notmuch

Release 0.21

March 12, 2016

Contents

Contents:

CHAPTER 1

notmuch

1.1 SYNOPSIS

notmuch [option ...] **command** [arg ...]

1.2 DESCRIPTION

Notmuch is a command-line based program for indexing, searching, reading, and tagging large collections of email messages.

This page describes how to get started using notmuch from the command line, and gives a brief overview of the commands available. For more information on e.g. **notmuch show** consult the **notmuch-show(1)** man page, also accessible via **notmuch help show**

The quickest way to get started with Notmuch is to simply invoke the `notmuch` command with no arguments, which will interactively guide you through the process of indexing your mail.

1.3 NOTE

While the command-line program `notmuch` provides powerful functionality, it does not provide the most convenient interface for that functionality. More sophisticated interfaces are expected to be built on top of either the command-line interface, or more likely, on top of the notmuch library interface. See http://notmuchmail.org for more about alternate interfaces to notmuch. The emacs-based interface to notmuch (available under **emacs/** in the Notmuch source distribution) is probably the most widely used at this time.

1.4 OPTIONS

Supported global options for `notmuch` include

`--help` [command-name] Print a synopsis of available commands and exit. With an optional command name, show the man page for that subcommand.

`--version` Print the installed version of notmuch, and exit.

`--config=FILE` Specify the configuration file to use. This overrides any configuration file specified by ${NOTMUCH_CONFIG}.

--uuid=HEX Enforce that the database UUID (a unique identifier which persists until e.g. the database is compacted) is HEX; exit with an error if it is not. This is useful to detect rollover in modification counts on messages. You can find this UUID using e.g. `notmuch count --lastmod`

All global options except `--config` can also be specified after the command. For example, `notmuch subcommand --uuid=HEX` is equivalent to `notmuch --uuid=HEX subcommand`.

1.5 COMMANDS

1.5.1 SETUP

The **notmuch setup** command is used to configure Notmuch for first use, (or to reconfigure it later).

The setup command will prompt for your full name, your primary email address, any alternate email addresses you use, and the directory containing your email archives. Your answers will be written to a configuration file in ${NOTMUCH_CONFIG} (if set) or ${HOME}/.notmuch-config . This configuration file will be created with descriptive comments, making it easy to edit by hand later to change the configuration. Or you can run **notmuch setup** again to change the configuration.

The mail directory you specify can contain any number of sub-directories and should primarily contain only files with individual email messages (eg. maildir or mh archives are perfect). If there are other, non-email files (such as indexes maintained by other email programs) then notmuch will do its best to detect those and ignore them.

Mail storage that uses mbox format, (where one mbox file contains many messages), will not work with notmuch. If that's how your mail is currently stored, it is recommended you first convert it to maildir format with a utility such as mb2md before running **notmuch setup .**

Invoking `notmuch` with no command argument will run **setup** if the setup command has not previously been completed.

1.5.2 OTHER COMMANDS

Several of the notmuch commands accept search terms with a common syntax. See **notmuch-search-terms**(7) for more details on the supported syntax.

The **search**, **show**, **address** and **count** commands are used to query the email database.

The **reply** command is useful for preparing a template for an email reply.

The **tag** command is the only command available for manipulating database contents.

The **dump** and **restore** commands can be used to create a textual dump of email tags for backup purposes, and to restore from that dump.

The **config** command can be used to get or set settings in the notmuch configuration file.

1.6 ENVIRONMENT

The following environment variables can be used to control the behavior of notmuch.

NOTMUCH_CONFIG Specifies the location of the notmuch configuration file. Notmuch will use ${HOME}/.notmuch-config if this variable is not set.

NOTMUCH_TALLOC_REPORT Location to write a talloc memory usage report. See **talloc_enable_leak_report_full** in **talloc(3)** for more information.

NOTMUCH_DEBUG_QUERY If set to a non-empty value, the notmuch library will print (to stderr) Xapian queries it constructs.

1.7 SEE ALSO

notmuch-config(1), **notmuch-count(1)**, **notmuch-dump(1)**, **notmuch-hooks(5)**, **notmuch-insert(1)**, **notmuch-new(1)**, **notmuch-reply(1)**, **notmuch-restore(1)**, **notmuch-search(1)**, **notmuch-search-terms(7)**, **notmuch-show(1)**, **notmuch-tag(1)**, **notmuch-address(1)**

The notmuch website: **http://notmuchmail.org**

1.8 CONTACT

Feel free to send questions, comments, or kudos to the notmuch mailing list <notmuch@notmuchmail.org> . Subscription is not required before posting, but is available from the notmuchmail.org website.

Real-time interaction with the Notmuch community is available via IRC (server: irc.freenode.net, channel: #notmuch).

CHAPTER 2

notmuch-address

2.1 SYNOPSIS

notmuch address [*option* ...] <*search-term*> ...

2.2 DESCRIPTION

Search for messages matching the given search terms, and display the addresses from them. Duplicate addresses are filtered out.

See **notmuch-search-terms(7)** for details of the supported syntax for <search-terms>.

Supported options for **address** include

--format=(json|sexp|text|text0) Presents the results in either JSON, S-Expressions, newline character separated plain-text (default), or null character separated plain-text (compatible with **xargs(1)** -0 option where available).

--format-version=N Use the specified structured output format version. This is intended for programs that invoke **notmuch(1)** internally. If omitted, the latest supported version will be used.

`--output=(sender|recipients|count)`

Controls which information appears in the output. This option can be given multiple times to combine different outputs. When neither –output=sender nor –output=recipients is given, –output=sender is implied.

sender Output all addresses from the *From* header.

Note: Searching for **sender** should be much faster than searching for **recipients**, because sender addresses are cached directly in the database whereas other addresses need to be fetched from message files.

recipients Output all addresses from the *To*, *Cc* and *Bcc* headers.

count Print the count of how many times was the address encountered during search.

Note: With this option, addresses are printed only after the whole search is finished. This may take long time.

`--deduplicate=(no|mailbox|address)`

Control the deduplication of results.

no Output all occurences of addresses in the matching messages. This is not applicable with –output=count.

mailbox Deduplicate addresses based on the full, case sensitive name and email address, or mailbox. This is effectively the same as piping the –deduplicate=no output to **sort | uniq**, except for the order of results. This is the default.

address Deduplicate addresses based on the case insensitive address part of the mailbox. Of all the variants (with different name or case), print the one occurring most frequently among the matching messages. If –output=count is specified, include all variants in the count.

--sort=(newest-first|oldest-first) This option can be used to present results in either chronological order (**oldest-first**) or reverse chronological order (**newest-first**).

By default, results will be displayed in reverse chronological order, (that is, the newest results will be displayed first).

However, if either –output=count or –deduplicate=address is specified, this option is ignored and the order of the results is unspecified.

--exclude=(true|false) A message is called "excluded" if it matches at least one tag in search.tag_exclude that does not appear explicitly in the search terms. This option specifies whether to omit excluded messages in the search process.

