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

Charles P. Schaum[†]

Released 2024/02/09

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

1 Quick Start **4**
1.1 Simple Example 4
1.2 How To Use the Manual . . . 6
1.3 Basic Concepts 7
1.3.1 Name Arguments . . . 7
1.3.2 Name Ambiguity . . . 8
1.4 Basic Interface 10
1.4.1 Western Names 10
1.4.2 Reversed Western . . 12
1.4.3 Eastern Names 13
1.4.4 Ancient Names 14
1.5 Quick Interface 15
1.5.1 Name Shorthands . . 15
1.5.2 Name Variants 17
1.5.3 *Alternate* Field 18
1.6 Select Macro Overview 19
1.6.1 With Name Args . . . 19
1.6.2 Prefix Macros 20
1.7 Names and Complexity . . . 21

2 Package Options **22**
2.1 Name Grammar and Syntax . 22
2.1.1 Affix Commas 22
2.1.2 Surname in Caps . . . 22
2.1.3 Reverse Name Order . 22
2.2 Indexing 23
2.2.1 Toggle Indexing 23
2.2.2 Toggle Index Sorting . 23
2.2.3 Verbose Warnings . . 23
2.3 Formatting 23
2.3.1 Choose System 23
2.3.2 Predefined Hooks . . . 24
2.4 Alternate Formatting 24
2.5 Scope of Decisions 24
2.6 Version Compatibility 25

3 Feature Priority **26**

4 Naming Macros **27**
4.1 `\Name` and `\Name*` 27
4.2 Forenames: `\FName` 28
4.3 Technical Details 29
4.3.1 Final Optargs 29
4.3.2 Name Arg Caveats . . 31
4.3.3 Full Stop Detection . 31
4.3.4 Grouping, Spaces . . . 32
4.3.5 Formatting Initials . . 33

5 Language Topics **33**
5.1 Active Characters 33
5.2 Hyphenation 34
5.3 Affixes Need Commas 35
5.4 Eastern Names, Name Caps . 36
5.5 Reversed Names 37
5.6 Listing by Surname 38
5.7 Particles in Names 38
5.7.1 Standard Rules 38
5.7.2 Non-Breaking Spaces . 39
5.7.3 Look-Alike Particles . 39
5.7.4 Capitalizing 39
5.8 Medieval Names 39

[*]This file describes version 4.0, last revised 2024/02/09.
[†]Email: charles[dot]schaum@comcast.net

6 Debugging **41**
6.1 Name Patterns 41
6.2 Indexing: States 43
6.3 Debugging Macros 45

7 Indexing Macros **49**
7.1 General Control 49
7.1.1 Toggle Indexing 49
7.1.2 Multiple Indexes . . . 49
7.1.3 Verbose Warnings . . 49
7.1.4 Index Protection . . . 50
7.2 Page Entries 50
7.3 Cross-References 51
7.3.1 Basic Macro 51
7.3.2 Fine Control 52
7.4 Prefix Macros 54
7.5 Automatic Rules 55
7.6 Sorting Names 56
7.6.1 General Approach . . 56
7.6.2 Sorting Initials 58
7.7 Index Tags 58
7.7.1 General Approach . . 58
7.7.2 Identical Names . . . 59
7.7.3 Special Tags 59
7.8 Categories/Sub-entries 61

8 Name Tags **65**

9 Formatting and Decisions **67**
9.1 Basic Formatting 67
9.2 Application: Footnotes 69
9.3 Making Name Decisions . . . 70
9.4 Formatting and Decisions . . 72
9.5 Testing Name Decisions . . . 73
9.5.1 Testing Macros 73
9.5.2 Applications 75
9.5.3 Beamer Example . . . 77

10 Name Authority Basics **80**
10.1 Variant Names 80
10.1.1 Alternate Argument . 80
10.1.2 Multiple Variants . . . 81
10.1.3 Nonstandard Caps . . 83
10.1.4 Variants and Xrefs . . 83
10.2 Using a Name Authority . . . 85

11 Advanced Formatting **86**
11.1 Formatting Hooks 87
11.2 Name Tags in Hooks 88
11.2.1 Hook Templates . . . 89
11.2.2 Ancient Names 90
11.2.3 Life Dates 91
11.3 Alternate Formatting 93
11.3.1 Enabling/Disabling . . 93
11.3.2 Using `\noexpand` . . . 94
11.3.3 Capitalization 95
11.3.4 Formatting Features . 95
11.3.5 History Text 97
11.3.6 Inflected Names . . . 98
11.3.7 Reference Work I . . . 99
11.4 Roman Names 101
11.4.1 General Market 101
11.4.2 Student Reference . . 102
11.4.3 Scholarly Reference . . 107
11.5 Special Uses 109
11.5.1 Reference Work II . . 110
11.5.2 Marginalia 113

12 Planned Obsolescence **116**
12.1 Pseudonyms 116
12.1.1 Special Syntax 116
12.1.2 The Macros 116
12.1.3 Workarounds 118
12.2 Obsolete Syntax 119

13 Advanced Customization **121**
13.1 Using Package Internals . . . 121
13.2 Using Separate Macros 122
13.3 Full Customization 125
13.3.1 Names In Boxes . . . 125
13.3.2 Change Parsing 127

14 Technical Notes and Tips **128**
14.1 Tips: General 128
14.2 Tips: Indexing 128
14.3 Tips: Name Args 129
14.4 Active Unicode 130
14.4.1 General Information . 130
14.4.2 Compatibility 132
14.4.3 Fragility 133

15 Implementation **134**
15.1 Boolean Flags 134
15.1.1 Flow Control 134
15.1.2 Syntax 134
15.1.3 Debugging 136
15.1.4 Indexing 136
15.1.5 Formatting 136
15.1.6 Name Decisions 137
15.1.7 Compatibility 137
15.2 Registers, Hooks, Values . . . 137
15.2.1 Token Registers 137
15.2.2 Hooks 138
15.2.3 Internal Values 139
15.3 Package Options 139
15.3.1 Syntax 139
15.3.2 Indexing 139
15.3.3 Formatting 139
15.3.4 Predefined Hooks . . . 140
15.3.5 Alternate Format . . . 140
15.3.6 Scope 140
15.3.7 Compatibility 140
15.4 Package Initialization 140

15.5 Internal Macros 141
15.5.1 Fundamental Macros . 141
15.5.2 Errors/Debugging . . 149
15.5.3 Core Name Engine . . 150
15.5.4 Indexing 155
15.6 User Macros: Prefixes 156
15.6.1 Syntax 156
15.6.2 Indexing 157
15.6.3 Format/Decisions . . . 158
15.7 User Macros: Helpers 158
15.7.1 Syntax 158
15.7.2 Indexing 159
15.7.3 Formatting 159
15.7.4 Alternate Format . . . 159
15.7.5 Name Decisions 160
15.7.6 Name Parser 161
15.8 User Macros: Name Args . . . 162
15.8.1 Basic Interface 162
15.8.2 Quick Interface 162
15.8.3 Debugging Macros . . 164
15.8.4 Indexing 168
15.8.5 Name Tags 178
15.8.6 Name Decisions 178
15.8.7 Pseudonyms 181

16 Change History 183

17 Index 186

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 express or imply any disrespect or bias.

Compatibility

Starting with version 4.0, `nameauth` uses `xparse` for starred macros and name arguments. Benefits include better macro design, more control over arguments, less code duplication, and better customization.

- Use with a pre-2017 TeX distribution might cause occasional errors. See also Sections 4.3.1, 13.3.2 and 14.4.2.
- Older user customizations using `xargs` may still work with the `oldargs` option. See Sections 2.6 and 13.3.2.

If one should require version 3.7, see section 7 of `README.md`.

Fair Use

Other works quoted herein either are in the public domain or they are copyrighted works cited under the terms of fair use, to which the present author claims no copyright. The purpose is to show the power of words and names.

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)

1 Quick Start

A **name authority** is a canonical, scholarly list of names to which variant name forms and aliases refer. Books can contain hundreds of names and index entries. The nameauth package assists academic and business writing to minimize the work of managing many names:

- Automate the display and formatting of names.
- Manage and display information that is associated with names.
- Make decisions relating to names (name control sequence patterns).
- 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.
- Adopt name forms in the text and index that are culturally appropriate.
- Do not force the user to adopt any one culture's naming conventions.
- Permit European academic conventions ("Continental" formatting).

Indexing rules implemented by nameauth are based on Nancy C. Mulvany, *Indexing Books* (Chicago: University of Chicago Press, 1994). All references [Mulvany] refer to this edition. See also similar information in *The Chicago Manual of Style* (15th ed., Chicago: Chicago UP, 2003, 309f., 755f.) or newer editions; referenced as [Chicago].

1.1 Simple Example

We begin with an excerpt from a biography written by Charles Waddell Chesnutt, a prominent African-American author at the turn of the twentieth century.[1] We create a new document, use only the default nameauth options, and create no extra formatting for names (cf. Section 9.1).

The nameauth environment (Section 1.5) in the preamble resembles a tabular with four columns. In column one we define the naming macros. In columns two and three we define the names that the macros will display. We leave column four empty.

We use \Forgetname (Section 9.3) in Group 2 to "forget" about the names in Group 1. When we reorder the statements, the names change form automatically.

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

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

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

```

\begin{document}

\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}\\
\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
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.

Right Is of No Sex — Truth Is of No Color — God Is the Father of Us All, and All We Are Brethren.

—Frederick Douglass
motto, *The North Star* (Rochester, NY, 1847)

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 Arguments}	[Optional Arguments]
This manual shows mandatory arguments in black.	This manual shows optional arguments in dark red.

Scope and Sequence

The table of contents indicates what topics are covered, the simpler and frequent to the complex and infrequent. Later topics require knowledge from multiple sections. The end of each major section includes a return link to the table of contents.

Key Concepts

Starting in Section 1.4, we use debugging macros to show **name control patterns** in the margins (Section 6.1). These patterns are the key to how names work in nameauth. Through Section 5.8 we show **basic index entries** in the margins in order to illustrate the things for which one should look when debugging.

Special Signs

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

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

† 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 caution is needed.

Example Files

The files `examples.tex` and `compat.tex` are located with this manual. For generating package testing files, see `README.md`, also located with this manual.

Thanks

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

In memoriam Robin Fairbairns

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

1.3 Basic Concepts

This section introduces fundamental concepts needed by package users. The rest of the manual requires these to be understood.

1.3.1 Name Arguments

Below we show how familiar categories of names map to name arguments in the appropriate macros, as defined by the current syntax. We use the excellent advice in [Mulvany, 152–82] and [Chicago], adapting them for use in LaTeX.

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

Forename(s):⟨*FNN*⟩

Personal name(s):
baptismal name
Christian name
multiple names
praenomen[2]

Surname(s):⟨*SNN*⟩

Family name:
of father, mother
ancestor, vocation
origin, region
nomen / cognomen
patronym

Qualifier:⟨*Affix*⟩

Sobriquet / title:
Sr., Jr., III...
notable attribute
origin, region

Alias:⟨*Alternate*⟩

Replaces ⟨*FNN*⟩ only in the text.

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

Family:⟨*SNN*⟩

Family / clan name

Personal:⟨*Affix*⟩

Usually one name

Alias:⟨*Alternate*⟩

Replaces ⟨*Affix*⟩ only in text.[3]

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

Personal:⟨*SNN*⟩

Given name(s)

Qualifier:⟨*Affix*⟩

Sobriquet / title:
Sr., Jr., III...
notable attribute
origin, region
patronym

Alias:⟨*Alternate*⟩

Replaces ⟨*Affix*⟩ only in text.[4]

---

[2] There are several ways of handling the Roman *tria nomina*. See Section 11.4.
[3] The obsolete syntax uses ⟨*Alternate*⟩ instead of ⟨*Affix*⟩ for a personal name (Section 12.2).
[4] The obsolete syntax uses ⟨*Alternate*⟩ instead of ⟨*Affix*⟩ for a qualifier (Section 12.2).

1.3.2 Name Ambiguity

- **Name forms are ambiguous.**

  In Western culture, one often sees forenames followed by a surname. Patronyms, as in Leif Erikson and Llywelyn ap Gruffudd, and other exceptions exist.[5] Some readers might assume the following:

Forename	Forename	Surname
Marcus	Tullius	Cicero
Forename		**Surname**
Pontius		Pilate
Jesus		Christ

  Even though all these initial assumptions are false, Western naming conventions tend to dominate the general readership market and, *mutatis mutandis*, they affect the use and indexing of names.

- **Cultural context resolves ambiguity.**

  Here we see how the names above have meaning in their proper context. The words themselves *as signs* do not offer the as many clues about their meaning until the signs become part of a cultural landscape.

Personal Name	Clan Name	Branch Family or Nickname
Marcus	Tullius	Cicero
	Pontius	Pilate
Personal Name		**Sobriquet**
Jesus		Christ

- **This package embraces such ambiguities.**

  Roman *praenomina* did not have the same significance as Western forenames.[6] Yet the family name (*nomen*) of the man we call Cicero is Tullius. In English, he used to be known as Tully. In this manual we choose the popular case for the *cognomen* Cicero as a Western surname. Scholarly publications can take at least two different approaches (see Section 11.4).

  The name Pontius Pilate has no personal name, which fits with Roman naming trends. Pontius is a clan name. Most texts and inscriptions favor the *cognomen* Pilatus, denoting martial prowess.

  Jesus Christ is a name derived from Greek Ἰησοῦς Χριστός, from *Y'shua ha-Mashiach*, Joshua the Anointed One. The personal name is Jesus. Other added names are titles or descriptors.

- **Macros can modify names in the text to fit cultural norms.**

  Name forms in the text are independent of their index entries. This is necessary for special cases like Hungarian names. Thus, a text oriented to Hungarian readers can talk about Liszt Frenec† (Franz Liszt) in the body text, but he can be indexed as Liszt, Frenec. One can do something similar with East Asian names, if Western index entries are required.

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

[6] See Wikipedia. Roman names in particular are sensitive to the historical and political development of Rome and its empire, the history of Roman families and culture, as well as other factors.

- **Name arguments determine index entry forms.**

  The way that one puts names into macro name arguments absolutely determines the entry form in the index:

Printed Name	Macro and Arguments	Western Index Entry
Person Family Affix	`\Name[Person]{Family, Affix}`	Family, Person, Affix
Person Family	`\Name[Person]{Family}`	Family, Person
Family Person	`\RevName%` `\Name[Person]{Family}`	Family, Person

Printed Name	Macro and Arguments	Nonwestern Index Entry
Family Person	`\Name{Family, Person}`	Family Person
Person Family	`\RevName%` `\Name{Family, Person}`	Family Person
Person Affix	`\Name{Person, Affix}`	Person Affix
Person	`\Name{Person}`	Person

  The following choices reflect non-scholarly English-language books. Later sections will show different ways to encode name arguments for scholarly works or non-English books.

Printed Name	Macro and Arguments	Western Index Entry
M.T. Cicero	`\Name[M.T.]{Cicero}`	Cicero, M.T.

Printed Name	Macro and Arguments	Nonwestern Index Entry
Pontius Pilate	`\Name{Pontius, Pilate}`	Pontius Pilate
Jesus Christ	`\Name{Jesus, Christ}`	Jesus Christ

- **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.

  Indeed, it is only through the process of making nameauth that the present author became aware of the many intricate and fascinating complexities of names. The hope is that this package might facilitate accurate and respectful cross-cultural use of names in quality publications.

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)

1.4 Basic Interface

The description of macro arguments in this section applies to all nameauth macros that take name arguments (Sections 1.3.1 and 1.6.1), making this section critical to using and mastering nameauth.

- 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.
- Extra spaces around each argument are stripped.
- Include name arguments consistently to have consistent index entries.
- For all name forms, see Section 4.3.1 regarding final optional arguments.

1.4.1 Western Names

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

Within nameauth, Western names have distinct features:

- Western names must use the first optional ⟨*FNN*⟩ argument.
- They require a comma to delimit any affixes (Section 5.3).
- Western index entries have two general forms:

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

- They have Western name patterns (Section 6.1) and index entry forms.

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.[7] 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}`
Later use:	George Washington	`\Name*[George]{Washington}`
Later use:	Washington	`\Name [George]{Washington}`
Later use:	George	`\FName[George]{Washington}`
First use:	George S. Patton Jr.	`\Name [George S.]{Patton, Jr.}`
Later use:	George S. Patton Jr.	`\Name*[George S.]{Patton, Jr.}`
Later use:	Patton	`\Name [George S.]{Patton, Jr.}`
Later use:	George S.	`\FName[George S.]{Patton, Jr.}`

[7]The intent is that one can just add an `F` to either `\Name` or `\Name*`.

Affixes and Alternate Forms

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 long name instance (forced by `\Name*`):

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

- Use an alternate forename in a long- or forename instance:

First use: John Davison Rockefeller IV
`\Name[J.D.]{Rockefeller, IV}[John Davison]`
Later use: George
`\FName[George S.]{Patton, Jr.}[George]`

- Drop the affix and alter the forenames:

First use: Jay Rockefeller
`\DropAffix\Name*[J.D.]{Rockefeller, IV}[Jay]`
Later 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}`
Later use: C.S. Lewis
`\Name*[Clive Staples]{Lewis}[C.S.]`
Later use: Jack
`\FName[Clive Staples]{Lewis}[Jack]`

To sort the index consistently and properly, all names should be sorted by their longest unique name forms and by the Arabic equivalents of Roman numerals. See Section 5.8, 7.6, 7.6.2, 10.1. and all of Section 11. For example:

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

The alternate domination of one faction over another, sharpened by the spirit of revenge, . . . 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)

1.4.2 Reversed Western Names

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

Reversed Western names (Section 5.5) have these features:

- They must use the first optional ⟨*FNN*⟩ argument.
- Avoid using affixes in order to avoid odd name forms. One also could use `\DropAffix` (Section 5.3), but that would not affect index entries.
- Index entries have the Western form: ⟨*SNN*⟩, ⟨*FNN*⟩.
- They have Western name patterns and index entry forms.
- They do not work with the obsolete syntax (Section 12.2).

Starting with Western Name Forms

These reversed Western forms are used optimally in a context where Hungarian names and similar cases of name order appear in a document because their index entries take a Western form [Mulvany, 166].[8]

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

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

Using the Reversing Macros

We use the prefix macros `\RevName` and optionally `\CapName` (Section 1.6) to print either a Hungarian or "non-native" Eastern name in the text while keeping Western forms in the index:

Same name patterns and index entries as above.

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

These macros, as is the case with many `nameauth` macros, work properly in context, not arbitrarily. **They always have Western index entries,** regardless of how they appear in the text.

[8] Regarding the margin note that shows name control sequences, with `pdflatex` and `latex`, in `Frenec!MolnÃąr` the glyphs `Ãą` correspond to `\IeC{\'a}`.

1.4.3 Eastern Names

All nonwestern name forms in nameauth have the syntax: ⟨*SNN,Affix*⟩. In Eastern names, ⟨*SNN*⟩ refers to a family name and ⟨*Affix*⟩ to a personal name. Otherwise, ⟨*SNN*⟩ refers to a person's name and ⟨*Affix*⟩ to added information. Knowing this difference helps one avoid nonsense names.

	Required ⟨*SNN*⟩ required ⟨*Affix*⟩	Optional (text only)
`\Name` `\Name*` `\FName`	`{`⟨***SNN, Affix***⟩`}`	`[`⟨***Alternate***⟩`]`

These features denote "native" Eastern names in nameauth (Sections 5.4, 5.5):

- They must **leave empty** the ⟨*FNN*⟩ argument.
- They use instead the ⟨*SNN, Affix*⟩ arguments.
- Their index entries take the nonwestern form: ⟨*SNN Affix*⟩.
- Names with the form ⟨*SNN, Affix*⟩ can use the ⟨*Alternate*⟩ argument to swap ⟨*Affix*⟩ with ⟨*Alternate*⟩.
- They have nonwestern name patterns and index entry forms.

"Native", Reversible 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}
Later use:  Miyazaki Sensei .............\Name*{Miyazaki, Hayao}[Sensei]
Later use:  Miyazaki ...............................\FName{Miyazaki, Hayao}
```

One must use `\ForceFN` with `\FName` (Section 4.2) to get a personal name. ⟨*Alternate*⟩ swaps with ⟨*FNN*⟩ (in both long forms and in short forms) in the text only. ⟨*Alternate*⟩ does not work with the obsolete syntax (Section 12.2):

Same name patterns
and index entries
as above.

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

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

1.4.4 Royal, Medieval, and Ancient Names

	Required ⟨*SNN*⟩ optional ⟨*Affix*⟩	Optional, special
`\Name` `\Name*` `\FName`	{⟨***SNN**, **Affix***⟩}	[⟨***Alternate***⟩]

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

- They must **leave empty** the ⟨*FNN*⟩ argument.
- They use either the ⟨*SNN, Affix*⟩ arguments or just ⟨*SNN*⟩.
- Their index entries take the nonwestern forms: ⟨*SNN Affix*⟩ or ⟨*SNN*⟩.
- Names with the form ⟨*SNN, Affix*⟩ can use the ⟨*Alternate*⟩ argument to swap ⟨*Affix*⟩ with ⟨*Alternate*⟩.
- Names with the form ⟨*SNN*⟩ should not use ⟨*Alternate*⟩ (cf. Section 12.2).
- They have nonwestern name patterns and index entry forms.
- One generally does not reverse these names (Section 5.5).

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 nonsense names in the text.

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

- Here we work with titles and sobriquets:

  First use: Elizabeth I "Gloriana"
  `\ForgetThis\Name{Elizabeth, I}[I ‘‘Gloriana’’]`
  Later use: Gloriana
  `\ForceFN\FName{Elizabeth, I}[Gloriana]`

- Here we show a non-royal:

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

- These are nonsensical name forms:

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

- The trivial case:

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

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:

```
\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:

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

`\White` produces E. B. White, the version with the thin space. We lost the first version of the name when we redefined it.[9]

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 (usually prepositions or clan designators), which are discussed in greater detail in Section 5.7.

[9]When building this package there should be a warning: `Shorthand macro already exists`. This is intentional, meant to test if the warning is working properly.

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           &         >
  \< JWG      & J.W. von      & Goethe            &         >
  \< VBuren   & Martin        & Van Buren         &         >
% Reversed Western Forms
  \< Noguchi  & Hideyo        & Noguchi           &         >
  \< Molnar   & Frenec        & 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:

Output		Macro
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}`
George Washington	Quick:	`\ForgetThis\Wash`
	Basic:	`\ForgetThis\Name[George]{Washington}`
Washington	Quick:	`\SubvertThis\Wash`
	Basic:	`\SubvertName[George]{Washington}`
		`\Name[George]{Washington}`
Washington	Quick:	`\SubvertThis\Wash`
	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. We hide the use of `\ForgetThis` (Section 9.3), which creates first instances.

Name Pattern(s):
`George!Washington`
`GeorgeS.!Patton,Jr.`
`J.D.!Rockefeller,IV`
`CliveStaples!Lewis`
`Hideyo!Noguchi`
`Miyazaki,Hayao`
Basic Index:
Washington, George
Patton, George S., Jr.
Rockefeller, J.D., IV
Lewis, Clive Staples
Noguchi, Hideyo
Miyazaki Hayao

- Western Names: Sections 4.1, 4.2

First use:	George Washington	`\Wash`
Later use:	George Washington	`\LWash`
Later use:	Washington	`\Wash`
Later use:	George	`\SWash`

- Nicknames and Affixes: Sections 4.2, 5.3

First use:	George S. Patton	`\DropAffix\Patton`
Later use:	George S. Patton Jr.	`\LPatton`
Later use:	Patton	`\Patton`
Later use:	George S. Patton	`\DropAffix\LPatton`
Later use:	George Patton	`\DropAffix\LPatton[George]`
Later use:	George S.	`\SPatton`
Later use:	George	`\SPatton[George]`
First use:	John Davison Rockefeller IV	`\JRIV[John Davison]`
Later use:	J.D. Rockefeller IV	`\LJRIV`
Later use:	Jay Rockefeller	`\DropAffix\LJRIV[Jay]`
First use:	Clive Staples Lewis	`\Lewis`
Later use:	C.S. Lewis	`\LLewis[C.S.]`
Later use:	Jack Lewis	`\LLewis[Jack]`
Later use:	Clive Staples	`\SLewis`
Later use:	Jack	`\SLewis[Jack]`
Later use:	C.S. Lewis	`\LCSL`
Later use:	C.S.	`\SCSL`

- "Native" Eastern Names: Section 5.4

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

- Reversed Western Names: Section 5.5

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

- Western Names Reversed by Surname: Section 5.6

First use:	Washington, George	`\RevComma\LWash`
Later use:	Washington, George	`\RevComma\LWash`

Name Pattern(s):
`Hernando!de~Soto`
`Aristotle`
`ÃĘthelred,II`
`John,Eriugena`
Basic Index:
de Soto, Hernando
Aristotle
Æthelred II
John Eriugena

- Particles: Section 5.7

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

- Royal and Medieval Names: Section 5.8

First use:	Æthelred II[10]	`\Aeth`
Later use:	Æthelred	`\Aeth`
Later use:	Æthelred II, "Unræd"	`\LAeth[II, ‘‘Unræd’’]`
First use:	John Scotus Eriugena	`\Eriugena[Scotus Eriugena]`
Later use:	John Eriugena	`\LEriugena`
Later use:	John	`\Eriugena`
First use:	Aristotle	`\Aris`
Later use:	Aristotle	`\Aris`

1.5.3 ⟨*Alternate*⟩ Name Field Tips

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 control patterns (Section 6.1).

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

- 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 reason why these macro versions produce undesired output is because ⟨*arg4*⟩ permanently populates ⟨*Alternate*⟩.
- Remember that ⟨*arg4*⟩ can be used for the obsolete syntax, as mentioned previously, but we do not cover that here.

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)

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

1.6 Select Macro Overview

1.6.1 Macros with Name Arguments

All macros that take name arguments (Section 1.3.1) can have final optional arguments (Section 4.3.1) and update `\NameauthPattern` (Section 11.2.1). The ⟨*xref args*⟩ are the same as ⟨*name args*⟩ for a cross-reference to a ⟨*target*⟩ (Section 7.3).

	Optional Prefix	Macro	Star	Arguments	
Naming	⟨*prefix macros*⟩	`\Name`	*	⟨*name args*⟩	
	⟨*prefix macros*⟩	`\FName`	*	⟨*name args*⟩	
Page entry	`\SeeAlso`	`\IndexName`		⟨*name args*⟩	
Only cross-ref	`\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*⟩	
Delete name tag		`\NameClearInfo`		⟨*name args*⟩	
Delete name cs		`\ForgetName`		⟨*name args*⟩	
Create name cs		`\SubvertName`		⟨*name args*⟩	
Name cs tests		`\IfMainName`		⟨*name args*⟩	{⟨*y*⟩}{⟨*n*⟩}
		`\IfFrontName`		⟨*name args*⟩	{⟨*y*⟩}{⟨*n*⟩}
		`\IfAKA`		⟨*name args*⟩	{⟨*y*⟩}{⟨*n*⟩}{⟨*x*⟩}
Debugging		`\ShowPattern`		⟨*name args*⟩	
		`\ShowIdxPageref`	*	⟨*name args*⟩	
		`\ShowNameInfo`		⟨*name args*⟩	
		`\ShowNameState`		⟨*name args*⟩	

Optional prefix macros are shown in the next subsection. Not shown above are `\AKA`, `\AKA*`, `\PName`, and `\PName*` (Section 12.1).

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.

—John F. Kennedy, Inaugural Address (1961)

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 in the Text

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

- Commas in the Text

`\ShowComma`	Add comma between ⟨*SNN*⟩ and ⟨*Affix*⟩.
`\NoComma`	No comma between ⟨*SNN*⟩ and ⟨*Affix*⟩. Overrides `\ShowComma`.

- Name Breaks in the Text

`\DropAffix`	Drop affix only for a long Western name instance.
`\KeepAffix`	Insert non-breaking space (NBSP) between ⟨*SNN*⟩, ⟨*FNN/Affix*⟩.
`\KeepName`	Insert NBSP between all name elements. Overrides `\KeepAffix`.

- Forcing Name Forms via Control Sequence

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

- Forcing Name Forms via Boolean Flags

`\ForceName`	Force first-use formatting hooks.
`\ForceFN`	Print ⟨*Affix*⟩/⟨*Alternate*⟩ in nonwestern short forms.

- Indexing

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

Some important notes include:

- Prefix macros stack:
  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.

Macros that do not take name arguments include prefix macros (Section 15.6), helper macros (Section 15.7), most internal package macros, and formatting macros (Sections 9.1 and 11).

1.7 Names and Complexity

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

Elizabeth I "Gloriana" . . . . `\LEliz[I ''Gloriana'']`

We can display the same thing with different macros. This next example does not offer more features; it only offers more complexity:

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

The next example offers more features and better automation. Based on Section 11.2.2 we use name tags and change the formatting hooks to match this manual's conventions. The key here is that **we make a few key changes once,** which then govern many appearances of names:

