UnGarage Documentation

Release 0.1

Laurent Peuch

August 11, 2012

CONTENTS

This is a guide on how I'm managing the irc bot UnGarage for la Quadrature du Net (and many others that I've deployed afterwards). My constrain are the modular, changing and original requests of a strange mustache owner. Therefor I was in need of a very flexible framework than can catch up with this strange phenomena.

The result of this work make an heavy use of the UNIX principle of a collection of small reusable tools. Because of this I can't release a source code so I'm releasing this documentation.

Using this method is:

- a very high flexibility, you'll be able to modify, stop, restart and change every part of it independently at any time
- it is language agnostic, program it in whatever you desire (but you'll very likely need at least a bit of shell script to glue everything together)
- make the problem of creating a multitasking IRC boot so simple and elegant
- very simple to use and maintained ... once you dig in it (I found this easy but this isn't the common way of approaching this kind of problem)
- might be a bit hard for other people to help you or understand your work for the same reason than described in the previous point
- is annoying if you need to relaunch everything (eg: server reboot) but I think I'm about to solve this problem (using supervisord) but I'm still experimenting
- **can also help to solve other problem than building an irc** (this is basically a rediscovery of the Unix way of doing things to handle stream of textual datas) and is also possible for xmpp by replace ii with jj

CHAPTER

ONE

CONTRIBUTE

You can find the source of this documentation here (github), just send a pull request or open bugs there.

This doc is in the process of a rewriting, everything after this might be a bit outdated (but still works)

CHAPTER

TWO

DOCUMENTATION

2.1 Setting up the bot

2.1.1 ii

Everything we are going to do will be based on ii. It's an irc file-system that simplify a lot all the interactions with irc. The suckless page describe the principle.

For debian(-based):

```
apt-get install ii
```

we'll need a reconnection file to keep ii up, here is mine with 2 optionals way to join the irc chan(s):

```
#!/bin/bash
while true
do
    # if you only have one chan
    (sleep 5; echo "/j #my\_chan" > path/to/the/server/in) &
    # if you have multiple chans and want to add new chans easily (you just need to do an "echo chan\
    fori in \`cat chans\`
    do (sleep 5; echo "/j $i" > path/to/the/server/in) &
    done
    ii \\\\
        -i "folder\_where\_ii\_will\_put\_is\_stuff" \\\\
        -s "irc.my\_server\_addr.pouet" \\\\
        -p "port number of the server" \\\\
        -n "bot name" \\\\
        -f "bot name"
done
```

(Don't forget to put the correct informations)

Just launch the script (chmod +x && ./script or sh script). We now have our irc bot.

2.2 RSS

2.2.1 Rsstail

We are going to use rsstail to retrieve our FriendTimeLine. It's a small software that read a RSS (or an atom feed) like a tail -f. But, in rsstail default formating, the url and the title of the RSS item are put on 2 lines instead of one. I don't like that, so I patch rsstail's source code, here is the diff (yes, this is **optional**):

```
430c430
<                                       printf(" %s", heading);
---
>                                       printf("%s ", heading);
434c434
<                                       printf("%s%s\\\\n",
no\_heading?" ":"Title: ", item\_cur -> title);
---
>                                       printf("%s%s",
no\_heading?"":"Title: ", item\_cur -> title);
```

If you don't care, just install it using your pkg manager.

For debian(-based):

```
apt-get install rsstail
```

2.3 Twitter/Status.net

2.3.1 Bti

We also need bti to post our tweet. You can find it here (the last version needs liboauth to be able to post on twitter. I'll explain this later).

2.3.2 Status.net

Let's start with status.net, since this is fucking much easier than with twitter (stupid Oauth).

Getting the FriendTimeLine

(The FriendTimeLine is all your friend dents + your dents).

This is really easy, just do:

```
rsstail -u
http://identi.ca/api/statuses/friends\_timeline/your\_user\_name.rss
-N -z -P -l -i 120 -n 0 > path/to/the/irc/server/\\\\#the\_chan/in
```

Yes, jut one line of bash and yes, it is really easy to transform this into a multi account status.net client.

For the options, *-N* remove "title: " and "link :" from the default formating. *-z* and *-P* is to avoid quitting if errors occurs. *-l* show the links. *-i 120* is the sleeping time between checking for news updates. *-n 0* is the number of item to show at startups, 0 means nothing will appears except the new ones.

