, such as the use of `\textSC`, is fairly standard. Other formatting, such as `\Fbox` above, is ornamental and may be handled better with custom features (Section 2.10.4), but those features appear only in the text.

Rolling Your Own: Intermediate

"Intermediate" and "advanced" refer to the way that formatting hooks were designed before version 3.1. We begin the journey to more customized formatting by looking at `\NameParser`, whose logic Sections 3.4 and 3.6 show in detail.

`\NameParser` **3.1**

This user-accessible parser (Section 3.6) builds a printed name from internal, locally-scoped macros `\FNN`, `\SNN`, `\rootb` and `\suffb`. It uses only these Boolean flags:[30]

Only one or the other of these can be true to avoid undocumented behavior.

`\if@nameauth@FullName`	Print a full name if true.
`\if@nameauth@FirstName`	Print a first name if true.

Reversing without commas overrides reversing with commas.

`\if@nameauth@RevThis`	Reverse name order if true.
`\if@nameauth@EastFN`	toggled by `\ForceFN`.
`\if@nameauth@RevThisComma`	Reverse Western name, add comma.

We create a hook that can ignore the output of `\@nameauth@Name`, which is the `#1` in the hook dispatcher's code `\bgroup`⟨*Hook*⟩`{#1}\egroup`:

```
\renewcommand*⟨FirstHook⟩[1]{...\NameParser...}
```

With the `altformat` option or `\AltFormatActive` we can design a subsequent-use hook that deactivates formatting inside of it:

```
\renewcommand*⟨SubsequentHook⟩[1]{...\AltOff\NameParser...}
```

If we used `\AltFormatActive*`, where the formatting macros are enabled, but deactivated, then we might want a hook that activates the macros:

```
\renewcommand*⟨Hook⟩[1]{...\AltOn\NameParser...}
```

Within the hooks we can use the user-side parser as often as we want. We also can change internal Boolean flags, for example:

```
\makeatletter
\renewcommand*\NamesFormat[1]{\small%
  \hbox to 3.5em{[now]\hfill}\space\NameParser\\%
  \@nameauth@FullNametrue%
  \hbox to 3.5em{[long]\hfill}\space\NameParser\\%
  \@nameauth@FullNamefalse%
  \@nameauth@FirstNametrue%
  \hbox to 3.5em{[short]\hfill}\space\NameParser}
\makeatother
\let\MainNameHook\NamesFormat
```

One instance of `\JRIV` displays:

[now] Rockefeller
[long] J.D. Rockefeller IV
[short] J.D.

The proof of concept above is interesting, but not very useful in a practical setting. Now we move on toward more useful and practical designs.

[30] The capitalization macros interact with the internal macros before the name parser, therefore, they do not directly engage the output of the name parser.

We begin in the document preamble by defining a series of conditionals and macros whose default expansion produces the index entry, yet whose other expansions occur only in the formatting hooks. Then we create a name that is composed only of macros, using `\noexpand` with `\WM` and `\SHK`. We use `\PretagName` to sort the names. `\Revert` is used to print a last name without a margin note.

```
\newif\ifSpecialFN
\newif\ifSpecialSN
\newif\ifRevertSN
\newcommand*\WM{\ifSpecialFN Wm.\else William\fi}
\newcommand*\SHK{\ifRevertSN \textSC{Shakespeare}\else
                 \ifSpecialSN \noexpand\AltCaps{t}he Bard\else
                 \textSC{Shakespeare}\fi\fi}
\newcommand*\Revert{\RevertSNtrue}

\begin{nameauth}
  \< Shak & \noexpand\WM & \noexpand\SHK & >
\end{nameauth}

\PretagName[\noexpand\WM]{\noexpand\SHK}{Shakespeare, William}
\PretagName[Robert]{\textSC{Burns}}{Burns, Robert}
```

Below we define the two formatting hooks that structure the ways in which these macros can expand when printed in the text. `\NamesFormat` allows only the canonical name via `\RevertSNfalse`, `\SpecialFNfalse`, and `\SpecialSNfalse`. We print the canonical name in the body text. If allowed, we print a margin paragraph with an alternate full name using `\NameParser` and two flags. Both hooks set `\RevertSNfalse` so that `\Revert` works on a per-name basis. The subsequent-use hook disables formatting with `\AltOff`, but it allows variant forms.

```
\makeatletter
\renewcommand*\NamesFormat[1]{%
  \RevertSNfalse\SpecialFNfalse\SpecialSNfalse#1%
  \unless\ifinner\marginpar{%
    \footnotesize\raggedleft%
    \@nameauth@FullNametrue%
    \@nameauth@FirstNamefalse%
    \@nameauth@EastFNfalse%
    \SpecialFNtrue\SpecialSNfalse%
    \NameParser}%
  \fi\global\RevertSNfalse}
\renewcommand*\MainNameHook[1]{%
  \AltOff\SpecialFNfalse\SpecialSNtrue#1%
  \unless\ifinner
    \unless\ifRevertSN
      \marginpar{%
      \footnotesize\raggedleft%
      \@nameauth@FullNamefalse%
      \@nameauth@FirstNamefalse%
      \@nameauth@EastFNfalse%
      \SpecialFNfalse\SpecialSNfalse%
      \NameParser}%
    \fi
  \fi\global\RevertSNfalse}
\makeatother
```

Wm. Shakespeare
Robert Burns

Shakespeare

William Shakespeare \ForgetThis\Shak is the national poet of England in much the same way that Robert Burns \Name[Robert]{\textSC{Burns}} is that of Scotland. With the latter's rise of influence in the 1800s, Shakespeare \Revert\Shak became known as "the Bard" \Shak.

Rolling Your Own: Advanced

Here is how formatting hooks were designed before version 3.0. Updating older hooks may be helpful, but is not necessary. We do not use the internal package macros. We only use \NameParser in the hooks to produce output. We still recommend using \AltFormatActive to mitigate errors. In the preamble, three flags replace package internals.[31] Setting \Fboxtrue is equivalent to using \AltFormatActive:

```
\newif\ifFbox%     Replaces \@nameauth@DoAlt
\newif\ifFirstCap% Replaces \@nameauth@DoCaps
\newif\ifInHook%   Replaces \@nameauth@InHook
\Fboxtrue
```

Also in the preamble, the formatting macro is like what we have seen, except it refers to \ifFbox:[32]

```
\newcommand*\Fbox[1]{%
  \ifFbox\protect\fbox{#1}\else#1\fi
}
```

Our new \AltCaps works like the built-in version, except it does not use the internal macros and flags:

```
\renewcommand*\AltCaps[1]{%
  \ifInHook
    \ifFirstCap\MakeUppercase{#1}\else#1\fi
  \else
    #1%
  \fi
}
```

Here we redefine \CapThis to use our flag instead of the internal flag:

```
\renewcommand*\CapThis{\FirstCaptrue}
```

We have to reproduce the logic and macros that the package would have provided. That means defining everything, including \NamesFormat, from scratch:

```
\renewcommand*\NamesFormat[1]
  {\InHooktrue\NameParser\global\FirstCapfalse}
```

Changes to \ifInHook (default false) and \ifFbox (default true) are local to the scope in which the hook macros are called. \ifFirstCap must be set globally. Below we reproduce the logic of \AltOff before \NameParser:

```
\renewcommand*\MainNameHook[1]
{\Fboxfalse\InHooktrue\NameParser\global\FirstCapfalse}
```

[31] The internal flag \@nameauth@DoAlt activates formatting, \CapThis sets \@nameauth@DoCaps true, and \@nameauth@InHook is set by the hook dispatcher.

[32] As previously noted, we define \Fbox locally in this manual because it has multiple definitions, but are very careful where we use names with it.

We avoid spurious index entries in the front matter by using the same hooks.

```
\let\FrontNamesFormat\Namesformat
\let\FrontNameHook\MainNameHook
```

Because we use `\noexpand`, our "old-style" macros will index the following names under the same entry as the "new-style" macros.

First	`\deSmet`	Pierre-Jean de Smet
Next	`\deSmet`	de Smet
Long	`\LdeSmet`	Pierre-Jean de Smet
Short	`\SdeSmet`	Pierre-Jean
	`\CapThis\deSmet`	De Smet
	`\ForceName\CapThis\deSmet`	De Smet

We can reuse new-style names with old-style macros, shown below in abbreviated fashion. We keep the flags `\ifFirstCap` and `\ifInHook`. We also keep the redefined `\AltCaps`, `\CapThis`, and `\NamesFormat`. We then add:

```
\newif\ifCaps
\Capstrue
\renewcommand*\textSC[1]{%
  \ifCaps\textsc{#1}\else#1\fi
}
\renewcommand*\MainNameHook[1]
{%
  \Capsfalse\InHooktrue\NameParser%
  \global\FirstCapfalse%
}
\let\FrontNameHook\MainNameHook
```

The names below have the same declarations and index entries as they did above. They look and work the same but use different back-end macros:

First	Next	Long	Short
John Adams	Adams	John Adams	John
Sammy Davis Jr.	Davis	Sammy Davis Jr.	Sammy
Harun al-Rashid	Harun	Harun al-Rashid	Harun
Mencius	Mencius	Mencius	Mencius

- Punctuation detection works: Sammy Davis Jr. Also Sammy Davis Jr. Then Davis. Now Davis. (We used `\ForceName` for formatting.)
- `\ForceName\DropAffix\LSDJR` gives Sammy Davis. Otherwise, using the macro `\DropAffix\LSDJR` gives Sammy Davis.
- `\RevComma\LAdams` yields Adams, John. All the reversing macros work.
- `\ForceName\ForceFN\SHAR` produces al-Rashid. `\ForceFN\SHAR` produces al-Rashid. If we add `\CapThis` we get Al-Rashid and Al-Rashid.

We now close the scope of this current example and resume normal formatting.

Back to Section 1.3

2.10.4 Customization

Assuming that redefining hooks and adding control sequences is insufficient, one could redesign the core name macros partially or wholly, then hook those modifications into the nameauth package without needing to patch the style file itself.

\NameauthName
\NameauthLName
\NameauthFName

All these macros are set by default to \@nameauth@Name, the internal name parser. \Name, or an unmodified shorthand, calls \NameauthName. \Name*, or an L-shorthand, sets \@nameauth@FullNametrue, then calls \NameauthLName. \FName, or an S-shorthand, sets \@nameauth@FirstNametrue, then calls \NameauthFName. One should not modify \Name and \FName directly.

Since nameauth depends on xargs, we use that in a minimal working example that implements the obsolete syntax (Section 2.11.4). We use few internal Boolean values, save those governing name forms. We do not implement short forms or any other features in nameauth. We must index the names with \IndexName. This example shows how to hook these redefined macros into the user interface. Note that the quote environment creates a local scope that we leverage below:

```
\makeatletter
\newcommandx*\MyName[3][1=\@empty, 3=\@empty]{%
  \protected@edef\a{\trim@spaces{#1}}%
  \protected@edef\b{\trim@spaces{#2}}%
  \protected@edef\c{\trim@spaces{#3}}%
  \ifx\b\empty fail \else
    \ifx\a\empty
      \ifx\c\empty \hbox to 5em{Mononym:\hfill} {\b}\else
      \hbox to 5em{Eastern:\hfill} {\b\ \c}\fi
    \else
      \ifx\c\empty \hbox to 5em{Western:\hfill} {\a\ \b}\else
      \hbox to 5em{Alternate:\hfill} {\c\ \b}\fi
    \fi
  \fi
  \global\@nameauth@FullNamefalse%
  \global\@nameauth@FirstNamefalse%
}
\makeatother
\let\MyLName\MyName
\let\MyFName\MyName
\renewcommand*\NameauthName{\MyName}
\renewcommand*\NameauthLName{\MyLName}
\renewcommand*\NameauthFName{\MyFName}
\IndexName[George]{Washington}
\IndexName[M.T.]{Cicero}
\IndexName{Dagobert}[I]
\IndexName{Aristotle}
```

\Wash	Western:	George Washington
\Cicero[Marcus Tullius]	Alternate:	Marcus Tullius Cicero
\Dagb	Eastern:	Dagobert I
\Aris	Mononym:	Aristotle

The previous example is not particularly useful. There is, however, a more practical use for these macros. One could choose to implement additional features, then pass the information in the name argument token registers to the extant parsing macros of nameauth (cf. Section 2.10.2).

We continue to use features of xargs, as well as the local scope of a quote environment. Below we introduce formatting that is additional to, inter-operative with, yet distinct from the formatting hooks:

```
\makeatletter
\newcommandx*\MyName[3][1=\@empty, 3=\@empty]{%
  \global\@nameauth@toksa\expandafter{#1}%
  \global\@nameauth@toksb\expandafter{#2}%
  \global\@nameauth@toksc\expandafter{#3}%
  \hbox to 4em{Normal: \hfill}%
  \fcolorbox{black}{gray!25!white}{\@nameauth@Name[#1]{#2}[#3]}%
}
\newcommandx*\MyLName[3][1=\@empty, 3=\@empty]{%
  \global\@nameauth@toksa\expandafter{#1}%
  \global\@nameauth@toksb\expandafter{#2}%
  \global\@nameauth@toksc\expandafter{#3}%
  \hbox to 4em{Long: \hfill}%
  \fcolorbox{black}{green!25!white}{\@nameauth@Name[#1]{#2}[#3]}%
}
\newcommandx*\MyFName[3][1=\@empty, 3=\@empty]{%
  \global\@nameauth@toksa\expandafter{#1}%
  \global\@nameauth@toksb\expandafter{#2}%
  \global\@nameauth@toksc\expandafter{#3}%
  \hbox to 4em{Short: \hfill}%
  \fcolorbox{black}{yellow!25!white}{\@nameauth@Name[#1]{#2}[#3]}%
}
\makeatother
\renewcommand*\NamesFormat[1]
  {\hbox to 9em{\hfil\scshape#1\hfil}}
\renewcommand*\MainNameHook[1]{\hbox to 9em{\hfil#1\hfil}}
\renewcommand*\NameauthName{\MyName}
\renewcommand*\NameauthLName{\MyLName}
\renewcommand*\NameauthFName{\MyFName}
```

```
\ForgetName[Adolf]{Harnack}
```

\Harnack	Normal:	Adolf Harnack
\LHarnack[Adolf von]	Long:	Adolf von Harnack
\Harnack	Normal:	Harnack
\SHarnack	Short:	Adolf

3.3 After the name is printed in the body text, the internal macros **globally** set \@nameauth@FullNamefalse and \@nameauth@FirstNamefalse, as well as other flags related to the prefix macros. This prevents certain cases of undocumented behavior in versions of nameauth before 3.3, where resetting flags locally could cause unexpected name forms. If an existing document leverages the local resetting of flags, one can use the oldreset option. Compare Section 2.4.1.

\global Like many of the macros in this package, these macros can be redefined or used locally within a scope without making global changes to the document unless you specifically use \global.

Back to Section 1.3

2.11 Technical Notes

2.11.1 General

About the package itself:

3.5

- Current features allow nameauth to meet its goals: stability, professional features, and backward-compatibility.
  - Internal macros not in a local scope start with `\@nameauth@`. No more assumptions of "throwaway" macro names.
  - Index control has become stricter and more sensitive to the order of both name and xref creation.
  - It is now easy to have a separate index of persons when using packages and classes that enable that.
  - Internals of all macros that handle name arguments use a standard, optimized logic.
- We keep xargs for backward compatibility. Future package versions will use xparse instead of xargs and shed all compatibility options. A "maintenance" version will be preserved for backward compatibility.
- The package works with both `texindy` and `makeindex`.

About the manual (which is the test suite):

- It has been reworked and expanded. Many months of testing has yielded better explanations that reflect best practices.
- We now emphasize current workflows and de-emphasize older, less-relevant macros and the obsolete syntax.
- It is compatible with both A4 and US letter formats.
- We mention when this manual changes package internals, does actions that are not visually discernible, or deviates from "normal" usage.

About package building:

- Consult `README.md` for building instructions.
- The nameauth package requires etoolbox, suffix, trimspaces, and xargs.
- The package and manual build on current and older LaTeX distributions.
- The `dvi` test modes (`latex` and `dvilualatex`) use `dvipdf` to make *TikZ* and tcolorbox render properly. The `pdf` test modes use `pdflatex`, `lualatex`, and `xelatex`. All modes use `makeindex`.
- This release was tested "officially" on Linux (Manjaro; vanilla TL 2020 and 2017) and Windows 10 (MikTeX). The CTAN release is created with the latest vanilla TL release on Manjaro.

This manual was created with `pdflatex`.

Back to Section 1.3

2.11.2 Package Warnings

Standard and Verbose Warnings

Standard warnings

Package warnings result if one redefines name shorthands in the `nameauth` environment. That could be a problem. Yet if one uses, for example, a new `nameauth` environment per chapter, such warnings might be harmless. For example:

```
1 \PretagName[E.\,B.]{White}{White, Elwyn}
2 \begin{nameauth}
3   \< White & E.B.    & White & > % v.1
4   \< White & E.\,B. & White & > % v.2
5 \end{nameauth}
```

`\White` gives "White". We lost the first version when we redefined it. We "forget" White for later (Section 2.8.1) with `\ForgetName[E.\,B.]{White}`.[33]

3.5 Additionally, the following situations cause package warnings, especially since the indexing macros have been made stricter:

- **Ignore & reset:** `\IndexName` and `\IndexRef` warn if `\SkipIndex` is active, and they reset its flag unless the `oldreset` option is used.
- **Ignore & reset:** `\IndexName` warns if it or a naming macro that contains it was preceded by `\SeeAlso`, whose Boolean flag is then reset.
- **Ignore:** `\IndexRef` warns if one tries to create a *see* reference from an extant name and ignores the attempt unless the `oldsee` option is used.
- **Ignore:** `\PretagName` warns if the `nopretag` option is used and it produces no sort tags in that case.
- **Warn:** `\PName` and `\PName*` warn if `\@nameauth@SkipIndextrue` on exit (only if the `oldreset` option is used).

Verbose warnings

Package warnings result from the following **only** when using the `verbose` option. The macros either allow or ignore certain actions:

- **Allow:** `\ExcludeName` with an extant name.
- **Allow:** `\PretagName` to sort cross-references.

3.5

- **Allow:** `\IndexRef` with the `oldsee` option.
- **Ignore:** make an index page reference from an xref or excluded name.
- **Ignore:** make the same cross-reference multiple times.
- **Ignore:** use `\ExcludeName` with a cross-reference.