```
\NameAddInfo{Elizabeth, I}{ ''Gloriana''}
\renewcommand*\NamesFormat[1]
{
  \color{blue}\sffamily #1
  \ifcsname\NameauthPattern!DB\endcsname
    \expandafter\csname\NameauthPattern!DB\endcsname
  \fi
}
\renewcommand*\MainNameHook{\sffamily}
```

First use: Elizabeth I "Gloriana" . . . . `\ForgetThis\LEliz`
Later use: Elizabeth I . . . . `\LEliz`
Later use: Elizabeth . . . . `\Eliz`

Section 5.7 shows a similar trade-off between simplicity and automation with an example using the name of poet e.e. cummings. Moving on to Section 11.4, complexity increases, yet the state of any given name remains well-defined.

Avoid the Rabbit Hole

- In nameauth, names are nouns that have state and modifiers.
- In nameauth, names are verbs capable of changing their environment.
- There are trade-offs between ease of use and automation.
- Simple examples often are not easy to automate.
- Automation works best with a few key changes.
- Most use-cases can be as simple as Section 1.1.
- Use the simple approach unless a complex approach is needed.

Back to Table of Contents

2 Package Options

There are many options for use in `nameauth`, which can be divided into specific areas of functionality. These range from defaults that make conforming to certain standards easier to backward compatibility with old versions. The expected syntax follows:

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

We discuss package options according to the structure of this package. That structure repeats among the Boolean flags, package options, user interface macros, and internal macros. The goal is understanding through repetition.

Section 3 shows the hierarchy of these options and related macros. Default options are in **boldface** and need not be invoked by the user. Non-default options are in **dark red** and must be invoked explicitly. Many of these options work together with macros that do the same thing, but with finer control.

2.1 Name Grammar and Syntax

2.1.1 Show/Hide Affix Commas

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

These options do not affect the index. They permit different standards for name affixes. The default `nocomma` option gives, e.g., James Earl Carter Jr. The `comma` option produces James Earl Carter, Jr. Macros that allow finer control of commas and affixes are shown in Section 5.3.

2.1.2 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 5.4 for finer control. To capitalize names in the index, use caps as desired or alternate formatting (Section 11.3).

2.1.3 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 nonwestern, and vice versa.
`allrevcomma`	Print all names in "Surname, Forenames" order, meant for Western names.

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

2.2.1 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. **Caution:** using `noindex` and `\IndexInactive` prevents index tags until you call `\IndexActive`, as explained in Section 7.1. For indexing feature priority, see Section 3.

2.2.2 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, as might be needed with, e.g., `xindy`. See Section 7.6.

2.2.3 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 Control Sequences

Formatting, which, in its simplest form is typographic post-processing of a name, and in its complex forms can affect the syntactic form of a name, refers to the appearance of a name in the body text.

2.3.1 Choose Formatting System

`mainmatter`	**Start with "main-matter names" and formatting hooks** (Section 2.3.2).
`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`. See Section 9.1.

The `alwaysformat` option forces "first use" hooks globally in both naming systems. Its use is limited 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.

2.3.2 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.[11] 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 naming systems are used in `nameauth`, one for main-matter text (default) and one for front-matter text.[12] 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, 11, and 13 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 "Continental" formatting that one sees in German, French, and so on. Continental standards often format surnames only, both in the text and in the index. Section 11.3 introduces the topic and should be sufficient for most users, while Section 13 goes into greater detail for customization.

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

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.

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

[12] `\NamesFormat` was once the only formatting hook. The other macros developed from there. Regrettably, this package originated in a time when the present author was ignorant of several cultural and technical aspects of handling names. The "learn as you go" approach contributed to a fair bit of "cargo-cult" programming.

2.6 Version Compatibility

`oldAKA`	Force `\AKA*` to act like it did before version 3.0, instead of like `\FName`.
`oldreset`	Reset per-use name flags locally; let `\ForgetThis` and `\SubvertThis` pass through `\AKA` (pre-v3.3). Let `\SeeAlso` pass through `\IndexName` and other macros. Keep `\IndexName` and `\IndexRef` from resetting `\SkipIndex` (pre-version 3.5).
`oldpass`	When `\Justindex` is called, allow long 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.

Using these options may increase the chance of undocumented behavior.[13] They are included for the sake of approximate backward compatibility with older documents, illustrated by the following table:

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

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)

[13] Previously, the prefix macros and mechanisms for long and short names used Boolean flags locally. With continued use of this package, it became clear that such local scope could produce unexpected results. Those results, in turn, could mask problems caused by some flags not being reset by `\AKA`, `\AKA*`, and `\JustIndex`. The result was undocumented behavior.

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`	`\SubvertThis`
			`\ForceName`
			`\NoComma`

`\SkipIndex`	`\AccentCapThis`	**`allrevcomma`**
		`\RevCommaActive`
		`\RevCommaInactive`
			`\KeepName`
			`\ForceFN`
			`\ShowComma`

`\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 should precede only naming macros that produce output in the text.

  Due to this behavior, even though `\JustIndex` does not “override” the caps and reversing macros and options, nevertheless it simply prevents any other macros related to the display of a name from taking effect.

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

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

In the body text, not the index, the ⟨*Alternate*⟩ argument replaces either ⟨*FNN*⟩ or, if ⟨*FNN*⟩ is absent, ⟨*Affix*⟩. If both ⟨*FNN*⟩ and ⟨*Affix*⟩ are absent when ⟨*Alternate*⟩ is present, then the obsolete syntax is used (Section 12.2, not shown below).

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

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

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
Confucius	\Name {Confucius} or \Confucius
Confucius	\Name {Confucius} or \Confucius
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

With the quick interface, the best way to get alternate names follows patterns like "James Earl Carter Jr." \LCarter[James Earl] and "Miyazaki Sensei" \LMiyazaki[Sensei]. The alternate forename is not shown in subsequent short name instances e.g., "Carter" \Carter[James Earl]. Thus, one must use either long-name instances or forename instances to see the alternate names.

4.2 Forenames: \FName

\FName
\FName*

\FName and its synonym \FName* print personal names only in subsequent name uses. That means when a name control sequence 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 was implemented before the quick interface.

```
\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 nonwestern 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 names 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.
Confucius
Miyazaki,Hayao
Elizabeth,I

Albert	\FName[Albert]{Einstein} or \SEinstein
Jimmy	\FName[J.E.]{Carter, Jr.}[Jimmy] or \SCarter[Jimmy]
Confucius	\FName{Confucius} or \SConfucius
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
Good Queen Bess	\ForceFN\SEliz[Good Queen Bess]

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.

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)

4.3 Technical Details

4.3.1 Final Optional Arguments

Macros that take name arguments (see also Section 1.6.1) sometimes use a final optional argument. Using these arguments with the current syntax gives:

Printed Name	Macro and Arguments	Western Index Entry
Alias Family Affix	`\Name[Person]{Family, Affix}[Alias]`	Family, Person, Affix
Alias Family	`\Name[Person]{Family}[Alias]`	Family, Person
Family Alias	`\RevName%` `\Name[Person]{Family}[Alias]`	Family, Person

Printed Name	Macro and Arguments	Nonwestern Index Entry
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 offers two approaches that have their own pros and cons. Below `\Namei` works the same as nameauth. `\Nameii` takes the alternate route. If one builds this package with a LaTeX distro from before 2018, `\Nameii` will not appear, and `\Namei` will work like `\Nameii` in newer versions. It was for the sake of stability and consistency that nameauth used xargs and suffix in its early days. Now that all the issues with xparse have been addressed, the latter is the best choice going forward.

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

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

`\Namei:` something1 Family1

`\Nameii:` Person2 Family2 [something2]

- `\Namei` ignores spaces between the mandatory argument and the final optional argument.
  - Subsequent text in brackets will be interpreted as a name argument, even if unintended.
  - Yet we avoid errors from unintended spaces when using alternate names and the obsolete syntax.
- `\Nameii` treats spaces between the mandatory argument and the final optional argument as significant.
  - Subsequent text in brackets will be seen as a name argument only if no spaces occur between the mandatory argument and final optional argument.
  - Yet when using alternate names and the obsolete syntax, with unintended spaces we could "lose" names from the arguments and see unwanted results in the text.

In the next example we show the default approach of nameauth in practice using the basic name interface. `\ForgetName` (Section 9.3) lets us simulate first instances of names, even if they have appeared already.

We want: "Albert Einstein [then] said"; "Miyazaki Hayao [apparently] said".
Also: "Einstein [then] said"; "Miyazaki [apparently] said".
Macros: ``` ``\Name[Albert]{Einstein} [then] said''; ```
``` ``\Name{Miyazaki, Hayao} [apparently] said''. ```
We get: "then Einstein said"; "Miyazaki apparently said".
Repeat: "Einstein said"; "Miyazaki said".

- Add explicit spaces:
  "Albert Einstein [then] said"; "Miyazaki Hayao [apparently] said".
  ``` ``\Name[Albert]{Einstein}\ [then] said''; ```
  ``` ``\Name {Miyazaki, Hayao}\ [apparently] said''. ```
- Add curly braces:
  "Einstein [then] said"; "Miyazaki [apparently] said".
  ``` ``\Name[Albert]{Einstein}{} [then] said'' and ```
  ``` ``\Name{Miyazaki, Hayao}{} [apparently] said''. ```

Now we show how the nameauth default argument handling works with the quick naming interface:

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

We want: "Albert Einstein [then] said"; "Miyazaki Hayao [apparently] said".
Also: "Einstein [then] said"; "Miyazaki [apparently] said".
Macros: ``` ``\Einstein [then] said''; ```
``` ``\Miyazaki [apparently] said''. ```
We get: "then Einstein said"; "Miyazaki apparently said".
Repeat: "Einstein said"; "Miyazaki said".

- Add explicit spaces:
  "Albert Einstein [then] said"; "Miyazaki Hayao [apparently] said".
  ``` ``\Einstein\ [then] said'' and ```
  ``` ``\Miyazaki\ [apparently] said''. ```
- Add curly braces:
  "Einstein [then] said"; "Miyazaki [apparently] said".
  ``` ``\Einstein{} [then] said'' and ```
  ``` ``\Miyazaki{} [apparently] said''. ```

Section 13.3.2 shows how one can customize and change argument handling, addressing some of the same issues and caveats illustrated above.

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*
~~~

4.3.2 Name Argument Caveats

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:
$\boxed{\text{Martin Luther King Jr.}}$ $\boxed{\text{King}}$ $\boxed{\text{Martin Luther}}$
Spaces:
$\boxed{\text{Martin Luther King Jr.}}$ $\boxed{\text{King}}$ $\boxed{\text{Martin Luther}}$

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:

Output	**Macro**	**Name Pattern**
foo bar	`\Name{foo~bar}`	`foo~bar`
foo bar	`\Name{foo\ bar}`	`foo\bar`
foo bar	`\Name{foo\space bar}`	`foo\spacebar`

Name patterns, not name appearance, determine if names are the same (Section 6.1). There may be cases where one might leverage that point.

4.3.3 Full Stop Detection

Western names tend to use full stops in various cases, including the following:

- After the initial letter abbreviation of forenames.
- After affixes: "Jr". (junior), "Sr". (senior), "d. Ä." (*der Ältere*), "d. J." (*der Jüngere*) etc.
- In some contexts, after degrees like "M.D." (*Medicinae Doctor*), J.D. (*Juris Doctor*), Ph.D. (*Philosophiae Doctor*), etc.

If name contains a full stop at the end, followed by a full stop in the text. the nameauth macros try to prevent two adjacent full stops. All macros that take name arguments and print names in the text check if the printed name ends with a full stop. They also check the lookahead token for a full stop. If both cases are true, they gobble the lookahead token.

We show this behavior in the example below. `\ForgetName` (Section 9.3) lets us simulate the first instances of names. The table tells us whether or not the full stop on the lookahead token is gobbled, why that is the case, as well as the output of the name in the text and its correlating source.

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

Full Stop Gobbled	**Long Name Form, First Use**
Dr. Martin Luther King Jr.	`Dr. \Name[Martin Luther]{King, Jr.}.`
Full Stop Gobbled	**Force Long Name Form**
Dr. Martin Luther King Jr.	`Dr. \Name*[Martin Luther]{King, Jr.}.`
Dr. Martin Luther King Jr.	`Dr. \LMLK.`
Full Stop Gobbled	**Alternate FNN with Initials**
M.L.	`\FName[Martin Luther]{King, Jr.}[M.L.].`
M.L.	`\SMLK[M.L.].`
Full Stop Remains	**Affix Auto-Drops**
Dr. King.	`Dr. \Name[Martin Luther]{King, Jr.}.`
Full Stop Remains	**Force Affix to Drop**
Martin Luther King.	`\DropAffix\Name*[Martin Luther]{King, Jr.}.`
Martin Luther King.	`\DropAffix \LMLK.`

4.3.4 Grouping and Spaces

We disable indexing for the examples below so that we do not generate unwanted entries. Take care when using curly braces `{ }` in naming macro arguments. They will produce different names, as the formatting shows:

	Macro	**Result**	**Name Pattern**	**Type**
1	`\Name[one]{two}`	one two	`one!two`	Western
2	`\Name[{one}]{{t}w{o}}`	one two	`one!{t}w{o}`	Western
3	`\Name{{one}, two}`	one two	`{one},two`	nonwestern
4	`\Name{{one}, {two}}`	one two	`{one},{two}`	nonwestern
5	`\Name{{one, two}}`	one, two	`{one,two}`	abnormal

These names have different patterns and different index entries, even if they look similar in the text (Sections 6.1, 7.6). Similar issues pertain to grouping used with alternate formatting (Section 11.3). Consistency is key.

The grouping used in name 5 creates an abnormality . Here we illustrate that with the output of `\ShowNameInfo` (Section 6.3). Normally, the comma should delimit or segment the argument into two names:

`\Name{one, two}` (SNN: one) (Affix*: two)
`\Name{{one, two}}` (SNN: one, two)

Curly braces can change the lookahead token and defeat punctuation detection (as well as create a new, unique name). Using spaces between a name and a full stop can have a similar effect. The next table shows us several actions that allow or inhibit full stop detection. We se ethe results of each action, the action taken, and both output and source related to that action.

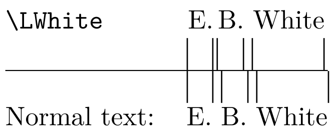
Full Stop Remains	**Not Contained in Group**
Dr. Martin Luther King Jr..	`Dr. {\Name*[Martin Luther]{King, Jr.}}.`
Dr. Martin Luther King Jr..	`Dr. {\LMLK}.`
Full Stop Gobbled	**Contained in Group**
Dr. Martin Luther King Jr.	`Dr. {\Name*[Martin Luther]{King, Jr.}.}`
Dr. Martin Luther King Jr.	`Dr. {\LMLK.}`
Full Stop Remains	**Grouped in Affix; New Name**
Dr. Martin Luther King Jr..	`Dr. \Name*[Martin Luther]{King, {Jr.}}.`
Full Stop Gobbled	**Not Grouped in Affix; New Name**
Dr. Martin Luther King Jr.	`Dr. \Name*[Martin Luther]{King, {Jr}.}.`
Full Stop Remains	**Intervening Space**
Dr. Martin Luther King Jr. .	`Dr. \Name*[Martin Luther]{King, Jr.} .`
Full Stop Gobbled	**After Shorthand**
Dr. Martin Luther King Jr.	`Dr. \LMLK .`
Full Stop Remains	**After Shorthand with Optional Argument**
Dr. M.L. King Jr. .	`Dr. \LMLK[M.L.] .`
Fixed; Space Removed	
Dr. M.L. King Jr.	`Dr. \LMLK[M.L.].`

4.3.5 Formatting Initials

This can be a thorny topic. 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.

Fighting with one's editor does little good. If a style guide requires spaces, try thin spaces. The quick interface simplifies the task. Below we use no formatting:

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



One might notice that we used `\PretagName` to sort this name by something other than its initials: "White, Elwyn". Sorting only by initials (and with embedded macros like a thin space) will produce unexpected entry order (Section 7.6.2).

Back to Table of Contents

5 Language Topics

Here we cover technical issues related to the use of names in various languages and cultures. Specifically, we show how they should be represented in English, but this material can apply to other language standards as well.

5.1 Caveats with Active Characters

Active characters affect name patterns and index sorting, depending on the LaTeX engine being used. We use `\PretagName` (Section 7.6; cf. 14.4) 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.[14]

   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 4.3.4). The index entry for this one is again different.

5.2 Hyphenation

In modern English, names can be hyphenated to reflect their cultural origins. With nameauth, one can handle such names by using default hyphenation, optional hyphens, or packages like babel and polyglossia.

Default Hyphenation

Name Pattern(s):
`John!Strietelmeier`

Here we use the default hyphenation. Using `\textls` from microtype we get the name to break badly. We use `\SkipIndex` to omit this version from the index.

```
\textls[20]{In English, some names come from other cultures.
Names like \SkipIndex\Name[John]{Strietelmeier}
can break badly in some cases.}
```

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

Discretionary Hyphens

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

We create a different name from that above. One must use the discretionary hyphens consistently in the macro arguments. We use `\Name[John]{Strie\-tel\-meier}` to omit this version from the index. We see that it breaks properly.

```
\textls[20]{In English, some names come from other cultures.
Names like \SkipIndex\Name[John]{Strie\-tel\-meier}
can break badly in some cases.}
```

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

[14]Regarding the margin note that shows name control sequences, with `pdflatex` and `latex`, in `ÃĘthelred,II` the glyphs `ÃĘ` correspond to `\IeC{\AE}`. Likewise in `BoÃńthius` the glyphs `Ãń` correspond to `\IeC{\"e}`.

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

Here we use an approach with macros in name arguments that can be applied to babel and polyglossia. This approach has issues that we discuss in Sections 5.7, 11.3, and elsewhere, showing how to prevent errors.

> When a macro occurs in a name argument appearing in both text and index, fix any concerns about macro expansion by using `\noexpand` before that macro.

Since the leading element of ⟨*SNN*⟩ is a macro, using `\CapThis` would halt LaTeX with errors unless we used alternate formatting (Section 11.3):

```
\newcommand\de[1]{\foreignlanguage{ngerman}{#1}}
%  or polyglossia: \newcommand\de[1]{\textgerman{#1}}

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

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

\textls[20]{In English, some names come from other cultures.
Names like \Striet\ or \Name[John]{\noexpand\de{Strietelmeier}}
break badly in some cases.}
```

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

5.3 Affixes Need Commas

> Regarding an ⟨*SNN, Affix*⟩ pair, the key to avoiding error is to apply macros to ⟨*SNN*⟩ and ⟨*Affix*⟩ as if they were independent arguments.

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 11.3.2 for more information when using macros in name arguments. Below we see the basics of commas in name arguments.

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

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` Western names with affixes must use ⟨*SNN,Affix*⟩, never the obsolete syntax, which is meant for nonwestern 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.

`\KeepName` In longer name forms, `\KeepName` turns spaces **between** all visible name elements into non-breaking spaces.

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

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.

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

`\ShowComma` `\NoComma` 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.

George S. Patton, Jr.	`\ShowComma\LPatton`
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).

5.4 Eastern Names and Family Names in All Caps

Proper Eastern names, included with ancient and medieval forms in the broader group of nonwestern names, are encoded using comma-delimited syntax (compare the obsolete syntax in Section 12.2). They have nonwestern index entries: ⟨*SNN*⟩ ⟨*Affix*⟩ (no comma between name elements). The current syntax permits alternate names; the obsolete does not.

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

Even in some academic texts, one sees nonwestern names encoded with Western forms.[15] The nameauth package seeks to remedy that issue, which has more to do

[15] 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). At this time, **NNDB** sorts Kim Jong Un under U, not K.

with outsourcing indexes to non-expert providers than with scholars. For example, the macro `\Name*{Sun, Yat-sen}` Sun Yat-sen ensures correct forms in the text and the index. We get cross-cultural sensitivity and scholarly accuracy.

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

Certain contexts call for one's entire family name to be capitalized. 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 in both the text and index (beyond doing it manually) see Sections 11.3 and 13.

`\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

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 have specific forms in the index, The reversing option and macros can adapt forms in the text to the index entries required in such cases.

`\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. In this manual, reversed Western forms are shown with a dagger (†).

`\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.

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

Output	**Macro**
Hideyo Noguchi	`\LNoguchi`
Doctor Noguchi	`\LNoguchi[Doctor]`
Bad result: Sensei Noguchi	`\LNoguchi[Sensei]`
Noguchi Hideyo†	`\RevName\LNoguchi\dag`
Bad result: Noguchi Doctor	`\RevName\LNoguchi[Doctor]`
Noguchi Sensei†	`\RevName\LNoguchi[Sensei]`
Miyazaki Hayao	`\LMiyazaki`
Miyazaki Sensei	`\LMiyazaki[Sensei]`
Bad result: Miyazaki Mr.	`\LMiyazaki[Mr.]`
Hayao Miyazaki	`\RevName\LMiyazaki`
Mr. Miyazaki	`\RevName\LMiyazaki[Mr.]`
Bad result: Sensei Miyazaki	`\RevName\LMiyazaki[Sensei]`

Reversing Western Names

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

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.

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 nonwestern index entry.

Reversing ancient names works the same as reversing Eastern names. The problem is that, in most cases, one produces a nonsense name.

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`

Oskar Hammerstein II	Hammerstein II, Oskar	change
Hideyo Noguchi	Noguchi, Hideyo	change
Æthelred II	Æthelred II	no change
Sun Yat-sen	Sun Yat-sen	no change

5.7 Particles in Names

Particles are small words attached to names, grouped either with forenames or surnames, that refer often to places of origin or noble houses. Some particles have been carried into modern names through various means.

5.7.1 Standard Rules

According to [Mulvany, 152–82], one may treat name particles in the following way:

- English use of *de*, *de la*, *d'*, *von*, *van*, and *ten* often keeps them with the surname. Capitalization may vary.
- *Le*, *La*, and *L'* always are capitalized unless preceded by *de*.
- Modern Romance languages keep particles with the surname.
- German and medieval Romance languages keep particles with forenames.
- Modern Welsh, Irish, and Scots names often merge particles with surnames: FitzRoy or Fitzroy; O'Leary; McDonald, MacLeish.

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

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

5.7.2 Non-Breaking Spaces

Despite the macros in Section 5.3, names with particles present their own challenges. We recommend inserting a tilde (active character for a non-breaking space) or `\nobreakspace` between some particles and names to prevent bad breaks, sorting them with `\PretagName` (Section 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 14.4). If one confuses these similar characters, spurious results can occur.

5.7.4 Capitalizing Particles

`\CapThis` In English and modern Romance languages, names like Hernando de Soto have their particles in the ⟨*SNN*⟩ argument: de Soto. When the particle appears at the beginning of a sentence, one must capitalize it, e.g.,:

> 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 nonstandard names and indexing.
- Sections 11.3 and 14.4 explain potential problems of `\CapThis`. That is a trade-off for using a natural language approach.
- `\CapName` overrides the ⟨*SNN*⟩ created by `\CapThis`. It is unlikely that the two would be used in the same context.

`\AccentCapThis` Before `nameauth` used automatic Unicode detection, `\AccentCapThis` placed before `\CapThis` handled Unicode characters. It remains for backward compatibility. Otherwise it is seldom used and not needed.

5.8 Medieval Names

Medieval names present some interesting difficulties, often based on the expected standards of the context in which they are used. Some publications use them like Western names while others do not. In the following preamble snippet we have:[16]

[16] Regarding the margin note that shows name control sequences, with `pdflatex` and `latex`, in `Thomas,Ãă~Kempis` the glyphs Ãă correspond to `\IeC{\'a}`.

```
\PretagName{Thomas, à~Kempis}{Thomas Akempis}  % medieval
\PretagName[Thomas]{à~Kempis}{Akempis, Thomas} % Western

% We do not index an alternate medieval form
\ExcludeName{Thomas, \`a~Kempis}

% 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}
```

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 is indexed as "Thomas à Kempis".
3. À Kempis `\CapThis\ForceFN\SKempMed` can start a sentence.
4. We use `\PretagName` (Section 7.6) to sort the name under `Thomas`, followed by `Akempis`, not `A kempis` (cf. 10 below).

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

5. Thomas à Kempis is a different name.
6. We used `\ExcludeName` (Section 7.3.2) before using the alternate form to keep it from generating an extra 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 via `\CapThis\KempW`.
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 again: `\JustIndex\KempMed`.

`\PretagName[Thomas]{à Kempis}{a Kempis, Thomas}` puts Thomas before `aardvark` in the index. We remove the extra spaces to get the proper sorting between `ajar` and `alkaline`: `\PretagName[Thomas]{à~Kempis}{Akempis, Thomas}`

Back to Table of Contents

6 Debugging

From this point onward, concepts that we introduced earlier will mesh together in complex ways, starting with indexing. Name patterns govern those interactions, illustrated by debugging macros. We explain what the debugging macros communicate, then we describe the macros.

Name Pattern(s):
`George!Washington`
`Miyazaki,Hayao`

Previously, we have seen simple name patterns displayed in the margin, which look like the examples to the left of this paragraph. Now we show full name patterns, which indicate the "system" or data set to which they belong. Name patterns are control sequences that usually expand to the empty string. They look like this:[17]

```
George!Washington!MN  % name, main matter
Miyazaki,Hayao!NF     % name, front matter
Jay!Rockefeller!PN    % index cross-reference
W.E.B.!Du~Bois!PRE    % index sort tag
Gregory,I!TAG         % index tag
Schuyler!Colfax!DB    % name tag
```

From this point onward, we show index entries in the body text instead of the margin due to their increasing complexity.

6.1 Name Pattern Overview

Below we show how macro arguments generate name patterns. The ⟨*Alternate*⟩ argument only affects patterns when using the obsolete syntax (Section 12.2). The patterns govern names in both the text and the index. The colon at the beginning of each pattern prevents any collisions with extant macros:

Macro Arguments	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*⟩	nonwestern
`{`⟨*SNN, Affix*⟩`}[`⟨*Alt*⟩`]`	:⟨*SNN*⟩,⟨*Affix*⟩	nonwestern
`{`⟨*SNN*⟩`}[`⟨*Alt*⟩`]`	:⟨*SNN*⟩,⟨*Alt*⟩	old syntax
`{`⟨*SNN*⟩`}`	:⟨*SNN*⟩	nonwestern

`\NameauthPattern`

Every time a macro that takes name arguments is called (Section 1.6.1), the internal name category logic will `\let` the current name pattern to `\NameauthPattern`, making it available to formatting hooks (cf. Section 11.2).

The name category logic is provided by `\@nameauth@Choice`, which determines the type of name (Western or nonwestern) by its arguments. Various `nameauth` macros append the appropriate "system" or name data set indicator to the pattern. We use this concept starting here, but mostly in Section 11.

Below are name patterns generated from name elements. We mark reversed Western names with a dagger (†) and names using the obsolete syntax with a double dagger (‡). We show that name patterns depend on name arguments, not on the

[17]To fit the information in the margin, we display them by taking the output of `\ShowPattern` and manually adding the system to correspond to the package internals.

appearance of a name. The obsolete nonwestern syntax and the current syntax share the same patterns.

Body Text	`\ShowPattern`	**Macro**
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 I Soter	`Demetrius,I`	`\Dem[I Soter]`
Demetrius I	`Demetrius,I`	`\LDem`
Aristotle	`Aristotle`	`\ForgetThis\Aris`
Aristotle	`Aristotle`	`\Aris`

Six suffixes are appended to these patterns to create six "systems" or data sets. The ⟨*pattern*⟩ element in the next table is the output of `\ShowPattern`. Within that ⟨*pattern*⟩ is a leading colon, name elements with spaces removed, and delimiters that separate those elements. Western names use an exclamation point and a comma. Nonwestern names only use a comma. For the use of ID, see `\ShowNameState` on page 47. For the way that exclusions use the cross-reference system as a special case for preventing index interaction, see Section 7.3.2.

Description	**Pattern**	**ID**	**Example**
Front-matter names	⟨*pattern*⟩`!NF`	front	`Adolf!Harnack!NF`
Main-matter names	⟨*pattern*⟩`!MN`	main	`Hideyo!Noguchi!MN`
Index cross-refs	⟨*pattern*⟩`!PN`	xref	`Yamamoto,Isoroku!PN`
Exclusions	⟨*pattern*⟩`!PN`	excl	(like xref, special value)
Index sort tags	⟨*pattern*⟩`!PRE`	pretag	`Henry,VIII!PRE`
Index data tags	⟨*pattern*⟩`!TAG`	idxtag	`Demetrius,I!TAG`
Name data tags	⟨*pattern*⟩`!DB`	namedb	`Aristotle!DB`

Next we see what macros write patterns to what systems. `\IndexName` does not generate a name pattern. This lets cross-references determine the index control logic.

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

6.2 Indexing: States

Six distinct “states” describe any name pattern with respect to the index. They are derived using `\ShowNameState`. Below we show these states and what they 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

Virtue never has been as respectable as money.

—Mark Twain, *The Innocents Abroad* (1869)

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

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

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

6.3 Debugging Macros

`\ShowPattern` We use `\ShowPattern` in Section 6.1 to illustrate name control patterns. It displays how the name arguments create name patterns that form name control sequences. One can debug pattern collisions and other issues with this macro:

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

Thus, `\texttt{\ShowPattern[Hernando]{de~Soto}}` will produce the output `Hernando!de~Soto`. As we have seen before, using inputenc/fontenc will cause names like `\texttt{\ShowPattern{Boëthius}}` to produce `BoÃńthius`.

`\ShowIdxPageref` `\ShowIdxPageref` displays a full index entry in the text as if it were only a page entry, not a cross-reference.

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

Index styles, `\PretagName`, and `\TagName` affect the output of `\ShowIdxPageref`. Active characters and macros appear as printed, not as in `idx` files. In a normal LaTeX document, for example, if we mention Hernando de Soto after setting up a sort tag (Section 7.6), we get the following:

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

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

We could not create index entries within the quote above in this `dtx` file because we changed `\IndexActual`. A `dtx` file uses a different default for `\IndexActual` (see Section 7.7). Here we get instead:

```
\PretagName[Hernando]{de~Soto}{Desoto, Hernando}
\TagName[Hernando]{de~Soto}{\string|hyperpage}
```

```
\ShowIdxPageref[Hernando]{de~Soto}
```
Desoto, Hernando=de Soto, Hernando|hyperpage

`\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⟩]
```

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.

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

Below we show several name forms. This macro shows detokenized name arguments to prevent potential errors and to disclose any macros in the name arguments, which are designated according to Section 1.3.1.

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

  `\ShowNameInfo[J.E.]{Carter, Jr.}[James Earl]`:
  (FNN: J.E.) (SNN: Carter) (Affix: Jr.) (Alt: James Earl)
- Miyazaki Sensei . . . . `\Name*{Miyazaki, Hayao}[Sensei]`

  `\ShowNameInfo{Miyazaki, Hayao}[Sensei]`:
  (SNN: Miyazaki) (Affix*: Hayao) (Alt: Sensei)
- Elizabeth I, "Gloriana" . . . . `\Name*{Elizabeth, I}[I, ``Gloriana'']`

  `\ShowNameInfo{Elizabeth, I}[I, ``Gloriana'']`:
  (SNN: Elizabeth) (Affix*: I) (Alt: I, "Gloriana")
- Henry VIII‡ . . . . `\Name*{Henry}[VIII]\ddag`

  `\ShowNameInfo{Henry}[VIII]` (obsolete syntax):
  (SNN: Henry) (Alt*: VIII)
- Confucius . . . . `\Name*{Confucius}`

  `\ShowNameInfo{Confucius}`:
  (SNN: Confucius)

For the following examples we activate alternate formatting (Section 11.3), which permits typesetting more complex name forms seen in European academic writing. This use of small caps may generate harmless font substitution warnings.

