blanco

Release 0.6.0

August 16, 2014

Contents

`blanco` is a simple tool to help you, or more specifically *me*, keep in touch with people. All it does is notify you if you're failing to keep in contact. It is just a quick solution to a simple problem, as long as you use abook and your sent mail is easily accessible that is.

It is written in Python, and requires v2.6 or later. `blanco` is released under the GPL v3

Contents:

CHAPTER 1

Background

I'm one of those people who routinely forgets to keep in touch with friends, mostly because there are no useful cues to use as a reminder. `blanco` is my attempt to fill that space, and stop losing track of people. If you're the kind of person who has a tendency to suddenly realise "oh wow, I've not spoke to her for *over a year*" then `blanco` may be useful to you too.

The premise is simple: assign a "contact frequency" to a contact and let `blanco` keep track of when you're drifting apart. This is the sort of thing that – I suspect – we would all like to handle without the need for a specific tool, but life all too often just gets in the way of good intentions.

CHAPTER 2

Installation

You can download releases from the downloads page on GitHub.

`blanco` requires Python v2.6(or newer) and the following Python modules:

- arrow
- configobj

The following optional packages will be used if available:

- libnotify Python bindings for popup notifications
- blessings for coloured output in the terminal

If you're using Gentoo the hard dependencies are available from the main tree, and the optional dependencies are available from a combination of the main tree and my overlay.

2.1 From source

One you have downloaded a source tarball you can install it with the following steps:

```
$ python setup.py build
# python setup.py install  # to install in Python's site-packages
$ python setup.py install --user  # to install for a single user
```

CHAPTER 3

Setup

blanco requires access to your sent mail for calculating last contact times, and also access to a map of contact-to-frequency for calculating forgotten people.

If you can't fulfil the requirements in the following two sections then **blanco** is not for you!

3.1 Sent mail

`mbox`, `maildir` and `MH` mailbox formats are supported, thanks to the wonderful mailbox Python module.

msmtp logs are also supported, and using them is the preferred method. Parsing simple log entries is appreciably faster than processing mailboxes, and this method should be chosen if at all possible.

There is also a faster gmail specific option when you're using the msmtp log method, which takes advantage of the extra data included in Google's responses to calculate the date a mail was sent.

3.2 Addressbook

The default addressbook format is the format used by abook, where one of the custom fields allowed by abook is used to store frequency information.

`blanco` expects your abook entries to have a frequency value in the `frequency` field [1]. The format is "<n> <units>", where `n` is a number and `units` is a character from the set `[dwmy]`. For example, an entry with a frequency of `3 m` will be triggered if there hasn't been a mail sent to that address in three months.

You can add the following snippet to your `~/.abook/abookrc` file to display a frequency field in the `other` tab:

```
field frequency = Frequency

view OTHER = frequency
```

You do *not* need to set this for `blanco` to work, but it makes the purpose of the field clearer.

`blanco` can be used without abook, as it only requires a ini formatted contacts file. To create your own contacts file without abook follow the format below:

```
[0]
name=Bill
email=test@example.com
freq=30d
```

[1] You can select a different field using the *--field* option to `blanco`.


```
[1]
name=Joe
email=joe@example.com
freq=30d
```

If you use the layout above you should specify `--field=freq` when calling `blanco`.

Another alternative would be to use abook just to convert your current address book in to a suitable format. Check the output of `abook --formats` for the file formats supported by your version of abook.

CHAPTER 4

Configuration

blanco stores its configuration in `$XDG_CONFIG_HOME/blanco/config.ini` [1].

The configuration file is a simple `INI` format file, The file is processed with the configobj module, the documentation for which will describe some of the advanced features available within the configuration file.

You can specify command line options in the configuration file as defaults, and optionally override them from the command line. To toggle boolean options from the command line use their `--no-` prefixed versions.

An example configuration file is below:

```
sent type = msmtp
field = custom5
```

[1] The default value for `${XDG_CONFIG_HOME}` is system dependent, but likely to be `~/.config` if you haven't set it. For more information see XDG base directory specification.

CHAPTER 5

Usage

blanco.py supports writing to standard output or using popup notifications.

```
Ξ Desktop/blanco git:(master)  blanco.py
Mail due for Luke Legate
Ξ Desktop/blanco git:(master)
```

If the *--notify* option is specified popups will be displayed on the desktop, see the example below.