3.3

- **Ignore:** use `\IncludeName` with an xref (but `\IncludeName*` works.
- **Ignore:** use `\TagName` and `\UntagName` with a cross-reference.

Back to Section 1.3

[33] There should be two package warnings for redefining `\White`. We defined it in the `dtx` driver, then redefined it twice above.

2.11.3 Debugging and Avoiding Errors

Debugging Macros

\ShowPattern **3.3**

We use \ShowPattern in Section 2.11.5 to illustrate name control patterns. It displays how the name arguments create name patterns that form name control sequences. One can debug pattern collisions and other issues with this macro:

\ShowPattern[⟨*FNN*⟩]{⟨*SNN, Affix*⟩}[⟨*Alternate*⟩]

Thus, \texttt{\ShowPattern[Hernando]{de~Soto}} will produce the output Hernando!de~Soto. As we have seen above, using inputenc/fontenc will cause names like \texttt{\ShowPattern{Boëthius}} to produce BoÃńthius.

\ShowIdxPageref
\ShowIdxPageref* **3.3**

\ShowIdxPageref displays a full index entry in the text. Its counterpart is \ShowIdxPageref*, which shows a short index entry. Both only show names formatted as page references, even if they are cross-references:

\ShowIdxPageref [⟨*FNN*⟩]{⟨*SNN, Affix*⟩}[⟨*Alternate*⟩]
\ShowIdxPageref*[⟨*FNN*⟩]{⟨*SNN, Affix*⟩}[⟨*Alternate*⟩]

Index styles, \PretagName, and \TagName affect the output of \ShowIdxPageref. Active characters and macros appear as printed, not as in idx files. In a normal LaTeX document without hyperlinks, for example, we would get:

```
1 \PretagName[Hernando]{de~Soto}{Desoto, Hernando}
2 \texttt{\ShowIdxPageref[Hernando]{de~Soto}}\\
  Desoto, Hernando@de Soto, Hernando
```

Throughout this manual, \ShowIdxPageref* illustrates basic index entries that do not contain sorting information or tags. The simple index entry of Hernando de Soto is ‘‘\ShowIdxPageref*[Hernando]{de~Soto}’’ “de Soto, Hernando”.

Avoiding Common Errors: General

- A missing square bracket or curly brace can cause errors like “Paragraph ended” and “Missing ⟨*grouping token*⟩ inserted.”
- In the nameauth environment, \< ⟨*arg1*⟩ & ⟨*arg2*⟩ & ⟨*arg3*⟩ & ⟨*arg4*⟩ > is a macro that cannot miss a delimiter, an argument, or an ampersand.
- Generally use \noexpand before macros in name arguments, which is required if the macros contain conditional statements.
- Ensure that macros and conditionals used in name arguments are defined in the preamble or outermost scope (see below).
- Do not format ⟨*SNN*⟩,⟨*Affix*⟩ together as a pair. Format ⟨*SNN*⟩ and ⟨*Affix*⟩ separately (Section 2.7). The comma will segment the input into two arguments, which could cause errors.
- Using \CapThis on a name with macros in its arguments while not using alternate formatting (Section 2.7) could trigger an error. Normally \CapThis also segments the input, which could cause errors.

- Something like `\edef\foo{\CapThis\Name{bar}}` will fail. Yet one can use `\CapThis\Name{bar}` as an argument to a macro defined with either `\edef` or `\xdef`.
- In `dtx` files, put the `nameauth` environment and tags in the `<driver>` section preamble that inputs the `dtx` file as documentation.
- `\Name [`⟨*FNN*⟩`]{`⟨*SNN*⟩`}[\ignorespaces]` prints only ⟨*SNN*⟩.
  `\Name*[`⟨*FNN*⟩`]{`⟨*SNN, Affix*⟩`}[\ignorespaces]` prints ⟨*SNN*⟩ ⟨*Affix*⟩.
  `\FName[`⟨*FNN*⟩`]{`⟨*SNN*⟩`}[\ignorespaces]` either prints ⟨*SNN*⟩ or acts like the macro `\leavevmode`.
  `\Name {`⟨*SNN, Affix*⟩`}[\ignorespaces]` prints only ⟨*SNN*⟩␣.
  `\FName{`⟨*SNN, Affix*⟩`}[\ignorespaces]` either prints ⟨*SNN*⟩␣ or acts like the macros `\leavevmode\space`.

Avoiding Common Errors: Indexing

- Be sure to define all *see* references with `\IndexRef` **before** making any `\Name` references to them. Otherwise `\IndexRef` will generate a warning that it could not create a *see* reference.
- Be sure to define all *see also* references with `\SeeAlso\IndexRef` **after** making all needed `\Name` references to the respective names. Otherwise `\IndexName` and the naming macros will not create page references to the respective `\Name` thereafter.
- Two names may look identical on the page, but their internal name patterns can differ (Sections 2.4.2 and 2.11.5). This will create spurious index entries. Check the `idx` file and possibly use (below).
- To fix spurious entries, compare index entries with names in the text.
  - Check if naming macros always use the same arguments.
  - Check sorting tags (`\PretagName` (Section 2.4.2).
  - Check use of active Unicode characters (Section 2.11.6).
  - Use `\ShowPattern` and `\ShowIdxPageref` (below).
  - Check if macros in name arguments did not follow `\noexpand`.
- Check `nameauth` package warnings. Set the `verbose` option, which will offer a number of "informational" warnings that could be of assistance with the index.

Macros in Name Arguments

- Use alternate formatting to avoid potential problems, especially when using `\CapThis` (Sections 2.7, 2.10.3).
- Use `\noexpand`⟨*macro*⟩ in name macro arguments as a best practice. This is required for all such macros that contain conditional statements.
- Macros used in name arguments must be defined either in the preamble or in the outermost document environment scope to avoid `Undefined control sequence` errors.
- Boolean flags (`\if`⟨*flag*⟩) used in formatting hooks must be defined either in the preamble or in the outermost document scope.

- The `\global` modifier does not work with `\newif` and `\newcommand`.[34] Yet `\global` can precede a macro defined with `\newcommand`, and the first `\def` used therein may be global.

The TEXbook, pages 275–277, shows what `\global` can and cannot do. In the following example, we declare a Boolean flag and a macro in the outer scope, then make several declarations and assignments in the inner scope. After the inner scope ends, we test to see what has happened:

```
\newif\ifCondA
\newcommand\MacroA{}
\begingroup
  \newif\ifCondB
  \global\newif\ifCondC
  \global\newcommand\MacroB{}
  \newcommand\MacroC{\def\MacroD{}}
  \global\MacroC
  \global\CondAtrue
\endgroup
```

- `\ifCondA` is defined in the outer scope (outer definition).
- `\MacroA` is defined in the outer scope (outer definition).
- `\ifCondB` is not defined in the outer scope (local definition).
- `\ifCondC` is not defined in the outer scope (no `\global\newif`).
- `\MacroB` is not defined in the outer scope (no `\global\newcommand`).
- `\MacroC` is not defined in the outer scope (local definition).
- `\MacroD` is defined in the outer scope (`\global` affects `\def` in `\MacroC`).
- `\ifCondA` is true (`\global` assignment works, not instantiation).

Any macro that is used in the argument of a naming macro must be defined in all scopes in which that name is used. Below we deactivate indexing and show this:

```
\begin{nameauth}
  \< Testi & & \noexpand\TESTi & >
  \< Testii & & \noexpand\TESTii & >
\end{nameauth}
\def\TESTi{Test One}
\indent \hbox to 10em{(Outer 1) \Testi\hfill}
\bgroup
  (Inner 1) \Testi\\
  \def\TESTii{Test Two}
  \hbox to 10em{(Inner 2) \Testii\hfill}
\egroup
(Outer 2) \unless\ifdefined\TESTii \cmd{\TESTii} undefined\fi
```

(Outer 1) Test One (Inner 1) Test One
(Inner 2) Test Two (Outer 2) \TESTii undefined

Back to Section 1.3

[34] See this page on redefining `\newcommand`, with the caveats that apply.

2.11.4 Obsolete Syntax

This non-Western syntax limits alternate names and xrefs, excludes comma-delimited names, and complicates indexing. It is a ghost of nameauth past.

```
\Name{⟨SNN⟩}[⟨Alternate⟩]   % obsolete syntax
```

- One must **leave empty** the first optional ⟨*FNN*⟩ argument.
- One must **never** use the comma-delimited argument ⟨*SNN, Affix*⟩.
- Instead, these names **always** use the final optional ⟨*Alternate*⟩ argument, which acts like ⟨*Affix*⟩ and affects both name and index patterns (Section 2.11.5).
- These names take the form ⟨*SNN Alternate*⟩ in the index.

In this manual we designate these names with a double dagger (‡):

```
\Name{Henry}[VIII]                  % Ancient
\Name{Chiang}[Kai-shek]             % Eastern
\begin{nameauth}
  \< Dagb & & Dagobert & I       > % Ancient
  \< Yosh & & Yoshida  & Shigeru > % Eastern
  \< OFukuyama & &
     \textUC{Fukuyama} & Takeshi > % Alt. format
\end{nameauth}
```

Name Pattern(s):

```
Henry,VIII!MN
Chiang,Kai-shek!MN
Dagobert,I!MN
Yoshida,Shigeru!MN
```

`\ForgetThis\Name{Henry}[VIII]`	Henry VIII‡
`\Name{Henry}[VIII]`	Henry‡
`\ForgetThis\Name{Chiang}[Kai-shek]`	Chiang Kai-shek‡
`\Name{Chiang}[Kai-shek]`	Chiang‡
`\Dagb`	Dagobert I‡
`\Dagb`	Dagobert‡
`\CapName\Yosh`	YOSHIDA Shigeru‡
`\CapName\RevName\LYosh`	Shigeru YOSHIDA‡
`\AltFormatActive`	
`\ForgetThis\OFukuyama`	FUKUYAMA Takeshi‡
`\OFukuyama`	FUKUYAMA‡
`\AltFormatInactive`	

2.6 Regardless of its flaws, the obsolete syntax shares name patterns, index tags, text tags, and index entries with the current syntax:

Obsolete syntax:	`\ForgetThis\Name{Henry}[VIII]`	Henry VIII‡
Current syntax:	`\Name{Henry, VIII}`	Henry

Back to Section 1.3

2.11.5 Name Pattern Overview

The table below shows how the macro arguments generate name patterns central to nameauth. The ⟨*Alternate*⟩ argument only affects patterns when using the obsolete syntax. The naming macro arguments create internal control sequences that affect names in both the text and the index:

Macro Arguments	Patterns	Type
`[`⟨*FNN*⟩`]{`⟨*SNN*⟩`}`	⟨*FNN*⟩!⟨*SNN*⟩	Western
`[`⟨*FNN*⟩`]{`⟨*SNN, Affix*⟩`}`	⟨*FNN*⟩!⟨*SNN*⟩,⟨*Affix*⟩	Western
`{`⟨*SNN, Affix*⟩`}`	⟨*SNN*⟩,⟨*Affix*⟩	non-Western
`{`⟨*SNN*⟩`}[`⟨*Alt*⟩`]`	⟨*SNN*⟩,⟨*Alt*⟩	obsolete
`{`⟨*SNN*⟩`}`	⟨*SNN*⟩	non-Western

The internal parser `\@nameauth@Parse` determines the type of name through the presence or absence of certain arguments. Then it assigns name control sequences keyed to the type of name pattern and the current naming system. Other macros do similar tasks with name control sequences associated with other data sets.

First we show name patterns generated from name elements and the type of name. "Non-native" Eastern names are marked by a dagger (†); names that use the obsolete syntax are marked by a double dagger (‡).

Macro	Body Text	`\ShowPattern`
`\ForgetThis\Harnack[Adolf von]`	Adolf von Harnack	`Adolf!Harnack`
`\LHarnack`	Adolf Harnack	`Adolf!Harnack`
`\ForgetThis\Pat`	George S. Patton Jr.	`GeorgeS.!Patton,Jr.`
`\DropAffix\LPat`	George S. Patton	`GeorgeS.!Patton,Jr.`
`\ForgetThis\Noguchi`	Hideyo Noguchi	`Hideyo!Noguchi`
`\RevName\LNoguchi`	Noguchi Hideyo†	`Hideyo!Noguchi`
`\ForgetThis\Yamt`	Yamamoto Isoroku	`Yamamoto,Isoroku`
`\RevName\LYamt`	Isoroku Yamamoto	`Yamamoto,Isoroku`
`\ForgetThis\Name{Henry,VIII}`	Henry VIII	`Henry,VIII`
`\Name*{Henry}[VIII]`	Henry VIII‡	`Henry,VIII`
`\ForgetThis\Dem[I Soter]`	Demetrius I Soter	`Demetrius,I`
`\LDem`	Demetrius I	`Demetrius,I`
`\ForgetThis\Aris`	Aristotle	`Aristotle`
`\Aris`	Aristotle	`Aristotle`

Six suffixes are appended to these patterns to create independent data sets:

Description	Pattern	Mnemonic	Example
Front-matter names	⟨*pattern*⟩`!NF`	"name front"	`Adolf!Harnack!NF`
Main-matter names	⟨*pattern*⟩`!MN`	"main name"	`Hideyo!Noguchi!MN`
Index cross-refs	⟨*pattern*⟩`!PN`	"pseudonym"	`Yamamoto,Isoroku!PN`
Index sorting tags	⟨*pattern*⟩`!PRE`	"pretag"	`Henry,VIII!PRE`
Index info tags	⟨*pattern*⟩`!TAG`	"tag"	`Demetrius,I!TAG`
"Text tag" database	⟨*pattern*⟩`!DB`	"database"	`Aristotle!DB`

The following macros **write** to these data sets; others also can read from them:

Macros	!NF	!MN	!PN	!PRE	!TAG	!DB
\Name \Name* \FName	■	■				
\ForgetName \SubvertName	■	■				
\PName\PName*	■	■	■			
\AKA \AKA* \IndexRef			■			
\ExcludeName			■			
\IncludeName \IncludeName*			■			
\PretagName				■		
\TagName \UntagName					■	
\NameAddInfo \NameClearInfo						■

Back to Section 1.3

2.11.6 Active Unicode Characters

General Information

Below we group characters by accents and diacritical marks:

acute	Á Ć É Ǵ · Í Ĺ	á ć é ǵ · í Í
	Ń Ó Ŕ Ś Ú Ý Ź	ń ó ŕ ś ú ý ź
grave	À · È · · Ì Ò Ù	à · è · · ì ò ù
circumflex	Â Ĉ Ê Ĝ Ĥ Î Ĵ Ô Ŝ Û Ŵ Ŷ	â ĉ ê ĝ ĥ î ĵ ô ŝ û ŵ ŷ
tilde	Ã · · · · Ĩ Ñ Õ Ũ	ã · · · · ĩ ñ õ ũ
diaresis[35]	Ä · Ë · · Ï Ö Ü Ÿ	ä · ë · · ï ö ü ÿ
cedilla	· Ç · Ģ Ķ Ļ Ņ Ŗ Ş Ţ	· ç · ģ ķ ļ ņ ŗ ş ţ
macron	Ā · Ē Ḡ · Ī Ō Ū Æ Ȳ	ā · ē ḡ · ī ō ū æ ȳ
breve	Ă · · Ğ · Ĭ Ŏ Ŭ	ă · · ğ · ĭ ŏ ŭ
dot/dotless	Ḃ Ċ Ė Ġ · İ Ż	ḃ ċ ė ġ · ı ż
ogonek	Ą · Ę · · Į Ǫ Ų	ą · ę · · į ǫ ų
caron	Ǎ Č Ď Ě Ǧ Ǐ Ǩ Ľ	ǎ č ď ě ǧ ǐ ǰ ǩ ľ
	Ň Ǒ Ř Š Ť Ǔ Ž	ň ǒ ř š ť ǔ ž
various	Å Æ Ð (eth) Đ (stroke) IJ Ł	å æ ð đ ij ł
	Ŋ Ø Œ Ő Ů Ű Ș Ț Þ	ŋ ø œ ő ů ű ș ß ț þ

[35] A diaresis mark is one way to indicate an umlaut, a sound change. German originally used a superscript e over a, o, and u. The cursive form of e simplified to a diaresis mark in the 1800s. A diaresis mark also signals a diaresis: reading a diphthong as two monophthongs.

With `\usepackage[T1]{fontenc}`, `latex` and `pdflatex` can use many active Unicode characters automatically. Use `\PretagName` to sort names with these characters (Section 2.4.2). Currently, most documents typeset with `latex` and `pdflatex` do not require explicit loading of either inputenc or inputenx.

Additional Unicode characters can be made available when using fonts with TS1 glyphs (pages 455–463 in *The Latex Companion*). Compare the list on this page or type `texdoc comprehensive` in a terminal window.

Active Unicode characters work much like macros. When using a font with TS1 glyphs and slots, the following preamble snippet is an example of how one might add more Unicode characters, such as a long s (*s-medialis*):

```
\usepackage[utf8]{inputenc} % For older TL releases
\usepackage[TS1,T1]{fontenc}
\usepackage{lmodern}% Contains TS1 glyph 115
\usepackage{newunicodechar}
\DeclareTextSymbolDefault{\textlongs}{TS1}
\DeclareTextSymbol{\textlongs}{TS1}{115}
\newunicodechar{ſ}{\textlongs}

In Congreſs, July 4, 1776
```

In Congreſs, July 4, 1776

Many Unicode characters have native support in `xelatex` and `lualatex`, but not in `pdflatex`. Yet the latter has certain features (e.g., with respect to microtype) that others lack. The features of `makeindex` do not always equate to those in `xindy`. Those differences impact design choices.

Compatibility: Old and New

As mentioned in Section 2.4.2, before 2018, some index styles could not work with characters that contained macrons:

Ā Ē Ḡ Ī Ō Ū Ǣ Ȳ ā ē ḡ ī ō ū ǣ ȳ

Since 2018, those restrictions have been removed due to better handling of Unicode characters in `latex` and `pdflatex`.

If compiled on a recent version of LaTeX, one will see a macron in the name below. To allow the manual to compile on older versions, the following code prints a version without the macron as needed:

```
\ifPDFTeX
  \IfFileExists{utf8-2018.def}%
    {\Name{Ghazāli}}{\Name{Ghazali}}%
  \else\Name{Ghazāli}%
\fi
```

Ghazāli

Even now, although one can use the Unicode characters with macrons, control sequences like `\=a` in the index will cause undocumented behavior when using `makeindex` and `gind.ist`. The latter index style, used for `dtx` files, changes the "actual" character from `@` to `=`.

