

# MAILARCHIVE

*(Version 1.16)*

Luca Fini<sup>1</sup>  
*lfini@arcetri.astro.it*

Revision: December 1999

<sup>1</sup>Osservatorio Astrofisico di Arcetri

**Arcetri Technical Report N° 10/1999**  
**Firenze, November 1999**

## Sommario

*MAILARCHIVE è uno script Perl che filtra messaggi di posta elettronica e li organizza in una struttura ad archivio provvista di un indice ipertestuale. L'indice è scritto in linguaggio HTML (HyperText Markup Language) in modo che l'archivio può essere gestito da un server HTTP (come ad esempio httpd di NCSA) consentendo l'accesso mediante un Browser WWW come Lynx, Mosaic o Netscape. La procedura può opzionalmente utilizzare un sistema di indici inversi che consente di rendere più efficienti le ricerche testuali nell'archivio. L'uscita prodotta dalla procedura di ricerca è anch'essa in HTML. MAILARCHIVE può essere usato per facilitare l'organizzazione di documenti in una struttura accessibile mediante browsers WWW.*

## Abstract

*MAILARCHIVE is a Perl script which processes mail messages and stores them in a sort of archive provided with an hypertext index. The index is formatted in HTML (HyperText Markup Language) so that the archive can be managed by an HTTP server (such as NCSA httpd) and accessed by a WWW browser such as Lynx, Mosaic, Netscape, etc. The script optionally supports the creation and maintenance of a reverse index so that very efficient textual searches can be performed. The output of the search procedure is also HTML. MAILARCHIVE may be used to ease the organization of any kind of pieces of information which must be accessed by WWW browsers.*

## COPYRIGHT NOTICE

MAILARCHIVE was developed by Luca Fini at the Osservatorio Astrofisico di Arcetri (Firenze, Italy). Permission is granted to copy this software, to redistribute it on a nonprofit basis, and to use it for any purpose, subject to the following restrictions and understandings.

1. Any copy made of this software must include this copyright notice in full.
2. All materials developed as a consequence of the use of this software shall duly acknowledge such use, in accordance with the usual standards of acknowledging credit in academic research.
3. The author have made no warranty or representation that the operation of this software will be error-free or suitable for any application, and he is under no obligation to provide any services, by way of maintenance, update, or otherwise. The software is an experimental prototype offered on an as-is basis.
4. Redistribution for profit requires the express, written permission of the author.

## RELEASE CONTENTS

MAILARCHIVE is distributed as a compressed `tar` archive file named:

`MailArch-n.nn.tgz`.

Where *n.nn* indicates a version number. The archive should contain the following files:

- `MailArchive` A mail filter and folder processor to create and maintain archives with hypertext index and searching capabilities.
- `MAsearch.pl` An CGI form processing script which allows textual searches on archives created by MAILARCHIVE.
- `.MAconfig` An example configuration file.
- `.MAtemplate` An example HTML script template used by MailArchive to create and update HTML documents from which archives can be accessed.

## ADDITIONAL SOFTWARE REQUIRED

MAILARCHIVE assumes the availability of a number of standard Unix utilities and some additional software to function properly. Here follows a list of everything is needed.

- You need *Perl* 5.x installed and running. All scripts assume that it is installed in the directory `/usr/local/bin`, and must be modified for different locations.
- You need a standard version of `sendmail` up and running on your machine (it is the UNIX standard utility that processes incoming and outgoing mail). Other mail delivery agents might be used, provided that a proper configuration is added to the configuration file (see Section 4.1), but no extensive tests have been done of this feature.
- The archive and the corresponding lists may be generated so that they can be browsed locally (i.e.: accessing files directly), but the text search feature can only be provided by an HTTP server executing the CGI script `MAsearch.pl`. So the target system must have a suitable HTTP server installed and running. The scripts have been tested against NCSA `httpd`, CERN `httpd` and `apache`, and should work with any other servers supporting FORMS.
- The server will be useless if you don't have a suitable client program (*Mosaic*, *Lynx*, *Netscape*, can do the job).
- If you want to be able to use the fast textual search system to search documents according to their content you also need *Glimpse*.

*Glimpse*<sup>1</sup> is an indexing and searching engine written by Udi Manber, Sun Wu, and Burra Gopal, Department of Computer Science, University of Arizona, Tucson, AZ 85721.