The default value, **true**, prevents excluded messages from matching the search terms.

false allows excluded messages to match search terms and appear in displayed results.

2.3 EXIT STATUS

This command supports the following special exit status codes

20 The requested format version is too old.

21 The requested format version is too new.

2.4 SEE ALSO

notmuch(1), **notmuch-config(1)**, **notmuch-count(1)**, **notmuch-dump(1)**, **notmuch-hooks(5)**, **notmuch-insert(1)**, **notmuch-new(1)**, **notmuch-reply(1)**, **notmuch-restore(1)**, **notmuch-search-terms(7)**, **notmuch-show(1)**, **notmuch-tag(1)**, **notmuch-search(1)**

CHAPTER 3

notmuch-compact

3.1 SYNOPSIS

notmuch compact [–quiet] [–backup=<*directory*>]

3.2 DESCRIPTION

The **compact** command can be used to compact the notmuch database. This can both reduce the space required by the database and improve lookup performance.

The compacted database is built in a temporary directory and is later moved into the place of the origin database. The original uncompacted database is discarded, unless the `--backup=`<directory> option is used.

Note that the database write lock will be held during the compaction process (which may be quite long) to protect data integrity.

Supported options for **compact** include

`--backup=<directory>` Save the current database to the given directory before replacing it with the compacted database. The backup directory must not exist and it must reside on the same mounted filesystem as the current database.

`--quiet` Do not report database compaction progress to stdout.

3.3 ENVIRONMENT

The following environment variables can be used to control the behavior of notmuch.

NOTMUCH_CONFIG Specifies the location of the notmuch configuration file. Notmuch will use ${HOME}/.notmuch-config if this variable is not set.

3.4 SEE ALSO

notmuch(1), **notmuch-count(1)**, **notmuch-dump(1)**, **notmuch-hooks(5)**, **notmuch-insert(1)**, **notmuch-new(1)**, **notmuch-reply(1)**, **notmuch-restore(1)**, **notmuch-search(1)**, **notmuch-search-terms(7)**, **notmuch-show(1)**, **notmuch-tag(1)**

CHAPTER 4

notmuch-config

4.1 SYNOPSIS

notmuch config get <*section*>.<*item*>

notmuch config set <*section*>.<*item*> [*value* ...]

notmuch config list

4.2 DESCRIPTION

The **config** command can be used to get or set settings in the notmuch configuration file.

get The value of the specified configuration item is printed to stdout. If the item has multiple values (it is a list), each value is separated by a newline character.

set The specified configuration item is set to the given value. To specify a multiple-value item (a list), provide each value as a separate command-line argument.

If no values are provided, the specified configuration item will be removed from the configuration file.

list Every configuration item is printed to stdout, each on a separate line of the form:

section.item=value

No additional whitespace surrounds the dot or equals sign characters. In a multiple-value item (a list), the values are separated by semicolon characters.

The available configuration items are described below.

database.path The top-level directory where your mail currently exists and to where mail will be delivered in the future. Files should be individual email messages. Notmuch will store its database within a sub-directory of the path configured here named `.notmuch`.

Default: `$MAILDIR` variable if set, otherwise `$HOME/mail`.

user.name Your full name.

Default: `$NAME` variable if set, otherwise read from `/etc/passwd`.

user.primary_email Your primary email address.

Default: `$EMAIL` variable if set, otherwise constructed from the username and hostname of the current machine.

user.other_email A list of other email addresses at which you receive email.

Default: not set.

new.tags A list of tags that will be added to all messages incorporated by **notmuch new**.

Default: `unread;inbox`.

new.ignore A list of file and directory names, without path, that will not be searched for messages by **notmuch new**. All the files and directories matching any of the names specified here will be ignored, regardless of the location in the mail store directory hierarchy.

Default: empty list.

search.exclude_tags A list of tags that will be excluded from search results by default. Using an excluded tag in a query will override that exclusion.

Default: empty list. Note that **notmuch-setup(1)** puts `deleted;spam` here when creating new configuration file.

maildir.synchronize_flags If true, then the following maildir flags (in message filenames) will be synchronized with the corresponding notmuch tags:

Flag	Tag
D	draft
F	flagged
P	passed
R	replied
S	unread (added when 'S' flag is not present)

The **notmuch new** command will notice flag changes in filenames and update tags, while the **notmuch tag** and **notmuch restore** commands will notice tag changes and update flags in filenames.

If there have been any changes in the maildir (new messages added, old ones removed or renamed, maildir flags changed, etc.), it is advisable to run **notmuch new** before **notmuch tag** or **notmuch restore** commands to ensure the tag changes are properly synchronized to the maildir flags, as the commands expect the database and maildir to be in sync.

Default: `true`.

crypto.gpg_path

Name (or full path) of gpg binary to use in verification and decryption of PGP/MIME messages.

Default: `gpg`.

4.3 ENVIRONMENT

The following environment variables can be used to control the behavior of notmuch.

NOTMUCH_CONFIG Specifies the location of the notmuch configuration file. Notmuch will use ${HOME}/.notmuch-config if this variable is not set.

4.4 SEE ALSO

notmuch(1), **notmuch-count(1)**, **notmuch-dump(1)**, **notmuch-hooks(5)**, **notmuch-insert(1)**, **notmuch-new(1)**, **notmuch-reply(1)**, **notmuch-restore(1)**, **notmuch-search(1)**, **notmuch-search-terms(7)**, **notmuch-show(1)**,

notmuch-tag(1)

CHAPTER 5

notmuch-count

5.1 SYNOPSIS

notmuch count [*option ...*] <*search-term*> ...

5.2 DESCRIPTION

Count messages matching the search terms.

The number of matching messages (or threads) is output to stdout.

With no search terms, a count of all messages (or threads) in the database will be displayed.

See **notmuch-search-terms(7)** for details of the supported syntax for <search-terms>.

Supported options for **count** include

`--output=(messages|threads|files)`

messages Output the number of matching messages. This is the default.

threads Output the number of matching threads.

files Output the number of files associated with matching messages. This may be bigger than the number of matching messages due to duplicates (i.e. multiple files having the same message-id).

`--exclude=(true|false)` Specify whether to omit messages matching search.tag_exclude from the count (the default) or not.

`--batch` Read queries from a file (stdin by default), one per line, and output the number of matching messages (or threads) to stdout, one per line. On an empty input line the count of all messages (or threads) in the database will be output. This option is not compatible with specifying search terms on the command line.

`--lastmod` Append lastmod (counter for number of database updates) and UUID to the output. lastmod values are only comparable between databases with the same UUID.

`--input=<filename>` Read input from given file, instead of from stdin. Implies `--batch`.

