mailbox — Manipulate Email Archives

Purpose: Work with email messages in various local file formats.

The `mailbox` module defines a common API for accessing email messages stored in local disk formats, including:

- Maildir
- mbox
- MH
- Babyl
- MMDF

There are base classes for `Mailbox` and `Message`, and each mailbox format includes a corresponding pair of subclasses to implement the details for that format.

mbox

The mbox format is the simplest to show in documentation, since it is entirely plain text. Each mailbox is stored as a single file, with all of the messages concatenated together. Each time a line starting with `"From "` ("From" followed by a single space) is encountered it is treated as the beginning of a new message. Any time those characters appear at the beginning of a line in the message body, they are escaped by prefixing the line with `">"`.

Creating an mbox Mailbox

Instantiate the `mbox` class by passing the filename to the constructor. If the file does not exist, it is created when `add()` is used to append messages.

```
# mailbox_mbox_create.py

import mailbox
import email.utils

from_addr = email.utils.formataddr(('Author',
                                    'author@example.com'))
to_addr = email.utils.formataddr(('Recipient',
                                  'recipient@example.com'))

payload = '''This is the body.
From (will not be escaped).
There are 3 lines.
'''

mbox = mailbox.mbox('example.mbox')
mbox.lock()
try:
    msg = mailbox.mboxMessage()
    msg.set_unixfrom('author Sat Feb  7 01:05:34 2009')
    msg['From'] = from_addr
    msg['To'] = to_addr
    msg['Subject'] = 'Sample message 1'
    msg.set_payload(payload)
    mbox.add(msg)
    mbox.flush()

    msg = mailbox.mboxMessage()
    msg.set_unixfrom('author')
    msg['From'] = from_addr
    msg['To'] = to_addr
    msg['Subject'] = 'Sample message 2'
    msg.set_payload('This is the second body.\n')
    mbox.add(msg)
    mbox.flush()
finally:
    mbox.unlock()
```

```
print(open('example.mbox', 'r').read())
```

The result of this script is a new mailbox file with two email messages.

```
$ python3 mailbox_mbox_create.py

From MAILER-DAEMON Sun Mar 18 20:20:59 2018
From: Author <author@example.com>
To: Recipient <recipient@example.com>
Subject: Sample message 1

This is the body.
>From (will not be escaped).
There are 3 lines.

From MAILER-DAEMON Sun Mar 18 20:20:59 2018
From: Author <author@example.com>
To: Recipient <recipient@example.com>
Subject: Sample message 2

This is the second body.
```

Reading an mbox Mailbox

To read an existing mailbox, open it and treat the `mbox` object like a dictionary. The keys are arbitrary values defined by the mailbox instance and are not necessary meaningful other than as internal identifiers for message objects.

```
# mailbox_mbox_read.py

import mailbox

mbox = mailbox.mbox('example.mbox')
for message in mbox:
    print(message['subject'])
```

The open mailbox supports the iterator protocol, but unlike true dictionary objects the default iterator for a mailbox works on the *values* instead of the *keys*.

```
$ python3 mailbox_mbox_read.py

Sample message 1
Sample message 2
```

Removing Messages from an mbox Mailbox

To remove an existing message from an mbox file, either use its key with `remove()` or use `del`.

```
# mailbox_mbox_remove.py

import mailbox

mbox = mailbox.mbox('example.mbox')
mbox.lock()
try:
    to_remove = []
    for key, msg in mbox.iteritems():
        if '2' in msg['subject']:
            print('Removing:', key)
            to_remove.append(key)
    for key in to_remove:
        mbox.remove(key)
finally:
    mbox.flush()
    mbox.close()

print(open('example.mbox', 'r').read())
```

The `lock()` and `unlock()` methods are used to prevent issues from simultaneous access to the file, and `flush()` forces the changes to be written to disk.

```
$ python3 mailbox_mbox_remove.py

Removing: 1
From MAILER-DAEMON Sun Mar 18 20:20:59 2018
From: Author <author@example.com>
To: Recipient <recipient@example.com>
Subject: Sample message 1

This is the body.
>From (will not be escaped).
There are 3 lines.
```

Maildir

The Maildir format was created to eliminate the problem of concurrent modification to an mbox file. Instead of using a single file, the mailbox is organized as directory where each message is contained in its own file. This also allows mailboxes to be nested, so the API for a Maildir mailbox is extended with methods to work with sub-folders.

