Jaro Mail 4.2

Denis Roio <jaromil @ dyne.org>

April 15, 2016

Contents

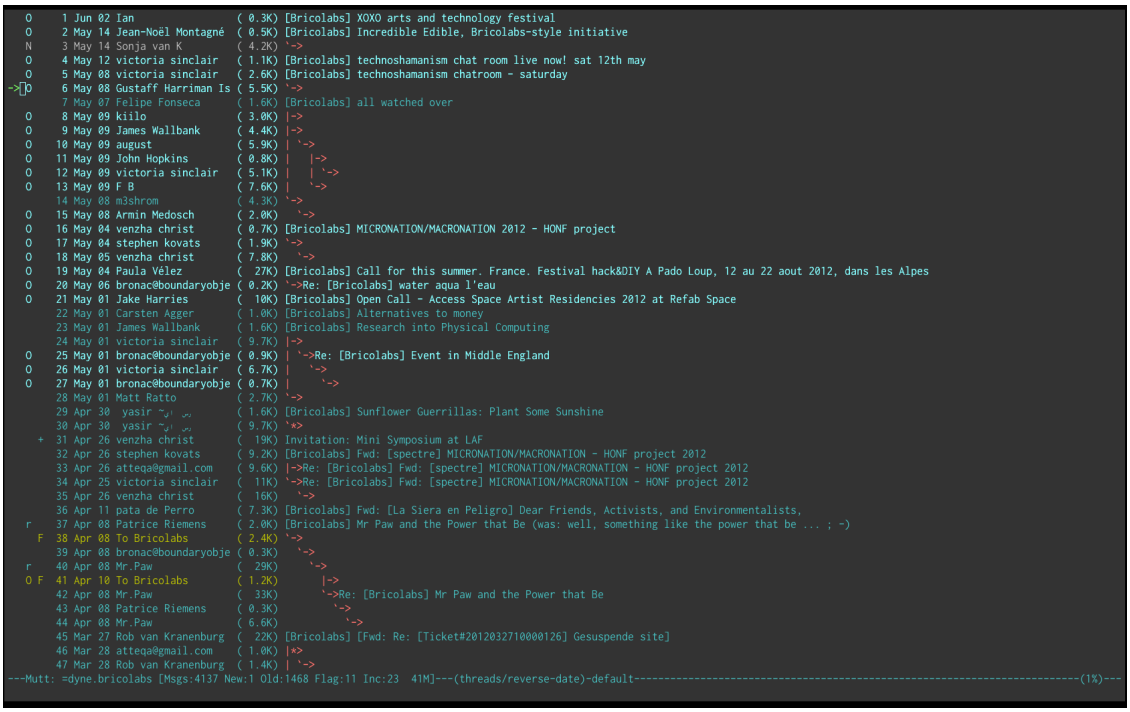
1 Introduction

Jaro Mail is an integrated suite of interoperable tools to manage e-mail communication in a private and efficient way, without relying too much on on-line services, in fact encouraging users to store their email locally.

Rather than reinventing the wheel, this suite reuses existing free and open source tools and protocols and is mainly targeted for GNU/Linux/BSD desktop usage.

This manual illustrates the usage of Jaro Mail. The newest version of this manual is made available on `http://files.dyne.org/jaromail/jaromail-manual.pdf`

1.1 Features



- Minimalistic and efficient interface with message threading
- Targets intensive usage of e-mails and mailinglists
- Stores e-mails locally in a reliable format (maildir)
- Integrates whitelisting and blacklisting, local and remote
- Can do search and backup by advanced expressions
- Automatically generates filter rules (sieve)
- Imports and exports VCard contacts to addressbook
- Computes and shows statistics on mail traffic
- Facilitates sending anonymous emails (Mixmaster)
- Encrypted password storage using OS native keyrings
- Advanced maildir tools (merge, backup, address extraction)
- Defers connections for off-line operations
- Checks SSL/TLS certificates when fetching and sending mails

- Supports strong encryption messaging (GnuPG)
- Multi platform: GNU/Linux/BSD, Apple/OSX
- Old school, used by its author for the past 10 years

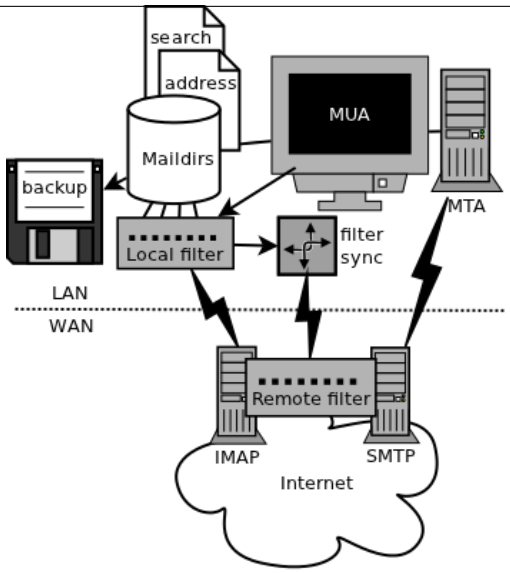
1.2 Vision

The internet offers plenty of free services, on the wave of the Web2.0 fuzz and the community boom, while all private informations are hosted on servers owned by global corporations and monopolies.

It is important to keep in mind that no-one else better than you can ensure the privacy of your personal data. Server hosted services and web integrated technologies gather all data into huge information pools that are made available to established economical and cultural regimes.

The vision behind this software is that of sharing a simple and consistent way to operate e-mail communication with tools that are available on most platforms and can be as well used remotely over a secure shell connection.

Jaro Mail aims to facilitate the task of downloading and storing e-mail archives off-line in a way that they can be still accessible in more than 10 years time and independently of any software. Nowadays many users have the habit of keeping all their e-mails on servers, accessing them through an often clumsy web interface, while downloading them can free space and improve their privacy.