5.3 SEE ALSO

notmuch(1), **notmuch-config(1)**, **notmuch-dump(1)**, **notmuch-hooks(5)**, **notmuch-insert(1)**, **notmuch-new(1)**, **notmuch-reply(1)**, **notmuch-restore(1)**, **notmuch-search(1)**, **notmuch-search-terms(7)**, **notmuch-show(1)**, **notmuch-tag(1)**

CHAPTER 6

notmuch-dump

6.1 SYNOPSIS

notmuch dump [–gzip] [–format=(batch-tag|sup)] [–output=<*file*>] [–] [<*search-term*> ...]

6.2 DESCRIPTION

Dump tags for messages matching the given search terms.

Output is to the given filename, if any, or to stdout.

These tags are the only data in the notmuch database that can't be recreated from the messages themselves. The output of notmuch dump is therefore the only critical thing to backup (and much more friendly to incremental backup than the native database files.)

See **notmuch-search-terms(7)** for details of the supported syntax for <search-terms>. With no search terms, a dump of all messages in the database will be generated. A "–" argument instructs notmuch that the remaining arguments are search terms.

Supported options for **dump** include

`--gzip` Compress the output in a format compatible with **gzip(1)**.

`--format=(sup|batch-tag)` Notmuch restore supports two plain text dump formats, both with one message-id per line, followed by a list of tags.

batch-tag

The default **batch-tag** dump format is intended to more robust against malformed message-ids and tags containing whitespace or non-**ascii(7)** characters. Each line has the form

+<*encoded-tag*> +<*encoded-tag*> ... – id:<*quoted-message-id*>

Tags are hex-encoded by replacing every byte not matching the regex **[A-Za-z0-9@=.,_+-]** with **%nn** where nn is the two digit hex encoding. The message ID is a valid Xapian query, quoted using Xapian boolean term quoting rules: if the ID contains whitespace or a close paren or starts with a double quote, it must be enclosed in double quotes and double quotes inside the ID must be doubled. The astute reader will notice this is a special case of the batch input format for **notmuch-tag(1)**; note that the single message-id query is mandatory for **notmuch-restore(1)**.

sup

The **sup** dump file format is specifically chosen to be compatible with the format of files produced by sup-dump. So if you've previously been using sup for mail, then the **notmuch restore** command provides you a way to import all of your tags (or labels as sup calls them). Each line has the following form

<message-id> **(** *<tag>* ... **)**

with zero or more tags are separated by spaces. Note that (malformed) message-ids may contain arbitrary non-null characters. Note also that tags with spaces will not be correctly restored with this format.

--output=<filename> Write output to given file instead of stdout.

6.3 SEE ALSO

notmuch(1), **notmuch-config(1)**, **notmuch-count(1)**, **notmuch-hooks(5)**, **notmuch-insert(1)**, **notmuch-new(1)**, **notmuch-reply(1)**, **notmuch-restore(1)**, **notmuch-search(1)**, **notmuch-search-terms(7)**, **notmuch-show(1)**, **notmuch-tag(1)**

CHAPTER 7

notmuch-emacs

7.1 About this Manual

This manual covers only the Emacs interface to Notmuch. For information on the command line interface, see section "Description" in the Notmuch Manual Pages. To save typing, we will sometimes use *notmuch* in this manual to refer to the Emacs interface to Notmuch. When this distinction is important, we'll refer to the Emacs interface as *notmuch-emacs*.

Notmuch-emacs is highly customizable via the the Emacs customization framework (or just by setting the appropriate variables). We try to point out relevant variables in this manual, but in order to avoid duplication of information, you can usually find the most detailed description in the variables' docstring.

7.2 notmuch-hello

`notmuch-hello` is the main entry point for Notmuch. You can start it with `M-x notmuch` or `M-x notmuch-hello`. The startup screen looks something like the following. There are some hints at the bottom of the screen. There are three main parts to the notmuch-hello screen, discussed below. The **bold** text indicates buttons you can click with a mouse or by positioning the cursor and pressing `<return>`

Welcome to **notmuch** You have 52 messages.

Saved searches: **[edit]**

52 **inbox** 52 **unread**

Search: _____

All tags: **[show]**

Type a search query and hit RET to view matching threads.
Edit saved searches with the `edit` button.
Hit RET or click on a saved search or tag name to view matching threads.
= to refresh this screen. `s` to search messages. `q` to quit.
Customize this page.

You can change the overall appearance of the notmuch-hello screen by customizing the variable notmuch-hello-sections.

7.2.1 notmuch-hello key bindings

<tab> Move to the next widget (button or text entry field)

<backspace> Move to the previous widget.

<return> Activate the current widget.

= Refresh the buffer; mainly update the counts of messages for various saved searches.

G Import mail, See *Importing Mail*

m Compose a message

s Search the notmuch database using *notmuch-search*

v Print notmuch version

q Quit

7.2.2 Saved Searches

Since notmuch is entirely search-based, it's often useful to organize mail around common searches. To facilitate this, the first section of notmuch-hello presents a customizable set of saved searches. Saved searches can also be accessed from anywhere in notmuch by pressing `j` to access notmuch-jump.

The saved searches default to various common searches such as `tag:inbox` to access the inbox and `tag:unread` to access all unread mail, but there are several options for customization:

notmuch-saved-searches The list of saved searches, including names, queries, and additional per-query options.

notmuch-saved-searches-sort-function This variable controls how saved searches should be sorted. A value of `nil` displays the saved searches in the order they are stored in 'notmuch-saved-searches'.

notmuch-column-control Controls the number of columns for displaying saved-searches/tags

7.2.3 Search Box

The search box lets the user enter a Notmuch query. See section "Description" in Notmuch Query Syntax, for more info on Notmuch query syntax. A history of recent searches is also displayed by default. The latter is controlled by the variable notmuch-hello-recent-searches-max.

7.2.4 Known Tags

One special kind of saved search provided by default is for each individual tag defined in the database. This can be controlled via the following variables.

notmuch-hello-tag-list-make-query Control how to construct a search ("virtual folder") from a given tag.

notmuch-hello-hide-tags Which tags not to display at all.

notmuch-column-control Controls the number of columns for displaying saved-searches/tags

7.3 notmuch-search

`notmuch-search-mode` is used to display the results from executing a query via `notmuch-search`. The syntax for these queries is the the same as *Saved Searches*. For details of this syntax see info:notmuch-search-terms

By default the output approximates that of the command line See section "Description" in notmuch search command.

The main purpose of the `notmuch-search-mode` buffer is to act as a menu of results that the user can explore further by pressing `<return>` on the appropriate line.

n, C-n, <down> Move to next line

p, C-p, <up> Move to previous line

<return> Open thread on current line in *notmuch-show* mode

? Display full set of key bindings

The presentation of results can be controlled by the following variables.

notmuch-search-result-format Control how each thread of messages is presented in the `notmuch-show-mode` buffer

notmuch-search-oldest-first Display the oldest threads at the top of the buffer

7.4 notmuch-show

7.5 notmuch-tree

7.6 Global key bindings

Several features are accessible from anywhere in notmuch through the following key bindings:

j Jump to saved searches using notmuch-jump.

