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

Charles P. Schaum†

Released 2025/08/27

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 instances.

Contents

*This file describes version 4.2, last revised 2025/08/27.

†Email: charles[dot]schaum@comcast.net

Disclaimer

This manual mentions names of historical figures both living and deceased. We intend to use all names herein with respect, for teaching purposes only, and never to express or imply any disrespect or bias.

Fair Use

Works quoted herein either are in the public domain or they are under copyright, but cited under the terms of fair use. The transformative purpose is to show the evocative power of words and names.

Thanks

For assistance, thanks to Marc van Dongen, Enrico Gregorio, Philipp Stephani, Heiko Oberdiek, Uwe Lueck, Dan Luecking, Robert Schlicht, Werner Lemberg, and others.

In memoriam Robin Fairbairns

He was very kind when I first uploaded nameauth, and gracious thereafter as well.

> I have already intimated to you the danger of parties in the State, with particular reference to the founding of them on geographical discriminations. Let me now take a more comprehensive view, and warn you in the most solemn manner against the baneful effects of the spirit of party generally.
>
> This spirit, unfortunately, is inseparable from our nature, having its root in the strongest passions of the human mind. It exists under different shapes in all governments, more or less stifled, controlled, or repressed; but, in those of the popular form, it is seen in its greatest rankness, and is truly their worst enemy.
>
> The alternate domination of one faction over another, sharpened by the spirit of revenge, natural to party dissension, which in different ages and countries has perpetrated the most horrid enormities, is itself a frightful despotism. But this leads at length to a more formal and permanent despotism. The disorders and miseries which result gradually incline the minds of men to seek security and repose in the absolute power of an individual; and sooner or later the chief of some prevailing faction, more able or more fortunate than his competitors, turns this disposition to the purposes of his own elevation, on the ruins of public liberty.
>
> —George Washington
> Farewell Address (1796)

1 Quick Start

A **name authority** (Section 10) is a canonical list of names to which variant names refer. Books can have hundreds of names and index entries. The nameauth package assists academic and business writing:

- Automate the syntax and formatting of names.
- Manage and display information that is associated with names.
- Make decisions relating to names.
- Sort name index entries properly.
- Automatically add information to index entries.
- Ensure correct indexing of names that span page breaks.
- Change name forms in the text while retaining consistent index entries.
- Use culturally appropriate name forms in the text and index.
- Do not force users to use any particular naming convention.
- Permit European academic conventions ("Continental" formatting).

Indexing rules are based on Nancy C. Mulvany, *Indexing Books* (Chicago: University of Chicago Press, 1994). All references [Mulvany] refer to this edition.[1]

1.1 Simple Example

Using the default nameauth options and formatting hooks (not the general formatting used in this manual; cf. Section 9.1), we show how names change automatically when we reorder statements by African-American author Charles Waddell Chesnutt.[2]

The `nameauth` environment (Section 1.5) resembles a `tabular` with four columns. In column one we define name shorthands. Columns two and three hold the names. We leave column four empty. `\ForgetName` (Section 9.3) used before Group 2 lets us "start over" with the names by forgetting their patterns.

```
\documentclass{article}
\input{compat.tex} % Included with nameauth; example file aids
% compatibility across different LaTeX versions and engines.
\usepackage{makeidx}
\usepackage{nameauth}
\makeindex

\begin{nameauth}
%    Col. 1   Col. 2      Col. 3     Col. 4
  \< Doug   & Frederick & Douglass &        >
  \< Bailey & Betsey    & Bailey   &        >
\end{nameauth}

\begin{document}

```

[1] Compare *The Chicago Manual of Style* (15th ed., Chicago: Chicago UP, 2003, 309f., 755f.) or newer editions. All references [Chicago] refer to this edition.

[2] Chesnutt, *Frederick Douglass* (Boston: Small, Maynard, 1899). See also **this web page**.

```
\textbf{Group 1}
\begin{enumerate}
  \item[\textbf{1.}] \Doug\ rose to eminence by sheer force
    of character and talents that neither slavery nor caste
    proscription could crush.
  \item[\textbf{2.}] \Doug's early life is perhaps the most
    complete indictment of the slave system ever presented at
    the bar of public opinion.
  \item[\textbf{3.}] \Doug\ was born in February, 1817. His
    earliest memories centered around the cabin of his
    grandmother, \Bailey.
\end{enumerate}

\textbf{Group 2} (Statement order changed.)
\ForgetName[Frederick]{Douglass}
\ForgetName[Betsey]{Bailey}
\begin{enumerate}
  \item[\textbf{2.}] \Doug's early life is perhaps the most
    complete indictment of the slave system ever presented at
    the bar of public opinion.
  \item[\textbf{3.}] \Doug\ was born in February, 1817. His
    earliest memories centered around the cabin of his
    grandmother, \Bailey.
  \item[\textbf{1.}] \Doug\ rose to eminence by sheer force
    of character and talents that neither slavery nor caste
    proscription could crush.
\end{enumerate}

\printindex
\end{document}
```

Group 1

1. Frederick Douglass rose to eminence by sheer force of character and talents that neither slavery nor caste proscription could crush.
2. Douglass's early life is perhaps the most complete indictment of the slave system ever presented at the bar of public opinion.
3. Douglass was born in February, 1817. His earliest memories centered around the cabin of his grandmother, Betsey Bailey.

Group 2 (Statement order changed.)

2. Frederick Douglass's early life is perhaps the most complete indictment of the slave system ever presented at the bar of public opinion.
3. Douglass was born in February, 1817. His earliest memories centered around the cabin of his grandmother, Betsey Bailey.
1. Douglass rose to eminence by sheer force of character and talents that neither slavery nor caste proscription could crush.

The goal of this example is to suggest that, if one has a project typeset using LaTeX, it takes a lot of time and attention to ensure the proper appearance of names. If a unit of text gets moved somewhere else, the knock-on effect could involve re-examining many names to see if their forms are acceptable. Our mission is to automate the editing process to a degree and let names take care of themselves.

1.2 How To Use the Manual

This manual tries to support various learning styles by using layout, colors, shapes, and similar ordering of both document sections and package code.

Macro Argument Types

Mandatory args in black **Optional args in dark red**

Scope and Sequence

The table of contents goes from simpler, frequently-used features to more complex, infrequent ones. The end of each major section has a return link to the TOC. Thus, as the section numbers increase in the lists below, so does the complexity.

General concepts:	1, 7, 8, 9, 10
Multilingual:	5, 11
Details and debugging:	2, 3, 4, 6, 12

Key Concepts

Starting in Section 1.4, **name patterns** appear in the margin (overview in Section 6.2), which are key to how names work in nameauth. Through Section 5.10.2 **basic index entries** appear in the margin to show how macro arguments and index entries relate. Thereafter, we use margin notes less due to greater complexity.

Special Signs and Typography

We highlight First Uses and Later Uses of names (Sections 9.1–9.3).

```
\renewcommand*\FrontNamesFormat{\color{violet}\sffamily}
\renewcommand*\FrontNameHook{\color{darkgray}\sffamily}
\renewcommand*\NamesFormat{\color{blue}\sffamily}
\renewcommand*\MainNameHook{\sffamily}
```

† A dagger indicates reversed Western forms (Sections 5.5).

‡ A double dagger shows usage of the obsolete syntax (Section 12.2).

§ A section mark denotes index entries of fictional names.


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

You profess to believe that "of one blood God made all nations of men to dwell on the face of the earth" — and hath commanded all men, everywhere, to love one another — yet you notoriously hate (and glory in your hatred!) all men whose skins are not colored like your own.

—Frederick Douglass
speech at Rochester, NY, 1852)

General Advice

- See `README.md` for details about package development and testing.
- Both `compat.tex` and `examples.tex` contain useful information. They are located with this manual.
- Do not put naming macros withing a macro defined with `\edef`.
- Naming macros can be arguments of a macro defined with `\edef`.
- In `dtx` files, it can be simpler to put the `nameauth` environment and initial tagging macros in the `<driver>` part. Yet `\newif` statements must occur in the "commented part".

We summarize some issues related to backward compatibility, which nameauth takes quite seriously as a partner to those who want to preserve their publications.

- Starting with version 4.0, nameauth has used xparse for starred macros and name arguments, which offers many benefits.
- Users of TEX distribution before 2018 might encounter unexpected or undefined behavior. To mitigate them, see Sections 4.3.1, 4.3.8f., 7.6.3, and 11.5.2. See also `compat.tex`.
- Older customizations using xargs may still work with the `oldargs` option. See Section 2.6 and `examples.tex`. It includes additional code snippets that deal with compatibility. If one should require nameauth version 3.7, see section 7 of `README.md`.
- To use older TEX distributions (2018 and before), one must include the textcomp package for backward compatibility.
- If users need to preserve older TEX distributions in order to maintain publications for reprint, they are encouraged to view and modify `compat.tex`, then `\input` it into the preambles of their projects.

> Man is by nature a social animal; an individual who is unsocial naturally and not accidentally is either beneath our notice or more than human. Society is something that precedes the individual. Anyone who either cannot lead the common life or is so self-sufficient as not to need to, and therefore does not partake of society, is either a beast or a god.
>
> —Aristotle
> *Politics* (post-338 BC)

1.3 Basic Concepts

1.3.1 Name Ambiguity

Culture resolves ambiguity.

Western culture often includes forenames followed by a surname and an optional affix. Exceptions can include patronyms like Leif Erikson and Llywelyn ap Gruffudd, as well as ancient and royal names.[3] Due to the conventions taught to them, Western readers might misinterpret the following names:

Forenames:	Marcus Tullius	Surname:	Cicero
Forename:	Pontius	Surname:	Pilate
Forename:	Jesus	Surname:	Christ

Yet these names have meanings that cannot be known **as signs in themselves** apart from their native cultural setting. That reveals the following information:

Praenomen:	Marcus	*Nomen*:	Tullius	*Cognomen*:	Cicero
⟨*empty*⟩		*Nomen*:	Pontius	*Cognomen*:	Pilatus
⟨*empty*⟩		Name:	Jesus	Sobriquet:	Christ

Roman names have a forename, a clan name, and one or more names that can denote branch families, nicknames, etc. Forenames (*praenomina*) do not carry great weight; the clan name (*nomen*) is of most importance. The clan name of Cicero actually is Tullius. In English, he used to be known as Tully. Pontius Pilate has no forename. Pontius is a clan name. The "nickname" (*cognomen*) Pilatus refers to skill with the *pilum*, a barbed spear. The name Jesus Christ comes from Ἰησοῦς ὁ Χριστός, from Hebrew *Y'shua ha-Mashiach*, "Joshua the Anointed One".

We embrace ambiguity.

The market determines usage, whether general or scholarly (Section 5.10).[4] Here we represent Cicero as a Western name and both Pilate and Christ as Non-Western names. Cicero is indexed by his *cognomen*; Pilate by his *nomen*.[5]

- Western Index Entry:

Text	Macro / Arguments	Index
M.T. Cicero	`\Name[M.T.]{Cicero}`	Cicero, M.T.

- Non-Western Index Entries:

Text	Macro / Arguments	Index
Pontius Pilate	`\Name{Pontius, Pilate}`	Pontius Pilate
Jesus Christ	`\Name{Jesus, Christ}`	Jesus Christ

[3] Indexed here under Erikson and Llywelyn. See **this document** on Welsh names.

[4] For an example of a resource that tries to span both the popular and the scholarly, see John Bartlett, *Bartlett's Familiar Quotations*, 16th ed., ed. Justin Kaplan (Boston: Little, Brown, 1991).

[5] `\IndexRef{Pilate}{Pontius Pilate}` makes "Pilate . . . *see* Pontius Pilate", a cross-reference that guides Western readers from the expected index entry to the entry used here (Section 7.3).

Name forms in the text can be changed independently of their index entries. This is necessary for cases like Hungarian names, Roman names, or when a publisher wants to use Western index entry forms with Eastern names.

For example, the Hungarian name Liszt Ferenc† has the index entry "Liszt, Ferenc". We used the macro arguments for Western names to get the correct index entry, but we reversed the name in the text via `\RevName\Name[Ferenc]{Liszt}`. Franz Liszt is his equivalent German name, cross-referenced to the Hungarian form.

Name arguments determine index entries.

The way that one encodes names using macro arguments primarily determines the form of the index entry — not the mutable form in the text. This is the key reason to use nameauth over manual methods. We do not index the names below, but we show how the index entries would appear:

- Western Index Entries:

Text	Macro / Arguments	Index
First Last Affix	`\Name[First]{Last, Affix}`	Last, First, Affix
Last	`\Name[First]{Last, Affix}`	Last, First, Affix
First Last	`\Name[First]{Last}`	Last, First
Last	`\Name[First]{Last}`	Last, First
Last First	`\RevName\Name*[First]{Last}`	Last, First

- Non-Western Index Entries:

Text	Macro / Arguments	Index
Family Person	`\Name{Family, Person}`	Family Person
Family	`\Name{Family, Person}`	Family Person
Person Family	`\RevName\Name*{Family, Person}`	Family Person
Person Affix	`\Name{Person, Affix}`	Person Affix
Person	`\Name{Person, Affix}`	Person Affix
Person	`\Name{Person}`	Person

Name complexity creates nameauth complexity.

We might take names for granted until we have to consider them, use them in multicultural contexts, index them, and so on. Even if one does not use the nameauth package, one cannot escape this complexity.

The process of making nameauth became the means by which the present author learned about both the many intricate and fascinating complexities of names and the complexities of LaTeX. One hopes that this package might facilitate accurate and respectful cross-cultural use of names in quality publications.

> The space program is not only scientific in purpose but also is an expression of man's insistent determination to do the nearly impossible — to explore the unknown, even at great risk.
>
> —Harold Urey (1961)

1.3.2 Name Arguments

We describe the general form of names according to the current nameauth syntax.[6] Mandatory and optional **macro arguments** aid a natural-language approach. They are subordinate to **required and optional name elements**, shown by black and red text. Issues with final optional arguments are covered in Section 4.3.1.

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

Forename(s):⟨*FNN*⟩	Surname(s):⟨*SNN*⟩	Qualifier:⟨*Affix*⟩
Personal name(s): *baptismal name* *Christian name* *multiple names* *(prae)nomen*[7]	Family name: *of father, mother* *ancestor, vocation* *origin, region* *nomen / cognomen* *patronym*	Sobriquet / title: *Sr., Jr., III. . .* *notable attribute* *origin, region* *agnomen*

Alias:⟨*Alternate*⟩
Replaces ⟨*FNN*⟩ only in the text, not in the index.

"Native" Eastern Names: `\Name{`⟨*SNN,Affix*⟩`}[`⟨*Alternate*⟩`]`

Family:⟨*SNN*⟩	Personal:⟨*Affix*⟩	Alias:⟨*Alternate*⟩
Family / clan name	Usually one name	Replaces ⟨*Affix*⟩ only in text.[8]

Ancient / Royal Names: `\Name{`⟨*SNN,Affix*⟩`}[`⟨*Alternate*⟩`]`

Personal:⟨*SNN*⟩	Qualifier:⟨*Affix*⟩	Alias:⟨*Alternate*⟩
Given name(s)	Sobriquet / title: *Sr., Jr., III. . .* *notable attribute* *origin, region* *patronym*	Replaces ⟨*Affix*⟩ only in text.[9]

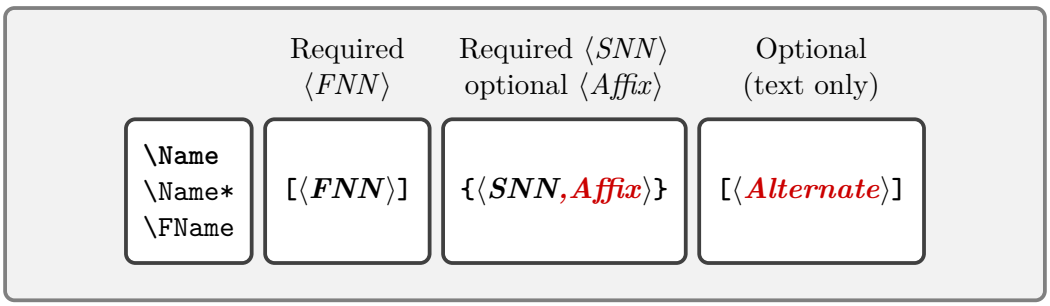
[6] Adopted from [Mulvany, 152–82] and [Chicago].
[7] There are several ways of handling the Roman *tria nomina*. See Section 5.10.
[8] The obsolete syntax uses ⟨*Alternate*⟩ instead of ⟨*Affix*⟩ for a personal name (Section 12.2).
[9] The obsolete syntax uses ⟨*Alternate*⟩ instead of ⟨*Affix*⟩ for a qualifier (Section 12.2).

1.4 Basic Interface

The description of macro arguments in this section applies to all nameauth macros that take name arguments (Sections 1.3.2, 1.6.1, and 4.3.1), making this section critical to using and mastering nameauth. Here are some general tips:

- If the required argument ⟨*SNN*⟩ is empty, nameauth issues a package error, even when the ⟨*Affix*⟩ part of an ⟨*SNN*⟩,⟨*Affix*⟩ pair is not empty. Macros that take null arguments (Section 6.1) are the exception.
- Extra spaces around each argument are stripped.
- Include name arguments consistently to have consistent index entries.

1.4.1 Western Names



- Western names must use the ⟨*FNN*⟩ argument.
- They require a comma to delimit any affixes (Section 4.3.2).
- They use ⟨*Alternate*⟩ to swap with ⟨*FNN*⟩ in the text.
- They have Western name patterns (Section 6.2) and index entries:

  ⟨*SNN*⟩, ⟨*FNN*⟩
  ⟨*SNN*⟩, ⟨*FNN*⟩, ⟨*Affix*⟩

Full, Last, and First Names

\Name prints first uses of names long, then short thereafter. \Name* ensures a long form. Both \FName and \FName* print long names in first uses, then just a forename in later uses. The intent is that one can just add an F to either \Name or \Name*. The affix in a surname only appears in long name instances.

Name Pattern(s):
George!Washington
GeorgeS.!Patton,Jr.
Basic Index:
Washington, George
Patton, George S., Jr.

First use: George Washington ..... \Name [George]{Washington}
Long use: George Washington ..... \Name*[George]{Washington}
Later use: Washington ..... \Name [George]{Washington}
Short use: George ..... \FName[George]{Washington}

First use: George S. Patton Jr. ..... \Name [George S.]{Patton, Jr.}
Long use: George S. Patton Jr. ..... \Name*[George S.]{Patton, Jr.}
Later use: Patton ..... \Name [George S.]{Patton, Jr.}
Short use: George S. ..... \FName[George S.]{Patton, Jr.}

Affixes and Alternate Forms

Here we use prefix macros to change name forms (Section 1.6.2). In a long name instance, `\DropAffix` drops the affix from a Western surname. The ⟨*Alternate*⟩ argument appears only in long names or forename-only names in the text. Otherwise, the automatic shortening of names will display only a short surname.

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

- Drop the affix in a first- or long use (`\Name*`):

  First use: George S. Patton
  `\DropAffix\Name[George S.]{Patton, Jr.}`
  Long use: George S. Patton
  `\DropAffix\Name*[George S.]{Patton, Jr.}`

- To force the printing of only ⟨*Affix*⟩ in the subsequent use of a Roman name, see the use of `\ForceAffix` in Section 5.10.1.

- Use an alternate forename in a first- or long use:

  First use: John Davison Rockefeller IV
  `\Name[J.D.]{Rockefeller, IV}[John Davison]`
  Long use: John Davison Rockefeller IV
  `\Name*[J.D.]{Rockefeller, IV}[John Davison]`

- Use an alternate forename in a forename-only use (`\FName`):

  Short use: George . . `\FName[George S.]{Patton, Jr.}[George]`
  Short use: Jay . . . . . `\FName[J.D.]{Rockefeller, IV}[Jay]`

- Drop the affix and alter the forenames:

  First use: Jay Rockefeller
  `\DropAffix\Name[J.D.]{Rockefeller, IV}[Jay]`
  Long use: Jay Rockefeller
  `\DropAffix\Name*[J.D.]{Rockefeller, IV}[Jay]`

- Use multiple alternate forenames for the same person:

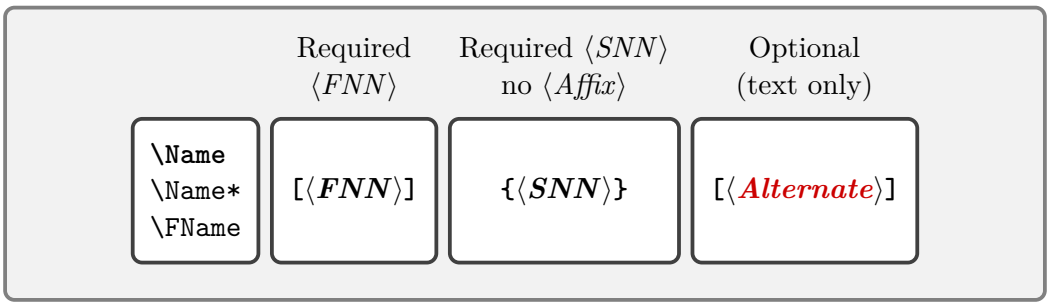
  First use: Clive Staples Lewis . . `\Name [Clive Staples]{Lewis}`
  Long use: C.S. Lewis . . . `\Name*[Clive Staples]{Lewis}[C.S.]`
  Short use: Jack . . . . . `\FName[Clive Staples]{Lewis}[Jack]`

All names should be sorted in the index by their longest unique forms. Convert Roman numerals to Arabic in order to avoid sorting the numbers alphabetically. See Sections 7.6, 7.6.2, and 10.1.
`\PretagName[J.D.]{Rockefeller, IV}{Rockefeller, John D 4}`

I don't think the mystical experience can be verbalized. When the ego disappears, so does power over language.

—W.H. Auden
Paris Review interview (1972) p. 206

1.4.2 Reversed Western Names



- Names must use the ⟨*FNN*⟩ argument. They do not use ⟨*Affix*⟩.[10]
- They use ⟨*Alternate*⟩ to swap with ⟨*FNN*⟩ in the text.
- They have Western name patterns and index entries:

  ⟨*SNN*⟩, ⟨*FNN*⟩

- They do not work with the obsolete syntax (Section 12.2).

Starting with Western Name Forms

Reversed Western forms (as with Hungarian and other names) start out as Western names before we put them into reverse order. They retain Western index entries [Mulvany, 166]. Without being reversed, they look like:[11]

Name Pattern(s):
`Ferenc!MolnÃąr`
`Hideyo!Noguchi`
Basic Index:
Noguchi, Hideyo
Molnár, Ferenc

First use: Ferenc Molnár ..... `\Name[Ferenc]{Molnár}`
Later use: Molnár ..... `\Name[Ferenc]{Molnár}`
First use: Hideyo Noguchi ..... `\Name[Hideyo]{Noguchi}`
Later use: Noguchi ..... `\Name[Hideyo]{Noguchi}`
Long use: Doctor Noguchi ..... `\Name*[Hideyo]{Noguchi}[Doctor]`

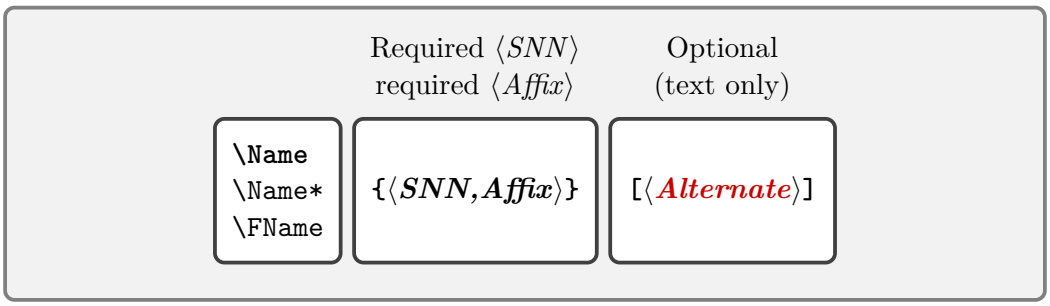
Using the Reversing and Other Macros

We use `\RevName` and optionally `\CapName` (Section 5.4) to print reversed names in the text without changing the Western forms of their index entries. They function well in context, not arbitrarily:

First use: Molnár Ferenc†
`\RevName\Name[Ferenc]{Molnár}\dag`
Later use: Molnár†
`\RevName\Name[Ferenc]{Molnár}\dag`
First use: NOGUCHI Sensei†
`\CapName\RevName\Name[Hideyo]{Noguchi}[Sensei]\dag`
Later use: NOGUCHI†
`\CapName\RevName\Name[Hideyo]{Noguchi}[Sensei]\dag`
Long use: NOGUCHI Sensei†
`\CapName\RevName\Name*[Hideyo]{Noguchi}[Sensei]\dag`

[10] This prevents strange name forms. Using `\DropAffix` (Section 4.3.2) will not help.
[11] Regarding the margin note that shows name patterns, with `pdflatex` and `latex` using `inputenc` and `fontenc`, in `Ferenc!MolnÃąr` the glyphs `Ãą` correspond to `\IeC{\'a}`.

1.4.3 "Native" Eastern Names



- They must **leave empty** the ⟨*FNN*⟩ argument.
- They use instead the ⟨*SNN*,*Affix*⟩ arguments.
- They can swap ⟨*Alternate*⟩ and ⟨*Affix*⟩ in the text.[12]
- They have Non-Western name patterns and index entries:

  ⟨*SNN*⟩ ⟨*Affix*⟩

- If name order is reversed in the text with `\RevName`, these names retain Non-Western index entries.

Starting with Eastern Name Forms

Among the names shown below, `\FName` does not show a personal name by default. This design helps to prevent Western writers from being culturally insensitive.

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

First use: Miyazaki Hayao ..... `\Name{Miyazaki, Hayao}`
Later use: Miyazaki ..... `\Name{Miyazaki, Hayao}`
Long use: Miyazaki Sensei ..... `\Name*{Miyazaki, Hayao}[Sensei]`
Short use: Miyazaki ..... `\FName{Miyazaki, Hayao}`

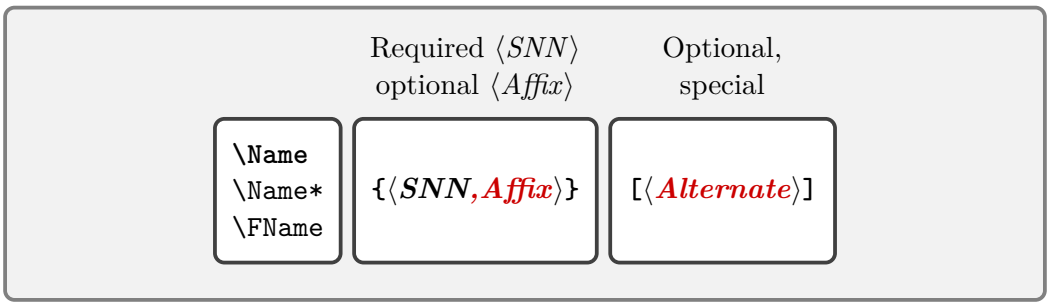
Using the Reversing and Other Macros

The reversing macros produce a Western name ordering in the text. One must use `\ForceFN` with `\FName` (Section 4.2) to print just a personal name:

First use: Hayao Miyazaki
`\RevName\Name{Miyazaki, Hayao}`
Short use: Hayao
`\ForceFN\FName{Miyazaki, Hayao}`
Long use: Mr. Miyazaki
`\RevName\Name*{Miyazaki, Hayao}[Mr.]`
Long use: MIYAZAKI Sensei
`\CapName\Name*{Miyazaki, Hayao}[Sensei]`
Short use: Sensei
`\ForceFN\FName{Miyazaki, Hayao}[Sensei]`

[12] We discourage the obsolete syntax (Section 12.2) because ⟨*Alternate*⟩ does not work with it.

1.4.4 Royal or Ancient Names



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

  ⟨*SNN*⟩ ⟨*Affix*⟩
  ⟨*SNN*⟩

- Names with ⟨*SNN,Affix*⟩ can swap ⟨*Alternate*⟩ and ⟨*Affix*⟩ in the text.
- Names with ⟨*SNN*⟩ should not use ⟨*Alternate*⟩.[13]

No School Like the Old School

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

Basic Index:
Elizabeth I
John Eriugena
Aristotle

- `\FName` normally prints ⟨*SNN*⟩ to avoid mistakes.

  First use: Elizabeth I ..... `\Name{Elizabeth, I}`
  Later use: Elizabeth ..... `\Name{Elizabeth, I}`
  Short use: Elizabeth ..... `\FName{Elizabeth, I}`

- Here we work with titles and sobriquets:

  First use: Elizabeth I "Gloriana"
  `\Name{Elizabeth, I}[I ''Gloriana'']`
  Short use: Gloriana
  `\ForceFN\FName{Elizabeth, I}[Gloriana]`

- In long name references, a space always occurs between ⟨*SNN*⟩ and ⟨*Affix*⟩ or ⟨*SNN*⟩ and ⟨*Alternate*⟩. When a sobriquet begins with a comma, semicolon, etc., use name tags (Sections 1.7 and 8):

  – Avoid forms like the following:
  Attila , the Scourge of God
  `\Name{Attila, the Hun}[, the Scourge of God]`
  – Use instead something like:
  Attila the Hun, the Scourge of God
  Attila, the Scourge of God
  `\NameAddInfo{Attila, the Hun}{, the Scourge of God}`
  `\ForgetThis\Name{Attila, the Hun}\NameQueryInfo{}`
  `\Name{Attila, the Hun}\NameQueryInfo{}`

[13] That would produce the obsolete syntax (Section 12.2).

- Here we show a non-royal, medieval name:

  First use: John Scotus Eriugena
  `\Name{John, Eriugena}[Scotus Eriugena]`
  Long use: John Eriugena ..... `\Name*{John, Eriugena}`
  Later use: John ..... `\Name{John, Eriugena}`

- Context matters; do not reverse names or force affixes arbitrarily:

  Short use: I ..... `\ForceFN\FName{Elizabeth, I}`
  Long use: Eriugena John ..... `\RevName\Name*{John, Eriugena}`

- Here is the trivial case:

  First use: Aristotle ..... `\Name{Aristotle}`
  Later use: Aristotle ..... `\Name{Aristotle}`

1.4.5 More about Selecting Name Types

How does one decide from among these name forms? Some choices are fairly obvious and standardized. Others depend on one's culture and publication standards.

- Dealing with patronyms can vary according to one's culture.

  Name Pattern(s):
  `Leif!Erikson`
  `Leif,Erikson`

  – To index under the patronym, use a Western form:

    Leif Erikson ..... `\Name[Leif]{Erikson}`
    Erikson ..... `\Name[Leif]{Erikson}`
    Leif ..... `\FName[Leif]{Erikson}`
    Index: Erikson, Leif

  – To index under the personal name, use a Non-Western form:

    Leif Erikson ..... `\Name{Leif, Erikson}`
    Leif ..... `\Name{Leif, Erikson}`
    Erikson ..... `\ForceFN\FName{Leif, Erikson}`
    Index: Leif Erikson

- Roman names are quite challenging (Sections 5.10f.).

  Name Pattern(s):
  `M.T.!Cicero`
  `Pontius!Pilate`
  `Pontius,Pilate`

  – Common Western usage draws on popular name forms:

    M.T. Cicero ..... `\Name[M.T.]{Cicero}`
    Cicero ..... `\Name[M.T.]{Cicero}`
    Index: Cicero, M.T.

  – That segues into names indexed by *cognomen*:

    Pontius Pilate ..... `\Name[Pontius]{Pilate}`
    Pilate ..... `\Name[Pontius]{Pilate}`
    Index: Pilate, Pontius

  – Yet other approaches can index by *nomen*:

    Pontius Pilate ..... `\Name{Pontius,Pilate}`
    Pontius ..... `\Name{Pontius,Pilate}`
    Pilate ..... `\ForceFN\FName{Pontius,Pilate}`
    Index: Pontius Pilate

There are ways to display a name using one form, while indexing under another, starting in Section 5.8. The quick interface discussed in the next section can make these choices easier to implement and change, if needed. One should consult a style guide or in-house guidelines to help choose the proper name forms.

1.5 Quick Interface

1.5.1 Name Shorthands

`nameauth` (*env.*)

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}
```

In this context, ⟨*arg1*⟩ becomes the root of 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
```

Usually we leave ⟨*arg4*⟩ empty, apart from specific contexts. That field permanently displays only alternate names, or is used with the obsolete syntax (Section 12.2). Here is another way of thinking about arguments in the `nameauth` environment that relates back to what we have seen with the basic interface:

```
\begin{nameauth}
    \< ⟨arg1⟩ & ⟨FNN⟩ & ⟨SNN,Affix⟩ & ⟨Alternate⟩ >
\end{nameauth}
```

By seeing the mandatory arguments in black and the optional ones in red, it helps us to see that, if either ⟨*arg1*⟩ or ⟨*arg3*⟩ are empty, or ⟨*SNN*⟩ is empty, nameauth will generate a package error. Forgetting the backslash, any ampersand, or angle bracket will cause fatal errors. See Section 4.3.1 on final optional arguments.

Package warnings result when one redefines name shorthands using the `nameauth` environment. For example, we use `White` in two different rows to populate ⟨*arg1*⟩. That causes `\White`, `\LWhite`, and `\SWhite` to be redefined:

```
1 \begin{nameauth}
2   \< White & E.B.   & White & >        % version 1
3   \< White & E.\,B. & White & >        % version 2
4 \end{nameauth}
```

Name Pattern(s):
`E.\,B.!White`
Basic Index:
White, E. B.

`\White` produces E. B. White, the version with the thin space. We lost the first version of the name when we redefined it.[14]

On the next page we will create an example `nameauth` environment using many of the names that we have so far encountered. We will add other names that we have not yet seen, introducing additional concepts in the process. Those include Western name forms that contain particles (articles and prepositions), which are discussed in greater detail in Section 5.7.

[14] When building nameauth an intended warning will appear: `Shorthand macro already exists.`

The comments below are merely explanatory and in no wise required to use the environment. Likewise, extra spaces that are added for clarity are stripped.

```
\begin{nameauth}
% Western Name Forms
%    <arg1>     <arg2>          <arg3>              <arg4>
  \< Wash     & George        & Washington        &        >
  \< Lewis    & Clive Staples & Lewis             &        >
% Western Name Forms with Affixes
  \< Patton   & George S.     & Patton, Jr.       &        >
  \< JRIV     & J.D.          & Rockefeller, IV   &        >
% Western Name Forms with Particles
  \< Soto     & Hernando      & de Soto           &        >
  \< JLF      & Jean de       & La Fontaine       &        >
  \< JWG      & J.W. von      & Goethe            &        >
% Reversed Western Forms
  \< Noguchi  & Hideyo        & Noguchi           &        >
  \< Molnar   & Ferenc        & Molnár            &        >
% ‘‘Native’’ Eastern Forms
  \< Miyazaki &               & Miyazaki, Hayao   &        >
%  Royal, Medieval, and Ancient Forms
  \< Eliz     &               & Elizabeth, I      &        >
  \< Aeth     &               & Æthelred, II      &        >
  \< Eriugena &               & John, Eriugena    &        >
  \< Aris     &               & Aristotle         &        >
% Name Forms Always Using Alternate Names
  \< CSL      & Clive Staples & Lewis             & C.S.   >
  \< MSens    &               & Miyazaki, Hayao   & Sensei >
\end{nameauth}
```

Here is an example of how much typing one can save with the quick interface, not to mention the prevention of error by not retyping arguments:

Name Pattern(s):
`George!Washington`
Basic Index:
Washington, George

Text		Macro / Arguments
Washington	Quick:	`\Wash`
	Basic:	`\Name[George]{Washington}`
George Washington	Quick:	`\LWash`
	Basic:	`\Name*[George]{Washington}`
George	Quick:	`\SWash`
	Basic:	`\FName[George]{Washington}`
George Washington	Quick:	`\ForgetThis\Wash`
	Basic:	`\ForgetName[George]{Washington}` `\Name[George]{Washington}` — **OR** —
	Basic:	`\ForgetThis\Name[George]{Washington}`
Washington	Quick:	`\SubvertThis\Wash`
	Basic:	`\SubvertName[George]{Washington}` `\Name[George]{Washington}` — **OR** —
	Basic:	`\SubvertThis\Name[George]{Washington}`
(unseen in text)	Quick:	`\JustIndex\Wash`
	Basic:	`\IndexName[George]{Washington}`

1.5.2 Quick Name Variant Overview

After setting up the previous `nameauth` environment, we use the resulting name shorthands. Below we show more prefix macros that affect name forms in the text (Section 1.6.2). We hide the use of `\ForgetThis` (Section 9.3), which makes the nameauth logic "forget" that a name exists, creating a first-use instance.

Name Pattern(s):
`George!Washington`
Basic Index:
Washington, George

- Western Names: Sections 4.1, 4.2

First use: George Washington ..... `\Wash`
Long use: George Washington ..... `\LWash`
Later use: Washington ..... `\Wash`
Short use: George ..... `\SWash`

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

- Nicknames and Affixes: Sections 4.2, 4.3.2

First use: George S. Patton ..... `\DropAffix\Patton`
Long use: George S. Patton Jr. ..... `\LPatton`
Long use: George S. Patton ..... `\DropAffix\LPatton`
Long use: George Patton ..... `\DropAffix\LPatton[George]`
Later use: Patton ..... `\Patton`
Short use: George S. ..... `\SPatton`
Short use: George ..... `\SPatton[George]`

First use: John Davison Rockefeller IV ..... `\JRIV[John Davison]`
Long use: J.D. Rockefeller IV ..... `\LJRIV`
Long use: Jay Rockefeller ..... `\DropAffix\LJRIV[Jay]`
Later use: Rockefeller ..... `\JRIV`

First use: Clive Staples Lewis ..... `\Lewis`
Long use: C.S. Lewis ..... `\LLewis[C.S.]`
Long use: Jack Lewis ..... `\LLewis[Jack]`
Short use: Clive Staples ..... `\SLewis`
Short use: Jack ..... `\SLewis[Jack]`
Long use: C.S. Lewis ..... `\LCSL`
Short use: C.S. ..... `\SCSL`

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

- Non-Western ("Native" Eastern) Names: Sections 5.3, 5.4, 5.5

First use: MIYAZAKI Hayao ..... `\CapName\Miyazaki`
Long use: MIYAZAKI Hayao ..... `\CapName\LMiyazaki`
Later use: MIYAZAKI ..... `\CapName\Miyazaki`
Long use: Hayao Miyazaki ..... `\RevName\LMiyazaki`
Long use: Mr. Miyazaki ..... `\RevName\LMiyazaki[Mr.]`
Short use: Miyazaki ..... `\SMiyazaki`
Short use: Hayao ..... `\ForceFN\SMiyazaki`

Name Pattern(s):
`Hideyo!Noguchi`
Basic Index:
Noguchi, Hideyo

- Reversed Western Names: Sections 5.4, 5.5

First use: Hideyo Noguchi ..... `\Noguchi`
Long use: Hideyo Noguchi ..... `\LNoguchi`
Long use: Doctor Noguchi ..... `\LNoguchi[Doctor]`
Short use: Hideyo ..... `\SNoguchi`
Long use: Noguchi Hideyo† ..... `\RevName\LNoguchi\dag`
Long use: NOGUCHI Hideyo† . `\CapName\RevName\LNoguchi\dag`
Later use: NOGUCHI† ..... `\CapName\Noguchi\dag`

Name Pattern(s):
`George!Washington`
Basic Index:
Washington, George

- Western Names Reversed by Surname: Section 5.6

  First use: Washington, George ..... `\RevComma\Wash`
  Long use: Washington, George ..... `\RevComma\LWash`

Name Pattern(s):
`Hernando!de~Soto`
Basic Index:
de Soto, Hernando

- Particles, English usage: Section 5.7

  First use: Hernando de Soto ..... `\Soto`
  Later use: De Soto ..... `\CapThis\Soto`

Name Pattern(s):
`Jeande!LaFontaine`
`J.W.von!Goethe`
Basic Index:
La Fontaine, Jean de
Goethe, J.W. von

- Particles, Non-English usage: Section 5.7

  First use: Jean de La Fontaine ..... `\JLF`
  Later use: La Fontaine ..... `\JLF`
  First use: Joh. Wolfgang v. Goethe .... `\JWG[Joh. Wolfgang v.]`
  Later use: Goethe ..... `\JWG`

Name Pattern(s):
`ÃĘthelred,II`
`John,Eriugena`
`Aristotle`
Basic Index:
Æthelred II
John Eriugena
Aristotle

- Royal, Medieval, and Ancient Names: Sections 5.8, 5.9

  First use: Æthelred II[15] ..... `\Aeth`
  Later use: Æthelred ..... `\Aeth`
  Long use: Æthelred II, "Unræd" ..... `\LAeth[II, ''Unræd'']`

  First use: John Scotus Eriugena .. `\Eriugena[Scotus Eriugena]`
  Long use: John Eriugena ..... `\LEriugena`
  Later use: John ..... `\Eriugena`

  First use: Aristotle ..... `\Aris`
  Later use: Aristotle ..... `\Aris`

1.5.3 ⟨*Alternate*⟩ Name Field

Two shorthands above use ⟨*arg4*⟩, the final field in each row of the `nameauth` environment. These are `\CSL` and `\MSens`. They correspond to similar name shorthands `\Lewis` and `\Miyazaki`, which leave ⟨*arg4*⟩ empty. Here is how they are related:

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

- They share identical name patterns (Section 6.2).

  First use: C.S. Lewis ..... `\CSL`
  Later use: Lewis ..... `\Lewis`
  First use: Miyazaki Sensei ..... `\MSens`
  Later use: Miyazaki ..... `\Miyazaki`

- Usually, one leaves ⟨*arg4*⟩ empty and adds alternate names in brackets as needed: C.S. Lewis `\LLewis[C.S.]`
- By using ⟨*arg4*⟩, one trades less work for more ambiguity: Can one add an alternate name or not? To answer that, one should keep track of all name shorthands that use ⟨*arg4*⟩.
- Failure to keep track of such macros creates output like C.S. Lewis[Jack] `\LCSL[Jack]` and Miyazaki Sensei[Sensei] `\LMSens[Sensei]`. The ⟨*arg4*⟩ content populates ⟨*Alternate*⟩; the text in brackets is just text.
- Remember that ⟨*arg4*⟩ can be used for the obsolete syntax, as mentioned previously, but we do not cover that here.

[15] Regarding the margin note with name patterns, with `pdflatex` and `latex` using inputenc and fontenc, in `ÃĘthelred,II` the glyphs `ÃĘ` correspond to `\IeC{\AE}`.

1.6 Select Macro Overview

1.6.1 Macros with Name Arguments

Macros that take name arguments (Section 1.3.2) can have final optional arguments (Section 4.3.1) and update `\NameauthPattern` (Section 6.2). The ⟨*xref args*⟩ are the same as ⟨*name args*⟩ for a cross-reference to a ⟨*target*⟩ (Section 7.3). For more on null arguments, see Section 6.1.

	Optional Prefix	Macro	Star	Arguments	
Print a name	⟨*prefix macros*⟩	`\Name`	*	⟨*name args*⟩	
	⟨*prefix macros*⟩	`\FName`	*	⟨*name args*⟩	
Page entry	only `\SeeAlso`	`\IndexName`		⟨*name args*⟩	
Only cross-reference	only `\SeeAlso`	`\IndexRef`		⟨*xref args*⟩	⟨*target*⟩
Stop page entry		`\ExcludeName`		⟨*name args*⟩	
Allow page entry		`\IncludeName`	*	⟨*name args*⟩	
Sort index		`\PretagName`		⟨*name args*⟩	⟨*sort key*⟩
Make index tag		`\TagName`		⟨*name args*⟩	⟨*tag*⟩
Delete index tag		`\UntagName`		⟨*name args*⟩	
Make name tag		`\NameAddInfo`		⟨*name args*⟩	⟨*tag*⟩
Show name tag		`\NameQueryInfo`		⟨*name args*⟩	or null argument
Delete name tag		`\NameClearInfo`		⟨*name args*⟩	
Delete name pattern		`\ForgetName`		⟨*name args*⟩	
Create name pattern		`\SubvertName`		⟨*name args*⟩	
Name pattern tests		`\IfMainName`		⟨*name args*⟩	`{`⟨*y*⟩`}{`⟨*n*⟩`}`
		`\IfFrontName`		⟨*name args*⟩	`{`⟨*y*⟩`}{`⟨*n*⟩`}`
		`\IfAKA`		⟨*name args*⟩	`{`⟨*y*⟩`}{`⟨*n*⟩`}{`⟨*x*⟩`}`
Debugging		`\ShowPattern`		⟨*name args*⟩	or null argument
		`\ShowIdxPageref`	*	⟨*name args*⟩	or null argument
		`\ShowNameInfo`		⟨*name args*⟩	or null argument
		`\ShowNameState`	*	⟨*name args*⟩	or null argument

Regarding prefix macros, see Section 1.6.2. Not shown above are `\AKA`, `\AKA*`, `\PName`, and `\PName*` (Section 12.1). They have their own rules.

We depict hatred, but it is to depict that there are more important things. We depict a curse, to depict the joy of liberation.

—Miyazaki Hayao
Proposal for *Princess Mononoke* (c. 1994)

1.6.2 Prefix Macros

Similar to the package options (Section 2), many prefix macros alter the values of the Boolean flags that reflect the state of names and name processing. The naming and indexing macros reset the Boolean flags after they are invoked.

- Capitalization in the text:

`\CapName`	Capitalize entire ⟨*SNN*⟩. Overrides `\CapThis`.
`\CapThis`	Capitalize first letter of all name components.
`\AccentCapThis`	Fallback when Unicode detection cannot be done.

- Reversing names in the text:

`\RevName`	Reverse order of any name. Overrides `\RevComma`
`\RevComma`	Reverse only Western names to ⟨*SNN*⟩, ⟨*FNN*⟩.

- Use of commas before Western affixes 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 instance.
`\KeepAffix`	Insert non-breaking space between ⟨*SNN*⟩, ⟨*FNN/Affix*⟩.
`\KeepName`	Insert NBSP between all name elements. Overrides `\KeepAffix`.

