Mon premier vrai programme en Go

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 8 décembre 2009

https://www.bortzmeyer.org/go-langage.html

Mon premier « vrai » programme en Go est un client whois, une mise en œuvre du RFC 3912 [1].

Go, créé par Google, est surtout intéressant comme langage de programmation « système ». Il est donc concurrent de C et C++, sur un créneau où il y a relativement peu de candidats (certains, comme D, n'ont eu aucun succès). Certains commentateurs s'étaient étonné que Google crée encore un nouveau langage de programmation lors qu'il y en a déjà des milliers mais, contrairement aux langages fonctionnels, les langages « système » sont bien plus rares.

Le code de mon client whois est en (en ligne sur https://www.bortzmeyer.org/files/whois.go). Mon point de départ avait été socketgo <http://github.com/tcpip4000/socketgo>. (Un autre programme de niveau de complexité analogue est un client pour l'entrepôt de données de Rennes <http://data.keolis-rennes.com/>, (en ligne sur https://www.bortzmeyer.org/files/gostar.go).) Quelques points qui m'ont interpellé pendant le développement du client whois :

- Le compilateur qui a un nom différent selon le type du processeur (très énervant, pour écrire des Makefile portables, ce n'est qu'après que j'ai appris à mettre `include $(GOROOT)/src/Make.$(GOARCH)`),
- La très bonne documentation des paquetages standard <http://golang.org/pkg/>,
- La conversion obligatoire des `string` en `[]byte`, les premières étant nécessaires pour l'affichage et les seconds servant aux entrées/sorties (j'ai créé un `Buffer` pour cela, y avait-il une meilleure solution ?),
- La séparation entre la création d'une variable (où on peut lui donner une valeur avec :=) et sa mutation ultérieure (où on doit utiliser =),
- L'absence d'exceptions, probablement le plus gros manque du langage,
- La déclaration automatique d'une variable, avec inférence de types, ce qui m'a rappelé Haskell,
- Et, bien sûr, comme tous les débutants en Go, le fait qu'une variable déclarée **doive** être utilisée, sous peine du désormais célèbre message "*XXX declared and not used*".

Mon deuxième programme, pour illustrer le parallélisme en Go est un serveur du protocole echo, NoNewContent <https://framagit.org/bortzmeyer/nonewcontent>. Plus perfectionné et tout aussi parallèle, un serveur de noms en Go <https://www.bortzmeyer.org/dnsserver-en-go.html>.

J'ai aussi fait un exposé en français sur Go <https://www.bortzmeyer.org/expose-go.html>. Je stocke mes liens vers des ressources Go en <http://delicious.com/bortzmeyer/golang>.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc3912.txt