7.6.1 notmuch-jump

Saved searches configured through notmuch-saved-searches can include a "shortcut key" that's accessible through notmuch-jump. Pressing `j` anywhere in notmuch followed by the configured shortcut key of a saved search will immediately jump to that saved search. For example, in the default configuration `j i` jumps immediately to the inbox search. When you press `j`, notmuch-jump shows the saved searches and their shortcut keys in the mini-buffer.

7.7 Configuration

7.7.1 Importing Mail

notmuch-poll

notmuch-poll-script

7.7.2 Init File

When Notmuch is loaded, it will read the `notmuch-init-file` (`~/.emacs.d/notmuch-config` by default) file. This is normal Emacs Lisp file and can be used to avoid cluttering your `~/.emacs` with Notmuch stuff. If the file with `.elc`, `.elc.gz`, `.el` or `.el.gz` suffix exist it will be read instead (just one of these, chosen in this order). Most often users create `~/.emacs.d/notmuch-config.el` and just work with it. If Emacs was invoked with the `-q` or `--no-init-file` options, `notmuch-init-file` is not read.

CHAPTER 8

notmuch-hooks

8.1 SYNOPSIS

$DATABASEDIR/.notmuch/hooks/*

8.2 DESCRIPTION

Hooks are scripts (or arbitrary executables or symlinks to such) that notmuch invokes before and after certain actions. These scripts reside in the .notmuch/hooks directory within the database directory and must have executable permissions.

The currently available hooks are described below.

pre-new This hook is invoked by the **new** command before scanning or importing new messages into the database. If this hook exits with a non-zero status, notmuch will abort further processing of the **new** command.

Typically this hook is used for fetching or delivering new mail to be imported into the database.

post-new This hook is invoked by the **new** command after new messages have been imported into the database and initial tags have been applied. The hook will not be run if there have been any errors during the scan or import.

Typically this hook is used to perform additional query-based tagging on the imported messages.

post-insert

This hook is invoked by the **insert** command after the message has been delivered, added to the database, and initial tags have been applied. The hook will not be run if there have been any errors during the message delivery; what is regarded as succesful delivery depends on the `--keep` option.

Typically this hook is used to perform additional query-based tagging on the delivered messages.

8.3 SEE ALSO

notmuch(1), **notmuch-config(1)**, **notmuch-count(1)**, **notmuch-dump(1)**, **notmuch-insert(1)**, **notmuch-new(1)**, **notmuch-reply(1)**, **notmuch-restore(1)**, **notmuch-search(1)**, **notmuch-search-terms(7)**, **notmuch-show(1)**, **notmuch-tag(1)**

CHAPTER 9

notmuch-insert

9.1 SYNOPSIS

notmuch insert [option ...] [+<*tag*>|-<*tag*> ...]

9.2 DESCRIPTION

notmuch insert reads a message from standard input and delivers it into the maildir directory given by configuration option **database.path**, then incorporates the message into the notmuch database. It is an alternative to using a separate tool to deliver the message then running **notmuch new** afterwards.

The new message will be tagged with the tags specified by the **new.tags** configuration option, then by operations specified on the command-line: tags prefixed by '+' are added while those prefixed by '-' are removed.

If the new message is a duplicate of an existing message in the database (it has same Message-ID), it will be added to the maildir folder and notmuch database, but the tags will not be changed.

The **insert** command supports hooks. See **notmuch-hooks(5)** for more details on hooks.

Option arguments must appear before any tag operation arguments. Supported options for **insert** include

--folder=<folder> Deliver the message to the specified folder, relative to the top-level directory given by the value of **database.path**. The default is to deliver to the top-level directory.

--create-folder Try to create the folder named by the `--folder` option, if it does not exist. Otherwise the folder must already exist for mail delivery to succeed.

--keep Keep the message file if indexing fails, and keep the message indexed if applying tags or maildir flag synchronization fails. Ignore these errors and return exit status 0 to indicate succesful mail delivery.

--no-hooks Prevent hooks from being run.

9.3 EXIT STATUS

This command returns exit status 0 on succesful mail delivery, non-zero otherwise. The default is to indicate failed mail delivery on any errors, including message file delivery to the filesystem, message indexing to Notmuch database, changing tags, and synchronizing tags to maildir flags. The `--keep` option may be used to settle for successful message file delivery.

The exit status of the **post-insert** hook does not affect the exit status of the **insert** command.

9.4 SEE ALSO

notmuch(1), **notmuch-config(1)**, **notmuch-count(1)**, **notmuch-dump(1)**, **notmuch-hooks(5)**, **notmuch-reply(1)**, **notmuch-restore(1)**, **notmuch-search(1)**, **notmuch-search-terms(7)**, **notmuch-show(1)**, **notmuch-tag(1)**

CHAPTER 10

notmuch-new

10.1 SYNOPSIS

notmuch new [options]

10.2 DESCRIPTION

Find and import any new messages to the database.

The **new** command scans all sub-directories of the database, performing full-text indexing on new messages that are found. Each new message will automatically be tagged with both the **inbox** and **unread** tags.

You should run **notmuch new** once after first running **notmuch setup** to create the initial database. The first run may take a long time if you have a significant amount of mail (several hundred thousand messages or more). Subsequently, you should run **notmuch new** whenever new mail is delivered and you wish to incorporate it into the database. These subsequent runs will be much quicker than the initial run.

Invoking `notmuch` with no command argument will run **new** if **notmuch setup** has previously been completed, but **notmuch new** has not previously been run.

notmuch new updates tags according to maildir flag changes if the **maildir.synchronize_flags** configuration option is enabled. See **notmuch-config(1)** for details.

The **new** command supports hooks. See **notmuch-hooks(5)** for more details on hooks.

Supported options for **new** include

`--no-hooks` Prevents hooks from being run.

`--quiet` Do not print progress or results.

10.3 SEE ALSO

notmuch(1), **notmuch-config(1)**, **notmuch-count(1)**, **notmuch-dump(1)**, **notmuch-hooks(5)**, **notmuch-insert(1)**, **notmuch-reply(1)**, **notmuch-restore(1)**, **notmuch-search(1)**, **notmuch-search-terms(7)**, **notmuch-show(1)**, **notmuch-tag(1)**

CHAPTER 11

notmuch-reply

11.1 SYNOPSIS

notmuch reply [option ...] <*search-term*> ...

11.2 DESCRIPTION

Constructs a reply template for a set of messages.

To make replying to email easier, **notmuch reply** takes an existing set of messages and constructs a suitable mail template. The Reply-to: header (if any, otherwise From:) is used for the To: address. Unless `--reply-to=sender` is specified, values from the To: and Cc: headers are copied, but not including any of the current user's email addresses (as configured in primary_mail or other_email in the .notmuch-config file) in the recipient list.

It also builds a suitable new subject, including Re: at the front (if not already present), and adding the message IDs of the messages being replied to to the References list and setting the In-Reply-To: field correctly.