- Forcing name forms in the text by deleting or creating patterns:

`\ForgetThis`	Force first name instance. Negates `\SubvertThis`.
`\SubvertThis`	Force subsequent name instance.

- Forcing other name forms:

`\ForceName`	Force first-use formatting hooks.
`\ForceAffix`	Print only the affix of a Western name (Section 5.10.1).
`\ForceFN`	Print ⟨*Affix*⟩ or ⟨*Alternate*⟩ in a Non-Western short name.

- Indexing:

`\SeeAlso`	For `\IndexName`, `\AKA`, and `\PName`; make a *see also* xref.
`\SkipIndex`	For naming macros; do not create index entry.
`\JustIndex`	For naming macros; index only; negated by `\AKA`, `\PName`.

- Prefix macros can "stack" their effects:
  Foo Bar . . . . . `\CapThis\RevName\SkipIndex\Name[bar]{foo}`
- 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 via `\JustIndex`.
- Even after using `\JustIndex`, several name form modifiers are reset. This prevents errors when handling the next name.
- Except for `\SeeAlso`, use prefix macros only before a naming macro that is designed to print output in the text.
- Use `\SeeAlso` only with `\IndexRef`, `\AKA`, and `\PName`. Otherwise it will be ignored and reset by `\IndexName` and the naming macros.

1.7 Names and Complexity

The nameauth package allows levels of complexity when representing names. Previously, we have seen this example:

Elizabeth I "Gloriana" ..... `\LEliz[I ‘‘Gloriana’’]`

We can display the same name form using different macros. This next example does not offer more features; it only presents more complexity:

Elizabeth I "Gloriana" ..... `\LEliz\ ‘‘\ForceFN\SEliz[Gloriana]’’`

The next example offers more features and better automation. Based on Section 5.9 (cf. 9.1), we use name tags and adapt the formatting hooks to this manual's conventions. We make key changes once, which govern names thereafter:

```
1 \makeatletter
2 \NameAddInfo{Elizabeth, I}{\if@nameauth@InHook{ }\fi‘‘Gloriana’’}
3 \makeatother
4 \renewcommand*\NamesFormat[1]
5   {\color{blue}\sffamily #1\NameQueryInfo{}}
6 \renewcommand*\MainNameHook{\sffamily}
```

Name Pattern(s):
`Elizabeth,I`
Basic Index:
Elizabeth I

First use: Elizabeth I "Gloriana" ..... `\ForgetThis\LEliz`
Long use: Elizabeth I ..... `\LEliz`
Later use: Elizabeth ..... `\Eliz`
"Gloriana" ..... `\NameQueryInfo{}`

Avoid the Rabbit Hole

- Names are nouns with changeable states.
  - ⟨*FNN*⟩ and ⟨*Alternate*⟩ can change easily in the text.
  - Changing ⟨*Affix*⟩ depends on the name type.
  - Changing ⟨*SNN*⟩ needs `\noexpand` with a macro that changes only in the formatting hooks (Section 11.2).
- Names also are verbs that affect their environment.
- There are design trade-offs:
  - Simple examples are easy to write, but they can require subtle, ongoing changes of repetitive macros.
  - Automation shifts the work to a complex, one-time setup that hides complexity thereafter.
- Use a simple approach unless otherwise needed.

Back to Table of Contents

2 Package Options

The nameauth options range from standards compliance, control, and appearance to backward compatibility with old versions. Thus:

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

We discuss package options according to the structure of this package. Section 3 shows the priority of options and related macros. Default options are in **boldface** and need not be invoked. Other options are in **dark red** and must be invoked explicitly. Many options work with macros offering finer control.

2.1 Name Grammar and Syntax

Syntax Warnings

`strictsyntax`	Show warnings when the obsolete syntax appears.

Disabled by default, this option helps one to avoid using the obsolete syntax.

Show/Hide Affix Commas

`nocomma`	**Modern standards: Suppress commas between Western surnames and affixes.**
`comma`	Older standards: Retain commas between Western surnames and affixes.

Name Pattern(s):
`J.E.!Carter,Jr.`
Basic Index:
Carter, J.E., Jr.

These options do not affect the index. The default `nocomma` option lets us produce, e.g., J.E. Carter Jr. (Jimmy). The `comma` option produces J.E. Carter, Jr. For equivalent macros, see Section 4.3.2.

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 5.3 for finer control. To capitalize names in the index, use caps as desired or alternate formatting (Section 11.2).

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; Western as Non-Western, and vice versa.
`allrevcomma`	Print all Western names as "Surname, Forenames".

These options do not affect the index and are mutually exclusive (Sections 5.5 and 5.6). Use `allrevcomma` option only for listing Western names by surname.

2.2 Indexing

Toggle 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.

Using `noindex` and `\IndexInactive` prevents index tags until `\IndexActive` is called, as explained in Section 7.1. For indexing feature priority, see Section 3.

Toggle 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. See Section 7.6.

Verbose 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. Section 7.1 shows macros that can enable and disable verbose warnings.

2.3 Formatting and Name Patterns

Formatting, in its simplest form, is typographic post-processing of a name in the body text. In its complex forms, formatting can affect the syntactic form of a name in both the body text and the index.

Choose Formatting System

`mainmatter`	**Start with "main-matter names" and formatting hooks.** (Section 9.1)
`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. Even when no extra formatting occurs, the formatting hooks are defined. Changes require `\renewcommand`.

The `alwaysformat` option forces "first use" hooks globally in both name systems. It is of limited usefulness in current versions of `nameauth`.

The `formatAKA` option permits `\AKA` to use the "first use" formatting hooks. This enables `\ForceName` to trigger those hooks at will (Section 12.1). Otherwise `\AKA` only uses "subsequent use" formatting hooks.

Predefined Formatting Hooks

`noformat`	**Pass the displayed name through the formatting hooks unchanged.**
`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. The default is no formatting.[16] See also Robert Bringhurst, *The Elements of Typographic Style*, version 3.2 (Point Roberts, Washington: Hartley & Marks, 2008), 53–60. All references [Bringhurst] refer to this edition.

The following macros govern the way that names in the text appear. Two name systems are used in nameauth, one for main-matter text (default) and one for front-matter text.[17] These hooks do not affect the index. Changes to the formatting hooks normally apply within the scope where they occur:

- `\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.

Sections 9.1 and 11f. explain these hooks and their redefinition in greater detail. Section 12.1 discusses how `\AKA` does not respect these formatting systems.

2.4 Alternate Formatting

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

A built-in framework provides an alternate formatting mechanism that can be used for European or "Continental" standads in German, French, and so on. These standards often format surnames, both in the text and in the index. See Section 11.2.

2.5 Change Scope of Name Decision Macros

`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 9.5). This option removes that scoping. Note that names tracked by nameauth within such decision paths are global with respect to their values.

[16] For the old default, use the `smallcaps` option. User feedback dictated this change.

[17] `\NamesFormat` was once the only formatting hook. The other macros developed from there. Regrettably, this approach contributed to some "cargo-cult" programming, hopefully remedied.

2.6 Version Compatibility

These options can introduce undocumented behavior. They exist only for backward compatibility to help users cope with this author's ignorance as development changed to better fit the problem domain.

`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 or short Boolean flags to pass through, as they did 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.
`oldargs`	Load the xargs and suffix packages in order to permit user-supplied modifications to work as in version 3.7 and before. The package macros will still use xparse because of its advantages.

The following table shows how one gets approximate backward compatibility:

Version	Needed Options	Possible Options
≤ 2.6	`oldreset,oldtoks,oldsee, oldAKA,oldpass`	`oldargs`
3.0–3.2	`oldreset,oldtoks,oldsee, oldpass`	`oldargs`
3.3–3.4	`oldreset,oldtoks,oldsee`	`oldargs`
3.5–3.7		`oldargs`

Back to Table of Contents

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

(A Christian person is a free lord, above all things, and subject to no one. A Christian person is a willing servant of all things and is subject to everyone.)

—Martin Luther
Von der Freiheit eines Christenmenschen (1520)

3 Feature Priority

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

Above we see the relative priority of package options and their related macros. Package options are shown in boldface.

- Lighter-colored rows show higher priority. Darker-colored rows show lower priority. Higher-priority options and macros have the ability to override lower-priority ones.
- All options and macros in a given row have equal priority and are able to countermand each other within a given column.
- Priority affects macros and options within columns. `\IndexInactive` overrides `\JustIndex`, which overrides `\SkipIndex`. If `\IndexInactive` is invoked, `\JustIndex` will have no effect.
- Section 7.4 shows the complex interaction between `\SkipIndex` and `\JustIndex`. It it is best to use `\SkipIndex` and `\JustIndex` only before a naming macro that can print to the text.
- Priority usually does not affect macros and options in different columns. Yet the macros themselves can have specific effects that change the expected behavior of macros in other columns.

  For example, `\JustIndex` prevents a name from being displayed in the text. Even if `\IndexInactive` overrides `\JustIndex` with respect to indexing, it has no effect on the fact that the name will not be printed.

  Also, `\JustIndex` resets the effects of `\ForgetThis` and `\SubvertThis` because those prefix macros (Section 1.6.2) should precede only naming macros that produce output in the text.

  Although `\JustIndex` does not override the caps and reversing macros and options, it prevents anything related to displaying a name.

Back to Table of Contents

4 Naming Macros

This section is a "pedantic" presentation of macros, their syntax, and their output. Section 1.4 is better for getting started. All naming macros that have the same arguments also create consistent index entries. These entries are created both at the start and at the end of a name, in case that name spans a page break.

4.1 \Name and \Name*

\Name
\Name*

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

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

In the body text only, 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 12.2, not shown below).

```
\begin{nameauth}
  \< Einstein  & Albert & Einstein        & >
  \< Carter    & J.E.   & Carter, Jr.     & >
  \< Miyazaki  &        & Miyazaki, Hayao & >
  \< Eliz      &        & Elizabeth, I    & >
  \< Confucius &        & Confucius       & >
\end{nameauth}
```

Name Pattern(s):
`Albert!Einstein`
`J.E.!Carter,Jr.`
`Miyazaki,Hayao`
`Elizabeth,I`
`Confucius`
Basic Index:
Einstein, Albert
Carter, J.E., Jr.
Miyazaki Hayao
Elizabeth I
Confucius

Albert Einstein	`\Name [Albert]{Einstein}` or `\Einstein`
Albert Einstein	`\Name*[Albert]{Einstein}` or `\LEinstein`
Einstein	`\Name [Albert]{Einstein}` or `\Einstein`
J.E. Carter Jr.	`\Name [J.E.]{Carter, Jr.}` or `\Carter`
James Earl Carter Jr.	`\Name*[J.E.]{Carter, Jr.}[James Earl]` or `\LCarter[James Earl]`
Carter	`\Name [J.E.]{Carter, Jr.}` or `\Carter`
Miyazaki Hayao	`\Name {Miyazaki, Hayao}` or `\Miyazaki`
Miyazaki Sensei	`\Name*{Miyazaki, Hayao}[Sensei]`
Miyazaki	`\Name {Miyazaki, Hayao}` or `\Miyazaki`
Elizabeth I	`\Name {Elizabeth, I}` or `\Eliz`
Elizabeth I	`\Name*{Elizabeth, I}` or `\LEliz`
Elizabeth	`\Name {Elizabeth, I}` or `\Eliz`
Confucius	`\Name {Confucius}` or `\Confucius`
Confucius	`\Name {Confucius}` or `\Confucius`

With the quick interface, the usual way to use alternate names is by appending them in brackets:

> James Earl Carter Jr. . . . . `\LCarter[James Earl]`
> Miyazaki Sensei . . . . . `\LMiyazaki[Sensei]`

Otherwise, one can populate the final field in the `nameauth` environment, with several caveats (Section 1.5.3). Alternate names appear only in long name forms or short name forms described in the next section.

4.2 Forenames: \FName

\FName
\FName*

\FName and its synonym \FName* print personal names only in subsequent name uses. That means when a name pattern does not exist, they print long name forms because it is a first use of a name.

Unlike all other starred forms of macros in nameauth, these macros are synonyms because one might edit either \Name or \Name* by adding an F to create a short name instead of the usual forms. This approach was implemented before the quick interface was developed.

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

These forename instance macros will permit all name types, but the normal behavior prints forenames only with Western names. With Non-Western names only ⟨*SNN*⟩ is printed. This is designed to prevent Western writers from causing unintended offense in Eastern contexts. It also prevents the display of nonsense forms in the context of ancient and royal names.

\ForceFN

To get an Eastern personal name or any affixed components of an ancient name, or to get ⟨*Alternate*⟩ to display in their place, one must precede these macros with \ForceFN. See also Section 5.8 for more uses of \ForceFN.

Name Pattern(s):
Albert!Einstein
J.E.!Carter,Jr.
Miyazaki,Hayao
Elizabeth,I
Confucius

Basic Index:
Einstein, Albert
Carter, J.E., Jr.
Miyazaki Hayao
Elizabeth I
Confucius

Albert	\FName[Albert]{Einstein} or \SEinstein
Jimmy	\FName[J.E.]{Carter, Jr.}[Jimmy] or \SCarter[Jimmy]
Miyazaki	\FName{Miyazaki, Hayao} or \SMiyazaki
Hayao	\ForceFN\FName{Miyazaki, Hayao} or \ForceFN\SMiyazaki
Sensei	\ForceFN\FName{Miyazaki, Hayao}[Sensei] or \ForceFN\SMiyazaki[Sensei]
Elizabeth	\FName{Elizabeth, I} or \SEliz
Gloriana	\ForceFN\FName{Elizabeth, I}[Gloriana] or \ForceFN\SEliz[Gloriana]
Confucius	\FName{Confucius} or \SConfucius

The ⟨*Alternate*⟩ argument replaces forenames in the text, which strongly shapes the use of \FName. We already have covered the use of ⟨*Arg4*⟩ of the nameauth environment in Section 1.5.3. Please refer to that material when using ⟨*Alternate*⟩, especially with the quick interface.

> As for me, I see no such great cause why I should either be fond to live or fear to die. I have had good experience of this world, and I know what it is to be a subject and what to be a sovereign. Good neighbors I have had, and I have met with bad; and in trust I have found treason.
>
> —Elizabeth I
> speech to Parliament, 1586

4.3 Technical Details

4.3.1 Final Optional Arguments

Macros that take name arguments using the current syntax (see also Section 1.6.1) can have a final optional argument. We do not index the names below in this manual, but we show how the index entries would appear:

- Western Index Entries:

Text	Macro / Arguments	Index
Alias Last Affix	`\Name*[First]{Last, Affix}[Alias]`	Last, First, Affix
Alias Last	`\Name*[First]{Last}[Alias]`	Last, First
Last Alias	`\RevName\Name*[First]{Last}[Alias]`	Last, First

- Non-Western Index Entries:

Text	Macro / Arguments	Index
Family Alias	`\Name*{Family, Person}[Alias]`	Family Person
Alias Family	`\RevName\Name*{Family, Person}[Alias]`	Family Person
Person Alias	`\Name*{Person, Affix}[Alias]`	Person Affix

Since May 2018, xparse either can ignore spaces before final optional arguments, or it can see those spaces as significant. Below `\Namei` ignores these spaces, the default for nameauth. `\Nameii` sees these spaces as significant.[18] The default advantages of `\Namei` outweigh those of `\Nameii`. Section 11.5.2 shows how one can change these defaults via advanced customization. There are no "perfect" choices, only trade-offs.

```
\NewDocumentCommand {\Namei}{O{} m  O{}}
  {\providecommand\Opt{#3}\ifx\Opt\empty #1\ #2\else \Opt\ #2\fi}
% Cannot use this definition before May 2018.
\NewDocumentCommand{\Nameii}{O{} m !O{}}
  {\providecommand\Opt{#3}\ifx\Opt\empty #1\ #2\else \Opt\ #2\fi}

\cmd{\Namei: } \Namei[Person1]{Family1} [something1]\\[1ex]
\cmd{\Nameii:} \Nameii[Person2]{Family2} [something2]
```

`\Namei:` something1 Family1

`\Nameii:` something1 Family2 [something2]

- What happens when we ignore these spaces (`\Namei`)?
  - Subsequent text in brackets always becomes a name argument.
  - We avoid errors arising from unintended spaces.
  - This follows standard LaTeX behavior.
- What happens when these spaces are significant (`\Nameii`)?
  - Spaces that follow the required argument prevent the final optional argument from being seen as such.
  - Unintended spaces could "orphan" final optional arguments.

[18] Before May 2018, `\Namei` worked like `\Nameii` does now. This manual checks for the appropriate changes and mitigates errors. Currently, xparse is just as stable as xargs and suffix.

The default behavior can cause problems when using interpolated text in square brackets following a name. We show this using both name interfaces. We simulate first instances of names (Section 9.3), and we suppress name formatting.

Name Pattern(s):
`Albert!Einstein`
`Miyazaki,Hayao`
Basic Index:
Einstein, Albert
Miyazaki Hayao

- We expect to see the following text:

  Albert Einstein [first] spoke about physics. Miyazaki Hayao [then] spoke about animation. Einstein [again] spoke about physics. Miyazaki [later] spoke about film.

- We use these macros and get poor results:

```
\Name[Albert]{Einstein} [first] spoke about physics.
\Name{Miyazaki, Hayao} [then] spoke about animation.
\Name[Albert]{Einstein} [again] spoke about physics.
\Name{Miyazaki, Hayao} [later] spoke about film.
```

  first Einstein spoke about physics. Miyazaki then spoke about animation. Einstein spoke about physics. Miyazaki spoke about film.

- We add explicit spaces or curly braces to get correct results:

```
\Name[Albert]{Einstein}\ [first] spoke about physics.
\Name{Miyazaki, Hayao}\ [then] spoke about animation.
\Name[Albert]{Einstein}{} [again] spoke about physics.
\Name{Miyazaki, Hayao}{} [later] spoke about film.
```

  Albert Einstein [first] spoke about physics. Miyazaki Hayao [then] spoke about animation. Einstein [again] spoke about physics. Miyazaki [later] spoke about film.

```
\begin{nameauth}
  \< Einstein & Albert & Einstein & >
  \< Miyazaki & & Miyazaki, Hayao & >
\end{nameauth}
```

- We expect to see the same text as above.
- We use these macros and get poor results:

```
\Einstein [first] spoke about physics.
\Miyazaki [then] spoke about animation.
\Einstein [again] spoke about physics.
\Miyazaki [later] spoke about film.
```

  first Einstein spoke about physics. Miyazaki then spoke about animation. Einstein spoke about physics. Miyazaki spoke about film.

- We add explicit spaces or curly braces to get correct results:

```
\Einstein\ [first] spoke about physics.
\Miyazaki\ [then] spoke about animation.
\Einstein{} [again] spoke about physics.
\Miyazaki{} [later] spoke about film.
```

  Albert Einstein [first] spoke about physics. Miyazaki Hayao [then] spoke about animation. Einstein [again] spoke about physics. Miyazaki [later] spoke about film.

4.3.2 Affixes Need Commas

A comma is not a required name element unless used in an ⟨*SNN,Affix*⟩ pair. When thus used it delimits a Western surname from its affix, an Eastern family name from a personal name, and an ancient name from an affix.

When a comma appears in a required name argument, each member of the ⟨*SNN,Affix*⟩ pair is treated as a separate argument. Thus, we say that the comma delimits or segments the name argument. Spaces around the comma are ignored. See Section 6.5 for more information when using macros in name arguments.

> Regarding an ⟨*SNN,Affix*⟩ pair, avoid error by applying macros to ⟨*SNN*⟩ and ⟨*Affix*⟩ as if they were independent arguments.

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

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

Name Pattern(s):
`Oskar!Hammerstein`
Basic Index:
Hammerstein, Oskar

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`

If the name displayed in the text shows both ⟨*SNN*⟩ and ⟨*Affix*⟩, then `\KeepAffix` turns the space **between** ⟨*SNN*⟩ and ⟨*Affix*⟩ into a non-breaking space. `\KeepAffix\Name{Louis, XIV}` will not break between Louis and XIV. All name types that use ⟨*Affix*⟩ are supported.

`\DropAffix`

If `\DropAffix` precedes only a Western name, it will suppress the affix after the surname in a first or long instance. We get "Oskar Hammerstein" From `\DropAffix\Name*[Oskar]{Hammerstein, II}`.

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.

`\ForceAffix`

Since the macro `\ForceAffix`, which prints only the affix of subsequent-use Western names, has such a narrow intended use, we cover it in Section 5.10.1.

`\ShowComma`
`\NoComma`

In a long name form, `\ShowComma` forces the display of 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.

Name Pattern(s):
`GeorgeS.!Patton,Jr.`
Basic Index:
Patton, George S., Jr.

With comma: George S. Patton, Jr. `\ShowComma\LPatton`

No comma: George S. Patton Jr. `\NoComma\LPatton`

Neither `\ShowComma`, `\NoComma`, nor their related package options interact with the use of `\RevComma` and friends (Sections 3 and 5.6).

4.3.3 Trimming Spaces

To get consistent index entries, all nameauth macros that take name arguments trim extra spaces around each name argument, shown in gray below:

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

We show this in practice while suppressing name formatting:

```
No spaces:\\
\fbox{\strut\Name*[Martin Luther]{King,Jr.}}
\fbox{\strut\Name [Martin Luther]{King,Jr.}}
\fbox{\strut\FName[Martin Luther]{King,Jr.}}

Spaces:\\
\fbox{\strut\Name*[ Martin  Luther ]{ King , Jr. }}
\fbox{\strut\Name [ Martin  Luther ]{ King , Jr. }}
\fbox{\strut\FName[ Martin  Luther ]{ King , Jr. }}
```

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

No spaces:

Martin Luther King Jr.	King	Martin Luther

Spaces:

Martin Luther King Jr.	King	Martin Luther

We resume name formatting to show names that look the same, but are different. Non-breaking spaces, explicit spaces, thin spaces, and macros that expand to spaces **are not trimmed**. They produce different name patterns, shown below:

Text	Macro / Arguments	Name Pattern
foo bar	`\Name{foo~bar}`	`foo~bar`
foo bar	`\Name{foo\ bar}`	`foo\bar`
foo bar	`\Name{foo\space bar}`	`foo\spacebar`

For more on uniqueness, see Sections 6.2 and 6.3. Names with macros in their arguments must be sorted with `\PretagName` (Section 7.6).

4.3.4 Formatting Initials

Some publishers are dead-set on having a space between initials. Many designers find that practice to be inelegant at best. [Bringhurst] wisely advises one to omit spaces between initials. Yet, fighting with one's editor does little good. If a style guide requires spaces, try thin spaces and sorting as below:

Name Pattern(s):
`E.\,B.!White`
Basic Index:
White, E. B.

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

Thin spaces	E.	B.	White
Normal spaces	E.	B.	White

We used `\PretagName` to sort this name as: "White, Elwyn" to ensure that his index entry is sorted properly. (Section 7.6.2).

4.3.5 Name Breakpoints

\KeepName

In longer name forms, \KeepName turns spaces **between** all visible name elements into non-breaking spaces. This macro does not alter spaces **within** name elements with multiple names, such as French and German forenames, as well as Spanish surnames. \KeepName works with all name types.

Name Pattern(s):
SandraDay!O'Connor
Basic Index:
O'Connor, Sandra Day

- Sandra Day O'Connor \Name[Sandra Day]{O'Connor} has two points where the name can break: after Sandra and after Day.
- Sandra Day O'Connor \KeepName\Name*[Sandra Day]{O'Connor} has one point where the name can break: after Sandra.

4.3.6 Full Stop Detection

Names may use full stops in some cases, including the following:

Name Pattern(s):
GeorgeS.!Patton,Jr.
J.D.!Rockefeller,IV
Marcus!Welby,M.D.
Basic Index:
Patton, George S., Jr.
Rockefeller, J.D., IV
Welby, Marcus, M.D.

- When forenames are abbreviated as initials: J.D. Rockefeller IV is J.D.
- After affixes: "Jr". (junior), "Sr". (senior), "d. Ä." (*der Ältere*), "d. J." (*der Jüngere*) etc.: George S. Patton Jr.
- In some contexts, after degrees like "M.D." (*Medicinae Doctor*), J.D. (*Juris Doctor*), Ph.D. (*Philosophiae Doctor*), etc.: Marcus Welby M.D.

The printed names above end with a full stop, and a full stop follows them in the text. The nameauth macros that print names in the text check if the name ends with a full stop. If that is true, and if the lookahead token contains a full stop, they gobble the lookahead token. Below we see how this works in greater detail:

```
\begin{nameauth}
  \< MLK & Martin Luther & King, Jr. & >
\end{nameauth}
```

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

Full stop gobbled: First use of a name; the last name element has a full stop and is followed by one: Martin Luther King Jr.
\Name[Martin Luther]{King, Jr.}. or \MLK.

Full stop gobbled: Long use of a name; the last name element has a full stop and is followed by one: Martin Luther King Jr.
\Name*[Martin Luther]{King, Jr.}. or \LMLK.

Full stop gobbled: Using ⟨*alternate*⟩ with long Western names will not affect the last element: Dr. M.L. King Jr.
\Name*[Martin Luther]{King, Jr.}[Dr. M.L.].
or \LMLK[Dr. M.L.].

Full stop gobbled: Using ⟨*alternate*⟩ with short Western names will affect the last element: Dr. M.L.
\FName[Martin Luther]{King, Jr.}[Dr. M.L.].
or \SMLK[Dr. M.L.].

Full stop remains: Affix drops in subsequent uses: King.
\Name[Martin Luther]{King, Jr.}. or \MLK.

Full stop remains: Affix forced to drop in first or long uses: Martin Luther King.
\DropAffix\Name*[Martin Luther]{King, Jr.}.
or \DropAffix\LMLK.

Curly braces can defeat full stop detection and create a new, unique name (Section 6.3), similar to using spaces between a name and a full stop:

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

Full stop remains: The whole name is within a group, but the full stop is outside of the group: Martin Luther King Jr..
`{\Name*[Martin Luther]{King, Jr.}}.` or `{\LMLK}.`

Full stop gobbled: Both name and full stop are in the group: Martin Luther King Jr.
`{\Name*[Martin Luther]{King, Jr.}.}` or `{\LMLK.}`

Full stop remains: Grouping tokens in ⟨*Affix*⟩ create a new name, break full stop detection (full stop in group): Martin Luther King Jr..
`\Name*[Martin Luther]{King, {Jr.}}.`

Full stop gobbled: Grouping tokens in ⟨*Affix*⟩ create a new name, do not break detection (full stop outside of group): Martin Luther King Jr.
`\Name*[Martin Luther]{King, {Jr}.}.`

Full stop remains: There is intervening space between the name and the full stop: Martin Luther King Jr. .
`\Name*[Martin Luther]{King, Jr.} .`

Full stop gobbled: Shorthands ignore spaces after them: Martin Luther King Jr.
`\LMLK .`

Full stop remains: Shorthands with optional arguments do not ignore intervening space: Dr. M.L. King Jr. .
`\LMLK[Dr. M.L.] .`

It is rare to see initials in Eastern names or ancient names. Nevertheless, if one had a Non-Western name form that terminated with a full stop, similar rules would apply. We demonstrate this below, but do not index the names:

Full stop gobbled: First use of a name; the last name element has a full stop and is followed by one: Full Stop.
`\Name{Full, Stop.}.`

Full stop gobbled: Long use of a name; the last name element has a full stop and is followed by one: Full Stop.
`\Name*{Full, Stop.}.`

Full stop gobbled: Using ⟨*alternate*⟩ with long Non-Western names will affect the last element: Full Cessation.
`\Name*{Full, Non-Stop}[Cessation.].`

In the other cases for Non-Western names, the full stop would not be gobbled because it would not terminate the displayed name.

> We believe firmly in the revelation of God in Jesus Christ. I can see no conflict between our devotion to Jesus Christ and our present action. In fact, I can see a necessary relationship. If one is truly devoted to the religion of Jesus he will seek to rid the earth of social evils. The gospel is social as well as personal.
>
> —Dr. Martin Luther King Jr.
> *Stride Towards Freedom* (1958)

4.3.7 Macros in Name Arguments

As we discuss specialized topics like language-specific needs and advanced formatting, we will encounter macros in name arguments with greater frequency. It would be good to get some general tips on how to handle such macros.

- Missing square brackets or curly braces can cause errors like "`Paragraph ended`" and "`Missing` ⟨*grouping token*⟩ `inserted.`"
- Alternate formatting (Section 11.2) helps to avoid potential problems, e.g., when using `\CapThis`.
- Use `\noexpand`⟨*macro*⟩ in name macro arguments as a best practice.
- Macros used in name arguments must be defined either in the preamble or in the outermost document scope to avoid 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`. Yet `\global` can precede the use of a macro defined with `\newcommand`. *The TEXbook*, pages 275–277, shows what `\global` can and cannot do. See more about this issue and `\newcommand` on **this page**.
- Below we show general aspects of scoping to apply them to this package:

```
\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 globally.
  - `\MacroA` is defined globally.
  - `\ifCondB` is defined locally.
  - `\ifCondC` is defined locally. No `\global\newif` allowed.
  - `\MacroB` is defined locally. No `\global\newcommand` allowed.
  - `\MacroC` is defined locally.
  - `\MacroD` is defined globally. The `\global` before `\MacroB` affects the `\def` inside `\MacroC`.
  - `\ifCondA` is true due to `\global`. Global assignment works, not global 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. The name patterns themselves are global, but macros in name arguments are not guaranteed to be so.

4.3.8 Active Unicode Characters

Both `xelatex` and `lualatex` support Unicode natively. Using the fontenc package with the `T1` option allows `latex` and `pdflatex` to use many Unicode characters via active characters that "secretly" invoke macros. Use `\PretagName` to sort names with these characters (Section 7.6).

Below we group characters that are supported in `pdflatex` without additional modification by accents and diacritical marks. We align similar letters, so that one gets a better idea of what letters take what accents.

Capitals																							
acute	Á	Ć		É	Ǵ	H	Í	J	K	Ĺ	Ń	Ó	Ŕ	Ś		Ú		Ý	Ź				
grave	À			È			Ì					Ò				Ù							
circumflex	Â	Ĉ		Ê	Ĝ	Ĥ	Î	Ĵ				Ô		Ŝ		Û	Ŵ	Ŷ					
tilde	Ã						Ĩ				Ñ	Õ				Ũ							
diaeresis[19]	Ä			Ë			Ï					Ö				Ü		Ÿ					
cedilla		Ç			Ģ				Ķ	Ļ	Ņ		Ŗ	Ş	Ţ								
macron	Ā			Ē	Ḡ		Ī					Ō				Ū	Æ	Ȳ					
breve	Ă				Ğ		Ĭ					Ŏ				Ŭ							
dot/dotless	Ḃ	Ċ		Ė	Ġ		İ												Ż				
ogonek	Ą			Ę			Į					Ǫ				Ų							
caron	Ǎ	Č	Ď	Ě	Ǧ		Ǐ		Ǩ	Ľ	Ň	Ǒ	Ř	Š	Ť	Ǔ			Ž				
various	Å	Æ	Đ	(eth)	Ð	(D stroke)					IJ	Ł	Ŋ	Ø	Œ	Ő	Ů	Ű	Ș	Ț	Þ		

Lowercase																							
acute	á	ć		é	ǵ		í			ĺ	ń	ó	ŕ	ś		ú		ý	ź				
grave	à			è			ì					ò				ù							
circumflex	â	ĉ		ê	ĝ	ĥ	î	ĵ				ô		ŝ		û	ŵ	ŷ					
tilde	ã						ĩ				ñ	õ				ũ							
diaeresis	ä			ë			ï					ö				ü		ÿ					
cedilla		ç			ģ				ķ	ļ	ņ		ŗ	ş	ţ								
macron	ā			ē	ḡ		ī					ō				ū	æ	ȳ					
breve	ă				ğ	h	ĭ					ŏ				ŭ							
dot/dotless	ḃ	ċ		ė	ġ		ɪ												ż				
ogonek	ą			ę			į					ǫ				ų							
caron	ǎ	č	ď	ě	ǧ		ǐ	ǰ	ǩ	ľ	ň	ǒ	ř	š	ť	ǔ			ž				
various	å	æ	ð	đ	ij	ł	ŋ	ø	œ	ő	ů	ű	ș	ß	ț	þ							

More Unicode characters can be made available when using fonts with TS1 glyphs (pages 455–463 in *The Latex Companion*). Compare **this page** or type either `texdoc comprehensive`, `texdoc symbols-A4`, or `texdoc symbols-letter` for more information.

When using a font with TS1 glyphs, the following preamble snippet is an example of how one might add more Unicode characters, such as a long s (*s-medialis*):

[19] A diaeresis mark is one way to indicate a sound change (*Umlaut*). German originally used a superscript `e` over `a`, `o`, and `u`. The cursive form of `e` simplified to a diaeresis in the 1800s. A diaeresis also signals pronouncing a diphthong or digraph as two monophthongs, e.g., "noëtic".

```
\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

4.3.9 Fragility of Active Unicode

This topic can affect different issues:

1. `\CapThis` does not cause errors because we check for an active Unicode character at the beginning of a name element. Macros in name arguments may need alternate formatting. See also Sections 5.7.4 and 11.2.3.
2. In older TEX distributions, names with active Unicode characters could expand differently in the index if the indexing macros were written first to the `aux` file. This wasmitigated since 2018.
3. If some of the macros represented by active characters get redefined, then both the names and their patterns that contain such characters may change unexpectedly, affecting both text and index.

TEX macros that partition their arguments can break active Unicode characters quite easily, as we show here with this delimited macro:

```
\def\Breaker#1#2#3!{<#1#2><#3>}
\def\Atsign{@}
\def\AtAt{\Atsign\Atsign}
```

The macro `\Breaker` will take an arbitrary stream of tokens as an argument until it terminates with a bang, which is a delimiter. It puts the first token in `#1`. The second goes in `#2`, while the third gets everything up to the bang.

Argument	Macro	Engine	Result
`abcde`	`\Breaker abcde!`	(any)	<ab><cde>
`{æ}bcde`	`\Breaker {æ}bcde!`	(any)	<æb><cde>
`\ae bcde`	`\Breaker \ae bcde!`	(any)	<æb><cde>
`a\AtAt bcde`	`\Breaker a\AtAt bcde!`	(any)	<a@@><bcde>
`æbcde`	`\Breaker æbcde!`	`xelatex`	<æb><cde>
`æbcde`	`\Breaker æbcde!`	`lualatex`	<æb><cde>
`æbcde`	`\Breaker æbcde!`	`pdflatex`	<æ><bcde>
`æbcde`	`\Breaker æbcde!`	`latex`	<æ><bcde>

Here is why the first two glyphs, or the first macro and glyph, are grouped together in `#1#2` below, and how this affects `nameauth`:

- The letter `a` is one argument, and `b` is the second.
- Since `{æ}` is in a group, it is one argument, and `b` is the second.
- The macro `\ae` also is one argument that prints æ, and `b` is the second.

- The letter `a` is one argument, and `\AtAt` is one argument that prints @@. But if it were evaluated or written to a file at some point during its expansion, for the purposes of name arguments in `nameauth`, something unexpected could occur.
- Both `xelatex` and `lualatex` treat the Unicode letter `æ` as one argument, with `b` as the second.
- In `latex` and `pdflatex`, however, `æ` is an active Unicode control sequence that uses two arguments all by itself. It fills up `#1#2`. The rest of the input is in `#3`.
- Any active Unicode character 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 }`.

Starting on page 151 we show how to 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` we capitalize `#1#2`, otherwise just `#1`. Should `#1` be a protected macro or something that does not expand to a sequence of letters, use `\AltCaps` (Section 11.2.4).

Back to Table of Contents

Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.

Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battle-field of that war. We have come to dedicate a portion of that field, as a final resting place for those who here gave their lives that that nation might live. It is altogether fitting and proper that we should do this.

But, in a larger sense, we can not dedicate — we can not consecrate — we can not hallow — this ground. The brave men, living and dead, who struggled here, have consecrated it, far above our poor power to add or detract. The world will little note, nor long remember what we say here, but it can never forget what they did here. It is for us the living, rather, to be dedicated here to the unfinished work which they who fought here have thus far so nobly advanced. It is rather for us to be here dedicated to the great task remaining before us — that from these honored dead we take increased devotion to that cause for which they gave the last full measure of devotion — that we here highly resolve that these dead shall not have died in vain — that this nation, under God, shall have a new birth of freedom — and that government of the people, by the people, for the people, shall not perish from the earth.

—Abraham Lincoln
Gettysburg Address (19 November 1863)

5 Language Topics

Here we cover technical issues related to the use of names in various languages and cultures. We draw on material from Sections 8.2 and 9.1. We also make use of the debugging macros in the next section. Although we show how non-English names are represented in English, this material can apply to other language standards.

5.1 Caveats with Active Characters

When using active characters, specific characters have been given the category code of a macro or control sequence in order to represent them in the classical regime of latex and pdflatex. Active characters and macros generate different names, even though the output on the page looks the same.

Active characters affect name patterns and index sorting, depending on the LaTeX engine being used. One finds more technical details on the matter in Section 4.3.8. We use `\PretagName` (Section 7.6) to get correct sorting and `\SkipIndex` (Section 7.4) to suppress bogus index entries:

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

1. Æthelred II ..... `\Name*{Æthelred, II}`

   We have seen this name earlier, as the formatting shows.[20]

   We sort this with `\PretagName{Æthelred, II}{Aethelred, 2}`

2. Æthelred II ..... `\SkipIndex\Name{\AE thelred, II}`

   This name is new, as the formatting shows us. It looks like the name above, but its control pattern differs by the macro `\AE`.

   We get a different index entry, regardless of how we sort it.

3. Boëthius ..... `\Name{Bo\"ethius}`

   This new name uses `\"e` to display a lowercase e with a diaeresis.

   We sort it with `\PretagName{Bo\"ethius}{Boethius}`.

4. Boëthius ..... `\SkipIndex\Name{Boëthius}`

   This name differs by the character ë. It cannot have the same index entry as Boëthius above.

5. Boëthius ..... `\SkipIndex\Name{Bo{\"e}thius}`

   Yet another different name by virtue of grouping tokens (Section 6.3). The index entry for this one is again different.

> For since good and bad, and likewise reward and punishment, are contraries, it necessarily follows that, corresponding to all that we see accrue as reward of the good, there is some penalty attached as punishment of evil. As, then, righteousness itself is the reward of the righteous, so wickedness itself is the punishment of the unrighteous.
>
> —Boëthius (Anic. Manl. Severinus Boethius)
> *On the Consolation of Philosophy* IV.III (523 AD)

[20] Regarding the margin note that shows name patterns, with `pdflatex` and `latex`, in `ÃĘthelred,II` the glyphs `ÃĘ` correspond to `\IeC{\AE}`. Likewise in `BoÃńthius` the glyphs `Ãń` correspond to `\IeC{\"e}`. The active Unicode character is being detokenized and its control sequence is interpreted as two glyphs. This is a result of using inputenc and fontenc.

5.2 Hyphenation

Unlike English standards up to the middle twentieth century, modern English conventions encourage that names of non-English origin can and should be hyphenated to reflect their cultural origins. For example, even with place names, modern English prefers German **Mainz** over the older English form **Mayence**. With nameauth, one can accommodate names from different cultures by using optional hyphens or packages like babel and polyglossia.

Default Hyphenation

Name Pattern(s):
`John!Strietelmeier`

Here we show potential issues that can arise when dealing with default English hyphenation. We get the name of a deceased, yet long-time, influential professor at Valparaiso University whose last name is of German origin, to hyphenate poorly.[21]

```
1 In English, some names come from other cultures. Names like
2 \Name[John]{Strietelmeier} can break badly in some cases.
```

In English, some names come from other cultures. Names like John Strietelmeier can break badly in some cases.

Discretionary Hyphens

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

We now use a different name from the name above. Even though it may look the same in the text, using discretionary hyphens will cause it to have a different index entry. As a result, one must use discretionary hyphens consistently in the macro arguments to avoid creating different names and index entries. Nevertheless, this name breaks properly.[22]

```
1 In English, some names come from other cultures. Names like
2 \Name[John]{Strie\-tel\-meier} can break badly in some cases.
```

In English, some names come from other cultures. Names like John Strietelmeier can break badly in some cases.

Language Packages

Name patterns and index entries are too big for the margin.

Here we use macros in name arguments that can work with either babel or polyglossia.[23] This approach has some caveats when using the macro `\CapThis`. Since the leading element of ⟨*SNN*⟩ is a macro, using `\CapThis` would halt LaTeX with errors unless we used alternate formatting (Section 11.2). Related issues appear in Sections 5.7 and 6.5, showing ways to mitigate possible errors.

> Address any concerns about expansion of macros in name arguments by using `\noexpand` before those macros.

[21] Not shown in these examples, we use `\textls` from microtype to ensure bad breaks. We omit all forms of this name from the index except for `\Name[John]{\noexpand\de{Strietelmeier}}` because we use babel-related macros.

[22] Here again, we omit `\Name[John]{Strie\-tel\-meier}` from the index.

[23] For best results, one should consult the documentation for babel and polyglossia.

```
\documentclass{article}

\newcommand\nolangs{} % Do not load the language packages via compat.tex
\input{compat.tex} % Included with nameauth for compatibility.

\ifloadbabel % Defined in compat.tex
% For any LaTeX format; compat.tex makes \loadbabel true
% only for use with dvilualatex, pdflatex, and latex.

  \usepackage[main=american,ngerman]{babel}
  \usepackage[babel=true]{microtype} % if desired

% Convenience macros, classical babel approach
  \newcommand{\de}[1]{\foreignlanguage{ngerman}{#1}}

\else
% For XeLaTeX and LuaLaTeX; compat.tex makes \loadbabel
% false by default, which is the historic behavior.
% Currently, one also can use babel with these engines.
  \usepackage{polyglossia}
  \setdefaultlanguage[variant=american]{english}
  \setotherlanguage[variant=german, spelling=new]{german}
  \usepackage{microtype} % if desired

% Convenience macros, not using babel-equivalent macros
  \newcommand{\de}[1]{\textgerman{#1}}
\fi

% Other setup
\usepackage{nameauth}

\begin{nameauth}
\< Striet & John & \noexpand\de{Strietelmeier} & >
\end{nameauth}

\PretagName[John]{\noexpand\de{Strietelmeier}}
  {Strietelmeier, John}

\begin{document}

In English, some names come from other cultures. Names like
\Striet\ or \Name[John]{\noexpand\de{Strietelmeier}} can
break badly in some cases.

\end{document}
```

The name pattern is: `John!\noexpand\de{Strietelmeier}`. The example above gives us the following:

> In English, some names come from other cultures. Names like John Strietelmeier or Strietelmeier can break badly in some cases.

5.3 Eastern Names

Proper Eastern names are encoded using comma-delimited syntax. They have Non-Western index entries: ⟨*SNN*⟩ ⟨*Affix*⟩ (no comma between name elements). The current syntax permits alternate names; the obsolete syntax does not (Section 12.2).

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

Name Pattern(s):
`Sun,Yat-sen`
Basic Index:
Sun Yat-sen

Even in some academic texts, one sees Non-Western names encoded with Western forms.[24] The nameauth package seeks to remedy that issue, which has more to do with outsourcing indexes to non-expert providers than with scholars. For example, the macro `\Name*{Sun, Yat-sen}` ensures correct forms of the name Sun Yat-sen in the text and the index. We get both cross-cultural sensitivity and scholarly accuracy.

5.4 Names in All Caps

`\AllCapsActive`
`\AllCapsInactive`
`\CapName`

Some contexts capitalize one's entire family name. This differs from the way in which initial letter capitalization is handled (Section 5.7). In addition to the `allcaps` package option (Section 2), `\AllCapsActive` and `\AllCapsInactive` work for blocks of text. All have priority over `\CapName`, which works once per name. These capitalize ⟨*SNN*⟩ in the body text only. They also work with `\AKA` and friends. For capitalization using macros in both the text and index, see Sections 11.2 and 11.5.

`\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.

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

Text	Macro
Hideyo Noguchi	`\LNoguchi`
Hideyo NOGUCHI	`\CapName\LNoguchi`
Miyazaki Hayao	`\LMiyazaki`
MIYAZAKI Hayao	`\CapName\LMiyazaki`

5.5 Reversed Names

Sometimes a publisher expects Western-style index entries. Sometimes certain names need to have specific forms in the index, The reversing macros can adapt forms in the text to required Western index entries.

`\ReverseActive`
`\ReverseInactive`
`\RevName`

In addition to the `allreversed` option for reversing name order (Section 2), `\ReverseActive` and `\ReverseInactive` do the same for blocks of text. These all have priority over the use of `\RevName`, used once per name. These macros do not affect the index. They work also with `\AKA` and friends. **Reversing only affects long name forms,** which is why we show only long names below.

`\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.

[24] Jaroslav Pelikan, *The Christian Tradition*, 5 vols. (Chicago: Chicago UP, 1971–1989); Immanuel Geiss, *Personen: Die biographische Dimension der Weltgeschichte*, Geschichte Griffbereit vol. 2 (Munich: Wissen Media Verlag, 2002). As of writing, **NNDB** sorts Kim Jong Un under U, not K. That prioritizes accesibility over accuracy. This is not the publisher's fault; it is a market reality.

Unwanted results that may arise from reversing names depend more on the context than on the syntactic forms of the names themselves. This is the crux of how nameauth deals with ambiguity across cultures.

Reversing Western Names

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

Name Pattern(s):
`GeorgeS.!Patton,Jr.`
Basic Index:
Patton, George S., Jr.

Avoid using ⟨*Affix*⟩ with such names. For example, `\RevName\LPatton\dag` produces "Patton Jr. George S.†". The name looks wrong unless one uses either `\RevComma` or `\DropAffix`. This name has a Western index entry.