Creating a Maildir Mailbox

The only real difference between creating a `Maildir` and `mbox` is that the argument to the constructor is a directory name instead of a file name. As before, if the mailbox does not exist, it is created when messages are added.

```
# mailbox_maildir_create.py
```

```
import mailbox
import email.utils
import os

from_addr = email.utils.formataddr(('Author',
                                    'author@example.com'))
to_addr = email.utils.formataddr(('Recipient',
                                  'recipient@example.com'))

payload = '''This is the body.
From (will not be escaped).
There are 3 lines.
'''

mbox = mailbox.Maildir('Example')
mbox.lock()
try:
    msg = mailbox.mboxMessage()
    msg.set_unixfrom('author Sat Feb  7 01:05:34 2009')
    msg['From'] = from_addr
    msg['To'] = to_addr
    msg['Subject'] = 'Sample message 1'
    msg.set_payload(payload)
    mbox.add(msg)
    mbox.flush()

    msg = mailbox.mboxMessage()
    msg.set_unixfrom('author Sat Feb  7 01:05:34 2009')
    msg['From'] = from_addr
    msg['To'] = to_addr
    msg['Subject'] = 'Sample message 2'
    msg.set_payload('This is the second body.\n')
    mbox.add(msg)
    mbox.flush()
finally:
    mbox.unlock()

for dirname, subdirs, files in os.walk('Example'):
    print(dirname)
    print('  Directories:', subdirs)
    for name in files:
        fullname = os.path.join(dirname, name)
```

```
        fullname = os.path.join(dirname, name)
        print('\n***', fullname)
        print(open(fullname).read())
        print('*' * 20)
```

When messages are added to the mailbox, they go to the new subdirectory.

Warning

Although it is safe to write to the same maildir from multiple processes, `add()` is not thread-safe. Use a semaphore or other locking device to prevent simultaneous modifications to the mailbox from multiple threads of the same process.

```
$ python3 mailbox_maildir_create.py

Example
  Directories: ['new', 'cur', 'tmp']
Example/new
  Directories: []

*** Example/new/1521404460.M306174P41689Q2.hubert.local
From: Author <author@example.com>
To: Recipient <recipient@example.com>
Subject: Sample message 2

This is the second body.

********************

*** Example/new/1521404460.M303200P41689Q1.hubert.local
From: Author <author@example.com>
To: Recipient <recipient@example.com>
Subject: Sample message 1

This is the body.
From (will not be escaped).
There are 3 lines.

********************
Example/cur
  Directories: []
Example/tmp
  Directories: []
```

After they are read, a client could move them to the `cur` subdirectory using the `set_subdir()` method of the `MaildirMessage`.

```
# mailbox_maildir_set_subdir.py

import mailbox
import os

print('Before:')
mbox = mailbox.Maildir('Example')
mbox.lock()
try:
    for message_id, message in mbox.iteritems():
        print('{:6} "{}"'.format(message.get_subdir(),
                                 message['subject']))
        message.set_subdir('cur')
        # Tell the mailbox to update the message.
        mbox[message_id] = message
finally:
    mbox.flush()
    mbox.close()

print('\nAfter:')
mbox = mailbox.Maildir('Example')
for message in mbox:
    print('{:6} "{}"'.format(message.get_subdir(),
```

```
                                message['subject']))

print()
for dirname, subdirs, files in os.walk('Example'):
    print(dirname)
    print('  Directories:', subdirs)
    for name in files:
        fullname = os.path.join(dirname, name)
        print(fullname)
```

Although a maildir includes a "tmp" directory, the only valid arguments for `set_subdir()` are "cur" and "new".

```
$ python3 mailbox_maildir_set_subdir.py

Before:
new    "Sample message 2"
new    "Sample message 1"

After:
cur    "Sample message 2"
cur    "Sample message 1"

Example
  Directories: ['new', 'cur', 'tmp']
Example/new
  Directories: []
Example/cur
  Directories: []
Example/cur/1521404460.M306174P41689Q2.hubert.local
Example/cur/1521404460.M303200P41689Q1.hubert.local
Example/tmp
  Directories: []
```

