

Bot IRC Syd v0.1

Jean-Pierre Lozi

2 juin 2008

Syd est un petit bot IRC, contenant un certain nombre de fonctionnalités, notamment le stockage et la restauration d'états.

1 Lancement

Pour lancer le programme, il suffit d'ouvrir le fichier `start.ss` avec *DrScheme* (langage *pretty big*) et de le lancer avec *Exécuter*. On peut arrêter l'exécution en utilisant *Stopper*.

2 Configuration

Il est possible de configurer le programme en modifiant les fichiers suivants :

- `syd-configuration.ss` : pour configurer certains paramètres du bot IRC syd.
- `syd-quote-server-configuration.ss` : pour configurer le serveur de citations.

Les commentaires de ces fichiers devraient permettre de comprendre aisément les différents paramètres.

3 Fonctionnalités

Voici une liste des fonctionnalités du programme.

- Possibilité de donner une fonction à exécuter lorsqu'une expression régulière est détectée.

Par exemple, on pourra essayer :

```
>>> syd: si quelqu'un dit "fac ([0-9]+)", exécute
(lambda (regular-expression output-port sender command args state env)
  (letrec ((fac (lambda (n) (if (zero? n) 1 (* n (fac (- n 1)))))))
    (syd-utils-send-private-message output-port
      (syd-utils-association-list-get state 'mode)
      (syd-utils-association-list-get env 'channel)
      (expr->string (fac (read-from-string
                          (cadr regular-expression))))
      state)))
```

Après cette déclaration, lorsque `fac n` sera présent sur le canal (où `n` est un nombre), la factorielle de ce nombre sera calculée. Les paramètres dans la fonction sont : le port de sortie, l'émetteur, la commande reçue, les arguments, l'état courant, et une liste d'association contenant la plupart des variables globales du module `syd`¹.

Il faut nécessairement renvoyer un état valide, pour éviter un plantage du bot. Les différentes expressions régulières sont évaluées dans l'ordre de déclaration des fonctions, et lorsqu'on a un match, les autres ne sont pas évaluées. Toutes les fonctions des module `syd-utils` sont disponibles. L'expression régulière utilisée est :

```
".*<nickname>.*(?:si quelqu'un dit.*\\")(.*)(?:i:\\\\".*ex[eé]cute )(.*)"
```

- Enregistrement de l'état du bot. Pour cela, on pourra par exemple essayer :

```
>>> <nickname>, sauve ton état sous le nom <nom>
```

L'expression régulière utilisée est la suivante :

```
".*<nickname>.*(?:sauve|enregistre|stocke) (?:i:ton [eé]tat sous le nom) (.*)"
```

- Restauration de l'état du bot. Pour cela, on pourra par exemple essayer :

```
>>> <nickname>, restaure l'état <nom>
```

L'expression régulière utilisée est la suivante :

```
".*<nickname>.*(?:restaure l'[eé]tat) (.*)"
```

- Répétition d'une phrase envoyée deux fois sur le canal. Gestion de quatre modes :

Le mode normal, mode par défaut.

Le mode rainbow, qui fait écrire le bot avec les couleurs de l'arc-en-ciel.

Le mode hacker, qui fait écrire le bot en l33t.

Le mode majuscules, qui fait écrire le bot en majuscules.

Pour changer de mode, on pourra par exemple utiliser :

```
>>> <nickname>, passe en mode <mode>
```

L'expression régulière utilisée est :

```
".*<nickname>.*(?:i:passe en mode) (.*)"
```

- Réponse à la vie, l'univers, et tout le reste. Le bot répond 42 lorsque la question est soulevée. On pourra par exemple écrire :

```
>>> Quelle est la réponse à la vie, l'univers, et tout le reste?
```

Ou encore :

```
>>> What is the answer to life, the universe, and everything?
```

Les expressions régulières utilisées sont :

```
".*(?:i:answer.*life.*universe.*everything).*"
```

et :

```
".*(?:i:vie.*univers.*tout le reste)."
```

- Nom d'hôte. Le bot peut renvoyer le nom d'hôte de l'utilisateur. On pourra par exemple écrire :

```
>>> <nickname>, c'est quoi mon host name?
```

Les expressions régulières utilisées sont :

¹Les variables disponibles sont : `service-port`, `current-nickname`, `first-name`, `last-name`, `connection-message`, `encoding`, `verbosity-ratio`, `default-number-of-nicknames-on-hl`, `default-number-of-recent-users`, `continuations`, `messages-1/someone-joins-the-channel`, `messages-1/someone-joins-the-channel-again`, `messages-1/someone-leaves`, `messages-1/i-am-kicked`, `messages-1/someone-is-kicked`, `messages-1/someone-tells-me-o<`, `messages-0/i-leave`.