Finally, the original contents of the emails are quoted by prefixing each line with '> ' and included in the body.

The resulting message template is output to stdout.

Supported options for **reply** include

`--format=`(**default**|**json**|**sexp**|**headers-only**)

default Includes subject and quoted message body as an RFC 2822 message.

json Produces JSON output containing headers for a reply message and the contents of the original message. This output can be used by a client to create a reply message intelligently.

sexp Produces S-Expression output containing headers for a reply message and the contents of the original message. This output can be used by a client to create a reply message intelligently.

headers-only Only produces In-Reply-To, References, To, Cc, and Bcc headers.

`--format-version=N` Use the specified structured output format version. This is intended for programs that invoke **notmuch(1)** internally. If omitted, the latest supported version will be used.

`--reply-to=`(**all**|**sender**)

all (default) Replies to all addresses.

sender Replies only to the sender. If replying to user's own message (Reply-to: or From: header is one of the user's configured email addresses), try To:, Cc:, and Bcc: headers in this order, and copy values from the first that contains something other than only the user's addresses.

--decrypt Decrypt any MIME encrypted parts found in the selected content (ie. "multipart/encrypted" parts). Status of the decryption will be reported (currently only supported with –format=json and –format=sexp) and on successful decryption the multipart/encrypted part will be replaced by the decrypted content.

Decryption expects a functioning **gpg-agent(1)** to provide any needed credentials. Without one, the decryption will fail.

See **notmuch-search-terms(7)** for details of the supported syntax for <search-terms>.

Note: It is most common to use **notmuch reply** with a search string matching a single message, (such as id:<message-id>), but it can be useful to reply to several messages at once. For example, when a series of patches are sent in a single thread, replying to the entire thread allows for the reply to comment on issues found in multiple patches. The default format supports replying to multiple messages at once, but the JSON and S-Expression formats do not.

11.3 EXIT STATUS

This command supports the following special exit status codes

20 The requested format version is too old.

21 The requested format version is too new.

11.4 SEE ALSO

notmuch(1), **notmuch-config(1)**, **notmuch-count(1)**, **notmuch-dump(1)**, **notmuch-hooks(5)**, **notmuch-insert(1)**, **notmuch-new(1)**, **notmuch-restore(1)**, **notmuch-search(1)**, **notmuch-search-terms(7)**, **notmuch-show(1)**, **notmuch-tag(1)**

notmuch-restore

12.1 SYNOPSIS

notmuch restore [–accumulate] [–format=(auto|batch-tag|sup)] [–input=<*filename*>]

12.2 DESCRIPTION

Restores the tags from the given file (see **notmuch dump**).

The input is read from the given filename, if any, or from stdin.

Supported options for **restore** include

`--accumulate` The union of the existing and new tags is applied, instead of replacing each message's tags as they are read in from the dump file.

`--format=(sup|batch-tag|auto)` Notmuch restore supports two plain text dump formats, with each line specifying a message-id and a set of tags. For details of the actual formats, see **notmuch-dump(1)**.

sup The **sup** dump file format is specifically chosen to be compatible with the format of files produced by sup-dump. So if you've previously been using sup for mail, then the **notmuch restore** command provides you a way to import all of your tags (or labels as sup calls them).

batch-tag The **batch-tag** dump format is intended to more robust against malformed message-ids and tags containing whitespace or non-**ascii(7)** characters. See **notmuch-dump(1)** for details on this format.

notmuch restore updates the maildir flags according to tag changes if the **maildir.synchronize_flags** configuration option is enabled. See **notmuch-config(1)** for details.

auto This option (the default) tries to guess the format from the input. For correctly formed input in either supported format, this heuristic, based the fact that batch-tag format contains no parentheses, should be accurate.

`--input=<filename>` Read input from given file instead of stdin.

12.3 GZIPPED INPUT

notmuch restore will detect if the input is compressed in **gzip(1)** format and automatically decompress it while reading. This detection does not depend on file naming and in particular works for standard input.

12.4 SEE ALSO

notmuch(1), **notmuch-config(1)**, **notmuch-count(1)**, **notmuch-dump(1)**, **notmuch-hooks(5)**, **notmuch-insert(1)**, **notmuch-new(1)**, **notmuch-reply(1)**, **notmuch-search(1)**, **notmuch-search-terms(7)**, **notmuch-show(1)**, **notmuch-tag(1)**

CHAPTER 13

notmuch-search

13.1 SYNOPSIS

notmuch search [*option* ...] <*search-term*> ...

13.2 DESCRIPTION

Search for messages matching the given search terms, and display as results the threads containing the matched messages.

The output consists of one line per thread, giving a thread ID, the date of the newest (or oldest, depending on the sort option) matched message in the thread, the number of matched messages and total messages in the thread, the names of all participants in the thread, and the subject of the newest (or oldest) message.

See **notmuch-search-terms(7)** for details of the supported syntax for <search-terms>.

Supported options for **search** include

--format=(json|sexp|text|text0) Presents the results in either JSON, S-Expressions, newline character separated plain-text (default), or null character separated plain-text (compatible with **xargs(1)** -0 option where available).

--format-version=N Use the specified structured output format version. This is intended for programs that invoke **notmuch(1)** internally. If omitted, the latest supported version will be used.

`--output=(summary|threads|messages|files|tags)`

summary Output a summary of each thread with any message matching the search terms. The summary includes the thread ID, date, the number of messages in the thread (both the number matched and the total number), the authors of the thread and the subject.

threads Output the thread IDs of all threads with any message matching the search terms, either one per line (–format=text), separated by null characters (–format=text0), as a JSON array (–format=json), or an S-Expression list (–format=sexp).

messages Output the message IDs of all messages matching the search terms, either one per line (–format=text), separated by null characters (–format=text0), as a JSON array (–format=json), or as an S-Expression list (–format=sexp).

files Output the filenames of all messages matching the search terms, either one per line (–format=text), separated by null characters (–format=text0), as a JSON array (–format=json), or as an S-Expression list (–format=sexp).

Note that each message may have multiple filenames associated with it. All of them are included in the output (unless limited with the –duplicate=N option). This may be particularly confusing for **folder:** or **path:** searches in a specified directory, as the messages may have duplicates in other directories that are included in the output, although these files alone would not match the search.

tags Output all tags that appear on any message matching the search terms, either one per line (–format=text), separated by null characters (–format=text0), as a JSON array (–format=json), or as an S-Expression list (–format=sexp).

--sort=(newest-first|oldest-first) This option can be used to present results in either chronological order (**oldest-first**) or reverse chronological order (**newest-first**).

Note: The thread order will be distinct between these two options (beyond being simply reversed). When sorting by **oldest-first** the threads will be sorted by the oldest message in each thread, but when sorting by **newest-first** the threads will be sorted by the newest message in each thread.

By default, results will be displayed in reverse chronological order, (that is, the newest results will be displayed first).