You may get *Glimpse* from: `ftp://cs.arizona.edu/glimpse`

The scripts have been tested both with version 1.1 and version 2.0.

- If you want to properly format and print the 14 pages user's manual you need L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> and the related tools. You may get everything at: `ftp://ftp.tug.org/tex/`.

---

<sup>1</sup>E-mail to `glimpse@cs.arizona.edu` for info.

# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>                          | <b>1</b>  |
| <b>2</b> | <b>Document Processing</b>                   | <b>1</b>  |
| 2.1      | Folder Processing . . . . .                  | 2         |
| 2.2      | Mail Message Processing . . . . .            | 2         |
| <b>3</b> | <b>Installing MAILARCHIVE</b>                | <b>3</b>  |
| <b>4</b> | <b>Configuring MAILARCHIVE</b>               | <b>4</b>  |
| 4.1      | The Configuration File (.MAconfig) . . . . . | 4         |
| 4.2      | The Template File (.MAtemplate) . . . . .    | 7         |
| 4.3      | How the File List is Built . . . . .         | 9         |
| 4.4      | Access Control . . . . .                     | 10        |
| 4.4.1    | Allow list . . . . .                         | 10        |
| 4.4.2    | Password check . . . . .                     | 10        |
| <b>5</b> | <b>How to Format Messages</b>                | <b>10</b> |
| 5.1      | Managing Expiration Dates . . . . .          | 12        |
| <b>6</b> | <b>Rebuilding Lists</b>                      | <b>12</b> |
| <b>7</b> | <b>The Browsing Interface</b>                | <b>13</b> |
| <b>8</b> | <b>MAILARCHIVE Options</b>                   | <b>13</b> |
| 8.1      | Processing Options . . . . .                 | 13        |
| 8.2      | Documentation Options . . . . .              | 14        |

# 1 Introduction

The MAILARCHIVE *Perl* script was born from the need to provide a small group of people with a simple way to store a number of messages in an archive which allows a fast and easy retrieval of messages.

The LAN at the Osservatorio di Arcetri is a pretty complex structure where some services are centralized and managed by a small group of professionals while some research groups have their own “system manager” (usually a scientist who spends some time doing management) to cope with special needs (dedicated software packages and the like).

The scenario is thus a number of people who must informally cooperate at least by informing each other of changes of software configurations, installation of new packages, bug fixes, and so on. At the same time it is useful to have an historical memory of what has happened to the system.

MAILARCHIVE was designed as a natural extension of a mailing list we previously used to do the job. When a piece of information must be stored in the archive it is sent as a mail message to a fake username which starts MAILARCHIVE by using a standard feature of the `sendmail` program. MAILARCHIVE stores the message in a file and adds a line in a list of messages, written in the HTML language, which provides hypertext links to the message. At the same time a reverse index which allows fast textual search operations is updated.

Document retrieval is performed by accessing the hypertext lists, or by performing a search via a suitable form processing script which returns an HTML formatted list of relevant documents.

In order to increase the flexibility of the systems some options have been added: the script may process either single mail messages or entire mail folders, as generated by the mail utility program (this is how an archive may be initially built from a set of existing mail folders); it is possible to include an expiration date in a document, so that it may be deleted when expired; it is possible to rebuild the document index from the stored document files (this allows to manually delete documents and update the index accordingly).

MAILARCHIVE’s flexibility allows to use it for various purposes, from maintaining a long term historical archive which continuously grows with time as explained above, to the management of a posting system for short term messages.

## 2 Document Processing

MAILARCHIVE can process documents stored into a mail folder, as created by the UNIX mail utility program or can be used to filter mail messages actually sent to a fake mail address. In both cases messages are splitted into separate files and stored into subdirectories of the archive main directory. Subdirectories are named according to the year contained in the message date, so that all messages in the archive are grouped by year. The files containing the HTML lists are created (or updated) in the same archive main directory.

The document list may either follow the directory structure (i.e.: it is subdivided by year) or may be flat (all documents are grouped together in a single list).

Figure 1 shows a picture of the archive file structure.