Reading a Maildir Mailbox

Reading from an existing Maildir mailbox works just like an mbox mailbox.

```
# mailbox_maildir_read.py

import mailbox

mbox = mailbox.Maildir('Example')
for message in mbox:
    print(message['subject'])
```

The messages are not guaranteed to be read in any particular order.

```
$ python3 mailbox_maildir_read.py

Sample message 2
Sample message 1
```

Removing Messages from a Maildir Mailbox

To remove an existing message from a Maildir mailbox, either pass its key to `remove()` or use `del`.

```
# mailbox_maildir_remove.py

import mailbox
import os

mbox = mailbox.Maildir('Example')
mbox.lock()
try:
    to_remove = []
    for key, msg in mbox.iteritems():
        if '2' in msg['subject']:
            print('Removing:', key)
            to_remove.append(key)
```

```
            to_remove.append(key)
    for key in to_remove:
        mbox.remove(key)
finally:
    mbox.flush()
    mbox.close()

for dirname, subdirs, files in os.walk('Example'):
    print(dirname)
    print('  Directories:', subdirs)
    for name in files:
        fullname = os.path.join(dirname, name)
        print('\n***', fullname)
        print(open(fullname).read())
        print('*' * 20)
```

There is no way to compute the key for a message, so use `items()` or `iteritems()` to retrieve the key and message object from the mailbox at the same time.

```
$ python3 mailbox_maildir_remove.py

Removing: 1521404460.M306174P41689Q2.hubert.local
Example
  Directories: ['new', 'cur', 'tmp']
Example/new
  Directories: []
Example/cur
  Directories: []

*** Example/cur/1521404460.M303200P41689Q1.hubert.local
From: Author <author@example.com>
To: Recipient <recipient@example.com>
Subject: Sample message 1

This is the body.
From (will not be escaped).
There are 3 lines.

********************
Example/tmp
  Directories: []
```

Maildir folders

Subdirectories or *folders* of a Maildir mailbox can be managed directly through the methods of the `Maildir` class. Callers can list, retrieve, create, and remove sub-folders for a given mailbox.

```
# mailbox_maildir_folders.py

import mailbox
import os


def show_maildir(name):
    os.system('find {} -print'.format(name))


mbox = mailbox.Maildir('Example')
print('Before:', mbox.list_folders())
show_maildir('Example')

print('\n{:#^30}\n'.format(''))

mbox.add_folder('subfolder')
print('subfolder created:', mbox.list_folders())
show_maildir('Example')

subfolder = mbox.get_folder('subfolder')
print('subfolder contents:', subfolder.list_folders())

print('\n{:#^30}\n'.format(''))
```

```
print('\n{:#^30}\n'.format(''))

subfolder.add_folder('second_level')
print('second_level created:', subfolder.list_folders())
show_maildir('Example')

print('\n{:#^30}\n'.format(''))

subfolder.remove_folder('second_level')
print('second_level removed:', subfolder.list_folders())
show_maildir('Example')
```

The directory name for the folder is constructed by prefixing the folder name with a period (`.`).

```
$ python3 mailbox_maildir_folders.py

Example
Example/new
Example/cur
Example/cur/1521404460.M303200P41689Q1.hubert.local
Example/tmp
Example
Example/.subfolder
Example/.subfolder/maildirfolder
Example/.subfolder/new
Example/.subfolder/cur
Example/.subfolder/tmp
Example/new
Example/cur
Example/cur/1521404460.M303200P41689Q1.hubert.local
Example/tmp
Example
Example/.subfolder
Example/.subfolder/.second_level
Example/.subfolder/.second_level/maildirfolder
Example/.subfolder/.second_level/new
Example/.subfolder/.second_level/cur
Example/.subfolder/.second_level/tmp
Example/.subfolder/maildirfolder
Example/.subfolder/new
Example/.subfolder/cur
Example/.subfolder/tmp
Example/new
Example/cur
Example/cur/1521404460.M303200P41689Q1.hubert.local
Example/tmp
Example
Example/.subfolder
Example/.subfolder/maildirfolder
Example/.subfolder/new
Example/.subfolder/cur
Example/.subfolder/tmp
Example/new
Example/cur
Example/cur/1521404460.M303200P41689Q1.hubert.local
Example/tmp
Before: []

##############################

subfolder created: ['subfolder']
subfolder contents: []

##############################

second_level created: ['second_level']

##############################

second_level removed: []
```