--offset=[-]N Skip displaying the first N results. With the leading '-', start at the Nth result from the end.

--limit=N Limit the number of displayed results to N.

--exclude=(true|false|all|flag) A message is called "excluded" if it matches at least one tag in search.tag_exclude that does not appear explicitly in the search terms. This option specifies whether to omit excluded messages in the search process.

The default value, **true**, prevents excluded messages from matching the search terms.

all additionally prevents excluded messages from appearing in displayed results, in effect behaving as though the excluded messages do not exist.

false allows excluded messages to match search terms and appear in displayed results. Excluded messages are still marked in the relevant outputs.

flag only has an effect when `--output=summary`. The output is almost identical to **false**, but the "match count" is the number of matching non-excluded messages in the thread, rather than the number of matching messages.

--duplicate=N For `--output=files`, output the Nth filename associated with each message matching the query (N is 1-based). If N is greater than the number of files associated with the message, don't print anything.

For `--output=messages`, only output message IDs of messages matching the search terms that have at least N filenames associated with them.

Note that this option is orthogonal with the **folder:** search prefix. The prefix matches messages based on filenames. This option filters filenames of the matching messages.

13.3 EXIT STATUS

This command supports the following special exit status codes

20 The requested format version is too old.

21 The requested format version is too new.

13.4 SEE ALSO

notmuch(1), **notmuch-config(1)**, **notmuch-count(1)**, **notmuch-dump(1)**, **notmuch-hooks(5)**, **notmuch-insert(1)**, **notmuch-new(1)**, **notmuch-reply(1)**, **notmuch-restore(1)**, **notmuch-search-terms(7)**, **notmuch-show(1)**, **notmuch-tag(1) notmuch-address(1)**

CHAPTER 14

notmuch-search-terms

14.1 SYNOPSIS

notmuch count [option ...] <*search-term*> ...

notmuch dump [–format=(batch-tag|sup)] [–] [–output=<*file*>] [–] [<*search-term*> ...]

notmuch search [option ...] <*search-term*> ...

notmuch show [option ...] <*search-term*> ...

notmuch tag +<*tag*> ... -<*tag*> [–] <*search-term*> ...

14.2 DESCRIPTION

Several notmuch commands accept a common syntax for search terms.

The search terms can consist of free-form text (and quoted phrases) which will match all messages that contain all of the given terms/phrases in the body, the subject, or any of the sender or recipient headers.

As a special case, a search string consisting of exactly a single asterisk ("*") will match all messages.

In addition to free text, the following prefixes can be used to force terms to match against specific portions of an email, (where <brackets> indicate user-supplied values):

- from:<name-or-address>
- to:<name-or-address>
- subject:<word-or-quoted-phrase>
- attachment:<word>
- mimetype:<word>
- tag:<tag> (or is:<tag>)
- id:<message-id>
- thread:<thread-id>
- folder:<maildir-folder>
- path:<directory-path> or path:<directory-path>/**
- date:<since>..<until>

- lastmod:<initial-revision>..<final-revision>

The **from:** prefix is used to match the name or address of the sender of an email message.

The **to:** prefix is used to match the names or addresses of any recipient of an email message, (whether To, Cc, or Bcc).

Any term prefixed with **subject:** will match only text from the subject of an email. Searching for a phrase in the subject is supported by including quotation marks around the phrase, immediately following **subject:**.

The **attachment:** prefix can be used to search for specific filenames (or extensions) of attachments to email messages.

The **mimetype:** prefix will be used to match text from the content-types of MIME parts within email messages (as specified by the sender).

For **tag:** and **is:** valid tag values include **inbox** and **unread** by default for new messages added by **notmuch new** as well as any other tag values added manually with **notmuch tag**.

For **id:**, message ID values are the literal contents of the Message-ID: header of email messages, but without the '<', '>' delimiters.

The **thread:** prefix can be used with the thread ID values that are generated internally by notmuch (and do not appear in email messages). These thread ID values can be seen in the first column of output from **notmuch search**

The **path:** prefix searches for email messages that are in particular directories within the mail store. The directory must be specified relative to the top-level maildir (and without the leading slash). By default, **path:** matches messages in the specified directory only. The "/**" suffix can be used to match messages in the specified directory and all its subdirectories recursively. **path:""** matches messages in the root of the mail store and, likewise, **path:**** matches all messages.

The **folder:** prefix searches for email messages by maildir or MH folder. For MH-style folders, this is equivalent to **path:**. For maildir, this includes messages in the "new" and "cur" subdirectories. The exact syntax for maildir folders depends on your mail configuration. For maildir++, **folder:""** matches the inbox folder (which is the root in maildir++), other folder names always start with ".", and nested folders are separated by "."s, such as **folder:.classes.topology**. For "file system" maildir, the inbox is typically **folder:INBOX** and nested folders are separated by slashes, such as **folder:classes/topology**.

Both **path:** and **folder:** will find a message if *any* copy of that message is in the specific directory/folder.

The **date:** prefix can be used to restrict the results to only messages within a particular time range (based on the Date: header) with a range syntax of:

date:<since>..<until>

See **DATE AND TIME SEARCH** below for details on the range expression, and supported syntax for <since> and <until> date and time expressions.

The time range can also be specified using timestamps with a syntax of:

<initial-timestamp>..<final-timestamp>

Each timestamp is a number representing the number of seconds since 1970-01-01 00:00:00 UTC.

The **lastmod:** prefix can be used to restrict the result by the database revision number of when messages were last modified (tags were added/removed or filenames changed). This is usually used in conjunction with the **–uuid** argument to **notmuch search** to find messages that have changed since an earlier query.

14.2.1 Operators

In addition to individual terms, multiple terms can be combined with Boolean operators (**and**, **or**, **not**, and **xor**). Each term in the query will be implicitly connected by a logical AND if no explicit operator is provided (except that terms with a common prefix will be implicitly combined with OR). The shorthand '-<term>' can be used for 'not <term>' but unfortunately this does not work at the start of an expression. Parentheses can also be used to control the combination

of the Boolean operators, but will have to be protected from interpretation by the shell, (such as by putting quotation marks around any parenthesized expression).

In addition to the standard boolean operators, Xapian provides several operators specific to text searching.

```
notmuch search term1 NEAR term2
```

will return results where term1 is within 10 words of term2. The threshold can be set like this:

```
notmuch search term1 NEAR/2 term2
```

The search

```
notmuch search term1 ADJ term2
```

will return results where term1 is within 10 words of term2, but in the same order as in the query. The threshold can be set the same as with NEAR:

```
notmuch search term1 ADJ/7 term2
```

14.2.2 Stemming

Stemming in notmuch means that these searches

```
notmuch search detailed
notmuch search details
notmuch search detail
```

will all return identical results, because Xapian first "reduces" the term to the common stem (here 'detail') and then performs the search.

There are two ways to turn this off: a search for a capitalized word will be performed unstemmed, so that one can search for "John" and not get results for "Johnson"; phrase searches are also unstemmed (see below for details). Stemming is currently only supported for English. Searches for words in other languages will be performed unstemmed.