2 Diagram

A little diagram that clarifies a bit where do we place the components and actions involved in managing one's email communication:

Acronym	Function	Software
MUA	Mail User Agent	Mutt
MTA	Mail Transport Agent	Fetchmail
LDA	Local Delivery Agent	Jaro Mail
MDA	Remote Delivery Agent	Sieve
SMTP	Mail delivery agent	MSmtp
ANON	Anonymous delivery	MixMaster
	Search engine	Notmuch
	Addressbook	ABook
GPG	Cryptographic Agent	GnuPG
STORE	Cryptographic Storage	Tomb

3 Setup

3.1 Build

Jaro Mail needs to be built on GNU/Linux systems.

For Apple/OSX users it comes in a pre-compiled bundle.

3.1.1 GNU/Linux

Some dependencies are needed in order to build this software. The Makefile for GNU/Linux configures the build environment automatically on Debian and Fedora systems, using their packaging to install all needed packages.

The dependencies to be installed on the system for JaroMail are

- to **build**: gcc bison flex make autoconf automake sqlite3 libglib2.0-dev libgnome-keyring-dev
- to **run**: fetchmail msmtp mutt pinentry abook wipe notmuch alot

  To install all needed components (done automatically, requires root):

```
make
```

Once compiled then **make install** will put all JaroMail files in **/usr/local/share/jaromail**.

3.1.2 Apple/OSX

Apple/OSX users that have no experience in building software can obtain a pre-built universal binary from our download zone on `http://files.dyne.org/jaromail/binary`

One can simply drag JaroMail into Applications. When started JaroMail opens a Terminal window preconfigured with its environment, to activate it for any terminal add this to '~/.profile':

```
export PATH=/Applications/JaroMail.app/Contents/Resources/jaro/bin:$PATH
```

3.2 Install

Installing Jaro Mail once all dependencies are build is fairly easy: make a directory where all the emails and settings needs to be, change to the directory and init the environment:

```
mkdir $HOME/Mail
cd $HOME/Mail
jaro init
```

Every installation of Jaro Mail is fully reentrant, meaning the directory where it gets initialised contains all maildirs, configurations, filters, whitelist, addressbooks and other necessary files.

A single user can have multiple Jaro Mail installations to permit the complete separation of E-Mail identities.

If called from outside the installation directory, the **jaro** command will use the environmental variable **$JAROMAILDIR** to find out the active installation being in use. If one is using a different installation path then should first change that, i.e:

```
export JAROMAILDIR=$HOME/OtherIdentities/Luther/Mail
```

4 Configuration

The place where Jaro Mail is installed (**$HOME/Mail** by default) contains all configuration files.

For Apple/OSX users, this directory is inside their **$HOME/Library**, then **Application Support** and then **JaroMail**.

From now own, we will call this place the **Mail directory**.

Inside the **Mail directory** are all needed configurations to operate JaroMail. Such configurations are in readable plain text files that can be edited using any editor. Inside them there are comments to explain the settings: all comment lines start by '#' and will be ignored by JaroMail.

The most important files to start configuring are:

- Identity.txt : set up the way your email identity appear to others
- Accounts/default.txt : main account configuration (there can be more)
- Aliases.txt : more email addresses one may receive on the configured accounts
- Filters.txt : Full set of mailinglist sorting rules
- Applications.txt : mime type associations to programs used to open attachments
- Mutt.txt : mutt specific custom configurations

4.1 Send and receive mail

Inside the Mail directory is found the folder **Accounts** with brief instructions and default templates to fill with Imap and Smtp account configurations to fetch mail. A default template will be found in fresh installations: **Accounts/default.txt**. The configuration can be edited with one's favourite text editor, the format of the file is pretty self-explanatory.

It is possible to have more than one account (simply as a new file in the Accounts/ directory) and in fact when retreiving e-mails using the **jaro fetch** command all accounts will be processed, unless one is explicitly selected using the **-a** commandline option.

The file **Identity.txt** is also found in the Mail directory and it contains basic settings on the published user identity (From: field) and any other Mutt specific configuration directives, such as custom headers appearing in composed e-mails and the default GPG[1] key to be used when signing and encrypting them. For more information about the vast amount of configurations that are supported please refer to the Mutt documentation[2]

4.2 Filter mail

In the mail directory a file named **Filters.txt** can be created and filled in with rules referencing the contents of the **From:** or **To:** fields of each e-mail that is fetched. The mails matching will be then saved in the indicated maildirs (created if not existing) to keep a bit of order, especially useful for mailinglist users.

The format of the filters configurarion is pretty easy and self explanatory, an example is found in the appendix of this manual.

[1] GPG stands for GNU Privacy Guard, a system to securely encrypt and decrypt messages and files so that noone can read their content, even when intercepting the communication.

[2] The Mutt configuration manual is found on `http://www.mutt.org/doc/manual` or simply typing 'man mutt' in a console terminal.

5 Organization

One of the main goals for Jaro Mail is to organize the e-mail workflow so that one's attention is dedicated to important communications, rather than being constantly distracted by various degrees of spam and the need to weed it out of the mailbox. This ambitious task is pursued by realizing an integrated approach consisting of flexible whitelisting and the distinction between mails from known people and the rest.

5.1 Folders

First lets start with a categorization of the standard maildirs and a brief description for each. This information is **very important** to understand how Jaro Mail works: these maildirs are standard in Jaro Mail, here they are listed in order of priority

Folder	What goes in there
known	Mails whose sender is known (Whitelist)
priv	Unknown sender, we are among explicit recipients
unsorted	Unknown sender, we are not among recipients
unsorted.ml	From a mailinglist that we haven't filtered yet
zz.blacklist	Mails whose sender is not desired (Blacklist)
zz.spam	Mails that are tagged as spam (server-side)
zz.bounces	Mail bounces like mailman and similar

The advantage using such a folder organization is that every time we open up the mail reader we will be presented with something we are likely to be most interested in (known people replying our mails) and progressively, as we will have the time to scroll through, mails from "new people" or mass mailings of sort.