Fragility of Active Unicode

TEX macros that partition their arguments can break active Unicode characters. Consider the simple macro `\def\foo#1#2#3!{<#1#2><#3>}`. It takes three undelimited arguments and groups the first two, then the third:

Argument	Macro	Engine	Result
`abc`	`\foo abc!`	(any)	<ab><c>
`{æ}bc`	`\foo {æ}bc!`	(any)	<æb><c>
`\ae bc`	`\foo \ae bc!`	(any)	<æb><c>
`æbc`	`\foo æbc!`	`xelatex`	<æb><c>
`æbc`	`\foo æbc!`	`lualatex`	<æb><c>
`æbc`	`\foo æbc!`	`pdflatex`	<æ><bc>
`æbc`	`\foo æbc!`	`latex`	<æ><bc>

The letter `a` is one argument. Since `{æ}` is in a group, it is one argument. The macro `\ae` also is one argument. Thus, the first two glyphs are grouped together in `#1#2` and `c` is left by itself in `#3`. Both `xelatex` and `lualatex` likewise treat the Unicode letter `æ` as one argument.

In `latex` and `pdflatex`, however, `æ` is an active Unicode control sequence that uses two arguments: `#1#2`. The tail of the input, `bc`, is crowded into `#3`. Any macro where this `#1#2` pair is divided into `#1` and `#2` will produce one of two errors: `Unicode char ...not set up for LaTeX` or `Argument of \UTFviii@two@octets has an extra }`.

Testing for Fragility

3.0 We test if `\Umathchar` is not defined. If so, we check if the leading token of the argument matches the start of an active Unicode control sequence: If `\@car`⟨*test*⟩`\@nil` is equal to `\@car ß\@nil` (page 96) we capitalize `#1#2`, otherwise just `#1`. Should `#1` be a protected macro or something that does not expand to a sequence of letters, we use alternate formatting and `\AltCaps` (Section 2.7.2).

Back to Section 1.3

2.11.7 LaTeX Engines

This preamble snippet lets us build `nameauth`, e.g., on TL 2017. We load `iftex.sty` only if it exists. We load transitional packages when `iftex` is absent or old:[36]

```
1 \IfFileExists{iftex.sty}{\usepackage{iftex}}{}
2 \unless\ifdefined\RequireTUTeX
3   \usepackage{ifxetex}
4   \usepackage{ifluatex}
5   \usepackage{ifpdf}
6 \fi
```

[36] A copy of this example is in `examples.tex`, located with this manual.

If we Next we test for the LaTeX engine and include packages accordingly. We could just include inputenc either way, but we are illustrating a point about testing.

Some statements below should be modified at need. The font packages do affect nameauth indirectly. The use of *TikZ* does not, but it is easy to let such concerns also use the test below. With fontspec, Latin Modern is the default. Otherwise, Computer Modern is the default. If we **only** make `pdf` documents, the test below simplifies to a test for for `\Umathchar`, then loading either fontspec (success) or fontenc (failure).

```
\newif\ifDoTikZ                         % If dvi-only workflow
\ifxetex
  \usepackage{fontspec}
  \usepackage{polyglossia}
  \setdefaultlanguage{american}         % Use own language
  \usepackage{tikz}
  \DoTikZtrue                           % If dvi-only workflow
\else
  \ifluatex
    \ifpdf
      \usepackage{fontspec}
      \usepackage{polyglossia}
      \setdefaultlanguage{american}     % Use own language
      \usepackage{tikz}
      \DoTikZtrue                       % If dvi-only workflow
    \else
      \IfFileExists{utf8-2018.def}{}
      {\usepackage[utf8]{inputenc}}
      \usepackage[TS1,T1]{fontenc}
      \usepackage[american]{babel}      % Use own language
      \usepackage{lmodern}
      % Perhaps add \usepackage{tikz}
    \fi
  \else
    \IfFileExists{utf8-2018.def}{}
    {\usepackage[utf8]{inputenc}}
    \usepackage[TS1,T1]{fontenc}
    \usepackage[american]{babel}        % Use own language
    \usepackage{lmodern}
    \ifpdf
      \usepackage{tikz}                 % If dvi-only workflow
      \DoTikZtrue                       % If dvi-only workflow
    \fi
  \fi
\fi
```

To avoid problems, `\ifDoTikZ` can help one conditionally load *TikZ*. One can observe some `dvi` viewers (e.g., `yap`, `dviout`) crash either when loading or at some later point if one loads *TikZ*. Neither `xdvi` nor `advi` crash.

Using `xdvi` or `advi` (from WhizzyTeX) may result in certain aspects of *TikZ* not rendering correctly until conversion to `ps`/`pdf`. Using either `dvipdf` or `dvips` with `ps2pdf` will fix that. Using `dvipdfm` does not help here.

In the body text we can use something like the test below for:

doing pdf things

```
\ifxetex
  doing \texttt{pdf} things
\else
  \ifpdf
    doing \texttt{pdf} things
  \else
    doing \texttt{dvi} things
  \fi
\fi
```

The following equivalent conditional statements can help a macro or just the body text to work under multiple engines:

```
\ifxetex xelatex%
\else
  \ifluatex
    \ifpdf lualatex (pdf)%
    \else lualatex (dvi)%
    \fi
  \else
    \ifpdf pdflatex%
    \else latex (dvi)%
    \fi
  \fi
\fi
```

```
\unless\ifxetex
  \unless\ifluatex
    \ifpdf pdflatex%
    \else latex (dvi)%
    \fi
  \else
    \ifpdf lualatex (pdf)%
    \else lualatex (dvi)%
    \fi
  \fi
\else xelatex%
\fi
```

Back to Section 1.3

This space is intentionally left blank.

3 Implementation

3.1 Flags and Registers

General Process Flow Control

Warning Levels

This flag controls how many warnings you get. Defaults to few warnings. Verbose gives you plenty of warnings about cross-references and page entries in the index.

```
\newif\if@nameauth@Verbose
```

Who Called Me?

Various macros use these flags to protect against stack overflows or choose the right output.

```
\newif\if@nameauth@InAKA
\newif\if@nameauth@InName
\newif\if@nameauth@Xref
```

Core Macro Locks

`\@nameauth@Name`, `\AKA`, and other macros use a lock to avoid a stack overflow. With `\@nameauth@BigLock` one prevents execution. See also Sections 2.10.2 and 2.10.3.

```
\newif\if@nameauth@Lock
\newif\if@nameauth@BigLock
\newif\if@nameauth@InHook
```

Name Decision Paths

`\IfMainName`, `\IfFrontName`, and `\IfAKA` use locally-scoped paths by default. This flag causes that scoping not to be used.

```
\newif\if@nameauth@GlobalScope
```

Debugging

Both flags below are used to show name patterns and index entries in the text.

```
\newif\if@nameauth@IdxDebug
\newif\if@nameauth@LongIdxDebug
```

Indexing

The indexing flags permit or prevent indexing and tags. `\IndexActive` and `\IndexInctive` or the `index` and `noindex` options toggle the first flag; `\SkipIndex` toggles the second. `\JustIndex` toggles the third, which makes the core naming engine `\@nameauth@Name` act like a call to `\IndexName`:

```
\newif\if@nameauth@DoIndex
\newif\if@nameauth@SkipIndex
\newif\if@nameauth@JustIndex
```

The `pretag` and `nopretag` options toggle the flag below, which allows or prevents the insertion of index sort keys.

```
\newif\if@nameauth@Pretag
```

The first flag determines whether `\IndexRef` creates a *see* reference or a *see also* reference. The second determines how strict to be with *see* references.

```
\newif\if@nameauth@SeeAlso
\newif\if@nameauth@OldSee
```

Formatting

`\NamesActive` and `\NamesInactive`, with the `mainmatter` and `frontmatter` options, toggle formatting hooks via `\if@nameauth@MainFormat`. `\if@nameauth@AKAFormat` permits `\AKA` to call the first-use hooks once.

```
\newif\if@nameauth@MainFormat
\newif\if@nameauth@AKAFormat
```

The next flag works with `\LocalNames` and `\GlobalNames`.

```
\newif\if@nameauth@LocalNames
```

These three flags are used only for backward compatibility. The first broadly determines how per-name flags are reset; the second affects the behavior of `\JustIndex`; and the third toggles whether or not the name argument token registers are set globally.

```
\newif\if@nameauth@OldReset
\newif\if@nameauth@OldPass
\newif\if@nameauth@OldToks
```

These two flags trigger `\ForgetName` and `\SubvertName` within `\@nameauth@Name`.

```
\newif\if@nameauth@Forget
\newif\if@nameauth@Subvert
```

`\if@nameauth@FirstFormat` triggers the first-use hooks to be called; otherwise the second-use hooks are called. Additionally, `\if@nameauth@AlwaysFormat` forces this true, except when `\if@nameauth@AKAFormat` is false.

```
\newif\if@nameauth@FirstFormat
\newif\if@nameauth@AlwaysFormat
```

Specific Name Form Modifications

Affix Commas

The `comma` and `nocomma` options toggle the first flag value below. `\ShowComma` and `\NoComma` respectively toggle the second and third.

```
\newif\if@nameauth@AlwaysComma
\newif\if@nameauth@ShowComma
\newif\if@nameauth@NoComma
```

Name Breaking

`\KeepAffix` toggles the first flag below, while `\KeepName` toggles the second. Both affect the use of non-breaking spaces in the text.

```
\newif\if@nameauth@NBSP
\newif\if@nameauth@NBSPX
```

Detect Punctuation

This Boolean value is used to prevent double full stops at the end of a name in the text.

```
\newif\if@nameauth@Punct
```

Long and Short Names

`\if@nameauth@FullName` is true for a long name reference. `\if@nameauth@FirstName` disables full-name references and causes only Western forenames to be displayed. The default is to reset both globally on a per-name basis.

`\if@nameauth@AltAKA` is toggled respectively by `\AKA` and `\AKA*` to print a longer or shorter name. `\if@nameauth@OldAKA` forces the pre-3.0 behavior of `\AKA*`.

`\if@nameauth@ShortSNN` is used with `\DropAffix` to suppress the affix of a Western name. `\if@nameauth@EastFN` toggles the forced printing of Eastern forenames.

```
\newif\if@nameauth@FullName
\newif\if@nameauth@FirstName
\newif\if@nameauth@AltAKA
\newif\if@nameauth@OldAKA
\newif\if@nameauth@ShortSNN
\newif\if@nameauth@EastFN
```

Eastern Names

The next flags values govern name reversing and full surname capitalization. The first of each pair is a global state. The second of each pair is an individual state.

```
\newif\if@nameauth@RevAll
\newif\if@nameauth@RevThis
\newif\if@nameauth@AllCaps
\newif\if@nameauth@AllThis
```

Last-Comma-First

This pair of flags deals with Western names reordered in a list according to surname.

```
\newif\if@nameauth@RevAllComma
\newif\if@nameauth@RevThisComma
```

Cap First Letter and Format

The next flags deal with first-letter capitalization. `\CapThis` sets the first Boolean value. The second is triggered by `\@nameauth@UTFtest` when it encounters an active Unicode character. The third is a fallback triggered by `\AccentCapThis`. The fourth disables `\CapThis` for alternate formatting. The fifth toggles alternate formatting.

```
\newif\if@nameauth@DoCaps
\newif\if@nameauth@UTF
\newif\if@nameauth@Accent
\newif\if@nameauth@AltFormat
\newif\if@nameauth@DoAlt
```

Name Argument Token Registers

`\@nameauth@toksa` `\@nameauth@toksb` `\@nameauth@toksc`

These three token registers contain the current values of the name arguments passed to `\Name`, its variants, and the cross-reference arguments of `\AKA`. Users can access them especially in formatting hooks.

```
\newtoks\@nameauth@toksa
\newtoks\@nameauth@toksb
\newtoks\@nameauth@toksc
```

These three token registers contain the current values of the name arguments in each line of the `nameauth` environment.

```
\newtoks\@nameauth@etoksb
\newtoks\@nameauth@etoksc
\newtoks\@nameauth@etoksd
```

3.2 Hooks

`\NamesFormat` Post-process "first" instance of final complete name form in text. See Sections 2.6 and 2.10.1. Called when both `\@nameauth@MainFormat` and `\@nameauth@FirstFormat` are true.

```
\newcommand*\NamesFormat{}
```

`\MainNameHook` Post-process subsequent instance of final complete name form in main-matter text. See Sections 2.6 and 2.10.1f. Called when `\@nameauth@MainFormat` is true and the Boolean flag `\@nameauth@FirstFormat` is false.

```
\newcommand*\MainNameHook{}
```

`\FrontNamesFormat` Post-process "first" instance of final complete name form in front-matter text. Called when `\@nameauth@MainFormat` is false and `\@nameauth@FirstFormat` is true.

```
\newcommand*\FrontNamesFormat{}
```

\FrontNameHook Post-process subsequent instance of final complete name form in front-matter text. Called when \@nameauth@MainFormat is false and \@nameauth@FirstFormat is false.

```
\newcommand*\FrontNameHook{}
```

\NameauthName The last three hooks usually point to \@nameauth@Name. See Section 2.10.4.

```
\newcommand*\NameauthName{\@nameauth@Name}
```

\NameauthLName Customization hook called after \@nameauth@FullName is set true. See Section 2.10.4.

```
\newcommand*\NameauthLName{\@nameauth@Name}
```

\NameauthFName Customization hook called after \@nameauth@FirstName is set true. See Section 2.10.4.

```
\newcommand*\NameauthFName{\@nameauth@Name}
```

\NameauthIndex Customization hook that allows one to redefine what happens when any naming or indexing function calls the equivalent of \index. See Section 2.4.1.

```
\newcommand*\NameauthIndex{\index}
```

3.3 Package Options

The following package options interact with many of the prior Boolean values.

```
\DeclareOption{mainmatter}{\@nameauth@MainFormattrue}
\DeclareOption{frontmatter}{\@nameauth@MainFormatfalse}
\DeclareOption{alwaysformat}{\@nameauth@AlwaysFormattrue}
\DeclareOption{formatAKA}{\@nameauth@AKAFormattrue}
\DeclareOption{index}{\@nameauth@DoIndextrue}
\DeclareOption{noindex}{\@nameauth@DoIndexfalse}
\DeclareOption{pretag}{\@nameauth@Pretagtrue}
\DeclareOption{nopretag}{\@nameauth@Pretagfalse}
\DeclareOption{verbose}{\@nameauth@Verbosetrue}
\DeclareOption{globaltest}{\@nameauth@GlobalScopetrue}
\DeclareOption{oldAKA}{\@nameauth@OldAKAtrue}
\DeclareOption{oldreset}{\@nameauth@OldResettrue}
\DeclareOption{oldpass}{\@nameauth@OldPasstrue}
\DeclareOption{oldtoks}{\@nameauth@OldTokstrue}
\DeclareOption{oldsee}{\@nameauth@OldSeetrue}
\DeclareOption{nocomma}{\@nameauth@AlwaysCommafalse}
\DeclareOption{comma}{\@nameauth@AlwaysCommatrue}
\DeclareOption{normalcaps}{\@nameauth@AllCapsfalse}
\DeclareOption{allcaps}{\@nameauth@AllCapstrue}
\DeclareOption{notreversed}%
  {\@nameauth@RevAllfalse\@nameauth@RevAllCommafalse}
\DeclareOption{allreversed}%
  {\@nameauth@RevAlltrue\@nameauth@RevAllCommafalse}
\DeclareOption{allrevcomma}%
  {\@nameauth@RevAllfalse\@nameauth@RevAllCommatrue}
\DeclareOption{noformat}{\renewcommand*\NamesFormat{}}
\DeclareOption{smallcaps}{\renewcommand*\NamesFormat{\scshape}}
\DeclareOption{italic}{\renewcommand*\NamesFormat{\itshape}}
\DeclareOption{boldface}{\renewcommand*\NamesFormat{\bfseries}}
\DeclareOption{altformat}{%
  \@nameauth@AltFormattrue\@nameauth@DoAlttrue}
\ExecuteOptions%
  {nocomma,mainmatter,index,pretag,%
   normalcaps,notreversed,noformat}
\ProcessOptions\relax
```

Now we load the required packages. They facilitate the first/subsequent name uses, the parsing of arguments, and the implementation of starred forms.

```
\RequirePackage{etoolbox}
\RequirePackage{trimspaces}
\RequirePackage{suffix}
\RequirePackage{xargs}
```

3.4 Internal Macros

Internal Values

`\@nameauth@Actual` This sets the "actual" character used by nameauth for index sorting. This lets one use, for example, `\global\IndexActual{=}`.

```
\def\@nameauth@Actual{@}
```

`\@nameauth@Exclude` This makes an xref into an "exclusion". An exclusion is any name control sequence ending in `!PN` that expands to this value. See `\ExcludeName`.

```
\newcommand*\@nameauth@Exclude{!}
```

Name Control Sequence: Who Am I?