Name Pattern(s):
`Hideyo!Noguchi`
Basic Index:
Noguchi, Hideyo

Text	Macro / Arguments
Hideyo Noguchi	`\LNoguchi`
Doctor Noguchi	`\LNoguchi[Doctor]`
Unwanted result: Sensei Noguchi	`\LNoguchi[Sensei]`
Noguchi Hideyo†	`\RevName\LNoguchi\dag`
Noguchi Sensei†	`\RevName\LNoguchi[Sensei]`
Unwanted result: Noguchi Doctor	`\RevName\LNoguchi[Doctor]`

Reversing Eastern Names

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

The index entry of this name form looks like ⟨*SNN*⟩ ⟨*FNN*⟩ (no comma). This name has a Non-Western index entry. **Reversing ancient names functions the same way, but often produces nonsensical name forms.**

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

Text	Macro / Arguments
Miyazaki Hayao	`\LMiyazaki`
Miyazaki Sensei	`\LMiyazaki[Sensei]`
Unwanted result: Miyazaki Mr.	`\LMiyazaki[Mr.]`
Hayao Miyazaki	`\RevName\LMiyazaki`
Mr. Miyazaki	`\RevName\LMiyazaki[Mr.]`
Unwanted result: Sensei Miyazaki	`\RevName\LMiyazaki[Sensei]`

5.6 Listing Western names by Surname

`\ReverseCommaActive`
`\ReverseCommaInactive`
`\RevComma`

To make lists of "last comma first" names, in addition to the `allrevcomma` option (Section 2), the macros `\ReverseCommaActive` and `\ReverseCommaInactive` function the same way with blocks of text.They both have priority over `\RevComma`. These all affect only long Western name forms. The first two are broad toggles, while the third is used once per name.

`\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.

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

- Western names change to reversed comma form:

Oskar Hammerstein II	Hammerstein II, Oskar
Hideyo Noguchi	Noguchi, Hideyo

- Non-Western names do not change:

Æthelred II	Æthelred II
Sun Yat-sen	Sun Yat-sen

5.7 Particles in Names

Particles usually are articles and prepositions grouped either with forenames or surnames. They can refer to places, noble houses, and so on.

In modern usage, particles have crossed into other languages and cultures. The same particles may be handled differently, depending on their source language. Users should consult style guides. See also Section 11.2.5.

> One must sort names with particles in ⟨*SNN*⟩ using `\PretagName` (Section 7.6), often by removing spaces in the sort key.

5.7.1 Standard Rules

Here we show how [Mulvany, 152–82] handles name particles, thus pertaining to naming macro arguments. This list is not exhaustive.

- **Afrikaans and English variants:** Particles go in ⟨*SNN*⟩. These can include *de*, *de la*, *d'*, *von*, *van*, and *ten*. Capitalization of particles may vary, but *Du* usually is capitalized, yet one sees, e.g., "du Cange", "Du Cange", and even "Ducange". *Le*, *La*, and *L'* are capitalized unless preceded by *de*.

Name Pattern(s):
S.J.!DuToit
Martin!VanBuren
Pierre!L'Enfant
Walter!delaMare
Hernando!de~Soto
Basic Index:
Du Toit, S.J.
Van Buren, Martin
L'Enfant, Pierre
de la Mare, Walter
de Soto, Hernando

Stephanus Johannes Du Toit
`\Name[S.J.]{Du Toit}[Stephanus Johannes]`
Du Toit ..... `\Name[S.J.]{Du Toit}`

Martin Van Buren ..... `\Name[Martin]{Van Buren}`
Van Buren ..... `\Name[Martin]{Van Buren}`

Pierre L'Enfant ..... `\Name[Pierre]{L'Enfant}`
L'Enfant ..... `\Name[Pierre]{L'Enfant}`

Walter de la Mare ..... `\Name[Walter]{de la Mare}`
de la Mare ..... `\Name[Walter]{de la Mare}`
De la Mare ..... `\CapThis\Name[Walter]{de la Mare}`

Hernando de Soto ..... `\Name[Hernando]{de~Soto}`
de Soto ..... `\Name[Hernando]{de~Soto}`
De Soto ..... `\CapThis\Name[Hernando]{de~Soto}`

Name Pattern(s):
Bernhardten!Brink
Daisy!VerBoven
Basic Index:
Brink, Bernhard ten
Ver Boven, Daisy

- **Dutch:** Particles go in ⟨*FNN*⟩, except for *ver*, which goes in ⟨*SNN*⟩.

Bernhard ten Brink ..... `\Name[Bernhard ten]{Brink}`
Brink ..... `\Name[Bernhard ten]{Brink}`

Daisy Ver Boven ..... `\Name[Daisy]{Ver Boven}`
Ver Boven ..... `\Name[Daisy]{Ver Boven}`

- **French:** If the surname has an article (or contracted article) plus preposition, the particles go in ⟨*SNN*⟩. Otherwise, prepositions by themselves go in ⟨*FNN*⟩.

Name Pattern(s):
`Jeande!LaFontaine`
Basic Index:
La Fontaine, Jean de

Jean de La Fontaine ..... `\Name[Jean de]{La Fontaine}`
La Fontaine ..... `\Name[Jean de]{La Fontaine}`

- **German:** If the surname has an article (or contracted article), optionally plus a preposition, the particles go in ⟨*SNN*⟩. Otherwise, the particles (usually just prepositions) go in ⟨*FNN*⟩.

Name Pattern(s):
`J.W.von!Goethe`
`Otto!ZurLinde`
Basic Index:
Goethe, J.W. von
Zur Linde, Otto

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

Otto Zur Linde ..... `\Name[Otto]{Zur Linde}`
Zur Linde ..... `\Name[Otto]{Zur Linde}`

- **Modern Italian:** Particles go in ⟨*SNN*⟩.
- **Medieval Italian:** Particles usually go in ⟨*FNN*⟩.
- **Portuguese:** Particles go in ⟨*FNN*⟩.
- **Romanian:** Particles go in ⟨*SNN*⟩, except for *de*, which goes in ⟨*FNN*⟩.
- **Scandinavian languages:** Particles go in ⟨*FNN*⟩ if they are of Scandinavian, Dutch, or German origin, except for the Dutch particle *de* and particles of other origin, which go in ⟨*SNN*⟩.
- **Spanish:** If the surname has an article, it goes in ⟨*SNN*⟩. Otherwise, particles go in ⟨*FNN*⟩. Compound surnames are indexed by their first element, including names connected by *y* (and).

Name Pattern(s):
`Franciscode!Figueroa`
`ManuelAntonio!LasHeras`
Basic Index:
Figueroa, Francisco de
Las Heras, Manuel Antonio

Francisco de Figueroa ..... `\Name[Francisco de]{Figueroa}`
Figueroa ..... `\Name[Francisco de]{Figueroa}`

Manuel Antonio Las Heras .. `\Name[Manuel Antonio]{Las Heras}`
Las Heras ..... `\Name[Manuel Antonio]{Las Heras}`

- **Modern Welsh, Irish, and Scots contexts:** names often merge particles with surnames: FitzRoy or Fitzroy; O'Leary; McDonald, MacLeish.

5.7.2 Non-Breaking Spaces

Despite the macros in Section 4.3.2, names with particles present their own challenges. We recommend inserting a tilde (active character equivalent to `\nobreakspace`) or Unicode NBSP as needed to prevent bad breaks (see also Sections 7.6). Here the quick interface helps greatly.

5.7.3 Look-Alike Particles

There are characters that look similar, but in fact are different. For example, *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 4.3.8). If one confuses these similar characters, spurious results can occur.

5.7.4 Capitalizing Particles

\CapThis In English, names like Hernando de Soto have their particles in the ⟨*SNN*⟩ argument: de Soto. At the beginning of a sentence, one must capitalize the name:

> De Soto was a famous Spanish explorer.
>
> `\CapThis\Soto\ was a famous Spanish explorer.`

With `latex` and `pdflatex`, using `\CapThis` should work with all of the Unicode characters available in the T1 encoding. For a broader set of characters, consider using `xelatex` and `lualatex`.

Section 10.1.3 applies `\CapThis` to names that have nonstandard capitalization. Section 11.2 explains how to avoid potential problems with `\CapThis` when using macros in name arguments. That is a trade-off for using a natural language approach.

The macro `\CapName` overrides the ⟨*SNN*⟩ created by `\CapThis`, effectively nullifying its effects. It is unlikely that the two macros would be used together,so this should not present a problem.

\AccentCapThis Before nameauth used automatic Unicode detection, `\AccentCapThis` placed before `\CapThis` handled Unicode. It remains only for backward compatibility.

5.8 Medieval Names

Medieval (and some ancient) names present some interesting difficulties, often based on the expected standards of the context in which they are used:

- Some publications present medieval names as Western names due to common convention. That also is reflected in index styles.
- Scholarly publications (sometimes) do not sacrifice details for the sake of convention. This makes their indexes more unfamiliar to the general reader. One might create cross-references from common forms to scholarly forms using `\IndexRef`, as explained in Section 7.3.

The design of nameauth helps users implement both approaches when needed, even though the two approaches are not compatible with each other. Indeed, we show both common and scholarly approaches in this manual. The trade-off for flexibility means that users have to put more work into designing how they address this complexity. That is why the manual provides so many examples.

In the following preamble snippet we illustrate how some different approaches may work, and how, even in one document, they can be made to play nicely with each other. Yet we should emphasize that, in general, a document will not use both common and scholarly approaches. We show the "worst-case" scenario in order to demonstrate how much easier the normal cases can be.[25]

```
1  \PretagName{Thomas, à~Kempis}{Thomas Akempis}  % medieval
2  \PretagName[Thomas]{à~Kempis}{Akempis, Thomas} % Western
3
4  % We do not index an alternate medieval form
5  \ExcludeName{Thomas, \‘a~Kempis}
6
```

[25] Regarding the margin note that shows name patterns, in `Thomas,Ãã~Kempis` the glyphs `Ãã` correspond to `\IeC{\'a}`. This is a result of using inputenc and fontenc.

```
% If we did use the alternate form, we would sort it as
\PretagName{Thomas, \‘a~Kempis}{Thomas Akempis}

\begin{nameauth}
  \< KempMed & & Thomas,   à~Kempis & >        % medieval
  \< KempW     & Thomas  & à~Kempis & >        % Western
\end{nameauth}

% Create xref before using the Western name.
\IndexRef[Thomas]{à~Kempis}{Thomas à~Kempis}
```

Below we illustrate different name forms using statements numbered from 1 to 10. The marginal information pertains as indicated to those respective numbered statements to show important differences among the name forms.

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)

- Medieval form: `\KempMed`

  1. In the text, we see “Thomas à Kempis” and “Thomas”. The added name “à Kempis” `\ForceFN\SKempMed` is a place name, not a surname. It is Latin for *von Kempen*.
  2. “Thomas” `\KempMed` is indexed as “Thomas à Kempis”.
  3. “À Kempis” `\CapThis\ForceFN\SKempMed` can start a sentence, but such usage would be infrequent.
  4. We use `\PretagName` (Section 7.6) to sort the name according to the expected collating sequence:
     `\PretagName{Thomas, à~Kempis}{Thomas Akempis}`

- Alternate medieval form: `\Name{Thomas,\‘a~Kempis}`

  5. “Thomas à Kempis” is a different name with a different pattern, about which the next section goes into some detail.
  6. We used `\ExcludeName` (Section 7.3.2) before using the alternate form to keep it from generating a duplicate index entry.
  7. We index the canonical form here with `\JustIndex\KempMed`.

- Western form: `\KempW`

  8. “Thomas à Kempis” is a Western name form with the index entry: “à Kempis, Thomas”.
  9. “À Kempis” appears by using `\CapThis\KempW`. We refer to “Thomas” via `\SKempW`. We get the same breadth of usage as the Non-Western forms, but the index entries differ, as do the specific ways of using the names.
  10. We first created a cross-reference from the Western form to the medieval form to prevent spurious page entries (Section 7.3). We index the canonical form: `\JustIndex\KempMed`.

      To prevent the name from appearing before “`aardvark`” in the index, we remove extra spaces to get the proper sorting between “`ajar`” and “`alkaline`”:
      `\PretagName[Thomas]{à~Kempis}{Akempis, Thomas}`

5.9 Ancient Names

Ancient names are fluid regarding the meaning and use of affixes. Certain scholarly contexts add more information to a name when it is first introduced. How do we handle that? We use information from the latter sections mentioned below. First we manually add affixes to a standard name:

- We use `\PretagName` (Section 7.6) to sort especially Roman numerals in the index. For example:

  Name Pattern(s): `Antiochus,IV!PRE`

  `\PretagName{Antiochus, IV}{Antiochus 4}`

- We use `\TagName` (Section 7.7) to ensure that added information is displayed in the index:

  Name Pattern(s): `Antiochus,IV!TAG`

  `\TagName{Antiochus, IV}{ Epiphanes, king}`.

- Using `\PretagName` and `\TagName` in the preamble ensures consistency.

- We use ⟨*Alternate*⟩ to add information in the text, e.g.:

  Name Pattern(s): `Antiochus,IV!MN`

  Antiochus IV Epiphanes.....`\Name {Antiochus, IV}[IV Epiphanes]`
  Antiochus IV.....`\Name*{Antiochus, IV}`
  Antiochus.....`\Name {Antiochus, IV}`

Next we show a snippet that uses the quick interface with this same method. We trigger a first use, followed by long and short subsequent uses:

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

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

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

Below, instead of manually using ⟨*Alternate*⟩, we use a more complex initial setup that automates tags via in first-use formatting hooks (Section 8.2). We also show that we can still have manual changes, here with babel and a macro `\el` that we made for using Greek, which is part of this manual's preamble:

```
6 \NameAddInfo{Demetrius, I}{ Soter}
7 \renewcommand*\NamesFormat[1]
8   {\color{blue}\sffamily#1\NameQueryInfo{}}
9 \renewcommand*\MainNameHook{\sffamily}
```

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

Demetrius I Soter.....`\ForgetThis\Dem`
Demetrius I.....`\LDem`
Demetrius.....`\Dem`
Demetrius I Σωτήρ.....`\LDem[I \noexpand \el{Σωτήρ}]`

The full index entry for both cases of Demetrius in a normal document would look like: `Demetrius 1@Demetrius I Soter, king`. The first use of a name, or one after `\ForgetName` or `\ForgetThis`, shows the most information. A long instance of an extant name shows less info, and a short instance shows the least.

5.10 Roman Names I

In this section we take a simple approach. Later on we will use macros in the arguments (Section 11.3). Roman names consist of the following elements:

- A personal name (*praenomen*)
- A clan name (*nomen*)
- An epithet or nickname (*cognomen*)

The *cognomen* can denote a branch family in a larger clan. Affixed to that, one might see additional names (*agnomina*). In ancient Rome, the *nomen* was most important, helping to structure Roman society. Naming conventions changed over time. *Praenomina* also have their own history. One finds a helpful video on the YouTube channel PolýMATHY by Luke Ranieri. We **do not use** the method where Roman names are indexed using recognizable mononyms.[26]

5.10.1 General Reference Materials

The basic approach uses Western name forms, but with some modifications. One often sees this method when indexing well-known Romans.[27]

Names without *Agnomina*

Name Pattern(s):
`Julius!Caesar`
`Pontius!Pilate`
Basic Index:
Caesar, Julius
Pilate, Pontius

Below we have the *nomen* in ⟨*FNN*⟩, the *cognomen* in ⟨*SNN*⟩, and we add the *praenomen* with the *nomen* in ⟨*Alternate*⟩ as needed:

Gaius Julius Caesar ..... `\Name[Julius]{Caesar}[Gaius Julius]`
Julius Caesar ..... `\Name*[Julius]{Caesar}`
Caesar ..... `\Name[Julius]{Caesar}`

If indexing according to the *cognomen*, perhaps use a Western name, where the *nomen* is in ⟨*FNN*⟩ and the *cognomen* in ⟨*SNN*⟩:

Pontius Pilate ..... `\Name*[Pontius]{Pilate}`

Name Pattern(s):
`Pontius,Pilate`
Basic Index:
Pontius Pilate

In Section 1.4.5 we mentioned that some Roman names could have Non-Western forms, especially those without *praenomina*. They would be indexed according to the *nomen*, which we see in some scholarly resources (Section 5.10.2).

Pontius Pilate ..... `\Name{Pontius, Pilate}`
Pilate ..... `\ForceFN\FName{Pontius, Pilate}`

Fere libenter homines, id quod volunt, credunt.
(In general, people willingly believe what they want [to believe].)

—Julius Caesar
De bello gallico 3.16.6 (58–49 BC)

[26] Justo L. González, *The Story of Christianity*, 2 vols. (San Francisco: Harper, 1984). [Mulvany] does not advise using such a method, which is ill-suited for `nameauth`.

[27] Philip J. Adler, *World Civilizations*, 3rd ed. (Belmont, Calif.: Thomson/Wadsworth, 2003).

Names with *Agnomina*: Tags

We can use index "pre-tags" and tags (Sections 7.6 and 7.7.2) to sort and differentiate names in the index. We also can use name tags (Section 8) for *agnomina*. Yet Romans tended to use and pass down a relatively small set of names over many generations.

By defining empty macros `\nri`, `\nrii`, `\nriii`..., we can disambiguate identical names. We use `\nr` (number) followed by Roman numerals to avoid collisions with, for example, the dotless ı. When put at the tail of name arguments, they avoid the need to use either `\noexpand` or alternate formatting. We can reuse these macros with many different names.

This approach solves most issues in the text. If we need specific index forms, we could index under alternate names, as we do on the next page. With a nod to Section 8.2, we automate the first use of a name to display the name tag. In a normal LaTeX file, we would see the following results:

```
\newcommand\nri{}
\newcommand\nrii{}

\begin{nameauth}
  \< Senecai  & Lucius Annaeus & Seneca\nri  & >
  \< Senecaii & Lucius Annaeus & Seneca\nrii & >
\end{nameauth}

\renewcommand*\NamesFormat[1]
  {\color{blue}\sffamily#1\NameQueryInfo{}}

\PretagName[Lucius Annaeus]{Seneca\nri}{Seneca, Lucius Annaeus 1}
\PretagName[Lucius Annaeus]{Seneca\nrii}{Seneca, Lucius Annaeus 2}
\TagName[Lucius Annaeus]{Seneca\nri}{ (the Elder)}
\TagName[Lucius Annaeus]{Seneca\nrii}{ (the Younger)}
\NameAddInfo[Lucius Annaeus]{Seneca\nri}{ the Elder}
\NameAddInfo[Lucius Annaeus]{Seneca\nrii}{ the Younger}

We can differentiate between
\Senecai\ and \Senecaii;
\Senecai \NameQueryInfo{} and \Senecaii \NameQueryInfo{}.

Their full index entries are:\\
\ShowIdxPageref[Lucius Annaeus]{Seneca\nri}\\
\ShowIdxPageref[Lucius Annaeus]{Seneca\ii}
```

Name patterns and index entries are too big for the margin.

We can differentiate between Lucius Annaeus Seneca the Elder and Lucius Annaeus Seneca the Younger; Seneca the Elder and Seneca the Younger.

Their full index entries are:
Seneca, Lucius Annaeus 1@Seneca, Lucius Annaeus (the Elder)
Seneca, Lucius Annaeus 2@Seneca, Lucius Annaeus (the Younger)

Bellum parate, quoniam pacem pati non potuistis.

(Prepare for war, for it seems that you are unable to tolerate peace.)

—Publius Cornelius Scipio Africanus
attr. by Titus Livius, *Ab urbe condita* B 30 (27–9 BC)

Names with *Agnomina*: ⟨*Affix*⟩

`\ForceAffix` In subsequent uses of a Western name, the prefix macro `\ForceAffix` allows one to print just ⟨*Affix*⟩ by itself. `\ForceFN` has a similar use with Non-Western names.

> Where formatting hooks also change syntactic name forms (Section 11f.), that could interfere with `\ForceFN` and `\ForceAffix`.

We move *agnomina* into the ⟨*Affix*⟩ field. The *agnomen* drops automatically in the text for subsequent name uses, but we get potentially undesirable index entries:

> The text has: `\Name[`⟨*praenomen*⟩ ⟨*nomen*⟩`]{`⟨*cognomen*⟩`,`⟨*agnomen*⟩`}`
>
> In the index we get: ⟨*cognomen*⟩, ⟨*praenomen*⟩ ⟨*nomen*⟩, ⟨*agnomen*⟩
>
> In the index we want: ⟨*cognomen*⟩ ⟨*agnomen*⟩, ⟨*praenomen*⟩ ⟨*nomen*⟩

We avoid this problem by using `\ExcludeName` for names that have *agnomina* in ⟨*Affix*⟩. We index different name forms, as in the next example:

```
\begin{nameauth}
  \< CatoY & Marcus Porcius & Cato, the Younger & >
  \< CatoYi & Marcus Porcius & Cato the Younger & >
\end{nameauth}
\renewcommand*\NamesFormat[1]
  {\color{blue}\sffamily#1\NameQueryInfo{}}
\ExcludeName[Marcus Porcius]{Cato, the Younger}
\makeatletter
\NameAddInfo[Marcus Porcius]{Cato, the Younger}
  {\if@nameauth@InHook\ (Uticensis, 94--45\ %
   {\rmfamily\textsc{bc}})\else Uticensis\fi}
\makeatother

\JustIndex\CatoYi
\CatoY\ is known as \CatoY\ \ForceAffix\CatoY, to distinguish him
from his great-grandfather. His names include \SCatoY[Marcus]
(\textit{praenomen}), \SCatoY[Porcius] (\textit{nomen}, one of the
Porcia clan), and \CatoY, the \textit{cognomen} by which he is best
known. His other \textit{agnomen}, \NameQueryInfo{}, means he was
born in Utica.
\JustIndex\CatoYi He is indexed as: \ShowIdxPageref*{}
```

Name patterns and index entries are too big for the margin.

Marcus Porcius Cato the Younger (Uticensis, 94–45 BC) is known as Cato the Younger, to distinguish him from his great-grandfather. His names include Marcus (*praenomen*), Porcius (*nomen*, one of the Porcia clan), and Cato, the *cognomen* by which he is best known. His other *agnomen*, Uticensis, means he was born in Utica. He is indexed as: Cato the Younger, Marcus Porcius

---

Igitur qui desiderat pacem, praeparet bellum.

(Accordingly, the person who would desire peace prepares for war.)

—Publius Vegetius Renatus
De re militari (c. 390 AD)

5.10.2 Scholarly Works

The *Oxford Classical Dictionary* and other scholarly sources index according to the *nomen*. That approach moves the *nomen* from ⟨*FNN*⟩ to ⟨*SNN*⟩, making this indexing method incompatible with the previous section:

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

Macro: `\IndexName[`⟨*praenomen*⟩`]{`⟨*nomen*⟩ ⟨*cognomen*⟩ ⟨*agnomen*⟩`}`

Name Pattern(s):
`Lucius!SergiusPaulus`
Basic Index:
Sergius Paulus, Lucius

By putting the *nomen* and *cognomen* into ⟨*SNN*⟩, with the *praenomen* in ⟨*FNN*⟩ and *agnomina* in name tags, we can take a basic approach to these names.[28]

Lucius Sergius Paulus ..... `\Name[Lucius]{Sergius Paulus}`
Sergius Paulus ..... `\Name[Lucius]{Sergius Paulus}`

If one is indexing according to the *cognomen*, one must index by an alternate name. This also allows for greater flexibility.

- Use this naming approach, seen in Section 5.10.1:

  `\Name[`⟨*praenomen*⟩ ⟨*nomen*⟩`]{`⟨*cognomen*⟩`,`⟨*agnomen*⟩`}`

  This allows the standard `nameauth` logic to handle changes of name forms, as well as the use of `\ForceAffix`.
- Use `\ExcludeName` for every name that should not appear in the index.
- Use `\IndexName` or `\JustIndex` as needed:

  `\IndexName[`⟨*praenomen*⟩`]{`⟨*nomen*⟩ ⟨*cognomen*⟩ ⟨*agnomen*⟩`}`

```
\begin{nameauth}
  \< CatoE & Marcus Porcius & Cato, the Elder & >
  \< CatoEi & Marcus & Porcius Cato the Elder & >
\end{nameauth}
\ExcludeName[Marcus Porcius]{Cato, the Elder}

\JustIndex\CatoEi
\DropAffix\CatoE\ (234--149 \textsc{bc}) is known as
\CatoE{} \ForceAffix\CatoE, among other sobriquets.
His \textit{praenomen} is \SCatoE[Marcus]. His
\textit{nomen} is \SCatoE[Porcius], a member of the
Porcia clan. His \textit{cognomen} is \CatoE.
\JustIndex\CatoEi
Index: \ShowIdxPageref*{}
```

Name patterns and index entries too large for margin.

Marcus Porcius Cato (234–149 BC) is known as Cato the Elder, among other sobriquets. His *praenomen* is Marcus. His *nomen* is Porcius, a member of the Porcia clan. His *cognomen* is Cato. Index: Porcius Cato the Elder, Marcus

Back to Table of Contents

[28] Cf. Paul L. Maier, *In the Fullness of Time: A Historian Looks at Christmas, Easter, and the Early Church*, revised ed. (San Francisco: Harper, 1991).

6 Debugging

Name patterns — generally unique macros based on names that usually expand to the empty string — govern interactions in nameauth. We take a deep dive here because name patterns greatly impact the remainder of the manual. We start to show index entries in the body text instead of the margin due to their complexity.

6.1 Null Arguments

All macros that evaluate name arguments (Section 1.6.1) leave information about the last name that was evaluated. Since version 4.1, some macros that take name arguments also can have a **null argument.** That occurs when the {⟨*SNN,Affix*⟩} argument is empty. Instead of creating a package error (the normal case), **all macro arguments are ignored** and the macro uses the persistent information from the last name arguments that were evaluated:

> Using the example Confucius, the following are equivalent:
> Confucius . . . . . \ShowPattern{}
> Confucius . . . . . \ShowPattern[this]{}[that]

- \@nameauth@LoadArgs isolates each name element, removes surrounding spaces, generates possible name patterns, and saves token registers (page 155 and following).
- \@nameauth@Choice determines the name type, then sets flags and selects a pattern according to that name type. Then it executes its arguments.
- The macro \NameauthPattern that expresses both the name and its type is covered in Section 6.2.
- The flags \ifNameauthWestern and \ifNameauthObsolete that express the name type are discussed in Section 11.4.

The macros that use both name arguments and null arguments include:

§ 6.6 \ShowPattern
§ 6.6 \ShowIdxPageref
§ 6.6 \ShowIdxPageref*
§ 6.6 \ShowNameInfo
§ 6.6 \ShowNameState
§ 6.6 \ShowNameState*
§ 8 \NameQueryInfo

No nameauth macros that take null arguments change name information; they only display information. This avoids errors.

For a truth, once established by proof, does neither gain force nor certainty by the consent of all scholars, nor lose by the general dissent.

—Maimonides
Guide for the Perplexed (1190)

6.2 Name Pattern Overview

\NameauthPattern

All macros that evaluate name arguments (Section 1.6.1) globally assign to \NameauthPattern a value that corresponds to the last-used name. The ⟨*Alternate*⟩ argument only affects patterns when using the obsolete syntax (Section 12.2).

Macro Name Arguments	Name Patterns	Name Type
[⟨*FNN*⟩]{⟨*SNN*⟩}	⟨*FNN*⟩!⟨*SNN*⟩	Western
[⟨*FNN*⟩]{⟨*SNN*⟩}[⟨*Alt*⟩]	⟨*FNN*⟩!⟨*SNN*⟩	Western
[⟨*FNN*⟩]{⟨*SNN,Affix*⟩}	⟨*FNN*⟩!⟨*SNN*⟩,⟨*Affix*⟩	Western
[⟨*FNN*⟩]{⟨*SNN,Affix*⟩}[⟨*Alt*⟩]	⟨*FNN*⟩!⟨*SNN*⟩,⟨*Affix*⟩	Western
{⟨*SNN,Affix*⟩}	⟨*SNN*⟩,⟨*Affix*⟩	Non-Western
{⟨*SNN,Affix*⟩}[⟨*Alt*⟩]	⟨*SNN*⟩,⟨*Affix*⟩	Non-Western
{⟨*SNN*⟩}[⟨*Alt*⟩]	⟨*SNN*⟩,⟨*Alt*⟩	old syntax
{⟨*SNN*⟩}	⟨*SNN*⟩	Non-Western

In the case of the name George Washington, we get a basic Western name pattern: `George!Washington`. For this pattern to be useful to nameauth, the internal macros add extra bits to it, which we can demonstrate various ways. Below we test manually whether or not various patterns relating to Washington, Tully, Elizabeth I, and Cato exist. After that, we shall unpack the meaning of these patterns and their affixes (`!MN, !NF, !PN, !PRE, !TAG, !DB`).

```
\newcommand\Defined[3]{\ifcsdef{#1}{#2}{#3}}

\begin{enumerate}[itemsep=0pt]
\item The main-matter name \Wash\
  \Defined{George!Washington!MN}{exists}{is undefined}.
\item The front-matter name \Wash\
  \Defined{George!Washington!NF}{exists}{is undefined}.
\item The index cross-reference \Name{Tully}
  \Defined{Tully!PN}{exists}{is undefined}.
\item The index sort tag for \Eliz\
  \Defined{Elizabeth,I!PRE}{is: \csuse{Elizabeth,I!PRE}}
  {is undefined.}
\item The index tag for \Wash\
  \Defined{George!Washington!TAG}{is: \csuse{George!Washington!TAG}}
  {is undefined.}
\item The name tag for \Wash\
  \Defined{George!Washington!DB}{is: \csuse{George!Washington!DB}}
  {is undefined.}
\item The name \CatoY\ is
  \Defined{MarcusPorcius!Cato,theYounger!PN}
  {excluded\csuse{MarcusPorcius!Cato,theYounger!PN}}
  {in another state.}
\end{enumerate}
```

1. The main-matter name Washington exists.
2. The front-matter name Washington is undefined.
3. The index cross-reference Tully exists.
4. The index sort tag for Elizabeth is: Elizabeth 1=
5. The index tag for Washington is: , pres.|hyperpage
6. The name tag for Washington is undefined.
7. The name Cato is excluded!

This demonstrates the basic functioning of nameauth. The base name pattern always has an affix that tells what kind of name or related element it is, and in what context it belongs. Basic name patterns correlate with consistent index entries; printed name forms may vary. Western name patterns have a bang and a comma; Non-Western name patterns have a comma or nothing.

Text	Basic Pattern	Macro / Arguments
Adolf von Harnack	`Adolf!Harnack`	`\Harnack[Adolf von]`
Adolf Harnack	`Adolf!Harnack`	`\LHarnack`
George S. Patton Jr.	`GeorgeS.!Patton,Jr.`	`\ForgetThis\Patton`
George S. Patton	`GeorgeS.!Patton,Jr.`	`\DropAffix\LPatton`
Hideyo Noguchi	`Hideyo!Noguchi`	`\ForgetThis\Noguchi`
Noguchi Hideyo†	`Hideyo!Noguchi`	`\RevName\LNoguchi\dag`
Yamamoto Isoroku	`Yamamoto,Isoroku`	`\ForgetThis\Yamt`
Isoroku Yamamoto	`Yamamoto,Isoroku`	`\RevName\LYamt`
Henry VIII	`Henry,VIII`	`\Name{Henry,VIII}`
Henry VIII‡	`Henry,VIII`	`\Name*{Henry}[VIII]\ddag`
Demetrius	`Demetrius,I`	`\Dem[I Soter]`
Demetrius I	`Demetrius,I`	`\LDem`
Aristotle	`Aristotle`	`\ForgetThis\Aris`

Six suffixes are added to name patterns to create six "systems" or data sets. Regarding their IDs, shown below, see also page 65. The `!MN` and `!NF` systems control name formatting. Macros in Section 12.1 use the `!PN` system with formatting. Indexing involves the `!PN`, `!PRE`, and `!TAG` systems. Name tags use the `!DB` system.

Description	ID	Pattern	Notes
Main-matter names	main	⟨*pattern*⟩`!MN`	expands to empty string
Front-matter names	front	⟨*pattern*⟩`!NF`	expands to empty string
Index cross-refs	xref	⟨*pattern*⟩`!PN`	expands to empty string
Exclusions	excl	⟨*pattern*⟩`!PN`	expands to exclusion value
Index sort tags	pretag	⟨*pattern*⟩`!PRE`	expands to tag value
Index tags	idxtag	⟨*pattern*⟩`!TAG`	expands to tag value
Name tags	namedb	⟨*pattern*⟩`!DB`	expands to tag value

Next we see how certain macros write patterns to systems, indicated by black boxes. `\IndexName` does not generate a name pattern; it only creates an index page entry, depending on the cross-references that control the index logic.

Macros	!NF	!MN	!PN	!PRE	!TAG	!DB
`\Name \Name* \FName \FName*`	■	■				
`\ForgetName \SubvertName`	■	■				
`\PName \PName*`	■	■	■			
`\AKA \AKA* \IndexRef`			■			
`\ExcludeName`			■			
`\IncludeName \IncludeName*`			■			
`\IndexName`						
`\PretagName`				■		
`\TagName \UntagName`					■	
`\NameAddInfo \NameClearInfo`						■

6.3 Name Uniqueness

We disable indexing for the examples below. Grouping tokens `{}` do not appear in the text, but they do appear in the name patterns. The same goes for non-printing macros, such as `\empty`. Thus, all names below are unique, as the "first-use" color shows. Even if they look alike in the text, they have different name patterns and different index entries. See also Sections 6.2, 6.5, and 7.6.

1. one two `\Name[one]{two}`
   Pattern: `one!two` ..... Western name
2. one two `\Name[{one}]{{two}}`[29]
   Pattern: `one!{two}` ..... Western name
3. one two `\Name[one]{\empty two}`
   Pattern: `one!\emptytwo` ..... Western name
4. one two `\Name{one, two}`
   Pattern: `one,two` .....Non-Western name
5. one two `\Name{{one}, {two}}`
   Pattern: `{one},{two}` .....Non-Western name
6. one, two `\Name{{one, two}}`
   Pattern: `{one,two}` ..... (Malformed ⟨*SNN*⟩)

> In ⟨*SNN,Affix*⟩ apply macros separately to ⟨*SNN*⟩ and ⟨*Affix*⟩.

There are several ways to make different names out of otherwise identical names. They vary in complexity. Among the methods we have:

- Index tags (`\TagName`, Section 7.7.2) can create different index entries for otherwise identical names. This requires keeping track of the context and where one changes those tags.
- Non-printing tokens and macros in name arguments help one to assign unique patterns for similar index and name tags (Sections 7.7.2 and 8). This requires one to sort names with `\PretagName` (Section 7.6).
- If such macros are at the leading edge of a name instead of the trailing edge, or if any macros might expand differently at different times, one may need to use `\noexpand` and alternate formatting (Section 6.5).

> Address any concerns about expansion of macros in name arguments by using `\noexpand` before those macros.

[29] The grouping tokens for the mandatory argument remain, while those of the optional argument go away. This ia a function of the xparse package. If one plugs the arguments directly into `\@nameauth@LoadArgs` and `\@nameauth@Choice`, they are not expanded unevenly. Still, the benefits of xparse outweigh the downsides. The takeaway here is to check the name pattern when debugging.

6.4 Indexing States

Six distinct "states" describe any name pattern with respect to page entries and cross-references in the index. While it may seem odd to discuss this here, this information relates directly to `\ShowNameState`, and these states are not reflected **as such** by the indexing macros. They are derived from a matrix of Boolean flags and control sequences, serving more as a useful debugging mnemonic for package users. Below we show what these states mean.

State 1: No Name Information Present

• `\IndexName` makes index page entry, creates no name pattern control sequence	Stay in State 1
• `\TagName` makes index tag, creates a pattern ending in `!TAG`; `\UntagName` destroys it	Stay in State 1
• `\PretagName` makes index sort tag, creates a pattern ending in `!PRE` (do only once)	Stay in State 1
• `\ForgetName` is redundant; it cannot destroy a control sequence that does not exist.	Stay in State 1
• Naming macro makes name pattern (`!MN` or `!NF`), prints name, makes two index page entries	Go to State 2
• `\SubvertName` makes a name pattern ending either in `!MN` or in `!NF`, or both at once	Go to State 2
• `\IndexRef` makes an xref pattern ending with `!PN`; that pattern expands to empty	Go to State 3
• `\SeeAlso\IndexRef` makes an xref pattern ending with `!PN`; that pattern expands to empty	Go to State 3
• `\ExcludeName` makes an xref pattern ending with `!PN`; that pattern expands to `!x!`	Go to State 5

State 2: Only a Name Pattern Exists

• `\IndexName` makes index page entry, creates no name pattern control sequence	Stay in State 2
• `\TagName` makes index tag, creates a pattern ending in `!TAG`; `\UntagName` destroys it	Stay in State 2
• `\PretagName` is doable, but not recommended (that will create spurious entries)	Stay in State 2
• Naming macro makes name pattern (`!MN` or `!NF`), prints name, makes two index page entries	Stay in State 2
• `\SubvertName` is redundant; it cannot create a control sequence that already exists.	Stay in State 2
• `\IndexRef` by itself does nothing because a name pattern already exists	Stay in State 2
• `\ForgetName` destroys a name pattern ending in `!MN`, `!NF`, or both at once	Go to State 1
• `\SeeAlso\IndexRef` makes an xref pattern ending with `!PN`; that pattern expands to empty	Go to State 4
• `\ExcludeName` makes an xref pattern ending with `!PN`, that pattern expands to `!x!`	Go to State 6

State 3: Only an Xref Pattern Exists

• `\PretagName` is doable, but not recommended (that will create spurious entries)	Stay in State 3
• `\TagName`, `\UntagName`, `\IndexName`, `\IndexRef` (also with `\SeeAlso`), `\ExcludeName`, and `\IncludeName` do nothing	Stay in State 3
• `\ForgetName` is redundant; it cannot destroy a control sequence that does not exist.	Stay in State 3
• `\Includename*` destroys an extant xref pattern (`!PN`)	Go to State 1
• Naming macro makes name pattern (`!MN` or `!NF`), prints name, makes no index entries	Go to State 4
• `\SubvertName` creates a name pattern ending either in `!MN` or in `!NF`, or both at once	Go to State 4

State 4: Both Name and Xref Patterns Exist

• `\PretagName` is doable, but not recommended (that will create spurious entries)	Stay in State 4
• `\TagName`, `\UntagName`, `\IndexName`, `\IndexRef` (also with `\SeeAlso`), `\ExcludeName`, and `\IncludeName` do nothing	Stay in State 4
• Naming macro makes name pattern (`!MN` or `!NF`), prints name, makes no index entries	Stay in State 4
• `\SubvertName` is redundant; it cannot create a control sequence that already exists.	Stay in State 4
• `\Includename*` destroys an extant xref pattern (`!PN`)	Go to State 2
• `\ForgetName` destroys a name pattern ending in `!MN`, `!NF`, or both at once	Go to State 3

State 5: Only an Exclusion Exists

• `\PretagName` is doable, but not recommended (that will create spurious entries)	Stay in State 5
• `\TagName`, `\UntagName`, `\IndexName`, `\IndexRef` (also with `\SeeAlso`), and `\ExcludeName` do nothing	Stay in State 5
• `\ForgetName` is redundant; it cannot destroy a control sequence that does not exist.	Stay in State 5
• `\Includename`, `\Includename*` destroy xref pattern ending with `!PN`, expanding to `!x!`	Go to State 1
• Naming macro makes name pattern (`!MN` or `!NF`), prints name, makes no index entries	Go to State 6
• `\SubvertName` creates a name pattern ending either in `!MN` or in `!NF`, or both at once	Go to State 6

State 6: Both Name Pattern and Exclusion Exist

• `\PretagName` is doable, but not recommended (that will create spurious entries)	Stay in State 6
• `\TagName`, `\UntagName`, `\IndexName`, `\IndexRef` (also with `\SeeAlso`), and `\ExcludeName` do nothing	Stay in State 6
• Naming macro makes name pattern (`!MN` or `!NF`), prints name, makes no index entries	Stay in State 6
• `\Includename`, `\Includename*` destroy xref pattern ending with `!PN`, expanding to `!x!`	Go to State 2
• `\ForgetName` destroys a name pattern ending in `!MN`, `!NF`, or both at once	Go to State 5

6.5 Using `\noexpand`

The natural-language approach of nameauth, combined with requirements of indexing, requires strict control over macro expansion. We use `\noexpand` in name arguments for the following reasons:

- A name uses the hyphenation of a foreign language via macros in name arguments.
- Name elements change within the local scope of formatting hooks, but remain uniform globally when generating index entries. This is the crux of alternate formatting (Section 11.2).
- A name specifically uses the alternate form of `\CapThis` via alternate formatting (Section 11.2.3).
- There is a possibility that macros in name arguments might expand unpredictably in different situations, resulting in spurious index entries and name pattern tests failing in some cases.

If a macro is undefined, even putting `\noexpand` before it will cause an error unless the macro is detokenized or verbatim. To avoid some errors, one can apply macros separately to ⟨*SNN*⟩ and ⟨*Affix*⟩:

`\Name{\noexpand\MyMacro{`⟨*SNN*⟩`},\noexpand\MyMacro{`⟨*Affix*⟩`}}`

> Die Quantenmechanik ist sehr achtunggebietend. Aber eine innere Stimme sagt mir, daß das noch nicht der wahre Jakob ist. Die Theorie liefert viel, aber dem Geheimnis des Alten bringt sie uns kaum näher. Jedenfalls bin ich überzeugt, daß der nicht würfelt.
>
> (Quantum mechanics is quite significant. But an inner voice tells me that it is not yet the right [path]. The theory delivers much, but it brings us hardly any closer to the secret of God. In any case, I am convinced that He does not throw dice.)
>
> —Albert Einstein
> Letter to Max Born, 4 December 1928

6.6 Debugging Macros

These macros never grant users access to the full internal name patterns used by nameauth, but they do query those patterns. These macros help users identify how their intent to use the naming and other macros correlates with what the macros are seeing as input and how they are functioning.

\ShowPattern

Although \NameauthPattern is useful in hooks when you want the pattern of the last name evaluated, \ShowPattern offers both that function, when called with a null argument, as well as the ability to display any pattern according to the name arguments provided to it:

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

- Let us refer to the name Æthelred II. \ShowPattern{} now gives us the pattern: ÃĘthelred,II. \NameauthPattern also: ÃĘthelred,II.[30]
- \ShowPattern[Hernando]{de~Soto} prints: Hernando!de~Soto. Since the name arguments were evaluated, now \ShowPattern{} gives us the pattern: Hernando!de~Soto.
- Section 1.6.1 lists all the macros that change the current name pattern.

\ShowIdxPageref

\ShowIdxPageref shows how macro arguments generate an index page reference, regardless of whether or not that page reference or cross-reference actually exists. Internally, it puts the indexing mechanism into a state that will print a full page reference in the text.

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

Index styles, \PretagName, and \TagName affect the output of \ShowIdxPageref. Active characters and macros appear as they would be printed in the text, not as they may be represented in the idx file. One should check that too when debugging.

In a normal LaTeX document, for example, if we mention Hernando de Soto after setting up a sort tag (Section 7.6), we get the following:

Desoto, Hernando@de Soto, Hernando

```
\PretagName[Hernando]{de~Soto}{Desoto, Hernando}
\ShowIdxPageref[Hernando]{de~Soto}
```

A dtx file uses a different default for \IndexActual (Section 7.6). Since we tag all names in this dtx file with \TagName to have hyperlinked index page references (Section 7.7), we instead have here:

Desoto, Hernando=de Soto, Hernando|hyperpage

```
\ShowIdxPageref[Hernando]{de~Soto}
```

[30] This again is due to using inputenc and fontenc.

\ShowIdxPageref*

Throughout this manual, \ShowIdxPageref* illustrates basic index entries that do not contain sorting information or tags. The syntax is:

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

Regardless of whether we have a normal LaTeX document or a dtx file, we get:

```
\ShowIdxPageref*[Hernando]{de~Soto}
```
de Soto, Hernando

\ShowNameInfo

Here we can see how the macros that take name arguments interpret them. We can check our intent against what the package actually sees as input. When we use a null argument, \ShowNameInfo uses the token registers that contain the arguments of the last name arguments that were evaluated.

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

Next we use \ShowNameInfo to illustrate detokenized name arguments from a number of names. This approach both prevents potential errors and discloses macros in name arguments. The "Affix*" and "Alt*" designations have an asterisk because, while those are the syntactic terms that describe name macro arguments, those terms do not describe the exact cultural meaning of their contents.

- James Earl Carter Jr. . . . . \Name*[J.E.]{Carter, Jr.}[James Earl]
  (FNN: J.E.) (SNN: Carter) (Affix: Jr.) (Alt: James Earl)
- Miyazaki Sensei . . . . . \Name*{Miyazaki, Hayao}[Sensei]
  (SNN: Miyazaki) (Affix*: Hayao) (Alt: Sensei)
- Elizabeth I, "Gloriana" . . . . . \Name*{Elizabeth, I}[I, ``Gloriana'']
  (SNN: Elizabeth) (Affix*: I) (Alt: I, "Gloriana")
- Henry VIII‡ . . . . . \Name*{Henry}[VIII]\ddag
  (SNN: Henry) (Alt*: VIII)
- Confucius . . . . . \Name*{Confucius}
  (SNN: Confucius)

> Address any concerns about expansion of macros in name arguments by using \noexpand before those macros.

When using macros in name arguments, \ShowNameInfo may give different results, depending on the use of \noexpand and the name interface. Macros in name arguments may be displayed with trailing spaces because nameauth uses \protected@edef when parsing name elements. We show how to mitigate these issues. In the following name examples, we assume that the \de macro was defined to use babel, as in Section 5.2.

- The macro in these arguments expands, which can change name patterns.

  (FNN: John) (SNN: \protect \foreignlanguage {ngerman}{Strietelmeier})
  `\ShowNameInfo[John]{\de{Strietelmeier}}`
- The macro in these arguments stays consistent due to `\noexpand`.