This setup is handy especially considering it produces **sieve** filters that can be uploaded to mail servers and processed server-side. Imagine having your email on a fixed computer, but occasionally checking it from a mobile phone: server-side filtering will save you time by presenting a clean INBOX of whitelisted contacts for the mobile phone use.

Please note this organization does not includes spam, which is supposedly weeded out on the server via spamlists: White/Blacklisting has more to do with our own selection of content sources than with the generic protection from random pieces of information.

At last, anything that is matched by filters configured in **Filters.txt** will be saved into the named maildir, whose name can be freely choosen.

5.2 Whitelist

The way whitelisting works if quite crucial to this setup and, at the same time, is fairly simple since it does not include any automatic detection, learning filters, Markov chains or Bayesian A/I. We believe the user should be in full control of prioritizing communication channels and at the same time constantly able to tweak the setup in an easy way.

To whitelist an address is sufficient to send it an e-mail: at the moment the message is sent Jaro Mail will remember the destination address and prioritize all messages coming back from it. This we call implicit whitelisting.

To explicitly whitelist an address from inside the mail reader index press [**a**] while selecting an email, this will add in the whitelist the sender address (From: header). If you want to add all addresses reached by the mail (From: To: and Cc: fields) use the same letter capitalized pressing shift [**A**].

All addresses selected this way will have the privilege of ending up in your **known/** folder, plus their name and e-mail will be completed automatically when composing a new email, pressing the **Tab** key while indicating them among the recipients.

5.3 Blacklist

To blacklist an address instead one can use the [**z**] key while an e-mail is selected on the index: the sender indicated in the From: field will be downgraded to the very bottom of your priorities, closes to spam than the rest, the most infamous **zz.blacklist/** folder.

5.4 Organization In Brief

Below a recapitulation of keys related to the white and blacklisting functionality, to be used in the e-mail index or when an e-mail is open inside the mail user agent:

List	Key	Function	Fields
White	**a**	Add the sender address	From:
White	**A** (shift)	Add all addresses	From: To: Cc:
Black	**z**	Blacklist the sender	From:
Black	**Z** (shift)	Blacklist all addresses	From: To: Cc:

6 Workflow

This section goes through a scenario of simple usage for Jaro Mail

6.1 Fetch and read your mail at home

As you acces your computer where Jaro Mail has been configured, you can open a Terminal and type:

```
jaro fetch
```

This will download all new mails.

If you have configured **fetchall** among the imap account options, then will delete them from the server, freeing online space.

If you have configured the **keep** option, which is the default, Jaro Mail will only download the email that you have not yet read and in any case it won't delete anything from the server. Remove the **keep** option to delete on the server all emails that are downloaded.

```
jaro
```

This will launch mutt on the first folder containing unread emails, starting from the **known** folder, then **priv**, then all the destinations specified by **Filters.txt** exactly in the ascending order they are listed in that configuration file..

From there on, pressing **=** or **c** you can change to other folders and your **unsorted** and **unsorted.ml** mails.

6.2 Write a new mail

If you like to write a mail to someone, hit **m** and write the recipient address, you will be then asked about any additional Cc: recipients.

If you don't remember the emails of the recipients, you can just type their name or parts of the email you remember, then press the [**Tab**] key for completion. A list of addresses may popup with matches found in your whitelist addressbook to help remind who are you looking for.

The email is composed using a special Vim configuration that facilitates justifying text to 72 columns using [**ctrl-j**]. After composing the email you will be able to review it and change:

- the From: field using [**ESC f**]
- the recipient in the To: field using [**t**]
- the recipients in the Cc: field using [**c**]
- the subject string using [**s**]

You'll also be able to add more attachments by pressing **a** and use the arrow keys to move over the existing ones and delete them using [**D**] (please note that is a uppercase D, because lowercase d will just add a description for the attachment).

At last, when ready, pressing **y** will queue the email into the outbox, ready for sending.

One can review at any time the sending queue, which is just another maildir named **outbox**

```
jaro outbox
```

Mails can be deleted from this view using [**d**] or edited using [**e**] which will allow tweaking of both the header and body of the email.

Once sure the outbox contains all what needs to be sent, make sure the computer is connected to the Internet and issue the **send** command:

```
jaro send
```

Jaro Mail will send all emails in outbox, one by one, listing their recipients and size while doing so. If successful, mails will be removed from the outbox and put into the **sent** folder, which can be accessed from inside mutt or with the command **jaro open sent**.

6.3 Write a new email from the commandline

Jaro Mail supports a lot of commandline operations based on stdin/stdout pipes, which makes it pretty easy to use in scripts that send emails and attachments.

If you have written a plain-text email using your favorite editor, you can send it quickly using the commandline: save the email into a txt file and then pipe it into **jaro compose** followed by a list of recipients and, optionally a list of filenames to attach. For example:

```
cat Greetings.txt | jaro compose friends@dyne.org picture01.jpg jingle02.mp3 ~/myicons/*
```

The command above may send an email with various separate attachments (using MIME encapsulation): a picture, an hopefully small audio file and a list of icons which are all the files contained into the myicons/ directory. In this case the recipient will be friends@dyne.org, but may be any other email address found on the commandline in any position.

Once executed you will find this email in **jaro outbox**, ready to be reviewed and sent with **jaro send**.

6.4 Reply messages

While browsing through the index of emails in various folders, one can reply any of them just by pressing the [**r**] key, which will ask if the original message should be quoted and then open your favorite editor to compose your text.