The uppercase E in the file name (see under directory 1995) tells that the file has an expiration date (see Section 5.1 below) and may be subject to deletion.

|                 |                         |
|-----------------|-------------------------|
| TheArchive/     | Main archive directory  |
| index.html      | Main Index              |
| 1993_index.html | List of 1993 documents  |
| 1994_index.html | List of 1994 documents  |
| 1995_index.html | List of 1995 documents  |
| 1993/           | 1993 Document directory |
| Napr16-00.txt   | Document 1              |
| Nmar18-00.txt   | Document 2              |
| Nmar18-01.txt   | Document 3              |
| ....            | ....                    |
| 1994/           | 1994 Document directory |
| ....            | ....                    |
| 1995/           | 1995 Document directory |
| Eapr20-00.txt   | Document 1              |
| Eapr20-01.txt   | Document 2              |
| ....            | ....                    |
| ....            | ....                    |

Figure 1: Typical MAILARCHIVE directory structure

The main list file (`index.html`<sup>2</sup>) may either contain a list of years which link to yearly lists (files `1993_index.html`, `1994_index.html`, etc.), or directly contain the file list if you select the “flat” list option (see Section 4.1). The documents in HTML lists may be ordered as needed (see Section 4.1).

Documents in the archive can be optionally structured in “category” subdirectories; i.e.: subdirectories are referred to a subdivision according to message content instead of by year. This can be obtained by specifying the proper configuration parameters (see Section 4.1).

## 2.1 Folder Processing

To initiate an archive from a mail folder, or to add an entire mail folder to an existing archive, you must run the script interactively:

```
MailArchive -f FolderName TheArchive
```

where: `FolderName` is the file containing the mail folder and `TheArchive` is the main archive directory.

E.g:

```
MailArchive -f MyFolder /usr/local/ArcAccount
```

## 2.2 Mail Message Processing

The script may be used as a mail filter to process messages sent to a special e-mail address adding them to an archive. To add a new document to the archive a user can send a mail message to that address which will be processed by MAILARCHIVE in the same way as described in the previous paragraph.

---

<sup>2</sup>You may notice that this is the file that the HTTP server automatically accesses when the URL points to the archive main directory (see the HTTP server documentation for details).

If the Subject of the message begins exactly with the word: !DEBUG, then the message is not stored but useful debug information is sent back to the caller (a similar effect may be obtained with the option -d when using the program interactively).

The `sendmail` Unix utility provides the mechanism to use MAILARCHIVE as a mail filter (see Section 3 for details).

### 3 Installing MAILARCHIVE

In order to install MAILARCHIVE as a mail filter and set up an archive you must follow the procedure explained below:

- The MAILARCHIVE script must be copied in a proper directory, maybe where you usually put all your favorite scripts (such as `/usr/local/bin`).
- For each archive managed by MAILARCHIVE, a fake user account provided with a home directory must be created. It isn't necessary that the account can be logged into: a no password account is quite suitable. The account home directory may be (and usually will be) the same as the archive main directory. This directory must be accessible to the HTTP server which will be used for browsing and searching documents.
- On the home directory you put a file named `.forward` containing a single line as in the following example:

```
"|/usr/local/bin/MailArchive -l /usr/local/ArcAccount"
```

The double quotes (") are required. In the example the fake username is *ArcAccount* and its home directory is `/usr/local/ArcAccount`.

The above command line requires that when a mail message for the user *ArcAccount* is received it is processed by sending it to the standard input of the script MAILARCHIVE.

- Since the script will be run by `sendmail` under the UID of the user to whom the message is sent, it is necessary that the main archive directory is owned by that user, with read and write access.
- The file `.MAconfig` on the archive main directory must contain the proper set of configuration parameters. The distribution kit provides an example of such a file and more details can be found in Section 4.1.
- The file `.MAtemplate` must contain a template for the creation of the main list. The distribution kit provides an example of the template file and more details can be found in Section 7. It will allow the script to create and update the HTML document through which the archive can be browsed and searched.
- To allow search operations on the archive the FORM processing script `MAsearch.pl` must be executable by the HTTP server. E.g.: it could be installed in the standard CGI-script directory of the server<sup>3</sup>.

---

<sup>3</sup>You may find directions in the HTTP server documentation. E.g.: see the online help of the NCSA `httpd` at:

<http://hoohoo.ncsa.uiuc.edu/docs/Overview.html>

## 4 Configuring MAILARCHIVE

### 4.1 The Configuration File (.MAconfig)

Many aspects of the MAILARCHIVE processing modes can be configured by commands contained in the file `.MAconfig` on the archive main directory (see the example file included in the distribution kit).

