Yarilo IRC bot user's guide

Thomas Maurice
Version 0.01-dev
10 avril 2013

http://github.com/svartbergtroll/yarilo

Warning : This doc is incomplete and untranslated.
Originally I documented it in French, my mothertongue
but I'm begining to translate it to English so be patient.
It shall be translated and improved soon :)

ЯРНЛФ

Table des matières

1 Introduction **1**
- 1.1 Qu'est ce que c'est ? . . . 1
- 1.2 À quoi ça sert ? . . . 1
- 1.3 Y'a que ça comme fonctions ? . . . 1
- 1.4 Comment étendre les fonctions le robot ? . . . 2
- 1.5 Pourquoi l'utiliser ? . . . 2

2 Installation **3**
- 2.1 Téléchargement . . . 3
- 2.2 Pour les Windowsiens . . . 3
- 2.3 Pour les Linuxiens . . . 3
- 2.4 A savoir . . . 3

3 Configuration **4**
- 3.1 Le fichier bot.conf . . . 4
  - 3.1.1 Les serveurs . . . 4
  - 3.1.2 Les modules . . . 4
- 3.2 Le fichier users/vous . . . 4
- 3.3 Les flags . . . 5
  - 3.3.1 Les flags globaux du robot . . . 5
  - 3.3.2 Les flags de canaux . . . 5
- 3.4 Lancement ! . . . 6

4 Installer des modules avec Ypack **7**
- 4.1 C'est quoi Ypack ? . . . 7
- 4.2 Visualiser les infos d'un paquet . . . 7
- 4.3 Installer un paquet . . . 7
- 4.4 Créer un paquet . . . 7

5 Modules **9**
- 5.1 admin . . . 9
  - 5.1.1 Description . . . 9
  - 5.1.2 Configuration . . . 9
  - 5.1.3 Commandes . . . 9
- 5.2 autojoin . . . 9
  - 5.2.1 Description . . . 9
  - 5.2.2 Configuration . . . 9
  - 5.2.3 Commandes . . . 9
- 5.3 autorejoin . . . 10
  - 5.3.1 Description . . . 10
  - 5.3.2 Configuration . . . 10
  - 5.3.3 Commandes . . . 10
- 5.4 automodes . . . 10
  - 5.4.1 Description . . . 10
  - 5.4.2 Configuration . . . 10
  - 5.4.3 Commandes . . . 10
- 5.5 memo . . . 10
  - 5.5.1 Description . . . 10

5.5.2 Configuration 11
5.5.3 Commandes 11
5.6 user 11
5.6.1 Description 11
5.6.2 Configuration 11
5.6.3 Commandes 11
5.7 version 11
5.7.1 Description 11
5.7.2 Configuration 11
5.7.3 Commandes 12

6 Modules supplémentaires **13**
6.1 python [Linux + Windows] - Expérimental 13
6.1.1 Description 13
6.1.2 Configuration 13
6.1.3 Commandes 13
6.2 MegaHAL [Linux] 13
6.2.1 Description 13
6.2.2 Configuration 13
6.2.3 Commandes 13
6.2.4 Pourquoi que pour Linux ? 14

1 Introduction

1.1 Qu'est ce que c'est ?

Yarilo est un robot IRC, c'est à dire un programme qui est constamment connecté à un serveur IRC pour y administrer un/ou des canaux IRC. Il permet de maintenir ouvert un canal quand aucun utilisateur humain n'y est connecté. Le robot permet également de réagir à certains évènements comme un message sur un canal, la jonction d'un canal par un utilisateur etc... Il peut interagir avec le serveur et les utilisateurs de celui ci. Les fonctionnalités du robot sont extensible par le biais de modules qui sont comme le robot codés en C++.

1.2 À quoi ça sert ?

Yarilo sert a maintenir ouvert un canal, mais pas seulement ; par le biais de ses modules, Yarilo peut contribuer à l'animation et la fonctionnalité d'un canal pour les serveurs qui ne fournissent que peu de fonctions aux utilisateurs. Par exemple le module `memo` permet aux utilisateurs de s'envoyer des memos entre eux et de communiquer en différé par le biais de ces pseudo boîtes mail.