The *--notify* mode is specifically meant from a desktop startup sequence, and that is how `blanco`'s author normally uses it.

5.1 Options

--version
: Show program's version number and exit

-h, --help
: Show this help message and exit

-a <file>, **--addressbook**=<file>
: Address book to read contacts from

-t <mailbox|msmtp>, **--sent-type**=<mailbox|msmtp>
: Sent source type(mailbox or msmtp)

-r, --all
: Include all recipients(CC and BCC fields)

--no-all
Include only the first recipient(TO field)

-s <name>, **--field**=<name>
Addressbook field to use for frequency value

-n, **--notify**
Display reminders using notification popups

--no-notify
Display reminders on standard out

-v, **--verbose**
Produce verbose output

-q, **--quiet**
Output only matches and errors

5.1.1 Mailbox options

-m <mailbox>, **--mbox**=<mailbox>
Mailbox used to store sent mail

5.1.2 msmtp log options

-l <file>, **--log**=<file>
msmtp log to parse

-g, **--gmail**
Log from a gmail account(use accurate filter)

--no-gmail
msmtp log for non-gmail account

CHAPTER 6

blanco.py

6.1 *"Hey, remember me?"*

Author James Rowe <jnrowe@gmail.com>

Date 2010-05-02

Copyright GPL v3

Manual section 1

Manual group Email

6.1.1 SYNOPSIS

blanco.py [option]...

6.1.2 DESCRIPTION

A simple tool to help you, or more specifically *me*, keep in touch with people. All it does is notify you if you're failing to keep in contact. It is just a quick solution to a simple problem, as long as you use abook and your sent mail is easily accessible.

6.1.3 OPTIONS

--version Show program's version number and exit

-h, --help Show this help message and exit

-a <file>, --addressbook=<file> Address book to read contacts from

-t <mailbox|msmtp>, --sent-type=<mailbox|msmtp> Sent source type(mailbox or msmtp)

-r, --all Include all recipients(CC and BCC fields)

--no-all Include only the first recipient(TO field)

-s <name>, --field=<name> Addressbook field to use for frequency value

-n, --notify Display reminders using notification popups

--no-notify Display reminders on standard out

-v, --verbose Produce verbose output

-q, --quiet Output only matches and errors

Mailbox options

-m <mailbox>, --mbox=<mailbox> Mailbox used to store sent mail

msmtp log options

-l <file>, --log=<file> msmtp log to parse

-g, --gmail Log from a gmail account(use accurate filter)

--no-gmail msmtp log for non-gmail account

6.1.4 CONFIGURATION FILE

The configuration file, **${XDG_CONFIG_HOME:-~/.config}/blanco/config.ini**, is a simple **INI** format file for storing defaults for the command line options. For example:

```
sent type = msmtp
field = custom5
```

With the above configuration file the default sent mail source will be a msmtp logfile, and frequency information will be stored in abook's `custom5` field.

6.1.5 BUGS

None known.

6.1.6 AUTHOR

Written by James Rowe

6.1.7 RESOURCES

Home page: https://github.com/JNRowe/blanco

6.1.8 COPYING

CHAPTER 7

Frequently Asked Questions

Contents

CHAPTER 8

Alternatives

Before diving in and spitting out this package I looked for alternatives, but couldn't find any. If I have missed something please drop me a mail, and I'll a link to them here.

CHAPTER 9

Release HOWTO

9.1 Test

In the general case tests can be run via `nose2`:

```
$ nose2 -vv tests
```

When preparing a release it is important to check that `blanco` works with all supported Python versions, and that the documentation is correct.

9.2 Prepare release

With the tests passing, perform the following steps

- Update the version data in `blanco.py`
- Update `NEWS.rst`, if there are any user visible changes
- Commit the release notes and version changes
- Create a signed tag for the release
- Push the changes, including the new tag, to the GitHub repository

9.3 Update PyPI

Create and upload the new release tarballs to PyPI:

```
$ ./setup.py sdist --formats=bztar,gztar bdist_wheel register upload --sign
```

Fetch the uploaded tarballs, and check for errors.

You should also perform test installations from PyPI, to check the experience `blanco` users will have.

CHAPTER 10

API documentation

Note: The documentation in this section is aimed at people wishing to contribute to `blanco`, and can be skipped if you are simply using the tool from the command line.