Configuration commands start at column one of the line with a keyword sometimes followed by values; keywords which require a value must be terminated with a colon `:`. Keywords and values can be separated by any number of blanks. Keywords are not case sensitive. Lines not starting with a valid keyword are simply ignored (so no error checking is performed<sup>4</sup>).

Here follows a list of keywords with explanations.

- **Allow.** Begins the list of accepted mail addresses for access control (see Section 4.4 for details). The lines between **Allow** and **End Allow** (see below) are read in as such, so no comment can be present anywhere.
- **Categories.** Begins the list of accepted categories. It is followed by a list of categories into which messages are subdivided. Each category is defined in a single line as in the following example:

```
astrometry      Astrometry
starform        Star Formation
planets         Planetary astronomy
misc           Miscellaneous
```

The first word in the line is the name of the subdirectory where files are stored. The rest of the line is the full title of the category.

The category list must be ended with the keyword **End Category** (see below).

When a category list is specified the keyword **Flat List** (see below) is ignored and the archive structure is forced to **Nested**.

When a category list is specified each message must contain a **Category** keyword which specifies the particular category to which the message belongs (see section 5).

- **Default Life:** It is followed by the default life time (number of days) to be used for all documents added to the archive if a valid expiration date or life time specification (see Section 5.1 for details) is not included in the message.
- **End Allow.** Terminates the list of accepted mail addresses. The list begins with the keyword **Allow** (see above).
- **End Category.** Terminates the list of categories. The list begins with the keyword **Categories** (see above).
- **Glimpseindex:.** It is followed by the full path of the command `glimpseindex`. If not specified the indexing procedure is not executed and the search procedure will use `grep`.

---

<sup>4</sup>A list of configuration parameters values as derived from the configuration file can be obtained by invoking the script in debug mode (option `-d`, or Subject: `!DEBUG`).

- **Index Numbers.** If this keyword is specified numbers which appear in the text will be included into the reverse index. By default only words are included. This option has no effect unless the `Glimpseindex` option is also present.
- **Flat List.** Selects the “Flat” format of the HTML document list (see Section 2 for details on structure of the document list). This keyword is ignored if a `Categories` list is defined (see above).
- **List Head:.** It is followed by a valid *Perl* string which is evaluated and printed at the beginning of the list of items. The string may contain the following variables:
  - `$Archive`: The name of the current archive.
  - `$Subdir`: The name of the current subdirectory (either the year or the category subdirectory).
  - `$Category`: The extended name of the current category.

after evaluation the string is printed at the beginning of the document list (see section 4.3 for details).

The provided default is the following string:

```
<h1> Index for folder: $Archive/$Subdir </h1>\n<dl>\n
```

Which will result in the HTML code needed for printing a title and for initiating a “definition list”.

**Note:** the `\n` generates a newline character in the output.

- **List Line:.** It is followed by a valid *Perl* string which will be evaluated and printed once for each item in the document list. (see section 4.3 for details).

The line may contain the following variables:

- `$URL`: The URL of the document file
- `$Subject`: The subject field of the mail message.
- `$Year`: The Year.
- `$From`: The sender field of the mail message.
- `$Date`: The date of the message.
- `$Expir`: The expiration date (empty if not specified in the message).

`MailArchive` provides two default lines:

```
<dt> $Date - From: $From (Expir: $Expir) <dd><a href=$URL>$Subject</a><p>\n
```

if the **Require Expiration** keyword is present in the file (see below), and:

```
<dt> $Date - From: $From <dd><a href=$URL>$Subject</a><p>\n
```

if the **Require Expiration** keyword is not present in the file.

The lines above contain text and HTML formatting commands (<dt>, <dd>, <p>), which are printed as such, and some variables: \$Date, \$From, \$Expir, \$URL, \$Subject, which are substituted by their values prior of printing.

The above elements may be intermixed in any way provided that the resulting string is a single line (I.e.: the line **must** be terminated by \n character, and **no other newline character must be contained in the line**. Otherwise the sort command will not be able to sort the document list. This also means that the format of this line is strictly connected to the value of keyword **sort** (see below).