1.3 Y'a que ça comme fonctions ?

Non bien sûr que non, le robot dispose également d'une gestion interactive (comprenez sur IRC) des modules et des utilisateurs par le biais des modules `mods` et `user` (très originaux les noms, hein ?).

Il dispose également de la possibilité d'auto-moder (op/hop/voice/kickban) les utilisateurs configurés comme tels dès leur entrée sur le canal, de fixer des limitations d'utilisateurs pour éviter que le canal soit join/part floodé, ou de garder constant le topic en leprotégeant des utilisateurs qui ne sont pas abilités à le changer. C'en est pour le moment tout de la fonction d'administrateur du bot.

Il peut également animer un canal en proposant un module d'intelligence artificielle permettant au bot de devenir un « chatterbot » qui répondra de temps en temps à vos utilisateurs et apprendra des conversations qu'ils auront entre eux (ce module ne fonctionne que pour Linux malheureusement).

(Pour plus d'infos sur les fonctions du robot, voir la section « Modules » qui détaille les modules livrés par défaut avec le robot.)

1.4 Comment étendre les fonctions le robot ?

Les modules d'extension doivent être codés en C++ via une API assez simple (les codes sources des modules et un bref coups d'oeil aux classes de `include/` devrait vous en persuader. Cependant je conçois que le C++ n'est pas fatalement un langage très abordable à ceux qui ne l'auraient jamais pratiqué et c'est pour cela que le robot est livré avec un module `python` qui fournit une interface avec ce langage, plus simple d'utilisation et avec une API similaire à celle du C++ (le module étant pour le moment expérimental est incomplet, je vous déconseille de le charger sans savoir précisément ce que vous faites) et qui permet à l'utilisateur de coder ses propres modules sois forme de scripts .py tout simples !

1.5 Pourquoi l'utiliser ?

Parce qu'il est stable. Le robot est basé sur une structure modulaire ce qui lui permet d'évoluer dans ses fonctionnalités simplement en ajoutant/supprimant des modules en n'ayant quasiment jamais besoin de le redémarrer (sauf en cas de mise à jour du coeur du robot). Hors mises àjour du coeur, toutes les modifications peuvent se faire depuis IRC.

Parce qu'il est léger. Avec tout ses modules chargés (IA + interpreteur python notemment) le robot ne consomme qu'entre 2% et 3% de mon temps de calcul processeur et 0.1% de RAM, et je n'ai pas une machine de guerre. Il occupe également peu d'espace disque, la version de base demande autour de 3Mo maximum pour les binaires seuls.

Parce qu'il est multi serveur. Yarilo supporte une configuration multi-serveur qui lui permet d'établir à lui tout seul des chan-relays, de délivrer des mémos sur plusieurs serveurs d'un seul coup, vous n'aurez pas à multiplier les robots en même temps que vous multiplierez les serveurs !

Parce qu'il est facile à configurer. La configuration se fait dans un fichier `conf/bot.conf` et celle des utilisateurs dans le dossier `conf/users` avec une syntaxe claire et compréhensible (et commentée). Notez que le seul utilisateur que vous aurez à configurer à la main sera vous même puisqu'une fois que vous serez dans la configuration du robot, toutes les autres manipulations seront exécutables simplement depuis IRC.

2 Installation

2.1 Téléchargement

Tout d'abbord, téléchargez la dernière version binaire (ou les sources si vous souhaitez compiler le programme) à l'adresse : `http://www.yarilo.fr.nf/dl.php`

2.2 Pour les Windowsiens

Si vous êtes sous Windows, je vous conseille très vivement de ne pas le compiler et de prendre la distribution binaire directement. Sinon il vous faudra installer l'environnement de compilation MinGW (dans sa bonne version sinon les Makefile ne fonctioneront pas) et encore d'autres joyeusetés qui vont rallonger grandement le temps qu'il vous faudra avant d'avoir un programme fonctionnel.

2.3 Pour les Linuxiens

Si vous êtes sous Linux, téléchargez les sources et `make` attendez une ou deux minutes, allez boire un café, trempez le dans l'huile, trempez le dans l'eau et vous aurez un robot prêt à l'emploi.