- John Strietelmeier
  `\ForgetThis\Name[John]{\noexpand\de{Strietelmeier}`

  `\ShowNameInfo[John]{\noexpand\de{Strietelmeier}}`:
  (FNN: John) (SNN: \de {Strietelmeier})
- Catherine de' MEDICI
  `\Name[Catherine \noexpand\AltCaps{d}e']{\noexpand\textSC{Medici}}`

  `\ShowNameInfo[Catherine \noexpand\AltCaps{d}e']`
  `{\noexpand\textSC{Medici}}`:
  (FNN: Catherine \AltCaps {d}e') (SNN: \textSC {Medici})

- Martin Luther
  `\Name[Martin]{\noexpand\textSC{Luther}}`

  `\ShowNameInfo[Martin]{\noexpand\textSC{Luther}}`:
  (FNN: Martin) (SNN: \textSC {Luther})
- Using `\noexpand` before macros in name arguments "fixates" them, ensuring consistency (important for name patterns and the index):
  - `\ShowNameInfo[John]{\noexpand\de{Strietelmeier}}`
    (FNN: John) (SNN: \de {Strietelmeier})
  - `\ShowNameInfo[John]{\de{Strietelmeier}}`
    (FNN: John) (SNN: \protect \foreignlanguage {ngerman}{Strietelmeier})
- If one uses an undefined macro in the arguments, but precedes it with `\noexpand`, `\ShowNameInfo` will not generate an error.
  - `\ShowNameInfo{\noexpand\SomeUndefinedMacro}`
    (SNN: \SomeUndefinedMacro)

When a macro occurs in a name argument appearing in both text and index, fix any concerns about macro expansion by using `\noexpand` before that macro.

`\ShowNameState` With this macro we can see all the systems with which a name pattern is associated and the current index "state" described in Section 6.2. Its syntax is:

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

`\ShowNameState` produces a line of text with the following information:

- Base name pattern, the base control pattern that nameauth sees. This would be the output of `\ShowPattern`.
- Type of name, referring to how the name in the argument was parsed.

w :	Western (regardless of whether it is reversed)
nw :	nonwestern (regardless of whether it is reversed) current syntax
nw,os :	nonwestern name, obsolete syntax (Section 12.2)

- Index state (Section 6.2) shows what index-related options are permissible and how the indexing macros will behave.

  states : 1, 2, 3, 4, 5, and 6
- All the control systems (Section 6.1) that the name pattern inhabits.

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

In the next example we show names in indexing states from one to six, and we illustrate all name types and many of the control systems. All extant names have index tags due to being used in a `dtx` file with the ltxdoc class and the hypdoc package (Section 7.7.3). Otherwise, names need not always have a control pattern in the `idxtag` system.

- Some yet unused name: . . . . `\ShowNameState{Not Seen, Yet}`
  Pattern: NotSeen,Yet Type: nw Index state: 1 Systems:
- New and extant names:
  - Rudolph Carnap . . . . `\Name*[Rudolph]{Carnap}`
    Pattern: Rudolph!Carnap Type: w Index state: 2 Systems: main idxtag
  - Henry VIII . . . . `\Name*{Henry, VIII}`
    Pattern: Henry,VIII Type: nw Index state: 2 Systems: main idxtag
  - Henry VIII‡ . . . . `\Name*{Henry}[VIII]\ddag`
    Pattern: Henry,VIII Type: nw,os Index state: 2 Systems: main idxtag
  - Thomas à Kempis . . . . `\Name*{Thomas, à~Kempis}`
    Pattern: Thomas,Ãă~Kempis Type: nw Index state: 2 Systems: main pretag idxtag
- Name used only in xref: . . . . `\IndexRef{Sun King}{Louis XIV}`
  Pattern: SunKing Type: nw Index state: 3 Systems: xref
- Name used after xref exists:
  Sun King . . . . `\Name*{Sun King}`
  Pattern: SunKing Type: nw Index state: 4 Systems: main xref
- Unused name excluded from index: . . . . . `\ExcludeName{Santa, Claus}`
  Pattern: Santa,Claus Type: nw Index state: 5 Systems: excl
- Previously excluded name gets used:
  Thomas à Kempis . . . . `\Name*{Thomas,\`a~Kempis}`
  Pattern: Thomas,\`a~Kempis Type: nw Index state: 6 Systems: main excl

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)

7 Indexing Macros

Indexing pulls together all the concepts from the previous sections, yet applies them to the index. In this section we enable (and trigger) verbose index warnings.

7.1 General Indexing Control

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.

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

\global

\IndexActive and \IndexInactive can be used as a pair or singly within a group. They have top priority (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 redefined this hook for use with multiple indexes. Below we use the index package to do this.

Test indexing rules:
Yamaha Torakusu
create index entry:
\IndexName
{Nippon Gakki}

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

\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. Throughout Section 7 we enable verbose warnings.

`\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:

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, using also `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. This manual uses the tag § for `\Name{foo\Name{bar}}`, not shown in the example for the sake of clarity.

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 nonwestern syntax (Section 12.2). Indexing follows [Mulvany, 152–82]. The points below deal with macros explained in Section 7.4.

- `\IndexName` obeys both `\IndexInactive` and `\IndexActive`, which are used to deactivate and activate indexing.
- `\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⟩}[⟨Alternate⟩]{⟨Target⟩}
\SeeAlso\IndexRef[⟨FNN⟩]{⟨SNN, Affix⟩}[⟨Alternate⟩]{⟨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.

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 control pattern, unless one uses the oldsee option. A fuller explanation of this non-trivial fact lies in Section 6.2, illustrated by a state diagram. A page entry usually correlates with a name control pattern, but that may not always be the case. We check for a control sequence (name pattern); usually, a page entry also exists.
- \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.

\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

Here we show how the index control logic pertinent to cross-references can be extended to establish fine-grained control that can exclude or include names.

`\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 prevent and permit all 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`

```
\ExcludeName{Mr. Baseball}
In this example we cannot get page entries for
\Name{Mr. Baseball}, the nickname of
\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 control patterns, allowing one to create a cross-reference. Section 6.2 explains this behavior as a set of states. Below we run some tests (cf. Section 9.5) and produce a few warnings because verbose warnings are active here.

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*⟩.

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

Cross-references get more protection than exclusions:

```
\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.

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`.

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

Option	Output	Source
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.

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

- The following three lines are equivalent:
  - `\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 in any given state. Here we do not attempt to concatenate page ranges. Above, we put a series of names and cross-references in margin notes, Here is the result:

Page	Macro	Index Result

49	`\Name{Yamaha, Torakusu}`
	`\IndexName{Nippon Gakki}`
		Yamaha Torakusu, 49
		Nippon Gakki, 49

50	`\Name{Yamaha, Torakusu}`	Yamaha Torakusu, 49, 50 Nippon Gakki, 49
51	`\SeeAlso\IndexRef{Yamaha, Torakusu}{Nippon Gakki}`	Yamaha Torakusu, 49, 50 *see also* Nippon Gakki Nippon Gakki, 49
52	`\Name{Yamaha, Torakusu}`	Yamaha Torakusu, 49, 50 *see also* Nippon Gakki Nippon Gakki, 49
54	`\Name{Nippon Gakki}`	Yamaha Torakusu, 49, 50 *see also* Nippon Gakki Nippon Gakki, 49, 54
55	`\IndexRef{Nippon Gakki}{Yamaha Corp.}`	Yamaha Torakusu, 49, 50 *see also* Nippon Gakki Nippon Gakki, 49, 54

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

1. On page 49 there are no name control patterns (Section 6.1) for either a name or a cross-reference. Then the name Yamaha Torakusu comes into being, along with its control pattern. A pair of index page entries are generated, one before and one after the name.

   Also on 49 an index page entry is created for Nippon Gakki, whose control pattern does not yet exist.

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

2. On page 50 the name Yamaha appears again. Since a control pattern exists, a short form is printed and two index page entries are generated.

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

3. On page 51 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 him.
4. On page 52 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): `NipponGakki!MN`

5. On page 54 we print the name Nippon Gakki, bringing its name control sequence 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): `YamahaCorp.!MN`

6. On page 55 we attempt to create a *see* cross-reference from Nippon Gakki to the name Yamaha Corp. (Notice that the extra full stop in the text gets gobbled.) That fails because, unlike a *see also* reference, a *see* reference cannot be created when a name control pattern already exists.

This manual's index entries for Nippon Gakki and Yamaha Corp. will also include this page in addition to those shown above.

7.6 Sorting Names in the Index

When using `makeindex`, all names with characters outside the ASCII range `[A-Za-z]` need to be sorted. All names with macros in their arguments need to be sorted. All names with particles need to be sorted. We do that with sort tags.

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:

Name Pattern(s):
`ÃĘgidius!PRE`

```
\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}}.
```

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 6.1). 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}
```

Every reference to Æthelred II and W.E.B. Du Bois is automatically tagged and sorted.[18] One should "pretag" all names that contain active characters or macros. That can differ when using `xindy` and Unicode-based LaTeX.

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

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, 11.3). We create a *see* reference before using this alias, for which no page entries will be generated. The name patterns are:

```
\textit{Doctorangelicus}!PRE
\textit{Doctorangelicus}!PN
Thomas,Aquinas!MN
\textit{Doctorangelicus}!MN
```

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

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

To give *Doctor angelicus* page entries and a *see also* reference, we omit line 2 above, wait until the end of the body text, after both names are fully indexed, 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`.

Collating Sequences

German ä ö ü ß map to English ae oe ue ss. Yet Norwegian æ ø å follow z in that order. Check a style guide regarding collating sequences, spaces, and sorting. This is where using `xindy` can be very helpful. See also Section 5.7.

Alternate Formatting

Additional examples starting with Section 11.3.5 deal with index sorting as it relates to alternate formatting, “Continental” practice, and macros in name arguments. Please ensure that such macros expand to the form desired in the index in order to get proper sorting with `\PretagName`.

Igitur qui desiderat pacem, praeparet bellum.

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

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

7.6.2 Sorting Initials

In order to sort the index consistently and properly, all names should be sorted by their longer name forms, not their 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 of handling alternate forms of surnames see Sections 5.8, 10.1, all subsections of Section 11, and 13.

7.7 Index Tags

Index data tags are information added automatically to index page entries for names used with nameauth macros.

7.7.1 General Approach

`\TagName` `\UntagName` This macro creates a tag that persists until one changes it with `\TagName` or destroys it with `\UntagName`. Tags can include life dates, regnal dates, and other information. Both `\TagName` and `\UntagName` have **global scope** and handle arguments 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:

`\index{`	`\PretagName`
	`Aethelred 2@`
		Naming macros
		`\IndexName`
		`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 entries created in the next example:

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

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

Pope Gregory I was known as Gregory the Great.

See Section 11.2.2 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.

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, while the name decision macros can set up specific logic for each name:

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

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

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

This refers to the classical composer: Engelbert Humperdinck.

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

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.[19] When `\`⟨*macro*⟩`#1{#1}` exists and ⟨*name args*⟩ are the arguments, one can use the form:

`\TagName`⟨*name args*⟩`{|`⟨*macro*⟩`}`

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, we use:

[19] This was implemented in v.3.3, based on the answer of Heiko Oberdiek to **this question**.

```
\TagName⟨name args⟩{\noexpand\string|hyperpage}
```
or
```
\index{⟨entry argument⟩\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 avoid both errors with the redefinition of the vertical bar and any possible confusion when using `\string`.

```
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⟩}`

Manual Breaks and Entries

The microtype package and its `Spacing` environment may be the best solution to fix index entries and sub-entries that break badly across columns or pages. Yet we could add manual breaks after editing is complete.

We must create a helper macro that takes an argument and adds a break after that argument. That is how macros like `\textbf` use implied page entries in the index, e.g.: `\index{Doe, John|textbf}`.

Below we use `\newpage` to jump to a new column. See also the multicol and idxlayout packages, as well as classes like memoir. On line 1 we define the `\Endbreak` macro that will break the column after the end of an index entry.

```
\newcommand*{\EndBreak}[1]{#1\newpage}
```

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

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

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

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

Page	Macro	Index Result
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**. A "quick and dirty" version corresponding to the `\EndBreak` macro follows:

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

Nevertheless, it is clear from some discussions that such macros can be rather fiddly at times and can produce unexpected results. One is advised to take caution when breaking index entries midway or otherwise modifying them in this manner.

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 must use both `\PretagName` and `\TagName`. For example, to sort a name under ⟨*category*⟩, use something like:

```
\PretagName[⟨FNN⟩]{⟨SNN⟩}{⟨category 1⟩!⟨SNN⟩, ⟨FNN⟩}
\PretagName{⟨SNN, Affix⟩}{⟨category 1⟩!⟨SNN⟩ ⟨Affix⟩}
```

Whenever one wants to generate a sub-entry for a name, one can use `\TagName` to create that sub-entry when needed via something like:

```
\TagName[⟨FNN⟩]{⟨SNN⟩}{!⟨name sub-entry⟩}
\TagName{⟨SNN, Affix⟩}{!⟨name sub-entry⟩}
```

One is not restricted to Western or nonwestern name arguments; the boxes above are meant just to show the basic tag formats.

One could use \TagName to change to another sub-entry or the default tag as needed, or use \UntagName to remove the tag. If a sub-entry contains a cross-reference via \IndexRef, it is necessary to follow that with \IncludeName* to permit any further page entries in other sub-entries or the main name entry. Below we demonstrate how one would implement sub-entries using index tags in a normal LaTeX document. We cannot show categories when using gind.ist.

Categories higher than names are handled by \PretagName. Categories lower in the hierarchy are handled by \TagName. It is best practice to have three or less levels of categories in an index. Two levels are more common. Using such levels also depends on the index style and formatting files.

```
\documentclass{article}
\input{compat.tex} % Included with nameauth; example file aids
% compatibility across different LaTeX versions and engines.
\usepackage[a6paper,landscape,left=1cm,right=1cm]{geometry}
\usepackage{makeidx}
\usepackage{nameauth}

\makeindex

\newcommand*{\EndBreak}[1]{#1\newpage}

%  Sort these names under: US Presidents.
\PretagName[George]{Washington}{US Presidents!Washington, George}
\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}

% This is not a sub-category.
\TagName[George S.]{Patton, Jr.}{, general}

\begin{nameauth}
  \< Wash & George & Washington & >
  \< Aris & & Aristotle & >
  \< Plato & & Plato & >
  \< Aeth & & \AE thelred, II & >
  \< Sun & & Sun, Yat-sen & >
  \< Linc & Abraham & Lincoln & >
  \< MLK & Martin Luther & King, Jr. & >
  \< Soto & Hernando & de Soto & >
  \< Goethe & J.W. von & Goethe & >
  \< 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}

Among philosophers we consider \Plato\ and \Aris.

\section{Historical Figures}

\TagName{Sun, Yat-sen}{|EndBreak}
We ponder about \Aeth, then \Aeth.
We speak of \Sun, then \Sun.
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, \LPatton, and \LIke\ were high-ranking generals.\\

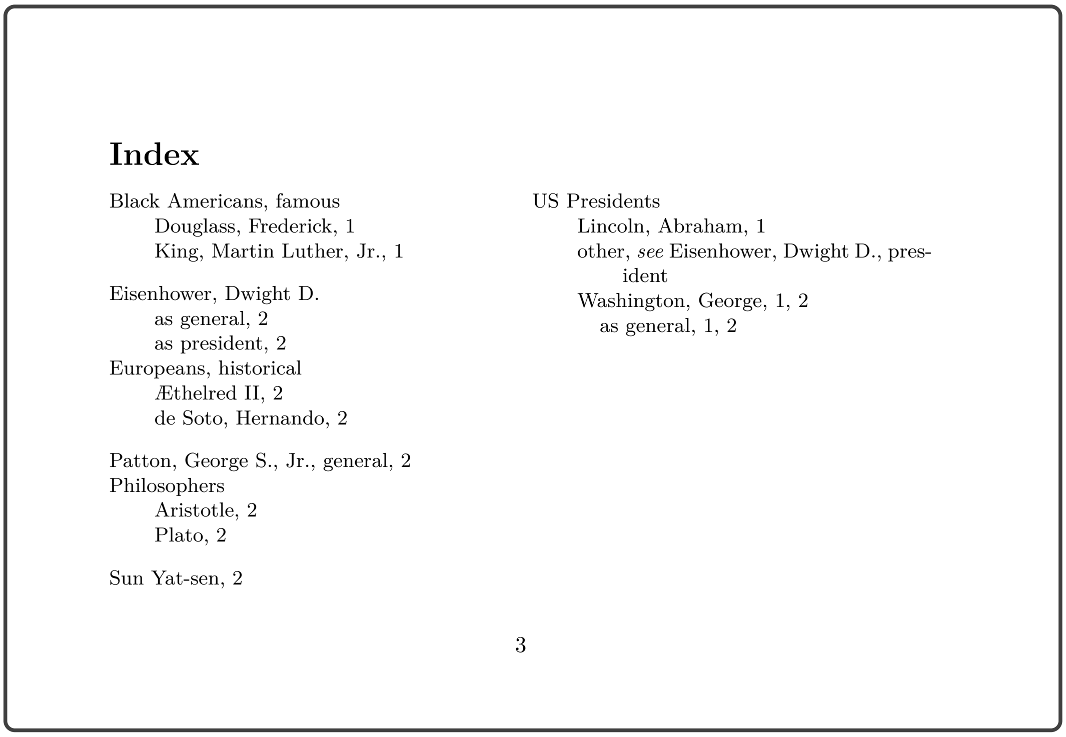
\TagName[Dwight D.]{Eisenhower}{!as president}
\UntagName[George]{Washington}
\Wash\ and \Ike\ also were US presidents.

% Instead of pre-tagging Ike we do the following:
\index{US Presidents!other|see{Eisenhower, Dwight D., president}}

\small
\printindex
\end{document}
```

Below we show only the index from this example. A few notable features include:

- Patton uses a regular index tag instead of a subcategory. That is a way to ‘cheat” and get more groupings of entries than allowed categories.
- Eisenhower illustrates multiple subcategories for a name. A cross-reference from a sub-entry of US presidents points to him.
- Washington illustrates having both a super-category and a subcategory.
- Eisenhower, Patton, and Sun are not sorted under categories. The other names are sorted under categories.



Index

Black Americans, famous
- Douglass, Frederick, 1
- King, Martin Luther, Jr., 1

Eisenhower, Dwight D.
- as general, 2
- as president, 2

Europeans, historical
- Æthelred II, 2
- de Soto, Hernando, 2

Patton, George S., Jr., general, 2

Philosophers
- Aristotle, 2
- Plato, 2

Sun Yat-sen, 2

US Presidents
- Lincoln, Abraham, 1
- other, *see* Eisenhower, Dwight D., president
- Washington, George, 1, 2
  - as general, 1, 2

3

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. . . . 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)

8 Name Tags

\NameAddInfo

All valid names in nameauth can have name tags; no restrictions exist. Unlike index tags, name tags are not printed automatically with every name managed by nameauth. Sections 9.5 and 11.2 have more examples. The macro is `\long`, allowing for some complexity in the ⟨*tag*⟩ argument:

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

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

For example, `\NameAddInfo[George]{Washington}{(1732--99)}` makes a name tag but does not print whenever Washington `\Wash` is used. The name tag needs to be displayed using the helper macro below.

\NameQueryInfo

To print a name tag for a given name, we use `\NameQueryInfo`, a helper macro that prints a name tag in the name info data set:

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

`\NameQueryInfo[George]{Washington}` prints the name tag for George Washington: (1732–99). In Section 11.2 and thereafter we show how one can automate the retrieval of name tags using formatting hooks, Yet therein also lies a caution:

- When using the recommended template for a formatting hook (Section 11.2.1), one can use `\noexpand` in arguments for `\NameAddInfo`, `\NameQueryInfo`, and `\NameClearInfo` without restriction.
- When using designs based on older templates, using `\noexpand` may cause `\NameQueryInfo` to print nothing if one uses the basic interface. Using the quick interface will work as expected.

One can insert a space at the start of a name tag; use signs like asterisks, daggers, and the like; and even create footnotes, such as footnote 20 (below):

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

```
\NameAddInfo[Ulysses S.]{Grant}
  {eighteenth US president (1869--1877)}
\NameAddInfo[Schuyler]{Colfax}
  {\footnote{\Name[Schuyler]{Colfax} was the seventeenth US
  vice-president during the first term (1869--73) of
  \Name*[Ulysses S.]{Grant}, \NameQueryInfo[Ulysses S.]{Grant}.}}

Remember \Name[Schuyler]{Colfax}?\NameQueryInfo[Schuyler]{Colfax}
Derived from the same origin as ``scholar'', this name can occur
as ``Skylar'' for girls and ``Skyler'' for boys.
```

Remember Schuyler Colfax?[20] Derived from the same origin as "scholar", this name can occur as "Skylar" for girls and "Skyler" for boys.

[20] Colfax was the seventeenth US vice-president during the first term (1869–73) of Ulysses S. Grant, eighteenth US president (1869–1877).

The previous example cannot be used in a formatting hook (Section 11.2). Within the formatting hooks, one is in a "locked path" that prevents calling a naming macro. This prevents a stack overflow. `\Name*[Ulysses S.]{Grant}` would not print inside a formatting hook, although `\NameQueryInfo[Ulysses S.]{Grant}` would print.

Name tags can call each other. To protect against a stack overflow, use Boolean flags and conditional statements. Below, `\NameQueryInfo` calls a tag that sets a Boolean flag true, which causes the **other** tag to stop any recursion and exit.

```
\newif\ifA
\newif\ifB
\NameAddInfo{A}
  {\Atrue A \ifB Stop \else \NameQueryInfo{B} \fi \Afalse}
\NameAddInfo{B}
  {\Btrue B \ifA Stop \else \NameQueryInfo{A} \fi \Bfalse}
\begin{itemize}
  \item \NameQueryInfo{A}
  \item \NameQueryInfo{B}
\end{itemize}
```

- A B Stop
- B A Stop

`\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`

We now revisit George Washington and his associated name tag.

```
The name tag is: \fbox{\NameQueryInfo[George]{Washington}}\quad
Clearing data.\NameClearInfo[George]{Washington}\quad
The name tag is empty: \fbox{\NameQueryInfo[George]{Washington}}
```

The name tag is: $\boxed{\text{(1732–99)}}$ Clearing data. The name tag is empty: □

Back to Table of Contents

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)

9 Basic Formatting and Name Decisions

9.1 Basic Formatting

This section offers a brief overview; Section 11 goes into great detail. Even when using the default options for nameauth wherein no formatting occurs, we can observe syntactic changes to names:

Syntactic Changes; No Formatting/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 syntactic changes. In its basic form, formatting is typographic post-processing. In its advanced form, formatting locally affects also the syntactic forms of names while leaving index entries undisturbed.

Many books are structured with front matter that includes a table of contents, foreword, introductory material or survey 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`). These formatting systems are linked to the existence of a name control pattern.

- First instance of a name
  - No name control sequence 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 control sequence is created.
- Subsequent instance of a name
  - A name control sequence already exists.
  - A name is printed using a shorter form (default).
  - The "subsequent-use" formatting hook is used (default).

The parser and related macros create name forms and formats only in the text. Macros in name arguments affect both text and index (Section 11.3).

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

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

`\NamesFormat` `\MainNameHook` `\FrontNamesFormat` `\FrontNameHook`

The main-matter system uses `\NamesFormat` to post-process first occurrences of names and `\MainNameHook` for subsequent uses. The front-matter system uses `\FrontNamesFormat` for first uses and `\FrontNameHook` for subsequent uses. The `alwaysformat` option causes only `\NamesFormat` and `\FrontNamesFormat` to be used. Since the formatting hooks always are defined when using nameauth, one must use `\renewcommand` when changing their definitions.[21] Section 6.1 shows how name control sequences are keyed either to the main-matter system or to the front-matter system. The two formatting systems are distinct, useful for separate document elements. We color-code them and "forget" any 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:

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

`\ForceName`

We show examples of `\ForceName` in Sections 9.3, 11.2, and 12.1. Use this prefix macro to force "first use" formatting for the next `\Name`, etc., but without deleting any name control sequences. Thus:

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

Carnap `\Name[Rudolph]{Carnap}`

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

`alwaysformat`

Below we simulate `alwaysformat` via package internals. Only the "first use" formatting hooks are used:

- 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`

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

```
\bgroup⟨Hook⟩{#1}\egroup.
```

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 that takes one argument can use, change, or discard it and invoke `\NameParser` (Section 11.5). Due to package design using local scope, both the following achieve exactly the same effect:

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

9.2 Application: Footnotes

The independent systems of names work with footnotes. Names in the body text, such as Adolf Harnack (later ennobled to Adolf von Harnack), normally affect name forms in the footnotes.[22] In footnote 22 `\MainNameHook` is called instead of `\NamesFormat` because Harnack already had occurred above. We can use the front-matter system to change that:

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

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

When we create another footnote, we see very different results.[23] Footnote 23 shows the first use of a name because it is the first use in the front-matter system. One can synchronize the two systems with `\ForgetThis` and `\SubvertThis` (Section 9.3). Below we revert footnotes with:

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

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)

[22] We have Harnack from `\Harnack` instead of Adolf Harnack.
[23] We have Adolf Harnack from `\Harnack`, then Harnack.

9.3 Making Name Decisions

By default, the macros below produce global effects. They change both the `!MN` and `!NF` data sets (Section 6.1). Those changes implicitly affect syntactic name forms, name formatting, index protection with respect to creating cross-references (Section 7.3), and the name testing macros (Section 9.5).

`\ForgetName` This macro takes the same arguments as `\Name`, but it prints no output. It "forgets" a name, forcing a "pre-first use" state. The next instance of that name will display as if the name did not yet exist:

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

`\ForgetName` "unprotects" names like `\IncludeName*` "unprotects" xrefs. This allows one to make both *see* and *see also* cross-references to a name, even if that name already has index page entries.

`\ForgetThis` This prefix macro causes the next instance of a naming macro or shorthand to "forget" a name before printing it. After knowing Einstein, we forget him and again have a first instance via `\ForgetThis\Einstein`: Albert Einstein. `\ForgetThis` no longer affects the index unless one uses the `oldreset` option.

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

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

`\SubvertThis` This prefix macro causes the next instance of a naming macro or shorthand to "subvert" a name before printing it. As indicated in Section 3, `\ForgetThis` has a higher priority than `\SubvertThis` and negates it. `\SubvertThis` no longer affects the index unless one uses the `oldreset` option.

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

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

To see how these two macros work, in the following example we define a macro that reports whether or not `\Name[Charlie]{Chaplin}` exists. This macro gives four possible results: the name exists in the main matter, it exists in the front matter, it exists in both systems, or it does not exist (see Section 9.5):

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

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

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

Name Pattern(s):
`Charlie!Chaplin!MN`

- Create a name in the "main matter":

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

Name Pattern(s):
`Charlie!Chaplin!NF`

- Switch to the "front-matter" system and create a name. If one is within a group or local scope, one may have to add `\global` to `\NamesInactive`:

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

- use `\LocalNames` to make both `\ForgetName` and `\SubvertName` work with only the current system.

  `\LocalNames`

- Had we not used `\global` above, we would have implicitly returned to the main-matter system due to scoping and environments like `quote` and `itemize`. We "forget" only the name in the front-matter system.

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

- Next "subvert" the front-matter name to "remember" it again. Then switch to main matter:

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

- Now the current system is main matter. We forget the main-matter name only, leaving the front-matter name intact:

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

- Use `\GlobalNames` to make `\ForgetName` and `\SubvertName` work with both systems again:

  `\GlobalNames`

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

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

9.4 Formatting and Decisions

We pull together information on name forms and formatting, focusing on what happens in one naming system only, since the two systems are independent.

First Instance: Betsey Bailey . . . . \Bailey
Betsey Bailey . . . . \LBailey
Betsey Bailey . . . . \SBailey
Name control pattern created with text output. Index state 2, 4, or 6 (Section 6.2). Name form: long. First-use hooks.

Later Instance: Bailey . . . . \Bailey
Betsey Bailey . . . . \LBailey
Betsey . . . . \SBailey
No change to name pattern. Index state 2, 4, or 6. Name form: short. Subsequent-use hooks.

Forgotten: (no output) . . . . \ForgetName[Betsey]{Bailey}
Name pattern deleted. Index state 1, 3, or 5 (Section 6.2). next instance usually will be a first use, e.g.: Betsey Bailey.

Subverted: (no output) . . . . \SubvertName[Betsey]{Bailey}
Name pattern created. Index state 2, 4, or 6. next instance usually will be a later use, e.g.: Bailey, Betsey Bailey, Betsey.

\ForgetThis: Betsey Bailey . . . . \ForgetThis\Bailey
Betsey Bailey . . . . \ForgetThis\LBailey
Betsey Bailey . . . . \ForgetThis\SBailey
Name pattern deleted, then created again. Index state 2, 4, or 6. Name form and format: same as first use above. next instance usually will be a later use, e.g.: Bailey, Betsey Bailey, Betsey.

\SubvertThis: Bailey . . . . \SubvertThis\Bailey
Betsey Bailey . . . . \SubvertThis\LBailey
Betsey . . . . \SubvertThis\SBailey
Name pattern created. Index state 2, 4, or 6. Name form and format: same as later use above. next instance usually will be a later use, e.g.: Bailey, Betsey Bailey, Betsey.

Format as First: Bailey . . . . \ForceName\Bailey
Betsey Bailey . . . . \ForceName\LBailey
Betsey . . . . \ForceName\SBailey
No change to name pattern. Index state 2, 4, or 6. Name form: short or long. Name format: First-use hooks.

In real life, unlike in Shakespeare, the sweetness of the rose depends upon the name it bears. Things are not only what they are. They are, in very important respects, what they seem to be.

—Hubert H. Humphrey, speech (26 March 1966)

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 any changes therein. If this is not desired, use the `globaltest` option or `\GlobalNameTest`. `\LocalNameTest` will re-enable the default. These commands affect assignment statements in test paths. By default, one must explicitly use `\global` when desired. See also the examples below.

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

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

For example we have not encountered `\Name[Bob]{Hope}` yet. Using `\IndexName` does not affect the tests in this section. We could do the following test that will reflect whether or not the name is present 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}`, a name that has occurred, and we also show the difference between local and global test paths. We see that the default keeps local any assignments made in the test paths:

```
\GlobalNameTest
\def\msg{We are unsure about \LEliz}

\IfMainName{Elizabeth,I}
  {\def\msg{We really do know of \LEliz}}
  {\def\msg{We do not know of \LEliz}}

\parbox{0.4\textwidth}{\msg} (\cmd{\GlobalNameTest}).