14.2.3 Wildcards

It is possible to use a trailing '*' as a wildcard. A search for 'wildc*' will match 'wildcard', 'wildcat', etc.

14.2.4 Boolean and Probabilistic Prefixes

Xapian (and hence notmuch) prefixes are either **boolean**, supporting exact matches like "tag:inbox" or **probabilistic**, supporting a more flexible **term** based searching. The prefixes currently supported by notmuch are as follows.

Boolean	Probabilistic
tag: id: **thread: folder: path:**	**from: to:** **subject: attachment: mimetype:**

14.2.5 Terms and phrases

In general Xapian distinguishes between lists of terms and **phrases**. Phrases are indicated by double quotes (but beware you probably need to protect those from your shell) and insist that those unstemmed words occur in that order. One useful, but initially surprising feature is that the following are equivalant ways to write the same phrase.

- "a list of words"

- a-list-of-words
- a/list/of/words
- a.list.of.words

Both parenthesised lists of terms and quoted phrases are ok with probabilisitic prefixes such as **to:**, **from:**, and **subject:**. In particular

```
subject:(pizza free)
```

is equivalent to

```
subject:pizza and subject:free
```

Both of these will match a subject "Free Delicious Pizza" while

```
subject:"pizza free"
```

will not.

14.3 DATE AND TIME SEARCH

notmuch understands a variety of standard and natural ways of expressing dates and times, both in absolute terms ("2012-10-24") and in relative terms ("yesterday"). Any number of relative terms can be combined ("1 hour 25 minutes") and an absolute date/time can be combined with relative terms to further adjust it. A non-exhaustive description of the syntax supported for absolute and relative terms is given below.

14.3.1 The range expression

date:<since>..<until>

The above expression restricts the results to only messages from <since> to <until>, based on the Date: header.

<since> and <until> can describe imprecise times, such as "yesterday". In this case, <since> is taken as the earliest time it could describe (the beginning of yesterday) and <until> is taken as the latest time it could describe (the end of yesterday). Similarly, date:january..february matches from the beginning of January to the end of February.

date:<expr>..! can be used as a shorthand for date:<expr>..<expr>. The expansion takes place before interpretation, and thus, for example, date:monday..! matches from the beginning of Monday until the end of Monday. (Note that entering date:<expr> without "..", for example date:yesterday, won't work, as it's not interpreted as a range expression at all. Again, use date:yesterday..!)

Currently, we do not support spaces in range expressions. You can replace the spaces with '_', or (in most cases) '-', or (in some cases) leave the spaces out altogether. Examples in this man page use spaces for clarity.

Open-ended ranges are supported (since Xapian 1.2.1), i.e. it's possible to specify date:..<until> or date:<since>.. to not limit the start or end time, respectively. Pre-1.2.1 Xapian does not report an error on open ended ranges, but it does not work as expected either.

14.3.2 Relative date and time

[N|number] (years|months|weeks|days|hours|hrs|minutes|mins|seconds|secs) [...]

All refer to past, can be repeated and will be accumulated.

Units can be abbreviated to any length, with the otherwise ambiguous single m being m for minutes and M for months.

Number can also be written out one, two, ..., ten, dozen, hundred. Additionally, the unit may be preceded by “last” or “this” (e.g., “last week” or “this month”).

When combined with absolute date and time, the relative date and time specification will be relative from the specified absolute date and time.

Examples: 5M2d, two weeks

14.3.3 Supported absolute time formats

- H[H]:MM[:SS] [(am|a.m.|pm|p.m.)]
- H[H] (am|a.m.|pm|p.m.)
- HHMMSS
- now
- noon
- midnight
- Examples: 17:05, 5pm

14.3.4 Supported absolute date formats

- YYYY-MM[-DD]
- DD-MM[-[YY]YY]
- MM-YYYY
- M[M]/D[D][/[YY]YY]
- M[M]/YYYY
- D[D].M[M][.[YY]YY]
- D[D][(st|nd|rd|th)] Mon[thname] [YYYY]
- Mon[thname] D[D][(st|nd|rd|th)] [YYYY]
- Wee[kday]

Month names can be abbreviated at three or more characters.

Weekday names can be abbreviated at three or more characters.

Examples: 2012-07-31, 31-07-2012, 7/31/2012, August 3

14.3.5 Time zones

- (+|-)HH:MM
- (+|-)HH[MM]

Some time zone codes, e.g. UTC, EET.

14.4 SEE ALSO

notmuch(1), **notmuch-config(1)**, **notmuch-count(1)**, **notmuch-dump(1)**, **notmuch-hooks(5)**, **notmuch-insert(1)**, **notmuch-new(1)**, **notmuch-reply(1)**, **notmuch-restore(1)**, **notmuch-search(1)**, **notmuch-show(1)**, **notmuch-tag(1)**

CHAPTER 15

notmuch-show

15.1 SYNOPSIS

notmuch show [*option* ...] <*search-term*> ...

15.2 DESCRIPTION

Shows all messages matching the search terms.

See **notmuch-search-terms(7)** for details of the supported syntax for <search-terms>.

The messages will be grouped and sorted based on the threading (all replies to a particular message will appear immediately after that message in date order). The output is not indented by default, but depth tags are printed so that proper indentation can be performed by a post-processor (such as the emacs interface to notmuch).

Supported options for **show** include

--entire-thread=(true|false) If true, **notmuch show** outputs all messages in the thread of any message matching the search terms; if false, it outputs only the matching messages. For `--format=json` and `--format=sexp` this defaults to true. For other formats, this defaults to false.

`--format=(text|json|sexp|mbox|raw)`

text (default for messages) The default plain-text format has all text-content MIME parts decoded. Various components in the output, (**message**, **header**, **body**, **attachment**, and MIME **part**), will be delimited by easily-parsed markers. Each marker consists of a Control-L character (ASCII decimal 12), the name of the marker, and then either an opening or closing brace, ('{' or '}'), to either open or close the component. For a multipart MIME message, these parts will be nested.

json The output is formatted with Javascript Object Notation (JSON). This format is more robust than the text format for automated processing. The nested structure of multipart MIME messages is reflected in nested JSON output. By default JSON output includes all messages in a matching thread; that is, by default, `--format=json` sets `--entire-thread`. The caller can disable this behaviour by setting `--entire-thread=false`. The JSON output is always encoded as UTF-8 and any message content included in the output will be charset-converted to UTF-8.

sexp The output is formatted as the Lisp s-expression (sexp) equivalent of the JSON format above. Objects are formatted as property lists whose keys are keywords (symbols preceded

by a colon). True is formatted as `t` and both false and null are formatted as `nil`. As for JSON, the s-expression output is always encoded as UTF-8.

mbox All matching messages are output in the traditional, Unix mbox format with each message being prefixed by a line beginning with "From " and a blank line separating each message. Lines in the message content beginning with "From " (preceded by zero or more '>' characters) have an additional '>' character added. This reversible escaping is termed "mboxrd" format and described in detail here:

http://homepage.ntlworld.com/jonathan.deboynepollard/FGA/mail-mbox-formats.html

raw (default if –part is given) Write the raw bytes of the given MIME part of a message to standard out. For this format, it is an error to specify a query that matches more than one message.