- **List Tail**: It is followed by a valid HTML string which will be printed as such at the end of the document list (see section 4.3 for details). A suitable default is provided (The default is the HTML code: </dl>).
- **Mail Command**: It is followed by the the command to be used to mail back the acknowledgment message or reply to !DEBUG requests. A typical example is the following:

```
mail -s \"MailArchive reply\" $Sender < $MailMessage
```

The line is evaluated by *Perl* prior of execution to expand the sender name (variable \$Sender) and the name of the temporary file holding the message (variable \$MailMessage). Note that double quotes (") must be escaped. There is no default, if it is not specified no acknowledge, error or debug message will be mailed back to the sender of the message.

- **Nested List**. Selects the “Nested” format of the HTML document list (see Section 2 for details on structure of the document list).
- **Password**: Followed by a string (no withespace allowed) which will be checked against the password provided within the message to be posted. The password in the message must be specified by means of the **Password** line in the message body (see Section 4.4).
- **Require Expiration**. If this keyword is included no message will be accepted unless a valid expiration date or life time specification (see Section 5.1 for details) is included in the message.
- **Require Subject**. If this keyword is included no message will be accepted unless the “Subject” field is included in the message header.
- **Sort**: It is followed by the sort command (Unix **sort**) suitable to sort the document list prior of its inclusion in the list document. You must take into account that the format of each line is specified in item **List line** (see section 4.3) for details. A default suitable for the default format of list file lines is provided:

```
sort +3nr -4 +2Mr -3 +1nr -2
```

This makes a list ordered in descending date order, provided that the default index line format (see: **List Line**) is used.

- **wwwPath**: It is followed by the absolute path to the archive directory as seen by a WWW client. It will usually be different from the directory path due to the redirection capabilities of the **http** server. If not specified it is assumed that the archive is seen under the **Document Root**<sup>5</sup>.

---

<sup>5</sup>You may find information in the **httpd** server documentation.

## 4.2 The Template File (.MAtemplate)

MAILARCHIVE builds the HTML document list on the basis of a template file which must be edited to suit the particular needs. The file named `.MAtemplate` must be on the archive main directory and will contain mostly HTML text with some special flags.

```
1 <html><head><title> Test Archive </title></head><body>
2 <h1> Test Archive</h1>
3 <h2> Complete list: </h2>
4 <!--Include Point>
5 <h2> Text search: </a> </h2>
6 <blockquote>
7 <form method="POST"
8 action="http://www.arcetri.astro.it/cgi-bin/MAsearch.pl">
9 <input type=hidden name=DirPath
10 value="/usr/local/HTDOCS/CC/admlog">
11 Type a string to search for:
12 <input type=text size=30 name="RegExp">
13 <input type="submit" value="Do search">
14 <p>
15 Case sensitive :
16 <input type="checkbox" name="Case" value=1>
```

Figure 2: HTML template script for archive accessing (*Part-1*)

Figures 2 and 3 show an example of template file. For those unfamiliar with the HTML language<sup>6</sup>

line numbers have been added in the figures to non blank lines as references for the following explanations (number are thus **not part** of the template file!).

- 1 Title of the document just for reference: it is not displayed. The line is required by the HTML language.
- 2 A string of text which appears in big characters: the actual readable title. You may edit or delete it at will.
- 3 Another title, but smaller (say a subtitle).
- 4 Here is the point where MAILARCHIVE puts the year list, or the document list. The line may appear anywhere in the file, but must be unchanged to be recognized.

---

<sup>6</sup>many sources of information about HTML are available. E.g.: see the on-line help at NCSA:

<http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html>

```

17 Allowed errors:
18 <input type=text name="SpellErrs" size=1>
19 (Number of spelling errors allowed)

20 <br>
21 <!--Uncomment the following line to allow debug mode selection>
22 <!--"Debug" mode:<input type="checkbox" name="Debug" value=1>

23 </form>

24 <p>
25 Returns the list of messages containing the given string.
26 The search is based on words (numbers are not searched for).
27 A regular expression can be specified.
28 Boolean expressions are expressed as follows:<p>
29 <ul>
30 <li> AND : word1;word2 <p>
31 <li> OR : word1,word2 <p>
32 </ul>
33 The two forms cannot be intermixed.
34 </blockquote>

35 </body></html>

```

Figure 3: HTML template script for archive accessing (*Part-2*)