\LocalNameTest
\def\msg{We are unsure about \LEliz}

\IfMainName{Elizabeth,I}
  {\def\msg{We really do know of \LEliz}}
  {\def\msg{We do not know of \LEliz}}

\parbox{0.4\textwidth}{\msg} (\cmd{\LocalNameTest}).
```

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

We are unsure about 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⟩}[⟨Alternate⟩]{⟨yes⟩}{⟨no⟩}
```

This macro works just like `\IfMainName`, except using the "front matter" name control sequences as the test subject. These testing macros prove their worth especially through combination. For example, 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⟩}[⟨Alternate⟩]{⟨y⟩}{⟨n⟩}{⟨excl⟩}
```

This macro also works like `\IfMainName`, except that it has an additional ⟨*excl*⟩ branch in order to detect the activity of `\ExcludeName` (Section 7.1). Cross-references are governed by name control sequences ending in `!PN` (Section 6.1).

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

Based on the known facts above, below we offer some examples:

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

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

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

We can combine all these macros to create a complete, unified test:

```
\IfAKA[FNN]{SNN, Affix}[Alternate]
{%
  % yes; it is an xref
}
{%
  % no, it is a name
  \IfFrontName[FNN]{SNN, Affix}[Alternate]
  {%
    % yes, it is in the front matter
    \IfMainName[FNN]{SNN, Affix}[Alternate]
    {%
      % it is in both front and main matter
    }
    {%
      % it is only in the front matter
    }%
  }
  {%
    % no, it is not in the front matter
    \IfMainName[FNN]{SNN, Affix}[Alternate]
    {%
      % it is only in the main matter
    }
    {%
      % it does not exist
    }%
  }%
}
{%
  % no; it is excluded
}
```

9.5.2 Applications: Game Books, Histories, Etc.

In any text where encountering certain names can change variables, character statistics, personal information, the macros described above can be used to key various pieces of information to the presence or the absence of a name. For example, in a series of independent document sections, one can craft notes like the one below to sketch out character development:

```
\ifMainName[Ferris]{Bueller}
  {\Name[Cameron]{Frye} is gloomy and introspective.}
  {\Name[Cameron]{Frye} is developing positive traits.}
```

In some instances, one might test for the presence of a name to determine whether or not to use a particular version of that name:

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

In Sections 11.2.2 and 11.5 we explore ways that one could automate something like what we have above. Section 11.5 applies better to Western names.

In addition to using the name decision testing macros by themselves, one can use them with name tags to ensure that the information associated with a given name is not anachronistic.

For example, we know that certain people are associated with chronological events. We associate those people and events to the information presented in a name tag via name testing macros:

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}
      {renamed himself \Name{Paul}, 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{Saul, of Tarsus}.
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{Saul, of Tarsus}.
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{Saul, of Tarsus}.
Under the name \Name{Paul} he wrote his letters.
```

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 renamed himself Paul, in honor of his patron. Under the name Paul he wrote his letters.

Caveats

Using these tests inside other macros or passing control sequences to them may create false results (see *The TEXbook*, 212–15). This was especially the case before 2018 with names using diacritical marks and other letters outside the basic Latin characters. That is why `nameauth` uses token registers to save name arguments.

We have stressed and will continue to stress using `\noexpand` in macros passed as name arguments to stabilize what happens. See also Section 14.4 regarding how one might engage possible Unicode issues in certain LATEX engines.

In addition to these points, using the `trace` package, `\show`, or `\meaning` can help one to mitigate problems.

9.5.3 Beamer Example

Below we keep names consistent with beamer overlays using some of the macros explained in this section. Otherwise, name forms will change automatically as one advances the slides. We do not use indexing in this example.

```
\documentclass{beamer}
\input{compat.tex} % Included with nameauth; example file aids
% compatibility across different LaTeX versions and engines.
\usepackage[noindex]{nameauth}

\mode<presentation>
\beamerdefaultoverlayspecification{<+->}

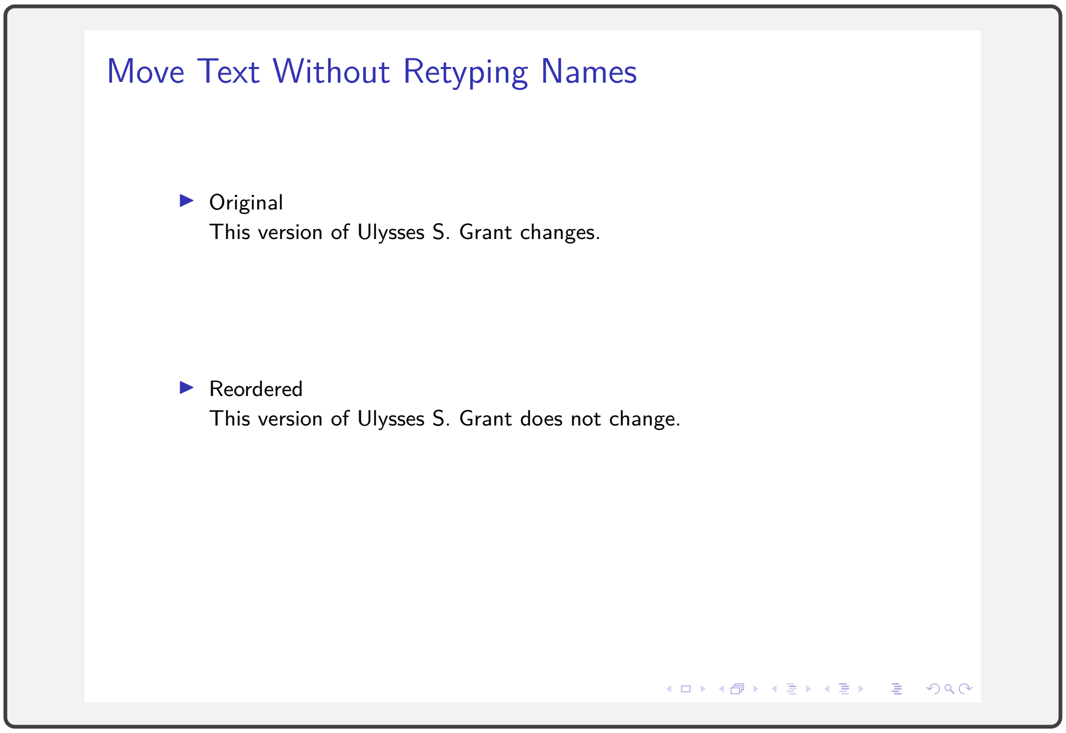
\begin{document}

\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 control sequence 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.

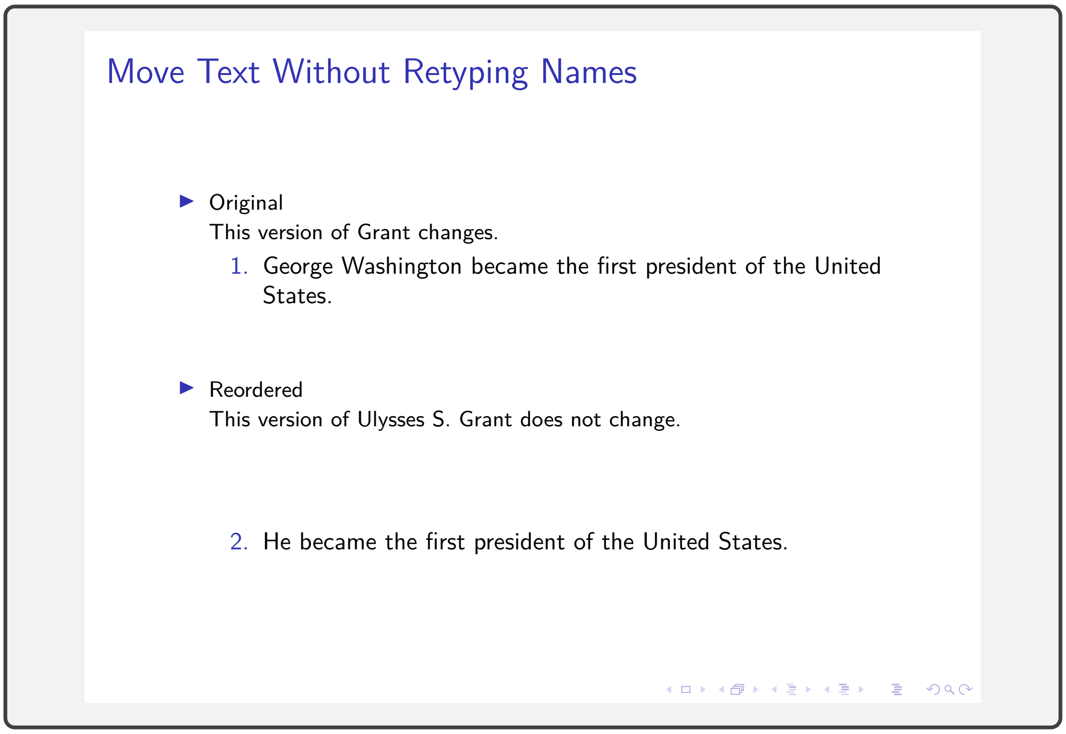


Move Text Without Retyping Names

- Original
  This version of Ulysses S. Grant changes.

- Reordered
  This version of Ulysses S. Grant does not change.

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.



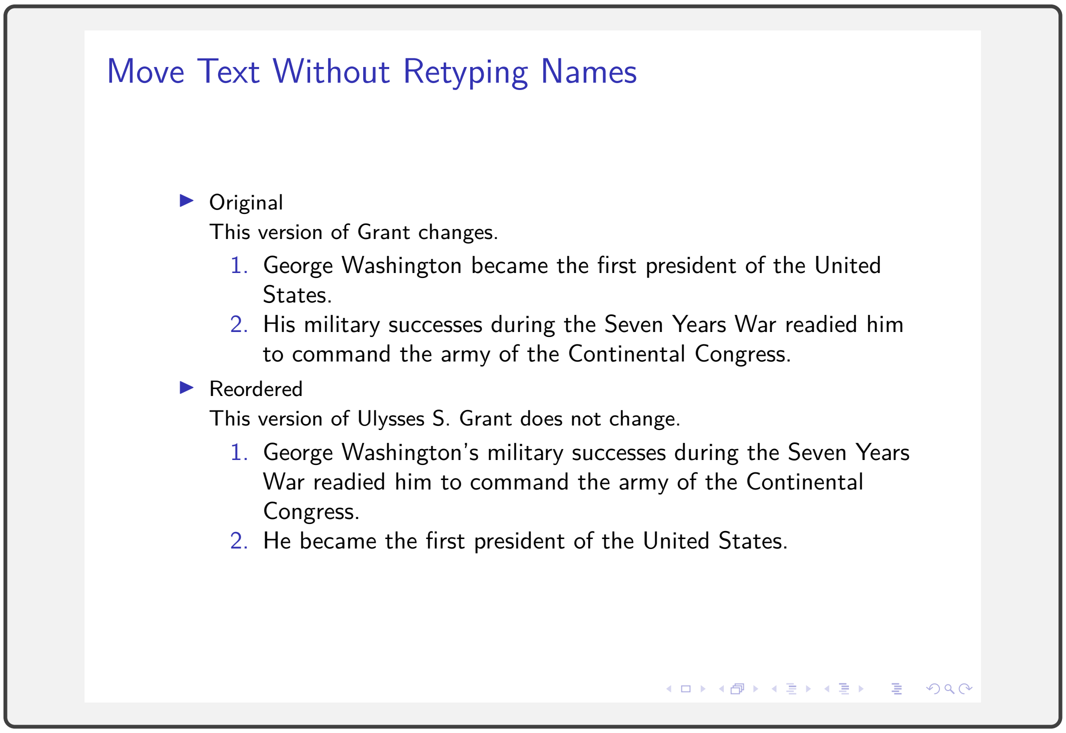
Move Text Without Retyping Names

- Original
  This version of Grant changes.
  1. George Washington became the first president of the United States.

- Reordered
  This version of Ulysses S. Grant does not change.

  2. He became the first president of the United States.

Overlay three fully illustrates how all these features integrate. This could allow a presenter to maintain information used in different presentations, making each element or slide a "drop-in" unit that can figure out how to present names, name forms, and related information without extensive retyping.



Move Text Without Retyping Names

- Original
  This version of Grant changes.
  1. George Washington became the first president of the United States.
  2. His military successes during the Seven Years War readied him to command the army of the Continental Congress.
- Reordered
  This version of Ulysses S. Grant does not change.
  1. George Washington's military successes during the Seven Years War readied him to command the army of the Continental Congress.
  2. He became the first president of the United States.

Back to Table of Contents

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

—William Shakespeare
"Romeo and Juliet", Act II, Scene II (published 1597)

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.

1. Use ⟨*Alternate*⟩ to create variant forms, yet retain consistent index entries. This is the default function of nameauth that readers already have seen.
2. Create several names. Index one or more occurrences of these names as "standard" variants that refer to each other via *see also* references. In relation to the "standard" variants, any other variants would be indexed only with a *see* reference. The macros \JustIndex, \IndexName, and \IndexRef play a big role here (Sections 7.2 and 7.3).
3. Use alternate formatting, \noexpand, and macros in the name arguments that expand differently under specific conditions in the formatting hooks, but expand consistently when indexed.

We will repeat the following rule in the discussion of Roman names (Section 11.4) and in the discussion of alternate formatting (Section 11.3). One cannot overstate this point when using mutable macros in name arguments:

When a macro occurs in a name argument appearing in both text and index, fix any concerns about macro expansion by using \noexpand before that macro.

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.

Output	Macro	Output	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).

We will engage this topic more, beginning with Section 11.2.2, Here we give a basic illustration of how one can start to manage these names. 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 class of name variants listed above, we get into pseudonyms, aliases, and variant family names. This class is more complicated:

1. Names that change capitalization in the surname, take grammatical endings, or vary in other ways. See Sections 5.7, 5.8, 11.2.2, 11.3.5ff., and 11.4, for increasingly complex examples.
2. Names having separate index entries that are linked together with cross-references. Sections 7.2 and 7.3 give many examples, which we will not repeat here because most of the work is done with indexing macros.
3. Names that the author wants to treat as different, yet which the nameauth internals might see as the same. One can use the naming macros, index tagging macros, and formatting macros to simulate the existence of multiple identical names (see below and Section 7.7).

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 control pattern: `W.E.B.!DuBois` (Section 6.1). 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, modify name forms, and check if the name straddles a page break in order to append `\JustIndex\DuBois` if needed:

  ```
  \newcommand\VarDuBois{\JustIndex\DuBois\AltDuBois}
  \newcommand\LVarDuBois{\JustIndex\DuBois\LAltDuBois}
  \newcommand\SVarDuBois{\JustIndex\DuBois\SAltDuBois}
  Speaking of \VarDuBois[William E.B.].\\
  Speaking of \LVarDuBois.\\
  Speaking of \SVarDuBois[William E.B.].
  ```

  Speaking of William E.B. DuBois.
  Speaking of W.E.B. DuBois.
  Speaking of William E.B.

- The macro below loses full stop detection, but it does automatically handle the name spanning a page break, just like the regular naming macros. Yet it is rather inelegant.

  ```
  \newcommand\NewDuBois[2]{%
    \def\Test{#1}%
    \def\Long{L}%
    \def\Short{S}%
    \JustIndex\DuBois%
    \ifx\Test\Long \LAltDuBois[#2] \else
      \ifx\Test\Short \SAltDuBois[#2] \else
      \AltDuBois \fi\fi
    \JustIndex\DuBois}
  \ForgetThis\NewDuBois{}{William E.B.}\\
  \NewDuBois{L}{}\\
  \NewDuBois{S}{William}
  ```

  William E.B. DuBois
  W.E.B. DuBois
  William

The cause of war is preparation for war.

—W.E.B. Du Bois
Darkwater (1920), C II: The Souls of White Folk

10.1.3 Nonstandard Capitalization and Indexing

Here we look at nonstandard capitalization. We consider poet e.e. cummings. As long as one sticks with the default `noformat` option, the easiest solution is to begin a sentence with something like:

Name Pattern(s):
`e.e.!cummings!MN`
Basic Index:
cummings, e.e.

```
\SubvertThis\CapThis\Name[e.e.]{cummings}'s motif of the
goat-footed balloon man has underlying sexual themes that
nevertheless present a childish facade.
```

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

Suppose, however, that we want both some kind of name formatting and still use capitalization. We can use the indexing macros discussed in Section 7.1:

Name Pattern(s):
`e.e.!cummings's`

```
\ExcludeName[e.e.]{cummings's}
We consider the work of \ForgetThis\Name[e.e.]{cummings}.
\SubvertThis\CapThis\Name[e.e.]{cummings's}
motif of the goat-footed balloon man has underlying
sexual themes that nevertheless present a childish facade.
```

We consider the work of e.e. cummings. 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.

Section 11.3 explains how to use `\CapThis` with alternate formatting when using macros in name arguments. Section 11.3.6 describes how automation lends itself to grammatical inflections of names.

10.1.4 Variant Names and Index Cross-References

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

Name Pattern(s):

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

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

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

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

   Jimmy Carter . . . . `\DropAffix\Name*[J.E.]{Carter, Jr.}[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 it 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. If we want to index the alternate name, we have to use the canonical name instead of the alternate name:

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

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

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

Multiple Connections

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

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

1. We use the canonical name to set up page 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 a couple of names from a name authority created for a translation of *De Diaconis et Diaconissis Veteris Ecclesiae Liber Commentarius* by Caspar Ziegler, of which the present author was the editor.[24]

Constructing that name authority of over 500 names was a challenge. The deceased translator left names in abbreviated Latin. He left many place-names in Latin and incorrectly translated some others. To get valid names that one can research, the present author recommends:

- CERL Thesaurus
- Virtual International Authority File
- EDIT16
- WorldCat
- Library of Congress
- Older version of Graesse, *Orbis Latinus*

I set the vernacular forms as canonical, with the Latin versions referring to them. I re-translated all the place-names. I also did the following:

1. Sort vernacular names with `\PretagName` due to particles (Section 7.6). 2. If Latin names are only cross-references, use `\IndexRef`⟨*name args*⟩ to generate cross-references before referring to any names (Section 7.3). 3. If Latin names have page entries, then place `\SeeAlso\IndexRef`⟨*name args*⟩ as needed at the end of the document, before `\printindex`. 4. Use `\CapThis` (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	`\Name[Jacques]{De~Pamele}[Jacques de~Joigny]` Jacques de Joigny De Pamele
Latin Name	`\Name[Jacobus]{Pamelius}` Jacobus Pamelius
Canonical Name	`\Name[Giovanni]{d'Andrea}` Giovanni d'Andrea
Latin Name	`\Name[Ioannes]{Andreae}` 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

[24]The book, *The Diaconate of the Ancient and Medieval Church* had been typeset using LaTeX, but then had to be converted to a different format.

11 Advanced Formatting

Up to this point, formatting hooks have taken a name whose form was set in the internal name parser and the hooks applied some typographic changes to that name.

In this section we start using formatting hooks in ways that interact with and change the syntactic form of a name, perhaps in addition to making typographic changes to that name.

We thus merge the two concepts of syntax and formatting to create more complex examples that are able to do more complicated things with names. We thus can meet a few specific, real-world cultural expectations and scholarly conventions.

Before we delve into this section, 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 all send their arguments to three hooks that, by default, point to `\@nameauth@Name` (Section 13.3). Here `\@nameauth@Name` handles the work requested by `\JustIndex`, `\SubvertThis`, `\ForgetThis`, and `\SkipIndex`. It then hands off the name arguments to the name parser, either with a main-matter `!MN` suffix or a front-matter `!NF` suffix. . .

Syntactic Element Layer

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

Name Display Layer

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

Format Hook Dispatcher

`\@nameauth@Hook`: Check the name to be printed for a final full stop. Check which naming system we are using. Call a formatting hook depending on if our name control sequence pattern exists, and if we are in the main-matter system or the front-matter system. The formatting hook prints the name in the text.

`\@nameauth@Parse`: Make name control sequence from name pattern.

`\@nameauth@NameEngine`: If the final full stop flag is true, check the lookahead token for a full stop and gobble it if needed.

11.1 Formatting Hooks

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, although we also could use a group to keep format changes in a local scope:

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

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}.

\printindex
\end{document}
```

The new format (front matter):

Vlad III Dracula
Vlad II Dracul

Vlad III Dracula, known as Vlad Țepeș (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 Țepeș (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.

The reason why we redefine `\NamesFormat` is because it is more common to add extra information with the first mention of a name. Yet one similarly could redefine `\MainNameHook` or `\FrontNameHook` for subsequent uses of names.

11.2 Name Tags in Hooks

By recalling name tags (Section 8) in formatting hooks, one can automate how they either appear with a name or not. This package is all about automation and the associated trade-offs.

Formatting hooks are called within a local scope. They take zero or one argument (Section 9.1). When they take one argument, they have the option to print that argument, add information to the argument, or discard the argument. Yet macros used in formatting hooks can present challenges that we show how to manage:

- When `\noexpand` is used in a name argument, `\NameQueryInfo` and perhaps other macros may not produce output within a formatting hook when using the basic interface, but the quick interface works fine.
- Macros that are used to make local changes in formatting hooks should never make the same changes outside of that context, else spurious index entries will occur.

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

Three token registers contain each of the name arguments used in a macro that takes them. They are needed to manage the proper expansion of name arguments, especially in the index. Historically, these registers have been necessary for names that contain accents and diacritics. In Section 12.1, these registers correspond to the **last** three name arguments. They can be used, if needed, in formatting hooks by nameauth macros that take name arguments.

Their chief use, historically speaking, was to facilitate retrieving name tags via `\NameQueryInfo`. That use has been superseded by `\NameauthPattern`, which is described in Section 6.1.

11.2.1 Hook Templates

Recommended Template

The hook below prints the name tag with the first use of a name in the main-matter system, if such a tag exists. It is simple, it avoids the `\NameQueryInfo` problem, and it is easy to debug.

```
\renewcommand*\NamesFormat[1]
{%
  #1%
  \ifcsname\NameauthPattern!DB\endcsname
    \expandafter\csname\NameauthPattern!DB\endcsname%
  \fi
}
```

Older Templates

Two other hook designs were used in the past to display name tags. They are included here only for the sake of illustration. After showing the templates, we test all three of them to show why we recommend the template above.

We use ϵ-TeX features next. See **this page** on the use of `\unexpanded` and **another page** pertinent to the structure of this hook. We print the name, expand the arguments of `\NameQueryInfo`, and use those arguments to print the tag.

Alternate A

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

The older form of this hook was the first to be used in `nameauth`, based on **this pdf article** from *TUGboat* that gives a tutorial on `\expandafter`. As one can see, it is more tedious, but gets the same result as above:

Alternate B

```
\let\ex\expandafter
\makeatletter
\renewcommand*\NamesFormat[1]{%
  #1%
  \ex\ex\ex\ex\ex\ex\ex\NameQueryInfo%
  \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@etoksc]%
}
\makeatother
```

Template Test

We set up the following test for our three templates, with indexing suppressed:

- A macro in a name argument is preceded by `\noexpand`. That occurs with greater frequency in advanced formatting.
- If we see the outcome, “Test passed”, we have success.
- If we see the outcome, “Test”, we have failure.

The macro `\testname` contains the first part of our test output. The name tag contains the second part. We will use both name interfaces with all three templates.

```
\newcommand\testname{Test}
\NameAddInfo{\noexpand\testname}{ passed}
\begin{nameauth}
  \< Test & & \noexpand\testname & >
\end{nameauth}
```

- The recommended hook gives us:

`\ForgetThis\Name{\noexpand\testname}`	Test passed	success
`\ForgetThis\Test`	Test passed	success

- Alternate A gives us:

`\ForgetThis\Name{\noexpand\testname}`	Test	failure
`\ForgetThis\Test`	Test passed	success

- Alternate B gives us:

`\ForgetThis\Name{\noexpand\testname}`	Test	failure
`\ForgetThis\Test`	Test passed	success

11.2.2 Application: Ancient Names

Ancient names tend to be the most fluid regarding the meaning and use of affixes. Certain scholarly contexts add more information to a name when it is first introduced. Here we look at ways to address that need. For the sake of clarity, here the examples do not use the formatting conventions of this manual.

First we explore the easiest way to handle royal or ancient variants by manually adding any epithets or sobriquets to a standard name form:

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

- We use `\PretagName` to sort especially Roman numerals in the index. For example: `\PretagName{Antiochus, IV}{Antiochus 4}`

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

- We use `\TagName` to ensure that any “long form” information is displayed in the index: `\TagName{Antiochus, IV]}{ Epiphanes, king}`.
- Using `\PretagName` and `\TagName` in the preamble ensures consistency.

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

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

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

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:

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

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

```
\ForgetName{Demetrius, I}
\Dem[I Soter] ........................................ Demetrius I Soter
\LDem ...................................................... Demetrius I
\Dem ......................................................... Demetrius
```

As discussed in Section 1.7, simple examples do not lend themselves to automation. Below we trade automation, where we do not need to add information manually to the ⟨*Alternate*⟩ argument, for a complex initial setup that uses name tags and the recommended template:

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

```
\NameAddInfo{Demetrius, I}{ Soter}
\renewcommand*\NamesFormat[1]
{%
  #1%
  \ifcsname\NameauthPattern!DB\endcsname
    \expandafter\csname\NameauthPattern!DB\endcsname%
  \fi
}
\renewcommand*\MainNameHook{}
```

```
\ForgetName{Demetrius, I}
\Dem ................................................. Demetrius I Soter
\LDem ...................................................... Demetrius I
\Dem ......................................................... Demetrius
```

In both cases, the index entry in a normal document would be sorted like the following: `Demetrius 1@Demetrius I Soter, king`.

The first use of a name, or one after the use of `\ForgetName` or `\ForgetThis`, shows the most information. A long instance of an extant name shows a little less info, and a short instance shows the least. Every difference in name form corresponds with a macro in a known state.

11.2.3 Application: Life Dates

History texts tend to use life dates, regnal dates, and dates when certain figures flourished. As we used more information with ancient names via name tags, we can do something similar here.

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 have a leading space because they are printed conditionally.

We also define a cross-reference “Atatürk,” yet we use naming macros with that name, printing formatted names in the text but making no index page entries. 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. Below we suppress the usual formatting of this manual.

```
\documentclass{article}
\input{compat.tex} % Included with nameauth; example file aids
% compatibility across different LaTeX versions and engines.
\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%
  \ifcsname\NameauthPattern!DB\endcsname
    \unless\ifNoTag
      \expandafter\csname\NameauthPattern!DB\endcsname%
    \fi
    \global\NoTagfalse%
  \fi
```

Up to this point it seems as if there has been a lot of setup. The payoff, however, comes in the main body text where the use of the naming macros does not look much different than normal.

```
}

\begin{document}

\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:
```

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 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:

```
\ForceName\Name[George]{Washington}.

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 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 tag: Atatürk.

11.3 Alternate Formatting

We build on the subject of complex formatting hooks that `\noexpand` playing a role therein, Before we engage the topic of Roman names and other complex examples, we need to cover alternate formatting.

11.3.1 Enabling and Disabling

The first thing we need to know is how to enable and disable alternate formatting:

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

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

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

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

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

Most `nameauth` macros that turn things on and off have a local scope unless one uses `\global`. These alternate formatting macros have global scope to eliminate implied scope becoming a point of failure, creating spurious index entries.

The macros above **always** make global changes. Using names designed for alternate formatting in a document section that uses regular formatting will produce an inconsistent appearance in the text and spurious index entries.

11.3.2 Using `\noexpand`

As we get into advanced formatting, we will encounter `\noexpand` with greater frequency. Even before we consider that discussion, we need to touch on a few ways that macros can break `nameauth` when used in name arguments.

In order to handle the inherent ambiguity of name forms, macros that take name arguments trade a certain "fragility" for the ability to be as close to natural language as possible. In versions of `nameauth` before 3.5, enclosing an ⟨*SNN*⟩,⟨*Affix*⟩ argument within a robust macro like `\textsc` would halt LaTeX with errors. That seems to be no longer the case. Nevertheless, depending on the macros used, it may be helpful to apply macros separately to ⟨*SNN*⟩ and ⟨*Affix*⟩ like this:

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

Using `\CapThis` with a name whose leading element is a macro may fail, depending on the particular macro. Use alternate formatting to have `\CapThis` activate the alternate capitalization mechanism.

When a macro occurs in a name argument appearing in both text and index, fix any concerns about macro expansion by using `\noexpand` before that macro.

In Section 6.3 we encountered reasons for using `\noexpand` in name arguments, as well as related caveats. Now we include more reasons to use `\noexpand`:

- If a macro is undefined, even putting `\noexpand` before it will permit an error unless the macro is detokenized or verbatim.
- The use of `\noexpand` isolates the local, conditional expansion of macros in the formatting hooks from the global macro state in the index.
- Indexing of such names normally does not occur within the formatting hooks. Otherwise, different index entries would result.
- One can use macros in the formatting hooks to trigger local changes. If using Boolean flags to trigger those changes, one needs two flags per change (e.g., grammatical inflection) but not per name.
- Local flags are false when the hook executes. Global flags are reset when the hook terminates.
- Take care when using macros in name arguments without using `\noexpand` before them.

11.3.3 Alternate Capitalization

Above we referred to potential problems using `\CapThis`. When alternate formatting is enabled, `\CapThis` changes its mechanism to avoid such problems.

`\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. We describe the syntax:

```
\noexpand\AltCaps{⟨Arg⟩}
```

We offer a silly example below, taking advantage of the local scope of the `quote` environment to disable indexing temporarily:

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

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

`\AltCaps` does not partition its argument, so it will not have the same issues as the normal use of `\CapThis`, adding to stability and robustness at the expense of doing a little more work.

11.3.4 Formatting Features

`\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.

By themselves, they do not change format. Yet the macros `\AltOff` and `\AltOn`, described shortly, are able to turn the formatting of these macros on and off. We advise using `\noexpand` before these macros because they can be made to change format. Assuming that we have sorted the following names with `\PretagName` (Section 7.6), we get the following, using this manual's formatting conventions:

```
\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

Font substitutions might occur with these macros, depending on the font used. `\CapName`, `\RevComma`, and `\RevName` can modify the names, but only in the text.