`\@nameauth@Clean` Thanks to Heiko Oberdiek, this macro produces a "sanitized" string to make a control sequence for a name. Testing the existence of that control sequence is the core of nameauth.

```
\newcommand*\@nameauth@Clean[1]
  {\expandafter\zap@space\detokenize{#1} \@empty}
```

`\@nameauth@MakeCS` Unless we are in `\AKA`, create a name control sequence in the core name engine.

```
\newcommand*\@nameauth@MakeCS[1]
{%
  \unless\ifcsname#1\endcsname
    \unless\if@nameauth@InAKA\csgdef{#1}{}\fi
  \fi
}
```

Parsing: Root and Suffix

`\@nameauth@Root` The following two macros return everything before a comma in ⟨*SNN*⟩.

```
\newcommand*\@nameauth@Root[1]{\@nameauth@@Root#1,\\}
```

`\@nameauth@@Root` Throw out the comma and suffix, return the radix.

```
\def\@nameauth@@Root#1,#2\\{\trim@spaces{#1}}
```

`\@nameauth@TrimTag` The following two macros return everything before a vertical bar (`|`) in an index tag.

```
\newcommand*\@nameauth@TrimTag[1]{\@nameauth@@TrimTag#1|\\}
```

`\@nameauth@@TrimTag` Throw out the bar and suffix, return the radix.

```
\def\@nameauth@@TrimTag#1|#2\\{#1}
```

`\@nameauth@Suffix` The following two macros parse ⟨*SNN*⟩ into a radix and a comma-delimited suffix, returning only the suffix after a comma in the argument, or nothing.

```
\newcommand*\@nameauth@Suffix[1]{\@nameauth@@Suffix#1,,\\}
```

`\@nameauth@@Suffix` Throw out the radix; return the suffix with no leading spaces. We use this when printing the suffix.

```
\def\@nameauth@@Suffix#1,#2,#3\\{%
  \ifx\\#2\\\@empty\else\trim@spaces{#2}\fi
}
```

\@nameauth@GetSuff The following two macros just grab the suffix for testing if the first non-space character is an active character from inputenc.

```
\newcommand*\@nameauth@GetSuff[1]{\@nameauth@@GetSuff#1,,\\}
```

\@nameauth@@GetSuff Throw out the radix; return the suffix.

```
\def\@nameauth@@GetSuff#1,#2,#3\\{#2}
```

Parsing: Capitalization

\@nameauth@TestToks Test if the leading token is the same as the leading token of an active Unicode character, using an *Esszett* (ß) as the control. We only run this macro if we are in the inputenc regime.

```
\newcommand*\@nameauth@TestToks[1]
{%
  \toks@\expandafter{\@car#1\@nil}%
  \edef\@nameauth@one{\the\toks@}%
  \toks@\expandafter{\@carß\@nil}%
  \edef\@nameauth@two{\the\toks@}%
  \ifx\@nameauth@one\@nameauth@two
    \@nameauth@UTFtrue%
  \else
    \@nameauth@UTFfalse%
  \fi
}
```

\@nameauth@UTFtest We choose how to capitalize a letter by determining if we are running under `xelatex` or `lualatex`. We test for `\Umathchar`. Then we see if inputenc is loaded. We set up the comparison and pass off to `\@nameauth@TestToks`.

```
\newcommand*\@nameauth@UTFtest[1]
{%
  \def\@nameauth@testarg{#1}%
  \ifdefined\Umathchar
    \@nameauth@UTFfalse%
  \else
    \ifdefined\UTFviii@two@octets
      \if@nameauth@Accent
        \@nameauth@UTFtrue\@nameauth@Accentfalse%
      \else
        \expandafter\@nameauth@TestToks%
          \expandafter{\@nameauth@testarg}%
      \fi
    \else
      \@nameauth@UTFfalse%
    \fi
  \fi
}
```

\@nameauth@UTFtestS This test is like the one above, but a special case when we have a suffix. We have to do a bit more in the way of expansion to get the comparison to work properly. Moreover, we only use this when the regular suffix macro is not `\@empty`.

```
\newcommand*\@nameauth@UTFtestS[1]
{%
  \expandafter\def\expandafter\@nameauth@testarg%
    \expandafter{\@nameauth@GetSuff{#1}}%
```

This following token register assignment looks weird, but it is how we get a test that works.

```
  \expandafter\toks@%
    \expandafter\expandafter\expandafter{\@nameauth@testarg}%
```

We take that token register and assign its value to a macro to do the test.

```
  \expandafter\def\expandafter\@nameauth@test@rg%
    \expandafter{\the\toks@}%
  \ifdefined\Umathchar
    \@nameauth@UTFfalse%
  \else
    \ifdefined\UTFviii@two@octets
      \if@nameauth@Accent
        \@nameauth@UTFtrue\@nameauth@Accentfalse%
      \else
        \expandafter\@nameauth@TestToks%
          \expandafter{\@nameauth@test@rg}%
      \fi
    \else
      \@nameauth@UTFfalse%
    \fi
  \fi
}
```

`\@nameauth@Cap` The following two macros cap the first letter of the argument.

```
\newcommand*\@nameauth@Cap[1]{\@nameauth@C@p#1\\}
```

`\@nameauth@C@p` Helper macro for the one above.

```
\def\@nameauth@C@p#1#2\\{%
  \expandafter\trim@spaces\expandafter{\MakeUppercase{#1}#2}%
}
```

`\@nameauth@CapUTF` The following two macros cap the first active Unicode letter under inputenc.

```
\newcommand*\@nameauth@CapUTF[1]{\@nameauth@C@pUTF#1\\}
```

`\@nameauth@C@pUTF` Helper macro for the one above.

```
\def\@nameauth@C@pUTF#1#2#3\\{%
  \expandafter\trim@spaces\expandafter{\MakeUppercase{#1#2}#3}%
}
```

`\@nameauth@CapArgs` Capitalize the first letter of all name arguments. Implements capitalization on demand in the body text (not the index) when not in alternate formatting. We only use this macro in the local scope of `\@nameauth@Parse`.

```
\newcommand*\@nameauth@CapArgs[3]
{%
  \ifdefined\@nameauth@InParser
    \unless\if@nameauth@AltFormat
      \let\carga\arga%
      \let\crootb\rootb%
      \let\csuffb\suffb%
      \let\cargc\argc%
```

We test the first argument for active Unicode characters, then cap the first letter.

```
      \unless\ifx\arga\@empty
        \def\test{#1}%
        \expandafter\@nameauth@UTFtest\expandafter{\test}%
```

Capitalize the first active Unicode character.

```
        \if@nameauth@UTF
          \expandafter\def\expandafter\carga\expandafter{%
            \expandafter\@nameauth@CapUTF\expandafter{\test}}%
```

Capitalize the first character (not active Unicode).

```
    \else
      \expandafter\def\expandafter\carga\expandafter{%
        \expandafter\@nameauth@Cap\expandafter{\test}}%
    \fi
  \fi
```

We test the root surname for active Unicode characters, then cap the first letter.

```
  \def\test{#2}%
  \expandafter\@nameauth@UTFtest\expandafter{\test}%
```

Capitalize the first active Unicode character.

```
  \if@nameauth@UTF
    \expandafter\def\expandafter\crootb\expandafter{%
      \expandafter\@nameauth@CapUTF\expandafter{\rootb}}%
```

Capitalize the first character (not active Unicode).

```
  \else
    \expandafter\def\expandafter\crootb\expandafter{%
      \expandafter\@nameauth@Cap\expandafter{\rootb}}%
  \fi
```

We test the suffix for active Unicode characters, then cap the first letter.

```
  \unless\ifx\suffb\@empty
    \def\test{#2}%
    \expandafter\@nameauth@UTFtestS\expandafter{\test}%
    \protected@edef\test{\@nameauth@GetSuff{#2}}%
```

Capitalize the first active Unicode character.

```
    \if@nameauth@UTF
      \protected@edef\test{\@nameauth@Suffix{#2}}%
      \expandafter\def\expandafter\csuffb\expandafter{%
        \expandafter\@nameauth@CapUTF\expandafter{\test}}%
```

Capitalize the first character (not active Unicode).

```
    \else
      \edef\@nameauth@test{\@nameauth@Suffix{#2}}%
      \expandafter\def\expandafter\csuffb\expandafter{%
        \expandafter\@nameauth@Cap\expandafter{\test}}%
    \fi
  \fi
```

We test the final argument for active Unicode characters, then cap the first letter.

```
  \unless\ifx\argc\@empty
    \def\test{#3}%
    \expandafter\@nameauth@UTFtest\expandafter{\test}%
```

Capitalize the first active Unicode character.

```
    \if@nameauth@UTF
      \expandafter\def\expandafter\cargc\expandafter{%
        \expandafter\@nameauth@CapUTF\expandafter{\test}}%
```

Capitalize the first character (not active Unicode).

```
    \else
      \expandafter\def\expandafter\cargc\expandafter{%
        \expandafter\@nameauth@Cap\expandafter{\test}}%
    \fi
  \fi
```

Let the arguments be the macros with caps.

```
      \let\arga\carga%
      \let\rootb\crootb%
      \let\suffb\csuffb%
      \let\argc\cargc%
    \fi
  \fi
}
```

Parsing: Punctuation Detection and Alteration

`\@nameauth@TestDot` This macro, based on a snippet by Uwe Lueck, checks for a full stop at the end of its argument using the two internal helper macros below.

```
\newcommand*\@nameauth@TestDot[1]
{%
```

If no full stop is present, `##1` is associated with the first `\@End`.. The second `\@End` gets absorbed as a parameter, leaving `##2` empty. If a full stop is present, `##2` will contain it.

```
  \def\@nameauth@TestD@t##1.\@End##2\\{\@nameauth@TestPunct{##2}}%
```

The two control sequences are equal if `##1` is empty (no full stop). If `##1` is not empty, it sets `\@nameauth@Puncttrue`, which triggers the call to `\@nameauth@CheckDot` below.

```
  \def\@nameauth@TestPunct##1%
  {%
    \ifx\@nameauth@TestPunct##1\@nameauth@TestPunct
    \else
      \global\@nameauth@Puncttrue%
    \fi
  }%
  \global\@nameauth@Punctfalse%
  \@nameauth@TestD@t#1\@End.\@End\\%
}
```

`\@nameauth@CheckDot` We assume that `\expandafter` precedes the invocation of `\@nameauth@CheckDot`, which only is called if the terminal character of the input is a period. We evaluate the lookahead `\@nameauth@token` while keeping it on the list of input tokens.

```
\newcommand*\@nameauth@CheckDot
  {\futurelet\@nameauth@token\@nameauth@EvalDot}
```

`\@nameauth@EvalDot` If `\@nameauth@token` is a full stop, we gobble the next token. Note that we cannot have this conditional statement span more than one line.

```
\newcommand*\@nameauth@EvalDot
{%
  \let\@nameauth@stop=.%
  \ifx\@nameauth@token\@nameauth@stop\expandafter\@gobble \fi
}
```

`\@nameauth@AddPunct` Here we govern whether spaces between name elements break or not, and whether to add commas or not. This all occurs in the text, not the index. The priority from highest to lowest is no commas, show commas, and always show commas. Much applies only to Western names, thus we check of $\langle FNN \rangle$ is empty or not. We only use this macro in the local scope of `\@nameauth@Parse`.

```
\newcommand*\@nameauth@AddPunct
{%
  \ifdefined\@nameauth@InParser
    \def\Space{ }%
    \def\SpaceX{ }%
```

`\SpaceX` is used for the space between a Western name and an affix, specifically tied to `\KeepAffix`. `\Space` is used for all other spaces between name elements.

```
    \if@nameauth@NBSP \edef\Space{\nobreakspace}\fi
    \if@nameauth@NBSPX \edef\SpaceX{\nobreakspace}\fi
```

Western names have a set of comma-use conventions that differ from all other name forms, so we only use the following logic if ⟨*FNN*⟩ is not empty, thus, a Western name.

```
    \unless\ifx\arga\@empty
      \if@nameauth@AlwaysComma
        \def\Space{, }%
        \if@nameauth@NBSP \edef\Space{,\nobreakspace}\fi
      \fi
      \if@nameauth@ShowComma
        \def\Space{, }%
        \if@nameauth@NBSP \edef\Space{,\nobreakspace}\fi
      \fi
      \if@nameauth@NoComma
        \def\Space{ }%
        \if@nameauth@NBSP \edef\Space{\nobreakspace}\fi
      \fi
    \fi
  \fi
}
```

Parsing: Name Argument Loading

`\@nameauth@LoadArgs` Assign name arguments to internal macros to determine name syntax. This is used in all macros that take name arguments.

```
\newcommand*\@nameauth@LoadArgs[3]
{%
```

We want these arguments to expand to `\@empty` (or not) when we test them.

```
  \protected@edef\@nameauth@A{\trim@spaces{#1}}%
  \protected@edef\@nameauth@B{\@nameauth@Root{#2}}%
  \protected@edef\@nameauth@SB{\@nameauth@Suffix{#2}}%
  \protected@edef\@nameauth@C{\trim@spaces{#3}}%
```

Make (usually) unique control sequence values from the name arguments.

```
  \def\@nameauth@csb{\@nameauth@Clean{#2}}%
  \def\@nameauth@csbc{\@nameauth@Clean{#2,#3}}%
  \def\@nameauth@csab{\@nameauth@Clean{#1!#2}}%
}
```

Parsing: Standard Parsing Logic

`\@nameauth@Choice` This standard logic applies to all macros that take name arguments.

```
\newcommand\@nameauth@Choice[3]
{%
  \ifx\@nameauth@A\@empty
    \ifx\@nameauth@C\@empty
```

This decision path is for non-Western names. The `#1` argument recurs below where `\@nameauth@SB` is present. Thus, for output to the text, the `#1` argument must test both `\@nameauth@C` and `\@nameauth@SB`, and swap the former with the latter if necessary. For output to the index or for handling control sequences, one ignores `\@nameauth@C`.

```
      #1%
    \else
      \ifx\@nameauth@SB\@empty
```

The #2 argument is only for non-Western names that use the obsolete syntax. In the #2 argument \@nameauth@SB never occurs. For indexing and control sequences, one cannot ignore the use of \@nameauth@C in this path.

```
        #2%
      \else
```

But if both \@nameauth@SB and \@nameauth@C are present, we invoke the #1 argument instead and let it do any further testing and processing.

```
        #1%
      \fi
    \fi
  \else
```

This decision path is for Western names. In those cases where one must work with name forms in the text, somewhere in the #3 argument one must test for \@nameauth@C and swap it for \@nameauth@A, as well as accounting for the presence or absence of \@nameauth@SB. Otherwise, for indexing and control sequences, one ignores \@nameauth@C in this path and handles \@nameauth@SB appropriately.

```
    #3%
  \fi
}
```

\@nameauth@Flags Reset flags after the naming macros and \AKA and friends create output in the text. This is not the only place where formatting flags are reset, but the other places in the core naming engine and name parser are special-use cases designed for the use of \JustIndex and macros like \PName.

```
\newcommand*\@nameauth@Flags
{%
  \if@nameauth@OldReset
```

The oldreset option implies not only a difference in scope regarding how flags are reset, but it also lets the effects of \ForgetThis and \SubvertThis to pass through \AKA and \AKA*. Regardless, we only reset \if@nameauth@AltAKA here due to macros like \PName.

```
    \if@nameauth@InAKA
      \@nameauth@AltAKAfalse%
    \fi
    \@nameauth@SkipIndexfalse%
    \if@nameauth@InName
      \@nameauth@Forgetfalse%
      \@nameauth@Subvertfalse%
    \fi
    \@nameauth@NBSPfalse%
    \@nameauth@NBSPXfalse%
    \@nameauth@DoCapsfalse%
    \@nameauth@Accentfalse%
    \@nameauth@AllThisfalse%
    \@nameauth@ShowCommafalse%
    \@nameauth@NoCommafalse%
    \@nameauth@RevThisfalse%
    \@nameauth@RevThisCommafalse%
    \@nameauth@ShortSNNfalse%
    \@nameauth@EastFNfalse%
  \else
```

The current way that the flags are reset makes them both global and more uniform, hopefully eliminating a few chances for errors that might be quite difficult to debug.

```
    \if@nameauth@InAKA
      \global\@nameauth@AltAKAfalse%
    \fi
    \global\@nameauth@SkipIndexfalse%
    \global\@nameauth@Forgetfalse%
    \global\@nameauth@Subvertfalse%
    \global\@nameauth@NBSPfalse%
    \global\@nameauth@NBSPXfalse%
    \global\@nameauth@DoCapsfalse%
    \global\@nameauth@Accentfalse%
    \global\@nameauth@AllThisfalse%
    \global\@nameauth@ShowCommafalse%
    \global\@nameauth@NoCommafalse%
    \global\@nameauth@RevThisfalse%
    \global\@nameauth@RevThisCommafalse%
    \global\@nameauth@ShortSNNfalse%
    \global\@nameauth@EastFNfalse%
  \fi
}
```

Error Detection and Debugging

`\@nameauth@Error` One can cause `nameauth` to halt with an error by leaving a required name argument empty, providing an argument that expands to empty, or creating an empty root within a malformed root/suffix pair. We provide meaningful feedback regarding these cases.

```
\newcommand*\@nameauth@Error[2]
{%
  \edef\@nameauth@msga{#2 SNN arg empty}%
  \edef\@nameauth@msgb{#2 SNN arg malformed}%
  \protected@edef\@nameauth@testname{\trim@spaces{#1}}%
  \protected@edef\@nameauth@testroot{\@nameauth@Root{#1}}%
  \ifx\@nameauth@testname\@empty
    \PackageError{nameauth}{\@nameauth@msga}%
  \fi
  \ifx\@nameauth@testroot\@empty
      \PackageError{nameauth}{\@nameauth@msgb}%
  \fi
}
```

`\@nameauth@Debug` In this Swiss-army knife for debugging, we set up a local scope because we make changes that would otherwise affect normal `nameauth` output. We redefine `\NameauthIndex` to print an argument in the text instead of the index, and we force indexing to occur.

```
\newcommandx*\@nameauth@Debug[3][1=\@empty, 3=\@empty]
{%
  \begingroup%
    \def\NameauthIndex##1{##1}%
    \@nameauth@DoIndextrue%
```

Process and load the arguments into the appropriate macros.

```
    \@nameauth@LoadArgs{#1}{#2}{#3}%
    \@nameauth@Error{#2}{macro \string\@nameauth@Debug}%
```

Below, given `\@nameauth@IdxDebugfalse` and `\@nameauth@LongIdxDebugfalse`, we produce the output of `\ShowPattern`.