Attention 1 Si pendant la compilation du robot, le compilateur vous formule des insultes avec "ssl" dedans c'est que vous n'avez pas installé le paquet de développement de OpenSSL (en général `libssl-dev`).

Attention 2 Si pendant la compilation il vous arrive des bricoles lors de la compilation du module Python c'est que vous n'avez pas installé les fichiers de developpement de python (le paquet `python-dev` en théorie.)

2.4 A savoir

Si vous avez le choix, utilisez le robot sous Linux, ça sera bien plus facile lors des mises à jour où un simple `make` suffira. De plus, pour des raisons indépendantes de ma volonté certains modules peuvent ne pas fonctionner sous Windows (le module d'intelligence conversationnelle notemment) a cause d'erreurs provenant de la bibliothèque de MegaHAL qui m'est étrangère mais qui fonctionne sans soucis sous Linux.

3 Configuration

3.1 Le fichier bot.conf

Maintenant que vous avez un robot compilé, frais et dispos vous pouvez commencer à le configurer. Chose promise, chose dûe la configuration est bidon, rendez vous pour commencer dans le fichier `conf/bot.conf`. Ce fichier permet deux choses, primo de spécifier les serveurs auxquels se connecter, comment s'y connecter et sous quel pseudo, segundo il permet de spécifier quels modules charger au démarrage.

3.1.1 Les serveurs

Un serveur s'ajoute grâce à la directive `server` comme il suit :
`server <serv_label> <serv_host> <serv_port> <nick> <ident> <pass> <ssl>`

- serv_label est le nom du serveur tel qu'il sera utilisé dans le bot
- serv_host est l'host du serveur (exemple irc.epiknet.org)
- serv_port est le port auquel on se connecte
- nick est le pseudo du bot
- ident est l'identifiant du robot (sans importance généralement)
- pass est le mot de passe si le pseudo est protégé
- ssl = 1 si on se connecte en ssl, ssl = 0 sinon

Attention ! Les variables doivent être en un seul mot, pas de passwords en plusieurs mots, sinon le programme lira mal les informations.

3.1.2 Les modules

Un module se charge grâce à la directive `loadmodule` comme il suit :
`loadmodule mod_name`
Où `mod_name` est le nom du module qui correspond au fichier `modules/mod_name.so` sous Linux ou `modules/mod_name.dll` sous Windows.

3.2 Le fichier users/vous

Il faut maintenant configurer votre fichier d'utilisateur, la configuration commentée ci dessous est simple et tiens lieu d'exemple. Enregistrez ce fichier sous conf/users/pseudo si votre pseudo est "pseudo". Notez que votre champ ID DOIT également être égal a "pseudo". Par exemple le fichier ci dessous s'enregistre sous conf/users/Epsilon

```
1 # User Epsilon for Yarilo bot
2
```

```
3 # General settings
4 ID Epsilon
5
6 # Servers configuration
7 # Nicknames configuration for Epiknet
8 SERVER_N Epiknet Epsilon
9 SERVER_N Epiknet Epsilon_
10
11 # Nicknames configuration for Fooserv
12 SERVER_N Fooserv Foonick
13
14 # Hosts configuration for Epiknet
15 SERVER_H Epiknet ~epsilon@EpiK-5CFA385E.fbx.proxad.net
16 SERVER_H Epiknet ~epsilon@2DD12490.95FD2338.9E3FAB29.EpiK
17
18 # Hosts configuration for Fooserv
19 SERVER_H Fooserv ~foo@test.foobar
20 SERVER_H Fooserv ~foo2@test.foobar
21
22 # Partyline configuration
23 PL 1
24 PL_N Epsilon
25 PL_P 206c80413b9a96c1312cc346b7d2517b84463edd
26 FLAGS OMRQPUohv
27
28 # End of Epsilon configuration
```

exemple_user

Les directives de partyline sont inutiles pour le moment dans la mesure où cette fonctionnalité n'est pas encore implémentée dans le robot.