The alternate formatting macros shown above become more interesting when we automate how they turn on and off. Using `\noexpand` is necessary here. Both macros below are used in formatting hooks. They hide some complexity from authors.

`\AltOff` Vaguely reminiscent of depressing an automobile's manual clutch lever, `\AltOff` deactivates `\textSC`, `\textBF`, `\textIT`, and `\textUC` only in a formatting hook. The alternate formatting mechanism is still "running", but it is not transferring "power" to the formatting macros. They display their arguments unmodified.

`\AltOn` Likewise, `\AltOn` activates `\textSC`, `\textBF`, `\textIT`, and `\textUC` only in a formatting hook, as if one let out the clutch pedal, causing "power" to transfer through the gearbox to the formatting macros. They now modify their arguments.

If one uses the `altformat` option or `\AltFormatActive`, the formatting "power" goes to the formatting macros by default in order to have formatted names in the index. Otherwise, the normal formatting regime isolates formatting in the text, as the Anglosphere seems wont to do.

We use `\noexpand` as discussed and add a formatting hook to get changes in the text, not in the index. We also suspend this manual's formatting conventions:

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

\begin{document}

\renewcommand*\MainNameHook{\AltOff}

\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}

\printindex
\end{document}
```

Konrad ZUSE; Zuse
Ada *Lovelace*; Lovelace
Charles **Babbage**; Babbage
KANADE Takeo; Kanade

11.3.5 History Text

First we engage the idea of a history text, where we use standards for medieval Italian to encode a name instead of those used in modern Romance languages.

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

Comparing the example above to Section 5.7 shows us some differences. Medieval Italian, similar to modern German, keeps the particle(s) with the forename(s). Modern Italian groups particles and surnames together. Thus, we must use here:

- de' Medici `\LCath[\noexpand\AltCaps{d}e']`
- De' Medici `\CapThis\LCath[\noexpand\AltCaps{d}e']`

11.3.6 Inflected Names

Next we design grammatical inflections for use with alternate formatting. We do not use the general formatting conventions in this manual. We shall build on this example in order to design the more complex hooks used for Roman names.

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; example file aids
% compatibility across different LaTeX versions and engines.
\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.

11.3.7 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 more complicated Western name that includes an alias. Here are a few points to consider:

- Sort all names that use alternate formatting.
- Match index entry forms to article head-words. Name variants are cross-referenced to the name form in the head-word.
- Ensure that the first appearance of a name displays only the active alternate formatting. Any additional formatting comes from the macro that formats entries.
- Deactivate alternate formatting in subsequent name instances.
- Since some LaTeX fonts do not combine small caps and boldface, we instead use slanted text to set the head-words off from the articles. That font switch is done in the macro that formats the articles (`\RefArticle`).

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

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)

11.4 Roman Names

Roman names tend to be among the most variable in terms of treatment. As far as nameauth is concerned, they range from being treated as Western names to being treated in very special ways. Below we show some variations.

11.4.1 Casual Reading / General Book Market

- Treat as a Western name, especially among well-known Roman names like Marcus Tullius Cicero, where the final name, Cicero, is a surname and the rest forenames:[25]

```
\TagName[Julius]{Caesar}{, imperator} % for index
\Name[Julius]{Caesar}[Gaius Julius]\\
\Name*[Julius]{Caesar}\\
\Name[Julius]{Caesar}
```

Name Pattern(s):
`Julius!Caesar!MN`

Gaius Julius Caesar
Julius Caesar
Caesar

Index: Caesar, Julius

- Treat as a Western name; put *nomen* and *cognomen* in ⟨*SNN*⟩:[26]

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

Name Pattern(s):
`Lucius!SergiusPaulus!MN`

Lucius Sergius Paulus
Sergius Paulus

Index: Sergius Paulus, Lucius

- Treat as ancient names, especially those with no *praenomen*:

```
\ForgetThis\Name{Pontius, Pilate}[Pilatus]\\
\Name*{Pontius, Pilate}\\
\ForceFN\FName{Pontius, Pilate}
```

Name Pattern(s):
`Pontius,Pilate!MN`

Pontius Pilatus
Pontius Pilate
Pilate

Index: Pontius Pilate

There is also a method where Roman names are indexed as mononyms using the most recognizable of their names.[27] That method is ill-suited for nameauth.

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)

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

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

[27]Justo L. González, *The Story of Christianity*, 2 vols. (San Francisco: Harper, 1984).

11.4.2 Student-Oriented Reference Works

We focus here on reference works meant for general education. In this subsection and the next, we do not use the general formatting conventions of this manual and we activate alternate formatting. Roman names have the following format:

- A personal name (*praenomen*)
- A clan name (*nomen*)
- A distinguishing name, sometimes denoting clan branches (*cognomen*); that could include various affixed names (*agnomina*)

In ancient Rome, the family name (*nomen*) was important, helping also to structure society. This page offers a good overview. The exact roles of names changed over time. Another page gives an overview of Roman naming conventions. A very helpful video can be found on the YouTube channel PolýMATHY by Luke Ranieri.

Some reference works treat the *cognomen* as if it were a Western surname.[28] Using this approach, Roman names encoded with `nameauth` have this form:

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

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

With both *praenomen* and *nomen* in ⟨*FNN*⟩, as well as the *agnomina* in ⟨*Affix*⟩, they drop automatically in subsequent name uses. With a *cognomen* as an effective surname, our choices for name components in the text take on this logic:

- Display one of the following in ⟨*FNN*⟩:
  - Only the *praenomen*
  - Only the *nomen*
  - Both *praenomen* and *nomen*
- Display one of the following in ⟨*SNN*⟩:
  - Only the *cognomen*
  - Both *cognomen* and *agnomina*

We accomplish this by using macros in the name arguments:

- If the macros in the name arguments will be "segmented" in some way, as `\CapThis` does, then use alternate formatting (Sections 11.3).
- 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.

[28] 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**.

When a macro occurs in a name argument appearing in both text and index, fix any concerns about macro expansion by using `\noexpand` before that macro.

Since we have four name components, we need at most eight Boolean flags. Our two examples each use six flags, with four in common and one separate pair each. In the preamble we define all macros and conditionals used in naming macro arguments. Instead of `\ifNoNomen` we use `\ifNoGens` for readability.

We must use `\noexpand` in the macro arguments. We define macros in both ⟨*FNN*⟩ and ⟨*SNN*⟩ that expand conditionally. We use the quick interface to define name shorthands, and we sort the name with `\PretagName`. False Boolean flags display longer name forms. True flags display shorter forms.

We used `\PretagName` and `\TagName` as needed (cf. Section 11.2.2. These formatting hooks used with Roman names also work with other name types.

```
\documentclass{article}
\input{compat.tex} % Included with nameauth; example file aids
% compatibility across different LaTeX versions and engines.
\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 need to be defined uniquely for each
% name. First is Scipio. Second is Gracchus.

\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. The first shows
% the concept of Roman names without extra features.
% The second (Gracchus) adds name info tags.

\begin{nameauth}
  \< Scipio & \noexpand\SCIPi & \noexpand\SCIPii & >
  \< TGrac & \noexpand\TSemp & Gracchus & >
\end{nameauth}

% We add the 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}

\begin{document}

% Although it is helpful to set everything up
% In the preamble, it is not absolutely necessary.
% Here we define the simpler set of formatting hooks
% for Scipio, although the complex hooks will work
```

Formatting hook macros need to be redefined only once. We reset the global Boolean flags to ensure the longest name forms in the index. The local Boolean flags automatically revert to false outside the scope of the formatting hooks.

```
% for both equally as well.

\renewcommand*\NamesFormat[1]
{%
  \ifNoPraenomen\XPraetrue\fi%
  \ifNoGens\XGenstrue\fi%
  \ifNoCognomen\XCogntrue\fi%
  \ifNoAgnomen\XAgnotrue\fi%
  #1%
  \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%
}

\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
```

> 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 name variations than were shown above. In addition, we use those same formatting hooks to display non-Roman names:

Publius Cornelius Scipio Africanus	`\ForgetThis\Scipio`
Publius Cornelius Scipio Africanus	`\LScipio`
Scipio Africanus	`\Scipio`
Publius Cornelius	`\SScipio`

Publius Cornelius Scipio	`\NoAgnomentrue\LScipio`
Scipio	`\NoAgnomentrue\Scipio`
Publius	`\NoGenstrue\SScipio`
Cornelius	`\NoPraenomentrue\SScipio`

Ptolemy IV Philopator	`\Name*{Ptolemy, IV}[IV Philopator]`
Ptolemy IV	`\Name*{Ptolemy, IV}`
Ptolemy	`\Name{Ptolemy, IV}`

In Section 11.2.2, we used name tags instead of ⟨*Alternate*⟩ in the first-use formatting hook. Here we show the changes needed to add name tags to the first-use formatting hook. Even though we are changing the formatting hooks throughout the document, none of the changes cause different index entries and will be unnoticed in the finished document.

```
\Name{Antiochus, III}[III the Great].

% We make no change to \MainNameHook, but we do
% change \NamesFormat to display any extant
% name tags.

\renewcommand*\NamesFormat[1]
{%
  \ifNoPraenomen\XPraetrue\fi%
  \ifNoGens\XGenstrue\fi%
  \ifNoCognomen\XCogntrue\fi%
  \ifNoAgnomen\XAgnotrue\fi%
  #1%
  \ifcsname\NameauthPattern!DB\endcsname
    \expandafter\csname\NameauthPattern!DB\endcsname%
  \fi
  \global\NoPraenomenfalse%
  \global\NoGensfalse%
  \global\NoCognomenfalse%
  \global\NoAgnomenfalse%
}

\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.

Below we show both a Roman name and a different name type working the same way with the formatting hook:

Tiberius Sempronius Gracchus (consul, 177 BC)	`\ForgetThis\TGrac`
Tiberius Sempronius Gracchus	`\LTGrac`
Tiberius Gracchus	`\NoGenstrue\LTGrac`
Gracchus	`\TGrac`
Tiberius Sempronius	`\STGrac`
Tiberius	`\NoGenstrue\STGrac`
Sempronius	`\NoPraenomentrue\STGrac`
Demetrius I Soter	`\ForgetThis\Dem`
Demetrius I	`\LDem`
Demetrius	`\Dem`

Bellum parate, quoniam pacem pati non potuistis.

(Prepare for war, for it seems that you are unable to tolerate peace.)[29]

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

[29] Here we enclosed this quote in a group and redefined the formatting hooks locally.

11.4.3 Scholarly Reference Works

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

> Index: ⟨*nomen*⟩ ⟨*cognomen*⟩ ⟨*agnomen*⟩, ⟨*praenomen*⟩
>
> Macro: `\Name[`⟨*praenomen*⟩`]{`⟨*nomen*⟩ ⟨*cognomen*⟩ ⟨*agnomen*⟩`}`

This indexing method is incompatible with the previous section, but we show both in this manual. We retain most of the code used in the last section, but we change the macros used in the name arguments. The logic still shows all names for the index, but we make different choices in the text:

- 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; example file aids
% compatibility across different LaTeX versions and engines.
\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 need to be 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%
  \ifcsname\NameauthPattern!DB\endcsname
    \expandafter\csname\NameauthPattern!DB\endcsname%
  \fi
  \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 names 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`

One may create convenience macros instead of using the Boolean flags. Yet this might make things less clear at first glance, for example:

```
\newcommand*\LucScipio{\NoGenstrue\NoAgnomentrue\OScipio}
Lucius Scipio ...................................... \ForgetThis\LucScipio
Scipio ......................................................... \LucScipio
```

The student and scholarly forms of Roman names are incompatible. Yet we show what the index entries would be in a normal LaTeX document without hyperlinks:

Popular Reference Works:
`\ShowIdxPageref[\noexpand\SCIPi]{\noexpand\SCIPii}`
Scipio Africanus@Scipio Africanus, Publius Cornelius

`\ShowIdxPageref[\noexpand\TSemp]{Gracchus}`
Gracchus, Tiberius Sempronius@Gracchus, Tiberius Sempronius, consul

Scholarly Reference Works:
`\ShowIdxPageref[Lucius]{\noexpand\CSB}`
Cornelius Scipio Barbatus@Cornelius Scipio Barbatus, Lucius, consul

11.5 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 15.7.6) builds a printed name from internal, locally-scoped macros. Its output is affected by a subset of the Boolean flags, namely, those flags that affect long and short forms, as well as name reversal. Within the hooks we can use the user-accessible parser as often as we want.

- `\if@nameauth@FullName` toggles long or short name forms, as in `\Name` versus `\Name*`. `\if@nameauth@FirstName` toggles forenames when `\if@nameauth@FullName` is false. When modifying these flags, one must know when they are normally true or false.
- `\if@nameauth@RevThis` reverses name order. `\ForceFN` normally toggles `\if@nameauth@EastFN`. Using `\if@nameauth@RevThis`, reversing without commas, overrides `\if@nameauth@RevThisComma` if both are true.
- `\if@nameauth@RevThisComma` creates the form ⟨*SNN*⟩, ⟨*FNN*⟩.

`\ifNameauthWestern`
`\ifNameauthObsolete` Two Boolean flags are available to users so that they can know quickly what type of name was processed most recently by `\@nameauth@Choice`, which is called by all macros that take name arguments. These allow hook designers to vary the hook behavior based on the name type.

These flags also persist after the macro that uses name arguments exits, unlike many other flags. This allows one to poll the last macro that took naming arguments, which may or may not be the last name printed in the text.

To illustrate how one might poll these flags, either in a formatting hook or after a macro that takes name arguments exits, we set up the following macros and show how the flags are set. Notice that the name type is a function solely of the name arguments used, not how the name eventually ends up printed in the text, and not based on the name control sequence.

```
\newcommand\ShowType
{%
  \ifNameauthWestern Western name\else
    \ifNameauthObsolete nonwestern name using obsolete syntax\else
    nonwestern name using current syntax\fi
  \fi
}

First, we begin with a Western name:

\Name*[Albert]{Einstein} is a \ShowType.

Even though this name is reversed in the text, it is still Western:

\RevName\Name*[Frenec]{Liszt}\dag\ is a \ShowType.

Here we present the same name twice, but using the two different
syntax versions for nonwestern names:

\Name*{Henry, VIII} is a \ShowType.

\Name*{Henry}[VIII]\ddag\ is a \ShowType.

Finally, even though we do not print the name,
\IndexName{Elizabeth, I} we get a \ShowType.
```

First, we begin with a Western name:

Albert Einstein is a Western name.

Even though this name is reversed in the text, it is still Western:

Liszt Frenec† is a Western name.

Here we present the same name twice, but using the two different syntax versions for nonwestern names:

Henry VIII is a nonwestern name using current syntax.

Henry VIII‡ is a nonwestern name using obsolete syntax.

Finally, even though we do not print the name, we get a nonwestern name using current syntax.

11.5.1 Reference Work II

Below we revisit the idea of a reference work, leveraging the formatting hooks to format headwords, add information from name info tags:

Since we are using the `altformat` option or `\AltFormatActive`, we can expect that formatting is activated by default. We design a subsequent-use hook that deactivates alternate formatting inside of it:

```
\renewcommand*⟨SubsequentHook⟩[1]{...\AltOff\NameParser...}
```

If we had used `\AltFormatActive*`, where the formatting macros are enabled, but **deactivated** by default, then we might want instead a first-use hook that activates the macros:

```
\renewcommand*⟨FirstHook⟩[1]{...\AltOn\NameParser...}
```

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

\makeindex

% Boolean flags; the first sets up headwords and the second
% indicates that a nonwestern 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)}

% Redefining the formatting hooks:
\makeatletter
\renewcommand\NamesFormat[1]
{%
  \ifHeadword
    \ifNameauthWestern
      \@nameauth@RevThisCommatrue%
      \bfseries \NameParser%
      \normalfont%
      \ifcsname\NameauthPattern!DB\endcsname
        \expandafter\csname\NameauthPattern!DB\endcsname%
      \fi
    \else
      \bgroup%
        \bfseries \NameParser%
        \unless\ifAncientName
          \normalfont; W:\AltOff\space
          \@nameauth@RevThistrue \NameParser%
        \fi
        \normalfont%
        \ifcsname\NameauthPattern!DB\endcsname
          \expandafter\csname\NameauthPattern!DB\endcsname%
      \fi
    \egroup%
  \fi
\else
  \NameParser%
\fi
\global\Headwordfalse%
\global\AncientNamefalse%
}
\makeatother
\renewcommand\MainNameHook{\AltOff}

% Define related macros:
\newcommand\Headword{\Headwordtrue\ForgetThis}
\newcommand{\RefArticle}[2]
{%
  \noindent\Headword #1 #2%

}

\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…

11.5.2 Marginalia

Starting where we left off with Roman names, 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. We use `\PretagName` to sort the names. `\Revert` is used to print a last name without a margin note.

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

\makeindex

% Global Boolean flags need to be defined only once.
\newif\ifSpecialFN
\newif\ifSpecialSN
\newif\ifRevertSN

% Name variant macros need to be defined
% uniquely 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}
```

Next we define the two formatting hooks that structure the ways in which these macros can expand when printed in the text. We force `\NamesFormat` to print only allows only the default name via `\RevertSNfalse`, `\SpecialFNfalse`, and `\SpecialSNfalse`. `\MainNameHook` disables alternate formatting with `\AltOff`, but it allows variant name forms.

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.

```
\PretagName[Robert]{\textSC{Burns}}{Burns, Robert}

% The ``first-use'' hook prints a name, then tries
% to insert a margin note using a different name form
% and the user-accessible parser. Finally it resets
% the reversion flag, which is only effective in the
% ``subsequent-use'' hook. Note how macros in the
% name arguments take the role of what the internal
% Boolean flags might otherwise handle.

\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%
}
```

Finally, we put all these macros to work in the text:

```
\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".

Back to Table of Contents

Antony: Friends, Romans, countrymen, lend me your ears;
I come to bury Caesar, not to praise him.
The evil that men do lives after them;
The good is oft interred with their bones;
So let it be with Caesar. The noble Brutus
Hath told you Caesar was ambitious:
If it were so, it was a grievous fault,
And grievously hath Caesar answer'd it.
Here, under leave of Brutus and the rest—
For Brutus is an honourable man;
So are they all, all honourable men—
Come I to speak in Caesar's funeral.
He was my friend, faithful and just to me:
But Brutus says he was ambitious;
And Brutus is an honourable man.

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

12 Planned Obsolescence

12.1 Almost Obsolete: Pseudonym Macros

The macros described in this section were early features of `nameauth`. They do not work like many of the other macros that display names and may be removed in the future. We retain them at present for backward compatibility.

12.1.1 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 Alternate*⟩`]`

- 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 ⟨*Alternate*⟩ nor the obsolete syntax cannot 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):

12.1.2 The Macros

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

- 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 Alternate*⟩ is present in a Western name form, then instead of ⟨*xref FNN*⟩, ⟨*xref Alternate*⟩ will be printed in the text.
- If ⟨*xref Alternate*⟩ is present in a nonwestern name form, then instead of ⟨*xref Affix*⟩ (if present), ⟨*xref Alternate*⟩ will be printed in the text.
- If ⟨*xref Alternate*⟩ is present without either ⟨*xref FNN*⟩ or ⟨*xref Affix*⟩, the obsolete syntax is used.

- `\AKA*` is analogous to `\FName`. `\ForceFN` works with it. The `oldAKA` option implies `\ForceFN` with every use of `\AKA*`.
- Neither `\AKA` nor its derivatives permit the effects of `\ForgetThis` and `\SubvertThis` to "pass through" because they produce output in the text. The `oldreset` option negates this.

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]`

We display other name forms after referring to some names to ensure that they have sensible page entries:

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.

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`

Sun King	`\AKA{Louis, XIV}{Sun King}`
Sun King	`\AKA*{Louis, XIV}{Sun King}`
Li Er	`\AKA{Lao-tzu}{Li, Er}`
Li	`\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}`

`\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 rubbish, these two macros are a biological hazard.

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

`\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 Alternate*⟩.

- 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.
- ⟨*Alternate*⟩ and the obsolete syntax work only with the cross-reference.

If we attempted to use `\PName*{William, I}[William]{the Conqueror}`, this macro would put "William I (William the Conqueror)" in the body text, but its index entry would be incorrect: "the Conqueror, William *see* William I". We use `\ForgetName{William, I}` to prepare for the following example. We do not show the name patterns in the margin.

Macro:	`\PName[Mark]{Twain}[Samuel L.]{Clemens}`
Text:	Mark Twain (Samuel L. Clemens) Twain (Samuel L. Clemens)
Macro:	`\PName*[Mark]{Twain}[Samuel L.]{Clemens}`
Text:	Mark Twain (Samuel L. Clemens)
Index:	Clemens, Samuel L. *see* Twain, Mark
Macro:	`\PName{Voltaire}[François-Marie]{Arouet}`
Text:	Voltaire (François-Marie Arouet)
Index:	Arouet, François-Marie *see* Voltaire
Macro:	`\PName{William, I}{William, the Conqueror}`
Text:	William I (William the Conqueror) William (William the Conqueror)
Index:	William the Conqueror *see* William I

Macro:	`\PretagName{\textit{Doctor mellifluus}}{Doctor mellifluus}`
	`\PName{Bernard, of Clairvaux}{\textit{Doctor mellifluus}`

Text:	Bernard of Clairvaux (*Doctor mellifluus*) Bernard (*Doctor mellifluus*)
Index:	*Doctor mellifluus* *see* Bernard of Clairvaux
Macro:	`\ForgetThis\PName{Lao-tzu}{Li, Er}`
Text:	Lao-tzu (Li Er)
Index:	Li Er *see* Lao-tzu

While these macros certainly work, much like the obsolete syntax, they can be criticized as a design idea that originally seemed to hold promise, yet disappointed in practice due to using the cross-reference system for name formatting.

12.1.3 Formatting Workarounds

By default, the macros in this section use only the formatting hooks for subsequent instances of names (Section 9.1).

`formatAKA` First, the `formatAKA` option will permit `\ForceName` to force the "first-use" formatting hooks to be employed, but under different conditions. The name patterns that control these uses (Section 6.1) apply only to cross-references; they are a distinct system of patterns that differs from normal names and ignores the difference between main-matter and front-matter name formatting.

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

	`\ForgetThis\Eliz`	`\AKA{Elizabeth,I}%`
		`[Good Queen]{Bess}`

Front Matter:	Elizabeth I	Good Queen Bess
Main Matter:	Elizabeth I	Good Queen Bess
With `\ForceName`:		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 not using the regular name patterns. We need to use `\ForceName`, which triggers the expected use of `\NamesFormat`, the first-use main-matter hook.

`alwaysformat` Second, we can use the `alwaysformat` option to force only the use of `\NamesFormat` and `\FrontNamesFormat`. Of course, this can look like rubbish in certain circumstances.

> 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 nonwestern syntax limits alternate names and cross-references, prevents the use of comma-delimited names, and complicates indexing. It is a "ghost of nameauth past" that remains for the sake of backward compatibility and to prevent "holes" where naming macro arguments are discarded without apparent reason.

```
\Name{⟨SNN⟩}[⟨Alternate⟩]
```

The genesis of this syntax was the use of the ⟨*Alternate*⟩ field for variant forenames in Western names, personal names in Eastern names, and sobriquets, titles, and so on in ancient names. Yet this prevented using alternate names for nonwestern names and it limited those names to an unacceptable second-tier status. Developing the comma delimiter in ⟨*SNN, Affix*⟩ presented significant challenges, but it was worth overcoming them to put all name forms on equal footing.

We show this syntax for the sake of completeness, but we strongly encourage using the comma-delimited syntax instead.

- One must **leave empty** the first optional ⟨*FNN*⟩ argument.
- One must **never** use the comma-delimited argument ⟨*Affix*⟩.
- These names **always** use ⟨*Alternate*⟩, which acts like ⟨*Affix*⟩ usually does, and affects both name and index patterns (Section 6.1).
- These names take the form ⟨*SNN Alternate*⟩ in the index.
- In this manual we designate these names with a double dagger (‡).

```
\Name{Henry}[VIII]                             % Ancient
\Name{Chiang}[Kai-shek]                        % Eastern
\begin{nameauth}
  \< Dagb & & Dagobert & I >                   % Ancient
  \< Yosh & & Yoshida  & Shigeru >            % Eastern
  \< 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‡
Henry‡
	`\ForgetThis\Name{Henry}[VIII]\ddag`
	`\Name{Henry}[VIII]\ddag`

Chiang Kai-shek‡
Chiang‡
	`\ForgetThis\Name{Chiang}[Kai-shek]\ddag`
	`\Name{Chiang}[Kai-shek]\ddag`

Dagobert I‡ Dagobert‡	`\Dagb\ddag` `\Dagb\ddag`
YOSHIDA Shigeru‡ Shigeru YOSHIDA‡	`\CapName\Yosh\ddag` `\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 . . . . `\Name[Henry]{VIII}`
  VIII . . . . `\Name[Henry]{VIII}`
  Tudor VIII . . . . `\Name*[Henry]{VIII}[Tudor]`
  VIII . . . . `\Name[Henry]{VIII}[Tudor]`
- These malformed names have the same index entry . . . . VIII, Henry

Back to Table of Contents

Names, once they are in common use, quickly become mere sounds, their etymology being buried, like so many of the earth's marvels, beneath the dust of habit.

—Salman Rushdie, *The Satanic Verses* (1988)

13 Advanced Customization

Here we discuss package internals to help users customize and add features. We set aside the common formatting in this manual and reset the formatting hooks to the default settings. We use alternate formatting for much of this section:

```
\renewcommand*\NamesFormat{}
\renewcommand*\FrontNamesFormat{}
\renewcommand*\MainNameHook{\AltOff}
\renewcommand*\FrontNameHook{\AltOff}
```

13.1 Using Package Internals

We start with alternate formatting (Section 11.3), 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; example file aids
% compatibility across different LaTeX versions and engines.
\usepackage{makeidx}
\usepackage[altformat]{nameauth}

\makeindex

% Alternate formatting macro definition
\makeatletter
\newcommand*\Fbox[1]{%
  \if@nameauth@DoAlt\protect\fbox{#1}\else#1\fi
}
```

Since `\AltCaps` is part of `nameauth`, one need not reinvent that wheel. Just use it in the name arguments and sorting macros:

```
\makeatother

% 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}}%
```

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.[30]

—Maimonides, *Guide for the Perplexed* (1190)

[30] Here we enclosed this quote in a group and redefined the formatting hooks locally.

The final step is redefining the hooks, which can be quite simple:

```
  {de~Smet, Pierre-Jean}

\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 $\boxed{\text{de Smet}}$ 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.

Some formatting, such as the use of `\textSC`, is fairly standard. Other formatting, such as `\Fbox` above, is more ornamental.

13.2 Using Separate Macros

Alternate formatting can work with greater customization, but that is more labor-intensive and requires one to keep track of more details. We still recommend using `\AltFormatActive` to mitigate errors. The following example shows greater customization that remains compatible with package internals.

Three Boolean flags replace package internals. The flag `\ifFbox` activates formatting, `\CapThis` sets `\ifFirstCap` true, and `\ifInHook` is set manually, instead of being set by the hook dispatcher.

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

\makeindex

\newif\ifFbox     % Replaces \if@nameauth@DoAlt  \AltOn \AltOff
\newif\ifFirstCap % Replaces \if@nameauth@DoCaps \AltCaps
\newif\ifInHook   % Replaces \if@nameauth@InHook hook dispatcher
```

The formatting macro is like what we have seen, except it refers to `\ifFbox`. We define `\Fbox` locally in this manual, but `\noexpand` helps isolate those effects.

```
\Fboxtrue         % Replaces \AltFormatActive

% Alternate formatting macro definition
\newcommand*\Fbox[1]{%
  \ifFbox\protect\fbox{#1}\else#1\fi
```

Our `\AltCaps` works like the built-in version, except it does not use the internal macros and flags. We redefine `\CapThis` to use our flag instead of the internal flag:

```
}

% Redefinition of \AltCaps and \CapThis
\renewcommand*\AltCaps[1]{%
  \ifInHook
    \ifFirstCap\MakeUppercase{#1}\else#1\fi
  \else
    #1%
  \fi
}
```

We have to reproduce the logic and macros that the package would have provided. That means defining everything from scratch. To emphasize these differences, instead of displaying the macro argument, we use `\NameParser` to do that.

Changes to `\ifInHook` (default false) and `\ifFbox` (default true) are local to the scope in which the hook macros are called. `\ifFirstCap` must be set globally. Below we reproduce the logic of `\AltOff` before `\NameParser` in `\MainNameHook`:

```
\renewcommand*\CapThis{\FirstCaptrue}

\renewcommand*\NamesFormat[1]
  {\InHooktrue\NameParser\global\FirstCapfalse}

\renewcommand*\MainNameHook[1]
```

By using the same hook logic and `\noexpand`, the same macros used before now use a different "back end" without creating spurious index entries.

```
  {\Fboxfalse\InHooktrue\NameParser\global\FirstCapfalse}

\let\FrontNamesFormat\Namesformat
\let\FrontNameHook\MainNameHook

% 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}

\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 $\boxed{\text{de Smet}}$ 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.

We can go one step further and use our own "back-end" macros with other documents that were designed for the built-in macros used in alternate formatting, as the next example shows:

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

\makeindex

\newif\ifCaps      % Replaces \if@nameauth@DoAlt
\newif\ifFirstCap  % Replaces \if@nameauth@DoCaps
\newif\ifInHook    % Replaces \if@nameauth@InHook
\Capstrue          % Replaces \AltFormatActive

% Alternate formatting macro definition
\renewcommand*\textSC[1]{\ifCaps\textsc{#1}\else#1\fi}

% Redefinition of \AltCaps and \CapThis
\renewcommand*\AltCaps[1]{%
  \ifInHook
    \ifFirstCap\MakeUppercase{#1}\else#1\fi
  \else
    #1%
  \fi
}
\renewcommand*\CapThis{\FirstCaptrue}

\renewcommand*\NamesFormat[1]
  {\InHooktrue#1\global\FirstCapfalse}

\renewcommand*\MainNameHook[1]
  {\Capsfalse\InHooktrue#1\global\FirstCapfalse}

\let\FrontNamesFormat\Namesformat
\let\FrontNameHook\MainNameHook

\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}

\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.

13.3 Full Customization

One can redesign or augment the core naming macros, then hook those modifications into the nameauth package without needing to patch the style file itself.

\NameauthName
\NameauthLName
\NameauthFName

All these macros are set by default to `\@nameauth@Name`, the internal name parser. `\Name`, or an unmodified shorthand, calls `\NameauthName`. `\Name*`, or an L-shorthand, sets `\@nameauth@FullNametrue`, then calls `\NameauthLName`. `\FName`, or an S-shorthand, sets `\@nameauth@FirstNametrue`, then calls `\NameauthFName`. One should not modify `\Name` and `\FName` directly.

13.3.1 Names In Boxes

Since nameauth uses xparse, the next examples use it also. Here we look at decorationin 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.