- 5 Another subtitle.
- 6 The following text is somewhat indented.
- 7 Here starts the FORM item. It will appear as a field which can be filled by the user with buttons to select special actions and so on.
- 8 The URL<sup>7</sup> of the FORM script which processes the search request (see below). You must change this line accordingly to the actual location of the script `MAsearch.pl`.
- 9-10 A “hidden” parameter<sup>8</sup> which specifies the path of the archive main directory (it is an absolute path to avoid ambiguities). The “value” of the parameter must be modified according to the actual path of your archive.
- 11-12 These lines generate the field that the user will fill with the search string or expression. Line 11 is simply a comment, line 12 must remain unchanged (the “size” specification can be different, anyway).
- 13 The “start searching” button. When you “press” it the request is sent to the server. The value field is the text written on the button: you may change it to suit your needs.
- 14 This puts some space (starts a new paragraph).

---

<sup>7</sup>The *Universal Resource Locator* in the WWW terminology, is the description of how to get to a file across the Network.

<sup>8</sup>A “hidden” parameter is not displayed on the form screen but is sent, with its value, to the processing procedure.

- 15-16 These two lines add a check button with a comment text. If the button is checked the corresponding argument is sent to the script and requires a "case sensitive" search (default is not case sensitive). The lines are not required.
- 17-19 These lines generate another fill-in box where the user can specify the number of spelling errors allowed in string comparisons (this is a standard feature of `glimpse` and will have no effect if the archive doesn't implement the reverse index and is searched with `grep`). The lines are not required. The default is exact matching.
- 20 Starts a new line on the screen.
- 21-22 Comments. Line 22, as such, is an HTML comment. If you uncomment it (by removing the characters `!--` at the beginning) a check box is included in the FORM: if checked the FORM processing script will include some debug information in the return file.
- 23 This line terminates the FORM section of the HTML file.
- 24-34 These lines are comment text which is included for the convenience of the user.
- 35 End of the HTML file.

### 4.3 How the File List is Built

The building of the document list file depends on a number of customizable elements. Here follows a brief explanation of the process.

The list of archived files must be either "Flat" (i.e.: all documents are gathered together in a single list) or "Nested" (i.e.: a main list will contain pointers to sub lists where documents are gathered on a yearly basis or based on a list of categories). You can choose the format most suitable to your needs by the related configuration parameters (see Section 4.1).

If a nested list is selected it is built with the following rules:

1. The template file is scanned up to the special line: `<!--Include Point >` and lines are output to the list file.
2. A list of years or of categories is generated into the list file by using the "unnumbered list" format (`<ul> ... <li> ... </ul>` tags) provided by HTML.
3. The scanning of the template file resumes up to the end.
4. For each year a document sublist file is generated as follows:
  - (a) Some fixed HTML lines are put in the file to define a suitable title.
  - (b) The line defined as `List Head` in the configuration file is put in the file, followed by an HTML comment line to be used by the document searching script.
  - (c) The list of document follows. Each line is formatted as dictated by the line defined as `List Line` in the configuration file, all the lines are sorted using the command specified in the related configuration parameter.
  - (d) The line defined as `List Tail` in the configuration file follows.
  - (e) Suitable, and fixed, file ending commands are put in the file.

If a flat list is selected, the processing is exactly as explained above, except that the list of files is expanded within the main list, where the include flag is found following steps *b*, *c*, and *d* above.

## 4.4 Access Control

MAILARCHIVE provides a rough access control mechanism in order to avoid unauthorized archiving of documents. The mechanism is very simple and will not completely prevent malicious attacks (i.e: somebody filling up your archive with trash messages), but can filter out archiving attempts due to errors, automatic mail responders and the like.

The access control is based on two mechanisms: a list of allowed mail addresses, and a password check.

### 4.4.1 Allow list

The “Allow” section of the configuration file may be use to specify a list of mail addresses which are allowed to post message to the archive.

The Allow section begins with the keyword `Allow` and ends with the keyword `End Allow` (see details on the configuration file in Section 4.1). If it is not empty, then only messages coming from the mail addresses listed in the section will be stored in the archive. Each address line may actually be a valid *Perl* regular expression (without the enclosing `//`) as in the following examples:

```
^lfini@(.*)?\oat\.ts\.it$ user lfini at any node of oat.ts.it
as\.arizona\.edu\$$          everybody at as.arizona.edu
^[^\@:\.]+$                  all local users, (i.e.: addresses
                              not containing @.:)
```

Note that some characters (e.g.: `@.`) in mail addresses must be escaped with “`\`” because they have special meaning in *Perl* regular expressions. The match is case insensitive.