Message Flags

Messages in mailboxes have flags for tracking aspects such as whether or not the message has been read, flagged as important by the reader, or marked for deletion later. Flags are stored as a sequence of format-specific letter codes and the `Message` classes have methods to retrieve and change the values of the flags. This example shows the flags on the messages in the `Example` maildir before adding the flag to indicate that the message is considered important.

```
# mailbox_maildir_add_flag.py
```

```
import mailbox

print('Before:')
mbox = mailbox.Maildir('Example')
mbox.lock()
try:
    for message_id, message in mbox.iteritems():
        print('{:6} "{}"'.format(message.get_flags(),
                                 message['subject']))
        message.add_flag('F')
        # Tell the mailbox to update the message.
        mbox[message_id] = message
finally:
    mbox.flush()
    mbox.close()

print('\nAfter:')
mbox = mailbox.Maildir('Example')
for message in mbox:
    print('{:6} "{}"'.format(message.get_flags(),
                             message['subject']))
```

By default messages have no flags. Adding a flag changes the message in memory, but does not update the message on disk. To update the message on disk store the message object in the mailbox using its existing identifier.

```
$ python3 mailbox_maildir_add_flag.py

Before:
       "Sample message 1"

After:
F      "Sample message 1"
```

Adding flags with `add_flag()` preserves any existing flags. Using `set_flags()` writes over any existing set of flags, replacing it with the new values passed to the method.

```
# mailbox_maildir_set_flags.py
```

```
import mailbox

print('Before:')
mbox = mailbox.Maildir('Example')
mbox.lock()
try:
    for message_id, message in mbox.iteritems():
        print('{:6} "{}"'.format(message.get_flags(),
                                 message['subject']))
        message.set_flags('S')
        # Tell the mailbox to update the message.
        mbox[message_id] = message
finally:
    mbox.flush()
    mbox.close()

print('\nAfter:')
mbox = mailbox.Maildir('Example')
for message in mbox:
    print('{:6} "{}"'.format(message.get_flags(),
                             message['subject']))
```

The F flag added by the previous example is lost when `set_flags()` replaces the flags with S in this example.

```
$ python3 mailbox_maildir_set_flags.py

Before:
F      "Sample message 1"

After:
S      "Sample message 1"
```

Other Formats

`mailbox` supports a few other formats, but none are as popular as mbox or Maildir. MH is another multi-file mailbox format used by some mail handlers. Babyl and MMDF are single-file formats with different message separators than mbox. The single-file formats support the same API as mbox, and MH includes the folder-related methods found in the Maildir class.

See also

- Standard library documentation for mailbox
- Python 2 to 3 porting notes for mailbox
- mbox manpage from qmail – Documentation for the mbox format.
- Maildir manpage from qmail – Documentation for the Maildir format.
- `email` – The `email` module.
- `imaplib` – The `imaplib` module can work with saved email messages on an IMAP server.

smtpd — Sample Mail Servers

imaplib — IMAP4 Client Library

Quick Links

mbox
Creating an mbox Mailbox
Reading an mbox Mailbox
Removing Messages from an mbox Mailbox
Maildir
Creating a Maildir Mailbox
Reading a Maildir Mailbox
Removing Messages from a Maildir Mailbox
Maildir folders
Message Flags
Other Formats

This page was last updated 2018-03-18.

Navigation

smtpd — Sample Mail Servers
imaplib — IMAP4 Client Library

The output from all the example programs from PyMOTW-3 has been generated with Python 3.7.1, unless otherwise noted. Some of the features described here may not be available in earlier versions of Python.

Looking for examples for Python 2?