- ".*<nickname>(?!.*mon.*h[ôo]te).*"
 et :
 ".*<nickname>(?!.*mon.*host.*name).*"
- Nom d'utilisateur. Le bot peut renvoyer le nom de l'utilisateur. On pourra par exemple écrire :


```
>>> <nickname>, c'est quoi mon nom d'utilisateur?
```

 Les expressions régulières utilisées sont :


```
".*<nickname>(?!.*mon.*nom.*utilisateur).*"
      et :
      ".*<nickname>(?!.*mon.*user.*name).*"
    
```
 - Réponse aux o<, O< et coin. Le bot renvoie une réponse adéquate lorsque ce type de message est reçu. On pourra par exemple écrire :


```
>>> <nickname>, o<
```

 Les expressions régulières utilisées sont :


```
".*<nickname>.*[oO00]\\<.*"
      et :
      ".*<nickname>.*(?!:coin).*"
    
```
 - Réponse à la commande !hl. La commande !hl donne les noms d'un certain nombre de personnes présentes sur le canal. Le nombre par défaut de noms est configurable dans le fichier `syd-configuration.ss`. Un paramètre entier peut être donné, qui spécifiera le nombre de noms à afficher. Si le paramètre n'est pas reconnu, le nombre par défaut de noms est utilisé. S'il y a moins d'utilisateurs sur le canal que le nombre de noms choisi, le nom de tous les utilisateurs sera affiché. On pourra par exemple écrire :


```
>>> !hl 3
```

 L'expression régulière utilisée est :


```
"^!hl ([0-9]*)"
    
```
 - Calculs. Le bot est capable d'effectuer des calculs avec les opérations de base de *mz-scheme*. On pourra par exemple essayer :


```
>>> <nickname>, évalue (letrec ((f (lambda (n)
                                     (if (zero? n) 1 (* n (f (- n 1)))))) (f 3))
```

 L'expression régulière utilisée est :


```
".*" current-nickname ".*(?!:calcule|[ée]value) (.*)"
    
```
 - Réponses aux salutations. Le bot répond à diverses salutations : bonjour, salut, etc. On pourra par exemple écrire :


```
>>> bonjour <nickname>!
```

 Les expressions régulières utilisées sont :


```
".*<nickname>.*(?:^[^éeËëàùÛüïî[:alnum:]]
      (?!:lu|salut|kikoo+|coucou|bonjour|salut|yop|plop|hello|hey)
      (?:$|[^éeËëàùÛüïî[:alnum:]]).*"
      et :
      ".*(?:^[^éeËëàùÛüïî[:alnum:]]
      (?!:lu|salut|kikoo+|coucou|bonjour|salut|yop|plop|hello|hey)
      (?:$|[^éeËëàùÛüïî[:alnum:]]).*<nickname>.*"
    
```
 - Réponses aux "re". On répond spécifiquement lorsqu'un utilisateur nous envoie un message "re". On pourra par exemple écrire :

>>> re, <nickname>...

Les expressions régulières utilisées sont :

```
".*<nickname>.*(?:^[^éeÈëàùÛüï[:alnum:]])(?i:re)(?:$|^[^éeÈëàùÛüï[:alnum:]]).*"
```

- Citation d'un auteur particulier. Le bot est capable de citer un auteur particulier ou une source particulière. L'argument donné sera recherché parmi les sources des citations. Attention : si la citation trouvée est trop longue, rien ne sera affiché. On pourra par exemple écrire :

>>> <nickname>, donne une citation de VanDamme

Ou encore :

>>> <nickname>, dis un proverbe

Les expressions régulières utilisées sont :

```
".*<nickname>.*(?i:citation|quote|message|histoire|texte).*(?i:d[eu]s?) (.*)"
```

et :

```
".*<nickname>.*(?i:citation|quote|message|histoire|texte).*(?i:d[eu]s?) (.*)"
```

- Changement d'humeur. Le bot peut être de bonne ou de mauvaise humeur. Selon son humeur, il utilise des expressions différentes lors de la réponse aux différents messages. Celles-ci sont configurables dans le fichier `syd-configuration.ss`. On pourra par exemple écrire :

>>> <nickname>, sois de bonne humeur.

ou :

>>> <nickname>, sois de mauvaise humeur.