Otherwise we locally delete any tag and xref control sequences as needed. They will be restored when the scope ends. If `\ShowIdxPageref` set `\@nameauth@IdxDebugtrue` and `\@nameauth@LongIdxDebugtrue` we produce a full index entry that shows all the tags and the "actual" character as well as the name. If `\ShowIdxPageref*` set `\@nameauth@IdxDebugtrue` we produce a short index entry that shows only the name.

```
    \@nameauth@Choice%
```

Non-Western names, current syntax.

```
    {%
      \unless\if@nameauth@IdxDebug
        \@nameauth@csb%
      \else
        \csundef{\@nameauth@csb!PN}%
        \unless\if@nameauth@LongIdxDebug
          \csundef{\@nameauth@csb!PRE}%
          \csundef{\@nameauth@csb!TAG}%
        \fi
        \IndexName[#1]{#2}[#3]%
      \fi
    }%
```

Non-Western names, obsolete syntax.

```
    {%
      \unless\if@nameauth@IdxDebug
        \@nameauth@csbc%
      \else
        \csundef{\@nameauth@csbc!PN}%
        \unless\if@nameauth@LongIdxDebug
          \csundef{\@nameauth@csbc!PRE}%
          \csundef{\@nameauth@csbc!TAG}%
        \fi
        \IndexName[#1]{#2}[#3]%
      \fi
    }%
```

Western names.

```
    {%
      \unless\if@nameauth@IdxDebug
        \@nameauth@csab%
      \else
        \csundef{\@nameauth@csab!PN}%
        \unless\if@nameauth@LongIdxDebug
          \csundef{\@nameauth@csab!PRE}%
          \csundef{\@nameauth@csab!TAG}%
        \fi
        \IndexName[#1]{#2}[#3]%
      \fi
    }%
```

We close the scope and reset the flags.

```
  \endgroup%
  \global\@nameauth@IdxDebugfalse%
  \global\@nameauth@LongIdxDebugfalse%
}
```

Core Name Engine

\@nameauth@Name Here is the heart of the package. Marc van Dongen provided the original basic structure. Parsing, indexing, and formatting are more discrete than in earlier versions.

```
\newcommandx*\@nameauth@Name[3][1=\@empty, 3=\@empty]
{%
```

Both \@nameauth@Name and \AKA engage the lock below, preventing a stack overflow. Tell the formatting mechanism that it is being called from \@nameauth@Name.

```
  \if@nameauth@BigLock \@nameauth@Locktrue\fi
  \unless\if@nameauth@Lock
    \@nameauth@Locktrue%
    \@nameauth@InNametrue%
```

Test for malformed input.

```
    \@nameauth@Error{#2}{macro \string\@nameauth@name}%
```

If we use \JustIndex then skip everything else. The oldpass option restores what we did before version 3.3, where we locally reset \@nameauth@JustIndexfalse and were done. Now, however, the default is a global reset to avoid undocumented behavior.

```
    \if@nameauth@JustIndex
      \IndexName[#1]{#2}[#3]%
      \if@nameauth@OldPass
        \@nameauth@JustIndexfalse%
      \else
        \if@nameauth@OldReset
          \@nameauth@FullNamefalse%
          \@nameauth@FirstNamefalse%
          \@nameauth@JustIndexfalse%
        \else
          \global\@nameauth@FullNamefalse%
          \global\@nameauth@FirstNamefalse%
          \global\@nameauth@JustIndexfalse%
        \fi
      \fi
    \else
```

Create or delete name cseq if directed, with deletion override. Ensure that names are printed in horizontal mode. Wrap the name with two index entries in case a page break occurs between name elements.

```
      \if@nameauth@Subvert \SubvertName[#1]{#2}[#3]\fi
      \if@nameauth@Forget  \ForgetName[#1]{#2}[#3]\fi
      \leavevmode\hbox{}%
      \unless\if@nameauth@SkipIndex \IndexName[#1]{#2}[#3]\fi
      \if@nameauth@MainFormat
        \@nameauth@Parse{#1}{#2}{#3}{!MN}%
      \else
        \@nameauth@Parse{#1}{#2}{#3}{!NF}%
      \fi
      \unless\if@nameauth@SkipIndex \IndexName[#1]{#2}[#3]\fi
```

Reset all the “per name” Boolean values after printing a name. The default is global.

```
      \@nameauth@Flags%
    \fi
    \@nameauth@Lockfalse%
    \@nameauth@InNamefalse%
```

Close the “locked” branch and complete the full stop detection and removal. This conditional statement must be on one line.

```
  \fi
  \if@nameauth@Punct\expandafter\@nameauth@CheckDot\fi
}
```

Core Name Engine: Syntactic Element Layer

`\@nameauth@Parse` Parse and print a name in the text. The final required argument tells us which naming system we are in (Section 2.11.5). Both `\@nameauth@Name` and `\AKA` call this parser, which only works in a locked state.

```
\newcommand\@nameauth@Parse[4]
{%
  \if@nameauth@BigLock \@nameauth@Lockfalse\fi
  \if@nameauth@Lock
```

Make token register copies of the current name args to be available for the hook macros.

```
    \if@nameauth@OldToks
      \@nameauth@toksa\expandafter{#1}%
      \@nameauth@toksb\expandafter{#2}%
      \@nameauth@toksc\expandafter{#3}%
    \else
      \global\@nameauth@toksa\expandafter{#1}%
      \global\@nameauth@toksb\expandafter{#2}%
      \global\@nameauth@toksc\expandafter{#3}%
    \fi
```

If global caps. reversing, and commas are true, set the per-name flags true.

```
      \if@nameauth@AllCaps      \@nameauth@AllThistrue\fi
      \if@nameauth@RevAll       \@nameauth@RevThistrue\fi
      \if@nameauth@RevAllComma \@nameauth@RevThisCommatrue\fi
```

Now we enter a local scope where we can use simple control strings without needing to worry about collisions. We process and load the arguments into the appropriate macros.

```
    \begingroup%
      \def\@nameauth@InParser{}%
      \@nameauth@LoadArgs{#1}{#2}{#3}%
```

Copy the protected control sequences to local, unprotected ones for backward compatibility and readability.

```
      \let\arga\@nameauth@A%
      \let\rootb\@nameauth@B%
      \let\suffb\@nameauth@SB%
      \let\argc\@nameauth@C%
```

Capitalization on demand in the body text if not in alternate formatting.

```
      \if@nameauth@DoCaps
        \@nameauth@CapArgs{#1}{#2}{#3}%
      \fi
```

We capitalize the entire surname when desired; different from above and overrides it.

```
      \if@nameauth@AllThis
        \protected@edef\rootb%
          {\MakeUppercase{\@nameauth@Root{#2}}}%
      \fi
```

Use non-breaking spaces and commas as desired.

```
      \@nameauth@AddPunct%
```

We parse names by attaching "meaning" to patterns of macro arguments primarily via `\FNN` and `\SNN`. Then we call the name printing macros, based on the optional arguments.

```
      \let\SNN\rootb%
      \@nameauth@Choice%
```

Non-Western names, current syntax. We test `\argc` and `\suffb` as needed.

```
      {%
        \ifx\argc\@empty
          \let\FNN\suffb%
        \else
          \let\FNN\argc%
        \fi
        \@nameauth@NonWest{\@nameauth@csb#4}%
        \@nameauth@MakeCS{\@nameauth@csb#4}%
      }%
```

Non-Western names, obsolete syntax. Here `\argc` is significant.

```
      {%
        \let\FNN\argc%
        \@nameauth@NonWest{\@nameauth@csbc#4}%
        \@nameauth@MakeCS{\@nameauth@csbc#4}%
      }%
```

Western names. We test for `\argc` and swap it for `\arga` and account for `\suffb`.

```
      {%
        \ifx\argc\@empty
          \let\FNN\arga%
        \else
          \let\FNN\argc%
        \fi
        \unless\ifx\suffb\@empty
          \def\SNN{\rootb\Space\suffb}%
          \if@nameauth@ShortSNN
            \let\SNN\rootb%
          \fi
        \fi
        \@nameauth@West{\@nameauth@csab#4}%
        \@nameauth@MakeCS{\@nameauth@csab#4}%
      }%
```

We end the local group and reset the flags for name forms here.

```
    \endgroup%
    \if@nameauth@OldReset
      \@nameauth@FullNamefalse%
      \@nameauth@FirstNamefalse%
      \@nameauth@FirstFormatfalse%
    \else
      \global\@nameauth@FullNamefalse%
      \global\@nameauth@FirstNamefalse%
      \global\@nameauth@FirstFormatfalse%
    \fi
  \fi
}
```

Core Name Engine: Name Display Layer

`\@nameauth@NonWest` Arrange forms of non-Western names. We inherit macros from the parser and only use this macro in the local scope of the parser.

```
\newcommand*\@nameauth@NonWest[1]
{%
  \ifdefined\@nameauth@InParser
    \@nameauth@Form{#1}%
    \ifx\FNN\@empty
```

```
      \@nameauth@Hook{\SNN}%
    \else
      \if@nameauth@FullName
        \if@nameauth@RevThis
          \@nameauth@Hook{\FNN\Space\SNN}%
        \else
          \@nameauth@Hook{\SNN\Space\FNN}%
        \fi
      \else
        \if@nameauth@FirstName
          \if@nameauth@EastFN
            \@nameauth@Hook{\FNN}%
          \else
            \@nameauth@Hook{\SNN}%
          \fi
        \else
          \@nameauth@Hook{\SNN}%
        \fi
      \fi
    \fi
  \fi
}
```

`\@nameauth@West` Arrange forms of Western names and "non-native" Eastern names. We inherit macros from the parser and only use this macro in the local scope of the parser.

```
\newcommand*\@nameauth@West[1]
{%
  \ifdefined\@nameauth@InParser
    \@nameauth@Form{#1}%
    \if@nameauth@FullName
      \if@nameauth@RevThis
        \@nameauth@Hook{\SNN\SpaceX\FNN}%
      \else
        \if@nameauth@RevThisComma
          \edef\RevSpace{,\SpaceX}%
          \@nameauth@Hook{\SNN\RevSpace\FNN}%
        \else
          \@nameauth@Hook{\FNN\SpaceX\SNN}%
        \fi
      \fi
    \else
      \if@nameauth@FirstName
        \@nameauth@Hook{\FNN}%
      \else
        \@nameauth@Hook{\rootb}%
      \fi
    \fi
  \fi
}
```

`\@nameauth@Form` Set up the flags per the formatting rules for first, subsequent, long, and short uses. We only use this macro in the local scope of the parser.

```
\newcommand*\@nameauth@Form[1]
{%
  \ifdefined\@nameauth@InParser
```

If the name does not exist yet or if the `alwaysformat` option is used, force first-use formatting, force a long name, and inhibit a short name.

```
    \unless\ifcsname#1\endcsname
      \@nameauth@FirstFormattrue%
      \@nameauth@FullNametrue%
      \@nameauth@FirstNamefalse%
    \else
      \if@nameauth@AlwaysFormat \@nameauth@FirstFormattrue\fi
    \fi
```

If we are not in `\AKA`, if a short name form is desired, inhibit a long form.

```
    \unless\if@nameauth@InAKA
      \if@nameauth@FirstName \@nameauth@FullNamefalse\fi
    \else
```

If we are in `\AKA` use special formatting rules. `\AKA*` acts like `\FName`, while `\AKA` acts like `\Name*`. Both prefer using the subsequent-use hooks unless the `formatAKA` option or the `alwaysformat` option are used.

```
      \if@nameauth@AltAKA
        \if@nameauth@OldAKA \@nameauth@EastFNtrue\fi
        \@nameauth@FullNamefalse%
        \@nameauth@FirstNametrue%
      \else
        \@nameauth@FullNametrue%
        \@nameauth@FirstNamefalse%
      \fi
      \unless\if@nameauth@AlwaysFormat
        \unless\if@nameauth@AKAFormat
          \@nameauth@FirstFormatfalse%
        \fi
      \fi
    \fi
  \fi
}
```

Core Name Engine: Format Hook Dispatcher

`\@nameauth@Hook` Boolean flags control which hook is called (first/subsequent use, name type). We only use this macro in the local scope of the parser.

```
\newcommand*\@nameauth@Hook[1]
{%
  \ifdefined\@nameauth@InParser
```

We tell the formatting hooks that they are in the hook dispatcher to enable alternate formatting. We test the printed name form to see if it has a trailing full stop.

```
    \@nameauth@InHooktrue%
    \protected@edef\test{#1}%
    \expandafter\@nameauth@TestDot\expandafter{\test}%
    \if@nameauth@MainFormat
```

We use the formatting hooks for the main-matter system.

```
      \if@nameauth@FirstFormat
        \bgroup\NamesFormat{#1}\egroup%
      \else
        \bgroup\MainNameHook{#1}\egroup%
      \fi
    \else
```

We use the formatting hooks for the front-matter system.

```
      \if@nameauth@FirstFormat
        \bgroup\FrontNamesFormat{#1}\egroup%
      \else
        \bgroup\FrontNameHook{#1}\egroup%
      \fi
    \fi
```

We tell the formatting hooks that they are not in the hook dispatcher.

```
    \@nameauth@InHookfalse%
  \fi
}
```

Indexing Internals: Entry Formatter

`\@nameauth@Index` This is the core index mechanism. If the indexing flag is true, create an index entry, otherwise do nothing. Add any tags automatically if they exist.

```
\newcommand*\@nameauth@Index[2]
{%
  \if@nameauth@DoIndex
```

If an index tag exists for the entry, get it. Also create a short version of the tag without any vertical bar or trailing macro. If we are creating a cross-reference, use the short tag, otherwise use the long tag.

```
    \ifcsname#1!TAG\endcsname
      \protected@edef\@nameauth@Tag{\csname#1!TAG\endcsname}%
      \expandafter\def\expandafter\@nameauth@ShortTag\expandafter{%
        \expandafter\@nameauth@TrimTag\expandafter{\@nameauth@Tag}}%
```

Create entries with a sorting tag and an info tag.

```
      \ifcsname#1!PRE\endcsname
        \protected@edef\@nameauth@Pre{\csname#1!PRE\endcsname}%
        \if@nameauth@Xref
          \protected@edef\@nameauth@IdxEntry%
            {\@nameauth@Pre#2\@nameauth@ShortTag}%
        \else
          \protected@edef\@nameauth@IdxEntry%
            {\@nameauth@Pre#2\@nameauth@Tag}%
        \fi
      \else
```

Create entries with just an info tag.

```
        \if@nameauth@Xref
          \protected@edef\@nameauth@IdxEntry%
            {#2\@nameauth@ShortTag}%
        \else
          \protected@edef\@nameauth@IdxEntry%
            {#2\@nameauth@Tag}%
        \fi
      \fi
    \else
```

Create entries with just a sorting tag.

```
      \ifcsname#1!PRE\endcsname
        \protected@edef\@nameauth@Pre{\csname#1!PRE\endcsname}%
        \protected@edef\@nameauth@IdxEntry{\@nameauth@Pre#2}%
      \else
        \protected@edef\@nameauth@IdxEntry{#2}%
      \fi
    \fi
```

Create entries with no tag.

```
    \expandafter\NameauthIndex\expandafter{\@nameauth@IdxEntry}%
  \fi
}
```

3.5 User Interface Macros: Prefix Macros

Syntactic Formatting — Capitalization

`\CapThis` Tells the root capping macro to cap the first character of all name elements.

```
\newcommand*\CapThis{\@nameauth@DoCapstrue}
```

`\AccentCapThis` Overrides the automatic test for active Unicode characters. This is a fall-back in case the automatic test for active Unicode characters fails.

```
\newcommand*\AccentCapThis
{%
  \@nameauth@Accenttrue%
  \@nameauth@DoCapstrue%
}
```

`\CapName` Capitalize entire required name. Overrides `\CapThis` for surnames.

```
\newcommand*\CapName{\@nameauth@AllThistrue}
```

`\AllCapsInactive` Turn off global surname capitalization.

```
\newcommand*\AllCapsInactive{\@nameauth@AllCapsfalse}
```

`\AllCapsActive` Turn on global surname capitalization. Activates `\CapName` for every name.

```
\newcommand*\AllCapsActive{\@nameauth@AllCapstrue}
```

Syntactic Formatting — Reversing

`\RevName` Reverse name order.

```
\newcommand*\RevName{\@nameauth@RevThistrue}
```

`\ReverseInactive` Turn off global name reversing.

```
\newcommand*\ReverseInactive{\@nameauth@RevAllfalse}
```

`\ReverseActive` Turn on global name reversing. Activates `\RevName` for every name.

```
\newcommand*\ReverseActive{\@nameauth@RevAlltrue}
```

`\ForceFN` Force the printing of an Eastern forename or ancient affix in the text, but only when using the "short name" macro `\FName` and the `\S`⟨*macro*⟩.

```
\newcommand*\ForceFN{\@nameauth@EastFNtrue}
```

Syntactic Formatting — Reversing with Commas

`\RevComma` Last name, comma, first name.

```
\newcommand*\RevComma{\@nameauth@RevThisCommatrue}
```

`\ReverseCommaInactive` Turn off global "last-name-comma-first".

```
\newcommand*\ReverseCommaInactive{\@nameauth@RevAllCommafalse}
```

`\ReverseCommaActive` Turn on global "last-name-comma-first". Activates `\RevComma` for every name.

```
\newcommand*\ReverseCommaActive{\@nameauth@RevAllCommatrue}
```

Alternate Formatting

`\AltFormatActive` Turn on alternate formatting, engage the formatting macros.

```
\newcommand*\AltFormatActive
{%
  \global\@nameauth@AltFormattrue%
  \global\@nameauth@DoAlttrue%
}
```

`\AltFormatActive*` Turn on alternate formatting, disengage the formatting macros.

```
\WithSuffix{\newcommand*}\AltFormatActive*
{%
  \global\@nameauth@AltFormattrue%
  \global\@nameauth@DoAltfalse%
}
```

`\AltFormatInactive` Turn off alternate formatting altogether.

```
\newcommand*\AltFormatInactive
{%
  \global\@nameauth@AltFormatfalse%
  \global\@nameauth@DoAltfalse%
}
```

`\AltOn` Locally turn on alternate formatting.

```
\newcommand*\AltOn
{%
  \if@nameauth@InHook
    \if@nameauth@AltFormat
      \@nameauth@DoAlttrue%
    \fi
  \fi
}
```

`\AltOff` Locally turn off alternate formatting.

```
\newcommand*\AltOff
{%
  \if@nameauth@InHook
    \if@nameauth@AltFormat
      \@nameauth@DoAltfalse%
    \fi
  \fi
}
```

`\AltCaps` Alternate discretionary capping macro triggered by `\CapThis`.

```
\newcommand*\AltCaps[1]
{%
  \if@nameauth@InHook
    \if@nameauth@DoCaps
      \MakeUppercase{#1}%
    \else
      #1%
    \fi
  \else
    #1%
  \fi
}
```

\textSC Alternate formatting macro: small caps when active.

```
\newcommand*\textSC[1]
{%
  \if@nameauth@DoAlt
    \textsc{#1}%
  \else
    #1%
  \fi
}
```

\textUC Alternate formatting macro: uppercase when active.

```
\newcommand*\textUC[1]
{%
  \if@nameauth@DoAlt
    \MakeUppercase{#1}%
  \else
    #1%
  \fi
}
```