  (FNN: John) (SNN: \de {Strietelmeier})
  `\ShowNameInfo[John]{\noexpand\de{Strietelmeier}}`

```
\begin{nameauth}
  \< Striet  & John & \noexpand\de{Strietelmeier} & >
\end{nameauth}
\NameAddInfo[John]{\noexpand\de{Strietelmeier}}
  {professor}
```

- John Strietelmeier ..... `\Striet`
  With the first use of the name, we note the info.
  - professor ..... `\NameQueryInfo{}`
  - John!\noexpand\de{Strietelmeier} ..... `\ShowPattern{}`
  - (FNN: John) (SNN: \de {Strietelmeier}) .. `\ShowNameInfo{}`
- Strietelmeier ..... `\Striet`
  The name is repeated; the same pattern recurs.
  - professor ..... `\NameQueryInfo{}`
  - John!\noexpand\de{Strietelmeier} ..... `\ShowPattern{}`
  - (FNN: John) (SNN: \de {Strietelmeier}) .. `\ShowNameInfo{}`
- Strietelmeier ..... `\Name[John]{\noexpand\de{Strietelmeier}}`
  We use the same name again, but with the basic interface. The name pattern still works, but `\ShowNameInfo` expands the arguments after pulling them from the internal token registers. We no longer refer to these registers for use in customization in order to avoid errors.
  - professor ..... `\NameQueryInfo{}`
  - John!\noexpand\de{Strietelmeier} ..... `\ShowPattern{}`
  - (FNN: John) (SNN: \protect \foreignlanguage {ngerman}{Strietelmeier})
    `\ShowNameInfo{}`
- Strietelmeier ..... `\Name[John]{\noexpand\de{Strietelmeier}}`
  This expansion is hidden when using full name arguments everywhere.
  - professor
    `\NameQueryInfo[John]{\noexpand\de{Strietelmeier}}`
  - John!\noexpand\de{Strietelmeier}
    `\ShowPattern[John]{\noexpand\de{Strietelmeier}}`
  - (FNN: John) (SNN: \de {Strietelmeier})
    `\ShowNameInfo[John]{\noexpand\de{Strietelmeier}}`

Thus, we rely on `\NameauthPattern` for name-related information instead of re-parsing name arguments. That is how recent versions of `\NameQueryInfo` work, simplifying the use of name tags in formatting hooks (Sections 8 and 8.2).

Below we activate alternate formatting (Section 11.2) and use name forms seen in European academic writing.

- Catherine de' Medici

  (FNN: Catherine \AltCaps {d}e') (SNN: \textSC {Medici})
- Martin Luther

  (FNN: Martin) (SNN: \textSC {Luther})

`\ShowNameState`
`\ShowNameState*`

With these macros we can see in plain text the name pattern, the name type, the current index state of the name, and all the systems with which its pattern is associated. The un-starred form prints the information in one long line of text. The starred form prints five shorter lines of text, each ending with `\par`. As with other macros in this section, these macros can have null arguments.

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

Here is the key to understanding the output of these macros:

Pattern: The base name pattern shown by `\NameauthPattern`.

Type: The category of the name; how the name arguments were parsed:

w :	Western name
nw :	Non-Western name, current syntax
nw,os :	Non-Western name, obsolete syntax (Section 12.2)

State: The index state, as defined in Section 6.4:

states : 1, 2, 3, 4, 5, and 6

Systems: All the control systems (Section 6.2) that the name pattern inhabits, including front matter, main matter, cross-references, excluded cross-references, index sorting tags, index tags, and name database tags:

systems : front, main, xref, excl, pretag, idxtag, namedb.

We begin by showing info about some unused names that will not appear in the index. We add margin notes using `\ShowNameState*`, while using `\ShowNameState` in the body text.

Pattern: Hieronymus!Bosch
Type: w
State: 1
Systems: none

Pattern: Tamerlane
Type: nw
State: 1
Systems: none

- `\ShowNameState[Hieronymus]{Bosch}`
  Pattern: Hieronymus!Bosch Type: w State: 1 Systems: none
- `\ShowNameState{Tamerlane}`
  Pattern: Tamerlane Type: nw State: 1 Systems: none

Pattern: Rudolph!Carnap
Type: w
State: 2
Systems: main idxtag

Now we display the info for new and extant names in the text.[31]

- Rudolph Carnap ..... `\Name*[Rudolph]{Carnap}`
  Pattern: Rudolph!Carnap Type: w State: 2 Systems: main idxtag

Pattern: Henry,VIII
Type: nw
State: 2
Systems: main idxtag

- Henry VIII ..... `\Name*{Henry, VIII}`
  Pattern: Henry,VIII Type: nw State: 2 Systems: main idxtag

Pattern: Henry,VIII
Type: nw,os
State: 2
Systems: main front idxtag

- Henry VIII‡ ..... `\NamesInactive \Name*{Henry}[VIII]\ddag \NamesActive`
  Pattern: Henry,VIII Type: nw,os State: 2 Systems: main front idxtag

Pattern: Thomas,Ãă~Kempis
Type: nw
State: 2
Systems: main pretag idxtag

- Thomas à Kempis ..... `\Name*{Thomas, à~Kempis}`
  Pattern: Thomas,Ãă~Kempis Type: nw State: 2 Systems: main pretag idxtag[32]

Here is a name used only in a cross-reference:[33]

Pattern: SunKing
Type: nw
State: 3
Systems: xref

- `\IndexRef{Sun King}{Louis XIV}`
  Pattern: SunKing Type: nw State: 3 Systems: xref

Next is a name displayed after a cross-reference exists:[34]

Pattern: SunKing
Type: nw
State: 4
Systems: main xref

- Sun King ..... `\Name*{Sun King}`
  Pattern: SunKing Type: nw State: 4 Systems: main xref

The following is an unused name excluded from the index:[35]

Pattern: Santa,Claus
Type: nw
State: 1
Systems: none excl

- `\ExcludeName{Santa, Claus}`
  Pattern: Santa,Claus Type: nw State: 1 Systems: none excl

Finally we have a previously excluded name used in the text:[36]

Pattern: Santa,Claus
Type: nw
State: 6
Systems: main excl

- Santa Claus ..... `\Name{Santa, Claus}`
  Pattern: Santa,Claus Type: nw State: 6 Systems: main excl

Back to Table of Contents

> I read in a book once that a rose by any other name would smell as sweet, but I've never been able to believe it. I don't believe a rose WOULD be as nice if it was called a thistle or a skunk cabbage.
>
> —L.M. Montgomery
> *Anne of Green Gables*, C 5 (1908)

[31] In a `dtx` file, with the ltxdoc class and hypdoc package, we tag all names in the index with hyperlinks (Section 7.7.3). Otherwise, names need not have a pattern in the `idxtag` system.
[32] This again is due to using inputenc and fontenc.
[33] Here we should populate the name arguments because the name has not appeared in the text.
[34] If the name is printed and `\ShowNameState` follows immediately, we can use a null argument.
[35] Here we should populate the name arguments because the name has not appeared in the text.
[36] If the name is printed and `\ShowNameState` follows immediately, we can use a null argument.

7 Indexing Macros

This section draws on what we have seen so far and applies that knowledge to creating indexes of names via the nameauth package macros, not indexing in general.

7.1 General Control

Before we get to the topics of index entries and cross-references, we cover those macros that globally affect the indexing of names.

7.1.1 Toggle Indexing

`\IndexInactive` `\IndexActive`

`\IndexInactive` deactivates the indexing functions of the naming macros, as well as `\IndexName`, and `\IndexRef`. `\IndexActive` enables indexing. These can be used throughout the document. They do not affect indexing apart from nameauth names.

- `\IndexInactive` broadly suppresses `\IndexName`, `\IndexRef`, and the indexing components of naming macros, `\AKA`, and `\PName`.
- For a fine degree of control, see Section 7.3.2.

`\global`

`\IndexActive` and `\IndexInactive` can be used as a pair or singly within a group. They have top priority (Section 3). Use `\global` to force a global effect.

7.1.2 Multiple Indexes

`\NameauthIndex`

LaTeX has various ways to produce multiple indexes; see **this page**. `\NameauthIndex` is the indexing hook defined by default as `\index`. Users can redefine this hook for use with multiple indexes. Below we use the index package to do this.

Here we test indexing rules:
Yamaha Torakusu
create index entry:
`\IndexName{Nippon Gakki}`

```
\documentclass{article}
\input{compat.tex} % Included with nameauth for compatibility.

\usepackage{makeidx} % Must have for defining \seealso macro.
\usepackage{index}
\usepackage{nameauth}

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

\begin{document}
  The Electric Boogaloo\index{Boogaloo, Electric} %  main index
  was created by \Name{Ollie~\& Jerry}.           %  name index

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

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

7.1.3 Verbose Warnings

\IndexWarnVerbose
\IndexWarnTerse

As with many of the other options in nameauth, we have added two macros that toggle verbose index warnings like the `verbose` option (default is terse). To disable verbose warnings, use `\IndexWarnTerse`. To enable verbose warnings, use the `verbose` option or `\IndexWarnVerbose`.

\global

`\IndexWarnVerbose` and `\IndexWarnTerse` can be used as a pair or singly within a group. They have the same priority as the `verbose` option, but they do not affect what is displayed in the document. Use `\global` to force a global effect.

7.1.4 Index Protection

\IndexProtect

This macro prevents naming macros from producing output, both in the text and in the index. It is local in scope. Its primary use is to prevent errors in the index, in case the naming macros get passed as arguments to themselves or passed into index entries by mistake. The core naming engine uses internal locks to protect against this problem in the text. One can use `\IndexProtect` right before `\printindex` to protect the index against bogus output. For example:

Here we test indexing rules: Yamaha

- `\Name{foo\Name{bar}}` in the text generates foo. Notice that the internal locks prevent `\Name{bar}` from producing output in the text.
- The first LaTeX pass creates `\indexentry{foo\Name {bar}}` in the `idx` file. With enough passes of LaTeX and `makeindex`, in the `ind` file we get both `\item foo\Name {bar}` from the text and an additional `\item bar` from the macro executed in the index.
- That gives one index entry "foobar" and another entry "bar".
- Using `\IndexProtect \printindex` still permits the index entry generated by `\item foo\Name {bar}`, but it does not allow `\Name {bar}` to generate any output or additional entries in the index.
- We get only one entry "foo", similar to what we see in the text.[37]

7.2 Page Entries

\IndexName

The internal version of this macro is used also by the naming macros. This is the user interface to create index page entries in the same way that the naming macros do. `\IndexName` 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 12.2). Indexing follows guidelines in [Mulvany, 152–82]. The following points also deal with macros explained in Section 7.4.

- `\IndexName` obeys both `\IndexInactive` and `\IndexActive`, which are used to deactivate and activate indexing.

[37] We add the index tag § to this example to show that it is a made-up name.

- `\IndexName` does not obey `\SkipIndex`. The latter only works with macros that display a name in the text, which `\IndexName` does not.
- `\IndexName` will not make page entries for any names that are excluded by `\ExcludeName`. Nor will it make entries names that have been used to create cross-references.
- `\IndexName` resets the effects of both `\SeeAlso` and `\SkipIndex` unless one uses the `oldreset` option.
- Section 7.4 shows behavior among `\SkipIndex`, `\JustIndex`, and the naming macros that differs from the same macros and `\IndexName`.

7.3 Cross-References

Index cross-references have two kinds. *See* references only point from a name to other name entries containing page entries. *See also* references occur at the end of an entry with page entries or sub-entries.

7.3.1 Basic Macros for Both Kinds of Xrefs

`\IndexRef` By default, `\IndexRef` creates a *see* reference from the name defined by its first three arguments to the target in its final argument. To create a *see also* reference, one must precede it with `\SeeAlso` (Section 7.4). Thus, the two forms used are:

```
\IndexRef[⟨FNN⟩]{⟨SNN,Affix⟩}[⟨Alt.⟩]{⟨Target⟩}
\SeeAlso\IndexRef[⟨FNN⟩]{⟨SNN,Affix⟩}[⟨Alt.⟩]{⟨Target⟩}
```

Although it might be redundant to point this out, in practice, when using `\IndexName` and `\IndexRef`, one might forget that the latter has four arguments, at least two of which are required. Missing text and bad xrefs can result.

Here we test indexing rules:
Creating *see also* xref
`\SeeAlso\IndexRef`
`{Yamaha, Torakusu}`
`{Nippon Gakki}`

- Define *see* references **before** making any `\Name` entries for them.
- Define *see also* references **after** all `\Name` instances to the respective names have been created.
- `\IndexRef` will not alter or repeat extant cross-references.
- `\IndexRef` will not cross-reference names excluded by `\ExcludeName`.
- `\IndexRef` will not add a *see* cross-reference that would map also to an extant name pattern, unless one uses the `oldsee` option. See Section 6.4 for a state diagram that explains this in detail. A page entry usually correlates with a name pattern, but that may not always be the case. We check for a control sequence (name pattern).
- `\IndexName` resets the effects of both `\SeeAlso` and `\SkipIndex`, unless one uses the `oldreset` option.
- To have multiple names and cross-references interact, see Section 10.1.4.
- To manage multiple sub-entries, see Section 7.8.

`\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

`\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 the following example, or create manual index entries:

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

7.3.2 Fine Control of Xref Logic

The index control logic for cross-references can establish fine-grained control that excludes individual names. We trade more macros for more automation.

`\ExcludeName`

We can prevent a name from being used as either an index entry or as an index cross-reference. This macro will not exclude extant cross-references:

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

Unlike `\IndexInactive` and `\IndexActive`, which affect all nameauth indexing, `\ExcludeName` only excludes a specific name from being printed as a page entry or cross-reference in the index. See the following example, as well as examples in Sections 5.7 and 10.1. Indexing remains active:

Name Pattern(s):
`Mr.Baseball!PN`
`Bob!Uecker!MN`

```
1 \ExcludeName{Mr. Baseball}
2 In this example we cannot get page entries for
3 \Name{Mr. Baseball}, the nickname of
4 \Name[Bob]{Uecker}, because it was excluded.
```

In this example we cannot get page entries for Mr. Baseball, the nickname of Bob Uecker, because it was excluded.

`\IncludeName`
`\IncludeName*`

Use the following macros to break a few indexing rules. They remove 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` removes an exclusion attached to a name by `\ExcludeName`. `\IncludeName*` completely erases both exclusion and cross-reference information. Once that protection is removed, one can create page entries to a name in the index that had been used as a cross-reference.

Analogously, `\ForgetName` (Section 9.3) removes name patterns, allowing one to create a cross-reference. Section 6.4 explains this behavior as a set of states. Below we run some tests (cf. Section 9.5).

Here we test indexing rules:
Yamaha

```
\begin{itemize}
\item \Name*{Mr. Baseball} is
  \IfAKA{Mr. Baseball}
    {\meta{an xref}}{\meta{a name}}{\meta{excluded}}.

\item Making an xref fails.\IndexRef{Mr. Baseball}{Uecker, Bob}
  \Name*{Mr. Baseball} is still
  \IfAKA{Mr. Baseball}
   {\meta{an xref}}{\meta{a name}}{\meta{excluded}}.

\item The inclusion macro\IncludeName{Mr. Baseball} makes it
  \IfAKA{Mr. Baseball}
    {\meta{an xref}}{\meta{a name}}{\meta{excluded}}.
  Now we could create page entries with a naming macro.

\item Instead, we forget the name\ForgetName{Mr. Baseball}
  to destroy the name pattern that governs the name. It is now
  \IfAKA{Mr. Baseball}
    {\meta{an xref}}
    {\IfMainName{Mr. Baseball}{\meta{extant}}
      {\IfFrontName{Mr. Baseball}{\meta{extant}}
        {\meta{destroyed}}}}
    {\meta{excluded}}.

\item Making another xref\IndexRef{Mr. Baseball}{Uecker, Bob}
  creates \IfAKA{Mr. Baseball}
    {\meta{an xref}}{\meta{a name}}{\meta{excluded}}.
\end{itemize}
```

- Mr. Baseball is ⟨*excluded*⟩.
- Making an xref fails. Mr. Baseball is still ⟨*excluded*⟩.
- The inclusion macro makes it ⟨*a name*⟩. Now we could create page entries with a naming macro.
- Instead, we forget the name to destroy the name pattern that governs the name. It is now ⟨*destroyed*⟩.
- Making another xref creates ⟨*an xref*⟩.

Cross-references get more protection than exclusions:

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

```
\begin{itemize}
\item \DropAffix\LJRIV[Jay] was indexed as
``\ShowIdxPageref*[J.D.]{Rockefeller, IV}.''

\item We create the
  xref\IndexRef[Jay]{Rockefeller}{Rockefeller, J.D., IV}.

\item The test calls \SJRIV[Jay] an
  \IfAKA[Jay]{Rockefeller}{\meta{xref}}{\meta{name}}{}.

\item After being ``included''\IncludeName[Jay]{Rockefeller}
  he still is an
  \IfAKA[Jay]{Rockefeller}{\meta{xref}}{\meta{name}}{}.

```

```
\item After ``forced inclusion''\IncludeName*[Jay]{Rockefeller}
  he can be a \IfAKA[Jay]{Rockefeller}{\meta{xref}}{\meta{name}}{}
  and create page entries.
\end{itemize}
```

- Jay Rockefeller was indexed as "Rockefeller, J.D., IV."
- We create the xref.
- The test calls Jay an ⟨*xref*⟩.
- After being "included" he still is an ⟨*xref*⟩.
- After "forced inclusion" he can be a ⟨*name*⟩ and create page entries.

> Using `\IncludeName*` is necessary when creating index sub-entries for a name using `\IndexTag`. If one creates a cross-reference in any sub-entry of a name, `\IncludeName*` will permit additional page entries to be made for the other sub-entries of that name or for the name itself. See Section 7.8.

7.4 Prefix Macros Used for Indexing

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

`\SeeAlso` Put `\SeeAlso` before `\IndexRef`, `\AKA`, and `\PName` (Section 12.1) 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.

Here we test indexing rules: Nippon Gakki

```
One can refer to \Name[the]{Rat Pack} as a group of entertainers
including \Name[Sammy]{Davis, Jr.}, \Name[Dean]{Martin}, and
\Name[Frank]{Sinatra}. No more page entries for
\Name*[the]{Rat Pack} will occur after this line, and a
\textit{see also} xref will exist.
\SeeAlso\IndexRef[the]{Rat Pack}
{Davis, Sammy, Jr.; Martin, Dean; Sinatra, Frank}
```

> One can refer to the Rat Pack as a group of entertainers including Sammy Davis Jr., Dean Martin, and Frank Sinatra. No more page entries for the Rat Pack will occur after this line, and a *see also* xref will exist.

Currently `\IndexName` and other nameauth macros that create index entries will reset the Boolean flag governed by `\SeeAlso` unless one uses the `oldreset` option. preventing a stray use of the macro from affecting the index.

`\SkipIndex` The prefix macro `\SkipIndex` will suppress indexing for just one instance of a name displayed by a naming macro. `\SkipIndex\Name[Monty]{Python}` produces Monty Python in the text, but with no index entry. `\SkipIndex` works with the naming macros. Side effects include:

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

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

`\JustIndex` This prefix macro makes `\Name`, `\Name*`, `\FName`, and the quick interface shorthand macros act similar to `\IndexName`. `\JustIndex` suppresses name output in the text, but flags for long and first name forms are reset as if the naming macro had produced output. Using the `oldreset` option prevents these flags from being reset, completely mimicking a call to `\IndexName`.

Option	Text	Macros
default	Washington	`\JustIndex\SWash \Wash`
default	Washington	`\JustIndex\LWash \Wash`
`oldpass`	George	`\JustIndex\SWash \Wash`
`oldpass`	George Washington	`\JustIndex\LWash \Wash`

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

- Both `\AKA` and `\PName` ignore `\JustIndex` and go on about their business. They also set `\@nameauth@JustIndexfalse`.
- `\JustIndex` resets the flags set by `\ForgetThis` and `\SubvertThis`, preventing them from passing through.
- The following three lines are equivalent:

  Here we test indexing rules: Creating *see* xref `\IndexRef{Nippon Gakki}{Yamaha Corp.}`

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

  `\JustIndex` takes priority with `\Name{A}` and passes `\SkipIndex` to `\Name{B}` (see also Section 3).
- `\JustIndex` and the naming macros do not replace `\IndexRef`.

7.5 Automatic Rules

Below we indicate what to expect regarding index rules. We do not attempt to concatenate page ranges. Previously we put names and cross-references in margin notes, indicating that we were testing the index rules. Here are the results:

Page 67: There are no name patterns (Section 6.2) for either a name or a cross-reference. Then we create the name Yamaha Torakusu along with its pattern. A pair of index page entries are generated, one before and one after the name. Also on page 67 an index page entry is created for Nippon Gakki, whose name pattern still does not yet exist.

Name Pattern(s): `Yamaha,Torakusu!MN`

Macros	Cumulative Index
`\Name{Yamaha, Torakusu}`	Nippon Gakki, 67
`\IndexName{Nippon Gakki}`	Yamaha Torakusu, 67

Page 68: The name Yamaha appears again. Since a pattern exists, the short form is printed and two index page entries are generated.

Name Pattern(s): `Yamaha,Torakusu!MN`

Macros	Cumulative Index
`\Name{Yamaha, Torakusu}`	Nippon Gakki, 67 Yamaha Torakusu, 67, 68

Page 69: We create a *see also* cross-reference from Yamaha Torakusu to the name Nippon Gakki. Now a cross-reference pattern exists for Mr. Yamaha. We can no longer create index page entries for Mr. Yamaha.

Name Pattern(s):
`Yamaha,Torakusu!MN`
`Yamaha,Torakusu!PN`

Macros	Cumulative Index
`\SeeAlso\IndexRef{Yamaha, Torakusu}{Nippon Gakki}`	Nippon Gakki, 67 Yamaha Torakusu, 67, 68 *see also* Nippon Gakki

Page 71: We attempt to make an index page entry to Mr. Yamaha by invoking his name. That attempt fails due to the extant xref.

Name Pattern(s):
`Yamaha,Torakusu!MN`
`Yamaha,Torakusu!PN`

Macros	Cumulative Index
`\Name{Yamaha, Torakusu}`	Nippon Gakki, 67 Yamaha Torakusu, 67, 68 *see also* Nippon Gakki

Page 72: We print the name Nippon Gakki, bringing its name pattern into being. Even though it was the **target** of an xref, that does not restrict the ability to make page entries for it.

Name Pattern(s):
`Yamaha,Torakusu!MN`
`Yamaha,Torakusu!PN`
`NipponGakki!MN`

Macros	Cumulative Index
`\Name{Nippon Gakki}`	Nippon Gakki, 67, 72 Yamaha Torakusu, 67, 68 *see also* Nippon Gakki

Page 73: We attempt to create a *see* cross-reference from Nippon Gakki to the target Yamaha Corp. That fails because, unlike a *see also* reference, a *see* reference cannot be created when a name pattern already exists. The nameauth package issues a warning at this point. Yet the name Yamaha Corp. is not created until page 74.

Name Pattern(s):
`Yamaha,Torakusu!MN`
`Yamaha,Torakusu!PN`
`NipponGakki!MN`

Macros	Cumulative Index
`\IndexRef{Nippon Gakki}{Yamaha Corp.}`	Nippon Gakki, 67, 72 Yamaha Torakusu, 67, 68 *see also* Nippon Gakki

Page 74: Index entries for Nippon Gakki and Yamaha Corp. also include this page due to the content of this list.

Name Pattern(s):
`Yamaha,Torakusu!MN`
`Yamaha,Torakusu!PN`
`NipponGakki!MN`
`YamahaCorp.!MN`

Macros	Cumulative Index
`\Name{Nippon Gakki}` `\Name{Yamaha Corp.}`	Nippon Gakki, 67, 72, 74 Yamaha Corp. 74 Yamaha Torakusu, 67, 68, *see also* Nippon Gakki

---

Censorship, in my opinion, is a stupid and shallow way of approaching the solution to any problem. Though sometimes necessary, as witness a professional and technical secret that may have a bearing upon the welfare and very safety of this country, we should be very careful in the way we apply it, because in censorship always lurks the very great danger of working to the disadvantage of the American nation.

—Dwight D. Eisenhower
Associated Press luncheon (24 April 1950)

7.6 Sorting Names in the Index

When using `makeindex`, names with characters outside the ASCII range `[A-Za-z]` need to be sorted, as do names with spaces, active characters, Roman numerals, and macros especially in the ⟨*SNN*⟩ argument. We do that in `nameauth` using sort tags. For other multilingual approaches, see the respective documentation for `xindex` and `xindy`. Those details go beyond the scope of this manual.

7.6.1 General Approach

`\IndexActual` Using `\index{`⟨*sort key*⟩`@`⟨*actual*⟩`}` works with both `makeindex` and `texindy`. The general practice for sorting with `makeindex -s` involves creating an `ist` file (pages 659–65 in *The Latex Companion*).

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 creating index entries with the naming macros. `\PretagName` does not care about the "actual" character, but it provides the information that is automatically added after that character.

`\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:[38]

```
\PretagName{Ægidius}{Aegidius}

In a \texttt{dtx} file the ``actual'' character is \texttt{=}.
\begingroup
  In a local scope we change to the normal character \texttt{@}
  and show the index entry:
  \texttt{\IndexActual{@}\ShowIdxPageref{Ægidius}}.
\endgroup
Now back to \texttt{dtx} mode: \texttt{\ShowIdxPageref{Ægidius}}.
```

Name Pattern(s): `ÃĘgidius!PRE`

In a `dtx` file the "actual" character is `=`. In a local scope we change to the normal character `@` and show the index entry: `Aegidius@Ægidius`. Now back to `dtx` mode: `Aegidius=Ægidius`.

`\PretagName` The `nameauth` package enables automatic index sorting using a "pretag" (cf. Section 6.2). 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*⟩`}`

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

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 throughout the document. For example:

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

[38] This example will generate a warning because we sort a name that does not exist.

Every reference to Æthelred II and W.E.B. Du Bois is automatically tagged and sorted.[39] One should "pretag" all names that contain active characters or macros unless one is using other multilingual tools like xindex and `xindy`.

We keep the next example simple to illustrate the concept. We do not use alternate formatting because we do not capitalize, mutate, or segment the alias "*Doctor angelicus*" (cf. Sections 5.8, 11.2).

Name patterns and index entries too large for margin.

```
1 \PretagName{\textit{Doctor angelicus}}{Doctor angelicus}
2 \IndexRef{\textit{Doctor angelicus}}{Thomas Aquinas}
3 \IndexRef[Thomas]{Aquinas}{Thomas Aquinas}
4
5 Perhaps the greatest medieval theologian was
6 \Name{Thomas, Aquinas}, later known as
7 \Name{\textit{Doctor angelicus}}.
```

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

We make *Doctor angelicus* a *see* references with no page entries. If we did want page entries and a *see also* reference from *Doctor angelicus* to Thomas, we would omit line 2 above, wait until the body text is complete at the the end of the document, such as before `\printindex`, and then use:

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

Name Particles

Spaces change sorting. For example, the sort tag `De␣Soto` precedes `deal` due to the space therein. The sort tag `DeSoto` falls as expected between `derp` and `determinism`. Remove spaces when sorting with `\PretagName`.

Collating Sequences

German ä ö ü ß map to English ae oe ue ss. Yet Norwegian æ ø å follow z in that order. One might have to create a sort tag that substitutes, e.g., za zb zc for æ ø å. Check a style guide regarding collating sequences, spaces, and sorting. This is where using other multilingual tools can be helpful.

Alternate Formatting

Additional examples starting with Section 11.2.5 deal with index sorting as it relates to alternate formatting, "Continental" practice, and macros in name arguments.

Antony: Friends, Romans, countrymen, lend me your ears;
I come to bury Caesar, not to praise him.

—William Shakespeare
"Julius Caesar", Act III, Scene II (first performed 1599)

[39] Regarding the margin note that shows name patterns, with `pdflatex` and `latex`, in `ÃĘthelred,II` the glyphs `ÃĘ` correspond to `\IeC{\AE}`.

7.6.2 Sorting Initials

Sorting J.D. Rockefeller IV (`\Name*[J.D.]{Rockefeller, IV}`) presents a problem that does not occur with Clive Staples Lewis (`\Name*[Clive Staples]{Lewis}`). In the case of Jay Rockefeller IV, the initials will appear in the index, while with C.S. Lewis, the longer names will appear in the index.

In nameauth we have a specific way to do that once per name, and all the remaining names will be sorted as expected. Before we use a name like Rockefeller, preferably in the preamble, we use the following macro:

```
\PretagName[J.D.]{Rockefeller, IV}{Rockefeller, John D 4}
```

- The index entry is "Rockefeller, J.D., IV".
- In the "pretag" we spell out the first forename and add enough of the second to get a unique sorting key.
- We turn the Roman numeral affix into an Arabic numeral so that it does not sort like letters.

For more examples, see Sections 5.8, 5.10, 10.1, and all of Section 11.

> In order to sort the index consistently and properly, all names should be sorted by their longer name forms, not their initials.

7.6.3 Unicode Compatibility

Here we engage how active Unicode characters affect index entries. Changes in the way that `pdflatex` and `latex` handle Unicode characters since 2018 have made indexing simpler and more intuitive, e.g.

Pre-2018		Index	Post-2018		Index
ä	→	`\IeC␣{\"a}`	ä	→	ä
æ	→	`\IeC␣{\ae␣}`	æ	→	æ

The `\IeC` macro plus its argument then would expand to `\T1` plus its argument, which would occur especially if accented characters were written out to a file, then read in again. This could cause problems that would generate multiple index entries. We can test for changes that can affect nameauth:

1. Test for the presence of `utf8-2018.def` and define a macro:

```
\makeatletter
  \@ifl@t@r\fmtversion{2018/10/05}{\newcommand\nameauthltx{}}{}
\makeatother
```

2. Test for the presence of `utf8-2018.def` by checking for a file:

```
\IfFileExists{utf8-2018.def}{do if present}{do if absent}
```

3. Test for the version of xparse that allows the current function of name arguments, and define a macro:

```
\makeatletter
  \@ifl@t@r\fmtversion{2018/04/30}{\newcommand\nameauthxp{}}{}
\makeatother
```

Before 2018, some index styles like `gind.ist` could not work with characters with macrons. Since 2018, those restrictions have been removed. To work with both old or new versions of `pdflatex` and `latex`, we can do the following:

```
\begin{nameauth}
\ifdefined\nameauthltx
  \< Ghazali & & Ghazāli & >
\else
  \< Ghazali & & Ghazali & >
\fi
\end{nameauth}

\ifdefined\nameauthltx
  \PretagName{Ghazāli}{Ghazali}
\fi

\Ghazali\ was an important Islamic philosopher.
```

Ghazāli was an important Islamic philosopher.

7.6.4 Debugging

If one sees multiple index entries for the same name, or if the names are not sorted correctly, either check the `idx` file or use the debugging macros (Section 6.6). Check nameauth package warnings. Set the `verbose` option, which will offer a number of "informational" warnings that could be of assistance.

- Do naming macros always use the same arguments?
- Are there any active characters, spaces, or control sequences in the name arguments? Use `\PretagName`.
- Are active Unicode characters expanding consistently?
- Use `\ShowPattern`, `\ShowIdxPageref`, and `\ShowNameState`.
- Did `\noexpand` precede macros in name arguments? Can those macros ultimately expand differently in the index? If so, check how alternate formatting is being used.
- Is alternate formatting used consistently? Are names used within sections of alternate formatting used outside of them?

Using `\protected@edef` in macros can add spaces to index entries. The nameauth macros must use `\protected@edef` to work with classes that write index entries to `aux` files. One must check this in the `idx` file. We show this below:

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

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

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

7.7 Index Tags

7.7.1 General Approach

\TagName
\UntagName

Index tags are information added automatically to entries created by nameauth macros. \TagName creates or changes a persistent tag. \UntagName deletes a tag. This information can include life dates, regnal dates, etc. Both \TagName and \UntagName have **global scope** and handle arguments in the same way as \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:

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

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 ␣the Great, pope to the index entry below (normal LaTeX document):

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

```
1 \PretagName{Gregory, I}{Gregory 1}
2 \TagName{Gregory, I}{ the Great, pope}
3 Pope \Name*{Gregory, I} is known as \Name*{Gregory, I}
4 [the Great].\\
5 Index: \ShowIdxPageref{Gregory, I}
```

Pope Gregory I is known as Gregory the Great.
Index: Gregory 1@Gregory I the Great, pope

See Section 5.9 for more ways to deal with ancient names. \TagName works with all names that produce index page entries. It does not work with with cross-references produced by \IndexRef, \AKA, etc. Tags can hold different kinds of information, but they should not be verbose. They can include 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.

> By what other voice, too, than that of the orator, is history, the evidence of time, the light of truth, the life of memory, the directress of life, the herald of antiquity, committed to immortality?
>
> —Marcus Tullius Cicero
> *De oratore* B II; C IX, §36 (55 BC)

7.7.2 Disambiguating Identical Names

We can format and index one name as two different people with `\TagName` and `\ForgetThis` (Section 9.3). The index tags group together their respective entries:

```
\TagName[Engelbert]{Humperdinck}{ (composer, 1854--1921)}
This refers to the classical German composer:
\Name[Engelbert]{Humperdinck}.

\TagName[Engelbert]{Humperdinck}{ (singer, *1936)}
This refers to the English singer from the 60s and 70s:
\ForgetThis\Name[Engelbert]{Humperdinck}.
```

This refers to the classical composer: Engelbert Humperdinck.

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

The name patterns are `Engelbert!Humperdinck!TAG` for the index tag and `Engelbert!Humperdinck!MN` for the name. This is the least "intrusive" way of creating unique names but it requires one to track potential changes to the text that might change the placement of index tags relative to their names.[40]

Sections 5.10 and 6.3 show how one can do similar things with non-printing macros in name arguments, where the names are sorted with `\PretagName` and also have index tags. This relieves one of having to keep track of otherwise identical names, because the name patterns are distinct, even if they look alike. One can go to greater lengths in Section 11 and its subsections.

One should try to use the nameauth macros in a way that is arguably the most "natural" and has the best fit for one's use case. That is a reason why this manual contains so many different examples. See also Section 1.7.

7.7.3 Special Tags for Special Cases

`\TagName` can create "special" index entries for names with the general form below. These tags are compatible with hyperref used in normal LaTeX documents.[41] When a ⟨*macro*⟩ that takes one or more arguments exists, one can use the form:

`\TagName[`⟨*FNN*⟩`]{`⟨*SNN,Affix*⟩`}[`⟨*Alternate*⟩`]{`⟨`|`*macro*⟩`}`

Hermogenes: I should explain to you, Socrates, that our friend Cratylus has been arguing about names; he says that they are natural and not conventional; not a portion of the human voice which men agree to use; but that there is a truth or correctness in them, which is the same for Hellenes as for barbarians.

—Plato
opening statement in *Cratylus* (c. 388 BC)

[40] The name decision macros might help automate this, as might the separate name systems.
[41] This was implemented in v.3.3, based on the answer of Heiko Oberdiek to **this question**.

When using the ltxdoc class with hypdoc, as in this manual, neither nameauth nor regular use of `\index` creates hyperlinked page entries. Index data tags in the `<driver>` section of the `dtx` file, which reads the "commented" part of the `dtx` file into a `document` environment, take the form:

`\TagName`⟨*name args*⟩`{|hyperpage}`

Within the "commented" part of the package documentation in this `dtx` file, the vertical bar is active. Hence, in this part, we use one of these two:

- `\TagName`⟨*name args*⟩`{\string|hyperpage}`
- `\index{`⟨*entry info*⟩`\noexpand\string|hyperpage}`

Internally, in `\@nameauth@IdxFormat`, when a cross-reference is being created, a tag of the form ⟨*some text*⟩`|`⟨*some macro*⟩ is reduced to ⟨*some text*⟩, allowing the macros `|see` and `|seealso` internally to be appended to the index entry even if a tag with a vertical bar exists.

Next we create a special tag. Since we used lines 1–2 in this `dtx` file, we put them in the driver section to keep it simple.

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

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

The microtype package and its tracking features is probably the best solution to fix index entries and sub-entries that break badly across columns or pages. Here we show how to add manual breaks, inelegant as that may be.

- Create a macro that takes an argument and appends a break:

  `\newcommand*{\EndBreak}[1]{#1\newpage}`
- The principle is similar to using:

  `\index{Doe, John|textbf}`
- The default for two column mode is `\newpage`. One might use `\newcolumn` with multicol and idxlayout, etc.

We use `\EndBreak` in an index tag for every index entry on the last page where such entries occur. If all instances of a name on that pagee have the same index tag, there will be no duplicate page entries, hyperlinks will work, and the index will break as indicated:

Page	Macro	Index
(preamble)	`\newcommand*{\EndBreak}[1]{#1\newpage}`	
10	`\Name{Some, Name}`	Some Name, 10
15	`\Name{Some, Name}`	Some Name, 10, 15
18	`\TagName{Some, Name|EndBreak}`	
	`\Name{Some, Name}`	Some Name, 10, 15, 18⟨*break*⟩

We do not have to supply an argument to `\EndBreak` because, as with font switching in the index, the page entry is implied. We also can intermix nameauth macros with manual index entries.[42] Instead of using `\TagName`, we could try this:

Page	Macro	Index
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. 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.

We do not recommend breaking an index entry in the middle. There are several discussions, such as **this page** and **that page**. Modifying `\EndBreak` above by putting `\@gobble` after `\newpage` would allow breaking an index entry in the middle, but there are no real guarantees that the new macro would work well.

7.8 Categories and Sub-entries

Indexes can have categories and sub-entries such as the following:

⟨*category 1*⟩

⟨*name entry*⟩

⟨*name sub-entry 1*⟩
⟨*name sub-entry 2*⟩...

⟨*name entry*⟩...

⟨*category 2*⟩...

To get nameauth to work with this structure, one uses `\PretagName` to organize names under categories. One uses `\TagName` to create sub-entrues under names. A simple mnemonic is: "Pre stands for C" when it comes to categories, while "Tags are on the inside" like sub-entries.

The following syntax box shows a couple of generic examples for creating categories and sorting names under them with `\PretagName`:

```
\PretagName[⟨FNN⟩]{⟨SNN⟩}{⟨category 1⟩!⟨SNN⟩, ⟨FNN⟩}
\PretagName{⟨SNN,Affix⟩}{⟨category 1⟩!⟨SNN⟩ ⟨Affix⟩}
```

Likewise, the following syntax box shows a couple of generic examples for creating sub-entries under names with `\TagName`:

```
\TagName[⟨FNN⟩]{⟨SNN⟩}{!⟨name sub-entry⟩}
\TagName{⟨SNN,Affix⟩}{!⟨name sub-entry⟩}
```

[42] One may need to look at the `idx` or `ind` files to craft matching entries on the page where the break occurs. Again, this approach is probably not as good as using microtype.

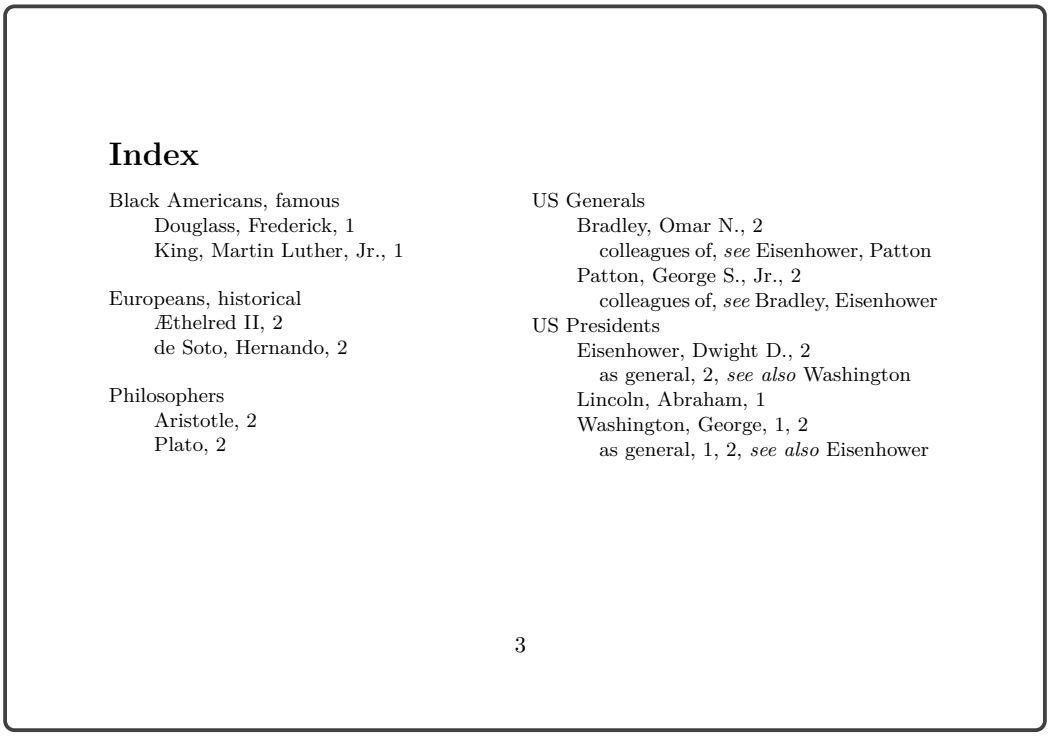
- If the name has no tag, `\UntagName` will restore the base name entry.
- `\TagName` can set up a default tag, set up or change sub-entries, and change back to the default tag.
- If a sub-entry of a name needs a *see* reference:
  - If the name has been used in the front matter or the main matter, use `\ForgetName` or no *see* reference will be created.
  - Pick sub-entry A with no page references.
    `\TagName[Some]{Name}{!Sub-entry A}`
  - Create a *see* reference using that sub-entry.
    `\IndexRef[Some]{Name}{Something else}`
  - Free up the name for more references.
    `\IncludeName*[Some]{Name}`
  - Change back to the base name entry or another sub-entry.
  - If needed, use `\SubvertName`.
- If a sub-entry of a name needs a *see also* reference:
  - Pick sub-entry B with complete page references.
    `\TagName[Some]{Name}{!Sub-entry B}`
  - Create a *see also* reference using that sub-entry.
    `\SeeAlso\IndexRef[Some]{Name}{Something else}`
  - Free up the name for more references.
    `\IncludeName*[Some]{Name}`
  - Change back to the base name emtry or another sub-entry.
- Have as few levels in an index as needed.

Next, we show categories and sub-entries used in a normal LaTeX document.

```
\documentclass{article}
\input{compat.tex} %  Included with nameauth for compatibility.
\usepackage[a6paper,landscape,left=1cm,right=1cm]{geometry}
\usepackage{makeidx}
\usepackage{nameauth}

\newcommand*{\EndBreak}[1]{#1\newpage}

\makeindex

%  Sort these names under: US Generals.
\PretagName[Omar N.]{Bradley}{US Generals!Bradley, Omar}
\PretagName[George S.]{Patton, Jr.}{US Generals!Patton, George}

%  Sort these names under: US Presidents.
\PretagName[George]{Washington}{US Presidents!Washington, George}
\PretagName[Dwight D.]{Eisenhower}{US Presidents!Eisenhower, Dwight D.}
\PretagName[Abraham]{Lincoln}{US Presidents!Lincoln, Abraham}

%  Sort these names under: Philosophers.
\PretagName{Aristotle}{Philosophers!Aristotle}
\PretagName{Plato}{Philosophers!Plato}

```

```
%  Sort these names under: Black Americans, famous.
\PretagName[Frederick]{Douglass}
  {Black Americans, famous!Douglass, Frederick}
\PretagName[Martin Luther]{King, Jr.}
  {Black Americans, famous!King, Martin Luther, Jr.}

%  Sort these names under: Europeans, historical.
\PretagName{\AE thelred, II}{Europeans, historical!Aethelred 2}
\PretagName[Hernando]{de Soto}
  {Europeans, historical!de Soto, Hernando}

\begin{nameauth}
  \< Wash & George & Washington & >
  \< Bradley & Omar N. & Bradley & >
  \< Aris & & Aristotle & >
  \< Plato & & Plato & >
  \< Aeth & & \AE thelred, II & >
  \< Linc & Abraham & Lincoln & >
  \< MLK & Martin Luther & King, Jr. & >
  \< Soto & Hernando & de Soto & >
  \< Patton & George S. & Patton, Jr. & >
  \< Ike & Dwight D. & Eisenhower & >
\end{nameauth}

\begin{document}
\small

\section{Famous Black Americans}

\Name[Frederick]{Douglass} rose to eminence by sheer force of
character and talents that neither slavery nor caste
proscription could crush. Circumstances made
\Name[Frederick]{Douglass} a slave, but they could not prevent
him from becoming a freeman and a leader among mankind.\\

We also celebrate \MLK, then \MLK.

\section{Patres Patriae}

We mention President \Wash; again, \Wash.
Family and close friends called him \SWash.\\

\TagName[George]{Washington}{!as general}
We can reminisce about \LWash[General].
\UntagName[George]{Washington}

When speaking of \Linc, we can refer to \LLinc[Abe].

\section{Philosophers}

\TagName{Plato}{|EndBreak}
Among philosophers we consider \Plato\ and \Aris.

```

```
\section{Historical Figures}

We ponder about \Aeth, then \Aeth.
We note \Soto, then just \Soto.
\CapThis\Soto{} starts a sentence.

\section{Further Discussion}

\TagName[George]{Washington}{!as general}
\TagName[Dwight D.]{Eisenhower}{!as general}
\LWash, \LIke, \LBradley, and \LPatton
  were high-ranking generals.\\
\SeeAlso\IndexRef[George]{Washington}{Eisenhower}
\SeeAlso\IndexRef[Dwight D.]{Eisenhower}{Washington}
\IncludeName*[George]{Washington}
\IncludeName*[Dwight D.]{Eisenhower}

\UntagName[George]{Washington}
\UntagName[Dwight D.]{Eisenhower}
\Wash\ and \Ike\ also were US presidents.