### 4.4.2 Password check

If you are concerned because the `Sender` field in a mail message can be faked very easily, you may add a password protection to the archive.

The `Password` keyword (see Section 4.1) in the configuration file may be used to define a password which is checked against the one provided within the message to be stored (see Section 5). The message is stored in the archive only if the passwords match.

Note that the password is stored in clear form into the configuration file so that it is accessible to everybody who has read access to that file.

## 5 How to Format Messages

Documents to be stored in the archive are mail messages which can be sent by any standard mail utility, but some guidelines on how to format messages may be useful.

- MAILARCHIVE uses the message date as date for the document and the subject of the message as title in lists. So the subject of the mail message must be meaningful<sup>9</sup>.
- The message body is not formatted in any way. Messages containing plain ASCII text will be displayed as typed because the default extension of the file created is `.txt`, which instructs the browser to consider it as preformatted text.

---

<sup>9</sup>There isn't any way to enforce meaningfulness of titles, but specification of the subject field may be enforced by means of the related configuration parameter (see Section 4.1).

If the message is HTML source code the keyword `!Html` may be specified (see below). In this case the extension of the file created will be `.html`, so that a browser will be able to correctly display the file content.

- The message may contain the redefinition of some fields of the mail message header, namely:

```
!Date:    followed by a valid date.
!From:    followed by any string.
!Subject: followed by any string.
```

The date format must follow the rules for standard mail messages as specified in RFC 822<sup>10</sup>. Here are some examples of valid dates:

```
!Date: Tue, 08 Aug 1995 13:50:42 +0200
!Date: 7 jul 94
```

- The message may also contain a line which specifies an expiration date for that document<sup>11</sup> (see Section 5.1 below).
- The message may contain a flag to force the interpreting of message content as HTML source code:

```
!Html
```

This will force the name of the created file to have an `.html` extension, so that HTML source messages will be correctly displayed by any browser.

- If the corresponding flag is contained in the configuration file (see Section 4.1, keyword: `Password`), the message **must** contain a line specifying a password as in the following example:

```
!Password: postingPassword
```

Passwords are specified by a line beginning with the string `!Password` followed by a colon (:), followed by a string of characters (spaces following the colon are ignored, no spaces are allowed within the password). The string is checked against the password specified in the configuration file prior of allowing the message to be posted.

- If the archive is structured in categories the message **must** contain a line specifying the category to which the message belongs, as in the following example:

```
!Category: planets
```

The string following the keyword `!Category:` is checked against the list of accepted categories as defined in the configuration file (see Section 4.1), and stored in the corresponding subdirectory. Matching is based on substring and is case insensitive.

If either no category is provided, or the string doesn't match any of the defined categories, the message is rejected.

---

<sup>10</sup>Practically speaking if you want to archive a message originated by somebody else and you want it to have any of the fields From, Date and Subject from the original message you simply leave the original From: and Date: lines of the message prepending a ! character to them.

<sup>11</sup>The specification of an expiration date can be enforced by means of the related configuration parameter (see Section 4.1).

## 5.1 Managing Expiration Dates

The message body of a message processed by MAILARCHIVE may contain the indication of an expiration date. In this case the expiration date is indicated in the header of the archived document and the file containing the document is named in a way which indicates that it is subject to deletion.

The expiration date may be specified in two different ways:

1. As an absolute date, with a line beginning with the word **!Expir** followed by any characters, followed by a colon (:), followed by a date in *day month year* format, as in the following examples:

```
!Expiring: 22 Jun 1996
!ExpIRAtion date :27 Agosto 1997
!Expir: February, 12 / 1998
```

**Note:** Date format may be quite free. Month must be a string, but the procedure is able to understand months in Italian or English, and date parsing is independent on the order of the elements and on the kind of separators.

If the year is not specified, the date is the next day/month date after today.

2. As a life duration, with a line beginning with the word **!Life** followed by any characters, followed by a colon (:), followed by a number of days, as in the following examples:

```
!Life: 30
!LIFE of document in days:365
```

The expiration date will be computed by adding the specified number of days to the document date.