\textIT Alternate formatting macro: italic when active.

```
\newcommand*\textIT[1]
{%
  \if@nameauth@DoAlt
    \textit{#1}%
  \else
    #1%
  \fi
}
```

\textBF Alternate formatting macro: boldface when active.

```
\newcommand*\textBF[1]
{%
  \if@nameauth@DoAlt
    \textbf{#1}%
  \else
    #1%
  \fi
}
```

Syntactic Formatting — Affixes

\ShowComma Put comma between name and suffix one time.

```
\newcommand*\ShowComma{\@nameauth@ShowCommatrue}
```

\NoComma Remove comma between name and suffix one time (with `comma` option).

```
\newcommand*\NoComma{\@nameauth@NoCommatrue}
```

\DropAffix Suppress the affix in a long Western name.

```
\newcommand*\DropAffix{\@nameauth@ShortSNNtrue}
```

\KeepAffix Trigger a name-suffix pair to be separated by a non-breaking space.

```
\newcommand*\KeepAffix{\@nameauth@NBSPtrue}
```

\KeepName Use non-breaking spaces between name syntactic forms.

```
\newcommand*\KeepName
{%
  \@nameauth@NBSPtrue%
  \@nameauth@NBSPXtrue%
}
```

Post-Processing — Main Versus Front Matter

\NamesInactive Switch to the "non-formatted" species of names.

```
\newcommand*\NamesInactive{\@nameauth@MainFormatfalse}
```

\NamesActive Switch to the "formatted" species of names.

```
\newcommand*\NamesActive{\@nameauth@MainFormattrue}
```

Name Decisions — First/Subsequent Reference

\ForgetThis Have the naming engine \@nameauth@Name call \ForgetName internally.

```
\newcommand*\ForgetThis{\@nameauth@Forgettrue}
```

\SubvertThis Have the naming engine \@nameauth@Name call \SubvertName internally.

```
\newcommand*\SubvertThis{\@nameauth@Subvertttrue}
```

\ForceName Set \@nameauth@FirstFormat to be true even for subsequent name uses. Works for one name only.

```
\newcommand*\ForceName{\@nameauth@FirstFormattrue}
```

\LocalNameTest Causes decision paths in the name decision macros to be in a local scope.

```
\newcommand*\LocalNameTest{\global\@nameauth@GlobalScopefalse}
```

\GlobalNameTest Causes decision paths in the name decision macros to have no scoping.

```
\newcommand*\GlobalNameTest{\global\@nameauth@GlobalScopetrue}
```

Name Occurrence Tweaks

\LocalNames \LocalNames sets @nameauth@LocalNames true so \ForgetName and \SubvertName do not affect both main and front matter naming systems.

```
\newcommand*\LocalNames{\global\@nameauth@LocalNamestrue}
```

\GlobalNames \GlobalNames sets @nameauth@LocalNames false. This restores the default behavior of \ForgetName and \SubvertName.

```
\newcommand*\GlobalNames{\global\@nameauth@LocalNamesfalse}
```

Index Operations

\IndexActual Change the "actual" character from the default. This allows one to use, for example, \global\IndexActual{=}.

```
\newcommand*\IndexActual[1]{\def\@nameauth@Actual{#1}}
```

\IndexInactive Turn off global indexing of names.

```
\newcommand*\IndexInactive{\@nameauth@DoIndexfalse}
```

\IndexActive Turn on global indexing of names.

```
\newcommand*\IndexActive{\@nameauth@DoIndextrue}
```

\SkipIndex Turn off the next instance of indexing in \Name, \FName, and starred forms.

```
\newcommand*\SkipIndex{\@nameauth@SkipIndextrue}
```

\JustIndex Makes the next call to \Name, \FName, and starred forms act like \IndexName. Overrides \SkipIndex.

```
\newcommand*\JustIndex{\@nameauth@JustIndextrue}
```

\SeeAlso Change the type of cross-reference from a *see* reference to a *see also* reference. Works once per xref, unless one uses \Include*, in which case, take care!

```
\newcommand*\SeeAlso{\@nameauth@SeeAlsotrue}
```

3.6 User Interface Macros: General

\ShowPattern This displays the pattern that the name arguments generate; maybe useful for debugging.

```
\newcommand*\ShowPattern{\@nameauth@Debug}
```

\ShowIdxPageref This displays (expanded, as printed) the index entry that will be generated, but not exactly what is in the idx file. This may be useful for debugging.

```
\newcommand*\ShowIdxPageref
{%
  \global\@nameauth@IdxDebugtrue%
  \global\@nameauth@LongIdxDebugtrue%
  \@nameauth@Debug%
}
```

\ShowIdxPageref* This displays a basic index entry with no tag. This may be useful for debugging.

```
\WithSuffix{\newcommand*}\ShowIdxPageref*
{%
  \global\@nameauth@IdxDebugtrue%
  \@nameauth@Debug%
}
```

\NameParser Generate a name form based on the current state of the nameauth macros in the locked path. Available for use only in the hook macros. We only use this macro in the local scope of the parser.

```
\newcommand*\NameParser
{%
  \if@nameauth@InHook
    \let\SNN\rootb%
    \@nameauth@Choice%
```

Non-Western names. We test both \argc and \suffb as needed.

```
    {%
      \ifx\argc\@empty
        \let\FNN\suffb%
      \else
        \let\FNN\argc%
      \fi
      \ifx\FNN\@empty
        \SNN%
      \else
        \if@nameauth@FullName
          \if@nameauth@RevThis
            \FNN\Space\SNN%
          \else
            \SNN\Space\FNN%
          \fi
        \else
          \if@nameauth@FirstName
```

```
            \if@nameauth@EastFN
              \FNN%
            \else
              \SNN%
            \fi
          \else
            \SNN%
          \fi
        \fi
      \fi
    }%
```

Non-Western names, obsolete syntax. Using `\argc` in this path affects indexing.

```
    {%
      \let\FNN\argc%
      \if@nameauth@FullName%
        \if@nameauth@RevThis
          \FNN\Space\SNN%
        \else
          \SNN\Space\FNN%
        \fi
      \else
        \if@nameauth@FirstName
          \if@nameauth@EastFN
            \FNN%
          \else
            \SNN%
          \fi
        \else
          \SNN%
        \fi
      \fi
    }%
```

Western names. We test for `\argc` and swap it for `\arga`, and account for `\suffb`.

```
    {%
      \ifx\argc\@empty
        \let\FNN\arga%
      \else
        \let\FNN\argc%
      \fi
      \unless\ifx\suffb\@empty
        \def\SNN{\rootb\Space\suffb}%
        \if@nameauth@ShortSNN
          \let\SNN\rootb%
        \fi%
      \fi
      \if@nameauth@FullName
        \if@nameauth@RevThis
          \SNN\SpaceX\FNN%
        \else
          \if@nameauth@RevThisComma
            \SNN\RevSpace\FNN%
          \else
            \FNN\SpaceX\SNN%
          \fi
        \fi
```

```
      \else
        \if@nameauth@FirstName
          \FNN%
        \else
          \let\SNN\rootb%
          \SNN%
        \fi
      \fi
    }%
  \fi
}
```

Traditional Naming Interface

\Name calls \NameauthName, the interface hook.

```
\newcommand\Name{\NameauthName}
```

\Name* sets up a long name reference and calls \NameauthLName, the interface hook.

```
\WithSuffix{\newcommand*}\Name*
{%
  \@nameauth@FullNametrue%
  \NameauthLName%
}
```

\FName sets up a short name reference and calls \NameauthFName, the interface hook.

```
\newcommand\FName
{%
  \@nameauth@FirstNametrue%
  \NameauthFName%
}
```

\FName* \FName and \FName* are identical in function.

```
\WithSuffix{\newcommand*}\FName*{\FName}
```

Index Operations

\IndexProtect We shut down all output from the naming and indexing macros to protect against problems in the index in case a macro in the index contains one of the naming macros.

```
\newcommand*\IndexProtect
{%
  \@nameauth@DoIndexfalse%
  \@nameauth@BigLocktrue%
}
```

\IndexName This creates an index entry with page references. It warns if the \SkipIndex prefix macro was used before it was called. It issues additional warnings if the verbose option is selected. It prints nothing. First we make copies of the arguments.

```
\newcommandx*\IndexName[3][1=\@empty, 3=\@empty]
{%
```

Process and load the arguments into the appropriate macros.

```
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \def\@nameauth@space{ }%
```

Test for malformed input.

```
  \@nameauth@Error{#2}{macro \string\IndexName}%
```

Warn if `\SkipIndex` was called before `\IndexName`, and reset it unless the `oldreset` option was used.

```
\if@nameauth@SkipIndex
  \PackageWarning{nameauth}%
  {\SkipIndex precedes \IndexName; check for problems}%
  \unless\if@nameauth@OldReset
    \@nameauth@SkipIndexfalse%
  \fi
\fi
```

Warn if `\SeeAlso` was called before `\IndexName` and reset it.

```
\unless\if@nameauth@OldReset
  \if@nameauth@SeeAlso
    \global\@nameauth@SeeAlsofalse%
    \PackageWarning{nameauth}%
    {\SeeAlso precedes \IndexName or a naming macro and was reset}%
  \fi
\fi
```

We create the appropriate index entries, calling `\@nameauth@Index` to handle sorting and tagging. We do not create an index entry for a cross-reference or exclusion.

```
\@nameauth@Choice%
```

Non-Western names. We ignore `\@nameauth@C` and handle `\@nameauth@SB` appropriately.

```
{%
  \ifcsname\@nameauth@csb!PN\endcsname
    \if@nameauth@Verbose
      \edef\@nameauth@testex{\csname\@nameauth@csb!PN\endcsname}%
      \ifx\@nameauth@testex\@nameauth@Exclude
        \PackageWarning{nameauth}%
        {macro \IndexName: Exclusion: #2 exists}%
      \else
        \PackageWarning{nameauth}%
        {macro \IndexName: XRef: #2 exists}%
      \fi
    \fi
  \else
    \ifx\@nameauth@SB\@empty
      \@nameauth@Index{\@nameauth@csb}{\@nameauth@B}%
    \else
      \@nameauth@Index{\@nameauth@csb}%
        {\@nameauth@B\@nameauth@space%
         \@nameauth@SB}%
    \fi
  \fi
}%
```

Non-Western names, obsolete syntax. Using `\@nameauth@C` in this path affects indexing.

```
{%
  \ifcsname\@nameauth@csbc!PN\endcsname
    \if@nameauth@Verbose
      \edef\@nameauth@testex{\csname\@nameauth@csbc!PN\endcsname}%
      \ifx\@nameauth@testex\@nameauth@Exclude
        \PackageWarning{nameauth}%
        {macro \IndexName: Exclusion: #2 #3 exists}%
      \else
        \PackageWarning{nameauth}%
        {macro \IndexName: XRef: #2 #3 exists}%
```

```
        \fi
      \fi
    \else
      \@nameauth@Index{\@nameauth@csbc}%
        {\@nameauth@B\@nameauth@space%
         \@nameauth@C}%
    \fi
  }%
```

Western names. We ignore `\@nameauth@C` and handle `\@nameauth@SB` appropriately.

```
  {%
    \ifcsname\@nameauth@csab!PN\endcsname
      \if@nameauth@Verbose
        \edef\@nameauth@testex{\csname\@nameauth@csab!PN\endcsname}%
        \ifx\@nameauth@testex\@nameauth@Exclude
          \PackageWarning{nameauth}%
          {macro \IndexName: Exclusion: #1 #2 exists}%
        \else
          \PackageWarning{nameauth}%
          {macro \IndexName: XRef: #1 #2 exists}%
        \fi
      \fi
    \else
      \ifx\@nameauth@SB\@empty
        \@nameauth@Index{\@nameauth@csab}%
          {\@nameauth@B,\@nameauth@space\@nameauth@A}%
      \else
        \@nameauth@Index{\@nameauth@csab}%
          {\@nameauth@B,\@nameauth@space%
           \@nameauth@A,\@nameauth@space\@nameauth@SB}%
      \fi
    \fi
  }%
}
```

`\IndexRef` Create a cross-reference that is not already an exclusion or a cross-reference. Print nothing.

```
\newcommandx*\IndexRef[4][1=\@empty, 3=\@empty]
{%
```

Process and load the arguments into the appropriate macros.

```
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \protected@edef\@nameauth@Target{#4}%
  \def\@nameauth@space{ }%
```

Test for malformed input.

```
  \@nameauth@Error{#2}{macro \string\IndexRef}%
  \@nameauth@Xreftrue%
```

Warn if `\SkipIndex` was called before `\IndexName`, and reset it unless the `oldreset` option was used.

```
  \if@nameauth@SkipIndex
    \PackageWarning{nameauth}%
    {\SkipIndex preceded \IndexRef; check for problems}%
    \unless\if@nameauth@OldReset
      \@nameauth@SkipIndexfalse%
    \fi
  \fi
```

We create either *see also* entries or *see* entries. The former are unrestricted with respect to names, not to extant cross-references. The latter are only created if they do not already exist as either page entries or cross-references.

```
\@nameauth@Choice%
```

Mononym or Eastern/ancient name, new syntax. First check if an xref or excluded, and if so, do nothing except issue warnings if so desired.

```
{%
  \ifcsname\@nameauth@csb!PN\endcsname
    \if@nameauth@Verbose
      \edef\@nameauth@testex{\csname\@nameauth@csb!PN\endcsname}%
      \ifx\@nameauth@testex\@nameauth@Exclude
        \PackageWarning{nameauth}%
        {macro \IndexRef: Exclusion: #2 exists}%
      \else
        \PackageWarning{nameauth}%
        {macro \IndexRef: XRef: #2 exists}%
      \fi
    \fi
```

If no xref or exclusion control sequence exists, either create a *see also* or a *see* reference. If the latter, forbid a *see* reference to an extant name unless the `oldsee` option is used; then allow, but issue a warning.

```
  \else
    \ifx\@nameauth@SB\@empty
      \if@nameauth@SeeAlso
        \@nameauth@Index{\@nameauth@csb}%
          {\@nameauth@B|seealso{\@nameauth@Target}}%
        \csgdef{\@nameauth@csb!PN}{}%
      \else
        \unless\if@nameauth@OldSee
          \unless\ifcsname\@nameauth@csb!MN\endcsname
            \unless\ifcsname\@nameauth@csb!NF\endcsname
              \@nameauth@Index{\@nameauth@csb}%
                {\@nameauth@B|see{\@nameauth@Target}}%
              \csgdef{\@nameauth@csb!PN}{}%
            \else
              \PackageWarning{nameauth}%
              {macro \IndexRef: Extant name #2 stops see ref.}%
            \fi
          \else
            \PackageWarning{nameauth}%
            {macro \IndexRef: Extant name #2 stops see ref.}%
          \fi
        \else
          \if@nameauth@Verbose
            \PackageWarning{nameauth}%
            {macro \IndexRef: Non-strict XRef #2 created}%
          \fi
          \@nameauth@Index{\@nameauth@csb}%
            {\@nameauth@B|see{\@nameauth@Target}}%
          \csgdef{\@nameauth@csb!PN}{}%
        \fi
      \fi
```

When the suffix is non-empty, either create a *see also* or a *see* reference. If the latter, forbid a *see* reference to an extant name unless the `oldsee` option is used; then allow and warn.

```
    \else
      \if@nameauth@SeeAlso
        \@nameauth@Index{\@nameauth@csb}%
          {\@nameauth@B\@nameauth@space%
           \@nameauth@SB|seealso{\@nameauth@Target}}%
        \csgdef{\@nameauth@csb!PN}{}%
      \else
        \unless\if@nameauth@OldSee
          \unless\ifcsname\@nameauth@csb!MN\endcsname
            \unless\ifcsname\@nameauth@csb!NF\endcsname
              \@nameauth@Index{\@nameauth@csb}%
                {\@nameauth@B\@nameauth@space%
                 \@nameauth@SB|see{\@nameauth@Target}}%
              \csgdef{\@nameauth@csb!PN}{}%
            \else
              \PackageWarning{nameauth}%
              {macro \IndexRef: Extant name #2 stops see ref.}%
            \fi
          \else
            \PackageWarning{nameauth}%
            {macro \IndexRef: Extant name #2 stops see ref.}%
          \fi
        \else
          \if@nameauth@Verbose
            \PackageWarning{nameauth}%
            {macro \IndexRef: Non-strict XRef #2 created}%
          \fi
          \@nameauth@Index{\@nameauth@csb}%
            {\@nameauth@B\@nameauth@space%
             \@nameauth@SB|see{\@nameauth@Target}}%
          \csgdef{\@nameauth@csb!PN}{}%
        \fi
      \fi
    \fi
  \fi
}%
```

Eastern or ancient name, obsolete syntax. First check if an xref or excluded.

```
{%
  \ifcsname\@nameauth@csbc!PN\endcsname
    \if@nameauth@Verbose
      \edef\@nameauth@testex{\csname\@nameauth@csbc!PN\endcsname}%
      \ifx\@nameauth@testex\@nameauth@Exclude
        \PackageWarning{nameauth}%
        {macro \IndexRef: Exclusion: #2 #3 exists}%
      \else
        \PackageWarning{nameauth}%
        {macro \IndexRef: XRef: #2 #3 exists}%
      \fi
    \fi
```

If no xref control sequence exists, either create a *see also* or a *see* reference. If the latter, forbid a *see* reference to an extant name unless the `oldsee` option is used; then allow, but issue a warning.

```
  \else
    \if@nameauth@SeeAlso
      \@nameauth@Index{\@nameauth@csbc}%
        {\@nameauth@B\@nameauth@space%
         \@nameauth@C|seealso{\@nameauth@Target}}%
      \csgdef{\@nameauth@csbc!PN}{}%
    \else
      \unless\if@nameauth@OldSee
        \unless\ifcsname\@nameauth@csbc!MN\endcsname
          \unless\ifcsname\@nameauth@csbc!NF\endcsname
            \@nameauth@Index{\@nameauth@csbc}%
              {\@nameauth@B\@nameauth@space%
               \@nameauth@C|see{\@nameauth@Target}}%
            \csgdef{\@nameauth@csbc!PN}{}%
          \else
            \PackageWarning{nameauth}%
            {macro \IndexRef: Extant name #2 #3 stops see ref.}%
          \fi
        \else
          \PackageWarning{nameauth}%
          {macro \IndexRef: Extant name #2 #3 stops see ref.}%
        \fi
      \else
        \if@nameauth@Verbose
          \PackageWarning{nameauth}%
          {macro \IndexRef: Non-strict XRef #2 #3 created}%
        \fi
        \@nameauth@Index{\@nameauth@csbc}%
          {\@nameauth@B\@nameauth@space%
           \@nameauth@C|see{\@nameauth@Target}}%
        \csgdef{\@nameauth@csbc!PN}{}%
      \fi
    \fi
  \fi
}%
```