\ForgetName[Omar N.]{Bradley}
\ForgetName[George S.]{Patton, Jr.}
\TagName[Omar N.]{Bradley}{!colleagues of}
\TagName[George S.]{Patton, Jr.}{!colleagues of}
\IndexRef[Omar N.]{Bradley}{Eisenhower, Patton}
\IndexRef[George S.]{Patton, Jr.}{Bradley, Eisenhower}

\printindex
\end{document}
```



Index

3

A few notable features from this example include:

- The entry categories are set up in the preamble with `\PretagName`. They do not change thereafter.
- Changes from one sub-entry to another, or back to the base name entry, occur using `\TagName` and `\UntagName`.
- We permit references to names with other sub-entries or main entries again with `\IncludeName*`. Otherwise, the creation of an xref would protect that name from further entries.
- We permit the creation of *see* references in subentries of extant names by using `\ForgetName`

Obviously, the more complex an index gets, the more complex it will be to deal with the nameauth macros. The user must decide where the break-even point lies.

Back to Table of Contents

The world is very different now. For man holds in his mortal hands the power to abolish all forms of human poverty and all forms of human life. And yet the same revolutionary beliefs for which our forebears fought are still at issue around the globe — the belief that the rights of man come not from the generosity of the state but from the hand of God.

. . . And so, my fellow Americans: ask not what your country can do for you — ask what you can do for your country.

My fellow citizens of the world: ask not what America will do for you, but what together we can do for the freedom of man.

Finally, whether you are citizens of America or citizens of the world, ask of us here the same high standards of strength and sacrifice which we ask of you. With a good conscience our only sure reward, with history the final judge of our deeds, let us go forth to lead the land we love, asking His blessing and His help, but knowing that here on earth God's work must truly be our own.

—John F. Kennedy
Inaugural Address (1961)

8 Name Tags

8.1 Basics

\NameAddInfo

All valid names in nameauth can have name tags. Unlike other kinds of tags, name tags are not generated automatically with every name managed by nameauth. \NameAddInfo is \long, allowing for more information 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 name tag but does not print whenever Washington \Wash is used. The name tag is displayed only by using the macro \NameQueryInfo{} (1732–99).

When making a tag, include a leading space if \NameQueryInfo is used in a formatting hook (Section 8.2) and such space is desired. Otherwise avoid leading spaces. Another way uses a conditional:

```
\makeatletter
\NameAddInfo[George]{Washington}
  {\if@nameauth@InHook{ }\fi(1732--99)}
\makeatother
```

\NameQueryInfo

To print a name tag for a given name, we use \NameQueryInfo, either from an arbitrary set of name arguments, or from the previously-evaluated name arguments if a null argument is given. It prints a name tag from the name info data set:

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

Name Pattern(s):
George!Washington!DB
Elizabeth,I!MN
Elizabeth,I!DB

- With name arguments, using the pattern for those arguments:
  \NameQueryInfo[George]{Washington} . . . . . (1732–99)
- With a null argument, using the pattern from the last name used:
  \LEliz\ \NameQueryInfo{} . . . . . Elizabeth I "Gloriana"

\NameClearInfo

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

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

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

We now revisit George Washington and his associated name tag.

```
The name tag for \Wash\ is: \NameQueryInfo{}\\
Clearing the tag.\NameClearInfo[George]{Washington}
\ifcsdef{George!Washington!DB}
  {The tag exists}{The tag is gone.}
```

The name tag for Washington is: (1732–99)
Clearing the tag. The tag is gone.

8.2 Name Tags in Hooks

When calling name tags in formatting hooks (Section 9.1), it helps to remember that they can call other tags as well. Here we offer a quick reminder to stop potential recursion, either with Boolean flags or locally-defined macros. We show the latter below; note the extra grouping tokens:

```
\NameAddInfo{A}
  {{\newcommand\A{Tag A} \A \unless\ifdefined\B \NameQueryInfo{B} \fi}}
\NameAddInfo{B}
  {{\newcommand\B{Tag B} \B \unless\ifdefined\A \NameQueryInfo{A} \fi}}

\begin{itemize}
  \item \NameQueryInfo{A}
  \item \NameQueryInfo{B}
\end{itemize}
```

- Tag A Tag B
- Tag B Tag A

With version 4.1 of nameauth, printing name tags in hooks has become easy. For example, we can print tags with the first use of a name. Formatting hooks are called within a local scope. They can take zero arguments, where they insert macros like font switches before a name. They also can take one argument, where they start to more complex actions (Section 11). Here are two basic tips:

- Use the recommended template as a base model for custom hooks.
- Keep all changes local to the formatting hooks.

Below we recommend a starter hook template that uses `\NameQueryInfo` with a null argument (Section 6.1) to print the tag with the first use of a name:

```
\renewcommand*\NamesFormat[1]
  {#1\NameQueryInfo{}}
```

Here is another example with more nuanced control:

```
\newif\ifNoTag
\renewcommand*\NamesFormat[1]
{%
  #1\unless\ifNoTag\NameQueryInfo{}\fi
  \global\NoTagfalse%
}
```

We saw this next example in Section 1.7. If the tag is in a hook, it adds a space before it. Otherwise it adds no space. The color and font switches match the conventions of this manual:

```
\makeatletter
\NameAddInfo{Elizabeth, I}{\if@nameauth@InHook{ }\fi``Gloriana''}
\makeatother
\renewcommand*\NamesFormat[1]
  {\color{blue}\sffamily #1\NameQueryInfo{}}
\renewcommand*\MainNameHook{\sffamily}
```

Even though the core name engine has a "locked path" to protect against being reentrant, that does not prevent all problems. Below we show an example of a solution that will avoid halting LaTeX with an error. We ensure that, if we are in a formatting hook, we will not call an infinite series of `\NameQueryInfo`. Take care to note what happens in a formatting hook, and what happens outside of a hook.

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

```
\makeatletter
\NameAddInfo[Ulysses S.]{Grant}
  {\if@nameauth@InHook\ (term 1869--1877)\else
   eighteenth US president (1869--1877)\fi}
\NameAddInfo[Schuyler]{Colfax}
  {\if@nameauth@InHook\ (seventeenth US vice-president)\else
   \footnote{\Name[Schuyler]{Colfax} was the seventeenth
   US vice-president during the first term (1869--1873)
   of \Name*[Ulysses S.]{Grant}, \NameQueryInfo{}.}\fi}
\makeatother

\renewcommand*\NamesFormat{}
\renewcommand*\MainNameHook{}

Let's remember \Name[Schuyler]{Colfax}.\NameQueryInfo{}
Derived from ``scholar'', this name can occur
as ``Skylar'' for girls and ``Skyler'' for boys.

\renewcommand*\NamesFormat[1]{#1\NameQueryInfo{}}

\ForgetThis\Name[Ulysses S.]{Grant} had a poor reputation among
radical Republicans, especially the Forty-Eighters. Again we note
\ForgetThis\Name[Schuyler]{Colfax}.
```

Let's remember Schuyler Colfax.[43] Derived from "scholar", this name can occur as "Skylar" for girls and "Skyler" for boys.

Ulysses S. Grant (term 1869–1877) had a poor reputation among radical Republicans, especially the Forty-Eighters. Again we note Schuyler Colfax (seventeenth US vice-president).

Back to Table of Contents

> Any great work of art ... revives and readapts time and space, and the measure of its success is the extent to which it makes you an inhabitant of that world — the extent to which it invites you in and lets you breathe its strange, special air.
>
> —Leonard Bernstein
> "What Makes Opera Grand?", *Vogue* (December 1958)

[43] Colfax was the seventeenth US vice-president during the first term (1869–1873) of Ulysses S. Grant, eighteenth US president (1869–1877).

9 Basic Formatting and Name Decisions

9.1 Basic Formatting

This section gives a basic overview. Section 11 explores automation and more.

Syntactic Changes

Even with the default options for nameauth wherein no font changes occur, we can observe syntactic changes to names:

Syntactic Changes; No Formatting or Post-Processing

First Instance	Macro	Later Instance	Macro
George S. Patton Jr.	`\Patton`	Patton	`\Patton`
George S. Patton Jr.	`\LPatton`	George S. Patton Jr.	`\LPatton`
George S. Patton Jr.	`\SPatton`	George S.	`\SPatton`
Yamamoto Isoroku	`\Yamt`	Yamamoto	`\Yamt`
Yamamoto Isoroku	`\LYamt`	Yamamoto Isoroku	`\LYamt`
Yamamoto Isoroku	`\SYamt`	Yamamoto	`\SYamt`

We can add formatting to these changes by using formatting hooks. In its basic form, this refers to font changes. In its advanced form, one can add or modify text elements, syntactic forms of names, and so on, while leaving index entries undisturbed (Section 11.2). Although we have encountered examples of syntactic name changes since the start of this manual, here we define that behavior formally:

- First instance of a name
  - No name pattern exists.
  - A name is printed with its long form (default).
  - The "first-use" formatting hook is used (default).
  - After the name is printed, a name pattern is created.
- Subsequent instance of a name
  - A name pattern already exists.
  - A name is printed using a shorter form (default).
  - The "subsequent-use" formatting hook is used (default).

Two Formatting Systems

Many books are structured with front matter that includes a table of contents, foreword, introductory material, and other instructive content that is not part of the main matter. The nameauth package has separate syntax and formatting system for front matter (`!NF`) and main matter (`!MN`), linked to name patterns (Section 6.2).

`\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

Both \NamesInactive and \NamesActive can be used explicitly as a pair or singly within a 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. Since the formatting hooks always are defined when using nameauth, one must use \renewcommand when changing their definitions.[44] Below we color-code the systems and "forget" previous name uses:

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

Front-matter system: \NamesInactive

Rudolph Carnap	\Name[Rudolph]{Carnap}
Carnap	\Name[Rudolph]{Carnap}
Nicolas Malebranche	\Name[Nicolas]{Malebranche}
Malebranche	\Name[Nicolas]{Malebranche}

Main-matter system: \NamesActive

Rudolph Carnap	\Name[Rudolph]{Carnap}
Carnap	\Name[Rudolph]{Carnap}
Nicolas Malebranche	\Name[Nicolas]{Malebranche}
Malebranche	\Name[Nicolas]{Malebranche}

We used the xcolor package with the following macros:

```
1 \renewcommand*\FrontNamesFormat[1]{\color{violet}\sffamily #1}
2 \renewcommand*\FrontNameHook[1]{\color{darkgray}\sffamily #1}
3 \renewcommand*\NamesFormat[1]{\color{blue}\sffamily #1}
4 \renewcommand*\MainNameHook[1]{\sffamily #1}
```

\ForceName

Use this prefix macro to force "first use" formatting for the next \Name, etc., but without deleting any name patterns. See also Sections 8.2, 9.3, and 12.1. Thus:

Name Pattern(s):
Rudolph!Carnap!MN

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

In cases where formatting also changes the syntactic form of a name in addition to its typography (Section 11.2f.), \ForceName could interfere with \ForceFN and \ForceAffix.

alwaysformat

Below we simulate alwaysformat via package internals. Only the "first use" formatting hooks are used. This is a relic from early versions of nameauth.

- 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

[44] The names of these macros grew from \NamesFormat, originally the only formatting hook. Especially with the macros in this section, the naming scheme is unfortunate because package development involved some groping in the dark regarding the concepts.

Hook caveats

The core name engine determines what syntactic name elements exist and how they are to be displayed (Sections 6.1 and 11). The format hook dispatcher passes the name to the formatting hooks within a local scope:[45]

`\bgroup\@nameauth@InHooktrue`⟨*Hook*⟩`{#1}\egroup`.

Due to this design, both the following achieve the same effect:

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

One can create formatting hooks that take either no argument or one argument. Since the formatting hooks are already defined, one must not use `\newcommand` to create new hooks. Instead, use `\renewcommand` e.g.:

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

A hook with one argument has more options than a font switch (Section 11.4). To test if one is in a formatting hook, use `\if@nameauth@InHook` (Sections 1.7 and 8).

9.2 Application: Footnotes

Names in the body text, such as Adolf Harnack (later ennobled to Adolf von Harnack), normally affect name forms in the footnotes.[46] In footnote 46 below, `\MainNameHook` is called instead of `\NamesFormat` because Harnack occurred above. We could use `\ForgetThis` (Section 9.3) in the footnote to change that behavior. Another way uses the front-matter system and a redefined footnote macro:

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

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

Now we see different results.[47] Footnote 47 shows the first use of a name because it is the first use in the front-matter system. Below we revert footnotes with:

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

> He who exercises government by means of his virtue may be compared to the north polar star, which keeps its place and all the stars turn towards it.
>
> —Confucius
> *The Analects*, C II (475–221 BC)

[45] We moved `\@nameauth@InHooktrue` into the immediate hook groups from the beginning of the hook dispatcher to make the logic and functionality more obvious to read.

[46] We have Harnack from `\Harnack` instead of Adolf Harnack.

[47] We have Adolf Harnack from `\Harnack`, then Harnack.

9.3 Making Name Decisions

The macros that switch between name systems are local (`\NamesActive` and `\NamesInactive`). Otherwise, all the macros below are **global** with respect to name patterns and scoping. Unless restricted to the current name system by `\LocalNames`, they affect both name systems. Creating and deleting name patterns affects syntactic name forms, formatting, index protection of cross-references (Section 7.3), and the outcomes of name tests (Section 9.5).

`\ForgetName` This macro "forgets" a name by deleting its pattern. That forces a "pre-first use" state. The core name engine treats the name as a first use. Some test results change (Section 9.5), and name protection for indexing is removed, similar to the way that `\IncludeName*` "unprotects" xrefs (Section 7.3.2).

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

`\ForgetThis` This macro makes the name engine "forget" the name pattern just before the parser, printing the name as a first use, then "remembering" the name. This can affect some tests in formatting hooks. After knowing Einstein, we forget him again:[48]

Albert Einstein ..... `\ForgetThis\Einstein`

`\SubvertName` This macro "subverts" a name by creating a name pattern. This makes the name engine handle a name as a subsequent use, changes some test results, and "protects" a name from being used as a *see* reference, analogous to `\ExcludeName` and `\IndexRef`:

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

`\SubvertThis` This macro causes the name egine to create a name pattern just before the parser, printing the name as a subsequent use. This can affect some tests in formatting hooks. `\ForgetThis` has a higher priority than `\SubvertThis` and negates it (Section 3).

> We advise users to avoid using `\ForgetThis` and `\SubvertThis` before any macro that does not print a name in the text.

`\LocalNames`
- The helper macro `\LocalNames` restricts the macros above to the **current** name system, but not to any scope.

`\GlobalNames`
- The helper macro `\GlobalNames` restores the effects of these macros to **both name systems.**

In the next example we define a macro that reports whether or not the specific name `\Name[Charlie]{Chaplin}` exists. There are four outcomes: the name exists in the main matter, in the front matter, in both name systems, or it does not exist. The naming macros use the current name system. The macros on this page affect both name systems by default.

[48] `\ForgetThis` no longer affects the index unless one uses the `oldreset` option. Likewise, `\SubvertThis` no longer affects the index unless one uses the `oldreset` option.

```
\newcommand\CheckChuck{{Name systems:
  \IfFrontName[Charlie]{Chaplin}
  {\bfseries\IfMainName[Charlie]{Chaplin}{in both}{in front}}
  {\bfseries\IfMainName[Charlie]{Chaplin}{in main}{in none}}}}

Create a name pattern in both systems\dotfill
\SubvertName[Charlie]{Chaplin}\CheckChuck

Restrict to the current (main) name system. \LocalNames

Forget a name in the main-matter system\dotfill
\ForgetName[Charlie]{Chaplin}\CheckChuck

Restored: \Name[Charlie]{Chaplin}\dotfill\CheckChuck

Switch to the front matter. \NamesInactive

Forget a name in the front-matter system\dotfill
\ForgetName[Charlie]{Chaplin}\CheckChuck

Restored: \Name[Charlie]{Chaplin}\dotfill\CheckChuck

Switch to the main matter. \NamesActive

Lift name system restrictions. \GlobalNames

Forget a name in both systems\dotfill
\ForgetName[Charlie]{Chaplin}\CheckChuck
```

Name Pattern(s):
`Charlie!Chaplin!MN`
`Charlie!Chaplin!NF`

Create a name pattern in both systems ..... Name systems: **in both**

Restrict to the current (main) name system.

Forget a name in the main-matter system..... Name systems: **in front**

Restored: Charlie Chaplin..... Name systems: **in both**

Switch to the front matter.

Forget a name in the front-matter system..... Name systems: **in main**

Restored: Charlie Chaplin..... Name systems: **in both**

Switch to the main matter.

Lift name system restrictions.

Forget a name in both systems..... Name systems: **in none**

9.4 Summary

We summarize info on name forms and formatting, using the following macros:

```
\renewcommand*\NamesFormat[1]
  {\color{blue}\sffamily#1\NameQueryInfo{}}
\renewcommand*\MainNameHook{\sffamily}
\begin{nameauth}
  \< Bailey & Betsey & Bailey & >
  \< Faisal & & Faisal, bin Abdulaziz & >
\end{nameauth}
\NameAddInfo{Faisal, bin Abdulaziz}{ Al Saud}
\TagName{Faisal, bin Abdulaziz}{ Al Saud, king}
```

Now we show the macros at work. Regarding the automatic use of the name tag in the first-use hook, see Section 8.2.

Name Pattern(s):
`Betsey!Bailey!MN`
`Faisal,binAbdulaziz!MN`

First Instance: Name pattern created with text output. Index state 2, 4, or 6 (Section 6.4). Name form: long. First-use hooks.

Betsey Bailey ..... `\Bailey`
`\LBailey`
`\SBailey`

Faisal bin Abdulaziz Al Saud ..... `\Faisal`
`\LFaisal`
`\SFaisal`

Later Instance: No change to name pattern. Index state 2, 4, or 6. Name form: short or long. Subsequent-use hooks.

Bailey ..... `\Bailey`

Betsey Bailey ..... `\LBailey`

Betsey ..... `\SBailey`

Faisal ..... `\Faisal`
`\SFaisal`

Faisal bin Abdulaziz ..... `\LFaisal`

bin Abdulaziz ..... `\ForceFN\SFaisal`

Forgotten: Name pattern deleted. Index state 1, 3, or 5. Next instance usually will be a first instance.

(no output) ..... `\ForgetName[Betsey]{Bailey}`

(no output) ..... `\ForgetName{Faisal, bin Abdulaziz}`

Subverted: Name pattern created. Index state 2, 4, or 6. Next instance usually will be a later instance.

(no output) ..... `\SubvertName[Betsey]{Bailey}`

(no output) ..... `\SubvertName{Faisal, bin Abdulaziz}`

`\ForgetThis`: Name pattern deleted, then created again. Index state 2, 4, or 6. Name form and format: first instance. Next instance usually will be a later instance.

Betsey Bailey ..... `\ForgetThis\Bailey`
`\ForgetThis\LBailey`
`\ForgetThis\SBailey`

Faisal bin Abdulaziz Al Saud ..... `\ForgetThis\Faisal`
`\ForgetThis\LFaisal`
`\ForgetThis\SFaisal`

`\SubvertThis`: Name pattern created. Index state 2, 4, or 6. Name form and format: later instance. Next instance usually will be a later instance.

Bailey ..... `\SubvertThis\Bailey`

Betsey Bailey ..... `\SubvertThis\LBailey`

Betsey ..... `\SubvertThis\SBailey`

Faisal ..... `\SubvertThis\Faisal`
`\SubvertThis\SFaisal`

Faisal bin Abdulaziz ..... `\SubvertThis\LFaisal`

bin Abdulaziz ..... `\SubvertThis\ForceFN\SFaisal`

Format as First: No change to name pattern. Index state 2, 4, or 6. Name form: short or long. Name format: First-use hooks.

Bailey ..... \ForceName\Bailey

Betsey Bailey ..... \ForceName\LBailey

Betsey ..... \ForceName\SBailey

Faisal Al Saud ..... \ForceName\Faisal

Faisal bin Abdulaziz Al Saud ..... \ForceName\LFaisal

Faisal Al Saud ..... \ForceName\SFaisal

bin Abdulaziz Al Saud ..... \ForceName\ForceFN\SFaisal

9.5 Testing Name Decisions

Since name patterns are control sequences like macros, we can test for their existence. This can relate names to each other dynamically throughout a document.

9.5.1 Testing Macros

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

The default behavior **encapsulates the decision paths in a local scope,** insulating some changes therein. If this is not desired, use the `globaltest` option or `\GlobalNameTest`. `\LocalNameTest` will re-enable the default. Any naming macros (`\Name`, etc.) used in these paths will have global assignments associated with them, i.e., name patterns, tags, etc.

\IfMainName

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

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

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

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

I heard someone say: No Bob here.

Now we test for `\Name{Elizabeth,I}`, which has occurred, and show how local and global test paths differ with respect to scope. At least some LaTeX commands that define macros will behave differently than `\def`:

```
\GlobalNameTest
\def\msg{We reserve judgment on \LEliz}
\IfMainName{Elizabeth,I}
  {\def\msg{\LEliz\ is Gloriana!}}
  {\def\msg{We do not know of \LEliz}}
\parbox{0.4\textwidth}{\msg} (\cmd{\GlobalNameTest}).

```

```
\LocalNameTest
\def\msg{We reserve judgment on \LEliz}
\IfMainName{Elizabeth,I}
  {\def\msg{\LEliz\ is Gloriana!}}
  {\def\msg{We do not know of \LEliz}}
\parbox{0.4\textwidth}{\msg} (\cmd{\LocalNameTest}).
```

Elizabeth I is Gloriana! (\GlobalNameTest).

We reserve judgment on Elizabeth I (\LocalNameTest).

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

\IfFrontName[⟨*FNN*⟩]{⟨*SNN,Affix*⟩}[⟨*Alt.*⟩]{⟨*yes*⟩}{⟨*no*⟩}

This macro works just like \IfMainName, except using the "front matter" name patterns as the test subject. These testing macros prove their worth especially through combination. For example, we do a test based on Section 9.1.

```
\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*⟩}[⟨*Alt.*⟩]{⟨*y*⟩}{⟨*n*⟩}{⟨*excl*⟩}

This macro also works like \IfMainName, except that it has an additional ⟨*excl*⟩ branch to detect names excluded by \ExcludeName (Section 7.1). If we do not use \ExcludeName, we can leave the ⟨*excl*⟩ branch empty.

- Cross-references (the ⟨*y*⟩ path) are name patterns ending in !PN (Section 6.2). They expand to empty.
- Excluded names (the ⟨*excl*⟩ path) have xrefs that expand to the value of \@nameauth@Exclude.
- \ExcludeName creates excluded xrefs. \IncludeName destroys them.
- \IndexRef, \AKA, \AKA*, \PName, and \PName* create xrefs, which are destroyed only by \IncludeName*.

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

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

First we mention Jesse Ventura `\Name[Jesse]{Ventura}`, pro wrestler and Minnesota governor. His lesser-known legal name, James Janos is an alias: `\IndexRef[James]{Janos}{Ventura, Jesse}\Name[James]{Janos}`. The order of the macros is important. The test above says: Jesse Ventura is a stage name. We also can create a complete, unified test:

```
\IfAKA[FNN]{SNN, Affix}[Alt.]
  {It is an xref.}
  {It is a name.
   \IfFrontName[FNN]{SNN, Affix}[Alt.]
     {% It is at least in the front matter
      \IfMainName[FNN]{SNN, Affix}[Alt.]
        {It is in both front and main matter.}
        {It is only in the front matter.}%
     }
     {% It is not in the front matter.
      \IfMainName[FNN]{SNN, Affix}[Alt.]
        {It is only in the main matter.}
        {It does not exist.}%
     }%
  }
  {It is excluded.}
```

9.5.2 Applications

As we have seen, we can control name forms with `\ForgetThis` and the like. Yet we also can use tests in situations where we might not know if a name has appeared, like when a name appears in a `table` or other float environment. Thus:

```
\IfMainName[M.S.]{Omartian}
  {\Name[M.S.]{Omartian}}
  {\Name[M.S.]{Omartian}[Michael]} produced many number-one
  records over three decades.
```

Name Pattern(s):
M.S.!Omartian

Michael Omartian produced many number-one records over three decades.

Next we recall the movie, *Ferris Bueller's Day Off*. Using conditionals, we can design statements that are sensitive to the plot.

```
This is \FName[Cameron]{Frye}.
\IfMainName[Ferris]{Bueller}
  {He is developing positive traits.}
  {He is gloomy and introspective.}
\Name[Ferris]{Bueller} decides to hang out with
\FName[Cameron]{Frye}.
\IfMainName[Ferris]{Bueller}
  {He is developing positive traits.}
  {He is gloomy and introspective.}
```

This is Cameron Frye. He is gloomy and introspective. Ferris Bueller decides to hang out with Cameron. He is developing positive traits.

Below, we can extend the use of conditionals to name tags. This allows different facts to appear with names, depending on what other names have appeared. We forget all names beforehand and show the following example:

Name Pattern(s):

```
Paul!PN
Saul,ofTarsus!DB
Jesus,Christ!MN
Lucius!SergiusPaulus!MN
Paul!MN
Saul,ofTarsus!MN
```

```
\NameAddInfo{Saul, of Tarsus}
  {\IfMainName{Jesus, Christ}
    {\IfMainName[Lucius]{Sergius Paulus}
      {called himself \Name{Paul},
       perhaps in honor of his patron}
      {next became a preacher to the Gentiles}}
    {wrote first that he persecuted Christians}}

\ForgetName{Jesus, Christ}
\ForgetName[Lucius]{Sergius Paulus}
\IndexRef{Paul}{Saul of Tarsus}

\Name{Saul, of Tarsus} \NameQueryInfo{}.
He wrote in the letter to the Galatians, later reported in
the book of Acts, that he saw a vision of \Name{Jesus, Christ}
on the road to Damascus.

\Name{Saul, of Tarsus} \NameQueryInfo{}.
He undertook three missionary journeys before being
sent to Rome for trial in an appeal to Caesar.
While in Cyprus, \Name{Saul, of Tarsus} converted
\Name[Lucius]{Sergius Paulus}, who became a patron.

\Name{Saul, of Tarsus} \NameQueryInfo{}.
Under the name \Name{Paul} he wrote his letters
during his journeys and imprisonment.
```

Saul of Tarsus wrote first that he persecuted Christians. He wrote in the letter to the Galatians, later reported in the book of Acts, that he saw a vision of Jesus Christ on the road to Damascus.

Saul next became a preacher to the Gentiles. He undertook three missionary journeys before being sent to Rome for trial in an appeal to Caesar. While in Cyprus, Saul converted Lucius Sergius Paulus, who became a patron.

Saul called himself Paul, perhaps in honor of his patron. Under the name Paul he wrote his letters during his journeys and imprisonment.

Caveats

- One should check and validate test results. That is particularly true when using the tests inside macros. See *The TEXbook*, 212–15.
- See Section 6.5 regarding the use of `\noexpand` in name arguments to stabilize name patterns so that comparisons are consistent and valid.
- See also Section 4.3.8 regarding possible Unicode issues. This is perhaps less of an issue in modern TEX distributions.
- The trace package, along with the macros `\show` and `\meaning`, can help debug problems.

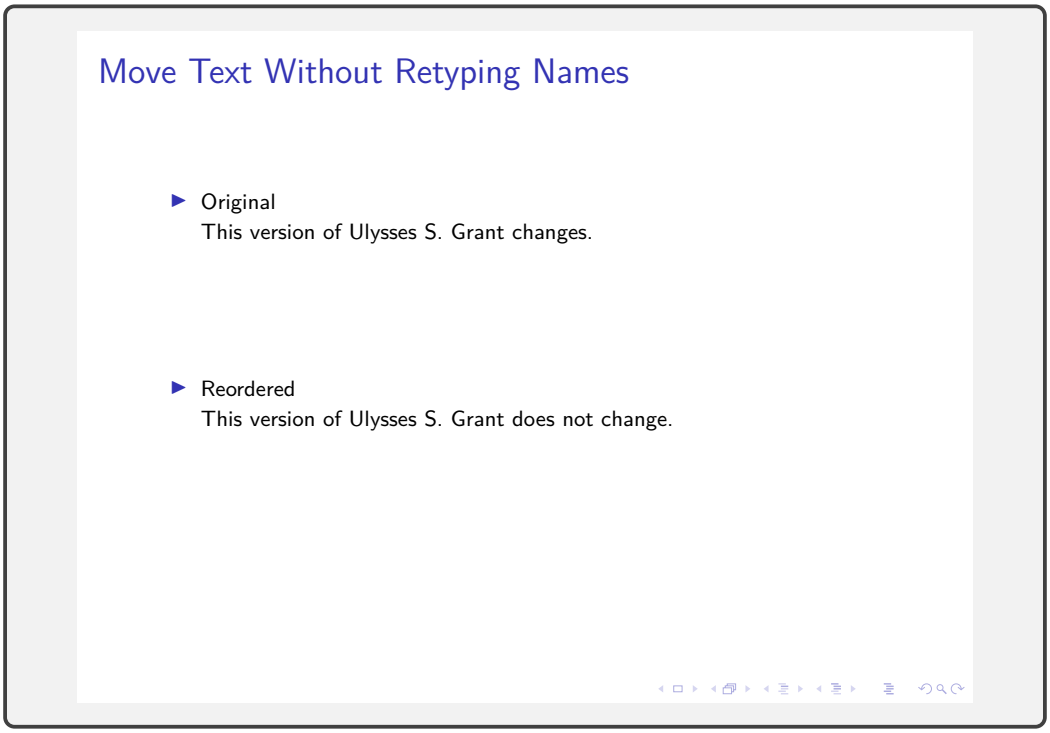
9.5.3 Beamer Example

Names will change automatically in beamer overlays as one advances the slides unless we control them, as shown in the following example.

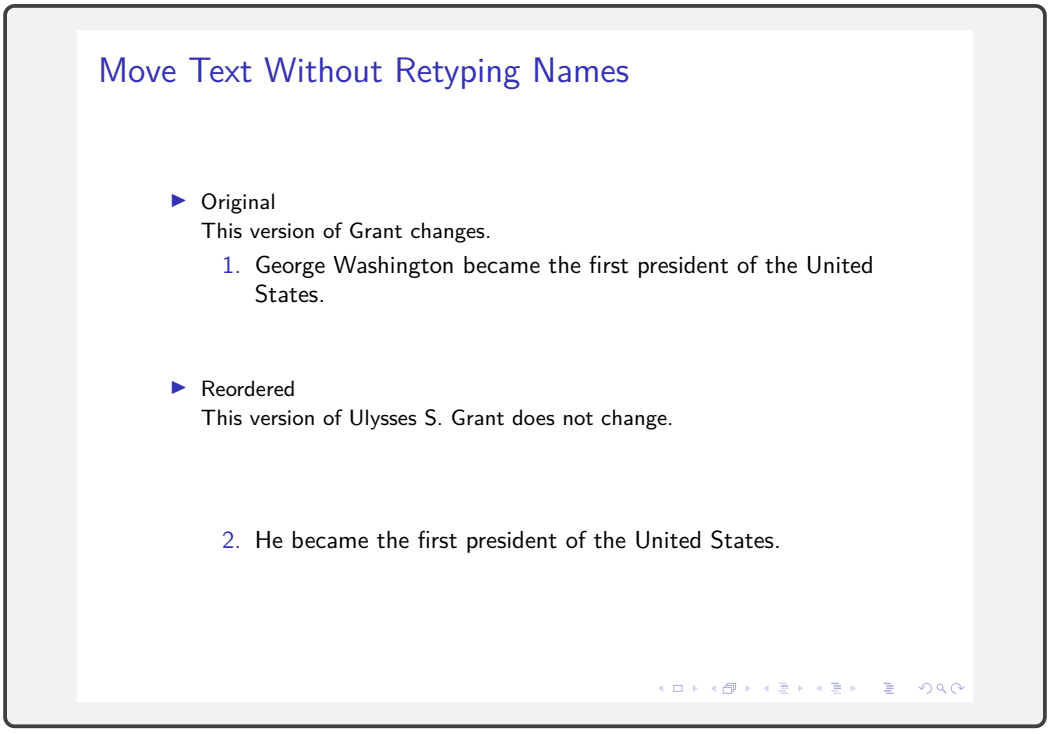
```
\documentclass{beamer}
\input{compat.tex} % Included with nameauth for compatibility.
\usepackage[noindex]{nameauth}

\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}%
            {\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}%
            {\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 from one to three, keep name forms consistent by deleting name patterns for each new overlay. Otherwise, name patterns would change for each new overlay.

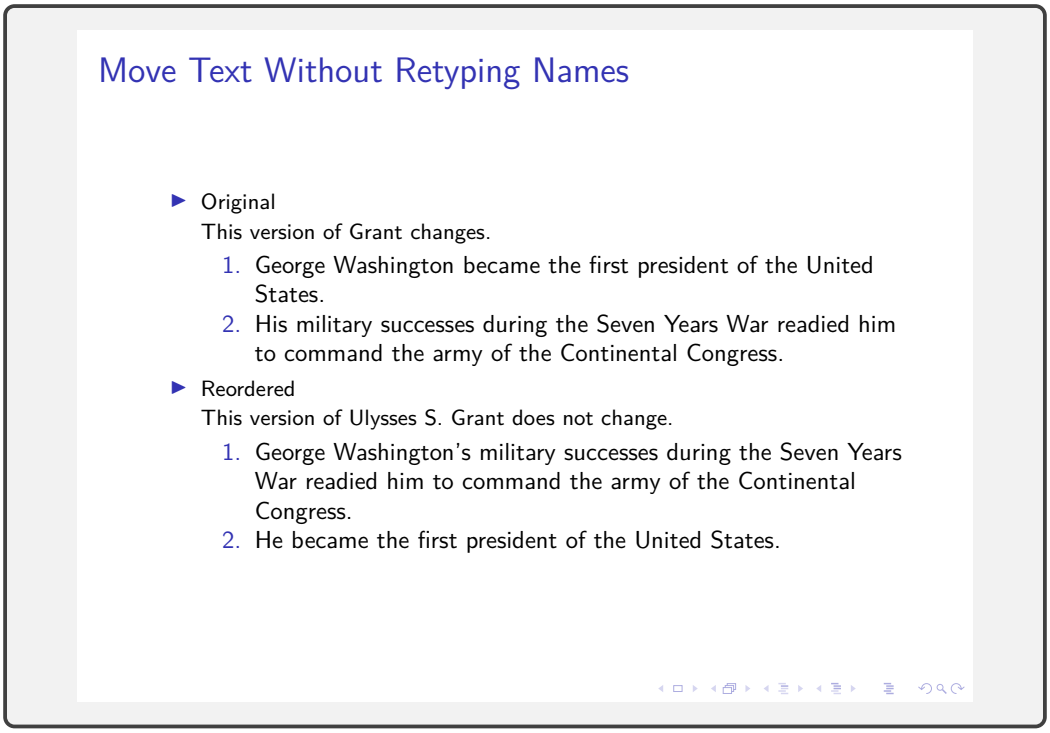
Name conditionals ensure specific, context-dependent forms based on what name has appeared. These conditionals allow the text in each overlay to be order-independent and able to be moved around at will. The first overlay shows the use of `\ForgetThis` to keep names constant.



The second overlay uses `\ForgetName` forcing specific name forms respective to each overlay, instead of respective to the overall sequence of overlays. We also observe the use of name conditionals in text elements that one might reorder.



Overlay three shows how these features integrate. A presenter can design slides that determine name forms, regardless of being re-ordered.



Back to Table of Contents

It is a very poor thing, whether for nations or individuals, to advance the history of great deeds done in the past as an excuse for doing poorly in the present; but it is an excellent thing to study the history of the great deeds of the past, and of the great men who did them, with an earnest desire to profit thereby so as to render better service in the present. In their essentials, the men of the present day are much like the men of the past, and the live issues of the present can be faced to better advantage by men who have in good faith studied how the leaders of the nation faced the dead issues of the past.

—Theodore Roosevelt
Introduction, *The Papers and Writings of Abraham Lincoln* (1905)

10 Name Authority Basics

10.1 Variant Names

This section explains how to manage simpler surname variants. There are several ways that nameauth can handle variants, in increasing levels of complexity.

Group 1: Use ⟨*Alternate*⟩ to create variant forms, yet retain consistent index entries. This is the default function of nameauth seen already.

Group 2: Use methods from Sections 7.2 and 7.3, and the manual approaches in Section 5.10.

- Use a particular name as a "standard" variant, then use other variants with a *see* reference.
- Use one or more names as "standard" variants that refer to each other via *see also* references.

Group 3: Use alternate formatting and macros in name arguments (Sections 11.2 and 5.10).

> Address any concerns about expansion of macros in name arguments by using `\noexpand` before those macros.

10.1.1 Variants and the ⟨*Alternate*⟩ Argument

We begin with the first kind of variant names listed above. We decide that the canonical name to be used is Mike Tyson. We set up both the canonical name and an alternate name in the `nameauth` environment:

Name Pattern(s):
`Mike!Tyson!MN`

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

\IndexRef{Iron Mike}{Tyson, Mike}
```

Because `\Iron` uses the ⟨*Alternate*⟩ column, all index page entries are the same as those for `\Tyson`, the canonical name. Adding the cross-reference via `\IndexRef` produces "Iron Mike *see* Tyson, Mike" in the index.

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

Yet ⟨*Alternate*⟩ does more than handle variant forenames in Western name forms. It can be used to manage alternate names in Eastern and ancient forms, as we have already seen. For this to work properly, we must have a name where ⟨*SNN*⟩ and ⟨*Affix*⟩ are both populated in order to use ⟨*Alternate*⟩. Otherwise, one winds up with the obsolete syntax (Section 12.2). That probably is undesired.

We will engage with different kinds of variant names more, beginning with Section 5.9, Here we give a basic illustration of how one can start to manage these names just by using ⟨*Alternate*⟩. For instance:

Name Pattern(s):
Elizabeth,I!MN

```
\begin{nameauth}
  \< Eliz & & Elizabeth, I & >
\end{nameauth}

\IndexRef{Gloriana}{Elizabeth I}
\IndexRef[Good Queen]{Bess}{Elizabeth I}

\LEliz[I, ``Gloriana''] was known also as
\ForceFN\SEliz[``Good Queen Bess''].
```

Elizabeth I, "Gloriana" was known also as "Good Queen Bess".

10.1.2 Managing Multiple Variant Names

With the second group of name variants listed above (relating to Sections 7.2 and 7.3), we get into pseudonyms, aliases, and variant family names. This class is more complicated for the following reasons:

- Names having separate index entries are linked together with cross-references; most of the work that interrelates these names is done with the indexing macros.
- Names that the author wants to treat as different may be seen by the nameauth internals as identical (Section 6.3). One can use nonprinting macros or grouping tokens in name arguments, index tagging macros, and formatting macros to create unique names (see below and Section 7.7).

We approach and cross the boundary between the second and third groups of names mentioned above when we deal with names that contain particles, that change capitalization, that have changing grammatical endings (Sections 5.7, 5.8, 5.9, 5.10, and 11.2.5f.).

The following method avoids using macros in name arguments and it is easier to set up. The trade-off is that, while macros in name arguments are harder to set up, they benefit from automation. Below we establish two names and a sort key for the main name under which both names are indexed:

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

Name Pattern(s):
W.E.B.!Du~Bois!MN
W.E.B.!DuBois!MN

- Based on historical research and some name authorities, we decide that the the canonical name will be: W.E.B. Du Bois `\DuBois`.
- We use the non-breaking space (the tilde active character) because internally, nameauth removes all regular spaces from name patterns. Both `\Name[W.E.B.]{Du Bois}` and `\Name[W.E.B.]{DuBois}` have the same name pattern: `W.E.B.!DuBois` (Section 6.2). The name pattern `W.E.B.!Du~Bois` differs from both of the other names.

- Another reason to use the non-breaking space is that it prevents a line break between the particle *Du* and the name *Bois*.
- The sort key that could be applicable to both names is `{DuBois, William}`. Had we used the sort key `{Du Bois, William}`, the name would be sorted before `dual`, which is not in order (Section 7.6.2).
- Instead of using `\SkipIndex\AltDuBois` many times, we create a cross-reference before the alternate name is used to prevent index page entries from being created for the alternate form:

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

- With the following setup we keep full stop detection and modify name forms. The only manual part would be to check if the name straddles a page break, then add `\JustIndex\DuBois` after the name if needed:

  ```
  1 \newcommand\VarDuBois{\JustIndex\DuBois\AltDuBois}
  2 \newcommand\LVarDuBois{\JustIndex\DuBois\LAltDuBois}
  3 \newcommand\SVarDuBois{\JustIndex\DuBois\SAltDuBois}
  4 \VarDuBois[William E.B.];
  5 \LVarDuBois;
  6 \SVarDuBois[William]
  ```

  William E.B. DuBois; W.E.B. DuBois; William

There are other solutions that one might envision, but they are a bit unwieldy. It does not seem practical to implement complex solutions for names unless perhaps one is using LaTeX as a back-end for perhaps an XML workflow.

10.1.3 Nonstandard Capitalization and Indexing

Name Pattern(s):
`e.e.!cummings!MN`
Basic Index:
cummings, e.e.

Here we look at simple nonstandard capitalization. Section 11.2 dals with more complex examples. We consider poet e.e. cummings. Using the default `noformat` option, one can begin a sentence with something like:

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

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

If we want formatting and capitalization, we can do it this way:

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

Name Pattern(s):
`e.e.!cummings's!PN`
`e.e.!cummings's!MN`

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

In both examples above, we use `\SubvertThis` to force a subsequent use in order to prevent a first use that looks like "E.e. Cummings's". The macro `\CapThis` will capitalize the first letter in **all name elements.** Using `\ExcludeName` keeps one from having to use `\SkipIndex` every time.

10.1.4 Variant Names and Index Cross-References

Single Connections

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

```
\begin{nameauth}
  \< Carter & J.E. & Carter, Jr. & >
\end{nameauth}
```

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 entries:

   James Earl Carter Jr. . . . . `\LCarter[James Earl]`

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

   Jimmy Carter . . . . . `\DropAffix\LCarter[Jimmy]`

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

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 the xref will be ignored and a warning will result:

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

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

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

5. We must index the alternate name with one of the following:

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

   `\JustIndex\Carter`

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 at the end of the document, **after** all uses of the alternate name:

   `\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 entries:

   Maimonides . . . . . `\Name{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 have a main name with a page entry and a "*see* reference" to that name. No page entries will occur below because we made the xref first:

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

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

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

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

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

Multiple Targets

There is a 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`

```
\PretagName{\textit{Snellius}}{Snellius}
\IndexRef{\textit{Snellius}}
  {Snel van Royen, R.; Snel van Royen, W.}
Both \Name[W.]{Snel van Royen}[Willebrord] and
his son \Name[R.]{Snel van Royen}[Rudolph] were known
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*.

One must plan the location of xrefs or use `\IncludeName*`. Above, we have no page entry for `\Name{\textit{Snellius}}` because `\IndexRef` comes first.

10.2 Using a Name Authority

Below are two names from a name authority that this author created for a translation of *De Diaconis et Diaconissis Veteris Ecclesiae Liber Commentarius* by Caspar Ziegler. To get valid names that one can research, here are some tips:

- CERL Thesaurus
- Virtual International Authority File
- EDIT16
- WorldCat
- Library of Congress
- Older version of Graesse, *Orbis Latinus*

1. Vernacular names are canonical. Latin names refer to them. That holds also for place names.
2. Sort vernacular names with `\PretagName` due to particles (Section 7.6).
3. If Latin names are only cross-references, use `\IndexRef`⟨*name args*⟩ to generate cross-references before referring to any names (Section 7.3).
4. If Latin names have page entries, then place `\SeeAlso\IndexRef`⟨*name args*⟩ as needed at the end of the document, before `\printindex`.
5. English usage requires `\CapThis` with particles in ⟨*SNN*⟩ (Section 5.7).

Name Pattern(s):
Jacques!De~Pamele!MN
Jacobus!Pamelius!MN
Giovanni!d'Andrea!MN
Ioannes!Andreae!MN
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 Jacques de Joigny De Pamele
`\Name[Jacques]{De~Pamele}[Jacques de~Joigny]`

Latin Name Jacobus Pamelius
`\Name[Jacobus]{Pamelius}`

Canonical Name Giovanni d'Andrea
`\Name[Giovanni]{d'Andrea}`

Latin Name Ioannes Andreae
`\Name[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 Table of Contents

In the awful cataclysm of World War, where from beating, slandering, and murdering us the white world turned temporarily aside to kill each other, we of the Darker Peoples looked on in mild amaze.

. . . The fateful day came. It had to come. The cause of war is preparation for war; and of all that Europe has done in a century there is nothing that has equaled in energy, thought, and time her preparation for wholesale murder. The only adequate cause of this preparation was conquest and conquest, not in Europe, but primarily among the darker peoples of Asia and Africa; conquest, not for assimilation and uplift, but for commerce and degradation. For this, and this mainly, did Europe gird herself at frightful cost for war.

—W.E.B. Du Bois
Darkwater: Voices from within the Veil (1920)

11 Advanced Formatting

Up to this point, formatting hooks have applied font and color changes to a parsed name, and perhaps printed a name tag. In this section we start to change the syntactic form of a name inside the formatting hooks. Below we see a general scheme of how the core name engine processes a name:

Naming Macro Layer

Naming macros (`\Name`, `\Name*`, `\FName`, `\FName*`, and the macros created by the `nameauth` environment) send arguments to three "interface hooks" that, by default, point to `\@nameauth@Name` (Section 11.5.2). That macro is the backend for both the basic and quick interfaces.

This backend handles work requested by `\JustIndex`, `\SubvertThis`, `\ForgetThis`, and `\SkipIndex`. Unless `\JustIndex` occurred, the backend then hands off the name arguments to the name parser, with either a main-matter `!MN` suffix or a front-matter `!NF` suffix.

Syntactic Element Layer

The `\@nameauth@Parse` macro determines name category (Western or Non-Western), capitalization, punctuation, and name elements to be used. All needed macros are made available to the next layer.

Name Display Layer

The two main macros in this layer are `\@nameauth@NonWest` and `\@nameauth@West`. The macro `\@nameauth@Form` has rules that set Boolean flags to determine the name form to be displayed, which goes to the format hook dispatcher.

Format Hook Dispatcher

`\@nameauth@Hook`: Check the name to be printed for a final full stop. Check which name system we are using.

Call the appropriate formatting hook, depending on whether or not the name pattern exists, and if we are in the main-matter system or the front-matter system.

`\@nameauth@Parse`: Create the name's pattern, a control sequence that makes the name "exist" for the sake of `nameauth`.

`\@nameauth@Name`: If the final full stop flag is true, check the lookahead token for a full stop and gobble it if needed.

The secret to being a bore is to tell everything.

—Voltaire
Sept Discours en Vers sur l'Homme (1738)

11.1 Formatting Hooks

General Hook Redefinition

```
\renewcommand*\FrontNamesFormat{}
\renewcommand*\FrontNameHook{}
\renewcommand*\NamesFormat{}
\renewcommand*\MainNameHook{}
```

This proof of concept puts the first mention of a name either in italics (front matter) or in boldface (main matter), and it adds a margin note if that is allowed. We use `\let` to save and restore the old hooks:

```
\documentclass{article}
\input{compat.tex} % Included with nameauth for compatibility.
\usepackage{makeidx}
\usepackage{nameauth}

\makeindex

% First save main- and front-matter hooks. Then change
% first-use hooks for both main matter and front matter.
\let\OldFormat\NamesFormat
\let\OldFrontNames\FrontNamesFormat

\renewcommand*\NamesFormat[1]{\textbf{#1}\unless\ifinner
   \marginpar{\raggedleft\scriptsize #1}\fi}
\renewcommand*\FrontNamesFormat[1]{\textit{#1}\unless\ifinner
   \marginpar{\raggedleft\scriptsize #1}\fi}

\PretagName{Vlad, {\c T}epe{\c s}}
  {Vlad Tepes}                    % for accented names
\TagName{Vlad, II}{ Dracul}       % for index information
\TagName{Vlad, III}{ Dracula}
\IndexRef{Dracula}{Vlad III}

\begin{document}

\begin{quote}
The new format (front matter):\NamesInactive

\Name{Vlad, III}[III Dracula], known as
\IndexRef{Vlad, {\c T}epe{\c s}}{Vlad III}%
\SubvertThis\Name*{Vlad, {\c T}epe{\c s}}
(\Name*{Vlad, {\c T}epe{\c s}}[the Impaler])
after his death, was the son of \Name{Vlad, II}[II Dracul],
a member of the Order of the Dragon. Later instances of
``\Name*{Vlad, III}'' and ``\Name{Vlad, III}'' appear thus.