```
\documentclass{article}
\input{compat.tex} % Included with nameauth; example file aids
% compatibility across different LaTeX versions and engines.
\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{}}{%
  \global\@nameauth@toksa\expandafter{#1}%
  \global\@nameauth@toksb\expandafter{#2}%
  \global\@nameauth@toksc\expandafter{#3}%
  \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{}}{%
  \global\@nameauth@toksa\expandafter{#1}%
  \global\@nameauth@toksb\expandafter{#2}%
  \global\@nameauth@toksc\expandafter{#3}%
  \fcolorbox{black}{green!25!white}{\@nameauth@Name[#1]{#2}[#3]}%
}

% Likewise change the macro when personal names are desired.
\NewDocumentCommand{\MyFName}{O{} m O{}}{%
  \global\@nameauth@toksa\expandafter{#1}%
  \global\@nameauth@toksb\expandafter{#2}%
  \global\@nameauth@toksc\expandafter{#3}%
  \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 that above.
\renewcommand*\NamesFormat[1]{\scshape#1}
\renewcommand*\MainNameHook[1]{#1}

% 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}
```

$\boxed{\text{Adolf Harnack}}$ was a theologian who stressed the Fatherhood of God and the brotherhood of man. $\boxed{\text{Harnack}}$ flourished in the early twentieth century; $\boxed{\text{Adolf von Harnack}}$; $\boxed{\text{Adolf}}$.

After the name is printed in the body text, the internal macros **globally** set `\@nameauth@FullNamefalse` and `\@nameauth@FirstNamefalse`, as well as other flags related to the prefix macros. This prevents certain cases of undocumented behavior in versions of `nameauth` before 3.3, where resetting flags locally could cause unexpected name forms. If an existing document leverages the local resetting of flags, one can use the `oldreset` option.

`\global` Like many of the macros in this package, these macros can be redefined or used locally within a scope without making global changes to the document unless you specifically use `\global`.

13.3.2 Change Parsing

In Section 4.3.1 we saw reasons why nameauth ignores spaces before final optional arguments. Here we show how to change that. With xparse since May 2018, the !O{} specifier makes spaces significant before an optional argument. Before then, the example below got the same result through undocumented behavior, now fixed.

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

\makeatletter
\ifdefined\nameauthxp
\NewDocumentCommand{\MyName}{O{} m !O{}}{%
  \global\@nameauth@toksa\expandafter{#1}%
  \global\@nameauth@toksb\expandafter{#2}%
  \global\@nameauth@toksc\expandafter{#3}%
  \@nameauth@Name[#1]{#2}[#3]%
}
\else
\NewDocumentCommand{\MyName}{O{} m O{}}{%
  \global\@nameauth@toksa\expandafter{#1}%
  \global\@nameauth@toksb\expandafter{#2}%
  \global\@nameauth@toksc\expandafter{#3}%
  \@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 [reportedly] said'',
 ``Miyazaki Hayao [supposedly] said'',
 ``Einstein [reportedly] said'', and
 ``Miyazaki [supposedly] said''.

 We get ``\Name[Albert]{Einstein} [reportedly] said'',
``\Name{Miyazaki, Hayao} [supposedly] said'',
``\Name[Albert]{Einstein} [reportedly] said'', and
``\Name{Miyazaki, Hayao} [supposedly] said''.
```

We want “Albert Einstein [reportedly] said”, “Miyazaki Hayao [supposedly] said”, “Einstein [reportedly] said”, and “Miyazaki [supposedly] said”.

We get “Albert Einstein [reportedly] said”, “Miyazaki Hayao [supposedly] said”, “Einstein [reportedly] said”, and “Miyazaki [supposedly] said”.

Back to Table of Contents

14 Technical Notes and Tips

14.1 Tips: General

- Technical details about package development and testing are in `README.md`.
- Both `compat.tex` and `examples.tex` contain useful information.
- Do not put naming macros withing a macro defined by `\edef`. Yet naming macros can be passed as arguments to such a macro.
- In `dtx` files, the `nameauth` environment and initial tagging macros work easiest in the `<driver>` section preamble. Yet if any of these contains one or more `\newif` statements, they must go in the "commented part" since the preamble falls between `\iffalse` and `\fi`.

14.2 Tips: Indexing and Sorting

- Two names may look alike on the page, but their name patterns can differ, creating spurious index entries. Check the `idx` file to be sure.
- To fix spurious entries, compare index entries with names in the text.
  - Check if naming macros always use the same arguments.
  - Check sorting tags (`\PretagName` (Section 7.6).
  - Check use of active Unicode characters (Section 14.4).
  - Use `\ShowPattern`, `\ShowIdxPageref`, and `\ShowNameState`.
  - Did `\noexpand` precede macros in name arguments?
- Check `nameauth` package warnings. Set the `verbose` option, which will offer a number of "informational" warnings that could be of assistance.
- If an entry is sorted incorrectly in the index, check the following:
  - Are there any active characters, spaces, or control sequences in the name arguments? Use `\PretagName`.
  - Is alternate formatting used consistently? Are names used within sections of alternate formatting used outside of them?
  - Are macros in the name arguments that can expand differently under different conditions preceded by `\noexpand`?

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*⟩`}`

14.3 Tips: Macros in Name Arguments

- A missing square bracket or curly brace in name arguments can cause errors like "`Paragraph ended`" and "`Missing` ⟨*grouping token*⟩ `inserted.`"
- Use alternate formatting to avoid potential problems, especially when using `\CapThis`, which segments arguments (Sections 11.3, 13).
- 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

\begin{itemize}
\item \ifdefined\CondAtrue \texttt{\textbackslash ifCondA} is defined
  \else \texttt{\textbackslash ifCondA} is not defined \fi
  in the outer scope (outer definition).

\item \ifdefined\MacroA \cmd{\MacroA} is defined
  \else \cmd{\MacroA} is not defined \fi
  in the outer scope (outer definition).

\item \ifdefined\CondBtrue \texttt{\textbackslash ifCondB} is defined
  \else \texttt{\textbackslash ifCondB} is not defined \fi
  in the outer scope (local definition).

\item \ifdefined\CondCtrue \texttt{\textbackslash ifCondC} is defined
  \else \texttt{\textbackslash ifCondC} is not defined \fi
  in the outer scope (no \cmd{\global}\cmd{\newif}).

\item \ifdefined\MacroB \cmd{\MacroB} is defined
  \else \cmd{\MacroB} is not defined \fi
  in the outer scope (no \cmd{\global}\cmd{\newcommand}).

\item \cmd{\MacroC} \ifdefined\MacroC is defined
  \else is not defined \fi
  in the outer scope (local definition).

```

```
\item \cmd{\MacroD} \ifdefined\MacroD is defined
  \else is not defined \fi
  in the outer scope (\cmd{\global} affects \cmd{\def} in \cmd{\MacroC}).

\item \ifCondA \texttt{\textbackslash ifCondA} is true
  \else \texttt{\textbackslash ifCondA} is false \fi
  (\cmd{\global} assignment works, not instantiation).
\end{itemize}
```

- `\ifCondA` is defined in the outer scope (outer definition).
- `\MacroA` is defined in the outer scope (outer definition).
- `\ifCondB` is not defined in the outer scope (local definition).
- `\ifCondC` is not defined in the outer scope (no `\global\newif`).
- `\MacroB` is not defined in the outer scope (no `\global\newcommand`).
- `\MacroC` is not defined in the outer scope (local definition).
- `\MacroD` is defined in the outer scope (`\global` affects `\def` in `\MacroC`).
- `\ifCondA` is true (`\global` assignment works, not instantiation).

Any macro that is used in the argument of a naming macro must be defined in all scopes in which that name is used. The names themselves are global, but macros in their arguments are not guaranteed to be so:

```
\begin{nameauth}
  \< Testi & & \noexpand\TESTi & >
  \< Testii & & \noexpand\TESTii & >
\end{nameauth}
\def\TESTi{Test One}
\indent \hbox to 10em{(Outer 1) \Testi\hfill}
\begingroup
  (Inner 1) \Testi\\
  \def\TESTii{Test Two}
  \hbox to 10em{(Inner 2) \Testii\hfill}
\endgroup
(Outer 2) \unless\ifdefined\TESTii \cmd{\TESTii} undefined\fi
```

(Outer 1) Test One   (Inner 1) Test One
(Inner 2) Test Two   (Outer 2) `\TESTii` undefined

14.4 Active Unicode Characters

14.4.1 General Information

Both `xelatex` and `lualatex` support Unicode natively. In `pdflatex`, this happens with active characters. There are features (e.g., in microtype) that require `pdflatex`. Both `makeindex` and `xindy` have their own design choices.

With `\usepackage[T1]{fontenc}`, `latex` and `pdflatex` can use many active Unicode characters automatically. Use `\PretagName` to sort names with these characters (Section 7.6). 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.

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

```
\usepackage[utf8]{inputenc} % For older TL releases
\usepackage[TS1,T1]{fontenc}
\usepackage{lmodern}% Contains TS1 glyph 115
\usepackage{newunicodechar}
\DeclareTextSymbolDefault{\textlongs}{TS1}
\DeclareTextSymbol{\textlongs}{TS1}{115}
\newunicodechar{ſ}{\textlongs}

In Congreſs, July 4, 1776
```

In Congreſs, July 4, 1776

Below we group characters that are supported in `pdflatex` without additional modification by accents and diacritical marks:

Capitals	
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[31]	Ä · · Ë · · Ï · · · · Ö · · · Ü · Ÿ ·
cedilla	· Ç · · Ģ · · · Ķ Ļ Ņ · Ŗ Ş Ţ · · · ·
macron	Ā · · Ē Ḡ · Ī · · · · Ō · · · Ū Ǣ 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 Ð (eth) Đ (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 ǣ ȳ ·
breve	ă · · · ğ · ĭ · · · · ŏ · · · ŭ · · ·
dot/dotless	ḃ ċ · ė ġ · ı · · · · · · · · · · · 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

[31] 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”.

14.4.2 Compatibility: Old and New

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 a number of problems.

There are two ways to test if one is working with a newer or older version of `pdflatex` or `latex`. The first involves LaTeX kernel macros in the document preamble to check the format date. We check if the date is later than the first use of the file `utf8-2018.def`, If that is the case, we can define macros or set Boolean flags, which we can test repeatedly. For example, in the document preamble we could have:

```
\makeatletter
  \@ifl@t@r\fmtversion{2018/10/05}{\def\nameauthltx{}}{}
  \@ifl@t@r\fmtversion{2018/04/30}{\def\nameauthxp{}}{}
\makeatother
```

The tests above apply to two things that are pertinent to nameauth. The first is the presence of `utf8-2018.def`. The second involves changes in xparse that could impact nameauth, which we used silently in Section 13.3.2. Another test checks directly for the presence of `utf8-2018.def`:

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

Now we apply the test to an example. Before 2018, some index styles like `gind.ist` could not work with characters that contained macrons. Since 2018, those restrictions have been removed. To create a document that can work with old or new versions of `pdflatex` and `latex`, one can choose to use macrons or not. In a recent version, one will see macrons in the following names, otherwise, no macron will be present.

```
\ifdefined\nameauthltx \Name{Ghazāli} \else \Name{Ghazali}\fi

\IfFileExists{utf8-2018.def}{\Name{Ghazāli}}{\Name{Ghazali}}
```

Ghazāli

Ghazāli

The challenge of compatibility arises in this manual in a few instances, which we summarize below to mention specific resources without going too far afield.

- To use older distributions, one must include the textcomp package for backward compatibility. Otherwise it is not needed in recent TeX distributions, e.g., since 2018.
- Use of certain text elements, such as `\dotfill` within tables, has become more permissive in recent TeX distributions.

- The file `examples.tex` (see `README.md`) includes additional code snippets that deal with compatibility.
- We `\input` the included snippet `compat.tex` in `examples.tex` to permit use with different LaTeX engines and older TeX distributions.

Users are encouraged to view and modify `compat.tex` to meet their own needs.

14.4.3 Fragility of Active Unicode

TeX macros that partition their arguments can break active Unicode characters.The simple macro `\def\foo#1#2#3!{<#1#2><#3>}` takes three arguments, groups the first two, then the third, followed by a delimiter that ends the argument list:

Arg.	Macro	Engine	Result
`abc`	`\foo abc!`	(any)	<ab><c>
`{æ}bc`	`\foo {æ}bc!`	(any)	<æb><c>
`\ae bc`	`\foo \ae bc!`	(any)	<æb><c>
`æbc`	`\foo æbc!`	`xelatex`	<æb><c>
`æbc`	`\foo æbc!`	`lualatex`	<æb><c>
`æbc`	`\foo æbc!`	`pdflatex`	<æ><bc>
`æbc`	`\foo æbc!`	`latex`	<æ><bc>

The letter `a` is one argument. Since `{æ}` is in a group, it is one argument. The macro `\ae` also is one argument. Both `xelatex` and `lualatex` likewise treat the Unicode letter `æ` as one argument. Thus, in all these cases, the first two glyphs are grouped together in `#1#2` and `c` is left by itself in `#3`.

In `latex` and `pdflatex`, however, `æ` is an active Unicode control sequence that uses two arguments all by itself: `#1#2`. The rest of the input, `bc`, is in `#3`. This is not intuitive. Any macro where this `#1#2` pair is divided into `#1` and `#2` will produce one of two errors: `Unicode char ...not set up for LaTeX` or `Argument of \UTFviii@two@octets has an extra }`.

Starting on page 142 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, we use alternate formatting and `\AltCaps` (Section 11.3.4).

Back to Table of Contents

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)

15 Implementation

This package has been reorganized so that the manual and the package have substantially the same ordering. This repetition should aid understanding how the components work.

15.1 Boolean Flags

The nameauth package is a parser. The flags in this section show and change the state of that parser.

15.1.1 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 it 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
```

15.1.2 Name Grammar and Syntax

Name Types

`\ifNameauthWestern`
`\ifNameauthObsolete`

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 nonwestern name. The second shows the kind of nonwestern syntax used. These are not reset after evaluation.

```
\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. The first is global. The second is 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 nonwestern surnames to be displayed when a shorter form is used. The default is to reset both globally on a per-name basis.

`\if@nameauth@ShortSNN` is used with `\DropAffix` to suppress the affix of a Western name. `\if@nameauth@EastFN` toggles the forced printing of Eastern forenames.

```
\newif\if@nameauth@FullName
\newif\if@nameauth@FirstName
\newif\if@nameauth@AltAKA
\newif\if@nameauth@ShortSNN
\newif\if@nameauth@EastFN
```

15.1.3 Debugging

When used with the index debugging macros, show complete index entries if true, otherwise show simple entries.

```
\newif\if@nameauth@LongIdxDebug
```

15.1.4 Indexing

Toggle Indexing

The indexing flags permit or prevent indexing and tags. `\IndexActive` and `\IndexInctive` or the `index` and `noindex` options toggle the first flag; `\SkipIndex` toggles the second.

```
\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 Warnings

Control the number of warnings concerning the index; default is terse.

```
\newif\if@nameauth@Verbose
```

Cross-References

Tell the index entry formatter to create a cross-reference.

```
\newif\if@nameauth@Xref
```

Determine whether `\IndexRef` creates a *see* reference or a *see also* reference.

```
\newif\if@nameauth@SeeAlso
```

15.1.5 Formatting and Name Control Sequences

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
```

15.1.6 Name Decisions

Creating and Destroying Name Patterns

Restrict the creation and destruction of name patters to the current name system if true.

```
\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 not apply local scope, retaining the current scope.

```
\newif\if@nameauth@GlobalScope
```

15.1.7 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 three flags are used only for backward compatibility. The first broadly determines how per-name flags are reset. The second affects the behavior of `\JustIndex`. 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 where user customizations were made.

```
\newif\if@nameauth@OldReset
\newif\if@nameauth@OldPass
\newif\if@nameauth@OldToks
\newif\if@nameauth@OldArgs
```

15.2 Token Registers, Hooks, and Internal Values

15.2.1 Name Argument Token Registers

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

These three token registers contain the current values of the name arguments passed to `\Name`, its variants, and the cross-reference arguments of `\AKA`. Users can access them especially in formatting hooks.

```
\newtoks\@nameauth@toksa
\newtoks\@nameauth@toksb
\newtoks\@nameauth@toksc
```

These three token registers contain the current values of the name arguments in each line of the `nameauth` environment, thus, `@nameauth@e` for that environment.

```
\newtoks\@nameauth@etoksb
\newtoks\@nameauth@etoksc
\newtoks\@nameauth@etoksd
```

15.2.2 Hooks

`\NamesFormat` Post-process "first" instance of final complete name form in text. See Sections 9.1 and 11.1. Called when both `\@nameauth@MainFormat` and `\@nameauth@FirstFormat` are true.

```
\newcommand*\NamesFormat{}
```

`\MainNameHook` Post-process subsequent instance of final complete name form in main-matter text. See Sections 9.1 and 11.1f. Called when `\@nameauth@MainFormat` is true and the Boolean flag `\@nameauth@FirstFormat` is false.

```
\newcommand*\MainNameHook{}
```

`\FrontNamesFormat` Post-process "first" instance of final complete name form in front-matter text. Called when `\@nameauth@MainFormat` is false and `\@nameauth@FirstFormat` is true.

```
\newcommand*\FrontNamesFormat{}
```

`\FrontNameHook` Post-process subsequent instance of final complete name form in front-matter text. Called when `\@nameauth@MainFormat` is false and `\@nameauth@FirstFormat` is false.

```
\newcommand*\FrontNameHook{}
```

The following three macros usually point to the core name engine, `\@nameauth@Name`. They allow users to customize the naming macros in the fullest sense. See Section 13.3.

`\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` Gives access to the current name control pattern. This hook can be used, e.g., in formatting hooks to recall a name tag (Section 11.2). We preset it to an empty value. With every call to a macro that takes name arguments (Section 1.6.1), this hook is updated.

```
\let\NameauthPattern\@empty
```

15.2.3 Internal Values

`\@nameauth@Actual` This sets the “actual” character used by nameauth for index sorting. This lets one use, for example, `\global\IndexActual{=}`.

```
\def\@nameauth@Actual{@}
```

`\@nameauth@Exclude` This makes an xref into an “exclusion”. An exclusion is any name control sequence ending in `!PN` that expands to this value. See `\ExcludeName`.

```
\newcommand*\@nameauth@Exclude{!}
```

`\@nameauth@space` This macro provides a consistent space character for index entries.

```
\def\@nameauth@space{ }
```

15.3 Package Options

15.3.1 Name Grammar and Syntax

Change the way that names are displayed, specifically 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}
```

15.3.2 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}
```

15.3.3 Formatting

Start off in a different naming regime or change formatting behavior in general or for `\AKA`.

```
\DeclareOption{mainmatter}{\@nameauth@MainFormattrue}
\DeclareOption{frontmatter}{\@nameauth@MainFormatfalse}
\DeclareOption{alwaysformat}{\@nameauth@AlwaysFormattrue}
\DeclareOption{formatAKA}{\@nameauth@AKAFormattrue}
```

15.3.4 Predefined Formatting Hooks

```
\DeclareOption{noformat}{\renewcommand*\NamesFormat{}}
\DeclareOption{smallcaps}{\renewcommand*\NamesFormat{\scshape}}
\DeclareOption{italic}{\renewcommand*\NamesFormat{\itshape}}
\DeclareOption{boldface}{\renewcommand*\NamesFormat{\bfseries}}
```

15.3.5 Alternate Formatting

Enable alternate formatting.

```
\DeclareOption{altformat}{%
  \@nameauth@AltFormattrue\@nameauth@DoAlttrue}
```

15.3.6 Scope

Name test paths are either local or the same scope in which they are called.

```
\DeclareOption{globaltest}{\@nameauth@GlobalScopetrue}
```

15.3.7 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}
```

15.4 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
```

15.5 Internal Macros

15.5.1 Fundamental Macros

The following macros are the most essential to the concept of "name".

Name Control Sequence: Who Am I?

\@nameauth@Clean Thanks to Heiko Oberdiek, this macro produces a "sanitized" string to make a control sequence for a name. Testing the existence of that control sequence is the core of nameauth.

```
\newcommand*\@nameauth@Clean[1]
  {\expandafter\zap@space\detokenize{#1} \@empty}
```

\@nameauth@MakeCS Unless we are in \AKA, create a name control sequence in the core name engine.

```
\newcommand*\@nameauth@MakeCS[1]
{%
  \unless\ifcsname#1\endcsname
    \unless\if@nameauth@InAKA\csgdef{#1}{}\fi
  \fi
}
```

Parsing: Root and Suffix

\@nameauth@Root These two macros return everything before a comma in ⟨*SNN*⟩. We do this with a delimited macro as a helper that determines the root, the suffix, and the end of input.

```
\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 These two macros return everything before a vertical bar (|) in an index tag (for sorting xrefs). We do this with a delimited macro as a helper, as above.

```
\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 These two macros parse ⟨*SNN*⟩ into a radix and a comma-delimited suffix, returning only the suffix after a comma in the argument, or nothing. We do this with a delimited macro as a helper, but more complicated this time.

```
\newcommand*\@nameauth@Suffix[1]{\@nameauth@@Suffix#1,,\\}
```

\@nameauth@@Suffix Throw out the radix; return the suffix with no leading spaces. Used to print the suffix.

```
\def\@nameauth@@Suffix#1,#2,#3\\{%
  \ifx\\#2\\\@empty\else\trim@spaces{#2}\fi
}
```

\@nameauth@GetSuff These two macros test the suffix for a leading active Unicode character. We use this for capitalization to avoid errors.

```
\newcommand*\@nameauth@GetSuff[1]{\@nameauth@@GetSuff#1,,\\}
```

`\@nameauth@@GetSuff` Throw out the radix; return the suffix.

```
\def\@nameauth@@GetSuff#1,#2,#3\\{#2}
```

Parsing: Capitalization

`\@nameauth@TestToks` Test if the leading token is the same as the leading token of an active Unicode character, using an *Esszett* (ß) as the control. We only run this macro if we are in the inputenc regime (using `pdflatex` and `latex`). Otherwise we use native Unicode.

```
\newcommand*\@nameauth@TestToks[1]
{%
  \toks@\expandafter{\@car#1\@nil}%
  \edef\@nameauth@one{\the\toks@}%
  \toks@\expandafter{\@carß\@nil}%
  \edef\@nameauth@two{\the\toks@}%
  \ifx\@nameauth@one\@nameauth@two
    \@nameauth@UTFtrue%
  \else
    \@nameauth@UTFfalse%
  \fi
}
```

`\@nameauth@UTFtest` We choose how to capitalize a letter by determining if we are using native Unicode (`xelatex` or `lualatex`). We test for `\Umathchar`. Then we see if inputenc is loaded. We set up the comparison and pass off to `\@nameauth@TestToks`.

```
\newcommand*\@nameauth@UTFtest[1]
{%
  \def\@nameauth@testarg{#1}%
  \ifdefined\Umathchar
    \@nameauth@UTFfalse%
  \else
    \ifdefined\UTFviii@two@octets
      \if@nameauth@Accent
        \@nameauth@UTFtrue\@nameauth@Accentfalse%
      \else
        \expandafter\@nameauth@TestToks
          \expandafter{\@nameauth@testarg}%
      \fi
    \else
      \@nameauth@UTFfalse%
    \fi
  \fi
}
```

`\@nameauth@UTFtestS` This test is like the one above, but a special case when we have a suffix. We have to do a bit more in the way of expansion to get the comparison to work properly. Moreover, we only use this when the regular suffix macro is not `\@empty`.

```
\newcommand*\@nameauth@UTFtestS[1]
{%
  \expandafter\def\expandafter\@nameauth@testarg%
    \expandafter{\@nameauth@GetSuff{#1}}%
```

This following token register assignment looks weird, but it is how we get a test that works.

```
\expandafter\toks@%
  \expandafter\expandafter\expandafter{\@nameauth@testarg}%
```

We take that token register and assign its value to a macro to do the test.

```
\expandafter\def\expandafter\@nameauth@test@rg%
  \expandafter{\the\toks@}%
\ifdefined\Umathchar
  \@nameauth@UTFfalse%
\else
  \ifdefined\UTFviii@two@octets
    \if@nameauth@Accent
      \@nameauth@UTFtrue\@nameauth@Accentfalse%
    \else
      \expandafter\@nameauth@TestToks%
        \expandafter{\@nameauth@test@rg}%
    \fi
  \else
    \@nameauth@UTFfalse%
  \fi
\fi
}
```

`\@nameauth@Cap` These two macros cap the first letter of the argument. Since they partition the argument into two segments, this can break some macro arguments unless one uses `\noexpand`.

```
\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` These two macros cap the first active Unicode letter when one is using inputenc (an argument "twice as wide" as normal, native Unicode).

```
\newcommand*\@nameauth@CapUTF[1]{\@nameauth@C@pUTF#1\\}
```

`\@nameauth@C@pUTF` Helper macro for the one above.

```
\def\@nameauth@C@pUTF#1#2#3\\{%
  \expandafter\trim@spaces\expandafter{\MakeUppercase{#1#2}#3}%
}
```

`\@nameauth@CapArgs` Capitalize the first letter of all name arguments. Implements capitalization on demand in the body text (not the index) when not in alternate formatting. We only use this macro in the local scope of `\@nameauth@Parse`. Uses the foregoing macros.

```
\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, non-native Unicode characters, then cap the first letter.

```
      \unless\ifx\arga\@empty
        \def\test{#1}%
        \expandafter\@nameauth@UTFtest\expandafter{\test}%
```

Capitalize the first active Unicode character.

```
        \if@nameauth@UTF
          \expandafter\def\expandafter\carga\expandafter{%
            \expandafter\@nameauth@CapUTF\expandafter{\test}}%
```

Capitalize the first native Unicode character (not active).

```
        \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 character (not active).

```
      \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 (not active).

```
        \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 (not active).

```
        \else
          \expandafter\def\expandafter\cargc\expandafter{%
            \expandafter\@nameauth@Cap\expandafter{\test}}%
        \fi
      \fi
```

Let the local arguments be the macros with caps. We cap them all and let the macros sort them out because we do not know which will be displayed.

```
      \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` below. One cannot use `\unless` below.

```
  \def\@nameauth@TestPunct##1%
  {%
    \ifx\@nameauth@TestPunct##1\@nameauth@TestPunct
    \else
      \global\@nameauth@Puncttrue%
    \fi
  }%
  \global\@nameauth@Punctfalse%
  \@nameauth@TestD@t#1\@End.\@End\\%
}
```

`\@nameauth@CheckDot` We assume that `\expandafter` precedes the invocation of `\@nameauth@CheckDot`, which only is called if the terminal character of the input is a period. We evaluate the lookahead `\@nameauth@token` while keeping it on the list of input tokens.

```
\newcommand*\@nameauth@CheckDot
  {\futurelet\@nameauth@token\@nameauth@EvalDot}
```

`\@nameauth@EvalDot` If `\@nameauth@token`, the lookahead, is a full stop, we gobble the next token because it is that full stop.

```
\newcommand*\@nameauth@EvalDot
{%
  \let\@nameauth@stop=.%
  \ifx\@nameauth@token\@nameauth@stop
    \expandafter\@gobble \fi
}
```

Parsing: Breaking, Spaces, and Commas

`\@nameauth@AddPunct` Here we govern whether (in the text, not the index) spaces between name elements break or not, and whether to add commas or not. Much applies only to Western names, thus we check if ⟨*FNN*⟩ is empty or not. 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 have a set of comma-use conventions that differ from all other name forms, so we only use the following logic if ⟨*FNN*⟩ is not empty, thus, a Western name.

```
    \unless\ifx\arga\@empty
      \if@nameauth@AlwaysComma
        \def\Space{, }%
        \if@nameauth@NBSP\edef\Space{,\nobreakspace}\fi
      \fi
      \if@nameauth@ShowComma
        \def\Space{, }%
        \if@nameauth@NBSP\edef\Space{,\nobreakspace}\fi
      \fi
      \if@nameauth@NoComma
        \def\Space{ }%
        \if@nameauth@NBSP\edef\Space{\nobreakspace}\fi
      \fi
    \fi
  \fi
}
```

Parsing: Name Argument Loading

`\@nameauth@LoadArgs` Assign name arguments to internal macros to determine name syntax. This is used in all macros that take name arguments.

```
\newcommand*\@nameauth@LoadArgs[3]
{%
```

We want these arguments to expand to `\@empty` (or not) when we test them.

```
  \protected@edef\@nameauth@A{\trim@spaces{#1}}%
  \protected@edef\@nameauth@B{\@nameauth@Root{#2}}%
  \protected@edef\@nameauth@SB{\@nameauth@Suffix{#2}}%
  \protected@edef\@nameauth@C{\trim@spaces{#3}}%
```

Make (usually) unique control sequence values from the name arguments.

```
  \def\@nameauth@csb{\@nameauth@Clean{#2}}%
  \def\@nameauth@csbc{\@nameauth@Clean{#2,#3}}%
  \def\@nameauth@csab{\@nameauth@Clean{#1!#2}}%
}
```

Parsing: Standard Parsing Logic

`\@nameauth@Choice` This standard logic applies to all macros that take name arguments. Here we update `\NameauthPattern`, `\ifNameauthWestern`, and `\ifNameauthObsolete` to show the resulting name pattern and type of name, usable in formatting hooks.

```
\newcommand\@nameauth@Choice[3]
{%
  \ifx\@nameauth@A\@empty
    \ifx\@nameauth@C\@empty
```

This path is for nonwestern 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 ifit exists. With indexing and other macros, one ignores `\@nameauth@C`.

```
      \let\NameauthPattern\@nameauth@csb%
      \NameauthWesternfalse \NameauthObsoletefalse%
      #1%
    \else
      \ifx\@nameauth@SB\@empty
```

The `#2` argument is only for nonwestern 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.

```
        \let\NameauthPattern\@nameauth@csbc%
        \NameauthWesternfalse \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.