Western name, without and with affix. First check if an xref or excluded.

```
{%
  \ifcsname\@nameauth@csab!PN\endcsname
    \if@nameauth@Verbose
      \edef\@nameauth@testex{\csname\@nameauth@csab!PN\endcsname}%
      \ifx\@nameauth@testex\@nameauth@Exclude
        \PackageWarning{nameauth}%
        {macro \IndexRef: Exclusion: #1 #2 exists}%
      \else
        \PackageWarning{nameauth}%
        {macro \IndexRef: XRef: #1 #2 exists}%
      \fi
    \fi
```

If no xref control sequence exists, either create a *see also* or a *see* reference. If the latter, forbid a *see* reference to an extant name unless the `oldsee` option is used; then allow, but issue a warning.

```
    \else
      \ifx\@nameauth@SB\@empty
        \if@nameauth@SeeAlso
          \@nameauth@Index{\@nameauth@csab}%
            {\@nameauth@B,\@nameauth@space%
             \@nameauth@A|seealso{\@nameauth@Target}}%
          \csgdef{\@nameauth@csab!PN}{}%
        \else
          \unless\if@nameauth@OldSee
            \unless\ifcsname\@nameauth@csab!MN\endcsname
              \unless\ifcsname\@nameauth@csab!NF\endcsname
                \@nameauth@Index{\@nameauth@csab}%
                  {\@nameauth@B,\@nameauth@space%
                   \@nameauth@A|see{\@nameauth@Target}}%
                \csgdef{\@nameauth@csab!PN}{}%
              \else
                \PackageWarning{nameauth}%
                {macro \IndexRef: Extant name #1 #2 stops see ref.}%
              \fi
            \else
              \PackageWarning{nameauth}%
              {macro \IndexRef: Extant name #1 #2 stops see ref.}%
            \fi
          \else
            \if@nameauth@Verbose
              \PackageWarning{nameauth}%
              {macro \IndexRef: Non-strict XRef #1 #2 created}%
            \fi
            \@nameauth@Index{\@nameauth@csab}%
              {\@nameauth@B,\@nameauth@space%
               \@nameauth@A|see{\@nameauth@Target}}%
            \csgdef{\@nameauth@csab!PN}{}%
          \fi
        \fi
```

When the suffix is non-empty, either create a *see also* or a *see* reference. If the latter, forbid a *see* reference to an extant name unless the `oldsee` option is used; then allow and warn.

```
      \else
        \if@nameauth@SeeAlso
          \@nameauth@Index{\@nameauth@csab}%
            {\@nameauth@B,\@nameauth@space%
             \@nameauth@A,\@nameauth@space%
             \@nameauth@SB|seealso{\@nameauth@Target}}%
          \csgdef{\@nameauth@csab!PN}{}%
        \else
          \unless\if@nameauth@OldSee
            \unless\ifcsname\@nameauth@csab!MN\endcsname
              \unless\ifcsname\@nameauth@csab!NF\endcsname
                \@nameauth@Index{\@nameauth@csab}%
                  {\@nameauth@B,\@nameauth@space%
                   \@nameauth@A,\@nameauth@space%
                   \@nameauth@SB|see{\@nameauth@Target}}%
                \csgdef{\@nameauth@csab!PN}{}%
```

```
              \else
                \PackageWarning{nameauth}%
                {macro \IndexRef: Extant name #1 #2 stops see ref.}%
              \fi
            \else
              \PackageWarning{nameauth}%
              {macro \IndexRef: Extant name #1 #2 stops see ref.}%
            \fi
          \else
            \if@nameauth@Verbose
              \PackageWarning{nameauth}%
              {macro \IndexRef: Non-strict XRef #1 #2 created}%
            \fi
            \@nameauth@Index{\@nameauth@csab}%
              {\@nameauth@B,\@nameauth@space%
               \@nameauth@A,\@nameauth@space%
               \@nameauth@SB|see{\@nameauth@Target}}%
            \csgdef{\@nameauth@csab!PN}{}%
          \fi
        \fi
      \fi
    \fi
  }%
  \@nameauth@Xreffalse%
  \if@nameauth@OldReset
    \@nameauth@SeeAlsofalse%
  \else
    \global\@nameauth@SeeAlsofalse%
  \fi
}
```

\ExcludeName Prevent a name from being indexed. Now, the set of macro expansions that comprise an exclusion contains one member: the value of `\@nameauth@Exclude`. Formerly, an exclusion was the set of all non-empty strings, preventing any other features from being added.

```
\newcommandx*\ExcludeName[3][1=\@empty, 3=\@empty]
{%
```

Process and load the arguments into the appropriate macros.

```
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\ExcludeName}%
```

Below we parse the name arguments and create an excluded form of cross-reference, unless one already exists.

```
  \@nameauth@Choice%
```

Non-Western names. Verbose warnings let one know that an extant name is being excluded, but the operation is still allowed.

```
  {%
    \if@nameauth@Verbose
      \ifcsname\@nameauth@csb!MN\endcsname
        \PackageWarning{nameauth}%
        {macro \ExcludeName: Name: #2 exists}%
      \fi
      \ifcsname\@nameauth@csb!NF\endcsname
        \PackageWarning{nameauth}%
        {macro \ExcludeName: Name: #2 exists}%
      \fi
    \fi
```

One cannot exclude an extant cross-reference or exclusion. Verbose warnings only.

```
    \ifcsname\@nameauth@csb!PN\endcsname
      \if@nameauth@Verbose
        \edef\@nameauth@testex{\csname\@nameauth@csb!PN\endcsname}%
        \ifx\@nameauth@testex\@nameauth@Exclude
          \PackageWarning{nameauth}%
          {macro \ExcludeName: Exclusion: #2 exists}%
        \else
          \PackageWarning{nameauth}%
          {macro \ExcludeName: XRef: #2 exists}%
        \fi
      \fi
    \else
      \csxdef{\@nameauth@csb!PN}{\@nameauth@Exclude}%
    \fi
  }%
```

Non-Western names, obsolete syntax. Verbose warnings let one know that an extant name is being excluded, but the operation is still allowed.

```
  {%
    \if@nameauth@Verbose
      \ifcsname\@nameauth@csbc!MN\endcsname
        \PackageWarning{nameauth}%
        {macro \ExcludeName: Name: #2 #3 exists}%
      \fi
      \ifcsname\@nameauth@csbc!NF\endcsname
        \PackageWarning{nameauth}%
        {macro \ExcludeName: Name: #2 #3 exists}%
      \fi
    \fi
```

One cannot exclude an extant cross-reference or exclusion. Verbose warnings only.

```
    \ifcsname\@nameauth@csbc!PN\endcsname
      \if@nameauth@Verbose
        \edef\@nameauth@testex{\csname\@nameauth@csbc!PN\endcsname}%
        \ifx\@nameauth@testex\@nameauth@Exclude
          \PackageWarning{nameauth}%
          {macro \ExcludeName: Exclusion: #2 #3 exists}%
        \else
          \PackageWarning{nameauth}%
          {macro \ExcludeName: XRef: #2 #3 exists}%
        \fi
      \fi
    \else
      \csxdef{\@nameauth@csbc!PN}{\@nameauth@Exclude}%
    \fi
  }%
```

Western names. Verbose warnings let one know that an extant name is being excluded, but the operation is still allowed.

```
  {%
    \if@nameauth@Verbose
      \ifcsname\@nameauth@csab!MN\endcsname
        \PackageWarning{nameauth}%
        {macro \ExcludeName: Name: #1 #2 exists}%
      \fi
      \ifcsname\@nameauth@csab!NF\endcsname
```

```
        \PackageWarning{nameauth}%
        {macro \ExcludeName: Name: #1 #2 exists}%
      \fi
    \fi
```

One cannot exclude an extant cross-reference or exclusion. Verbose warnings only.

```
    \ifcsname\@nameauth@csab!PN\endcsname
      \if@nameauth@Verbose
        \edef\@nameauth@testex{\csname\@nameauth@csab!PN\endcsname}%
        \ifx\@nameauth@testex\@nameauth@Exclude
          \PackageWarning{nameauth}%
          {macro \ExcludeName: Exclusion: #1 #2 exists}%
        \else
          \PackageWarning{nameauth}%
          {macro \ExcludeName: XRef: #1 #2 exists}%
        \fi
      \fi
    \else
      \csxdef{\@nameauth@csab!PN}{\@nameauth@Exclude}%
    \fi
  }%
}
```

`\IncludeName` This macro allows a name to be indexed once again only if it had been excluded.

```
\newcommandx*\IncludeName[3][1=\@empty, 3=\@empty]
{%
```

Process and load the arguments into the appropriate macros.

```
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\IncludeName}%
  \@nameauth@Choice%
```

Non-Western names.

```
  {%
    \ifcsname\@nameauth@csb!PN\endcsname
      \edef\@nameauth@testex{\csname\@nameauth@csb!PN\endcsname}%
      \ifx\@nameauth@testex\@nameauth@Exclude
        \global\csundef{\@nameauth@csb!PN}%
      \else
        \if@nameauth@Verbose
          \PackageWarning{nameauth}%
          {macro \IncludeName: Xref: #2 exists}%
        \fi
      \fi
    \fi
  }%
```

Non-Western names, obsolete syntax.

```
  {%
    \ifcsname\@nameauth@csbc!PN\endcsname
      \edef\@nameauth@testex{\csname\@nameauth@csbc!PN\endcsname}%
      \ifx\@nameauth@testex\@nameauth@Exclude
        \global\csundef{\@nameauth@csbc!PN}%
      \else
        \if@nameauth@Verbose
          \PackageWarning{nameauth}%
          {macro \IncludeName: Xref: #2 #3 exists}%
        \fi
```

```
      \fi
    \fi
  }%
```

Western names.

```
  {%
    \ifcsname\@nameauth@csab!PN\endcsname
      \edef\@nameauth@testex{\csname\@nameauth@csab!PN\endcsname}%
      \ifx\@nameauth@testex\@nameauth@Exclude
        \global\csundef{\@nameauth@csab!PN}%
      \else
        \if@nameauth@Verbose
          \PackageWarning{nameauth}%
          {macro \IncludeName: Xref: #1 #2 exists}%
        \fi
      \fi
    \fi
  }%
}
```

`\IncludeName*` This macro allows any name to be indexed by voiding any exclusion or cross-reference.

```
\WithSuffix{\newcommandx*}\IncludeName*[3][1=\@empty, 3=\@empty]
{%
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\IncludeName*}%
  \@nameauth@Choice%
  {\global\csundef{\@nameauth@csb!PN}}%
  {\global\csundef{\@nameauth@csbc!PN}}%
  {\global\csundef{\@nameauth@csab!PN}}%
}
```

`\PretagName` This creates an index entry tag that is applied before a name by `\@nameauth@Index`.

```
\newcommandx*\PretagName[4][1=\@empty, 3=\@empty]
{%
```

Process and load the arguments into the appropriate macros.

```
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\PretagName}%
```

Sort only when permitted.

```
  \if@nameauth@Pretag
    \@nameauth@Choice%
```

Non-Western names. Verbose warnings let us know if we are sorting either exclusions or cross-references. The former will be ignored. The latter will be used.

```
    {%
      \ifcsname\@nameauth@csb!PN\endcsname
        \if@nameauth@Verbose
          \edef\@nameauth@testex{\csname\@nameauth@csb!PN\endcsname}%
          \ifx\@nameauth@testex\@nameauth@Exclude
            \PackageWarning{nameauth}%
            {macro \PretagName: tagging exclusion: #2}%
          \else
            \PackageWarning{nameauth}%
            {macro \PretagName: tagging xref: #2}%
          \fi
        \fi
      \fi
```

```
      \csgdef{\@nameauth@csb!PRE}{#4\@nameauth@Actual}%
    }%
```

Non-Western names, obsolete syntax. Verbose warnings let us know if we are sorting either exclusions or cross-references. The former will be ignored. The latter will be used.

```
    {%
      \ifcsname\@nameauth@csbc!PN\endcsname
        \if@nameauth@Verbose
          \edef\@nameauth@testex{\csname\@nameauth@csbc!PN\endcsname}%
          \ifx\@nameauth@testex\@nameauth@Exclude
            \PackageWarning{nameauth}%
            {macro \PretagName: tagging exclusion: #2 #3}%
          \else
            \PackageWarning{nameauth}%
            {macro \PretagName: tagging xref: #2 #3}%
          \fi
        \fi
      \fi
      \csgdef{\@nameauth@csbc!PRE}{#4\@nameauth@Actual}%
    }%
```

Western names. Verbose warnings let us know if we are sorting either exclusions or cross-references. The former will be ignored. The latter will be used.

```
    {%
      \ifcsname\@nameauth@csab!PN\endcsname
        \if@nameauth@Verbose
          \edef\@nameauth@testex{\csname\@nameauth@csab!PN\endcsname}%
          \ifx\@nameauth@testex\@nameauth@Exclude
            \PackageWarning{nameauth}%
            {macro \PretagName: tagging exclusion: #1 #2}%
          \else
            \PackageWarning{nameauth}%
            {macro \PretagName: tagging xref: #1 #2}%
          \fi
        \fi
      \fi
      \csgdef{\@nameauth@csab!PRE}{#4\@nameauth@Actual}%
    }%
  \else
    \PackageWarning{nameauth}%
      {macro \PretagName: deactivated}%
  \fi
}
```

`\TagName` This creates an index entry tag for a name that is not either an exclusion or a cross-reference.

```
\newcommandx*\TagName[4][1=\@empty, 3=\@empty]
{%
```

Process and load the arguments into the appropriate macros.

```
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\TagName}%
  \@nameauth@Choice%
```

Non-Western names.

```
  {%
    \ifcsname\@nameauth@csb!PN\endcsname
      \if@nameauth@Verbose
        \edef\@nameauth@testex{\csname\@nameauth@csb!PN\endcsname}%
```

```
        \ifx\@nameauth@testex\@nameauth@Exclude
          \PackageWarning{nameauth}%
          {macro \TagName: not tagging exclusion: #2}%
        \else
          \PackageWarning{nameauth}%
          {macro \TagName: not tagging xref: #2}%
        \fi
      \fi
    \else
      \csgdef{\@nameauth@csb!TAG}{#4}%
    \fi
  }%
```

Non-Western names, obsolete syntax.

```
  {%
    \ifcsname\@nameauth@csbc!PN\endcsname
      \if@nameauth@Verbose
        \edef\@nameauth@testex{\csname\@nameauth@csbc!PN\endcsname}%
        \ifx\@nameauth@testex\@nameauth@Exclude
          \PackageWarning{nameauth}%
          {macro \TagName: not tagging exclusion: #2 #3}%
        \else
          \PackageWarning{nameauth}%
          {macro \TagName: not tagging xref: #2 #3}%
        \fi
      \fi
    \else
      \csgdef{\@nameauth@csbc!TAG}{#4}%
    \fi
  }%
```

Western names.

```
  {%
    \ifcsname\@nameauth@csab!PN\endcsname
      \if@nameauth@Verbose
        \edef\@nameauth@testex{\csname\@nameauth@csab!PN\endcsname}%
        \ifx\@nameauth@testex\@nameauth@Exclude
          \PackageWarning{nameauth}%
          {macro \TagName: not tagging exclusion: #1 #2}%
        \else
          \PackageWarning{nameauth}%
          {macro \TagName: not tagging xref: #1 #2}%
        \fi
      \fi
    \else
      \csgdef{\@nameauth@csab!TAG}{#4}%
    \fi
  }%
}
```

`\UntagName` This deletes an index tag.

```
\newcommandx*\UntagName[3][1=\@empty, 3=\@empty]
{%
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\UntagName}%
  \@nameauth@Choice%
  {\global\csundef{\@nameauth@csb!TAG}}%
```

```
  {\global\csundef{\@nameauth@csbc!TAG}}%
  {\global\csundef{\@nameauth@csab!TAG}}%
}
```

Name Info Data Set: "Text Tags"

\NameAddInfo This creates a macro that expands to information associated with a given name, similar to an index tag, but usable in the body text.

```
\newcommandx\NameAddInfo[4][1=\@empty, 3=\@empty]
{%
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\NameAddInfo}%
  \@nameauth@Choice%
  {\csgdef{\@nameauth@csb!DB}{#4}}%
  {\csgdef{\@nameauth@csbc!DB}{#4}}%
  {\csgdef{\@nameauth@csab!DB}{#4}}%
}
```

\NameQueryInfo This prints the information created by \NameAddInfo if it exists.

```
\newcommandx*\NameQueryInfo[3][1=\@empty, 3=\@empty]
{%
  \unless\if@nameauth@BigLock
    \@nameauth@LoadArgs{#1}{#2}{#3}%
    \@nameauth@Error{#2}{macro \string\NameQueryInfo}%
    \@nameauth@Choice%
    {\ifcsname\@nameauth@csb!DB\endcsname
      \csname\@nameauth@csb!DB\endcsname\fi}%
    {\ifcsname\@nameauth@csbc!DB\endcsname
      \csname\@nameauth@csbc!DB\endcsname\fi}%
    {\ifcsname\@nameauth@csab!DB\endcsname
      \csname\@nameauth@csab!DB\endcsname\fi}%
  \fi
}
```

\NameClearInfo This deletes a text tag. It has the same structure as \UntagName.

```
\newcommandx*\NameClearInfo[3][1=\@empty, 3=\@empty]
{%
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\NameClearInfo}%
  \@nameauth@Choice%
  {\global\csundef{\@nameauth@csb!DB}}%
  {\global\csundef{\@nameauth@csbc!DB}}%
  {\global\csundef{\@nameauth@csab!DB}}%
}
```

This space is intentionally left blank.