3.3 Les flags

Le robot gère les permissions selon un système de flags qui correspond à un certain de niveau de permission de l'utilisateur.

3.3.1 Les flags globaux du robot

- O : Owner - proprio du robot, tout les droits
- M : Master
- R : Utilisateur privilégié de niveau 3
- Q : Utilisateur privilégié de niveau 2
- P : Utilisateur privilégié de niveau 1
- U : Utilisateur normal, accès aux commandes de base
- N : Accès a rien

3.3.2 Les flags de canaux

- o : Auto-op (sur tout les chans ou le bot est op)

- h : Auto-hop (sur tout les chans ou le bot est op)
- v : Auto-voice (sur tout les chans ou le bot est op)
- d : Auto-deop (sur tout les chans ou le bot est op)
- k : Auto-kick (sur tout les chans ou le bot est op)
- m : Martyr (auto deop/voice/hop & kick & aucun accès a quoi que ce soit)

3.4 Lancement !

Vous pouvez maintenant lancer le robot en double cliquant sur Yarilo.exe sous Windows ou en tapant ./Yarilo sous Linux. Notez que si vous êtes sous Windows, l'antivirus Avast ! signale le programme comme étant malveillant, je ne sais pas pourquoi. Si vous avez des doutes (pourquoi pas hein) lancez le dans la SandBox (cliquez sur l'option qui va bien) et le programme s'exécutera quand même sans aucun problème (ni dangers pour votre ordinateur).

4 Installer des modules avec Ypack

4.1 C'est quoi Ypack ?

Ypack est le gestionnaire de paquets du robot. Il permet d'installer des paquets indépendament du système d'exploitation. Cela évite de devoir systématiquement compiler les modules depuis les sources sur votre système ou de devoir faire 15 distributions binaires différentes pour le même module. Les paquets .ypkg sont en fait une concaténation des fichiers source, des binaires windows et/ou linux ainsi que des DLLs externes et d'autres informations relatives au module.

4.2 Visualiser les infos d'un paquet

Pour visualiser les informations sur un paquet, il faut taper dans la console

```
./ypack --display paquet.ypkg
```

et les infos s'affichent. Notez que si vous faites juste

```
./ypack paquet.ypkg
```

les infos s'afficheront et le programme vous demanderas si vous voulez installer le paquet. Pour les utilisateurs de windows, un glissé déplacé du fichier .ypkg sur ypack.exe aura le même effet.

4.3 Installer un paquet

Tout d'abord il faut télécharger le paquet .ypkg depuis `http://yarilo.fr.nf/modules.php` (ou depuis le site du développeur du module) puis le copier dans le répetoire du robot. La commande pour installer le module est simple :

```
./ypack --install paquet.ypkg
```

pour installer les binaires ou

```
./ypack --install-src paquet.ypkg
```

pour les binaires plus les sources.

4.4 Créer un paquet

Pour cela vous avez besoin d'un fichier particulier dans votre dossier de module, le fichier PACKAGE. On supposera que votre dossier est module/test et que votre module est de nom "test". Le fichier PACKAGE se remplit comme il suit :

```
1 NAME nom_du_module
2 AUTHOR auteur_du_module
3 WIN32 1 ou 0           (selon si ca marche sous windows ou pas)
4 LINUX 1 ou 0           (selon si ca marche sous linux ou pas)
5 SRC fichier.cpp        (ou fichier.cpp est module/test/fichier.cpp)
6 DATA fic.dat           (ou fic.dat se rapporte a data/fic.dat
```

```
7 EXE_WIN32 test.dll     (ou test.dll est modules/test.dll
8 EXE_LINUX test.so      (ou test.so est modules/test.so
9 LIB_WIN32 lib.dll      (a la racine du repertoire sous windows)
10 LIB_LINUX lib.so      (a la racine du repertoire sous linux)
```

Ensuite tapez la commande

```
./ypack --create modules/test/PACKAGE
```

en étant SUR que tout les fichiers (.so, .dll etc) existent et sont à jour. Votre paquet est prêt à être distribué !

5 Modules

5.1 admin

5.1.1 Description

Ce module regroupe quelques fonctions que seul l'administrateur du robot (le +O) peut utiliser.