Les expressions régulières utilisées sont :

```
".*<nickname>.*(?i:sois de bonne humeur).*"
```

et :

```
".*<nickname>.*(?i:sois de mauvaise humeur).*"
```

- Gestion du "di". Lorsque la chaîne "di", non suivie d'un n ou d'un m est présente dans une phrase, alors on répète tout ce qu'il y a après. Si plusieurs chaînes "di" sont présentes, on ne répète que ce qui se trouve après le dernier "di". On pourra par exemple écrire :

>>> Je suis un disciple.

L'expression régulière utilisée est :

```
".*di(?:[nm]).*"
```

- Gestion du "cri". Lorsque la chaîne "cri", non suivie d'un n ou d'un m est présente dans une phrase, alors on répète tout ce qu'il y a après, en majuscule, avec "!!!" à la fin. De la même manière que pour le "di", si plusieurs chaînes "cri" sont présentes, on ne répète que ce qui se trouve après le dernier "cri". On pourra par exemple écrire.

>>> Je suis critique.

L'expression régulière utilisée est :

```
".*cri(?:[nm]).*"
```

L'application essaie de trouver un appel à l'une des fonctionnalités citées ci-dessus dans l'ordre. Lorsqu'une commande est identifiée, les autres expressions ne sont pas analysées pour éviter de nombreuses réponses de la part du bot.

Enfin, si la phrase ne correspond à aucune des fonctionnalités ci-dessus, la phrase est transmise au serveur de citations, qui utilise un algorithme assez complexe pour sélectionner

une citation en rapport avec la phrase donnée. Cet algorithme dépend de plusieurs paramètres configurables dans le fichier `syd-quote-server-configuration.ss`.

Le serveur de citations commence par tirer un nombre au hasard entre deux valeurs configurables. Ce nombre représente le nombre de mots à considérer dans le message reçu. Le message est ensuite filtré : on lui enlève les mots trop courts (le seuil étant configurable). Puis, si on a moins de mots que le nombre tiré aléatoirement, on peut soit s'arrêter, soit continuer avec tous les mots (ceci est également configurable). Sinon, on choisit aléatoirement le bon nombre de mots, les mots les plus longs ayant plus de chances d'être sélectionnés. Chacun des mots est ensuite recherché dans la liste de citations. Au plus il y a d'occurrences de ces mots dans l'une des citations, au plus le score attribué à cette citation sera élevé. On peut vouloir que les citations contiennent tous les mots tirés au sort, ou seulement une sous-partie d'entre eux.

Enfin, parmi les citations trouvées (s'il y en a), on en choisit une. Celle-ci est choisie aléatoirement, en ayant plus de chances d'être sélectionnée si son score est élevé.

Notons par ailleurs que le bot envoie des messages dépendant de son humeur lorsque l'un des messages PING, KICK, PART, QUIT ou JOIN sont reçus.

4 Structure

Le programme est composé de 3 modules et d'un fichier de conversion. Les 3 modules sont :

- **syd** : module principal de l'application. Ce module s'occupe d'ouvrir la connexion et de traiter les différents messages.
- **syd-utils** : module contenant des fonctions utilitaires. Ce module contient diverses fonction utilitaires utilisables par les autres modules.
- **syd-quote-server** : serveur de citations. Ce module lance un serveur de citations dans un nouveau thread. Celui-ci répond à des messages qui lui sont envoyés sur un tube.

Enfin, le fichier `convert-fortune-fr-files.ss` permet de convertir les citations du paquetage fortune-fr en un format lisible par notre application.

5 Licence et contact

Ce programme est diffusé sous la GPLv2. Une copie de la licence est fournie avec le programme.

Pour tout commentaire ou suggestion, merci d'envoyer un mail à jean-pierre@lozi.org.