Name Decisions

`\IfMainName` This macro expands one path if a main matter name exists, or else the other. The state of `\if@nameauth@GlobalScope` determines whether or not the paths are in a local scope.

```
\newcommandx\IfMainName[5][1=\@empty, 3=\@empty]
{%
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\IfMainName}%
  \@nameauth@Choice%
  {%
    \ifcsname\@nameauth@csb!MN\endcsname
      \if@nameauth@GlobalScope
        #4%
      \else
        {#4}%
      \fi
    \else
      \if@nameauth@GlobalScope
        #5%
      \else
        {#5}%
      \fi
    \fi
  }%
  {%
    \ifcsname\@nameauth@csbc!MN\endcsname
      \if@nameauth@GlobalScope
        #4%
      \else
        {#4}%
      \fi
    \else
      \if@nameauth@GlobalScope
        #5%
      \else
        {#5}%
      \fi
    \fi
  }%
  {%
    \ifcsname\@nameauth@csab!MN\endcsname
      \if@nameauth@GlobalScope
        #4%
      \else
        {#4}%
      \fi
    \else
      \if@nameauth@GlobalScope
        #5%
      \else
        {#5}%
      \fi
    \fi
  }%
}
```

\IfFrontName This macro expands one path if a front matter name exists, or else the other. The state of \if@nameauth@GlobalScope determines whether or not the paths are in a local scope.

```
\newcommandx\IfFrontName[5][1=\@empty, 3=\@empty]
{%
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\IfFrontName}%
  \@nameauth@Choice%
  {%
    \ifcsname\@nameauth@csb!NF\endcsname
      \if@nameauth@GlobalScope
        #4%
      \else
        {#4}%
      \fi
    \else
      \if@nameauth@GlobalScope
        #5%
      \else
        {#5}%
      \fi
    \fi
  }%
  {%
    \ifcsname\@nameauth@csbc!NF\endcsname
      \if@nameauth@GlobalScope
        #4%
      \else
        {#4}%
      \fi
    \else
      \if@nameauth@GlobalScope
        #5%
      \else
        {#5}%
      \fi
    \fi
  }%
  {%
    \ifcsname\@nameauth@csab!NF\endcsname
      \if@nameauth@GlobalScope
        #4%
      \else
        {#4}%
      \fi
    \else
      \if@nameauth@GlobalScope
        #5%
      \else
        {#5}%
      \fi
    \fi
  }%
}
```

`\IfAKA` This macro expands one path if a cross-reference exists, another if it does not exist, and a third if it is excluded. The state of `\if@nameauth@GlobalScope` determines whether or not the paths are in a local scope.

```
\newcommandx\IfAKA[6][1=\@empty, 3=\@empty]
{%
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\IfAKA}%
```

For each class of name we test first if a cross-reference exists, then if it is excluded.

```
  \@nameauth@Choice%
  {%
    \ifcsname\@nameauth@csb!PN\endcsname
      \edef\@nameauth@testex{\csname\@nameauth@csb!PN\endcsname}%
      \ifx\@nameauth@testex\@nameauth@Exclude
        \if@nameauth@GlobalScope
          #6%
        \else
          {#6}%
        \fi
      \else
        \if@nameauth@GlobalScope
          #4%
        \else
          {#4}%
        \fi
      \fi
    \else
      \if@nameauth@GlobalScope
        #5%
      \else
        {#5}%
      \fi
    \fi
  }%
  {%
    \ifcsname\@nameauth@csbc!PN\endcsname
      \edef\@nameauth@testex{\csname\@nameauth@csbc!PN\endcsname}%
      \ifx\@nameauth@testex\@nameauth@Exclude
        \if@nameauth@GlobalScope
          #6%
        \else
          {#6}%
        \fi
      \else
        \if@nameauth@GlobalScope
          #4%
        \else
          {#4}%
        \fi
      \fi
    \else
      \if@nameauth@GlobalScope
        #5%
      \else
        {#5}%
      \fi
```

```
    \fi
  }%
  {%
    \ifcsname\@nameauth@csab!PN\endcsname
      \edef\@nameauth@testex{\csname\@nameauth@csab!PN\endcsname}%
      \ifx\@nameauth@testex\@nameauth@Exclude
        \if@nameauth@GlobalScope
          #6%
        \else
          {#6}%
        \fi
      \else
        \if@nameauth@GlobalScope
          #4%
        \else
          {#4}%
        \fi
      \fi
    \else
      \if@nameauth@GlobalScope
        #5%
      \else
        {#5}%
      \fi
    \fi
  }%
}
```

Changing Name Decisions

`\ForgetName` This undefines a control sequence to force a "first use".

```
\newcommandx*\ForgetName[3][1=\@empty, 3=\@empty]
{%
```

Process and load the arguments into the appropriate macros.

```
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\ForgetName}%
```

Now we parse the arguments, undefining the control sequences either by current name type (via `@nameauth@MainFormat`) or completely (toggled by `@nameauth@LocalNames`).

```
  \@nameauth@Choice%
```

Non-Western names.

```
  {%
    \if@nameauth@LocalNames
      \if@nameauth@MainFormat
        \global\csundef{\@nameauth@csb!MN}%
      \else
        \global\csundef{\@nameauth@csb!NF}%
      \fi
    \else
      \global\csundef{\@nameauth@csb!MN}%
      \global\csundef{\@nameauth@csb!NF}%
    \fi
  }%
```

Non-Western names, obsolete syntax.

```
  {%
```

```
    \if@nameauth@LocalNames
      \if@nameauth@MainFormat
        \global\csundef{\@nameauth@csbc!MN}%
      \else
        \global\csundef{\@nameauth@csbc!NF}%
      \fi
    \else
      \global\csundef{\@nameauth@csbc!MN}%
      \global\csundef{\@nameauth@csbc!NF}%
    \fi
  }%
```

Western names.

```
  {%
    \if@nameauth@LocalNames
      \if@nameauth@MainFormat
        \global\csundef{\@nameauth@csab!MN}%
      \else
        \global\csundef{\@nameauth@csab!NF}%
      \fi
    \else
      \global\csundef{\@nameauth@csab!MN}%
      \global\csundef{\@nameauth@csab!NF}%
    \fi
  }%
}
```

`\SubvertName` This defines a control sequence to force a "subsequent use".

```
\newcommandx*\SubvertName[3][1=\@empty, 3=\@empty]
{%
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\SubvertName}%
```

Now we parse the arguments, defining the control sequences either locally by section type or globally. `@nameauth@LocalNames` toggles the local or global behavior, while we select the type of name with `@nameauth@MainFormat`.

```
  \@nameauth@Choice%
```

Non-Western names.

```
  {%
    \if@nameauth@LocalNames
      \if@nameauth@MainFormat
        \csgdef{\@nameauth@csb!MN}{}%
      \else
        \csgdef{\@nameauth@csb!NF}{}%
      \fi
    \else
      \csgdef{\@nameauth@csb!MN}{}%
      \csgdef{\@nameauth@csb!NF}{}%
    \fi
  }%
```

Non-Western names, obsolete syntax.

```
  {%
    \if@nameauth@LocalNames
      \if@nameauth@MainFormat
        \csgdef{\@nameauth@csbc!MN}{}%
      \else
```

```
        \csgdef{\@nameauth@csbc!NF}{}%
      \fi
    \else
      \csgdef{\@nameauth@csbc!MN}{}%
      \csgdef{\@nameauth@csbc!NF}{}%
    \fi
  }%
```

Western names.

```
  {%
    \if@nameauth@LocalNames
      \if@nameauth@MainFormat
        \csgdef{\@nameauth@csab!MN}{}%
      \else
        \csgdef{\@nameauth@csab!NF}{}%
      \fi
    \else
      \csgdef{\@nameauth@csab!MN}{}%
      \csgdef{\@nameauth@csab!NF}{}%
    \fi
  }%
}
```

Alternate Names

\AKA `\AKA` prints an alternate name and creates index cross-references. It prevents multiple generation of cross-references and suppresses double periods.

```
\newcommandx*\AKA[5][1=\@empty, 3=\@empty, 5=\@empty]
{%
```

Prevent entering `\AKA` via itself or `\@nameauth@Name`. Prevents and resets `\JustIndex`. Tell the formatting system that `\AKA` is running.

```
  \if@nameauth@BigLock
    \@nameauth@Locktrue%
  \fi
  \unless\if@nameauth@Lock
    \@nameauth@Locktrue%
    \@nameauth@InAKAtrue%
    \if@nameauth@OldReset
      \@nameauth@JustIndexfalse%
    \else
      \global\@nameauth@JustIndexfalse%
    \fi
```

Test for malformed input.

```
    \@nameauth@Error{#2}{macro \string\AKA}%
    \@nameauth@Error{#4}{macro \string\AKA}%
```

Names occur in horizontal mode; we ensure that. Next we make copies of the target name arguments and we parse and print the cross-reference name.

```
    \leavevmode\hbox{}%
    \protected@edef\@nameauth@Ai{\trim@spaces{#1}}%
    \protected@edef\@nameauth@Bi{\@nameauth@Root{#2}}%
    \protected@edef\@nameauth@Si{\@nameauth@Suffix{#2}}%
    \@nameauth@Parse{#3}{#4}{#5}{!PN}%
    \def\@nameauth@space{ }%
```

Create an index cross-reference based on the arguments.

```
    \unless\if@nameauth@SkipIndex
      \ifx\@nameauth@Ai\@empty
        \ifx\@nameauth@Si\@empty
          \IndexRef[#3]{#4}[#5]{\@nameauth@Bi}%
        \else
          \IndexRef[#3]{#4}[#5]%
          {\@nameauth@Bi\@nameauth@space%
           \@nameauth@Si}%
        \fi
      \else
        \ifx\@nameauth@Si\@empty
          \IndexRef[#3]{#4}[#5]%
          {\@nameauth@Bi,\@nameauth@space\@nameauth@Ai}%
        \else
          \IndexRef[#3]{#4}[#5]%
          {\@nameauth@Bi,\@nameauth@space%
           \@nameauth@Ai,\@nameauth@space\@nameauth@Si}%
        \fi
      \fi
    \fi
```

Reset all the "per name" Boolean values. The default is global.

```
    \@nameauth@Flags%
    \@nameauth@Lockfalse%
    \@nameauth@InAKAfalse%
```

Close the "locked" branch and call the full stop detection. This conditional statement must be on one line.

```
  \fi
  \if@nameauth@Punct\expandafter\@nameauth@CheckDot\fi
}
```

`\AKA*` This starred form sets a Boolean to print only the alternate name argument, if that exists, and calls `\AKA`.

```
\WithSuffix{\newcommand*}\AKA*{\@nameauth@AltAKAtrue \AKA}
```

`\PName` `\PName` is a convenience macro that calls `\NameauthName`, then `\AKA`.

```
\newcommandx*\PName[5][1=\@empty,3=\@empty,5=\@empty]
{%
```

If we used `\JustIndex`, we ignore and reset its flag to false.

```
  \if@nameauth@OldReset
    \@nameauth@JustIndexfalse%
  \else
    \global\@nameauth@JustIndexfalse%
  \fi
```

If we used `\SkipIndex`, we reset the flag of `\SeeAlso` and activate `\SkipIndex`for both `\NameauthName` and `\AKA`.

```
  \if@nameauth@SkipIndex
    \unless\if@nameauth@OldReset
      \global\@nameauth@SeeAlsofalse%
    \fi
    \NameauthName[#1]{#2} (\SkipIndex\AKA[#1]{#2}[#3]{#4}[#5])%
  \else
```

Otherwise, if we used `\SeeAlso` we set the flag of `\SeeAlso` false for `\NameauthName` and true for `\AKA`. The "normal" case after that is trivial.

```
    \if@nameauth@SeeAlso
      \@nameauth@SeeAlsofalse\NameauthName[#1]{#2}
      \@nameauth@SeeAlsotrue(\AKA[#1]{#2}[#3]{#4}[#5])%
    \else
      \NameauthName[#1]{#2}
      (\AKA[#1]{#2}[#3]{#4}[#5])%
    \fi
  \fi
```

Warn if `\SkipIndex` remains in effect (potentially due to the `oldreset` option). Normally, this state should not occur.

```
  \if@nameauth@SkipIndex
    \PackageWarning{nameauth}%
    {\SkipIndex still active after \PName; check for problems}%
  \fi
}
```

`\PName*` This sets up a long name reference and calls `\PName`.

```
\WithSuffix{\newcommand*}\PName*{\@nameauth@FullNametrue \PName}
```

Quick Interface

`nameauth` The `nameauth` environment creates macro shorthands. First we define a control sequence `\<` that takes four parameters, delimited by three ampersands and `>`. This macro is defined only within the `nameauth` environment, but the shorthand macros that it creates are globally defined.

```
\newenvironment{nameauth}{%
  \begingroup%
  \let\ex\expandafter%
  \csdef{<}##1&##2&##3&##4>{%
    \protected@edef\@arga@{\trim@spaces{##1}}%
    \protected@edef\@larga@{L\trim@spaces{##1}}%
    \protected@edef\@sarga@{S\trim@spaces{##1}}%
    \protected@edef\@testb@{\trim@spaces{##2}}%
    \protected@edef\@testd@{\trim@spaces{##4}}%
    \@nameauth@etoksb\ex{##2}%
    \@nameauth@etoksc\ex{##3}%
    \@nameauth@etoksd\ex{##4}%
```

The first argument must have some text to create a set of control sequences with it. The third argument is the required name argument. Redefining a shorthand creates a warning.

```
    \ifx\@arga@\@empty
      \PackageError{nameauth}%
      {environment nameauth: Control sequence missing}%
    \fi
    \@nameauth@Error{##3}{environment nameauth}%
    \ifcsname\@arga@\endcsname
      \PackageWarning{nameauth}%
      {environment nameauth: Shorthand macro already exists}%
    \fi
```

Set up shorthands according to name form. We have to use `\ex`, not the ϵ-TEX way, due to `\protected@edef` in the naming macros.

We begin with mononyms and non-Western names that use the new syntax. We use one \ex per token because we only have one argument to expand first.

```
\ifx\@testd@\@empty
  \ifx\@testb@\@empty
    \ex\csgdef\ex{\ex\@arga@\ex}%
      \ex{\ex\NameauthName\ex{\the\@nameauth@etoksc}}%
    \ex\csgdef\ex{\ex\@larga@\ex}%
      \ex{\ex\@nameauth@FullNametrue%
      \ex\NameauthLName\ex{\the\@nameauth@etoksc}}%
    \ex\csgdef\ex{\ex\@sarga@\ex}%
      \ex{\ex\@nameauth@FirstNametrue%
      \ex\NameauthFName\ex{\the\@nameauth@etoksc}}%
  \else
```

Next we have Western names with no alternate names. Here we have two arguments to expand in reverse order, so we need three, then one uses of \ex per token.

```
    \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@arga@\ex\ex\ex}%
      \ex\ex\ex{\ex\ex\ex\NameauthName%
      \ex\ex\ex[\ex\the\ex\@nameauth@etoksb\ex]%
      \ex{\the\@nameauth@etoksc}}%
    \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@larga@\ex\ex\ex}%
      \ex\ex\ex{\ex\ex\ex\@nameauth@FullNametrue%
      \ex\ex\ex\NameauthLName%
      \ex\ex\ex[\ex\the\ex\@nameauth@etoksb\ex]%
      \ex{\the\@nameauth@etoksc}}%
    \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@sarga@\ex\ex\ex}%
      \ex\ex\ex{\ex\ex\ex\@nameauth@FirstNametrue%
      \ex\ex\ex\NameauthFName%
      \ex\ex\ex[\ex\the\ex\@nameauth@etoksb\ex]%
      \ex{\the\@nameauth@etoksc}}%
  \fi
\else
```

Below are “native” Eastern names with alternates and the older syntax. Again, we have three or one use of \ex per step before the respective arguments.

```
  \ifx\@testb@\@empty
    \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@arga@\ex\ex\ex}%
      \ex\ex\ex{\ex\ex\ex\NameauthName%
      \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
      \ex[\the\@nameauth@etoksd]}%
    \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@larga@\ex\ex\ex}%
      \ex\ex\ex{\ex\ex\ex\@nameauth@FullNametrue%
      \ex\ex\ex\NameauthLName%
      \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
      \ex[\the\@nameauth@etoksd]}%
    \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@sarga@\ex\ex\ex}%
      \ex\ex\ex{\ex\ex\ex\@nameauth@FirstNametrue%
      \ex\ex\ex\NameauthFName%
      \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
      \ex[\the\@nameauth@etoksd]}%
  \else
```

Here are Western names with alternates. We have three arguments to expand, so we have seven, three, and one respective use of \ex.

```
        \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
          \ex\ex\ex\ex\ex\ex\ex\@arga@\ex\ex\ex\ex\ex\ex\ex}%
          \ex\ex\ex\ex\ex\ex\ex\ex{\ex\ex\ex\ex\ex\ex\ex\NameauthName%
          \ex\ex\ex\ex\ex\ex\ex\ex[\ex\ex\ex\the%
          \ex\ex\ex\@nameauth@etoksb\ex\ex\ex]%
          \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
          \ex[\the\@nameauth@etoksd]}%
        \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
          \ex\ex\ex\ex\ex\ex\ex\@larga@\ex\ex\ex\ex\ex\ex\ex}%
          \ex\ex\ex\ex\ex\ex\ex\ex{%
          \ex\ex\ex\ex\ex\ex\ex\@nameauth@FullNametrue%
          \ex\ex\ex\ex\ex\ex\ex\NameauthLName%
          \ex\ex\ex\ex\ex\ex\ex\ex[\ex\ex\ex\the\ex\ex\ex%
          \@nameauth@etoksb\ex\ex\ex]%
          \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
          \ex[\the\@nameauth@etoksd]}%
        \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
          \ex\ex\ex\ex\ex\ex\ex\@sarga@\ex\ex\ex\ex\ex\ex\ex}%
          \ex\ex\ex\ex\ex\ex\ex\ex{%
          \ex\ex\ex\ex\ex\ex\ex\@nameauth@FirstNametrue%
          \ex\ex\ex\ex\ex\ex\ex\NameauthFName%
          \ex\ex\ex\ex\ex\ex\ex\ex[\ex\ex\ex\the\ex\ex\ex%
          \@nameauth@etoksb\ex\ex\ex]%
          \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
          \ex[\the\@nameauth@etoksd]}%
      \fi
    \fi\ignorespaces%
  }\ignorespaces%
}{\endgroup\ignorespaces}
```

4 Change History

5 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.