If the email you are replying has been sent to multiple recipients (for instance using multiple addresses in the Cc: or From: fields) they will all be included, but you will have the possibility to exclude them by hand, editing the Cc: field. To remove them all at once use [**ctrl-k**] just like deliting a line on the terminal.

It is also possible to forward a message to someone else than the sender, for instance to submit it to his or her attention, or that of a mailinglist. To do that, you can use the [**f**] key which will present you with the full message and the possibility to write something on top of it, to describe its contents to its new recipients. Forwards include all attachments and are sent as attachments themselves, but this behavious can be changed as a confirmation to "send forward as attach" is asked.

6.5 Peek without downloading anything

If you are around and like to see your new mails without downloading them, then you can use the **peek** function:

```
jaro peek
```

This will open the default configured IMAP account and folder over SSL protocol (securing the data transfer) and allow you to browse, read and reply your emails without downloading them.

Using peek you can reply and even delete emails, but be careful since what you delete here will be removed from the server and won't be there when you download it from home.

This functionality can be also very useful if you are from a slow connection and need to delete some email that is clogging it and that you are not able to download because of its size.

The peek command will automatically open the INBOX, but also other remote imap folders can be specified, like for instance **priv** or **unsorted** if whitelisting is also setup server-side (the sieve filters generated by Jaro Mail need to be uploaded on the server). To have a list of imap folders on the server a command is also available:

```
jaro imap listfolders
```

Will list on the terminal all folders found on the imap account, one per line.

6.6 Save important emails for later

Sometimes one can be on the rush while reading emails (local or via imap) and flagging them as important can be useful to keep focus on priorities. In some cases it is very useful to save such important messages locally for later reference, for instance in a folder keeping messages that need to be remembered and that will constitute a kind of TODO list (a'la GTD).

Jaro Mail implements such functionalities: by pressing the [**F**] key (uppercase) one can flag an email, which will turn bright-green in the index. In addition to that there is a folder called **remember/** where one can copy emails on the fly using the [**R**] key (uppercase) any time. Messages will be duplicated into the remember folder (which of course can be opened with the command **jaro remember**) so they can also be edited with annotations on the task they refer to, for instance using the [**e**] key, without affecting the original message.

6.7 Workflow in brief

Below a recapitulation of keys commonly used in our workflow

Key	Function
m	Compose a new message
Tab	Complete addresses and folders input
r	Reply to the sender of a message
d	Delete a message
y	Send a message (queue in outbox)
f	Forward a message to new recipients
=	List all filtered maildir folders
c	Change to another folder
F	Flag a message as important
R	Copy a message to remember
s	Move a message to another folder
C	Copy a message to another folder

7 Searching

Searching across all your emails it is as important as demanding of a task. Jaro Mail implements it using Notmuch which is relying on the Xapian search engine, completely relying on local computations made on your machine, there is no data at all being communicated on-line.

To index and tag all your emails that are locally archived in Jaro Mail use:

```
jaro index
```

This will take a while and increase the size of the storage of about one sixth of its total occupation, but will definitely come useful when in need of searching rapidly across all available emails. To run a search for emails containing the '*open source*' string, do

```
jaro search open source
```

To search for all emails containing this string and dated between now and the last two weeks, do

```
jaro search open source date:2w..
```

The search command prints out a list of found filenames which may be useful to a script, but less useful to a human. In order to read a quick summary of the emails found it is possible to pipe the results into the **headers** command which will print out date, sender and subject of each file

```
jaro search open source date:2w.. | jaro headers
```

Searching has also an interactive interface called **alot** which pops up to show search results and browse through them, refine the terms and in general operate on emails with the usual keys. One can also reply to emails directly from alot:

```
jaro alot search expression strings folder:known
```

To restrict the search to a single folder, one can use the **folder:** prefix to search terms. Tags can be used also with **tag:** as well dates can be specified with ranges using **date:**. Consecutive string expressions are aloud to refine the search match, connected with logical and/or, plus also the header to search can be indicated, as for instance **from:** or **to:**. Read more about this below in the *Search term* and *Date and time search* sections (extracts from the **notmuch-search-terms** manpage) and on the notmuch webpage at `http://notmuchmail.org`

With the **addr** command the search will be run on the whitelist addressbook entries instead of actual email contents.

```
jaro addr joe
```

Will list all addresses matching the string 'joe' inside the *whitelist* addressbook. Also the blacklist can be searched this way adding the switch **-l blacklist**.

7.1 Combining terms

In addition to individual terms, multiple terms can be combined with Boolean operators (**and**, **or**, **not** , etc.). Each term in the query will be implicitly connected by a logical AND if no explicit operator is provided.

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).

7.2 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>
tag:<tag> (or is:<tag>)
id:<message-id>
thread:<thread-id>
folder:<directory-path>
date:<since>..<until>
```

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.

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 *folder:* prefix can be used to search for email message files that are contained within particular directories within the mail store. If the same email message has multiple message files associated with it, it's sufficient for a match that at least one of the files is contained within a matching directory. Only the directory components below the top-level mail database path are available to be searched.

7.3 Date and time search

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

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>
```

The syntax *<initial-timestamp>..<final-timestamp>* can be represented using the number of seconds since 1970-01-01 00:00:00 UTC.

The search syntax also 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.

7.3.1 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.

Currently, we do not support spaces in range expressions. You can replace the spaces with '$\backslash'$,$or(inmostcases)'-',or(insomecases)lea$

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.

Entering date:expr without ".." (for example date:yesterday) won't work, as it's not interpreted as a range expression at all. You can achieve the expected result by duplicating the expr both sides of ".." (for example date:yesterday..yesterday).