The new format (main matter):\NamesActive

```

```
\Name{Vlad, III}[III Dracula], known as
\IndexRef{Vlad, {\c T}epe{\c s}}{Vlad III}%
\SubvertThis\Name*{Vlad, {\c T}epe{\c s}}
(\Name*{Vlad, {\c T}epe{\c s}}[the Impaler])
after his death, was the son of \Name{Vlad, II}[II Dracul],
a member of the Order of the Dragon. Later instances of
``\Name*{Vlad, III}'' and ``\Name{Vlad, III}'' appear thus.

\let\NamesFormat\OldFormat
\let\FrontNamesFormat\OldFrontNames

We are back in the old format.

in the front matter we see: \NamesInactive
\ForgetThis\Name{Vlad, III}[III Dracula],
\Name*{Vlad, III}, and  \Name{Vlad, III}.

in the main matter we see: \NamesActive
\ForgetThis\Name{Vlad, III}[III Dracula],
\Name*{Vlad, III}, and  \Name{Vlad, III}.
\end{quote}

\printindex
\end{document}
```

The new format (front matter):

Vlad III Dracula
Vlad II Dracul

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

The new format (main matter):

Vlad III Dracula
Vlad II Dracul

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

We are back in the old format.

in the front matter we see: Vlad III Dracula, Vlad III, and Vlad.

in the main matter we see: Vlad III Dracula, Vlad III, and Vlad.

We redefine `\NamesFormat` and `\FrontNamesFormat` because it is more common to add extra information with the first mention of a name.

11.1.1 Application: Life Dates

History texts tend to use life dates, regnal dates, and dates when certain figures flourished. We can use name tags for such dates, which can work with all name types. When used with Roman names (Section 5.10), if using name tags for dates, one would put *agnomina* in ⟨*Affix*⟩.

First we must create any name tags that might be used. Whether it is in the preamble or in the body text, the main point is that the tag can only be used if it exists. The tags used below have a leading space because they are printed conditionally in formatting hooks.

We also define a cross-reference "Atatürk". After doing so, we use naming macros with that name, printing formatted names in the text but making no index page entries. That is a result of indexing rules in nameauth.

We add a Boolean flag to the formatting hook that lets us suppress dates in first uses as needed, while normally displaying dates in first uses by default. We set aside the usual formatting of this manual.

```
\documentclass{article}
\input{compat.tex} % Included with nameauth for compatibility.
\usepackage{makeidx}
\usepackage{nameauth}


\makeindex

% Add name tags to names.
\NameAddInfo[George]{Washington}{ (1732--99)}
\NameAddInfo[Mustafa]{Kemal}{ (1881--1938)}
\NameAddInfo{Atat\"urk}{ (in 1934, a special surname)}

% Ensure that Atat\"urk is a cross-reference that
% has no page entries in the index.
\IndexRef{Atat\"urk}{Kemal, Mustafa}

% Manually suppress name tag in ‘‘first’’ instance
\newif\ifNoTag

% Redesign formatting hook to usually print a tag
% only in ‘‘first’’ instance. On exit, It resets
% the flag that suppresses tags, making that flag
% work only once per name use.

\renewcommand*\NamesFormat[1]
{%
  #1\unless\ifNoTag\NameQueryInfo{}\fi
  \global\NoTagfalse%
}

\begin{document}

```

Up to now there has been a lot of setup. The payoff, however, comes in the main body text where the naming macros do not need manual additions:

```
\ForgetThis\Name[George]{Washington} held office as
the first US president from 1789 to 1797.
\Name[George]{Washington} was the only president whose
term in office was completely in the eighteenth century.

If we need to trigger the first use hook at some point,
we can suppress dates and get an automatic long instance via:
\NoTagtrue\ForgetThis\Name[George]{Washington}.

Or we can trigger the first-use hook in a subsequent name use
and still have dates: \ForceName\Name[George]{Washington}.

```

George Washington (1732–99) held office as the first US president from 1789 to 1797. Washington was the only president whose term in office was completely in the eighteenth century.

If we need to trigger the first use hook at some point, we can suppress dates and get an automatic long instance via: George Washington.

Or we can trigger the first-use hook in a subsequent name use and still have dates: Washington (1732–99).

Since we already have 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 entries in the index:

```
We can add name info tags to names used only as cross-%
references. For example, \Name[Mustafa]{Kemal} was granted
the name \Name{Atat\"urk}.

We mention \Name[Mustafa]{Kemal}
and \Name{Atat\"urk} again. Likewise, we can trigger a
first use, but with no name tag:
\NoTagtrue\ForgetThis\Name{Atat\"urk}.

\printindex
\end{document}
```

We can add name info tags to names used only as cross-references. For example, Mustafa Kemal (1881–1938) was granted the name Atatürk (in 1934, a special surname).

We mention Kemal and Atatürk again. Likewise, we can trigger a first use, but with no name tag: Atatürk.

11.2 Alternate Formatting

Alternate formatting is a framework that isolates local name changes in the text, especially to ⟨*SNN*⟩, from the name forms used globally in index entries. This allows some European or "Continental" academic formatting.

- Sections 11.2.1 amd 11.2.2 show how to enable and disable alternate formatting in either an activated or deactivated state.
- Section 11.2.3 shows how `\CapThis` works only through its helper macro, `\AltCaps`, when alternate formatting is enabled.
- Section 11.2.4 shows how some built-in formatting macros work in formatting hooks along with the helper macros `\AltOn` and `\AltOff`.

> Alternate formatting requires `\noexpand` before macros in name arguments to isolate all local changes in the formatting hooks from the global context, thus protecting index entries.
>
> Every user-defined state change in custom macros used for alternate formatting requires two Boolean flags: one for the global state, and one for the local state.
>
> Names that depend on alternate formatting cannot be used without it. Otherwise, one will see incorrect name forms in the text and spurious entries in the index.

11.2.1 Enabling and Disabling

Alternate formatting involves two separate things: capitalization and name formatting. Thus, we have two "switches" to manage it. The first is like the ignition switch of an automobile that either "enables" or "disables" the whole car. Alternate capitalization will work only while the auto is running.

The second switch is like the manual clutch pedal of an auto. When one depresses the clutch, the output of the motor does not transfer to the wheels. It is "deactivated"; the alternate formatting macros do not format their arguments.

If one lets out the clutch, the power will transfer to the wheels and the car will be "activated" and start moving. Such activation happens when the alternate formatting macros format their arguments.

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

- **Enabled** means that the alternate formatting mechanism inhibits the normal behavior of `\CapThis` and permits both `\AltCaps` and the local changes in formatting hooks triggered by `\AltOn`.
- **Disabled** means that `\CapThis` again works normally and alternate formatting macros (`\AltCaps`, `\AltOn`, and `\AltOff`) are inhibited.
- **Activated** means that built-in alternate formatting macros like `\textSC` globally format their arguments, except when turned off in formatting hooks by `\AltOff`.
- **Deactivated** means that built-in alternate formatting macros like `\textSC` globally do not format their arguments, except when turned on in formatting hooks by `\AltOn`.

`\AltFormatActive` Both the `altformat` option and `\AltFormatActive` enable and globally activate alternate formatting. `\CapThis` works via `\AltCaps`. Built-in formatting macros will format their arguments. `\AltFormatActive` countermands `\AltFormatActive*`.

`\AltFormatActive*` The starred form `\AltFormatActive*` enables alternate formatting but leaves it in a globally deactivated state. It countermands both the `altformat` option and `\AltFormatActive`. It causes `\CapThis` to work via `\AltCaps`. Built-in formatting macros will not format their arguments.

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

> These alternate formatting macros have global scope.

11.2.2 Formatting Environment

The sections in this manual can get complicated. Some quotations use alternate formatting. Some sections also use it, then switch back to regular formatting.

Although many people will use either regular or alternate formatting, but not both, we have added environments to aid the use of text designed for alternate

formatting into an arbitrary context, without worrying if the switching macros have set the proper state. `AltFormatZone` works like `\AltFormatActive`, while `AltFormatZone*` works like `\AltFormatActive*`. When they exit, they disable alternate formatting. Thus, they can serve as a wrapper.

`AltFormatZone` (*env.*)
`AltFormatZone*` (*env.*)

`\FixateFormat`

These format environments create a local scope. `\FixateFormat` works only within `AltFormatZone` and `AltFormatZone*` to prevent the format switches from making any changes. `\FixateFormat` makes the environments the sole controller of alternate formatting within their scope. Below we show the use of `AltFormatZone` and `\FixateFormat`. Hooks do no formatting; the alternate formatting macros do that instead (Section 11.2.4). `\SkipIndex` prevents spurious index entries:

```
\FixateFormat Format cannot be fixated here.

\AltFormatActive
\qquad \Name*[Konrad]{\noexpand\textSC{Zuse}};
formatting is enabled and active.

\AltFormatActive*
\qquad \SkipIndex \Name*[Konrad]{\noexpand\textSC{Zuse}};
formatting is enabled and inactive.
\AltFormatInactive \hfill Formatting disabled.

\begin{AltFormatZone}
  Zone is active; format is not fixated.

  \qquad \Name*[Konrad]{\noexpand\textSC{Zuse}};
  formatting is enabled and active.

  \AltFormatActive*
  \qquad \SkipIndex \Name*[Konrad]{\noexpand\textSC{Zuse}};
  formatting is enabled and inactive.
\end{AltFormatZone} \hfill Formatting disabled.

\begin{AltFormatZone}
  \FixateFormat Format is fixated as: enabled and active.

  \qquad \Name*[Konrad]{\noexpand\textSC{Zuse}};
  formatting is enabled and active.

  \AltFormatActive*
  \qquad \SkipIndex \Name*[Konrad]{\noexpand\textSC{Zuse}};
  formatting is still enabled and active.
\end{AltFormatZone} \hfill Formatting disabled.
```

Format cannot be fixated here.

Konrad Zuse; formatting is enabled and active.

Konrad Zuse; formatting is enabled and inactive. Formatting disabled.

Zone is active; format is not fixated.

Konrad Zuse; formatting is enabled and active.

Konrad Zuse; formatting is enabled and inactive. Formatting disabled.

Format is fixated as: enabled and active.

Konrad Zuse; formatting is enabled and active.

Konrad Zuse; formatting is still enabled and active. Formatting disabled.

11.2.3 Alternate Capitalization

We begin a big region of alternate formatting

Previously, we referred to potential problems using `\CapThis`. When alternate formatting is enabled, `\CapThis` changes its mechanism to avoid such problems. This aspect of alternate formatting does not engage the formatting hooks.

`\AltCaps`

Using the aid of the helper macro `\AltCaps`, `\CapThis` will cause `\AltCaps` to capitalize its argument only in a formatting hook. `\AltCaps` is enabled whenever alternate formatting is enabled, but it works independently of both `\AltOn` and `\AltOff`, which are covered in the next section. `\AltCaps` does not partition its argument, which avoids any problems with `\CapThis`. We describe the syntax:

`\noexpand\AltCaps{`⟨*Argument*⟩`}`

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

\ForgetThis\Cath\ was a powerful queen of France.
\CapThis\LCath[\noexpand\AltCaps{d}e']
was blamed for the St.\ Bartholomew's Day massacre.
```

Catherine de' Medici was a powerful queen of France. De' Medici was blamed for the St. Bartholomew's Day massacre.

One major difference between formatting regimes is that, with regular formatting, `\CapThis` will cause the core name engine to capitalize every name element because we do not know which one to display. With alternate formatting, `\AltCaps` only capitalizes its argument. That causes `\CapThis` to work in a more granular way.

11.2.4 Formatting Features

This part of alternate formatting works within the formatting hooks. First, we show the basic concept where macros do not change between the local scope of the hooks and the global scope of the index. After that, we show where the macros do change.

`\textSC` `\textIT` `\textBF` `\textUC`

Using alternate formatting can be as easy as using any one of four predefined macros. These macros are analogous to the predefined formatting hooks that are accessible via package options. They **always format** their arguments when using the `altformat` option or `\AltFormatActive`. They **never format** their arguments when `\AltFormatActive*` is used or alternate formatting is disabled. `\CapName`, `\RevComma`, and `\RevName` can modify names that use these hooks, but the results appear only in the text.

There are subtle differences among these macros. `\textIT` and `\textBF` use font commands to make one font change. `\textUC` makes no font changes; it just calls `\MakeUppercase`. `\textSC`, however, makes two font changes. It first makes a roman font declaration, then calls `\textsc`. This mitigates font substitution warnings.

No Local Changes

Here we show what happens when these formatting macros are used with `\AltFormatActive`. One should use `\noexpand` before these macros to prevent spurious index entries. Assuming that we have sorted the following names with `\PretagName` (Section 7.6), we get the following with this manual's formatting:

```
\Name[Konrad]{\noexpand\textSC{Zuse}};
\Name[Konrad]{\noexpand\textSC{Zuse}}\\
\Name[Ada]{\noexpand\textIT{Lovelace}};
\Name[Ada]{\noexpand\textIT{Lovelace}}\\
\Name[Charles]{\noexpand\textBF{Babbage}};
\Name[Charles]{\noexpand\textBF{Babbage}}\\
\Name{\noexpand\textUC{Kanade}, Takeo};
\Name{\noexpand\textUC{Kanade}, Takeo}
```

Konrad Zuse; Zuse
Ada *Lovelace*; *Lovelace*
Charles **Babbage**; **Babbage**
KANADE Takeo; KANADE

Local Changes

Now we permit the macros above to change their output within the formatting hooks. Using `\noexpand` is necessary to avoid unwanted index entries.

`\AltOff` Recalling our automobile analogy, `\AltOff` is like depressing the clutch, It prevents macros like `\textSC`, `\textBF`, `\textIT`, and `\textUC` from formatting their arguments, but only within a formatting hook.

`\AltOn` Likewise, `\AltOn` is like letting out the clutch. It allows macros like `\textSC`, `\textBF`, `\textIT`, and `\textUC` to format their arguments, but only within a formatting hook. The index reflects only the global state.

If one uses the `altformat` option, `\AltFormatActive`, or `AltFormatZone`, the formatting "power" goes to the formatting macros by default in both the text and the index. If we want name forms to change in the text, we can add a formatting hook that uses `\AltOff` to suppress formatting in subsequent uses of a name (even when using `\FixateFormat`), without affecting the index:

```
\documentclass{article}
\input{compat.tex} % Included with nameauth for compatibility.
\usepackage{makeidx}
\usepackage[altformat]{nameauth}

\makeindex

\PretagName[Konrad]{\noexpand\textSC{Zuse}}{Zuse, Konrad}
\PretagName[Ada]{\noexpand\textIT{Lovelace}}{Lovelace, Ada}
\PretagName[Charles]{\noexpand\textBF{Babbage}}
  {Babbage, Charles}
\PretagName{\noexpand\textUC{Kanade}, Takeo}{Kanade Takeo}
\renewcommand*\MainNameHook{\AltOff}

\begin{document}

```

```
\begin{quote}
\ForgetThis\Name[Konrad]{\noexpand\textSC{Zuse}};
\Name[Konrad]{\noexpand\textSC{Zuse}}\\
\ForgetThis\Name[Ada]{\noexpand\textIT{Lovelace}};
\Name[Ada]{\noexpand\textIT{Lovelace}}\\
\ForgetThis\Name[Charles]{\noexpand\textBF{Babbage}};
\Name[Charles]{\noexpand\textBF{Babbage}}\\
\ForgetThis\Name{\noexpand\textUC{Kanade}, Takeo};
\Name{\noexpand\textUC{Kanade}, Takeo}
\end{quote}

\printindex
\end{document}
```

Konrad Zuse; Zuse
Ada *Lovelace*; Lovelace
Charles **Babbage**; Babbage
KANADE Takeo; Kanade

11.2.5 History Text

```
\documentclass{article}
\input{compat.tex} % Included with nameauth for compatibility.
\usepackage{makeidx}
\usepackage[altformat]{nameauth}

\makeindex

\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}

\renewcommand*\MainNameHook{\AltOff}

\begin{document}

\ForgetThis\Luth\ was a leading figure in the Protestant
Reformation. \Luth\ believed that one is declared
righteous in a forensic sense by divine grace through faith
created by the Holy Spirit via the Gospel and the Sacraments.

\ForgetThis\Cath\ was not only Queen of France in her own right,
but she also guided the reigns of her three sons.
\CapThis\LCath[\noexpand\AltCaps{d}e']
was blamed for the St.\ Bartholomew's Day massacre that saw the
murder of thousands of Huguenots.

\printindex
\end{document}
```

> Martin Luther was a leading figure in the Protestant Reformation. Luther believed that one is declared righteous in a forensic sense by divine grace through faith created by the Holy Spirit via the Gospel and the Sacraments.
>
> Catherine de' Medici was not only Queen of France in her own right, but she also guided the reigns of her three sons. De' Medici was blamed for the St. Bartholomew's Day massacre that saw the murder of thousands of Huguenots.

Here we keep the particle with the forename, but display them as a long reference with the surname, showing a degree of flexibility (Section 5.7):

- de' Medici `\LCath[\noexpand\AltCaps{d}e']`
- De' Medici `\CapThis\LCath[\noexpand\AltCaps{d}e']`

11.2.6 Inflected Names

Next we design grammatical inflections for use with alternate formatting. Perhaps this could be useful also for paradigms in a grammar book.

We need two Boolean flags for one change in name form, which is a grammatical inflection in this case. Thus, we set up `\ifGenitive` as the global flag and `\ifDoGenitive` as the local flag.

`\DoGenitivetrue` occurs only in the formatting hook. The macro that produces the genitive (or possessive) ending only does so when `\ifDoGenitive` is true. This keeps the index entries consistent via `\noexpand`. Likewise, `\AltOff` only occurs in the formatting hook `\MainNameHook`.

```
\documentclass{article}
\input{compat.tex} % Included with nameauth for compatibility.
\usepackage{makeidx}
\usepackage[altformat]{nameauth}

\makeindex

\newif\ifGenitive
\newif\ifDoGenitive
\newcommand*\GEN{\ifDoGenitive\textSC{'s}\fi}

\begin{nameauth}
  \< Jeff & Thomas &
     \noexpand\textSC{Jefferson}\noexpand\GEN{} & >
\end{nameauth}

\PretagName[Thomas]{\noexpand\textSC{Jefferson}\noexpand\GEN{}}
  {Jefferson, Thomas}
\TagName[Thomas]{\noexpand\textSC{Jefferson}\noexpand\GEN{}}
  {, pres.}

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

\begin{document}

```

```
Consider \Genitivetrue\Jeff\ legacy as the author of the
colonies' \textit{Declaration of Independence} and his impact
as third president of the United States. \Jeff\ was a complex
historical figure whose actions defy a consistent moral compass
both in public policy and in personal affairs.

\printindex
\end{document}
```

Consider Thomas Jefferson's legacy as the author of the colonies' *Declaration of Independence* and his impact as third president of the United States. Jefferson was a complex historical figure whose actions defy a consistent moral compass both in public policy and in personal affairs.

11.2.7 Reference Work I

Here we show how nameauth might be used in a reference work, where names are also the head-words of articles. We present examples that include a basic Western name, a basic Eastern name, and a Western name with an alias. Some tips include:

- Sort all names that use alternate formatting.
- Match index entry forms to article head-words.
- Cross-reference name variants to the head-word.
- The first instance of a name displays the active alternate formatting.
- Deactivate alternate formatting in subsequent name instances.
- Since some LaTeX fonts do not combine small caps and boldface, we use slanted text in `\RefArticle` below.

```
\documentclass{article}
\input{compat.tex} % Included with nameauth for compatibility.
\usepackage{makeidx}
\usepackage[altformat]{nameauth}

\makeindex

% Sort any names with macros in the arguments.

\PretagName[Greta]{\noexpand\textSC{Garbo}}{Garbo, Greta}
\PretagName{\noexpand\textSC{Misora}, Hibari}{Misora Hibari}
\PretagName[Heinz]{\noexpand\textSC{R\"uhmann}}{Ruehmann, Heinz}
\PretagName[Heinrich Wilhelm]{\noexpand\textSC{R\"uhmann}}
  {Ruehmann, Heinrich Wilhelm}

% Make a cross-reference from a variant name form to the
% form of the head-words

\IndexRef[Heinrich Wilhelm]{\noexpand\textSC{R\"uhmann}}
  {\noexpand\textSC{R\"uhmann}, Heinz}

% Define the formatting hooks. Since we use the 'altformat'
% option, alternate formatting is turned off in later
% name uses.

```

```
\renewcommand*\MainNameHook{\AltOff}

% Typeset head-words with a slanted font.

\newcommand{\RefArticle}[3]
{%
  \def\Check{#2}%
  \ifx\Check\empty
    \noindent\ForgetThis\textsl{#1}\ #3
  \else
    \noindent\ForgetThis\textsl{#1}\ #2\ #3
  \fi\medskip
}

\begin{document}

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

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

\RefArticle
  {\RevComma\Name[Heinz]{\noexpand\textSC{R\"uhmann}}}
  {(\SubvertThis\ForceName
    \FName[Heinrich Wilhelm]{\noexpand\textSC{R\"uhmann}};}
  {7 March 1902\,--\,3 October 1994) was a German actor
   in over 100 films.
   \Name[Heinz]{\noexpand\textSC{R\"uhmann}}\dots}

\printindex
\end{document}
```

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

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

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

We will revisit the topic of a reference work in Section 11.4.1, where we integrate custom formatting hooks into the concept. This puts the work into setup and automation instead of using macros with several arguments.

11.3 Roman Names II

11.3.1 General Reference Works

In this subsection and the next, we do not use the formatting conventions of this manual and we activate alternate formatting. We focus here on reference works that treat the *cognomen* as a Western surname and index according to it.[49]

This approach corresponds roughly with the manual approach that we saw in Section 5.10.1, except that we will not use ⟨*Affix*⟩ to drop *agnomina* and we will not use ⟨*Alternate*⟩ to display variants of the praenomen and *nomen*. Instead, we use macros that take advantage of alternate formatting to handle name form changes.

> Index: ⟨*cognomen*⟩ ⟨*agnomen*⟩, ⟨*praenomen*⟩ ⟨*nomen*⟩
> Macro: `\Name[`⟨*praenomen*⟩ ⟨*nomen*⟩`]{`⟨*cognomen*⟩ ⟨*agnomen*⟩`}`

This method increases the number of macros by **two per name,** one for each individual ⟨*FNN*⟩ and ⟨*SNN*⟩. That might be a deal-breaker for some. The general logic used by the respective macros used in each name component follows the plan below, with the `nameauth` logic for Western names doing the rest:

- Display either *praenomen*, *nomen*, or both in ⟨*FNN*⟩.
- Display either *cognomen*, *agnomen*, or both in ⟨*SNN*⟩.

We use alternate formatting and `\noexpand` in the macro arguments to ensure that index entries remain consistent. We define macros in both ⟨*FNN*⟩ and ⟨*SNN*⟩ that expand conditionally. We use the quick interface to define name shorthands, and we use `\PretagName` and `\TagName` as needed (cf. Section 5.9. In the preamble we define all macros and conditionals used in naming macro arguments to avoid errors. We use `\ifNoGens` for the *nomen* because `\ifNoNomen` invites haplography.

> Address any concerns about expansion of macros in name arguments by using `\noexpand` before those macros.

- When using Boolean flags (`\if`⟨*flag*⟩) in the macros that represent name elements, ensure that those flags only change their value within the local scope of the name formatting hooks (Section 9.1).
- Two Boolean flags are needed for each automated variant in the name. One flag reflects the **global** state and the name form in the index. The other reflects the **local** state of the formatting hooks.
- Since we have four name **components**, we need at most eight Boolean flags, two per **potential change**. The logic shows longer forms when the flags are false, shorter forms when the flags are true.

[49] 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**.

```
\documentclass{article}
\input{compat.tex} % Included with nameauth for compatibility.
\usepackage{makeidx}
\usepackage[altformat]{nameauth}

\makeindex

% Global Boolean flags need to be defined only once.
\newif\ifNoPraenomen
\newif\ifNoCognomen
\newif\ifNoGens
\newif\ifNoAgnomen

% Local Boolean flags need to be defined only once.
\newif\ifXPrae
\newif\ifXCogn
\newif\ifXGens
\newif\ifXAgno

```

For every name, one must define two macros. True, there can be multiple names that are alike and use the same macros, as with many a Cato, but then one must use `\TagName` and non-printing macros in name arguments to create unique names and index entries (Section 6.3). The Romans had a small set of acceptable names and often named fathers and sons alike.

```
% Name variant macros are defined uniquely for each name.
\newcommand*\SCIPi
{%
  \ifXGens Publius\else
    \ifXPrae Cornelius\else
      Publius Cornelius%
    \fi
  \fi
}

\newcommand*\SCIPii
  {\ifXAgno Scipio\else Scipio Africanus\fi}

\newcommand*\TSemp
{%
  \ifXGens Tiberius\else
    \ifXPrae Sempronius\else
      Tiberius Sempronius%
    \fi
  \fi
}

% Quick interface definitions.
\begin{nameauth}
  \< Scipio & \noexpand\SCIPi & \noexpand\SCIPii & >
  \< TGrac & \noexpand\TSemp & Gracchus & >
\end{nameauth}

```

If one uses `\NameQueryInfo{}` within formatting hooks, it is best to put a space at the beginning of a tag if space is needed, instead of in the hook.

```
% We add a name tag.
\NameAddInfo[\noexpand\TSemp]{Gracchus}
  { (consul, 177 \textsc{bc})}

% Sorting and tagging the names
\PretagName[\noexpand\SCIPi]{\noexpand\SCIPii}
  {Scipio Africanus}
\PretagName[\noexpand\TSemp]{Gracchus}
  {Gracchus, Tiberius Sempronius}
\TagName[\noexpand\TSemp]{Gracchus}{, consul}

```

We reset the global Boolean flags to ensure the longest name forms in the index. Local Boolean flags revert to false outside the scope of the formatting hooks.

```
% We define formatting hooks.
\renewcommand*\NamesFormat[1]
{%
  \ifNoPraenomen\XPraetrue\fi%
  \ifNoGens\XGenstrue\fi%
  \ifNoCognomen\XCogntrue\fi%
  \ifNoAgnomen\XAgnotrue\fi%
  #1\NameQueryInfo{}%
  \global\NoPraenomenfalse%
  \global\NoGensfalse%
  \global\NoCognomenfalse%
  \global\NoAgnomenfalse%
}

\renewcommand*\MainNameHook[1]
{%
  \ifNoPraenomen\XPraetrue\fi%
  \ifNoGens\XGenstrue\fi%
  \ifNoCognomen\XCogntrue\fi%
  \ifNoAgnomen\XAgnotrue\fi%
  #1%
  \global\NoPraenomenfalse%
  \global\NoGensfalse%
  \global\NoCognomenfalse%
  \global\NoAgnomenfalse%
}

\begin{document}

\NoAgnomentrue\Scipio\ was born around 236 \textsc{bc}
into the Scipiones branch of the Cornelii clan.
\NoAgnomentrue\Scipio\ rose to military fame during the
Second Punic War. Thereafter he was known as \Scipio.
He flourished during the Egyptian reigns of
\Name{Ptolemy, IV}[IV Philopator] and
\Name{Ptolemy, V}[V Epiphanes], and the Syrian
reigns of \Name{Seleucus, III}[III Ceraunus] and
\Name{Antiochus, III}[III the Great].

```

Publius Cornelius Scipio was born around 236 BC into the Scipiones branch of the Cornelii clan. Scipio rose to military fame during the Second Punic War. Thereafter he was known as Scipio Africanus. He flourished during the Egyptian reigns of Ptolemy IV Philopator and Ptolemy V Epiphanes, and the Syrian reigns of Seleucus III Ceraunus and Antiochus III the Great.

Below we show more name variations than were shown above:

In Section 5.9, we used name tags instead of ⟨*Alternate*⟩ in the first-use formatting hook. Here we adopt a similar approach. Since formatting hooks are local by design, whatever changes inside of them does not affect the index.

```
\TGrac\ served as tribune of the plebs in 184 \textsc{bc}.
\NoGenstrue\STGrac\ was elected praetor for 180 \textsc{bc},
after which he was appointed governor of Hispania Citerior,
serving with the rank of proconsul. In 177 \textsc{bc},
he was elected consul, again in 163 \textsc{bc}.

\printindex
\end{document}
```

Tiberius Sempronius Gracchus (consul, 177 BC) served as tribune of the plebs in 184 BC. Gracchus was elected praetor for 180 BC, after which he was appointed governor of Hispania Citerior, serving with the rank of proconsul. In 177 BC, he was elected consul, again in 163 BC.

Again we see more name variations:

Next we see other types of names using these same formatting hooks:

11.3.2 Scholarly Reference Works

We retain most of the code from the previous section in the example below, but we change the macros used in the name arguments, following this logic:

- Display the *praenomen* in ⟨*FNN*⟩:
- Display one of the following in ⟨*SNN*⟩:
  - Only the *cognomen*
  - Both the *cognomen* and *agnomina*
  - Only the *nomen*
  - Both the *nomen* and *cognomen*
  - The *nomen*, *cognomen*, and *agnomina*

```
\documentclass{article}
\input{compat.tex} % Included with nameauth for compatibility.
\usepackage{makeidx}
\usepackage[altformat]{nameauth}

\makeindex

% Global Boolean flags need to be defined only once.
\newif\ifNoPraenomen
\newif\ifNoCognomen
\newif\ifNoGens
\newif\ifNoAgnomen

% Local Boolean flags need to be defined only once.
\newif\ifXPrae
\newif\ifXCogn
\newif\ifXGens
\newif\ifXAgno

% Name variant macros are defined uniquely for each name.
\newcommand*\CSB
{%
  \ifXGens
    \ifXAgno Scipio\else Scipio Barbatus\fi
  \else
    \ifXCogn Cornelius\else
      \ifXAgno Cornelius Scipio\else
        Cornelius Scipio Barbatus%
      \fi
    \fi
  \fi
}

% Quick interface definition
\begin{nameauth}
  \< OScipio & Lucius & \noexpand\CSB & > % O for Oxford
\end{nameauth}

```

```
% Sorting and tagging
\PretagName[Lucius]{\noexpand\CSB}{Cornelius Scipio Barbatus}
\TagName[Lucius]{\noexpand\CSB}{, consul}

\renewcommand*\NamesFormat[1]
{%
  \ifNoPraenomen\XPraetrue\fi%
  \ifNoGens\XGenstrue\fi%
  \ifNoCognomen\XCogntrue\fi%
  \ifNoAgnomen\XAgnotrue\fi%
  #1\NameQueryInfo{}%
  \global\NoPraenomenfalse%
  \global\NoGensfalse%
  \global\NoCognomenfalse%
  \global\NoAgnomenfalse%
}

\begin{document}

\OScipio\ was born around 337 \textsc{bc} into the
Scipiones branch of the Cornelii clan, one of the large
patrician clans. \NoGenstrue\NoAgnomentrue\OScipio\ was
one of the two elected consuls in 298 \textsc{bc}
and served during the Third Samnite War.

\printindex
\end{document}
```

Lucius Cornelius Scipio Barbatus was born around 337 BC into the Scipiones branch of the Cornelii clan, one of the large patrician clans. Scipio was one of the two elected consuls in 298 BC and served during the Third Samnite War.

Instead of relying on some nameauth features that drop some name elements automatically, in all but ⟨*FNN*⟩ we have to state explicitly what we want:

Lucius Cornelius Scipio Barbatus ..... `\ForgetThis\OScipio`
Lucius Cornelius Scipio Barbatus ..... `\LOScipio`
Cornelius Scipio Barbatus ..... `\OScipio`
Lucius ..... `\SOScipio`
Cornelius ..... `\NoCognomentrue\OScipio`
Cornelius Scipio ..... `\NoAgnomentrue\OScipio`
Scipio Barbatus ..... `\NoGenstrue\OScipio`
Scipio ..... `\NoGenstrue\NoAgnomentrue\OScipio`

I have often thought that nothing would do more extensive good at small expense than the establishment of a small circulating library in every county, to consist of a few well-chosen books, to be lent to the people of the country under regulations as would secure their safe return in due time.

—Thomas Jefferson
letter to John Wyche, 1809

11.4 Special Name Uses

Previously, formatting hooks either changed typefaces or mutated their arguments. Now we move beyond parsing the argument as we have received it.

\NameParser

The user-accessible parser (Section 13.7.6) is distinct from the internal name parser. Yet it only can be called within the formatting hooks, the point where the internal parser generates output. Thus, the two parsers are linked.

Capitalization and ⟨*Alternate*⟩ swapping happen early in the parsing process before we know what name elements to display. `\NameParser` only sees the name elements after these actions are done. It displays a printed name based on the elements it sees and a matrix of Boolean flags:

- When `\if@nameauth@FullName` is true:
  - Name forms are long.
  - If true, `\if@nameauth@ShortSNN` causes the affix of a Western surname to not be displayed.
  - If true, `\if@nameauth@RevThis` causes name order to be reversed, as applicable.
  - If true, `\if@nameauth@RevThisComma` causes a Western name to be reversed, adding a comma between ⟨*SNN*⟩ and ⟨*FNN*⟩ only if `\if@nameauth@RevThis` is false.
- When `\if@nameauth@FullName` is false:
  - Name forms are short.
  - If true, `\if@nameauth@ForceAffix` causes only the suffix of a Western name to print.
  - If true, `\if@nameauth@FirstName` causes only a Western forename to print.
  - If true, `\if@nameauth@FirstName` causes an Eastern personal name or an ancient sobriquet to print, but only if `\if@nameauth@EastFN` is true.

\ifNameauthWestern
\ifNameauthObsolete

We provide an example that uses `\NameParser`, `\ifNameauthWestern`, and `\ifNameauthObsolete` (Section 6.2) to show and evaluate name parts:

```
\makeatletter
\renewcommand\MainNameHook[1]{%
  \@nameauth@FullNamefalse%
  \@nameauth@FirstNamefalse%
  \@nameauth@ForceAffixfalse%
  \@nameauth@EastFNtrue%
  \ifNameauthWestern
    Western name:\\
    \hbox to 3.5cm {Forename: \@nameauth@FirstNametrue%
      \NameParser\hfill}
    \hbox to 3.5cm {Surname:  \@nameauth@FirstNamefalse%
      \NameParser\hfill}
    Affix: \@nameauth@ForceAffixtrue \NameParser\smallskip
```

```
  \else
    \hbox to 3.5cm{Non-Western name:\hfill}
    \ifNameauthObsolete
      (obsolete syntax)\\ \hbox to 3.5cm{\hfill} \fi
    \hbox to 3.5cm {Name: \@nameauth@FirstNamefalse%
      \NameParser\hfill}
    Person/Sobriquet: \@nameauth@FirstNametrue%
      \NameParser\smallskip
  \fi
}
\makeatother

\Name[Martin]{Van Buren}\\
\Name[George S.]{Patton, Jr.}[George]\\
\Name{Miyazaki, Hayao}\\
\Name{Henry}[VIII]\\
\Name{Confucius}
```

Western name:		
Forename: Martin	Surname: Van Buren	Affix:
Western name:		
Forename: George	Surname: Patton	Affix: Jr.
Non-Western name:	Name: Miyazaki	Person/Sobriquet: Hayao
Non-Western name:	(obsolete syntax) Name: Henry	Person/Sobriquet: VIII
Non-Western name:	Name: Confucius	Person/Sobriquet:

11.4.1 Reference Work II

Below we revisit the idea of a reference work, using the formatting hooks to automate tasks instead of populating the different arguments of `\RefArticle` in Section 11.2.7. We shift the work from proofreading to setup.

```
\documentclass{article}
\input{compat.tex} % Included with nameauth for compatibility.
\usepackage{makeidx}
\usepackage[altformat]{nameauth}

\makeindex

% Boolean flags; the first sets up headwords and the second
% indicates that a Non-Western form should not be reversed.
\newif\ifHeadword
\newif\ifAncientName

% Sorting and tagging the names:
\PretagName[Charles]{\noexpand\textBF{Babbage}}{Babbage, Charles}
\PretagName{\noexpand\textUC{Kanade}, Takeo}{Kanade Takeo}
\PretagName[Ada]{\noexpand\textIT{Lovelace}}{Lovelace, Ada}

```

```
% Adding name information:
\NameAddInfo{Aristotle}{ (384--322 \textsc{bc})}
\NameAddInfo[Charles]{\noexpand\textBF{Babbage}}{ (1791--1871)}
\NameAddInfo{\noexpand\textUC{Kanade}, Takeo}{ (1945-- )}
\NameAddInfo[Ada]{\noexpand\textIT{Lovelace}}
 { (Augusta Ada King, Countess of Lovelace
   [n\'ee Byron]; 1815--52)}

```

The formatting hook does all the heavy lifting. If the name appears in the headword part of `\RefArticle`, the hook reverses the name with commas only if it is a Western name. Otherwise, if it is a Non-Western name, but not ancient or royal, then both the Eastern name and its Western version are displayed. Only headwords are printed in boldface. After the first name occurrence, forced by the headword part of `\RefArticle`, the name tag also is printed.

```
% Redefining the formatting hooks:
\makeatletter
\renewcommand\NamesFormat[1]
{%
  \ifHeadword
    \ifNameauthWestern
      \@nameauth@RevThisCommatrue%
      \bfseries \NameParser%
      \normalfont\NameQueryInfo{}%
    \else
      \begingroup%
        \bfseries \NameParser%
        \unless\ifAncientName
          \normalfont; W:\AltOff\space
          \@nameauth@RevThistrue \NameParser%
        \fi
        \normalfont\NameQueryInfo{}%
      \endgroup%
    \fi
  \else
    \NameParser%
  \fi
  \global\Headwordfalse%
  \global\AncientNamefalse%
}
\makeatother
\renewcommand\MainNameHook{\AltOff}

```

In Section 11.2.7 the `\RefArticle` macro took three arguments, one of which frequently was empty, and required information like dates to be put at the front of each entry. Now `\RefArticle` simply takes two arguments: a name and some information. The rest is automated.

```
% Define related macros:
\newcommand\Headword{\Headwordtrue\ForgetThis}
\newcommand{\RefArticle}[2]
  {\noindent\Headword #1 #2\medskip}

```

```
\begin{document}

\RefArticle{\AncientNametrue\Name{Aristotle}}{was the first to offer
a system of logic, most notably syllogistic logic, that would
become the basis for discrete states and circuitry of
digital computers. \Name{Aristotle}\dots}

\RefArticle{\Name[Charles]{\noexpand\textBF{Babbage}}}{designed and
built the Difference Engine and began work on the Analytical
Engine. \Name[Charles]{\noexpand\textBF{Babbage}}\dots}

\RefArticle{\Name{\noexpand\textUC{Kanade}, Takeo}}{is one of the
foremost pioneers in the field of computer vision.
\Name{\noexpand\textUC{Kanade}, Takeo}\dots}

\RefArticle{\Name[Ada]{\noexpand\textIT{Lovelace}}}{collaborated with
\Name*[Charles]{\noexpand\textBF{Babbage}}* and wrote what some
consider to be the first computer program for the Analytical
Engine. \Name[Ada]{\noexpand\textIT{Lovelace}}\dots}

\printindex
\end{document}
```

Aristotle (384–322 BC) was the first to offer a system of logic, most notably syllogistic logic, that would become the basis for discrete states and circuitry of digital computers. Aristotle…

Babbage, Charles (1791–1871) designed and built the Difference Engine and began work on the Analytical Engine. Babbage…

KANADE Takeo; W: Takeo Kanade (1945–) is one of the foremost pioneers in the field of computer vision. Kanade…

***Lovelace*, Ada** (Augusta Ada King, Countess of Lovelace [née Byron]; 1815–52) collaborated with Charles Babbage* and wrote what some consider to be the first computer program for the Analytical Engine. Lovelace…

‘Tis but thy name that is my enemy;
Thou art thyself, though not a Montague.
What’s Montague? It is nor hand, nor foot,
Nor arm, nor face. O, be some other name
Belonging to a man.
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 (published 1597)

11.4.2 Marginalia

This example further illustrates aspects of automation be it practical or not. We begin in the document preamble by defining Boolean flags and macros. As with Roman names, their default values produce the name form in the index entry.

The default name form aligns with the default Boolean flag setting: false. All non-default values and expansions of these macros occur only in the formatting hooks. This design helps to prevent spurious index entries.

We must use `\PretagName` to sort the names in the index. `\Revert` is used to print a specific form of the surname (via the macros in the naming arguments) as well as suppressing the margin note (via the formatting hooks). Thus we show a complex interplay between syntactic name forms and typographic name formatting.

```
\documentclass{article}
\input{compat.tex} % Included with nameauth for compatibility.
\usepackage{makeidx}
\usepackage[altformat]{nameauth}

\makeindex

% Global Boolean flags need to be defined only once.
\newif\ifSpecialFN
\newif\ifSpecialSN
\newif\ifRevertSN

% Name variant macros must be defined for each name.
% For a long name, we want to use ``William'' in the text
% and ``Wm.'' in the margin.

\newcommand*\WM{\ifSpecialFN Wm.\else William\fi}

% The first surname use will be ``Shakespeare'', but ``the
% Bard'' thereafter. We allow for alternate caps.
% We can get ``Shakespeare'' thereafter by toggling a flag.

\newcommand*\SHK
{%
  \ifRevertSN
    \textSC{Shakespeare}\else
    \ifSpecialSN
      \noexpand\AltCaps{t}he Bard\else
      \textSC{Shakespeare}%
    \fi
  \fi
}

% Here is how we toggle that flag.
\newcommand*\Revert{\RevertSNtrue}

% Quick interface name definition:
\begin{nameauth}
  \< Shak & \noexpand\WM & \noexpand\SHK & >
\end{nameauth}

```

```
% Sorting and tagging the name:
\PretagName[\noexpand\WM]{\noexpand\SHK}
  {Shakespeare, William}
\PretagName[Robert]{\textSC{Burns}}{Burns, Robert}

```

Next we define the two formatting hooks that govern changes in the text. We force `\NamesFormat` to print only the default name via `\RevertSNfalse`, `\SpecialFNfalse`, and `\SpecialSNfalse`. `\MainNameHook` disables alternate formatting with `\AltOff`, but it allows variant name forms.

Macros in the name arguments take the role usually played by internal Boolean flags. In the hooks we print the default name in the body text. If allowed, we print a margin paragraph with an alternate full name using `\NameParser`. Both hooks set `\RevertSNfalse` on exit, so that `\Revert` works on a per-name basis.

```
\makeatletter
\renewcommand*\NamesFormat[1]
{%
  \RevertSNfalse\SpecialFNfalse\SpecialSNfalse%
  #1%
  \unless\ifinner
    \marginpar
    {%
      \footnotesize\raggedleft%
      \SpecialFNtrue\SpecialSNfalse%
      \NameParser%
    }%
  \fi
  \global\RevertSNfalse%
}

\renewcommand*\MainNameHook[1]
{%
  \AltOff\SpecialFNfalse\SpecialSNtrue%
  #1%
  \unless\ifinner
    \unless\ifRevertSN
      \marginpar
      {%
        \footnotesize\raggedleft%
        \SpecialFNfalse\SpecialSNfalse%
        \NameParser%
      }%
    \fi
  \fi
  \global\RevertSNfalse%
}
\makeatother

```

```
\begin{document}

\ForgetThis\Shak\ is the national poet of England
in much the same way that \Name[Robert]{\textSC{Burns}}
is that of Scotland. With the latter's rise of influence
in the 1800s, \Revert\Shak\ became known as ``\Shak''.

\printindex
\end{document}
```

Wm. Shakespeare
Robert Burns
Shakespeare

William Shakespeare is the national poet of England in much the same way that Robert Burns is that of Scotland. With the latter's rise of influence in the 1800s, Shakespeare became known as "the Bard".

11.5 Advanced Customization

Here we discuss customization and adding features. We set aside the formatting in this manual, using alternate formatting for much of this section.

11.5.1 Using Package Internals

We start with alternate formatting (Section 11.2), changing the "back-end" macros to our custom code. Here, we need only check `\if@nameauth@DoAlt`, which is toggled by `\AltOn` and `\AltOff`. Instead of using `\textSC` and friends, we define a new macro that works in similar fashion. `\Fbox` draws a frame around the name:

```
\documentclass{article}
\input{compat.tex} % Included with nameauth for compatibility.
\usepackage{makeidx}
\usepackage[altformat]{nameauth}

\makeindex

% Alternate formatting macro definition
\makeatletter
\newcommand*\Fbox[1]{%
  \if@nameauth@DoAlt\protect\fbox{#1}\else#1\fi
}
\makeatother

```

Since `\AltCaps` is part of nameauth, one need not reinvent that wheel. Just use it in the name arguments and sorting macros:

```
% Quick interface definition
\begin{nameauth}
  \< deSmet & Pierre-Jean &
     \noexpand\Fbox{\noexpand\AltCaps{d}e~Smet} & >
\end{nameauth}

% Sorting and tagging
\PretagName[Pierre-Jean]%
  {\noexpand\Fbox{\noexpand\AltCaps{d}e~Smet}}%
  {de~Smet, Pierre-Jean}

```

The final step is redefining the hooks, which can be quite simple:

```
\renewcommand*\MainNameHook{\AltOff}
\renewcommand*\FrontNameHook{\AltOff}

\begin{document}

\deSmet\ was a Jesuit missionary who arrived in North
America in 1821 at the age of twenty, after a year of seminary
education. \CapThis\deSmet\ was ordained in 1827 and worked
among American Indian nations after 1837. We can show the forms
\LdeSmet\ and \SdeSmet.

\printindex
\end{document}
```

Pierre-Jean de Smet was a Jesuit missionary who arrived in North America in 1821 at the age of twenty, after a year of seminary education. De Smet was ordained in 1827 and worked among American Indian nations after 1837. We can show the forms Pierre-Jean de Smet and Pierre-Jean.