5.1.2 Configuration

Aucune en particulier, seulement un utilisateur qui a le flag O

5.1.3 Commandes

- `!admin.die` Eteint le robot
- `!admin.join <#chan>` Join le chan
- `!admin.part <#chan>` Part du chan

5.2 autojoin

5.2.1 Description

Ce module permet de rejoindre automatiquement un canal à la connexion au serveur.

5.2.2 Configuration

Il se configure dans data/autojoin.lst ou on ajoute un canal en ajoutant une ligne au fichier, comportant le label du serveur puis le nom du chan :
`ServLabel #chan`

5.2.3 Commandes

Aucune commandes particulières

5.3 autorejoin

5.3.1 Description

Rejoint automatiquement un canal après en avoir été kické.

5.3.2 Configuration

Aucune.

5.3.3 Commandes

Aucune.

5.4 automodes

5.4.1 Description

Applique automatiquement les modes aux utilisateurs configurés (ex +o aux utilisateurs o, +h aux h etc...)

5.4.2 Configuration

Aucune.

5.4.3 Commandes

Aucune.

5.5 memo

5.5.1 Description

Permet aux utilisateurs de s'envoyer des mémos en différé par l'intermédiaire du bot.

5.5.2 Configuration

Automatique à chaque lancement du module, le robot fait tout.

5.5.3 Commandes

- `!memo.read` Lit vos mémos.
- `!memo.send <User> <message>` Envoi un message à l'user dont l'ID dans lebot est User.
- `!admin.purge` Vide les boites mémo de vos utilisateurs en supprimant les vieux messages.

5.6 user

5.6.1 Description

Gère vos utilisateurs, le flag requis est M.

5.6.2 Configuration

Aucune.

5.6.3 Commandes

- Pour les admins (M) : `!user.[add|del|addhere|delhere|chid|save|list|info]`
- `!user.[addnick|delnick|addhost|delhost]`

Tapez une commande en particulier pour avoir sa syntaxe.

5.7 version

5.7.1 Description

Fournit une réponse au CTCP VERSION.

5.7.2 Configuration

Dans le fichier data/version.txt, mettex ce que vous voulez.

5.7.3 Commandes

Aucune.

6 Modules supplémentaires

6.1 python [Linux + Windows] - Expérimental

6.1.1 Description

Fournit un moyen de programmer des scripts python se comportant comme des modules C++ avec une API similaire. Les scripts se placent dans le répertoire modules/ et sont chargés dynamiquement par l'utilisateur. L'API sera documentée plus tard, elle est encore loin d'être aboutie pour le moment.

6.1.2 Configuration

Vous devez obligatoirement avoir Python 2.7 installé sur votre machine lors de l'exécution. Si vous lancez Yarilo avec le module python chargé au lancement sans avoir python installé sur votre machine le programme crashera purement et simplement. Veillez à mettre le "python27.dll" fourni dans l'archive dans le même répertoire que Yarilo.exe si vous êtes sous windows.

6.1.3 Commandes

- `!python[load|reload|unload] <script>` ou script est le fichier modules/¡script¿.py
- `!python.list` Liste les scripts chargés.

6.2 MegaHAL [Linux]

6.2.1 Description

Fournit une IA au robot, le robot apprend des conversations qui ont lieu sur le canal et réagit parfois à celles ci dans les limites spécifiées dans le fichier data/megahal.conf

6.2.2 Configuration

Editez le fichier data/megahal.conf, il est suffisemment bien commenté !

6.2.3 Commandes

- `!megahal.save` sauve le « cerveau » de MegaHAL
- `!megahal.reconf` recharge le fichier de config après modification

6.2.4 Pourquoi que pour Linux ?

Parce que la bibliothèque que j'utilise pour l'IA n'est pas développée par moi et que je ne la comprend quasiment pas. Je me contente de fournir une interface basique avec IRC de manière à obtenir un robot parlant. Cette bibliothèque fonctionne très bien sous Linux mais cause des erreurs de segmentation (fatales) lors de son utilisation sous Windows, c'est pour cela que ce module n'est pas fourni par défaut.