7.3.2 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
```

7.3.3 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
```

7.3.4 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
```

7.3.5 Time zones

`(+|-)HH:MM`

`(+|-)HH[MM]`

Some time zone codes.

Examples:

```
UTC
EET
```

8 Compute and visualize statistics

The **stats** command is useful to quickly visualize statistics regarding folder usage as well the frequency of emails found in a stream from stdin. Such streams can be produced by the **search** and **extract** commands for instance and passed to stats in order to have a more graphical (yet ASCII based) visualization of results.

For example lets visualize the frequency of email domain hosts in our whitelist:

```
jaro addr | jaro stat emails
```

Will print out bars and domains in descending order, highlighting the most frequent email domain in our contacts, which turns out to be very often gmail.com, unfortunately for our own privacy.

To visualize the frequency of traffic across our filtered folders in the past month:

```
jaro search date:1w.. | jaro stat folders
```

Will show quantities of mails filed to folders during the past week, quickly highlighting the mailinglists that have seen more recent activity.

To see who is most active in a mailinglist which is filtered to a folder:

```
jaro search folder:org.dyne.dng | jaro extract stdin from | jaro stat names
```

Will give an overview on who is the most prolific writer in the *org.dyne.dng* mailinglist, filed into the folder by a rule in **Filters.txt** like:

```
to    dng@lists.dyne       save         org.dyne.dng
```

Please note the **extract** command is there to extract email addresses and names found in the *From:* field of all search hits, the command is explained better in the next chapter: *Addressbook*.

8.1 Statistics in brief

All **stats** commands takes lists of addresses or email messages from stdin.

command	effect
stats email	reads addresses from stdin, prints out stats on frequency of emails found
stats names	reads addresses from stdin, prints out stats on frequency of names found
stats folders	reads paths to messages from stdin, prints out stats on frequency of folders

So in case of **stats email** or **stats names** any result of search must be first filtered by **extract** in order to provide addresses to stats, else errors will occur. To limit the stats to the *From:* field use the **extract stdin from** also shown in examples, any other refinement can be done also in the domain of the search commands.

9 Addressbook

Addressbooks are the files storing the whitelist, the blacklist and optionally other custom lists of addresses. The format we use is native **abook** database files, by convention in *$JAROMAILDIR/whitelist.abook* and *$JAROMAILDIR/blacklist.abook*. More custom addressbooks can be used by specifying them using **-l** on the commandline, for instance **-l family** will query the *$JAROMAILDIR/family.abook* addressbook; when not used, **whitelist** is the default.

Addressbooks can be edited using a interactive console interface, for instance to add or delete entries by hand: use the **abook** command and optionally the **-l** option.

```
jaro abook
```

This will open the current whitelist for edit. To edit the blacklist add **-l blacklist** instead.

To quickly dump to the console all names and addresses in the Jaro Mail addressbook, one can use the **list** command

```
jaro list
```

To match a string across the addressbook, simply use the composite command **addr** followed by strings, for instance:

```
jaro addr dyne
```

will list all addresses containing 'dyne' in your whitelist.

9.1 Address lists

Jaro Mail handles lists of addresses as plain text files or streams with entries formatted as '*Name <email>*' and newline terminated. This simple format conforms (or is normalized to) the RFC822 standard and UTF-8 charset encoding, both produced on *stdout* and read from *stdin* by various useful commands to take advantage of console piping.

Such lists of addresses are the output of the **extract** command, which is able to read the output of other commands and extract a list of email addresses found.

```
jaro search open source date:2w..  | jaro extract stdin
```

Will print to stdout the list of addresses found among the results of a search for *open source* through all the emails archived in the past 2 weeks.

```
jaro search date:1y.. and folder:known | jaro extract
```

Will print a sorted list of unique addresses found in the emails matching the search expression '*date:1y.. and folder:known*', meaning all messages stored in the '*known*' folder and not older than 1 year from now.

The **import** command is complementary to extraction: it reads an address list from stdin and imports it inside an addressbook specified using '-l' or a *group* list file provided as argument.

```
jaro search folder:unsorted | jaro extract | jaro import -l blacklist
```

Will extract all addresses found in unsorted (the maildir collecting all non-mailinglist emails in which we are not an explicit recipient) and put them into our blacklist.

9.2 Export to VCard and other formats

VCard is an exchange format useful to interface with other addressbook software and mobile phones, as well with spyware as Google and Apple mail. Jaro Mail supports converting address lists to a variety of formats thanks to *abook*:

```
jaro addr | jaro export vcard
```

Will take the list of addresses in whitelist and convert it to the **vcard** format on stdout, ready to be redirected to a file.

Here below a list of output formats supported as argument to export:

Format	Description
abook	abook native format
ldif	ldif / Netscape addressbook (.4ld)
vcard	vCard 2 file
mutt	mutt alias
muttq	mutt query format (internal use)
html	html document
pine	pine addressbook
csv	comma separated values
allcsv	comma separated values (all fields)
palmcsv	Palm comma separated values
elm	elm alias
text	plain text
wl	Wanderlust address book
spruce	Spruce address book
bsdcal	BSD calendar
custom	Custom format

Of course **export** works with any list of addresses from stdin, for instance the result of **extract** operations on search queries, so that multiple commands can be concatenated.

9.3 Addressbook in brief

Here a roundup on the addressbook commands that are available from the *jaro* commandline script. Arguments '-l abook' take the string to identify

Command	Arguments	Function (print on stdout, import from stdin)
abook	-l listname	edit the addressbook (default whitelist)
addr	search expr	print list of addresses matching expression
extract	maildir	print address list of all mails in maildir
extract	gpg keyring	print address list of gpg public keyring
extract	gpg pubkey	print address list of gpg key signatures
extract	vcard file	print address list of entries in VCard file
import	-l listname	import address list from stdin to addressbook
export	format	convert address list to a format (default vcard)

10 Storage and backup

Most existing e-mail systems have their own storage format which is often over-complicated and forces us to convert our archives to it.

Jaro Mail stores emails using the well documented format **Maildir** which is common to many other free and open source e-mail software and was developed and well documented by D.J. Bernstein.

We can safely say that the Maildir format to store e-mails will stay the same and well compatible in 10 years from now, if not more, mostly because of its simplicity: that's what we need the most from a storage format after all.

Quoting him about the wonders of this format:

> Why should I use maildir?
>
> Two words: no locks. An MUA can read and delete messages while new mail is being delivered: each message is stored in a separate file with a unique name, so it isn't affected by operations on other messages. An MUA doesn't have to worry about partially delivered mail: each message is safely written to disk in the tmp subdirectory before it is moved to new. The maildir format is reliable even over NFS.[3]

10.1 Merge maildirs

Jaro Mail can safely merge two different maildirs basically gathering all e-mails stored in them into a unique place. This is done using two arguments, both maildir folders: the first is the source and the second is the destination e-mails from both will be gathered:

```
jaro merge ml.saved-mails ml.global-archive
```

The above command will move all emails stored inside the maildir folder "ml.saved-mails" to the other maildir folder "ml.global-archive". Upon success the first argument "ml.saved-mails" will be deleted and all its contents will be found in "ml.global-archive".

10.2 Backup mails

To facilitate the separation of stored email files across maildirs, for instance to move from a maildir to another all those mails that are older than a certain period, Jaro Mail implements the **copy** and **move** commands, reading a list of paths from stdin (as result of a search, for instance) and moving them to a destination maildir while preserving their reading state (new or cur).

For instance to move all archived mails older than 3 years into a separate folder:

```
jaro search date:3y.. | jaro move /media/backup/old.mails
```

This will move all the emails found by the search expression *date:3y..* (all mails older than 3 years) into '*media/backup/old.mails*' which must be a maildir.

[3] `http://cr.yp.to/proto/maildir.html`
What this virtuous, sometimes very cryptical man is trying to say here is that the Maildir format in its simplicity of implementation represents an extremely reliable way to retreive and store emails without the risk of losing any if the Internet connection goes down.

While skipping over the internal details of this storage system, which basically consists in plain text files saved into subdirectories, we will have a look at some very interesting features that Jaro Mail can offer to its users and to the even larger audience of Maildir format users.

The same way one could use **jaro copy** to not delete originals or even **jaro link** to create symlinks to results into a new maildir, without increasing occupation and allowing to review results with the help of an external program supporting maildirs, for instance using directly

```
mutt -f /media/backup/old.mails
```

This functionality is studied explicitly to be flexibly adopted in various situations and scripts, so the backups should really be customized ad-hoc for the particular setup.

10.3 Filter a maildir

If filters are updated or one desires to import a maildir into Jaro Mail processing it through its filters, the **filter** command is provided to (re)filter a maildir. First edit **Filters.txt** with matches for the to: (which includes cc:) and from: header fields, then run:

```
jaro update
```

To tell Jaro Mail to update its internal filters according to the modifications, and then:

```
jaro filter my-old-maildir
```

Beware that filtering is a lengthy operation, especially on big maildirs: it will first pass all messages found through your filters, refiling them to folders (which may create duplicates if filenames are different).

It is possible to filter any maildir, also those coming from other programs of course. Best practice is to copy the maildir inside the $JAROMAILDIR directory (typically ~/Mail) and then refer to it by its name: all arguments to the filter command can be relative to that directory.

10.4 Storage in brief

Here a recap of the commands dealing with maildir storage in Jaro Mail. Please note the syntax is subject to change in future:

Command	Syntax
move	(reads stdin) destination-maildir
copy	(reads stdin) destination-maildir
link	(reads stdin) destination-maildir
merge	origin-maildir destination-maildir
filter	maildir

11 Security

11.1 Password storage

Our MUA (Mutt) and our MTA (Fetchmail) normally required the user to input the email account password every time or write it clear inside a plain text file, jeopardizing the secrecy of it.

But most desktops nowadays have a keyring that stores passwords that are often used during a session, saving the user from retyping them every time.

Jaro Mail provides an interesting (and long awaited) feature even for those who are already using Mutt for their email: **it stores passwords securely**. This is done in different ways depending from the operating system is being running on.

Jaro Mail will use the default keyring whenever present to store all new passwords for each account used: the first time will prompt for a password and, while using it, will save it in relation to the particular account. This way the user can simply authenticate into the keyring at login and, while managing such sensitive informations using OS specific tools, Jaro Mail can be launched without the tedious task of a password input every time e-mails are being checked.

On **Apple/OSX** the default internal keyring is being used.

On **GNU/Linux** gnome-keyring is supported if found, else JaroMail will revert to use its own encrypted[4] database called **keyring**. Every time a password will be retrieved or saved, the keyring password will be asked. However, it is recommended to use Gnome-Keyring over the native one, which has still some glitches.

11.2 A tip for GNU/Linux users

Those using a GNU/Linux system might want to have a look at our other software **Tomb, the Crypto Undertaker** [5] which takes care of quick mount and umount of an encrypted volume when desired and includes a **hook** mechanism to automatize the execution of commands to make a directory inside the encrypted volume immediately available in the user's home.

Using a light combination of scripts between Jaro Mail and Tomb is possible to achieve a strong level of personal security, definitely above the average.

In particular, Jaro Mail does not needs system-wide installation, but can be installed and used in a way that makes it totally self-contained and transportable across systems inside a Tomb. When installing, just specify a prefix that is writable by the user, then make sure the **JAROMAILDIR** environmental variable points to the path where downloaded maildirs must be stored and the **JAROWORKDIR** environmental variable points to the path where jaromail was installed:

```
cd JaroMail-3.0
make
PREFIX=/media/secrets.tomb/usr make install
export JAROWORKDIR=/media/secrets.tomb/usr/share/jaromail
export JAROMAILDIR=/media/secrets.tomb/Mail
```

For more information about Tomb please refer to its own documentation: environmental variables can also be set via hooks and file paths can be automatically overlayed into $HOME when the Tomb is open.

12 Advanced usage

12.1 Replay: avoid repeating long operations

Working on the commandline can have some disadvantages. One of them is that if one runs a long operation to see its result and forgets to save it also on a file (i.e. using tee) the operation needs to be re-run and saved.

Jaro Mail helps the user to **replay** the last output print by saving it everytime in its own cache. Replay can also save per-command outputs so that long pipe chains can be repeated selectively by naming the command.

[4] The keyring is encrypted using weak symmetric encryption via GnuPG, the only protection for the data inside then is the password memorized by the used.

To explicitly change a password one can operate the default keyring manager or use the command **jaro passwd** (and specify other acconts using **-a accountname**)) which will prompt to set for a new password even if an old one is known.

[5] `http://tomb.dyne.org`

Only some commands have the replay capability, to have a list of available replays on your system do, based on your last run commands:

```
jaro replay list
```

To replay the last search command and pipe it into headers to have a better view of it:

```
jaro replay search | jaro headers
```

For instance imagine giving the command that searches for all mails sent to *nettime-l* and extracts all addresses in the *From:* including duplicates, then sorts them and eliminates duplicates

```
jaro search to:nettime-l | jaro extract stdin from | sort | uniq
```

Depending from the size of your nettime archives, this operation may take some time and one may not want to repeat it in order to compute some stats on the extract result. So one can go on and send the old output to a new command:

```
jaro replay extract | jaro stat names
```

This will print out statistics about the most prolific write to the nettime list according to your archives.

12.2 Send anonymous emails

Some people live difficult situations sometimes and are in need to send anonymous emails: for instance those endangered by the information they have, still in need to communicate it without being traced. Just imagine being a whistleblower part of a corrupt military organization, or a victim of mafia blackmailing, or a self determined woman in patriarcal societies. Situations like those may vary, still anonymity of communication is an important condition for personal safety and integrity.

Anonymizing an email is not as simple as changing the From: field of an email, since its headers will carry the history of the envelope and server logs will be held by the various Internet hosts interacting with its delivery. Often those hosts are run by corporate organizations ready to sell the logged information to anyone with the money to afford it.

To help these situations the MixMaster network exists since more than two decades, regularly routing emails across a chain of anonymizing servers that encrypt the envelope and delete logs, making it very difficult to track the origin and identity of those writing them. Anyway, such an operation requires long time and sometimes even fails to deliver: better send multiple copies of an anonymous email, then consider waiting one or two days before it gets delivered.

Setting up MixMaster and using it is a fairly complex task, but here Jaro Mail comes to the rescue making it easy for its users: after composing your email just change the From: field to **anon@mixmaster**. Our application will recognize that as a request to send the email across the MixMaster anonymous network.

To change the From: field after composition, just when headers and attachments are shown in Mutt, press [**ESC**] and then **f**, then type the special sender address **anon@mixmaster** and press [**Enter**].

12.3 Zsh commandline completion

For Zsh users out there there is a completion recipe that can facilitate the use of Jaro Mail by adding tab completion on the console terminal: commands and accounts will be listed and completed automatically just like with other commands.

To activate the completion move the file **src/completion/_jaromail** into the path where zsh loads vendor completions, typically that is **/usr/share/zsh/vendor-completions**.

12.4 Quickly send a file via email on Apple/OSX

To right-click on a file and send it via email attach using Jaro Mail you must create a "Service" using the application "Automator". It is fairly simple:

1. Start Automator
2. Choose the Service template
3. In the dropdown boxes that appear choose "files or folders" and "Finder"
4. Look for "Run Applescript" in the Library tree
5. Drag "Run Applescript" in the workflow area and paste this script into it:

```
on run {input, parameters}
tell application "Terminal"
activate
tell window 1
do script "/Applications/JaroMail.app/Contents/Resources/jaro/bin/jaro " & POSIX path of input
end tell
end tell
end run
```

Now Save the new service (you can name it "Send file via Jaro Mail") and when you will right click on a file, in the submenu "Services" you will find the option you just scripted, which will open a Terminal asking you the email address, while the file will be already configured as attach.

13 Acknowledgements

Jaro Mail would have never been possible without the incredible amount of Love shared by the free and open source community, since it is relying on the development of software like Mutt, Fetchmail and even more code which is included and used by this program.

Heartfelt thanks go to all those contributing code and sharing it to make the world a better place by not letting down all users in the hands of corporate non-sense and proprietary technologies and protocols.

This manual is written and maintained by Jaromil who is also the one who wrote the Jaro Mail scripts. Still he is far from being the person that wrote most of the code running here, just the one who organized it in an hopefully intuitive way for users.

In the following chapters the best is done in order to credit most people that contributed to free and open source software that Jaro Mail makes use of.

13.1 License

The following copyright notice applies to this manual, the software included is licensed under the same or different GNU GPL or BSD licenses, or available in the public domain.

```
Permission is granted to make and distribute verbatim copies of this
manual page provided the above copyright notice and this permission
notice are preserved on all copies.
```

13.2 Jaro Mail credits

Jaro Mail is written and maintained by Denis Roio (aka Jaromil) it started from the intention to share his own 10 years old e-mail setup, encouraged by the geek tradition of exchanging configuration files between friends.

Special thanks to Parazyd for useful code contributions and to Alvise Gottieri, Anatole Shaw, Francesco Politi and Fabio Pietrosanti for early testing and debugging.

The email envelop NyanCat graphics is kindly contributed by the Société ECOGEX.

13.3 Mutt credits

Please note that this is by no means an exhaustive list of all the persons who have been contributing to Mutt. Please see the manual for a (probably still non complete) list of the persons who have been helpful with the development of Mutt. Our special thanks go to Antonio Radici, the Mutt maintainer in Debian, for his suggestions and encouragement.