If the specified part is a leaf part, this outputs the body of the part after performing content transfer decoding (but no charset conversion). This is suitable for saving attachments, for example.

For a multipart or message part, the output includes the part headers as well as the body (including all child parts). No decoding is performed because multipart and message parts cannot have non-trivial content transfer encoding. Consumers of this may need to implement MIME decoding and similar functions.

--format-version=N Use the specified structured output format version. This is intended for programs that invoke **notmuch(1)** internally. If omitted, the latest supported version will be used.

--part=N Output the single decoded MIME part N of a single message. The search terms must match only a single message. Message parts are numbered in a depth-first walk of the message MIME structure, and are identified in the 'json', 'sexp' or 'text' output formats.

Note that even a message with no MIME structure or a single body part still has two MIME parts: part 0 is the whole message (headers and body) and part 1 is just the body.

--verify Compute and report the validity of any MIME cryptographic signatures found in the selected content (ie. "multipart/signed" parts). Status of the signature will be reported (currently only supported with –format=json and –format=sexp), and the multipart/signed part will be replaced by the signed data.

--decrypt Decrypt any MIME encrypted parts found in the selected content (ie. "multipart/encrypted" parts). Status of the decryption will be reported (currently only supported with –format=json and –format=sexp) and on successful decryption the multipart/encrypted part will be replaced by the decrypted content.

Decryption expects a functioning **gpg-agent(1)** to provide any needed credentials. Without one, the decryption will fail.

Implies –verify.

--exclude=(true|false) Specify whether to omit threads only matching search.tag_exclude from the search results (the default) or not. In either case the excluded message will be marked with the exclude flag (except when output=mbox when there is nowhere to put the flag).

If –entire-thread is specified then complete threads are returned regardless (with the excluded flag being set when appropriate) but threads that only match in an excluded message are not returned when `--exclude=true`.

The default is `--exclude=true`.

--body=(true|false) If true (the default) **notmuch show** includes the bodies of the messages in the output; if false, bodies are omitted. `--body=false` is only implemented for the json and sexp formats and it is incompatible with `--part > 0`.

This is useful if the caller only needs the headers as body-less output is much faster and substantially smaller.

--include-html Include "text/html" parts as part of the output (currently only supported with –format=json and –format=sexp). By default, unless `--part=N` is used to select a specific part or `--include-html` is used to include all "text/html" parts, no part with content type "text/html" is included in the output.

A common use of **notmuch show** is to display a single thread of email messages. For this, use a search term of "thread:<thread-id>" as can be seen in the first column of output from the **notmuch search** command.

15.3 EXIT STATUS

This command supports the following special exit status codes

20 The requested format version is too old.

21 The requested format version is too new.

15.4 SEE ALSO

notmuch(1), **notmuch-config(1)**, **notmuch-count(1)**, **notmuch-dump(1)**, **notmuch-hooks(5)**, **notmuch-insert(1)**, **notmuch-new(1)**, **notmuch-reply(1)**, **notmuch-restore(1)**, **notmuch-search(1)**, **notmuch-search-terms(7)**, **notmuch-tag(1)**

CHAPTER 16

notmuch-tag

16.1 SYNOPSIS

notmuch tag [options ...] +<*tag*>|-<*tag*> [–] <*search-term*> ...

notmuch tag –batch [–input=<*filename*>]

16.2 DESCRIPTION

Add/remove tags for all messages matching the search terms.

See **notmuch-search-terms(7)** for details of the supported syntax for <*search-term*>.

Tags prefixed by '+' are added while those prefixed by '-' are removed. For each message, tag changes are applied in the order they appear on the command line.

The beginning of the search terms is recognized by the first argument that begins with neither '+' nor '-'. Support for an initial search term beginning with '+' or '-' is provided by allowing the user to specify a "–" argument to separate the tags from the search terms.

notmuch tag updates the maildir flags according to tag changes if the **maildir.synchronize_flags** configuration option is enabled. See **notmuch-config(1)** for details.

Supported options for **tag** include

--remove-all Remove all tags from each message matching the search terms before applying the tag changes appearing on the command line. This means setting the tags of each message to the tags to be added. If there are no tags to be added, the messages will have no tags.

--batch Read batch tagging operations from a file (stdin by default). This is more efficient than repeated **notmuch tag** invocations. See *TAG FILE FORMAT* below for the input format. This option is not compatible with specifying tagging on the command line.

--input=<filename> Read input from given file, instead of from stdin. Implies `--batch`.

16.3 TAG FILE FORMAT

The input must consist of lines of the format:

+<*tag*>|-<*tag*> [...] [–] <*query*>

Each line is interpreted similarly to **notmuch tag** command line arguments. The delimiter is one or more spaces ' '. Any characters in <*tag*> **may** be hex-encoded with %NN where NN is the hexadecimal value of the character. To hex-encode a character with a multi-byte UTF-8 encoding, hex-encode each byte. Any spaces in <tag> **must** be hex-encoded as *%*20. Any characters that are not part of <*tag*> **must not** be hex-encoded.

In the future tag:"tag with spaces" style quoting may be supported for <*tag*> as well; for this reason all double quote characters in <*tag*> **should** be hex-encoded.

The <*query*> should be quoted using Xapian boolean term quoting rules: if a term contains whitespace or a close paren or starts with a double quote, it must be enclosed in double quotes (not including any prefix) and double quotes inside the term must be doubled (see below for examples).

Leading and trailing space ' ' is ignored. Empty lines and lines beginning with '#' are ignored.

16.3.1 EXAMPLE

The following shows a valid input to batch tagging. Note that only the isolated '*' acts as a wildcard. Also note the two different quotings of the tag **space in tags**

```
+winner *
+foo::bar%25 -- (One and Two) or (One and tag:winner)
+found::it -- tag:foo::bar%
# ignore this line and the next

+space%20in%20tags -- Two
# add tag '(tags)', among other stunts.
+crazy{ +(tags) +&are +#possible\ -- tag:"space in tags"
+match*crazy -- tag:crazy{
+some_tag -- id:"this is ""nauty)"""
```

16.4 SEE ALSO

notmuch(1), **notmuch-config(1)**, **notmuch-count(1)**, **notmuch-dump(1)**, **notmuch-hooks(5)**, **notmuch-insert(1)**, **notmuch-new(1)**, **notmuch-reply(1)**, **notmuch-restore(1)**, **notmuch-search(1)**, **notmuch-search-terms(7)**, **notmuch-show(1)**,

CHAPTER 17

Indices and tables

- genindex
- modindex
- search

N