(You can ind those links here: http://identi.ca/api/statuses/friends_timeline/your_user_name.rss)

Since not all software are perfect, rsstail might stop working after few days (dunno why), here is a quick fix to keep it up:

```
echo "true" > .while\_cond #only once
while true; do rsstail -u
http://identi.ca/api/statuses/friends\_timeline/your\_user\_name.rss
-N -z -P -l -i 120 -n 0 > path/to/the/irc/server/\\\\#the\_chan/in;
if [ $(cat .while\_cond) = "false" ]; then break; fi; done
```

This will keep rsstail up forever. If you want to stop it, just do an *echo "false" > .while_cond* and kill the rsstail instance (*ps aux | grep rsstail* etc...). **Don't forget to do an *echo "true" > .while_cond* after that to avoid breaking the reboot mechanism of your other rsstail instances.**

One last (dirty) trick to have an header with colors if you wish to distinguish multiple feeds easily:

```
rsstail -u http://identi.ca/api/statuses/friends\_timeline/your\_user\_name.rss -N -z -P -l -i 120 -r
```

Posting

To post we need to parse the output of the irc chan. Here is a small script that do that, the command is *!twitter*:

Warning: dirty code.

```
#!/bin/bash
CHAN="#your\_irc\_chan"
tail -n 0 -f path/to/the/server/$CHAN/out | grep -v --line-buffered "<bot\_name>" | grep --line-buffe
do
    # debug
    printf  '%s\\\\n' "$line"
    message\_text=\`printf '%s\\\\n' "$line" | sed 's/.\\\\+> //'\`
    case $message\_text in         !debug)
          echo "working!" >> path/to/the/server/$CHAN/in ;;
        !twitter\\\\ \*)
            temp=\`printf '%s\\\\n' ""$message\_text"" | sed 's/!twitter \\\\?//'\`
            printf '%s\\\\n'  "$temp" | bti --account user\_name --password your\_password --action u
            if [ ${#temp} -lt 141 ]
            then echo "Grand succès !" > /path/to/the/server/$CHAN/in
            else
                echo "Big tweet, might not work (${#temp} chars)" > path/to/the/server/$CHAN/in
                # here, I still post because identica automaticaly tinyfied the
                # urls he received and I was to lazy to wrote a way to
                # calculate the effective size of a dent with urls in it
            fi ;;
    esac
done
```

As you can see, this script is really simple and very extensible. you can add as much accounts as you want and as much commands as you want. Congratulation, you now have a multi-account status.net client in just a few lines of bash.

If you wrote more code use printf + '' instead of echo to avoid execute of malicious code coming from irc.

The status.net api with curl examples that you can easily add to the previous script.

And voilà, you have everything you need to build your own status.net client.

2.3.3 Twitter

Twitter is really similar to status.net at the exception that they have this stupid oauth. Here are the tricks to works with oauth:

Posting

If it's you identi.ca/status.net account that post on twitter, jump this part and just post on the identi.ca one as specified before.

You'll need bti compiled with 'liboauth <http://liboauth.sourceforge.net/>'_.

If you install liboauth from sources on a debian, don't forget the –prefix=/usr.

This is the easiest part of working with twitter, you just have to configure bti according to this blog post (the last part of the post).

Getting the FriendTimeLine

Now, the dirty part. You'll need twurl, it is a curl with the oauth support, and a webserver where you can place a static file. If you don't have a webserver, you can simple do:

```
python -m SimpleHTTPServer
```

This will launch a basic http server that will display the current folder.

If you are running a debian lenny, you'll need the lasted versions of rubygems (not the one in the pkgs) to make twurl works.

Twurl README explain how to register it to twitter.

Now, you can get your FriendTimeLine. Just do:

```
twurl /1/statuses/home\_timeline.xml > path/to/your/httpserver/folder/twitterfriendtimeline.xml
```

I don't know why, but this command has refused to works in crontab, so I've simply do a (in a screen):

```
while true; do twurl /1/statuses/friends\_timeline.rss > path/to/your/httpserver/folder/twitterfrienc
```

Now you have your FriendTimeLine accessible by rsstail and you can use the same solution than for status.net.

2.4 Misc

2.4.1 Some useful stuff

- Tinyfiying an url with ur1.ca:

  ```
  curl -s http://ur1.ca/ -d "longurl=the url" | sed -u -e '/Your ur1/!d;s: *::g;s:<[^>]*>::g;s:^.*
  ```

- The status.net api.
- Twitter api.
- **Url to have the atom feed of a research on twitter:** http://search.twitter.com/search.atom?q=pouet
- IceRocket help you build complexes query on searching twitter and provide an RSS for those queries.

2.4.2 [Bonus] Pipe a RSS to status.net/twitter

You can pipe the output of rsstail to an status.net/twitter account with bti. This is an easy and stable way to post a RSS feed to status.net/twitter.

Here is an example:

```
rsstail -u http://example.com/rss.xml -n 0 -l -N -z -P -i 30 -Z "[an header]" | bti
```

Warning, since it reads a RSS, it is very sensible, any change in the title or in the url will create a new post on your status.net/twitter account.

2.5 Run your own irc server

If you want to set up your own irc server, I recommend you ngIRCd. It is very easy to configure (only one config file to edit and you're done) and really stable. Thus, it is pkged in debian.

For debian(-based):

```
apt-get install ngircd
```