```
Copyright (C) 1996-2007 Michael R. Elkins <me@cs.hmc.edu>
Copyright (C) 1996-2002 Brandon Long <blong@fiction.net>
Copyright (C) 1997-2008 Thomas Roessler <roessler@does-not-exist.org>
Copyright (C) 1998-2005 Werner Koch <wk@isil.d.shuttle.de>
Copyright (C) 1999-2009 Brendan Cully <brendan@kublai.com>
Copyright (C) 1999-2002 Tommi Komulainen <Tommi.Komulainen@iki.fi>
Copyright (C) 2000-2004 Edmund Grimley Evans <edmundo@rano.org>
Copyright (C) 2006-2008 Rocco Rutte <pdmef@gmx.net>
```

13.4 Notmuch credits

Jaro Mail includes a search engine for e-mails that is also licensed GNU GPL v3+. Here below the names of the copyright holders and all those who have written it:

```
Carl Worth <cworth@cworth.org> is the primary author of Notmuch.
But there's really not much that he's done. There's been a lot of
standing on shoulders here:

William Morgan deserves credit for providing the primary inspiration
for Notmuch with his program Sup (http://sup.rubyforge.org/).

Some people have contributed code that has made it into Notmuch
without their specific knowledge (but with their full permission
thanks to the GNU General Public License). This includes:

Brian Gladman (with Mikhail Gusarov <dottedmag@dottedmag.net>)
Implementation of SHA-1 (nice and small) (libsha1.c)

Please see the various files in the Notmuch distribution for
```

```
individual copyright statements.
```

13.5 Fetchmail credits

Fetchmail is licensed GNU GPL v2

```
Copyright (C) 2002, 2003 Eric S. Raymond
Copyright (C) 2004 Matthias Andree, Eric S. Raymond, Robert M. Funk, Graham Wilson
Copyright (C) 2005 - 2006, 2010 Sunil Shetye
Copyright (C) 2005 - 2010 Matthias Andree
```

13.6 MSmtp credits

MSmtp is developed and maintained by Martin Lambers.

You can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

13.7 Statistics modules

We are including some (experimental, still) modules for statistical visualization using JQuery libraries. The first module inspiring us to implement such a functionality is Timecloud, then other modules followed.

```
Timecloud is Copyright (C) 2008-2009 by Stefan Marsiske
Dual licensed under the MIT and GPLv3 licenses.

TagCloud version 1.1.2
(c) 2006 Lyo Kato <lyo.kato@gmail.com>
TagCloud is freely distributable under the terms of an MIT-style license.

ExCanvas is Copyright 2006 Google Inc.
Licensed under the Apache License, Version 2.0 (the "License");

jQuery project is distributed by the JQuery Foundation under the
terms of either the GNU General Public License (GPL) Version 2.

The Sizzle selector engine (which is included inside the jQuery
library) is held by the Dojo Foundation and is licensed under the
MIT, GPL, and BSD licenses.

JQuery.sparkline 2.0 is licensed under the New BSD License

Visualize.JQuery is written by Scott Jehl
Copyright (c) 2009 Filament Group
licensed under MIT (filamentgroup.com/examples/mit-license.txt)
```

14 Appendix

14.1 Configuration examples

14.1.1 Accounts/default.txt

```
# Name and values are separated by spaces or tabs
# comments start the line with a hash

# Give a name to this account
name To Be Configured
# configure Identity.txt to set your From: field

# Email address (default is same as login)
email unknown@dyne.org

# Username
login USERNAME@dyne.org

## Change the settings only if you need

# Imap host address
imap mail.dyne.org

# Imap port: usually 443, 220 or 993
imap_port 993


# Smtp host address
smtp mail.dyne.org

# Smtp port: usually 25 or 465
smtp_port 25

# Authentication type
auth plain # or kerberos, etc

# Server certificate: check or ignore
cert ignore

# Transport protocol: ssl, tls or plain
transport tls


# Options when fetching
# to empty your mailbox you can use: 'fetchall' 'flush'
# by default this is 'keep': don't delete mails from server
options keep

# Remote IMAP folders to be retreived
# fill to provide a list of folders to be fetched
```

```
# default is to detect and fetch all remote folders
## folders INBOX priv unsorted filters

# list of folders to exclude from fetch
# comment or change to avoid leaving them on server
# please note we filters social networks by default
# (see Filters.txt and change it as you like)
exclude zz.spam zz.bounces zz.blacklist zz.social


#
# The password field will be filled in automatically
#
```

14.1.2 Filters.txt

```
# Default filter configuration for Jaro Mail

# Mailinglist filters are in order of importance
# syntax: to <list email> save <folder>
# below some commented out examples, note the use of a prefix,
# which makes it handy when browsing with file completion.

# to   crypto@lists.dyne save dyne.crypto
# to   dynebolic     save dyne.dynebolic
# to   freej     save dyne.freej
# to   frei0r-devel save dyne.frei0r
# to   taccuino     save ml.freaknet
# to   deadpoets     save ml.freaknet
# to   linux-libre save gnu.linux-libre
# to   foundations@lists save gnu.foundations
# to   debian-mentors save debian.mentors
# to   debian-blends save debian.blends

# Other filters for web 2.0 using folder names with a prefix:
# they can facilitate folder maintainance.
# These are on by default, comment out if not desired.

from      github.com            save zz.social
from      launchpad             save zz.social
from      identi.ca             save zz.social
from      twitter.com     save zz.social
from      linkedin.com     save zz.social
from      googlealerts     save zz.social
from      plus.google.com     save zz.social
from      youtube.com     save zz.social
from      wmt-noreply@google save zz.social
from      facebook         save zz.social
from      FriendFeed     save zz.social
from      academia-mail.com     save zz.social
from      statusnet         save zz.social
```

```
from      basecamp          save zz.social
```