10.1 Blanco

Note: The documentation in this section is aimed at people wishing to contribute to `blanco`, and can be skipped if you are simply using the tool from the command line.

`blanco.`**`parse_msmtp`**(*log*, *all_recipients=False*, *addresses=None*, *gmail=False*)

Parse sent messages mailbox for contact details.

Parameters

- **log** (*str*) – Location of the msmtp logfile
- **all_recipients** (*bool*) – Whether to include all recipients in results, or just the first
- **addresses** (*list*) – Addresses to look for in sent mail, all if not specified
- **gmail** (*bool*) – Log is for a gmail account

Return type `dict` of `str` keys and `arrow.Arrow` values

Returns Keys of email address, and values of seen date

`blanco.`**`parse_sent`**(*path*, *all_recipients=False*, *addresses=None*)

Parse sent messages mailbox for contact details.

Parameters

- **path** (*str*) – Location of the sent mailbox
- **all_recipients** (*bool*) – Whether to include CC and BCC addresses in results, or just the first
- **addresses** (*list*) – Addresses to look for in sent mail, all if not specified

Return type `dict` of `str` keys and `arrow.Arrow` values

Returns Keys of email address, and values of seen date

`blanco.`**`show_note`**(*notify*, *message*, *contact*, *urgency=1*, *expires=-1*)

Display reminder.

Parameters

- **notify** (*bool*) – Whether to use notification popup
- **message** (*str*) – Message string to show
- **contact** (*Contact*) – Contact to show message for
- **urgency** (*int*) – Urgency state for message
- **expires** (*int*) – Time to show notification popup in milliseconds

Raises OSError Failure to show notification

class `blanco.`**`Contact`**(*name*, *addresses*, *frequency*)
Initialise a new `Contact` object.

`notify_str`()
Calculate trigger date for contact.

Return type `str`

Returns Stylised name for use with notifications

`trigger`(*sent*)
Calculate trigger date for contact.

Parameters **sent** (`dict` of `str` keys and `arrow.Arrow` values) – Address to last seen dictionary

Return type `arrow.Arrow`

Returns Date to start reminders on

class `blanco.`**`Contacts`**(*contacts=None*)
Initialise a new `Contacts` object.

`addresses`()
Fetch all addresses of all `Contact` objects.

Return type `list` of `str`

Returns Addresses of every `Contact`

`parse`(*addressbook*, *field*)
Parse address book for usable entries.

Parameters

- **addressbook** (*str*) – Location of the address book to useful
- **field** (*str*) – Address book field to use for contact frequency

10.1.1 Examples

```
>>> contact = Contact('James Rowe', 'jnrowe@gmail.com', 200)
>>> contact.trigger({'jnrowe@gmail.com': datetime.date(1942, 1, 1)})
<Arrow [1942-07-20T00:00:00+00:00]>
```

10.2 Command line

Note: The documentation in this section is aimed at people wishing to contribute to `blanco`, and can be skipped if you are simply using the tool from the command line.

blanco.**process_command_line**()
Main command line interface.

Return type argparse.Namespace

Returns Parsed options and arguments

blanco.**main**()
Main script.

10.3 Utilities

Note: The documentation in this section is aimed at people wishing to contribute to blanco, and can be skipped if you are simply using the tool from the command line.

10.3.1 Time handling

blanco.**parse_duration**(*duration*)
Parse human readable duration.

Parameters **duration** (*str*) – Duration definition

Return type int

Returns Number of days in duration

Raises ValueError Invalid value for duration

10.3.2 Text formatting

blanco.**_colourise**(*text*, *colour*)
Colour text, if possible.

Parameters

- **text** (*str*) – Text to colourise
- **colour** (*str*) – Colour to display text in

Return type str

Returns Colourised text, if possible

blanco.**success**(*text*)

blanco.**fail**(*text*)

blanco.**warn**(*text*)

Examples

10.3.3 Time handling

```
>>> parse_duration('4w')
28
>>> parse_duration('2 m')
56
```

10.3.4 Text formatting

```
>>> fail('Error!')
'Error!'
>>> success('Excellent')
'Excellent'
>>> warn('Ick')
'Ick'
```

CHAPTER 11

Indices and tables

- *genindex*
- *modindex*
- *search*

b