```
        \let\NameauthPattern\@nameauth@csb%
        \NameauthWesternfalse \NameauthObsoletefalse%
        #1%
      \fi
    \fi
  \else
```

This decision 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`. One also must check for and handle `\@nameauth@SB`. Otherwise, for indexing and control sequences, one ignores `\@nameauth@C` in this path and handles `\@nameauth@SB` appropriately.

```
    \let\NameauthPattern\@nameauth@csab%
    \NameauthWesterntrue \NameauthObsoletefalse%
    #3%
  \fi
}
```

`\@nameauth@Flags` Reset flags after the naming macros and `\AKA` and friends create output in the text. Other places in the core naming engine where flags are reset are for special cases like `\JustIndex`.

```
\newcommand*\@nameauth@Flags
{%
  \if@nameauth@OldReset
```

The `oldreset` option implies not only a difference in scope regarding how flags are reset, but it also lets the effects of `\ForgetThis` and `\SubvertThis` to pass through `\AKA` and `\AKA*`. Regardless, we only reset `\if@nameauth@AltAKA` here due to macros like `\PName`.

```
    \if@nameauth@InAKA\@nameauth@AltAKAfalse\fi
    \@nameauth@SkipIndexfalse%
    \if@nameauth@InName
      \@nameauth@Forgetfalse%
      \@nameauth@Subvertfalse%
    \fi
    \@nameauth@NBSPfalse%
    \@nameauth@NBSPXfalse%
    \@nameauth@DoCapsfalse%
    \@nameauth@Accentfalse%
    \@nameauth@AllThisfalse%
    \@nameauth@ShowCommafalse%
    \@nameauth@NoCommafalse%
    \@nameauth@RevThisfalse%
    \@nameauth@RevThisCommafalse%
    \@nameauth@ShortSNNfalse%
    \@nameauth@EastFNfalse%
  \else
```

The current way that the flags are reset makes them both global and more uniform, hopefully eliminating a few chances for errors that might be quite difficult to debug.

```
    \if@nameauth@InAKA\global\@nameauth@AltAKAfalse\fi
    \global\@nameauth@SkipIndexfalse%
    \global\@nameauth@Forgetfalse%
    \global\@nameauth@Subvertfalse%
    \global\@nameauth@NBSPfalse%
    \global\@nameauth@NBSPXfalse%
    \global\@nameauth@DoCapsfalse%
    \global\@nameauth@Accentfalse%
    \global\@nameauth@AllThisfalse%
    \global\@nameauth@ShowCommafalse%
    \global\@nameauth@NoCommafalse%
    \global\@nameauth@RevThisfalse%
    \global\@nameauth@RevThisCommafalse%
    \global\@nameauth@ShortSNNfalse%
    \global\@nameauth@EastFNfalse%
  \fi
}
```

15.5.2 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, has an empty root in a malformed 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%
}
```

15.5.3 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 naming system we are in (Section 6.1). Both \@nameauth@Name and \AKA call this parser, which only works in a locked state.

```
\newcommand\@nameauth@Parse[4]
{%
  \if@nameauth@BigLock \@nameauth@Lockfalse\fi
  \if@nameauth@Lock
```

Make token register copies of the current name args to be available for the hook macros.

```
    \if@nameauth@OldToks
      \@nameauth@toksa\expandafter{#1}%
      \@nameauth@toksb\expandafter{#2}%
      \@nameauth@toksc\expandafter{#3}%
    \else
      \global\@nameauth@toksa\expandafter{#1}%
      \global\@nameauth@toksb\expandafter{#2}%
      \global\@nameauth@toksc\expandafter{#3}%
    \fi
```

If global caps. reversing, and commas are true, set the per-name flags true.

```
      \if@nameauth@AllCaps     \@nameauth@AllThistrue\fi
      \if@nameauth@RevAll      \@nameauth@RevThistrue\fi
      \if@nameauth@RevAllComma \@nameauth@RevThisCommatrue\fi
```

Now we enter a local scope where we can use simple control strings without needing to worry about collisions. We process and load the arguments into the appropriate macros.

```
    \begingroup%
      \def\@nameauth@InParser{}%
      \@nameauth@LoadArgs{#1}{#2}{#3}%
```

Copy the protected control sequences to local, unprotected ones.

```
      \let\arga\@nameauth@A%
      \let\rootb\@nameauth@B%
      \let\suffb\@nameauth@SB%
      \let\argc\@nameauth@C%
```

Capitalization on demand in the body text if not in alternate formatting.

```
      \if@nameauth@DoCaps
        \@nameauth@CapArgs{#1}{#2}{#3}%
      \fi
```

We capitalize the entire surname when desired; different from above and overrides it.

```
    \if@nameauth@AllThis
      \protected@edef\rootb%
        {\MakeUppercase{\@nameauth@Root{#2}}}%
    \fi
```

Use non-breaking spaces and commas as desired.

```
    \@nameauth@AddPunct%
```

We parse names by attaching "meaning" to patterns of macro arguments primarily via `\FNN` and `\SNN`. Then we call the name printing macros, based on optional arguments.

```
    \let\SNN\rootb%
    \@nameauth@Choice
```

Nonwestern 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}%
    }%
```

Nonwestern names, obsolete syntax. Here `\argc` is significant.

```
    {%
      \let\FNN\argc%
      \@nameauth@NonWest{\@nameauth@csbc#4}%
      \@nameauth@MakeCS{\@nameauth@csbc#4}%
    }%
```

Western names. We test for `\argc` and swap it for `\arga` and account for `\suffb`.

```
    {%
      \ifx\argc\@empty
        \let\FNN\arga%
      \else
        \let\FNN\argc%
      \fi
      \unless\ifx\suffb\@empty
        \def\SNN{\rootb\Space\suffb}%
        \if@nameauth@ShortSNN
          \let\SNN\rootb%
        \fi
      \fi
      \@nameauth@West{\@nameauth@csab#4}%
      \@nameauth@MakeCS{\@nameauth@csab#4}%
    }%
```

We end the local group and reset the flags for name forms here.

```
  \endgroup%
  \if@nameauth@OldReset
    \@nameauth@FullNamefalse%
    \@nameauth@FirstNamefalse%
    \@nameauth@FirstFormatfalse%
    \else
      \global\@nameauth@FullNamefalse%
      \global\@nameauth@FirstNamefalse%
      \global\@nameauth@FirstFormatfalse%
    \fi
  \fi
}
```

Name Display Layer

\@nameauth@NonWest Arrange forms of nonwestern names. We inherit macros from the parser and only use this macro in the local scope of the parser.

```
\newcommand*\@nameauth@NonWest[1]
{%
  \ifdefined\@nameauth@InParser
    \@nameauth@Form{#1}%
    \ifx\FNN\@empty
      \@nameauth@Hook{\SNN}%
    \else
      \if@nameauth@FullName
        \if@nameauth@RevThis
          \@nameauth@Hook{\FNN\Space\SNN}%
        \else
          \@nameauth@Hook{\SNN\Space\FNN}%
        \fi
      \else
        \if@nameauth@FirstName
          \if@nameauth@EastFN
            \@nameauth@Hook{\FNN}%
          \else
            \@nameauth@Hook{\SNN}%
          \fi
        \else
          \@nameauth@Hook{\SNN}%
        \fi
      \fi
    \fi
  \fi
}
```

\@nameauth@West Arrange forms of Western names and "non-native" Eastern names. We inherit macros from the parser and only use this macro in the local scope of the parser.

```
\newcommand*\@nameauth@West[1]
{%
  \ifdefined\@nameauth@InParser
    \@nameauth@Form{#1}%
    \edef\RevSpace{,\SpaceW}%
    \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
      \if@nameauth@FirstName
        \@nameauth@Hook{\FNN}%
      \else
        \@nameauth@Hook{\rootb}%
      \fi
    \fi
  \fi
}
```

\@nameauth@Form Set up the flags per the formatting rules for first, subsequent, long, and short uses. We only use this macro in the local scope of the parser.

```
\newcommand*\@nameauth@Form[1]
{%
  \ifdefined\@nameauth@InParser
```

If the name does not exist yet or if the `alwaysformat` option is used, force first-use formatting, force a long name, and inhibit a short name.

```
    \unless\ifcsname#1\endcsname
      \@nameauth@FirstFormattrue%
      \@nameauth@FullNametrue%
      \@nameauth@FirstNamefalse%
    \else
      \if@nameauth@AlwaysFormat\@nameauth@FirstFormattrue\fi
    \fi
```

If we are not in `\AKA`, if a short name form is desired, inhibit a long form.

```
    \unless\if@nameauth@InAKA
      \if@nameauth@FirstName\@nameauth@FullNamefalse\fi
    \else
```

If we are in `\AKA` use special formatting rules. `\AKA*` acts like `\FName`, while `\AKA` acts like `\Name*`. Both prefer using the subsequent-use hooks unless the `formatAKA` option or the `alwaysformat` option are used.

```
      \if@nameauth@AltAKA
        \if@nameauth@OldAKA\@nameauth@EastFNtrue\fi
        \@nameauth@FullNamefalse%
        \@nameauth@FirstNametrue%
      \else
        \@nameauth@FullNametrue%
        \@nameauth@FirstNamefalse%
      \fi
      \unless\if@nameauth@AlwaysFormat
        \unless\if@nameauth@AKAFormat
          \@nameauth@FirstFormatfalse%
        \fi
      \fi
    \fi
  \fi
}
```

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`.

```
    \@nameauth@InHooktrue%
    \protected@edef\test{#1}%
    \expandafter\@nameauth@TestDot\expandafter{\test}%
    \if@nameauth@MainFormat
```

We use the formatting hooks for the main-matter system.

```
      \if@nameauth@FirstFormat
        \bgroup\NamesFormat{#1}\egroup%
      \else
        \bgroup\MainNameHook{#1}\egroup%
      \fi
    \else
```

We use the formatting hooks for the front-matter system.

```
      \if@nameauth@FirstFormat
        \bgroup\FrontNamesFormat{#1}\egroup%
      \else
        \bgroup\FrontNameHook{#1}\egroup%
      \fi
    \fi
  \fi
}
```

15.5.4 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[2]
{%
  \if@nameauth@DoIndex
```

If an index tag exists for the entry, get it. Also create a short version of the tag without any vertical bar or trailing macro. If we are creating a cross-reference, use the short tag, otherwise use the long tag.

```
    \ifcsname#1!TAG\endcsname
      \protected@edef\@nameauth@Tag{\csname#1!TAG\endcsname}%
      \expandafter\def\expandafter\@nameauth@ShortTag\expandafter{%
        \expandafter\@nameauth@TrimTag\expandafter{\@nameauth@Tag}}%
```

Create entries with a sorting tag and an info tag.

```
      \ifcsname#1!PRE\endcsname
        \protected@edef\@nameauth@Pre{\csname#1!PRE\endcsname}%
        \if@nameauth@Xref
          \protected@edef\@nameauth@IdxEntry
          {\@nameauth@Pre#2\@nameauth@ShortTag}%
        \else
          \protected@edef\@nameauth@IdxEntry
            {\@nameauth@Pre#2\@nameauth@Tag}%
        \fi
      \else
```

Create entries with just an info tag.

```
        \if@nameauth@Xref
          \protected@edef\@nameauth@IdxEntry
            {#2\@nameauth@ShortTag}%
        \else
          \protected@edef\@nameauth@IdxEntry
            {#2\@nameauth@Tag}%
        \fi
      \fi
    \else
```

Create entries with just a sorting tag.

```
      \ifcsname#1!PRE\endcsname
        \protected@edef\@nameauth@Pre{\csname#1!PRE\endcsname}%
        \protected@edef\@nameauth@IdxEntry{\@nameauth@Pre#2}%
      \else
        \protected@edef\@nameauth@IdxEntry{#2}%
      \fi
    \fi
```

Create entries with no tag.

```
    \expandafter\NameauthIndex\expandafter{\@nameauth@IdxEntry}%
  \fi
}
```

15.6 For Users: Prefix Macros

All prefix macros are meant to precede a particular name and only affect a particular name.

15.6.1 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. This is a fall-back in case the automatic test for active Unicode characters does not work.

```
\newcommand*\AccentCapThis
  {\@nameauth@Accenttrue\@nameauth@DoCapstrue}
```

`\CapName` Capitalize entire ⟨*SNN*⟩. Overrides `\CapThis` for surnames.

```
\newcommand*\CapName{\@nameauth@AllThistrue}
```

Reversing

`\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

`\DropAffix` Suppress the affix in a long Western name.

```
\newcommand*\DropAffix{\@nameauth@ShortSNNtrue}
```

`\KeepAffix` Trigger a name-suffix pair to be separated by a non-breaking space.

```
\newcommand*\KeepAffix{\@nameauth@NBSPtrue}
```

`\KeepName` Use non-breaking spaces between name syntactic forms.

```
\newcommand*\KeepName
  {\@nameauth@NBSPtrue\@nameauth@NBSPXtrue}
```

15.6.2 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}
```

15.6.3 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.

```
670 \newcommand*\ForceName{\@nameauth@FirstFormattrue}
```

`\ForgetThis` Have the naming engine `\@nameauth@Name` call `\ForgetName` internally.

```
671 \newcommand*\ForgetThis{\@nameauth@Forgettrue}
```

`\SubvertThis` Have the naming engine `\@nameauth@Name` call `\SubvertName` internally.

```
672 \newcommand*\SubvertThis{\@nameauth@Subverttrue}
```

15.7 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. That is why they usually come in pairs.

15.7.1 Name Syntax

Capitalization

`\AllCapsInactive` Turn off global surname capitalization.

```
673 \newcommand*\AllCapsInactive{\@nameauth@AllCapsfalse}
```

`\AllCapsActive` Turn on global surname capitalization. Activates `\CapName` for every name.

```
674 \newcommand*\AllCapsActive{\@nameauth@AllCapstrue}
```

Reversing

`\ReverseInactive` Turn off global name reversing.

```
675 \newcommand*\ReverseInactive{\@nameauth@RevAllfalse}
```

`\ReverseActive` Turn on global name reversing. Activates `\RevName` for every name.

```
676 \newcommand*\ReverseActive{\@nameauth@RevAlltrue}
```

Reversing with Commas

`\ReverseCommaInactive` Turn off global “last-name-comma-first”.

```
677 \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 due to the structure of the parser.

```
678 \newcommand*\ReverseCommaActive{\@nameauth@RevAllCommatrue}
```

15.7.2 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}
```

15.7.3 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}
```

15.7.4 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}
```

`\AltOn` Locally turn on alternate formatting.

```
\newcommand*\AltOn
{%
  \if@nameauth@InHook
    \if@nameauth@AltFormat\@nameauth@DoAlttrue\fi
  \fi
}
```

`\AltOff` Locally turn off alternate formatting.

```
\newcommand*\AltOff
{%
  \if@nameauth@InHook
    \if@nameauth@AltFormat\@nameauth@DoAltfalse\fi
  \fi
}
```

`\AltCaps` Alternate discretionary capping macro triggered by `\CapThis`.

```
\newcommand*\AltCaps[1]
{%
  \if@nameauth@InHook
    \if@nameauth@DoCaps\MakeUppercase{#1}\else#1\fi
  \else
    #1%
  \fi
}
```

`\textSC` Alternate formatting macro: small caps when active.

```
\newcommand*\textSC[1]
  {\if@nameauth@DoAlt\textsc{#1}\else#1\fi}
```

`\textUC` Alternate formatting macro: uppercase when active.

```
\newcommand*\textUC[1]
  {\if@nameauth@DoAlt\MakeUppercase{#1}\else#1\fi}
```

`\textIT` Alternate formatting macro: italic when active.

```
\newcommand*\textIT[1]
  {\if@nameauth@DoAlt\textit{#1}\else#1\fi}
```

`\textBF` Alternate formatting macro: boldface when active.

```
\newcommand*\textBF[1]
  {\if@nameauth@DoAlt\textbf{#1}\else#1\fi}
```

15.7.5 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 at once, only the current one.

```
\newcommand*\LocalNames{\global\@nameauth@LocalNamestrue}
```

\GlobalNames \GlobalNames restores the default behavior of \ForgetName and \SubvertName, which affect both name systems at once.

```
\newcommand*\GlobalNames{\global\@nameauth@LocalNamesfalse}
```

15.7.6 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.

```
\newcommand*\NameParser
{%
  \if@nameauth@InHook
    \let\SNN\rootb%
    \@nameauth@Choice
```

Nonwestern names. We test both \argc and \suffb as needed.

```
    {%
      \ifx\argc\@empty \let\FNN\suffb \else \let\FNN\argc \fi
      \ifx\FNN\@empty
        \SNN%
      \else
        \if@nameauth@FullName
          \if@nameauth@RevThis \FNN\Space\SNN \else \SNN\Space\FNN%
          \fi
        \else
          \if@nameauth@FirstName
            \if@nameauth@EastFN \FNN \else \SNN \fi
          \else
            \SNN%
          \fi
        \fi
      \fi
    }%
```

Nonwestern 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@FirstName \FNN \else \let\SNN\rootb \SNN \fi
      \fi
    }%
  \fi
}
```

15.8 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 any possible bad output.

15.8.1 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}
```

15.8.2 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. Redefining a shorthand 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 nonwestern names that use the new syntax. We use one `\ex` per token because we only have one argument to expand first.

```
\ifx\@testd@\@empty
  \ifx\@testb@\@empty
    \ex\csgdef\ex{\ex\@arga@\ex}%
      \ex{\ex\NameauthName\ex{\the\@nameauth@etoksc}}%
    \ex\csgdef\ex{\ex\@larga@\ex}%
      \ex{\ex\@nameauth@FullNametrue%
      \ex\NameauthLName\ex{\the\@nameauth@etoksc}}%
    \ex\csgdef\ex{\ex\@sarga@\ex}%
      \ex{\ex\@nameauth@FirstNametrue%
      \ex\NameauthFName\ex{\the\@nameauth@etoksc}}%
  \else
```

Next we have Western names with no alternate names. Here we have two arguments to expand in reverse order, so we need three uses, then one use of `\ex` per token.

```
    \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@arga@\ex\ex\ex}%
      \ex\ex\ex{\ex\ex\ex\NameauthName%
      \ex\ex\ex[\ex\the\ex\@nameauth@etoksb\ex]%
      \ex{\the\@nameauth@etoksc}}%
    \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@larga@\ex\ex\ex}%
      \ex\ex\ex{\ex\ex\ex\@nameauth@FullNametrue%
      \ex\ex\ex\NameauthLName%
      \ex\ex\ex[\ex\the\ex\@nameauth@etoksb\ex]%
      \ex{\the\@nameauth@etoksc}}%
    \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@sarga@\ex\ex\ex}%
      \ex\ex\ex{\ex\ex\ex\@nameauth@FirstNametrue%
      \ex\ex\ex\NameauthFName%
      \ex\ex\ex[\ex\the\ex\@nameauth@etoksb\ex]%
      \ex{\the\@nameauth@etoksc}}%
  \fi
\else
```

Below are “native” Eastern names with alternates and the older syntax. We again have two arguments to expand first.

```
      \ifx\@testb@\@empty
        \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@arga@\ex\ex\ex}%
          \ex\ex\ex{\ex\ex\ex\NameauthName%
          \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
          \ex[\the\@nameauth@etoksd]}%
        \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@larga@\ex\ex\ex}%
          \ex\ex\ex{\ex\ex\ex\@nameauth@FullNametrue%
          \ex\ex\ex\NameauthLName%
          \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
          \ex[\the\@nameauth@etoksd]}%
        \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@sarga@\ex\ex\ex}%
          \ex\ex\ex{\ex\ex\ex\@nameauth@FirstNametrue%
          \ex\ex\ex\NameauthFName%
          \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
          \ex[\the\@nameauth@etoksd]}%
      \else
```

Here are Western names with alternates. We have three arguments to expand, so we have seven uses, three uses, and one use of `\ex`.

```
        \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
          \ex\ex\ex\ex\ex\ex\ex\@arga@\ex\ex\ex\ex\ex\ex\ex}%
          \ex\ex\ex\ex\ex\ex\ex{\ex\ex\ex\ex\ex\ex\ex\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]}%
        \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]}%
        \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\ignorespaces}
```

15.8.3 Debugging Macros

`\ShowPattern` This displays the pattern that the name arguments generate; useful for debugging. We test for bad input, then load the argument values into the appropriate macros.

We determine the name type and produce the output of the appropriate name control sequence.

```
\NewDocumentCommand{\ShowPattern}{O{} m O{}}
{%
    \@nameauth@Error{#2}{macro: \string\ShowPattern}%
    \@nameauth@LoadArgs{#1}{#2}{#3}%
    \@nameauth@Choice{}{}{}\NameauthPattern%
}
```

\ShowIdxPageref
\ShowIdxPageref* Show an index page entry appearance for given name parameters.

```
\NewDocumentCommand{\ShowIdxPageref}{s O{} m O{}}
{%
  \IfBooleanTF {#1}%
```

The starred form displays a basic index entry with no tag. Test for bad input, load the argument values into the appropriate macros, then call the back-end.

```
  {%
    \@nameauth@Error{#3}{macro: \string\ShowIdxPageref*}%
    \@nameauth@LoadArgs{#2}{#3}{#4}%
    \@nameauth@IdxPageref{#2}{#3}{#4}%
  }%
```

The un-starred form displays (expanded, as printed) the index entry that will be generated, but not exactly what is in the `idx` file. Test for bad input, load the argument values into the appropriate macros, then call the back-end.

```
  {%
    \@nameauth@Error{#3}{macro: \string\ShowIdxPageref}%
    \@nameauth@LoadArgs{#2}{#3}{#4}%
    \global\@nameauth@LongIdxDebugtrue%
    \@nameauth@IdxPageref{#2}{#3}{#4}%
  }%
}
```

\ShowNameInfo Show how the name arguments are being interpreted by the nameauth macros, but as detokenized text,

```
\NewDocumentCommand{\ShowNameInfo}{O{} m O{}}
{%
```

Test for bad input, then load the argument values into the appropriate macros.

```
  \@nameauth@Error{#2}{macro: \string\ShowNameInfo}%
  \@nameauth@LoadArgs{#1}{#2}{#3}%
```

We produce what we know about the name arguments. First is nonwestern names using the current syntax.

```
  \@nameauth@Choice
  {%
    (SNN: \expandafter\detokenize\expandafter{\@nameauth@B})%
    \unless\ifx\@nameauth@SB\@empty
      \ (Affix*:
      \expandafter\detokenize\expandafter{\@nameauth@SB})%
    \fi
    \unless\ifx\@nameauth@C\@empty
      \ (Alt:
      \expandafter\detokenize\expandafter{\@nameauth@C})%
    \fi
  }%
```

Next is nonwestern names using the obsolete syntax.

```
  {%
    (SNN: \expandafter\detokenize\expandafter{\@nameauth@B})%
    \unless\ifx\@nameauth@C\@empty
      \ (Alt*:
      \expandafter\detokenize\expandafter{\@nameauth@C})%
    \fi
  }%
```

Finally we have Western names.

```
  {%
    (FNN: \expandafter\detokenize\expandafter{\@nameauth@A})
    (SNN: \expandafter\detokenize\expandafter{\@nameauth@B})%
    \unless\ifx\@nameauth@SB\@empty
      \ (Affix:
      \expandafter\detokenize\expandafter{\@nameauth@SB})%
    \fi
    \unless\ifx\@nameauth@C\@empty
      \ (Alt:
      \expandafter\detokenize\expandafter{\@nameauth@C})%
    \fi
  }%
}
```

`\ShowNameState` This macro tells the user what control sequence patterns exist for any given name.

```
\NewDocumentCommand{\ShowNameState}{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.

```
  \begingroup%
  \@nameauth@Error{#2}{macro: \string\NamePatterns}%
  \@nameauth@LoadArgs{#1}{#2}{#3}%
```

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
    \def\@nameauth@idxstate{1}%
    \ifdefined \@nameauth@xref
      \def\@nameauth@idxstate{3}%
    \fi
    \ifdefined \@nameauth@excl
      \def\@nameauth@idxstate{5}%
    \fi
  \fi
```

Display the output.

```
  Pattern: {\NameauthPattern}
  Type: {\@nameauth@nametype}
  Index state: {\@nameauth@idxstate}
  Systems:%
  \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%
}
```

15.8.4 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 into the appropriate macros; 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
```

Create the appropriate index entries, calling \@nameauth@Index to handle sorting and tagging. We do not create an index entry for a cross-reference or exclusion.

```
  \@nameauth@Choice
```

Nonwestern names. We ignore \@nameauth@C and handle \@nameauth@SB appropriately.

```
  {%
    \def\@nameauth@Temp{\expandafter\detokenize\expandafter{#2}}%
    \ifcsname\@nameauth@csb!PN\endcsname
      \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
      \ifx\@nameauth@SB\@empty
        \@nameauth@Index{\@nameauth@csb}{\@nameauth@B}%
      \else
```

```
        \@nameauth@Index{\@nameauth@csb}
          {\@nameauth@B\@nameauth@space\@nameauth@SB}%
      \fi
    \fi
  }%
```

Nonwestern names, obsolete syntax. Using `\@nameauth@C` in this path affects indexing.

```
  {%
    \def\@nameauth@Temp{\expandafter\detokenize\expandafter{#2 #3}}%
    \ifcsname\@nameauth@csbc!PN\endcsname
      \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
      \@nameauth@Index{\@nameauth@csbc}
        {\@nameauth@B\@nameauth@space\@nameauth@C}%
    \fi
  }%
```

Western names. We ignore `\@nameauth@C` and handle `\@nameauth@SB` appropriately.

```
  {%
    \def\@nameauth@Temp{\expandafter\detokenize\expandafter{#1 #2}}%
    \ifcsname\@nameauth@csab!PN\endcsname
      \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
      \ifx\@nameauth@SB\@empty
        \@nameauth@Index{\@nameauth@csab}
          {\@nameauth@B,\@nameauth@space\@nameauth@A}%
      \else
        \@nameauth@Index{\@nameauth@csab}
          {\@nameauth@B,\@nameauth@space%
           \@nameauth@A,\@nameauth@space\@nameauth@SB}%
      \fi
    \fi
  }%
}
```

`\IndexRef` Create a cross-reference that is not already an exclusion or a cross-reference. Print nothing.

```
\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}%
  \@nameauth@Xreftrue%
```

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
```

Nonwestern name, new syntax. First check if an xref or excluded, and if so, do nothing except issue warnings if so desired.

```
  {%
    \def\@nameauth@Temp{\expandafter\detokenize\expandafter{#2}}%
    \ifcsname\@nameauth@csb!PN\endcsname
      \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
```

If no xref or exclusion exists, either create a *see also* or a *see* reference. We permit the latter when a name exists only if the `oldsee` option is used; then issue a warning.

```
    \else
      \ifx\@nameauth@SB\@empty
        \if@nameauth@SeeAlso
          \@nameauth@Index{\@nameauth@csb}
            {\@nameauth@B|seealso{\@nameauth@Target}}%
          \csgdef{\@nameauth@csb!PN}{}%
        \else
          \unless\if@nameauth@OldSee
            \unless\ifcsname\@nameauth@csb!MN\endcsname
              \unless\ifcsname\@nameauth@csb!NF\endcsname
                \@nameauth@Index{\@nameauth@csb}
                  {\@nameauth@B|see{\@nameauth@Target}}%
                \csgdef{\@nameauth@csb!PN}{}%
              \else
                \PackageWarning{nameauth}
                {\string\IndexRef: extant name;
                 no xref \@nameauth@Temp}%
              \fi
            \else
              \PackageWarning{nameauth}
              {\string\IndexRef: extant name;
               no xref \@nameauth@Temp}%
            \fi
          \else
            \if@nameauth@Verbose
              \PackageWarning{nameauth}
              {\string\IndexRef: non-strict xref \@nameauth@Temp}%
            \fi
            \@nameauth@Index{\@nameauth@csb}
              {\@nameauth@B|see{\@nameauth@Target}}%
            \csgdef{\@nameauth@csb!PN}{}%
          \fi
        \fi
```

When the suffix is non-empty, either create a *see also* or a *see* reference. We permit the latter when a name exists only if the oldsee option is used; then issue a warning.

```
    \else
      \if@nameauth@SeeAlso
        \@nameauth@Index{\@nameauth@csb}
          {\@nameauth@B\@nameauth@space%
           \@nameauth@SB|seealso{\@nameauth@Target}}%
        \csgdef{\@nameauth@csb!PN}{}%
      \else
        \unless\if@nameauth@OldSee
          \unless\ifcsname\@nameauth@csb!MN\endcsname
            \unless\ifcsname\@nameauth@csb!NF\endcsname
              \@nameauth@Index{\@nameauth@csb}
                {\@nameauth@B\@nameauth@space%
                 \@nameauth@SB|see{\@nameauth@Target}}%
              \csgdef{\@nameauth@csb!PN}{}%
            \else
              \PackageWarning{nameauth}
              {\string\IndexRef: extant name;
               no xref \@nameauth@Temp}%
            \fi
          \else
            \PackageWarning{nameauth}
            {\string\IndexRef: extant name;
             no xref \@nameauth@Temp}%
          \fi
        \else
          \if@nameauth@Verbose
            \PackageWarning{nameauth}
            {\string\IndexRef: non-strict xref \@nameauth@Temp}%
          \fi
          \@nameauth@Index{\@nameauth@csb}
            {\@nameauth@B\@nameauth@space%
             \@nameauth@SB|see{\@nameauth@Target}}%
          \csgdef{\@nameauth@csb!PN}{}%
        \fi
      \fi
    \fi
  \fi
}%
```

Eastern or ancient name, obsolete syntax. First check if an xref or excluded.