11.5.2 Using Name Hooks

One can redesign or augment the core naming macros that handle regular, long, and short names, then hook those changes into the core name engine. It should be obvious that such changes could break nameauth if not done carefully. These macros are set by default to `\@nameauth@Name`, the internal name parser.

`\NameauthName`
- `\Name`, or an unmodified shorthand, calls `\NameauthName`.

`\NameauthLName`
- `\Name*`, or an L-shorthand, sets `\@nameauth@FullNametrue`, then calls `\NameauthLName`.

`\NameauthFName`
- `\FName`, or an S-shorthand, sets `\@nameauth@FirstNametrue`, then calls `\NameauthFName`.

> One should not modify `\Name` and `\FName` directly.

Names In Boxes

Since nameauth uses xparse, the next examples use it also. Here we look at decoration in the sense of putting a name into something around it. This could be useful if, in certain parts of a document, one wanted to turn names into hyperlinks or some other kind of feature. Here we simply put names into colored boxes.[50]

`\global` These macros can be redefined or used locally within a scope without making global changes to the document unless you specifically use `\global`.

[50] After the parsed and formatted 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.

```
\documentclass{article}
\input{compat.tex} % Included with nameauth for compatibility.
\usepackage{makeidx}
\usepackage[altformat]{nameauth}
\usepackage{xcolor}

\makeindex

\makeatletter
% Change the general-case name macro to show
% a name in a framed, colored box.

\NewDocumentCommand{\MyName}{O{} m O{}}
  {\fcolorbox{black}{gray!25!white}{\@nameauth@Name[#1]{#2}[#3]}}

% Likewise change the macro for when names are forced long.
\NewDocumentCommand{\MyLName}{O{} m O{}}
  {\fcolorbox{black}{green!25!white}{\@nameauth@Name[#1]{#2}[#3]}}

% Likewise change the macro when personal names are desired.
\NewDocumentCommand{\MyFName}{O{} m O{}}
  {\fcolorbox{black}{yellow!25!white}{\@nameauth@Name[#1]{#2}[#3]}}
\makeatother

% Change the formatting hooks, but do not use alternate.
% formatting, which is separate from the customization above.
\renewcommand*\NamesFormat{\scshape}
\renewcommand*\MainNameHook{}

% Change the naming macro hooks.
\renewcommand*\NameauthName{\MyName}
\renewcommand*\NameauthLName{\MyLName}
\renewcommand*\NameauthFName{\MyFName}

\begin{document}

\ForgetThis\Name[Adolf]{Harnack} was a theologian who stressed
the Fatherhood of God and the brotherhood of man.
\Name[Adolf]{Harnack} flourished in the early twentieth
century; \Name*[Adolf]{Harnack}[Adolf von]; \FName[Adolf]{Harnack}.

\printindex
\end{document}
```

ADOLF HARNACK was a theologian who stressed the Fatherhood of God and the brotherhood of man. Harnack flourished in the early twentieth century; Adolf von Harnack; Adolf.

Virtue never has been as respectable as money.

—Mark Twain
The Innocents Abroad (1869)

Change Parsing

Section 4.3.1 discussed final optional arguments. Here we use xparse to change the default behavior of optional arguments, but only for TEX distros since May 2018.

```
\makeatletter
\@ifl@t@r\fmtversion{2018/04/30}{\newcommand\nameauthxp{}}{}

\ifdefined\nameauthxp
\NewDocumentCommand{\MyName}{O{} m !O{}}
  {\@nameauth@Name[#1]{#2}[#3]}
\else
\NewDocumentCommand{\MyName}{O{} m O{}}
  {\@nameauth@Name[#1]{#2}[#3]}
\fi
\makeatother

\let\NameauthName\MyName
\let\NameauthLName\MyName
\let\NameauthFName\MyName
\ForgetName[Albert]{Einstein}
\ForgetName{Miyazaki, Hayao}

We want: ``Albert Einstein [first] spoke about physics.
    Miyazaki Hayao [later] spoke about animation.
    Einstein [first] spoke about physics.
    Miyazaki [later] spoke about animation.''

We get: ``\Name[Albert]{Einstein} [first] spoke about physics.
    \Name{Miyazaki, Hayao} [later] spoke about animation.
    \Name[Albert]{Einstein} [first] spoke about physics.
    \Name{Miyazaki, Hayao} [later] spoke about animation.''
```

We want: "Albert Einstein [first] spoke about physics. Miyazaki Hayao [later] spoke about animation. Einstein [first] spoke about physics. Miyazaki [later] spoke about animation."

We get: "Albert Einstein [first] spoke about physics. Miyazaki Hayao [later] spoke about animation. Einstein [first] spoke about physics. Miyazaki [later] spoke about animation."

Back to Table of Contents

We end a big region of alternate formatting

Now, as the storm of war, of revolution and of emotion subsides there is left even with us of the United States much unrest, much discontent with the surer forces of human advancement. To all of us, out of this crucible of actual, poignant, individual experience has come a deal of new understanding, and it is for all of us to ponder these new currents if we are to shape our future with intelligence.

—Herbert Hoover
American Individualism (1922)

12 Obsolete Features

General Hook Redefinition

```
\renewcommand*\FrontNamesFormat{\color{violet}\sffamily}
\renewcommand*\FrontNameHook{\color{darkgray}\sffamily}
\renewcommand*\NamesFormat{\color{blue}\sffamily}
\renewcommand*\MainNameHook{\sffamily}
```

12.1 Pseudonym Macros

The macros described in this section are early features of nameauth. They do not work like other macros that display names. They remain only for backward compatibility, but will not be developed further.

Special Syntax

To save space, we show ⟨*SAFX*⟩ as the equivalent of ⟨*SNN,Affix*⟩. The macros in this section all take arguments of the form:

Target Name	Cross-Reference Name
`[`⟨*FNN*⟩`]{`⟨*SAFX*⟩`}`	`[`⟨*xref FNN*⟩`]{`⟨*xref SAFX*⟩`}[`⟨*xref Alt.*⟩`]`

- The target name comes first, which is the **opposite** of `\IndexRef`. There the target name comes last because it is text passed to the index macros.
- The cross-reference comes last to allow for the widest range of name forms (again, the opposite of `\IndexRef`). We avoid two optional arguments in succession by preventing the target from having a final optional argument. Neither ⟨*Alt.*⟩ nor the obsolete syntax can be used with the target name. Both can be used with the cross-reference.
- ⟨*SAFX*⟩ and ⟨*xref SAFX*⟩ can have comma-delimited suffixes.
- One cannot use `\TagName` with a cross-reference, but one can sort a cross-reference with `\PretagName` (Section 7.6).

This syntax can produce results in the text that can look legitimate, yet the index entries are wrong due to confusion. The normal name arguments make it easier to determine name forms. For example,

- We see William I (William the Conqueror) in the text.
- `\PName*{William, I}[William]{the Conqueror}`
- The syntax obfuscates the fact that the xref has a Western form.
- The index is wrong: “the Conqueror, William *see* William I”

\AKA
\AKA*

The **also known as** macro and its starred form display an alias in the text and create a cross-reference in the index. They format names differently than the naming macros because they use the name patterns for cross-references as a means to account for the name forms that they print in the test.

```
\AKA [⟨FNN⟩]{⟨SAFX⟩}[⟨xref FNN⟩]{⟨xref SAFX⟩}[⟨xref Alt.⟩]
\AKA*[⟨FNN⟩]{⟨SAFX⟩}[⟨xref FNN⟩]{⟨xref SAFX⟩}[⟨xref Alt.⟩]
```

- Both macros create a cross-reference in the index to the target name.
- `\AKA` prints a long form of the cross-reference name in the text. `\SeeAlso` works with `\AKA`, `\AKA*`, `\PName`, and `\PName*`.
- If ⟨*xref Alt.*⟩ is present in a Western name form, then instead of ⟨*xref FNN*⟩, ⟨*xref Alt.*⟩ will be printed in the text.
- If ⟨*xref Alt.*⟩ is present in a Non-Western name form, then instead of ⟨*xref Affix*⟩ (if present), ⟨*xref Alt.*⟩ will be printed in the text.
- If ⟨*xref Alt.*⟩ is present without either ⟨*xref FNN*⟩ or ⟨*xref Affix*⟩, the obsolete syntax is used only with the xref.
- `\AKA*` is analogous to `\FName`. `\ForceFN` works with it. The `oldAKA` option implies `\ForceFN` with every use of `\AKA*`.
- `\AKA`, `\AKA*`, `\PName*`, and `\PName*` do not permit the effects of `\ForgetThis` and `\SubvertThis` to "pass through" because they produce output in the text. The `oldreset` option negates this.

Below we make cross-references to Bob Hope `\Name[Bob]{Hope}`; all of the forms listed create the cross-reference "Hope, Leslie Townes *see* Hope, Bob".

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

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

Using these macros with Roman names and other specialized names likely will produce unsatisfactory results. Below we display other name forms after referring to some names to ensure that they have sensible page entries:

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

Louis XIV	`\ForgetThis\Name{Louis, XIV}`
Lao-tzu	`\Name{Lao-tzu}`
Lafcadio Hearn	`\Name[Lafcadio]{Hearn}`
Charles du Fresne	`\Name[Charles]{du~Fresne}`

Caps and reversing macros work on the arguments that are printed in the text.

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

\PName
\PName*

These convenience macros were an early feature of nameauth. They print a main name followed by a cross-reference in parentheses. If one is inclined to view \AKA as quirky rubbish, these two macros are a dumpster fire.

\PName [⟨*FNN*⟩]{⟨*SAFX*⟩}[⟨*xref FNN*⟩]{⟨*xref SAFX*⟩}[⟨*xref Alt.*⟩]
\PName*[⟨*FNN*⟩]{⟨*SAFX*⟩}[⟨*xref FNN*⟩]{⟨*xref SAFX*⟩}[⟨*xref Alt.*⟩]

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

- Most prefix macros only affect ⟨*FNN*⟩⟨*SAFX*⟩, not the cross-reference.
- The cross-reference always has a long form.
- \SkipIndex keeps both names out of the index.
- ⟨*Alt.*⟩ and the obsolete syntax work only with the cross-reference.

First use: Mark Twain (Samuel L. Clemens)
Later use: Twain (Samuel L. Clemens)
```
\PName[Mark]{Twain}[Samuel L.]{Clemens}
```
Long use: Mark Twain (Samuel L. Clemens)
```
\PName*[Mark]{Twain}[Samuel L.]{Clemens}
```
Index: Clemens, Samuel L. *see* Twain, Mark

First use: Voltaire (François-Marie Arouet)
Later use: Voltaire (François-Marie Arouet)
```
\PName{Voltaire}[François-Marie]{Arouet}
```
Index: Arouet, François-Marie *see* Voltaire

First use: William I (William the Conqueror)
Later use: William (William the Conqueror)
```
\PName{William, I}{William, the Conqueror}
```
Long use: William I (William the Conqueror)
```
\PName*{William, I}{William, the Conqueror}
```
Index: William the Conqueror *see* William I

First use: Bernard of Clairvaux (*Doctor mellifluus*)
Later use: Bernard (*Doctor mellifluus*)
```
\PName{Bernard, of Clairvaux}
  {\textit{Doctor mellifluus}}
```
Long use: Bernard of Clairvaux (*Doctor mellifluus*)
```
\PName*{Bernard, of Clairvaux}
  {\textit{Doctor mellifluus}}
```
Index: *Doctor mellifluus see* Bernard of Clairvaux

First use: Lao-tzu (Li Er)
Later use: Lao-tzu (Li Er)
```
\PName{Lao-tzu}{Li, Er}
```
Index: Li Er *see* Lao-tzu

These macros only use subsequent-use formatting hooks (Section 9.1). When `\NamesFormat` was the only formatting hook, cross-references printed in the text were not formatted.

`formatAKA` The `formatAKA` option permits `\ForceName` to force "first-use" formatting hooks, but under different conditions because the name patterns are in the cross-reference system (Section 6.2). When using `formatAKA` we get:

Name Pattern(s):
`GoodQueen!Bess!PN`

- Front Matter
  - Good Queen Bess.....`\AKA{Elizabeth,I}[Good Queen]{Bess}`
  - Good Queen Bess.....`\AKA{Elizabeth,I}[Good Queen]{Bess}`
- Main Matter
  - Good Queen Bess.....`\AKA{Elizabeth,I}[Good Queen]{Bess}`
  - Good Queen Bess..`\ForceName\AKA{Elizabeth,I}[Good Queen]{Bess}`

The first appearance of Good Queen Bess above uses `\FrontNamesFormat` as its formatting hook because it is the first occurrence of the alternate name in the front matter. After that, even though Good Queen Bess occurs for the first time in the main matter, it uses the subsequent-use `\MainNameHook` because we are using the cross-reference name patterns. We need to use `\ForceName`, which triggers the expected use of `\NamesFormat`, the first-use main-matter hook.

`alwaysformat` We can use the `alwaysformat` option to force only the use of `\NamesFormat` and `\FrontNamesFormat`, but that can look like rubbish:

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

12.2 Obsolete Syntax

This older version of the Non-Western syntax limits the use of alternate names and cross-references, prevents the use of comma-delimited names, and complicates indexing. It remains for backward compatibility.

`\Name{`⟨*SNN*⟩`}[`⟨*Alternate*⟩`]`

Unlike the comma-delimited ⟨*SNN,Affix*⟩ pair, it was easy to implement the older syntax with ⟨*Alternate*⟩. This syntax created an unacceptable second-tier status, thus, the current syntax replaced it. We show this syntax only for the sake of completeness; we advise not to use it.

- One must **leave empty** the first optional ⟨*FNN*⟩ argument.
- One must **never use** the comma-delimited argument ⟨*Affix*⟩.
- These names **always use** ⟨*Alternate*⟩.
- ⟨*Alternate*⟩ acts like ⟨*Affix*⟩. This is the only time that the ⟨*Alternate*⟩ argument affects both name and index patterns (Section 6.2).
- These names take the form ⟨*SNN Alternate*⟩ in the index.

```
\Name{Henry}[VIII]                          % Ancient
\Name{Chiang}[Kai-shek]                     % Eastern
\begin{nameauth}
  \< Dagb & & Dagobert & I >                % Ancient
  \< Yosh & & Yoshida  & Shigeru >          % Eastern
  \< Fukuyama & &
     \noexpand\textUC{Fukuyama} & Takeshi > % Alt. format
\end{nameauth}
```

After studying in the US during the 1930s, in 1954 Rev. FUKUYAMA Takeshi‡ \Fukuyama\ddag published *Nihon Fukuin Rūteru Kyōkai Shi* (History of the Evangelical Lutheran Church in Japan).

Henry VIII‡	\ForgetThis\Name{Henry}[VIII]\ddag
Henry‡	\Name{Henry}[VIII]\ddag
Chiang Kai-shek‡	\ForgetThis\Name{Chiang}[Kai-shek]\ddag
Chiang‡	\Name{Chiang}[Kai-shek]\ddag
Dagobert I‡	\Dagb\ddag
Dagobert‡	\Dagb\ddag
YOSHIDA Shigeru‡	\CapName\Yosh\ddag
Shigeru YOSHIDA‡	\CapName\RevName\LYosh\ddag
FUKUYAMA Takeshi‡ FUKUYAMA‡	\AltFormatActive \ForgetThis\Fukuyama\ddag \Fukuyama\ddag \AltFormatInactive

Regardless of its flaws, the obsolete syntax shares name patterns, index tags, name tags, and index entries with the current syntax:

Obsolete syntax:	Henry VIII‡	\ForgetThis\Name{Henry}[VIII]\ddag
	Henry,VIII	\ShowPattern{Henry}[VIII]
Current syntax:	Henry	\Name{Henry, VIII}
	Henry,VIII	\ShowPattern{Henry, VIII}

We do not expect people to use the obsolete syntax much anymore. Here we list more advantages to using the current syntax and avoiding the old.

- Only the newer syntax permits such variants:

  Henry Tudor ..... \Name*{Henry, VIII}[Tudor]

- The proper form for the old syntax is, e.g.:

  Henry VIII ..... \Name*{Henry}[VIII]

- Next we see malformed Western names:

  Henry VIII, VIII ..... \Name[Henry]{VIII}
  Tudor VIII, VIII ..... \Name*[Henry]{VIII}[Tudor]

  These malformed names have the same index entry ..... VIII, Henry

If users want to receive warnings about this syntax, use the **strictsyntax** option.

Back to Table of Contents

13 Implementation

Both the manual and the package code follow a similar organization.

13.1 Boolean Flags

The nameauth package is a name parser and indexer. These flags reflect its states.

13.1.1 Flow Control

Who Called Me?

Let name formatting macros in the core name engine know if they were called by the naming macros or by the pseudonym macros.

```
\newif\if@nameauth@InAKA
\newif\if@nameauth@InName
```

Core Macro Locks

`\@nameauth@Name`, `\AKA`, and macros that call them use `\if@nameauth@Lock` to avoid a stack overflow. Setting `\if@nameauth@BigLock` true will prevent the core name engine from executing until the "big lock" is set false.

```
\newif\if@nameauth@Lock
\newif\if@nameauth@BigLock
```

Formatting Hook Indicator

Tell alternate formatting control macros that they are in a formatting hook.

```
\newif\if@nameauth@InHook
```

Core Name Engine Choices

`\JustIndex` toggles this flag, which makes the core name engine act like `\IndexName`.

```
\newif\if@nameauth@JustIndex
```

These two flags trigger `\ForgetName` and `\SubvertName` within `\@nameauth@Name`.

```
\newif\if@nameauth@Forget
\newif\if@nameauth@Subvert
```

13.1.2 Name Grammar and Syntax

Strict Syntax

If this flag is true, one will get warnings when using the obsolete syntax.

```
\newif\if@nameauth@StrictSyntax
```

Name Types

These flags reflect the last name type evaluated by any macro that takes name arguments. The first shows whether or not we have a Western or Non-Western name. The second shows the kind of Non-Western syntax used. They are not reset.

```
\newif\ifNameauthWestern
\newif\ifNameauthObsolete
```

Show/Hide Affix Commas

The `comma` and `nocomma` options toggle the first flag below. `\ShowComma` and `\NoComma` respectively toggle the second and third.

```
\newif\if@nameauth@AlwaysComma
\newif\if@nameauth@ShowComma
\newif\if@nameauth@NoComma
```

Capitalize Entire Surnames

The first flag is global. The second is for individual names.

```
\newif\if@nameauth@AllCaps
\newif\if@nameauth@AllThis
```

Reverse Name Order

These flags govern name reversing, first global, then for individual names.

```
\newif\if@nameauth@RevAll
\newif\if@nameauth@RevThis
```

These flags deals with Western names ordered in a list according to surname.

```
\newif\if@nameauth@RevAllComma
\newif\if@nameauth@RevThisComma
```

Full Stop Detection

This flag is used to prevent double full stops after a name is displayed.

```
\newif\if@nameauth@Punct
```

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
```

Long and Short Names

`\if@nameauth@FullName` is true for a long name form. `\if@nameauth@FirstName` causes only Western forenames or (potentially) Non-Western surnames to be displayed when a shorter form is used.

`\if@nameauth@ShortSNN` works with `\DropAffix` to suppress the affix of a Western name. `\if@nameauth@ForceAffix` toggles the display of only the Western affix. `\if@nameauth@EastFN` forces the printing of Eastern forenames via `\ForceFN`.

```
\newif\if@nameauth@FullName
\newif\if@nameauth@FirstName
\newif\if@nameauth@AltAKA
\newif\if@nameauth@ShortSNN
\newif\if@nameauth@ForceAffix
\newif\if@nameauth@EastFN
```

13.1.3 Debugging

When used with the index debugging macros, show complete index entries in the text if true, otherwise show simple entries.

```
\newif\if@nameauth@LongIdxDebug
```

13.1.4 Indexing

Toggle Indexing

The indexing flags permit or prevent name indexing and tags. `\IndexActive` and `\IndexInctive` or the `index` and `noindex` options toggle the first flag and `\SkipIndex` toggles the second.

```
\newif\if@nameauth@DoIndex
\newif\if@nameauth@SkipIndex
```

Toggle Index Sorting

Allow or prevent the insertion of index sort keys.

```
\newif\if@nameauth@Pretag
```

Verbose Index Warnings

Control the number of warnings concerning the index; default is terse.

```
\newif\if@nameauth@Verbose
```

Index Cross-References

Determine whether `\IndexRef` creates a *see* reference or a *see also* reference.

```
\newif\if@nameauth@SeeAlso
```

13.1.5 Formatting and Name Patterns

Choose Formatting System

`\NamesActive` and `\NamesInactive`, with the `mainmatter` and `frontmatter` options, toggle formatting hooks via `\if@nameauth@MainFormat`.

```
\newif\if@nameauth@MainFormat
```

Modify Pseudonym Formatting

Permit `\AKA` and related macros to call the first-use formatting hooks once.

```
\newif\if@nameauth@AKAFormat
```

Select Formatting Hooks

`\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
```

Caps and Alternate Formatting

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 within formatting hooks.

```
\newif\if@nameauth@DoCaps
\newif\if@nameauth@UTF
\newif\if@nameauth@Accent
\newif\if@nameauth@AltFormat
\newif\if@nameauth@DoAlt
```

13.1.6 Name Decisions

Creating and Destroying Name Patterns

Restrict creation and destruction of name patters to the current name system if true. Otherwise, work with both systems.

```
\newif\if@nameauth@LocalNames
```

Scope of Name Decision Macros

`\IfMainName`, `\IfFrontName`, and `\IfAKA` use locally-scoped paths by default. When true, this flag causes these macros to have global scope.

```
\newif\if@nameauth@GlobalScope
```

13.1.7 Version Compatibility

`\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*`.

```
\newif\if@nameauth@OldAKA
```

Determine how strict to be with *see* references.

```
\newif\if@nameauth@OldSee
```

These flags are used only for backward compatibility. The first determines how per-name flags are reset. The second affects the behavior of `\JustIndex`. The third toggles whether or not the name argument token registers are set globally. The fourth toggles the inclusion of xargs and suffix for legacy cases.

```
\newif\if@nameauth@OldReset
\newif\if@nameauth@OldPass
\newif\if@nameauth@OldToks
\newif\if@nameauth@OldArgs
```

13.2 Token Registers, Hooks, and Internal Values

13.2.1 Name Argument Token Registers

These three token registers contain the current values of the name arguments passed to macros that accept naming arguments.

```
\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, thus, `@nameauth@e` for that environment.

```
\newtoks\@nameauth@etoksb
\newtoks\@nameauth@etoksc
\newtoks\@nameauth@etoksd
```

13.2.2 Hooks

`\NamesFormat` Used for "first" instances of names in main-matter text. See Sections 9.1 and 11.1. Called when both `\@nameauth@MainFormat` and `\@nameauth@FirstFormat` are true.

```
\newcommand*\NamesFormat{}
```

`\MainNameHook` Used for later instances of names in main-matter text. See Sections 9.1 and 11.1f. Called when `\@nameauth@MainFormat` is true and `\@nameauth@FirstFormat` is false.

```
\newcommand*\MainNameHook{}
```

`\FrontNamesFormat` Used for "first" instances of names in front-matter text. This formatting hook is called when `\@nameauth@MainFormat` is false and `\@nameauth@FirstFormat` is true.

```
\newcommand*\FrontNamesFormat{}
```

`\FrontNameHook` Used for subsequent instances of names in front-matter text. This hook is called when `\@nameauth@MainFormat` is false and `\@nameauth@FirstFormat` is false.

```
\newcommand*\FrontNameHook{}
```

The following are customization points for the core name engine (Section 11.5.2).

`\NameauthName` Hook called when no special name modification is made.

```
\newcommand*\NameauthName{\@nameauth@Name}
```

`\NameauthLName` Hook called after a name is forced long via `\if@nameauth@name` being set to true.

```
\newcommand*\NameauthLName{\@nameauth@Name}
```

`\NameauthFName` Hook called after `\if@nameauth@FirstName` is set true.

```
\newcommand*\NameauthFName{\@nameauth@Name}
```

`\NameauthIndex` This hook allows one to redefine what happens when any naming macro or indexing macro calls the equivalent of `\index`. See Section 7.1.

```
\newcommand*\NameauthIndex{\index}
```

`\NameauthPattern` The current name pattern is stored here. With every call to a macro that takes name arguments (Section 1.6.1), this hook is updated.

```
\let\NameauthPattern\@empty
```

13.2.3 Internal Values

\@nameauth@Actual This sets the "actual" character used by nameauth for index sorting.

```
\def\@nameauth@Actual{@}
```

\@nameauth@Exclude This makes an index xref into an "exclusion": any name pattern ending in !PN that expands to this value. See \ExcludeName below.

```
\newcommand*\@nameauth@Exclude{!}
```

\@nameauth@space This macro provides a consistent space character for index entries.

```
\def\@nameauth@space{ }
```

13.3 Package Options

13.3.1 Name Grammar and Syntax

Show warnings when using the obsolete syntax.

```
\DeclareOption{strictsyntax}{\@nameauth@StrictSyntaxtrue}
```

Change the way that names are displayed with respect to their syntactic forms.

```
\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}
```

13.3.2 Indexing

Global setting for enabling indexing, sort tags, and verbose warnings.

```
\DeclareOption{index}{\@nameauth@DoIndextrue}
\DeclareOption{noindex}{\@nameauth@DoIndexfalse}
\DeclareOption{pretag}{\@nameauth@Pretagtrue}
\DeclareOption{nopretag}{\@nameauth@Pretagfalse}
\DeclareOption{verbose}{\@nameauth@Verbosetrue}
```

13.3.3 Formatting

Determine the name system or change formatting behavior.

```
\DeclareOption{mainmatter}{\@nameauth@MainFormattrue}
\DeclareOption{frontmatter}{\@nameauth@MainFormatfalse}
\DeclareOption{alwaysformat}{\@nameauth@AlwaysFormattrue}
\DeclareOption{formatAKA}{\@nameauth@AKAFormattrue}
```

13.3.4 Predefined Formatting Hooks

```
\DeclareOption{noformat}{\providecommand*\NamesFormat{}}
\DeclareOption{smallcaps}{\providecommand*\NamesFormat{\scshape}}
\DeclareOption{italic}{\providecommand*\NamesFormat{\itshape}}
\DeclareOption{boldface}{\providecommand*\NamesFormat{\bfseries}}
```

13.3.5 Alternate Formatting

Enable alternate formatting.

```
\DeclareOption{altformat}{%
  \@nameauth@AltFormattrue\@nameauth@DoAlttrue}
```

13.3.6 Scope

Name test paths are either local or the same scope in which they are called.

```
\DeclareOption{globaltest}{\@nameauth@GlobalScopetrue}
```

13.3.7 Version Compatibility

Revert package behavior to mimic older versions.

```
\DeclareOption{oldAKA}{\@nameauth@OldAKAtrue}
\DeclareOption{oldreset}{\@nameauth@OldResettrue}
\DeclareOption{oldpass}{\@nameauth@OldPasstrue}
\DeclareOption{oldtoks}{\@nameauth@OldTokstrue}
\DeclareOption{oldsee}{\@nameauth@OldSeetrue}
\DeclareOption{oldargs}{\@nameauth@OldArgstrue}
```

13.4 Package Initialization

Execute default options and process options passed by the user. Load required packages for ϵ-TEX features, trimming spaces from arguments, starred commands, and optional arguments.

```
\ExecuteOptions
  {nocomma,mainmatter,index,pretag,
   normalcaps,notreversed,noformat}
\ProcessOptions\relax
\RequirePackage{etoolbox}
\RequirePackage{trimspaces}
\RequirePackage{xparse}
```

Test for the OldArgs flag and include legacy packages if the flag is true.

```
\if@nameauth@OldArgs
  \RequirePackage{xargs}
  \RequirePackage{suffix}
\fi
```

13.5 Internal Macros

13.5.1 Fundamental Macros

The following macros are the most essential to the concept of names.

Name Patterns: Who Am I?

\@nameauth@Clean Thanks to Heiko Oberdiek, this macro produces a "sanitized" string to make a control sequence for a name. Testing for the existence and providing for macros based on 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 pattern 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 Return everything before a comma in ⟨*SNN*⟩, using the helper macro below.

```
\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 Return everything before a vertical bar (|) in an index tag, using the helper below.

```
\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 Return only the affix in an ⟨*SNN,Affix*⟩ pair, using the helper macro below.

```
\newcommand*\@nameauth@Suffix[1]{\@nameauth@@Suffix#1,,\\}
```

\@nameauth@@Suffix Throw out the radix; return the suffix with no leading spaces.

```
\def\@nameauth@@Suffix#1,#2,#3\\{%
  \ifx\\#2\\\@empty\else\trim@spaces{#2}\fi
}
```

\@nameauth@GetSuff Test the suffix for a leading active Unicode character.

```
\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. Used only with inputenc / fontenc in the next two macros below.

```
\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` Fully test what method to use when capitalizing a letter, whether native Unicode (`xelatex` or `lualatex` via `\Umathchar`) or inputenc / fontenc.

```
\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 for a suffix, which requires more work to function 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 token register assignment looks weird, but it works.

```
  \expandafter\toks@%
    \expandafter\expandafter\expandafter{\@nameauth@testarg}%
```

Assign the token register 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` Capitalize the first letter of the argument, using the helper below.

```
\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` Capitalize the first active Unicode letter when using 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, 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 ⟨*FNN*⟩ 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 when using inputenc.

```
        \if@nameauth@UTF
          \expandafter\def\expandafter\carga\expandafter{%
            \expandafter\@nameauth@CapUTF\expandafter{\test}}%
```

Capitalize the first native Unicode character.

```
    \else
      \expandafter\def\expandafter\carga\expandafter{%
        \expandafter\@nameauth@Cap\expandafter{\test}}%
    \fi
  \fi
```

We test ⟨*SNN*⟩ 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 native Unicode character.

```
  \else
    \expandafter\def\expandafter\crootb\expandafter{%
      \expandafter\@nameauth@Cap\expandafter{\rootb}}%
  \fi
```

We test ⟨*Affix*⟩ 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 native Unicode character.

```
    \else
      \edef\@nameauth@test{\@nameauth@Suffix{#2}}%
      \expandafter\def\expandafter\csuffb\expandafter{%
        \expandafter\@nameauth@Cap\expandafter{\test}}%
    \fi
  \fi
```

We test ⟨*Alternate*⟩ 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 native Unicode character.

```
        \else
          \expandafter\def\expandafter\cargc\expandafter{%
            \expandafter\@nameauth@Cap\expandafter{\test}}%
        \fi
      \fi
```

Let all the local name elements have caps. We do not know which ones will be used.

```
      \let\arga\carga%
      \let\rootb\crootb%
      \let\suffb\csuffb%
      \let\argc\cargc%
    \fi
  \fi
}
```

Parsing: Full Stops

`\@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, 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`.

```
  \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 full stop. 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`, the lookahead, is a full stop, we gobble it.

```
\newcommand*\@nameauth@EvalDot
{%
  \let\@nameauth@stop=.%
  \ifx\@nameauth@token\@nameauth@stop
    \expandafter\@gobble \fi
}
```

Parsing: Breaking, Spaces, and Commas

`\@nameauth@AddPunct` Here we make spaces between name elements in the text break or not, and decide whether to add commas or not, depending on the name type. We only use this macro in `\@nameauth@Parse`.

```
\newcommand*\@nameauth@AddPunct
{%
  \ifdefined\@nameauth@InParser
    \def\Space{ }%
    \def\SpaceW{ }%
```

`\SpaceW` 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\SpaceW{\nobreakspace}\fi
```

Western names use commas differently from all other names, so we only use the following logic if ⟨*FNN*⟩ is not empty.

```
    \unless\ifx\arga\@empty
      \if@nameauth@AlwaysComma\@nameauth@ShowCommatrue\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 for all possible name forms. We use `\@nameauth@Choice` to pick the name pattern that we want. Using `\edef` ensures that these macros are just the detokenized arguments.

```
  \xdef\@nameauth@csb{\@nameauth@Clean{#2}}%
  \xdef\@nameauth@csbc{\@nameauth@Clean{#2,#3}}%
  \xdef\@nameauth@csab{\@nameauth@Clean{#1!#2}}%
```

Make token register copies of the current name arguments.

```
  \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
}
```

Parsing: Standard Parsing Logic

`\@nameauth@Choice` This standard logic applies to all macros that take name arguments. Here we update `\NameauthPattern`, `\ifNameauthWestern`, and `\ifNameauthObsolete` to show the resulting name pattern and type of name.

```
\newcommand\@nameauth@Choice[3]
{%
  \ifx\@nameauth@A\@empty
    \ifx\@nameauth@C\@empty
```

This path is for Non-Western names. The `#1` argument is used both here and below when `\@nameauth@SB` is present. The `#1` path always corresponds to the present syntax. Thus, when printing names in the text, the `#1` argument must test both `\@nameauth@C` and `\@nameauth@SB`, replacing the latter with former if it exists. With indexing and other macros, one ignores `\@nameauth@C`.

```
      \global\let\NameauthPattern\@nameauth@csb%
      \global\NameauthWesternfalse%
      \global\NameauthObsoletefalse%
      #1%
    \else
      \ifx\@nameauth@SB\@empty
```

The `#2` argument is only for Non-Western names that use the obsolete syntax. Here `\@nameauth@SB` never occurs. For indexing and control sequences, one cannot ignore the use of `\@nameauth@C` in this path.

```
        \if@nameauth@StrictSyntax
          \PackageWarning{nameauth}{Use of obsolete syntax}\fi
        \global\let\NameauthPattern\@nameauth@csbc%
        \global\NameauthWesternfalse%
        \global\NameauthObsoletetrue%
        #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. That shows we are again using the current syntax with a potential name swap.

```
        \global\let\NameauthPattern\@nameauth@csb%
        \global\NameauthWesternfalse%
        \global\NameauthObsoletefalse%
        #1%
      \fi
    \fi
  \else
```

This path is for Western names. When printing names in the text, somewhere in the `#3` argument one must test for `\@nameauth@C` and swap it for `\@nameauth@A`. Also check for and handle `\@nameauth@SB`. Otherwise, for indexing and patterns, ignore `\@nameauth@C` in this path and handle `\@nameauth@SB` appropriately.

```
    \global\let\NameauthPattern\@nameauth@csab%
    \global\NameauthWesterntrue%
    \global\NameauthObsoletefalse%
    #3%
  \fi
}
```

`\@nameauth@Flags` Reset flags after the naming macros and `\AKA` and friends create output in the text. Other places where flags are reset are for special cases.

```
\newcommand*\@nameauth@Flags
{%
  \global\@nameauth@ForceAffixfalse%
  \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` pass through `\AKA` and `\AKA*`. We 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
}
```

13.5.2 Error Detection and Debugging

\@nameauth@Error The nameauth package will halt with a meaningful error when a required name argument is empty, expands to empty, or has an empty root in a root/suffix pair.

```
\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@IdxPageref Here 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.

```
\newcommand*\@nameauth@IdxPageref[3]
{%
```

Warn if \SkipIndex was called before \ShowIdxPageref, and reset it.

```
  \if@nameauth@SkipIndex
    \PackageWarning{nameauth}
    {\string\SkipIndex precedes \string\ShowIdxPageref; check}%
    \unless\if@nameauth@OldReset
      \@nameauth@SkipIndexfalse%
    \fi
  \fi
```

Start a local scope to isolate any changes and redefine \NameauthIndex (the index macro hook) to print an entry in the text.

```
  \begingroup%
    \def\NameauthIndex##1{##1}%
    \@nameauth@DoIndextrue%
```

We locally delete any tag and xref control sequences as needed. They will be restored when the scope ends. If \ShowIdxPageref set \@nameauth@LongIdxDebugtrue we produce a full index entry that shows all the tags and the "actual" character as well as the name. Otherwise we produce a short index entry that shows only the name.

```
    \@nameauth@Choice{}{}{}%
    \csundef{\NameauthPattern!PN}%
    \unless\if@nameauth@LongIdxDebug
      \csundef{\NameauthPattern!PRE}%
      \csundef{\NameauthPattern!TAG}%
    \fi
    \IndexName[#1]{#2}[#3]%
```

We close the scope and reset the flags.

```
  \endgroup%
  \global\@nameauth@LongIdxDebugfalse%
}
```

13.5.3 Core Name Engine

Argument Processing Layer

\@nameauth@Name Marc van Dongen provided the original basic structure. Parsing, indexing, and formatting are more modularized than in earlier versions.

```
\NewDocumentCommand{\@nameauth@Name}{O{} m O{}}
{%
```

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 pattern if directed. Deletion has priority because it occurs after creation. Ensure that names are printed in horizontal mode. Wrap the name with two index entries in case a page break occurs between them.

```
      \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.

```
  \fi
  \if@nameauth@Punct\expandafter\@nameauth@CheckDot\fi
}
```

Syntactic Element Layer

`\@nameauth@Parse` Parse and print a name in the text. The final required argument tells us which name system we are in (Section 6.2). 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
```

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
```

Enter a local scope where we can use simple macros without worry about name space collisions. We process and load the arguments into the appropriate macros.

```
  \begingroup%
    \def\@nameauth@InParser{}%
    \@nameauth@LoadArgs{#1}{#2}{#3}%
```

Let the protected control sequences to local ones.

```
    \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 attach “meaning” to patterns of macro arguments primarily via \FNN and \SNN. Then we call the name printing macros based on 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. We allow \ForceAffix to work only if if no affix exists

```
      {%
        \ifx\argc\@empty
          \let\FNN\arga%
        \else
          \let\FNN\argc%
        \fi
        \ifx\suffb\@empty
          \@nameauth@ForceAffixfalse%
        \else
          \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
}
```

`\@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
```

Toggle flags based on the name arguments.

```
    \@nameauth@Form{#1}%
```

Send a mononym to the hook dispatcher.

```
    \ifx\FNN\@empty
      \@nameauth@Hook{\SNN}%
    \else
```

Send a full name or reversed full name to the hook dispatcher.

```
      \if@nameauth@FullName
        \if@nameauth@RevThis
          \@nameauth@Hook{\FNN\Space\SNN}%
        \else
          \@nameauth@Hook{\SNN\Space\FNN}%
        \fi
      \else
```

Send a partial name to the hook dispatcher.

```
        \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
```

Toggle flags based on the name arguments.

```
    \@nameauth@Form{#1}%
    \edef\RevSpace{,\SpaceW}%
```

Send a reversed full name, reversed full name with commas, or a full name to the hook dispatcher.

```
    \if@nameauth@FullName
      \if@nameauth@RevThis
        \@nameauth@Hook{\SNN\SpaceW\FNN}%
      \else
        \if@nameauth@RevThisComma
          \@nameauth@Hook{\SNN\RevSpace\FNN}%
        \else
          \@nameauth@Hook{\FNN\SpaceW\SNN}%
        \fi
      \fi
    \else
```

Send only an affix to the hook dispatcher.

```
      \if@nameauth@ForceAffix
        \@nameauth@Hook{\suffb}%
      \else
```

Send a partial name to the hook dispatcher.

```
        \if@nameauth@FirstName
          \@nameauth@Hook{\FNN}%
        \else
          \@nameauth@Hook{\rootb}%
        \fi
      \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
}
```

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. The flag `\if@nameauth@InHook` will reset outside of the local scope in `\@nameauth@Parse`.

```
    \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\@nameauth@InHooktrue\NamesFormat{#1}\egroup%
      \else
        \bgroup\@nameauth@InHooktrue\MainNameHook{#1}\egroup%
      \fi
    \else
```

We use the formatting hooks for the front-matter system.

```
      \if@nameauth@FirstFormat
        \bgroup\@nameauth@InHooktrue\FrontNamesFormat{#1}\egroup%
      \else
        \bgroup\@nameauth@InHooktrue\FrontNameHook{#1}\egroup%
      \fi
    \fi
  \fi
}
```

13.5.4 Indexing

\@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[4][\@empty]
{%
  \if@nameauth@DoIndex
  \edef\@nameauth@XrefType{#1}%
```

If a cross-reference will be created, its type is in the first argument. We put that argument into a macro to evaluate it. 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#2!TAG\endcsname
      \protected@edef\@nameauth@Tag{\csname#2!TAG\endcsname}%
      \expandafter\def\expandafter\@nameauth@ShortTag\expandafter{%
        \expandafter\@nameauth@TrimTag\expandafter{\@nameauth@Tag}}%
```

Set up either a page entry or an xref with both a sorting tag and an info tag.

```
      \ifcsname#2!PRE\endcsname
        \protected@edef\@nameauth@Pre{\csname#2!PRE\endcsname}%
        \ifx\@nameauth@XrefType\@empty
          \protected@edef\@nameauth@IdxEntry
            {\@nameauth@Pre#3\@nameauth@Tag}%
        \else
          \protected@edef\@nameauth@IdxEntry
            {\@nameauth@Pre#3\@nameauth@ShortTag|#1{#4}}%
        \fi
      \else
```

Set up either a page entry or a cross-reference with an info tag.

```
        \ifx\@nameauth@XrefType\@empty
          \protected@edef\@nameauth@IdxEntry
            {#3\@nameauth@Tag}%
        \else
          \protected@edef\@nameauth@IdxEntry
            {#3\@nameauth@ShortTag|#1{#4}}%
        \fi
      \fi
    \else
```

Set up either a page entry or a cross-reference with a sorting tag.

```
      \ifcsname#2!PRE\endcsname
        \protected@edef\@nameauth@Pre{\csname#2!PRE\endcsname}%
        \ifx\@nameauth@XrefType\@empty
          \protected@edef\@nameauth@IdxEntry{\@nameauth@Pre#3}%
        \else
          \protected@edef\@nameauth@IdxEntry{\@nameauth@Pre#3|#1{#4}}%
        \fi
      \else
```

Set up either a page entry or a cross-reference with no tag.

```
        \ifx\@nameauth@XrefType\@empty
          \protected@edef\@nameauth@IdxEntry{#3}%
        \else
          \protected@edef\@nameauth@IdxEntry{#3|#1{#4}}%
        \fi
      \fi
    \fi
```

Create the index entry.

```
    \expandafter\NameauthIndex\expandafter{\@nameauth@IdxEntry}%
  \fi
}
```

13.6 For Users: Prefix Macros

Prefix macros precede a particular name (or xref) to affect only that name.

13.6.1 Name Syntax

Commas Before 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}
```

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; a fallback.

```
\newcommand*\AccentCapThis
  {\@nameauth@Accenttrue\@nameauth@DoCapstrue}
```

`\CapName` Capitalize entire ⟨*SNN*⟩. Overrides `\CapThis` for surnames.

```
\newcommand*\CapName{\@nameauth@AllThistrue}
```

Reversing and Name Parts

`\RevName` Reverse name order.

```
\newcommand*\RevName{\@nameauth@RevThistrue}
```

`\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}
```

Reversing with Commas

`\RevComma` Last name, comma, first name.

```
\newcommand*\RevComma{\@nameauth@RevThisCommatrue}
```

Affixes and Breaking

\ForceAffix Force the printing of just the affix of a short Western name in the text.

```
\newcommand*\ForceAffix{\@nameauth@ForceAffixtrue}
```

\DropAffix Suppress printing the affix of a long Western name.

```
\newcommand*\DropAffix{\@nameauth@ShortSNNtrue}
```

\KeepAffix Bind a name-affix pair with a non-breaking space.

```
\newcommand*\KeepAffix{\@nameauth@NBSPtrue}
```

\KeepName Use non-breaking spaces between name syntactic forms.

```
\newcommand*\KeepName
  {\@nameauth@NBSPtrue\@nameauth@NBSPXtrue}
```

13.6.2 Indexing

\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*.

```
\newcommand*\SeeAlso{\@nameauth@SeeAlsotrue}
```

13.6.3 Formatting and Name Decisions

\ForceName Set \@nameauth@FirstFormat to be true even for subsequent name uses. Makes the core name engine use \NamesFormat. Works for one name only.

```
\newcommand*\ForceName{\@nameauth@FirstFormattrue}
```

\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@Subverttrue}
```

13.7 For Users: Helper Macros

Helper macros do not need to precede a particular name and their effects endure for multiple names. They tend to affect an entire scope and usually come in pairs.