MAILARCHIVE itself may be used to remove expired documents from the archive (updating lists accordingly) as in the following example:

```
MailArchive -x /usr/local/ArcAccount
```

Here the archive whose main directory is **ArcAccount** is searched to identify expired documents, which are removed both from the archive and from the lists. Document files are not actually deleted but instead they are moved onto directory:

```
/usr/local/ArcAccount/Expired
```

If the keyword: **Require Expiration** is included in the configuration file a message is not accepted unless it contains a valid expiration specification (see Section 4.1).

## 6 Rebuilding Lists

MAILARCHIVE may also be used to rebuild the list files based on the message files stored under the specified directory. This may be useful to rebuild the lists from scratch, E.g. if any document has been deleted manually or if you want to change the format of the list. It is also recommended to rebuild indexes when you've edited some file in the archive in order to update the reverse indexes, if they're included in your configuration.

To rebuild the lists you may use a command as in the following example:

The command will be usually run with **root** privileges<sup>12</sup>.

## 7 The Browsing Interface

An archive created and maintained by MAILARCHIVE can be browsed and searched by means of a WWW interface such as *Mosaic*, *Netscape*, or *Lynx*.

To do that you need an HTTP server installed and running which can access the archive main directory. I've used NCSA `httpd`, but any other HTTP server supporting FORMS should work.

The browser will access the archive via a proper HTML page with the required links and, possibly, with a FORM section which allows to specify search expressions.

The HTML page is created and maintained by MAILARCHIVE as described in Section 4.3.

To support searching of documents based on the content, it is necessary that the FORM processing procedure `MAsearch.pl` has been installed and is available to the HTTP server.

`MAsearch.pl` is a standard FORM processing procedure, written in *Perl*, which processes text searching requests in an archive generated by MAILARCHIVE and returns an HTML formatted document with a list of pointers to selected messages. `MAsearch.pl` uses the information contained in the configuration file `.MAconfig` described in Section 4.1 to format the returned document so that it has exactly the same format as the one used for the search, with the difference that it only contains references to selected files instead of the full list.

`MAsearch.pl` will be usually put in the standard directory for HTTP FORM processing scripts and will be executed in the same way as any other FORM processing script (see directions in the HTTP server documentation).

Some documentation about the script and how to debug it may be found at the beginning of the script itself.

## 8 MAILARCHIVE Options

Here follows a complete description of all the options recognized by MailArchive.

### 8.1 Processing Options

The following options specify the way MAILARCHIVE operates on documents. They are used when the script is executed interactively.

- d** Debug mode: print out debug information. This switch is used mainly in interactive mode. A similar effect may be obtained when the script is executed as a mail filter by sending a message whose subject begins with the word `!DEBUG`.
- l** Writes some log records onto file `/tmp/MailArchive.<username>` where `username` is the username of the process which started `MailArchive` (this switch can be used also when `MailArchive` operates as a mail filter). The log file is overwritten every time the script is executed. It is recommended to specify this option in the mail filter command to use it as simple "first aid" debug tool.
- f** The next argument is the the filename of a mail folder to process. The mail folder must have the correct format (it must be generated by the Unix `mail` utility).

---

<sup>12</sup>MAILARCHIVE will set the effective UID to that of the main archive directory owner before proceeding.

- r** Operate in 'Rebuild mode'. Recreates the document list from the document files in the specified archive. The standard input is ignored. This mode is useful to update the index when a text message is modified or deleted from the directory. The reverse index is also rebuilt, if the configuration file requires so (see Section 4.1).
- x** Remove expired documents from the archive. Document files are removed from the archive and moved to the subdirectory **Expired** under the archive main directory. The document list is updated accordingly and the reverse index is rebuilt, if the configuration file requires so (see Section 4.1).

## 8.2 Documentation Options

The following options will produce various forms of documentation.

- c** Print the Copyright notice and exit.
- h** Print a short help page and exit.
- m** Print the entire user's manual (i.e. the document you're reading right now) in  $\LaTeX$  format and exit. You may redirect the output into a file and process it with  $\LaTeX$ .
- n** Print a README Notice (i.e. the content of the README file included into the distribution kit).
- v** Prints the version number (without the ending newline).