```
{%
  \def\@nameauth@Temp{\expandafter\detokenize\expandafter{#2 #3}}%
  \ifcsname\@nameauth@csbc!PN\endcsname
    \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
```

If no xref control sequence exists, either create a *see also* or a *see* reference. We permit the latter when a name exists only if the oldsee option is used; then issue a warning.

```
  \else
    \if@nameauth@SeeAlso
      \@nameauth@Index{\@nameauth@csbc}
        {\@nameauth@B\@nameauth@space%
         \@nameauth@C|seealso{\@nameauth@Target}}%
      \csgdef{\@nameauth@csbc!PN}{}%
    \else
      \unless\if@nameauth@OldSee
        \unless\ifcsname\@nameauth@csbc!MN\endcsname
          \unless\ifcsname\@nameauth@csbc!NF\endcsname
            \@nameauth@Index{\@nameauth@csbc}
              {\@nameauth@B\@nameauth@space%
               \@nameauth@C|see{\@nameauth@Target}}%
            \csgdef{\@nameauth@csbc!PN}{}%
          \else
            \PackageWarning{nameauth}
            {\string\IndexRef: extant name;
             no xref \@nameauth@Temp}%
          \fi
        \else
          \PackageWarning{nameauth}
          {\string\IndexRef: extant name;
           no xref \@nameauth@Temp}%
        \fi
      \else
        \if@nameauth@Verbose
          \PackageWarning{nameauth}
          {\string\IndexRef: non-strict xref \@nameauth@Temp}%
        \fi
        \@nameauth@Index{\@nameauth@csbc}
```

```
          {\@nameauth@B\@nameauth@space%
           \@nameauth@C|see{\@nameauth@Target}}%
        \csgdef{\@nameauth@csbc!PN}{}%
      \fi
    \fi
  \fi
}%
```

Western name, without and with affix. First check if an xref or excluded.

```
{%
  \def\@nameauth@Temp{\expandafter\detokenize\expandafter{#1 #2}}%
  \ifcsname\@nameauth@csab!PN\endcsname
    \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
```

If no xref control sequence exists, either create a *see also* or a *see* reference. We permit the latter when a name exists only if the `oldsee` option is used; then issue a warning.

```
  \else
    \ifx\@nameauth@SB\@empty
      \if@nameauth@SeeAlso
        \@nameauth@Index{\@nameauth@csab}
          {\@nameauth@B,\@nameauth@space%
           \@nameauth@A|seealso{\@nameauth@Target}}%
        \csgdef{\@nameauth@csab!PN}{}%
      \else
        \unless\if@nameauth@OldSee
          \unless\ifcsname\@nameauth@csab!MN\endcsname
            \unless\ifcsname\@nameauth@csab!NF\endcsname
              \@nameauth@Index{\@nameauth@csab}
                {\@nameauth@B,\@nameauth@space%
                 \@nameauth@A|see{\@nameauth@Target}}%
              \csgdef{\@nameauth@csab!PN}{}%
            \else
              \PackageWarning{nameauth}
              {\string\IndexRef: extant name;
               no xref \@nameauth@Temp}%
            \fi
          \else
            \PackageWarning{nameauth}
            {\string\IndexRef: extant name;
             no xref \@nameauth@Temp}%
          \fi
        \else
          \if@nameauth@Verbose
            \PackageWarning{nameauth}
            {\string\IndexRef: non-strict xref \@nameauth@Temp}%
          \fi
          \@nameauth@Index{\@nameauth@csab}
            {\@nameauth@B,\@nameauth@space%
             \@nameauth@A|see{\@nameauth@Target}}%
          \csgdef{\@nameauth@csab!PN}{}%
        \fi
      \fi
```

When the suffix is non-empty, either create a *see also* or a *see* reference. We permit the latter when a name exists only if the oldsee option is used; then issue a warning.

```
    \else
      \if@nameauth@SeeAlso
        \@nameauth@Index{\@nameauth@csab}
          {\@nameauth@B,\@nameauth@space%
           \@nameauth@A,\@nameauth@space%
           \@nameauth@SB|seealso{\@nameauth@Target}}%
        \csgdef{\@nameauth@csab!PN}{}%
      \else
        \unless\if@nameauth@OldSee
          \unless\ifcsname\@nameauth@csab!MN\endcsname
            \unless\ifcsname\@nameauth@csab!NF\endcsname
              \@nameauth@Index{\@nameauth@csab}
                {\@nameauth@B,\@nameauth@space%
                 \@nameauth@A,\@nameauth@space%
                 \@nameauth@SB|see{\@nameauth@Target}}%
              \csgdef{\@nameauth@csab!PN}{}%
            \else
              \PackageWarning{nameauth}
              {\string\IndexRef: extant name;
               no xref \@nameauth@Temp}%
            \fi
          \else
            \PackageWarning{nameauth}
            {\string\IndexRef: extant name;
             no xref \@nameauth@Temp}%
          \fi
        \else
          \if@nameauth@Verbose
            \PackageWarning{nameauth}
            {\string\IndexRef: non-strict xref \@nameauth@Temp}%
          \fi
          \@nameauth@Index{\@nameauth@csab}
            {\@nameauth@B,\@nameauth@space%
             \@nameauth@A,\@nameauth@space%
             \@nameauth@SB|see{\@nameauth@Target}}%
          \csgdef{\@nameauth@csab!PN}{}%
        \fi
      \fi
    \fi
  \fi
}%
\@nameauth@Xreffalse%
\if@nameauth@OldReset
  \@nameauth@SeeAlsofalse%
\else
  \global\@nameauth@SeeAlsofalse%
  \fi
}
```

\ExcludeName Prevent a name from being indexed 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
\IncludeName* Allow names to be included as page entries, even if they have been used as cross-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 entry tag that is applied before a name by `\@nameauth@Index`.

```
\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 cross-references.

```
    \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 \@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}%
}
```

15.8.5 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
    \@nameauth@LoadArgs{#1}{#2}{#3}%
    \@nameauth@Error{#2}{macro \string\NameQueryInfo}%
    \@nameauth@Choice{}{}{}%
    \ifcsname\NameauthPattern!DB\endcsname
      \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}%
}
```

15.8.6 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
}
```

15.8.7 Pseudonyms

\AKA \AKA* \AKA prints an alternate name and creates index cross-references. The starred form displays the alternate name like \FName.

```
\NewDocumentCommand{\AKA}{s O{} m O{} m O{}}
{%
```

Prevent entering \AKA via itself or \@nameauth@Name. Prevents and resets \JustIndex. Tell the formatting system that \AKA is running.

```
  \if@nameauth@BigLock
    \@nameauth@Locktrue%
  \fi
  \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.

```
    \@nameauth@Error{#3}{macro \string\AKA}%
    \@nameauth@Error{#5}{macro \string\AKA}%
```

Names occur in horizontal mode; we ensure that. Next we make copies of the target name arguments and we parse and print the cross-reference name.

```
    \leavevmode\hbox{}%
    \protected@edef\@nameauth@Ai{\trim@spaces{#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
}
```

16 Change History

0.7
General: Initial release 1
0.75
General: Standardized arguments 1
0.85
General: Show or hide commas 1
0.9
\@nameauth@@Suffix: Added 141
\@nameauth@Suffix: Added 141
\FName: Added 162
\SubvertName: Added 180
0.94
\@nameauth@Index: Added 155
\CapThis: Added 156
\ExcludeName: Added 175
\IndexActive: Added 159
\IndexInactive: Added 159
1.0
General: Works with microtype, memoir 1
1.2
\TagName: Added 177
\UntagName: Added 178
1.26
\AKA: Fix affixes 181
\IndexName: Fix affixes 168
1.4
\ShowComma: Added 156
1.5
\@nameauth@@Suffix: Trim spaces 141
\@nameauth@Name: Reversing/caps 150
\AKA: Reversing/caps 181
\AllCapsActive: Added 158
\AllCapsInactive: Added 158
\CapName: Added 157
\RevComma: Added 157
\ReverseActive: Added 158
\ReverseCommaActive: Added 158
\ReverseCommaInactive: Added 158
\ReverseInactive: Added 158
\RevName: Added 157
1.6
nameauth: Environment added 162
1.9
\ForgetName: Global undef 180
\KeepAffix: Added 157
\TagName: Fix cs collisions 177
\UntagName: Global undef, no cs collisions . 178
2.0
\@nameauth@@Root: Trim spaces 141
\@nameauth@Actual: Added 139
\@nameauth@Index: New tagging 155
\@nameauth@Name: Trim spaces; fix tags . . . 150
\AKA: Trim spaces; fix tags 181
\IndexActual: Added 159
\IndexName: Fix spaces, tagging 168
nameauth: Better arg handling 162
\PretagName: Added 176
\TagName: Redesign tagging 177
\UntagName: Redesign tagging 178
2.1
\@nameauth@Name: Fix Unicode 150
\AccentCapThis: Added 157
\AKA: Fix Unicode 181
2.2
\NameauthFName: Added 138
\NameauthName: Added 138
2.3
General: New back-end for naming macros . . . 1
\@nameauth@Name: Now internal 150
\AKA: Fix starred mode 181
\ExcludeName: New xref test 175
\ForgetName: Global or local 180
\GlobalNames: Added 161
\IfAKA: Added 179
\IfFrontName: Added 179
\IfMainName: Added 178
\LocalNames: Added 160
\NameauthLName: Added 138
\PName: Work with hooks 182
\SubvertName: Global or local 180
2.4
\@nameauth@Hook: Current form 155
\@nameauth@Name: Set token regs 150
\FrontNameHook: Added 138
\GlobalNames: Ensure global 161
\IfAKA: Test for excluded 179
\LocalNames: Ensure global 160
\MainNameHook: Added 138
\NameAddInfo: Added 178
\NameClearInfo: Added 178
\NameQueryInfo: Added 178
2.41
\@nameauth@Name: Fix token regs 150
\AKA: Fix token regs 181
nameauth: No local \newtoks 162
2.5
General: No default format 1
\@nameauth@Hook: Improve hooks 155
\@nameauth@Name: Fix old syntax 150
\FrontNamesFormat: Added 138
2.6
\@nameauth@Name: Better indexing 150
\AKA: Fix index commas 181
\IndexName: Fix commas 168
\NoComma: Added 156

3.0
\@nameauth@@Root: Redesigned 141
\@nameauth@@Suffix: New test 141
\@nameauth@@TrimTag: Added 141
\@nameauth@Error: Added 149
\@nameauth@Hook: Fix punct. detection . . . 155
\@nameauth@Name: Redesigned 150
\@nameauth@NonWest: Added 153
\@nameauth@Parse: Added 151
\@nameauth@TrimTag: Added 141
\@nameauth@UTFtest: Added 142
\@nameauth@West: Added 153
\AKA: Redesigned 181
\DropAffix: Added 157
\ExcludeName: Redesigned 175
\ForceFN: Added 157
\IfAKA: Redesigned 179
\IncludeName: Added 175
\IndexName: Redesigned 168
\IndexRef: Added 169
\NameParser: Added 161
\SeeAlso: Added 157
3.01
\@nameauth@Error: Fixed 149
3.02
\@nameauth@NonWest: Restrict \ForceFN . . 153
3.03
\NameParser: Restrict first names 161
3.1
\@nameauth@C@p: Added 143
\@nameauth@C@pUTF: Added 143
\@nameauth@Cap: Redesigned 143
\@nameauth@Name: New workflow 150
\@nameauth@Parse: New workflow, caps . . . 151
\@nameauth@UTFtest: Can skip test 142
\AKA: Can skip index 181
\AltCaps: Added 160
\AltFormatActive: Added 159
\AltFormatInactive: Added 159
\AltOff: Added 160
\AltOn: Added 160
\ForceName: Added 158
\ForgetThis: Added 158
\IndexName: Better tests 168
\IndexRef: Better tests 169
\JustIndex: Added 157
\KeepName: Added 157
\NameParser: Fix old syntax; add NBSP . . . 161
\NameQueryInfo: Short macro 178
\PName: Can skip index 182
\SkipIndex: Added 157
\SubvertName: Fix old syntax 180
\SubvertThis: Added 158
\textBF: Added 160
\textIT: Added 160
\textSC: Added 160
\textUC: Added 160
3.2
General: Root, suffix macros renamed, redesigned 1
\@nameauth@@GetSuff: Added 142
\@nameauth@@Root: Renamed 141
\@nameauth@@Suffix: Renamed 141
\@nameauth@C@p: Renamed, use \MakeUppercase 143
\@nameauth@C@pUTF: Use \MakeUppercase . 143
\@nameauth@Cap: Non-UTF 143
\@nameauth@CapUTF: Added 143
\@nameauth@GetSuff: Added 141
\@nameauth@Parse: Fix alt. format, affixes, use \MakeUppercase 151
\@nameauth@TestToks: Added 142
\@nameauth@TrimTag: Renamed 141
\@nameauth@UTFtest: Non-suffix only 142
\@nameauth@UTFtestS: Added 142
\AltCaps: Use \MakeUppercase 160
\NameParser: Fix alt. format, affixes 161
\textUC: Use \MakeUppercase 160
3.3
\@nameauth@IdxPageref: Added 149
\@nameauth@Index: Support hyperref 155
\@nameauth@Name: Global flag reset 150
\@nameauth@NonWest: global flag reset 153
\@nameauth@West: global flag reset 153
\AKA: Global flag reset 181
\ExcludeName: Better warnings 175
\IncludeName: Added warnings 175
\IndexProtect: Added 159
\IndexRef: Global flag reset 169
\NameQueryInfo: Lock added 178
\ShowIdxPageref: Added 165
\ShowPattern: Added 164
3.4
General: Update manual, examples.tex 1
3.5
General: Update manual, Readme.md, Makefile,examples.tex; combine Readme.md and examples.tex files in dtx file 1
\@nameauth@Actual: Use \def 139
\@nameauth@AddPunct: Added 146
\@nameauth@CapArgs: Added 143
\@nameauth@Choice: Added 147
\@nameauth@Error: Fix namespace 149
\@nameauth@Exclude: Added 139
\@nameauth@Flags: Added 148
\@nameauth@Form: Added 154
\@nameauth@Hook: Fix namespace 155
\@nameauth@IdxPageref: Use index hook, optimize logic, fix name space, use Boolean flags 149
\@nameauth@Index: Fix namespace 155

\@nameauth@LoadArgs: Added 146
\@nameauth@MakeCS: Added 141
\@nameauth@Parse: Global token regs, optimize logic, fix namespace 151
\@nameauth@TestDot: Redesigned 145
\@nameauth@TestToks: Fix namespace 142
\@nameauth@UTFtest: Fix namespace 142
\@nameauth@UTFtestS: Fix namespace 142
\AKA: Fix namespace 181
\ExcludeName: New warnings, new exclusion test, fix bug in old syntax, new logic, fix namespace 175
\ForgetName: Improve logic, fix namespace . 180
\GlobalNameTest: Added 160
\IfAKA: New exclusion test, optimize logic, fix namespace, local or global scope 179
\IfFrontName: Improve logic, fix namespace, local or global scope 179
\IfMainName: Improve logic, fix namespace, local or global scope 178
\IncludeName: New exclusion test, optimize logic, fix namespace 175
\IndexActual: Use \def 159
\IndexName: New warnings, new exclusion test, improve logic, fix namespace 168
\IndexRef: Strict *see* refs, new warnings, new exclusion test, improve logic, fix namespace 169
\LocalNameTest: Added 160
\NameAddInfo: Improve logic, fix namespace 178
nameauth: Fix namespace 162
\NameauthIndex: Added 138
\NameClearInfo: Improve logic, fix namespace 178
\NameParser: Optimize logic 161
\NameQueryInfo: Improve logic, fix namespace 178
\PName: Warning and flag resets added 182
\PretagName: New warnings, new exclusion test, improve logic, fix namespace 176
\ShowIdxPageref: Fix name space, use Boolean flags 165
\SubvertName: Improve logic, fix namespace 180
\TagName: New warnings, new exclusion test, improve logic, fix namespace 177
\UntagName: Improve logic, fix namespace . . 178

3.6

General: Update Readme.md, Makefile 1

3.7

General: Major updates to all files 1
\@nameauth@Choice: Access name pattern and type; redesigned to optimize many macros 147
\@nameauth@IdxPageref: Renamed; only show index entries; add warning; optimized . . 149
\@nameauth@West: always define local macros 153
\@nameauth@space: Made global 139
\ExcludeName: Fix warnings, optimized . . . 175
\ForgetName: Optimized 180
\IfAKA: Optimized 179
\IfFrontName: Optimized 179
\IfMainName: Optimized 178
\IncludeName: Fix warnings, optimized . . . 175
\IndexName: Fix warnings 168
\IndexRef: Fix warnings 169
\IndexWarnTerse: Added 159
\IndexWarnVerbose: Added 159
\NameAddInfo: Optimized 178
nameauth: Improve warnings 162
\NameauthPattern: Added 138
\NameClearInfo: Optimized 178
\NameQueryInfo: Optimized 178
\PName: Fix warnings 182
\PretagName: Fix warnings, optimized 176
\ShowIdxPageref: Use common back-end . . 165
\ShowNameInfo: Added 165
\ShowNameState: Added 166
\ShowPattern: Redesigned 164
\SubvertName: Optimized 180
\TagName: Optimized 177
\UntagName: Optimized 178

4.0

General: Improve manual, Readme.md, compat.tex 1
\@nameauth@Name: Use xparse 150
\AKA: Combine starred, un-starred macros . . 181
\AltFormatActive: Combine starred, un-starred macros 159
\ExcludeName: Use xparse 175
\FName: Combine starred, un-starred macros 162
\ForgetName: Use xparse 180
\IfAKA: Use xparse 179
\IfFrontName: Use xparse 179
\IfMainName: Use xparse 178
\IncludeName: Combine starred, un-starred macros 175
\IndexName: Use xparse 168
\IndexRef: Use xparse 169
\Name: Combine starred, un-starred macros . 162
\NameAddInfo: Use xparse 178
\NameClearInfo: Use xparse 178
\NameQueryInfo: Use xparse 178
\PName: Combine starred, un-starred macros 182
\PretagName: Use xparse 176
\ShowIdxPageref: Combine starred, un-starred macros 165
\ShowNameInfo: Use xparse, improve output 165
\ShowNameState: Use xparse 166
\ShowPattern: Use xparse 164
\SubvertName: Use xparse 180
\TagName: Use xparse 177
\UntagName: Use xparse 178

17 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols

\@nameauth@@GetSuff 130
\@nameauth@@Root 122
\@nameauth@@Suffix 126
\@nameauth@@TrimTag 124
\@nameauth@Actual 67
\@nameauth@AddPunct 273
\@nameauth@C@p 185
\@nameauth@C@pUTF 189
\@nameauth@Cap 184
\@nameauth@CapArgs 192
\@nameauth@CapUTF 188
\@nameauth@CheckDot 265
\@nameauth@Choice 306
\@nameauth@Clean 113
\@nameauth@Error 368
\@nameauth@EvalDot 267
\@nameauth@Exclude 68
\@nameauth@Flags 330
\@nameauth@Form 568
\@nameauth@GetSuff 129
\@nameauth@Hook 597
\@nameauth@IdxPageref 381
\@nameauth@Index 618
\@nameauth@LoadArgs 296
\@nameauth@MakeCS 115
\@nameauth@Name 403
\@nameauth@NonWest 517
\@nameauth@Parse 443
\@nameauth@Root 121
\@nameauth@Suffix 125
\@nameauth@TestDot 252
\@nameauth@TestToks 131
\@nameauth@TrimTag 123
\@nameauth@UTFtest 143
\@nameauth@UTFtestS 161
\@nameauth@West 544
\@nameauth@space 69
\@nameauth@toksa 52, *88*
\@nameauth@toksb 52, *88*
\@nameauth@toksc 52, *88*

A

\AccentCapThis *39*, 657
Æthelred II, king 18, 34, 38, 56, 64
\AKA *116*, 1568
\AKA* *116*, 1568
à Kempis, Thomas *see* Thomas à Kempis
\AllCapsActive *37*, 674
\AllCapsInactive *37*, 673
\AltCaps *95*, 709
\AltFormatActive *93*, 688
\AltFormatActive* *93*, 688
\AltFormatInactive *94*, 695
\AltOff *96*, 703
\AltOn *96*, 697
Andreae, Ioannes *see* d'Andrea, Giovanni
Antiochus III the Great, king 105
Antiochus IV Epiphanes, king 90
Aristotle 14, 18, 42, 64, 112
Arouet, François-Marie *see* Voltaire
Atatürk *see* Kemal, Mustafa
Auden, W.H. 44

B

Babbage, Charles 95, 96, 112
Bailey, Betsey 5, 72
Bernard of Clairvaux 118
Bernstein, Leonard 3
Bess, Good Queen *see* Elizabeth I
Boëthius 34, 45
BURNS, Robert 115

C

Caesar, Julius, imperator 101
\CapName *37*, 659
\CapThis *39*, 656
Carnap, Rudolph 48, 68, 74
Carter, J.E., Jr., pres. 22, 27, 28, 46, 83
Carter, Jimmy *see* Carter, J.E., Jr.
Chaplin, Charlie 71
Chesnutt, Charles W. 4
Chiang Kai-shek‡, pres. 120
Cicero, M.T. 8, 9, 68, 69, 101
civil rights leaders, quotes of 5, 25, 82
Clemens, Samuel L. *see* Twain, Mark
Colfax, Schuyler, v.p. 65
Confucius 27, 28, 46, 66, 68
Cornelius Scipio Barbatus, Lucius, consul 108, 109
Cratylus 133
creatives, quotes of 3, 30, 43, 44, 79, 100, 115, 120
cummings, e.e. 21, 83

D

Dagobert I‡, king 120
d'Andrea, Giovanni 85
Davis, Sammy, Jr. 54
Demetrius I Soter, king 42, 91, 106
De Pamele, Jacques 85
de Smet, Pierre-Jean 122, 124
de Soto, Hernando 18, 39, 45, 46, 64
Doctor angelicus *see* Thomas Aquinas
Doctor mellifluus *see* Bernard of Clairvaux
Dongen, Marc van 6, 150
Douglass, Frederick 5, 64
\DropAffix *36*, 663
DuBois, W.E.B. *see* Du Bois, W.E.B.
Du Bois, W.E.B. 56, 81, 82
du Cange *see* du Fresne, Charles
du Fresne, Charles 117

E

Einstein, Albert 27, 28, 30, 68, 70, 110, 127
Eisenhower, Dwight D., pres. 28, 62, 64
Elizabeth I, queen 14, 21, 27, 28, 46, 68, 73, 81, 110, 118, 119
environments:
  nameauth *15*, 793
Erikson, Leif 8
\ExcludeName *52*, 1351

F

Fairbairns, Robin 6
\FName *28*, 791
\FName* *28*, 791
foo § 50
\ForceFN *28*, 661
\ForceName *68*, 670
\ForgetName *70*, 1536
\ForgetThis *70*, 671
\FrontNameHook 61, *68*
\FrontNamesFormat 60, *68*
FUKUYAMA Takeshi‡ 119, 120

G

GARBO, Greta 100
Ghazālī 132
\GlobalNames *70*, 728

\GlobalNameTest *73*, 726
Goethe, J.W. von 39, 75
Gracchus, Tiberius Sempronius, consul 106
Grant, Ulysses S., pres. 65, 77–79
Gregorio, Enrico 6
Gregory I the Great, pope . . . 59

H

Hammerstein, Oskar, II 35, 36, 38
Harnack, Adolf 42, 69, 126
Hearn, Lafcadio 117
Henry VIII, king 42, 46, 48, 110, 120
Hermogenes 133
Hope, Bob 73, 117
Hope, Leslie Townes *see* Hope, Bob
Humperdinck, E. (composer) . 59
Humperdinck, E. (singer) . . . 59
Humphrey, Hubert H., v.p. . . 72

I

\IfAKA *74*, 1519
\IfFrontName *74*, 1508
\IfMainName *73*, 1497
\IncludeName *52*, 1385
\IncludeName* *52*, 1385
\IndexActive *49*, 681
\IndexActual *56*, 679
\IndexInactive *49*, 680
\IndexName *50*, 1017
\IndexProtect *50*, 684
\IndexRef *51*, 1103
\IndexWarnTerse *49*, 683
\IndexWarnVerbose . . . *49*, 682

J

Janos, James *see* Ventura, Jesse
Jefferson, Thomas, pres. . . 98
Jesus Christ 8, 9, 25, 76
John Eriugena 14, 18
\JustIndex *54*, 668

K

KANADE Takeo 95, 96, 112
\KeepAffix *36*, 664
\KeepName *36*, 665
Kemal, Mustafa 93
Kennedy, John F., pres. 19
Kim Jong Un 36
King, Martin Luther, Jr. 25, 31–33, 64
Koizumi Yakumo . . . . *see* Hearn, Lafcadio

L

Lao-tzu 117, 118
Lewis, Clive Staples 11, 17, 18, 58
Li Er . . . . *see* Lao-tzu
Lincoln, Abraham, pres. 64
Liszt, Franz . . . *see* Liszt, Frenec
Liszt, Frenec† 8, 110
Livius, Titus 106
Llywelyn ap Gruffudd, prince . 8
\LocalNames *70*, 727
\LocalNameTest *73*, 725
Louis XIV, king 35, 36, 117
Lovelace, Ada 95, 96, 112
Lueck, Uwe 6, 145
Luecking, Dan 6
Luther, Martin . 47, 48, 97, 125

M

Maimonides 84, *see also* Rambam
\MainNameHook 59, *68*
Malebranche, Nicolas 68
Martin, Dean 54
Medici, Catherine de' 46, 97, 125
Misora Hibari 100
Miyazaki Hayao 13, 17, 18, 27, 28, 30, 37, 46, 127
Molnár, Frenec† 12
Montgomery, L.M. 100
Moses ben-Maimon *see* Maimonides
Mr. Baseball . . . *see* Uecker, Bob
Mulvany, Nancy C. 4

N

\Name *27*, 785
\Name* *27*, 785
Name, Lost § . . . . perdit(60)
\NameAddInfo *65*, 1472
nameauth (env.) *15*, 793
\NameauthFName 64, *125*
\NameauthIndex *49*, 65
\NameauthLName 63, *125*
\NameauthName 62, *125*
\NameauthPattern *41*, 66
\NameClearInfo *66*, 1490
\NameParser *109*, 729
\NameQueryInfo *65*, 1479
\NamesActive *67*, 687
\NamesFormat 58, *68*
\NamesInactive *67*, 686
Nippon Gakki 49, 54, 55
\NoComma *36*, 655
Noguchi, Hideyo† 12, 17, 37, 38, 42
north star, concept of 5, 66

O

Oberdiek, Heiko 6, 59, 141
O'Connor, Sandra Day, justice 36

P

Pamelius, Jacobus . . . . *see* De Pamele, Jacques
Patton, George S., Jr. . . . 10, 11, 17, 36, 38, 42, 64, 67
Paul . . . . *see* Saul of Tarsus
philosophers, quotes of 66, 121, 133
Plato 64, 133
\PName *117*, 1614
\PName* *117*, 1614
politicians, quotes of 9, 11, 19, 28, 64, 72
Pontius Pilate 8, 9, 101
\PretagName *56*, 1415
Ptolemy IV Philopator, king 105
Ptolemy V Epiphanes, king . 105

R

Rambam 84, *see also* Maimonides
Ranieri, Luke 102
Rat Pack, the . . . . . . 54, *see also* Davis, Sammy, Jr.; Martin, Dean; Sinatra, Frank
\RevComma *38*, 662
\ReverseActive *37*, 676
\ReverseCommaActive . *38*, 678
\ReverseCommaInactive *38*, 677
\ReverseInactive *37*, 675
\RevName *37*, 660
Rockefeller, Jay . . . . *see* Rockefeller, J.D., IV
Rockefeller, J.D., IV 11, 17, 53, 58
Romans, quotes of 57, 69, 101, 106
Roosevelt, Theodore, pres. 9
Rühmann, Heinrich Wilhelm . . . . *see* Rühmann, Heinz
Rühmann, Heinz 100
Rushdie, Salman 120

S

Saul of Tarsus 76
Schlicht, Robert 6
scientists, quotes of 18
Scipio Africanus, Publius Cornelius 105, 106
\SeeAlso *54*, 669
Seleucus III Ceraunus, king . 105
Sergius Paulus, Lucius . . . 76, 101
Shakespeare, William . . 79, 115
Shakespearian rose, references to 72, 79, 95, 100
\ShowComma *36*, 654
\ShowIdxPageref *45*, 895
\ShowIdxPageref* *46*, 895
\ShowNameInfo *46*, 910
\ShowNameState *47*, 946
\ShowPattern *45*, 889
Sinatra, Frank 54
\SkipIndex *54*, 667
Snel van Royen, R. 84

Snel van Royen, W. 84
Snellius . *see* Snel van Royen, R.; Snel van Royen, W.
Socrates 133
Stephani, Philipp 6
Strietelmeier, John . . . 34, 35, 46
`\SubvertName` *70*, 1552
`\SubvertThis` *70*, 672
Sun King . . . . . . *see* Louis XIV
Sun Yat-sen, pres. . 35, 37, 38, 64

T

`\TagName` *58*, 1441
`\textBF` *95*, 723
`\textIT` *95*, 721
`\textSC` *95*, 717
`\textUC` *95*, 719
theologians, quotes of 25, 48, 121
Thomas à Kempis 40, 48
Thomas Aquinas 57
Tully . . . . *see* Cicero, M.T.
Twain, Mark 43, 118
Tyson, Mike 80

U

Uecker, Bob 52
`\UntagName` *58*, 1465
Urey, Harold 18

V

Van Buren, Martin, pres. 39
Vegetius Renatus, Publius . . . 57
Ventura, Jesse 74
Vlad II Dracul 88
Vlad III Dracula 88
Vlad Ţepeş . . . . *see* Vlad III
Voltaire 118

W

Washington, George, pres. 10, 11, 16, 17, 54, 62, 64–66, 77–79, 92
White, E. B. 15, 33
William I, king 118
William the Conqueror . . . . *see* William I

Y

Yamaha Corp. 55
Yamaha Torakusu . . . . . . 49, 50, *see also* Nippon Gakki
Yamamoto Isoroku 42, 67
Yoshida Shigeru‡, PM 120

Z

Ziegler, Caspar 85
ZUSE, Konrad 95, 96