13.7.1 Name Syntax

Capitalization

`\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}
```

Reversing

`\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}
```

Reversing with Commas

`\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. The macro `\ReverseActive` takes priority over this macro.

```
\newcommand*\ReverseCommaActive{\@nameauth@RevAllCommatrue}
```

13.7.2 Indexing

`\IndexActual` Change the "actual" character from the default. This allows one to use, for example, `\global\IndexActual{=}` in `dtx` files.

```
\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}
```

`\IndexWarnVerbose` Turn on verbose warnings for indexing.

```
\newcommand*\IndexWarnVerbose{\@nameauth@Verbosetrue}
```

`\IndexWarnTerse` Turn off verbose warnings for indexing.

```
\newcommand*\IndexWarnTerse{\@nameauth@Verbosefalse}
```

`\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. This macro is deliberately local, so one can use scoping to isolate its effects.

```
\newcommand*\IndexProtect
  {\@nameauth@DoIndexfalse\@nameauth@BigLocktrue}
```

13.7.3 Formatting

`\NamesInactive` Switch to the front-matter name system.

```
\newcommand*\NamesInactive{\@nameauth@MainFormatfalse}
```

`\NamesActive` Switch to the main-matter name system.

```
\newcommand*\NamesActive{\@nameauth@MainFormattrue}
```

13.7.4 Alternate Formatting

`\AltFormatActive` `\AltFormatActive*` Turn on alternate formatting and disengage the formatting macros if using the starred form or engage the formatting macros if using the un-starred form.

```
\NewDocumentCommand{\AltFormatActive}{s}
{%
  \global\@nameauth@AltFormattrue%
  \IfBooleanTF{#1}
    {\global\@nameauth@DoAltfalse}
    {\global\@nameauth@DoAlttrue}%
}
```

`\AltFormatInactive` Turn off alternate formatting altogether.

```
\newcommand*\AltFormatInactive
  {\global\@nameauth@AltFormatfalse\global\@nameauth@DoAltfalse}
```

`AltFormatZone` (*env.*) We enable alternate formatting in its active (on) state.

```
\newenvironment{AltFormatZone}
  {\AltFormatActive\begingroup\def\@nameauth@InZone{}\ignorespaces}
  {\endgroup\AltFormatInactive\ignorespacesafterend}
```

`AltFormatZone*` (*env.*) We enable alternate formatting in its inactive (off) state.

```
\newenvironment{AltFormatZone*}
  {\AltFormatActive*\begingroup\def\@nameauth@InZone{}\ignorespaces}
  {\endgroup\AltFormatInactive\ignorespacesafterend}
```

`\FixateFormat` Redefine alternate formatting switches to do nothing only within the environments above. The user must choose to invoke this macro.

```
\newcommand{\FixateFormat}
{%
  \ifdefined\@nameauth@InZone
    \RenewDocumentCommand{\AltFormatActive}{s}{}%
    \renewcommand\AltFormatInactive{}%
  \fi
}
```

`\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. Change first to roman, then small caps, to avoid font warning.

```
\newcommand*\textSC[1]
  {\if@nameauth@DoAlt{\rmfamily\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}
```

13.7.5 Name Decisions

\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}
```

\LocalNames \LocalNames sets @nameauth@LocalNames true so \ForgetName and \SubvertName do not affect both main and front matter name systems, only the current one.

```
\newcommand*\LocalNames{\global\@nameauth@LocalNamestrue}
```

\GlobalNames \GlobalNames restores the default behavior of \ForgetName and \SubvertName, which affects both name systems at once.

```
\newcommand*\GlobalNames{\global\@nameauth@LocalNamesfalse}
```

13.7.6 User-Accessible Name Parser

\NameParser Print a name form based on the current state of the nameauth flags in the locked path. Used only in the hook macros, within the local scope of \@nameauth@Parse. We \let values to \FNN and \SNN not to be efficient but to be pedantic. Sometimes, the attempt to short-circuit parser logic will seem to work, only for a case to emerge that breaks things. Being pedantic reduces that risk, even if it is slower at times.

```
\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
        \unless\if@nameauth@FirstName \SNN \fi
      \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\SpaceW\FNN%
        \else
          \if@nameauth@RevThisComma
            \SNN\RevSpace\FNN%
          \else
            \FNN\SpaceW\SNN%
          \fi
        \fi
      \else
        \if@nameauth@ForceAffix
          \let\SNN\suffb \SNN%
        \else
          \if@nameauth@FirstName \FNN%
          \else
            \let\SNN\rootb \SNN%
          \fi
        \fi
      \fi
    }%
  \fi
}
```

13.8 For Users: Macros That Take Name Arguments

The rest of the nameauth macros all take name arguments. They all update \NameauthPattern, \ifNameauthWestern, and \ifNameauthObsolete when called. The file examples.tex iterates through all possible argument variations of these macros except the debugging macros, the non-printing arguments of \AKA, and \PName. It thus tests for spurious spaces and bad output.

13.8.1 Basic Interface

\Name
\Name*

\Name calls \NameauthName, the interface hook, printing a long name and calling \NameauthLName when using the starred form.

```
\NewDocumentCommand{\Name}{s}
{%
  \IfBooleanTF{#1}
    {\@nameauth@FullNametrue\NameauthLName}
    {\NameauthName}%
}
```

\FName
\FName*

\FName sets up a short name instance and calls \NameauthFName, the interface hook. Its starred form is identical in function.

```
\NewDocumentCommand{\FName}{s}
  {\@nameauth@FirstNametrue\NameauthFName}
```

13.8.2 Quick Interface

nameauth (*env.*)

Here we create macro shorthands. First we define a macro \< that takes four arguments, delimited by three ampersands and >. This macro is local to the nameauth environment, but the shorthand macros that it creates are global.

```
\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. Lacking either will produce an error. Redefining a shorthand only creates a warning.

```
    \ifx\@arga@\@empty
      \PackageError{nameauth}%
      {environment nameauth: Macro name missing;
      \expandafter\detokenize\expandafter{##3}}%
    \fi
    \@nameauth@Error{##3}{macro \ex\zap@space\string\ \@empty##1}%
    \ifcsname\@arga@\endcsname
      \PackageWarning{nameauth}
      {Environment nameauth: shorthand macro already exists}%
    \fi
```

Set up shorthands according to name form. We use `\expandafter` due to `\protected@edef` in the naming macros. We begin with Non-Western names that use the new syntax.

```
    \ifx\@testd@\@empty
      \ifx\@testb@\@empty
```

Standard name:

```
        \ex\csgdef\ex{\ex\@arga@\ex}%
          \ex{\ex\NameauthName\ex{\the\@nameauth@etoksc}}%
```

Long name:

```
        \ex\csgdef\ex{\ex\@larga@\ex}%
          \ex{\ex\@nameauth@FullNametrue%
          \ex\NameauthLName\ex{\the\@nameauth@etoksc}}%
```

Short name:

```
        \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 uses, then one use of `\ex` per token.

Standard name:

```
        \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}}%
```

Long name:

```
    \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}}%
```

Short name:

```
    \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 or the older syntax. We again have two arguments to expand first.

Standard name:

```
  \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]}%
```

Long name:

```
    \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]}%
```

Short name:

```
    \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 uses, three uses, and one use of `\ex`.

Standard name:

```
    \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\NameauthName%
      \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]}%
```

Long name:

```
        \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\@nameauth@FullNametrue%
          \ex\ex\ex\ex\ex\ex\ex\NameauthLName%
          \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]}%
```

Short name:

```
        \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\@nameauth@FirstNametrue%
          \ex\ex\ex\ex\ex\ex\ex\NameauthFName%
          \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\ignorespacesafterend}
```

13.8.3 Debugging Macros

`\ShowPattern` This macro shows a name pattern created by its arguments, unless invoked with an empty argument. Then it returns `\NameauthPattern` from the last name arguments that were evaluated.

```
\NewDocumentCommand{\ShowPattern}{O{} m O{}}
{%
  \protected@edef\@nameauth@testname{\trim@spaces{#2}}%
  \unless\ifx\@nameauth@testname\@empty
    \@nameauth@Error{#2}{macro \string\ShowPattern}%
    \@nameauth@LoadArgs{#1}{#2}{#3}%
    \@nameauth@Choice{}{}{}\NameauthPattern%
  \else\NameauthPattern\fi
}
```

`\ShowIdxPageref` `\ShowIdxPageref*` Show an index page entry appearance for the given name arguments or, if a null argument is given, show an entry for the name arguments in the token registers.

```
\NewDocumentCommand{\ShowIdxPageref}{s O{} m O{}}
{%
  \begingroup%
  \let\ex\expandafter%
  \IfBooleanTF {#1}%
```

The starred form displays a basic index entry with no tag.

```
  {%
    \protected@edef\@nameauth@testname{\trim@spaces{#3}}%
    \unless\ifx\@nameauth@testname\@empty
      \@nameauth@Error{#3}{macro \string\ShowIdxPageref*}%
```

```
      \@nameauth@LoadArgs{#2}{#3}{#4}%
      \@nameauth@IdxPageref{#2}{#3}{#4}%
    \else
```

Pull name arguments from the token regs.

```
      \ex\ex\ex\ex\ex\ex\ex\@nameauth@LoadArgs%
      \ex\ex\ex\ex\ex\ex\ex{%
      \ex\ex\ex\the\ex\ex\ex\@nameauth@toksa\ex\ex\ex}%
      \ex\ex\ex{\ex\the\ex\@nameauth@toksb\ex}%
      \ex{\the\@nameauth@toksc}%
      \ex\ex\ex\ex\ex\ex\ex\@nameauth@IdxPageref%
      \ex\ex\ex\ex\ex\ex\ex{%
      \ex\ex\ex\the\ex\ex\ex\@nameauth@toksa\ex\ex\ex}%
      \ex\ex\ex{\ex\the\ex\@nameauth@toksb\ex}%
      \ex{\the\@nameauth@toksc}%
    \fi
  }%
```

The un-starred form displays (expanded, as printed) the index entry that will be generated, but not exactly what is in the `idx` file.

```
  {%
    \global\@nameauth@LongIdxDebugtrue%
    \protected@edef\@nameauth@testname{\trim@spaces{#3}}%
    \unless\ifx\@nameauth@testname\@empty
      \@nameauth@Error{#3}{macro \string\ShowIdxPageref}%
      \@nameauth@LoadArgs{#2}{#3}{#4}%
      \@nameauth@IdxPageref{#2}{#3}{#4}%
    \else
```

Pull name arguments from the token regs.

```
      \ex\ex\ex\ex\ex\ex\ex\@nameauth@LoadArgs%
      \ex\ex\ex\ex\ex\ex\ex{%
      \ex\ex\ex\the\ex\ex\ex\@nameauth@toksa\ex\ex\ex}%
      \ex\ex\ex{\ex\the\ex\@nameauth@toksb\ex}%
      \ex{\the\@nameauth@toksc}%
      \ex\ex\ex\ex\ex\ex\ex\@nameauth@IdxPageref%
      \ex\ex\ex\ex\ex\ex\ex{%
      \ex\ex\ex\the\ex\ex\ex\@nameauth@toksa\ex\ex\ex}%
      \ex\ex\ex{\ex\the\ex\@nameauth@toksb\ex}%
      \ex{\the\@nameauth@toksc}%
    \fi
  }%
  \endgroup%
}
```

`\ShowNameInfo` Show how the name arguments are interpreted by the nameauth macros, but as detokenized text. If a null argument is given, use the arguments in the token regs, left from the last name evaluated. The starred form puts the separate data on different lines.

```
\NewDocumentCommand{\ShowNameInfo}{O{} m O{}}
{%
  \begingroup%
  \let\ex\expandafter%
```

Test for bad input, then load the argument values into the appropriate macros.

```
  \protected@edef\@nameauth@testname{\trim@spaces{#2}}%
  \unless\ifx\@nameauth@testname\@empty
    \@nameauth@Error{#2}{macro \string\ShowNameInfo}%
    \@nameauth@LoadArgs{#1}{#2}{#3}%
  \else
```

In the case of a null argument, load name arguments from the token regs.

```
    \ex\ex\ex\ex\ex\ex\ex\@nameauth@LoadArgs%
    \ex\ex\ex\ex\ex\ex\ex{%
    \ex\ex\ex\the\ex\ex\ex\@nameauth@toksa\ex\ex\ex}%
    \ex\ex\ex{\ex\the\ex\@nameauth@toksb\ex}%
    \ex{\the\@nameauth@toksc}%
  \fi
  \protected@edef\@nameauth@cleanA
    {\expandafter\detokenize\expandafter{\@nameauth@A}}%
  \protected@edef\@nameauth@cleanB
    {\expandafter\detokenize\expandafter{\@nameauth@B}}%
  \protected@edef\@nameauth@cleanSB
    {\expandafter\detokenize\expandafter{\@nameauth@SB}}%
  \protected@edef\@nameauth@cleanC
    {\expandafter\detokenize\expandafter{\@nameauth@C}}%
```

First we have Non-Western names using the current syntax.

```
  \@nameauth@Choice
  {%
    (SNN: \@nameauth@cleanB)%
    \unless\ifx\@nameauth@SB\@empty
      \ (Affix*: \@nameauth@cleanSB)%
    \fi
    \unless\ifx\@nameauth@C\@empty
      \ (Alt: \@nameauth@cleanC)%
    \fi
  }%
```

Next is Non-Western names using the obsolete syntax.

```
  {%
    (SNN: \@nameauth@cleanB)%
    \unless\ifx\@nameauth@C\@empty
      \ (Alt*: \@nameauth@cleanC)%
    \fi
  }%
```

Finally we have Western names.

```
  {%
    (FNN: \@nameauth@cleanA)
    (SNN: \@nameauth@cleanB)%
    \unless\ifx\@nameauth@SB\@empty
      \ (Affix: \@nameauth@cleanSB)%
    \fi
    \unless\ifx\@nameauth@C\@empty
      \ (Alt: \@nameauth@cleanC)%
    \fi
  }%
  \endgroup%
}
```

\ShowNameState
\ShowNameState*

These macros show the user the name pattern, name type, index state, and name systems of a given name. The un-starred form prints on one line; the starred form, multiple lines. If a null argument is given, they use the arguments from the last printed name.

```
\NewDocumentCommand{\ShowNameState}{s O{} m O{}}
{%
```

Create a local scope to kill any local definitions on exit. Test for bad input, then load the argument values into the appropriate macros. Test for a star and set the break to \par or a space.

```
  \begingroup%
  \IfBooleanTF{#1}
    {\def\@nameauth@break{\par}}
    {\def\@nameauth@break{ }}
  \let\ex\expandafter%
  \protected@edef\@nameauth@testname{\trim@spaces{#3}}%
  \unless\ifx\@nameauth@testname\@empty
    \@nameauth@Error{#3}{macro \string\ShowNameState}%
    \@nameauth@LoadArgs{#2}{#3}{#4}%
  \else
```

Pull name arguments from the token regs.

```
    \ex\ex\ex\ex\ex\ex\ex\@nameauth@LoadArgs%
    \ex\ex\ex\ex\ex\ex\ex{%
    \ex\ex\ex\the\ex\ex\ex\@nameauth@toksa\ex\ex\ex}%
    \ex\ex\ex{\ex\the\ex\@nameauth@toksb\ex}%
    \ex{\the\@nameauth@toksc}%
  \fi
```

Parse the name arguments and determine name type. We use this method instead of examining the Boolean flags because it is more efficient here.

```
  \@nameauth@Choice
    {\def\@nameauth@nametype{nw}}
    {\def\@nameauth@nametype{nw,os}}
    {\def\@nameauth@nametype{w}}
```

Check to see what control sequences exist and collect the information.

```
  \ifcsname\NameauthPattern!MN\endcsname
    \def\@nameauth@mainname{main}%
  \fi
  \ifcsname\NameauthPattern!NF\endcsname
    \def\@nameauth@frontname{front}%
  \fi
  \ifcsname\NameauthPattern!PN\endcsname
    \edef\@nameauth@testex
      {\csname\NameauthPattern!PN\endcsname}%
    \ifx\@nameauth@testex\@nameauth@Exclude
      \def\@nameauth@excl{excl}%
    \else
      \def\@nameauth@xref{xref}%
    \fi
  \fi
  \ifcsname\NameauthPattern!PRE\endcsname
    \def\@nameauth@pre{pretag}%
  \fi
```

```
  \ifcsname\NameauthPattern!TAG\endcsname
    \def\@nameauth@tag{idxtag}%
  \fi
  \ifcsname\NameauthPattern!DB\endcsname
    \def\@nameauth@db{namedb}%
  \fi
```

If either a main name or a front name exist, create a macro that reflects this condition.

```
  \ifdefined \@nameauth@mainname \def\@nameauth@namecs{}\fi
  \ifdefined \@nameauth@frontname \def\@nameauth@namecs{}\fi
```

If an xref and an exclusion exist for a name, something went wrong.

```
  \ifdefined \@nameauth@xref
    \ifdefined \@nameauth@excl
      \PackageWarning{nameauth}
      {Both xref and exclusion exist for \NameauthPattern}%
    \fi
  \fi
```

Determine the state of the "index finite state machine".

```
  \ifdefined \@nameauth@namecs
    \def\@nameauth@idxstate{2}%
    \ifdefined \@nameauth@xref
      \def\@nameauth@idxstate{4}%
    \fi
    \ifdefined \@nameauth@excl
      \def\@nameauth@idxstate{6}%
    \fi
   \else
    \ifdefined \@nameauth@xref
      \def\@nameauth@idxstate{3}%
      \ifdefined \@nameauth@excl
        \def\@nameauth@idxstate{5}%
      \fi
    \else
      \def\@nameauth@idxstate{1}%
      \def\@nameauth@nosystem{none}%
    \fi
  \fi
```

Display the output.

```
  Pattern: \NameauthPattern\@nameauth@break
  Type: \@nameauth@nametype\@nameauth@break
  State: \@nameauth@idxstate\@nameauth@break
  Systems:%
  \ifdefined \@nameauth@nosystem\ \@nameauth@nosystem \fi
  \ifdefined \@nameauth@mainname\ \@nameauth@mainname \fi
  \ifdefined \@nameauth@frontname\ \@nameauth@frontname \fi
  \ifdefined \@nameauth@xref\ \@nameauth@xref \fi
  \ifdefined \@nameauth@excl\ \@nameauth@excl \fi
  \ifdefined \@nameauth@pre\ \@nameauth@pre \fi
  \ifdefined \@nameauth@tag\ \@nameauth@tag \fi
  \ifdefined \@nameauth@db\ \@nameauth@db \fi
  \endgroup%
}
```

13.8.4 Indexing

\IndexName This creates an index entry with page entries. 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.

```
\NewDocumentCommand{\IndexName}{O{} m O{}}
{%
```

Process and load the arguments; test for malformed input.

```
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@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}
      {\string\SkipIndex precedes \string\IndexName; check for issues}%
    \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}{\string\SeeAlso was reset}%
    \fi
  \fi
```

Enter a local scope, and if so forced, enable indexing only within that scope. 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.

Non-Western names. We ignore \@nameauth@C.

```
  \@nameauth@Choice
  {%
    \def\@nameauth@Temp{\expandafter\detokenize\expandafter{#2}}%
    \ifcsname\@nameauth@csb!PN\endcsname
```

Verbose warnings: Does an exclusion or cross-reference exist?

```
      \if@nameauth@Verbose
        \edef\@nameauth@testex
          {\csname\@nameauth@csb!PN\endcsname}%
        \ifx\@nameauth@testex\@nameauth@Exclude
          \PackageWarning{nameauth}
            {\string\IndexName: exclusion exists \@nameauth@Temp}%
        \else
          \PackageWarning{nameauth}
            {\string\IndexName: xref exists \@nameauth@Temp}%
        \fi
      \fi
    \else
```

Mononym entry

```
      \ifx\@nameauth@SB\@empty
        \@nameauth@Index{\@nameauth@csb}{\@nameauth@B}{}%
      \else
```

Non-Western entry

```
        \@nameauth@Index{\@nameauth@csb}
          {\@nameauth@B\@nameauth@space\@nameauth@SB}{}%
      \fi
    \fi
  }%
```

Non-Western names, obsolete syntax. Using `\@nameauth@C` affects indexing here.

```
  {%
    \def\@nameauth@Temp{\expandafter\detokenize\expandafter{#2 #3}}%
    \ifcsname\@nameauth@csbc!PN\endcsname
```

Verbose warnings: Does an exclusion or cross-reference exist?

```
      \if@nameauth@Verbose
        \edef\@nameauth@testex
          {\csname\@nameauth@csbc!PN\endcsname}%
        \ifx\@nameauth@testex\@nameauth@Exclude
          \PackageWarning{nameauth}
            {\string\IndexName: exclusion exists \@nameauth@Temp}%
        \else
          \PackageWarning{nameauth}
            {\string\IndexName: xref exists \@nameauth@Temp}%
        \fi
      \fi
    \else
```

Non-Western entry

```
      \@nameauth@Index{\@nameauth@csbc}
        {\@nameauth@B\@nameauth@space\@nameauth@C}{}%
    \fi
  }%
```

Western names. We ignore `\@nameauth@C` and handle `\@nameauth@SB`.

```
  {%
    \def\@nameauth@Temp{\expandafter\detokenize\expandafter{#1 #2}}%
    \ifcsname\@nameauth@csab!PN\endcsname
```

Verbose warnings: Does an exclusion or cross-reference exist?

```
      \if@nameauth@Verbose
        \edef\@nameauth@testex
          {\csname\@nameauth@csab!PN\endcsname}%
        \ifx\@nameauth@testex\@nameauth@Exclude
          \PackageWarning{nameauth}
            {\string\IndexName: exclusion exists \@nameauth@Temp}%
        \else
          \PackageWarning{nameauth}
            {\string\IndexName: xref exists \@nameauth@Temp}%
        \fi
      \fi
    \else
```

Western entry, no affix

```
      \ifx\@nameauth@SB\@empty
        \@nameauth@Index{\@nameauth@csab}
          {\@nameauth@B,\@nameauth@space\@nameauth@A}{}%
      \else
```

Western entry, with affix

```
        \@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.

```
\NewDocumentCommand{\IndexRef}{O{} m O{} m}
{%
```

Process and load the arguments into the appropriate macros.

```
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \protected@edef\@nameauth@Target{#4}%
```

Test for malformed input.

```
  \@nameauth@Error{#2}{macro \string\IndexRef}%
```

Warn if `\SkipIndex` was called before `\IndexName`, and reset it unless the `oldreset` option was used.

```
  \if@nameauth@SkipIndex
    \PackageWarning{nameauth}
    {\string\SkipIndex preceded \string\IndexRef; check for issues}%
    \unless\if@nameauth@OldReset
      \@nameauth@SkipIndexfalse%
    \fi
  \fi
  \@nameauth@Choice
```

Non-Western name, new syntax, no ⟨*Affix*⟩

```
  {%
    \def\@nameauth@Temp{\expandafter\detokenize\expandafter{#2}}%
    \ifcsname\@nameauth@csb!PN\endcsname
```

Verbose warnings: Does an exclusion or cross-reference exist?

```
      \if@nameauth@Verbose
        \edef\@nameauth@testex
          {\csname\@nameauth@csb!PN\endcsname}%
        \ifx\@nameauth@testex\@nameauth@Exclude
          \PackageWarning{nameauth}
          {\string\IndexRef: exclusion exists \@nameauth@Temp}%
        \else
          \PackageWarning{nameauth}
          {\string\IndexRef: xref exists \@nameauth@Temp}%
        \fi
      \fi
    \else
```

If no xref or exclusion exists, proceed. If creating a *see also* reference, just do it.

```
    \ifx\@nameauth@SB\@empty
      \if@nameauth@SeeAlso
        \@nameauth@Index[seealso]{\@nameauth@csb}
          {\@nameauth@B}{\@nameauth@Target}%
        \csgdef{\@nameauth@csb!PN}{}%
      \else
```

When creating a *see* reference, check if a name already exists. If so, skip and warn.

```
        \unless\if@nameauth@OldSee
          \unless\ifcsname\@nameauth@csb!MN\endcsname
            \unless\ifcsname\@nameauth@csb!NF\endcsname
              \@nameauth@Index[see]{\@nameauth@csb}
                {\@nameauth@B}{\@nameauth@Target}%
              \csgdef{\@nameauth@csb!PN}{}%
            \else
              \PackageWarning{nameauth}
              {\string\IndexRef: extant name;
               no xref \@nameauth@Temp}%
            \fi
          \else
            \PackageWarning{nameauth}
            {\string\IndexRef: extant name;
             no xref \@nameauth@Temp}%
          \fi
        \else
```

Extra verbose warnings and old behavior

```
          \if@nameauth@Verbose
            \PackageWarning{nameauth}
            {\string\IndexRef: non-strict xref \@nameauth@Temp}%
          \fi
          \@nameauth@Index[see]{\@nameauth@csb}
            {\@nameauth@B}{\@nameauth@Target}%
          \csgdef{\@nameauth@csb!PN}{}%
        \fi
      \fi
    \else
```

Non-Western name, new syntax, ⟨*Affix*⟩ exists; create a *see also* reference if needed.

```
      \if@nameauth@SeeAlso
        \@nameauth@Index[seealso]{\@nameauth@csb}
          {\@nameauth@B\@nameauth@space%
           \@nameauth@SB}{\@nameauth@Target}%
        \csgdef{\@nameauth@csb!PN}{}%
      \else
```

When creating a *see* reference, check if a name already exists. If so, skip and warn.

```
          \unless\if@nameauth@OldSee
            \unless\ifcsname\@nameauth@csb!MN\endcsname
              \unless\ifcsname\@nameauth@csb!NF\endcsname
                \@nameauth@Index[see]{\@nameauth@csb}
                  {\@nameauth@B\@nameauth@space%
                   \@nameauth@SB}{\@nameauth@Target}%
                \csgdef{\@nameauth@csb!PN}{}%
              \else
                \PackageWarning{nameauth}
                {\string\IndexRef: extant name;
                 no xref \@nameauth@Temp}%
              \fi
            \else
              \PackageWarning{nameauth}
              {\string\IndexRef: extant name;
               no xref \@nameauth@Temp}%
            \fi
          \else
```

Extra verbose warnings and old behavior

```
            \if@nameauth@Verbose
              \PackageWarning{nameauth}
              {\string\IndexRef: non-strict xref \@nameauth@Temp}%
            \fi
            \@nameauth@Index[see]{\@nameauth@csb}
              {\@nameauth@B\@nameauth@space%
               \@nameauth@SB}{\@nameauth@Target}%
            \csgdef{\@nameauth@csb!PN}{}%
          \fi
        \fi
      \fi
    \fi
  }%
```

Eastern or ancient name, obsolete syntax

```
  {%
    \def\@nameauth@Temp{\expandafter\detokenize\expandafter{#2 #3}}%
    \ifcsname\@nameauth@csbc!PN\endcsname
```

Verbose warnings: Does an exclusion or cross-reference exist?

```
      \if@nameauth@Verbose
        \edef\@nameauth@testex
          {\csname\@nameauth@csbc!PN\endcsname}%
        \ifx\@nameauth@testex\@nameauth@Exclude
          \PackageWarning{nameauth}
          {\string\IndexRef: exclusion exists \@nameauth@Temp}%
        \else
          \PackageWarning{nameauth}
          {\string\IndexRef: xref exists \@nameauth@Temp}%
        \fi
      \fi
    \else
```

If no xref or exclusion exists, proceed. If creating a *see also* reference, just do it.

```
      \if@nameauth@SeeAlso
        \@nameauth@Index[seealso]{\@nameauth@csbc}
          {\@nameauth@B\@nameauth@space\@nameauth@C}
          {\@nameauth@Target}%
        \csgdef{\@nameauth@csbc!PN}{}%
      \else
```

When creating a *see* reference, check if a name already exists. If so, skip and warn.

```
        \unless\if@nameauth@OldSee
          \unless\ifcsname\@nameauth@csbc!MN\endcsname
            \unless\ifcsname\@nameauth@csbc!NF\endcsname
              \@nameauth@Index[see]{\@nameauth@csbc}
                {\@nameauth@B\@nameauth@space\@nameauth@C}
                {\@nameauth@Target}%
              \csgdef{\@nameauth@csbc!PN}{}%
            \else
              \PackageWarning{nameauth}
              {\string\IndexRef: extant name;
               no xref \@nameauth@Temp}%
            \fi
          \else
            \PackageWarning{nameauth}
            {\string\IndexRef: extant name;
             no xref \@nameauth@Temp}%
          \fi
        \else
```

Extra verbose warnings and old behavior

```
          \if@nameauth@Verbose
            \PackageWarning{nameauth}
            {\string\IndexRef: non-strict xref \@nameauth@Temp}%
          \fi
          \@nameauth@Index[see]{\@nameauth@csbc}
            {\@nameauth@B\@nameauth@space\@nameauth@C}
            {\@nameauth@Target}%
          \csgdef{\@nameauth@csbc!PN}{}%
        \fi
      \fi
    \fi
  }%
```

Western name

```
  {%
    \def\@nameauth@Temp{\expandafter\detokenize\expandafter{#1 #2}}%
    \ifcsname\@nameauth@csab!PN\endcsname
```

Verbose warnings: Does an exclusion or cross-reference exist?

```
      \if@nameauth@Verbose
        \edef\@nameauth@testex
          {\csname\@nameauth@csab!PN\endcsname}%
        \ifx\@nameauth@testex\@nameauth@Exclude
          \PackageWarning{nameauth}
          {\string\IndexRef: exclusion exists \@nameauth@Temp}%
        \else
          \PackageWarning{nameauth}
          {\string\IndexRef: xref exists \@nameauth@Temp}%
        \fi
      \fi
    \else
```

If no xref or exclusion exists, proceed. Process names without an affix. If creating a *see also* reference, just do it.

```
      \ifx\@nameauth@SB\@empty
        \if@nameauth@SeeAlso
          \@nameauth@Index[seealso]{\@nameauth@csab}
            {\@nameauth@B,\@nameauth@space\@nameauth@A}
            {\@nameauth@Target}%
          \csgdef{\@nameauth@csab!PN}{}%
        \else
```

When creating a *see* reference, check if a name already exists. If so, skip and warn.

```
          \unless\if@nameauth@OldSee
            \unless\ifcsname\@nameauth@csab!MN\endcsname
              \unless\ifcsname\@nameauth@csab!NF\endcsname
                \@nameauth@Index[see]{\@nameauth@csab}
                  {\@nameauth@B,\@nameauth@space\@nameauth@A}
                  {\@nameauth@Target}%
                \csgdef{\@nameauth@csab!PN}{}%
              \else
                \PackageWarning{nameauth}
                {\string\IndexRef: extant name;
                 no xref \@nameauth@Temp}%
              \fi
            \else
              \PackageWarning{nameauth}
              {\string\IndexRef: extant name;
               no xref \@nameauth@Temp}%
            \fi
          \else
```

Extra verbose warnings and old behavior

```
            \if@nameauth@Verbose
              \PackageWarning{nameauth}
              {\string\IndexRef: non-strict xref \@nameauth@Temp}%
            \fi
            \@nameauth@Index[see]{\@nameauth@csab}
              {\@nameauth@B,\@nameauth@space\@nameauth@A}
              {\@nameauth@Target}%
            \csgdef{\@nameauth@csab!PN}{}%
          \fi
        \fi
      \else
```

Process names with an affix. If creating a *see also* reference, just do it.

```
        \if@nameauth@SeeAlso
          \@nameauth@Index[seealso]{\@nameauth@csab}
            {\@nameauth@B,\@nameauth@space%
             \@nameauth@A,\@nameauth@space\@nameauth@SB}
            {\@nameauth@Target}%
          \csgdef{\@nameauth@csab!PN}{}%
        \else
```

When creating a *see* reference, check if a name already exists. If so, skip and warn.

```
          \unless\if@nameauth@OldSee
            \unless\ifcsname\@nameauth@csab!MN\endcsname
              \unless\ifcsname\@nameauth@csab!NF\endcsname
                \@nameauth@Index[see]{\@nameauth@csab}
                  {\@nameauth@B,\@nameauth@space%
                   \@nameauth@A,\@nameauth@space\@nameauth@SB}
                  {\@nameauth@Target}%
                \csgdef{\@nameauth@csab!PN}{}%
              \else
                \PackageWarning{nameauth}
                {\string\IndexRef: extant name;
                 no xref \@nameauth@Temp}%
              \fi
            \else
              \PackageWarning{nameauth}
              {\string\IndexRef: extant name;
               no xref \@nameauth@Temp}%
            \fi
          \else
```

Extra verbose warnings and old behavior

```
            \if@nameauth@Verbose
              \PackageWarning{nameauth}
              {\string\IndexRef: non-strict xref \@nameauth@Temp}%
            \fi
            \@nameauth@Index[see]{\@nameauth@csab}
              {\@nameauth@B,\@nameauth@space%
               \@nameauth@A,\@nameauth@space\@nameauth@SB}
              {\@nameauth@Target}%
            \csgdef{\@nameauth@csab!PN}{}%
          \fi
        \fi
      \fi
    \fi
  }%
```

Reset the *see also* flag to false.

```
  \if@nameauth@OldReset
    \@nameauth@SeeAlsofalse%
  \else
    \global\@nameauth@SeeAlsofalse%
  \fi
}
```

\ExcludeName Prevent a name from being indexed by initializing a regular cross-reference control sequence with the value of \@nameauth@Exclude.

```
\NewDocumentCommand{\ExcludeName}{O{} m O{}}
{%
```

Process and load the arguments into the appropriate macros.

```
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\ExcludeName}%
```

Parse the name arguments and create an excluded xref, unless one already exists.

```
  \@nameauth@Choice
    {\def\@nameauth@Temp{\expandafter\detokenize\expandafter{#2}}}
    {\def\@nameauth@Temp{\expandafter\detokenize\expandafter{#2 #3}}}
    {\def\@nameauth@Temp{\expandafter\detokenize\expandafter{#1 #2}}}%
```

Verbose warnings say that an extant name is being excluded; the operation is allowed.

```
  \if@nameauth@Verbose
    \ifcsname\NameauthPattern!MN\endcsname
      \PackageWarning{nameauth}
      {\string\ExcludeName: extant name \@nameauth@Temp}%
    \fi
    \ifcsname\NameauthPattern!NF\endcsname
      \PackageWarning{nameauth}
      {\string\ExcludeName: extant name \@nameauth@Temp}%
    \fi
  \fi
```

One cannot exclude an extant cross-reference or exclusion. Verbose warnings only.

```
  \ifcsname\NameauthPattern!PN\endcsname
    \if@nameauth@Verbose
      \edef\@nameauth@testex
        {\csname\NameauthPattern!PN\endcsname}%
      \ifx\@nameauth@testex\@nameauth@Exclude
        \PackageWarning{nameauth}
        {\string\ExcludeName: exclusion exists \@nameauth@Temp}%
      \else
        \PackageWarning{nameauth}
        {\string\ExcludeName: xref exists \@nameauth@Temp}%
      \fi
    \fi
   \else
    \csxdef{\NameauthPattern!PN}{\@nameauth@Exclude}%
  \fi
}
```

\IncludeName Allow names to be included as page entries, even if they have been used as cross-
\IncludeName* references.

```
\NewDocumentCommand{\IncludeName}{s O{} m O{}}
{%
  \IfBooleanTF {#1}%
```

The starred form allows any name to be indexed by voiding any exclusion or cross-reference. Process and load the arguments into the appropriate macros. Check for errors, get the current name pattern, then nuke it.

```
  {%
    \@nameauth@LoadArgs{#2}{#3}{#4}%
    \@nameauth@Error{#3}{macro \string\IncludeName*}%
    \@nameauth@Choice{}{}{}%
    \global\csundef{\NameauthPattern!PN}%
  }%
```

The un-starred form allows a name to be indexed once again only if it had been excluded. Process and load the arguments into the appropriate macros. Get the current name type, pattern, and contents if a warning is needed.

```
  {%
    \@nameauth@LoadArgs{#2}{#3}{#4}%
    \@nameauth@Error{#3}{macro \string\IncludeName}%
    \@nameauth@Choice
      {\def\@nameauth@Temp
        {\expandafter\detokenize\expandafter{#3}}}
      {\def\@nameauth@Temp
        {\expandafter\detokenize\expandafter{#3 #4}}}
      {\def\@nameauth@Temp
        {\expandafter\detokenize\expandafter{#2 #3}}}%
```

Test whether the name is an exclusion or a regular xref. If the former, delete its control sequence. If the latter, do nothing and issue a warning.

```
    \ifcsname\NameauthPattern!PN\endcsname
      \edef\@nameauth@testex
        {\csname\NameauthPattern!PN\endcsname}%
      \ifx\@nameauth@testex\@nameauth@Exclude
        \global\csundef{\NameauthPattern!PN}%
      \else
        \if@nameauth@Verbose
          \PackageWarning{nameauth}
          {\string\IncludeName: extant xref \@nameauth@Temp}%
        \fi
      \fi
    \fi
  }%
}
```

`\PretagName` This creates an index sort tag that is applied before a name.

```
\NewDocumentCommand{\PretagName}{O{} m O{} m}
{%
```

Process and load the arguments into the appropriate macros.

```
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\PretagName}%
```

Sort only when permitted. Get the current name type, pattern, and contents if a warning is needed.

```
  \if@nameauth@Pretag
  \@nameauth@Choice
    {\def\@nameauth@Temp{\expandafter\detokenize\expandafter{#2}}}
    {\def\@nameauth@Temp{\expandafter\detokenize\expandafter{#2 #3}}}
    {\def\@nameauth@Temp{\expandafter\detokenize\expandafter{#1 #2}}}%
```

Create the sort tag. Verbose warnings let us know if we are sorting either exclusions or names that are either cross-references or do not exist yet.

```
    \if@nameauth@Verbose
      \edef\@nameauth@testex
        {\csname\NameauthPattern!PN\endcsname}%
      \ifx\@nameauth@testex\@nameauth@Exclude
        \PackageWarning{nameauth}
        {\string\PretagName: tag exclusion \@nameauth@Temp}%
      \else
        \PackageWarning{nameauth}
        {\string\PretagName: tag xref/no exist \@nameauth@Temp}%
      \fi
    \fi
    \csgdef{\NameauthPattern!PRE}{#4\@nameauth@Actual}%
  \else
    \PackageWarning{nameauth}
      {\string\PretagName: deactivated}%
  \fi
}
```

\TagName This creates an index entry tag for a name that is not either an exclusion or a cross-reference.

```
\NewDocumentCommand{\TagName}{O{} m O{} m}
{%
```

Process and load the arguments into the appropriate macros. Get the current name type, pattern, and contents if a warning is needed.

```
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\TagName}%
  \@nameauth@Choice
    {\def\@nameauth@Temp{\expandafter\detokenize\expandafter{#2}}}
    {\def\@nameauth@Temp{\expandafter\detokenize\expandafter{#2 #3}}}
    {\def\@nameauth@Temp{\expandafter\detokenize\expandafter{#1 #2}}}%
```

Verbose warnings let us know if we are sorting either exclusions or cross-references. Do not create a tag if that is the case; otherwise, create a tag.

```
  \ifcsname\NameauthPattern!PN\endcsname
    \if@nameauth@Verbose
      \edef\@nameauth@testex
        {\csname\NameauthPattern!PN\endcsname}%
      \ifx\@nameauth@testex\@nameauth@Exclude
        \PackageWarning{nameauth}
        {\string\TagName: no tag, exclusion \@nameauth@Temp}%
      \else
        \PackageWarning{nameauth}
        {\string\TagName: no tag, xref \@nameauth@Temp}%
      \fi
    \fi
  \else
    \csgdef{\NameauthPattern!TAG}{#4}%
  \fi
}
```

\UntagName This deletes an index tag.

```
\NewDocumentCommand{\UntagName}{O{} m O{}}
{%
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\UntagName}%
  \@nameauth@Choice{}{}{}%
  \global\csundef{\NameauthPattern!TAG}%
}
```

13.8.5 Name 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.

```
\NewDocumentCommand{\NameAddInfo}{O{} m O{} +m}
{%
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\NameAddInfo}%
  \@nameauth@Choice{}{}{}%
  \csgdef{\NameauthPattern!DB}{#4}%
}
```

\NameQueryInfo This prints the information created by \NameAddInfo if it exists.

```
\NewDocumentCommand{\NameQueryInfo}{O{} m O{}}
{%
  \unless\if@nameauth@BigLock
    \protected@edef\@nameauth@testname{\trim@spaces{#2}}%
    \unless\ifx\@nameauth@testname\@empty
      \@nameauth@Error{#2}{macro \string\NameQueryInfo}%
      \@nameauth@LoadArgs{#1}{#2}{#3}%
      \@nameauth@Choice{}{}{}%
    \fi
    \ifcsname\NameauthPattern!DB\endcsname
      \expandafter\csname\NameauthPattern!DB\endcsname%
    \fi
  \fi
}
```

\NameClearInfo This deletes a text tag. It has the same structure as \UntagName.

```
\NewDocumentCommand{\NameClearInfo}{O{} m O{}}
{%
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\NameClearInfo}%
  \@nameauth@Choice{}{}{}%
  \global\csundef{\NameauthPattern!DB}%
}
```

13.8.6 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. First we load the arguments into the standard macros, check for error, and get the current name pattern.

```
\NewDocumentCommand{\IfMainName}{O{} m O{} +m +m}
{%
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\IfMainName}%
  \@nameauth@Choice{}{}{}%
```

Take this path if the pattern exists.

```
  \ifcsname\NameauthPattern!MN\endcsname
    \if@nameauth@GlobalScope #4\else {#4}\fi
  \else
```

Take this path if the pattern does not exist.

```
    \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. First we load the arguments into the standard macros, check for error, and get the current name pattern.

```
\NewDocumentCommand{\IfFrontName}{O{} m O{} +m +m}
{%
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\IfFrontName}%
  \@nameauth@Choice{}{}{}%
```

Take this path if the pattern exists.

```
  \ifcsname\NameauthPattern!NF\endcsname
    \if@nameauth@GlobalScope #4\else {#4}\fi
  \else
```

Take this path if the pattern does not exist.

```
    \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. First we load the arguments into the standard macros, check for error, and get the current name pattern.

```
\NewDocumentCommand{\IfAKA}{O{} m O{} +m +m +m}
{%
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\IfAKA}%
  \@nameauth@Choice{}{}{}%
  \ifcsname\NameauthPattern!PN\endcsname
    \edef\@nameauth@testex
      {\csname\NameauthPattern!PN\endcsname}%
```

Take this path if the pattern is an exclusion.

```
    \ifx\@nameauth@testex\@nameauth@Exclude
      \if@nameauth@GlobalScope #6\else {#6}\fi
    \else
```

Take this path if the pattern exists.

```
      \if@nameauth@GlobalScope #4\else {#4}\fi
    \fi
  \else
```

Take this path if the pattern does not exist.

```
    \if@nameauth@GlobalScope #5\else {#5}\fi
  \fi
}
```

\ForgetName This undefines a control sequence to force a “first use”.

```
\NewDocumentCommand{\ForgetName}{O{} m O{}}
{%
```

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, destroying the control sequences either by current name system type or completely. `@nameauth@LocalNames` toggles current system or both, while we select the type of name with `@nameauth@MainFormat`.

```
  \@nameauth@Choice{}{}{}%
  \if@nameauth@LocalNames
    \if@nameauth@MainFormat
      \global\csundef{\NameauthPattern!MN}%
    \else
      \global\csundef{\NameauthPattern!NF}%
    \fi
  \else
    \global\csundef{\NameauthPattern!MN}%
    \global\csundef{\NameauthPattern!NF}%
  \fi
}
```

\SubvertName This defines a control sequence to force a “subsequent use”.

```
\NewDocumentCommand{\SubvertName}{O{} m O{}}
{%
  \@nameauth@LoadArgs{#1}{#2}{#3}%
  \@nameauth@Error{#2}{macro \string\SubvertName}%
```

Now we parse the arguments, defining the control sequences either by current name system type or completely. `@nameauth@LocalNames` toggles current system or both, while we select the type of name with `@nameauth@MainFormat`.

```
  \@nameauth@Choice{}{}{}%
  \if@nameauth@LocalNames
    \if@nameauth@MainFormat
      \csgdef{\NameauthPattern!MN}{}%
    \else
      \csgdef{\NameauthPattern!NF}{}%
    \fi
  \else
    \csgdef{\NameauthPattern!MN}{}%
    \csgdef{\NameauthPattern!NF}{}%
  \fi
}
```

13.8.7 Pseudonyms

\AKA \AKA prints an alternate name and creates index cross-references. The starred form
\AKA* displays the alternate name like \FName.

```
\NewDocumentCommand{\AKA}{s O{} m O{} m O{}}
{%
```

Prevent entering \AKA via itself or \@nameauth@Name. Tell the formatting system that \AKA is running. Prevent and reset \JustIndex.

```
  \if@nameauth@BigLock
    \@nameauth@Locktrue%
  \fi
  \IfBooleanTF {#1}{\@nameauth@AltAKAtrue}{}%
  \unless\if@nameauth@Lock
    \@nameauth@Locktrue%
    \@nameauth@InAKAtrue%
    \if@nameauth@OldReset
      \@nameauth@JustIndexfalse%
    \else
      \global\@nameauth@JustIndexfalse%
    \fi
```

Test for malformed input, then load args and save the persistent info of the xref.

```
    \@nameauth@Error{#3}{macro \string\AKA}%
    \@nameauth@Error{#5}{macro \string\AKA}%
    \@nameauth@LoadArgs{#4}{#5}{#6}%
    \@nameauth@Choice{}{}{}%
```

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{#2}}%
    \protected@edef\@nameauth@Bi{\@nameauth@Root{#3}}%
    \protected@edef\@nameauth@Si{\@nameauth@Suffix{#3}}%
    \@nameauth@Parse{#4}{#5}{#6}{!PN}%
```

Create an index cross-reference based on the arguments.

```
    \unless\if@nameauth@SkipIndex
      \ifx\@nameauth@Ai\@empty
        \ifx\@nameauth@Si\@empty
          \IndexRef[#4]{#5}[#6]{\@nameauth@Bi}%
        \else
          \IndexRef[#4]{#5}[#6]
          {\@nameauth@Bi\@nameauth@space\@nameauth@Si}%
        \fi
      \else
        \ifx\@nameauth@Si\@empty
          \IndexRef[#4]{#5}[#6]
          {\@nameauth@Bi,\@nameauth@space\@nameauth@Ai}%
        \else
          \IndexRef[#4]{#5}[#6]
          {\@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
}
```

\PName
\PName*

`\PName` is a convenience macro that calls `\NameauthName`, then `\AKA`. Its starred form prints a long name.

```
\NewDocumentCommand{\PName}{s O{} m O{} m O{}}
{%
```

If we have a starred form, we will display a long name. If we used `\JustIndex`, we ignore and reset its flag to false.

```
  \IfBooleanTF {#1}{\@nameauth@FullNametrue}{}%
  \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[#2]{#3} (\SkipIndex\AKA[#2]{#3}[#4]{#5}[#6])%
  \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[#2]{#3}
      \@nameauth@SeeAlsotrue(\AKA[#2]{#3}[#4]{#5}[#6])%
    \else
      \NameauthName[#2]{#3}
      (\AKA[#2]{#3}[#4]{#5}[#6])%
    \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}
    {\string\SkipIndex still active after \string\PName; check}%
  \fi
}
```

14 Change History

Minor changes, such as only to documentation and example files, are not shown.

15 Index

Page numbers in *italic* refer to the page where the corresponding entry is **described.** Page numbers that are underlined refer to the **code line** of the definition. Unmodified roman page numbers are used by names and other entries in the documentation.

S

T

U

V

W

Y

Z