# Le choix d'un langage de programmation

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 25 janvier 2008

https://www.bortzmeyer.org/choix-langage-prog.html

---

Le choix d'un langage de programmation pour une tâche de développement logiciel est souvent un choix difficile et qui suscite des débats passionnés. Utiliserons-nous Java ? Perl ? Caml ? Lua ? Ces débats sont parfois qualifiés de « religieux » du fait de l'enthousiasme qu'y mettent leurs participants, et de la rapidité avec laquelle ils écrasent les alternatives (« Java est le standard de l'industrie » ou « Python est trop lent ») sans trop chercher à collecter des faits précis. Mais appeler ces débats « religieux » est en fait souvent un moyen de nier l'importance de ce choix et de se moquer sans risques de ceux qui cherchent à défendre leur choix. C'est une fuite bien commode alors que, comme tout choix technique, celui d'un langage de programmation peut être basé sur des faits, à condition de chercher ces faits, non seulement dans le langage lui-même, mais aussi dans tout son environnement, technique et social.

On pourrait vouloir un seul langage de programmation <http://martinfowler.com/bliki/OneLanguage.html>, qui éviterait la difficulté du choix (les discours sur tel langage qui serait « le standard de l'industrie » relèvent souvent de cette vision). Mais ce n'est pas le cas. Il y a donc un vrai choix et qualifier les débats sur ce choix de « religieux » est un moyen peu courageux de refuser la discussion. Par contre, il est exact qu'il faut un gros effort sur soi-même pour ne pas trop se laisser aveugler par la passion et pour étudier les faits. Cette passion est normale (heureusement que les programmeuses sont passionnées par la programmation) et elle est même raisonnable, dans la mesure où le choix de l'outil influe, non seulement sur la productivité (si le langage d'assemblage a presque disparu, ce n'est pas par hasard, c'est largement à cause de la plus faible productivité qu'il permet), mais également sur les modes de pensée. **L'outil détermine le travail**, même si les décideurs, ne comprenant pas les arguments techniques du choix d'un langage, continuent à prétendre qu'on peut réaliser n'importe quelle tâche, avec n'importe quel style, dans tous les langages. C'est factuellement exact mais cela ne mène pas loin. C'est comme dire que tous les moyens de transport se valent et qu'on peut traverser l'Amérique à pied. Oui, on peut. Comme on peut réécrire un SGBD en assembleur. Et alors ? Qui va le faire ?

Il y a donc bien des arguments techniques au choix d'un langage. Un langage où il faut gérer la mémoire soi-même (allocation et désallocation, éviter les pointeurs pendants) comme C est une source de bogues sans fin. Un langage où les variables sont automatiquement déclarées (comme les vieux Basic ou comme Perl ou PHP par défaut) permet des bogues très difficiles à détecter en cas de faute de frappe sur le nom d'une variable. Ces arguments sont très importants et doivent être pris en compte.

Mais certains professionnels des langages de programmation, emportés par leur goût pour ces analyses et ces discussions, oublient un autre aspect : le langage lui-même, tel que spécifié dans son manuel de référence, n'est qu'une partie du système avec lequel les programmeurs vont travailler. Il y a aussi la disponibilité de bibliothèques, l'existence de plusieurs implémentations de qualité (et, si on développe du logiciel libre, sous une licence acceptable), la taille de la communauté, qui conditionne la facilité à recruter, l'aide qu'on pourra trouver...

Parmi ces critères non strictement liés au langage, on trouve la disponibilité de bibliothèques toutes faites pour assurer des tâches courantes comme parler à un serveur LDAP ou bien calculer une somme de contrôle SHA. Un langage comme Ada avait beaucoup souffert du manque de telles bibliothèques. À l'inverse, Haskell en a trop sur chaque sujet, et aucune n'étant « standard » (il existe par exemple de nombreuses façons <http://stackoverflow.com/questions/1361307/which-haskell-xml-library-t> de traiter du XML en Haskell). Lua a de telles bibliothèques mais elles ne sont pas installées par défaut <https://www.bortzmeyer.org/lua-sur-machine-generaliste.html>. À l'inverse, un langage comme Perl doit une partie de son succès à la CPAN, un immense ensemble de bibliothèques de qualité, documentées, cohérentes et facilement disponibles.

Parmi les critères plus sociaux, il y a par exemple la taille de la communauté <http://beust.com/weblog/archives/000471.html>, qui gouverne la facilité à trouver des programmeurs ou de l'aide. Encore que la taille puisse avoir des effets pervers comme le fait que les listes de diffusion d'un langage très populaire seront trop actives pour qu'on puisse s'en servir. Ou comme le fait qu'un langage « trop » populaire attirera des masses de programmeurs sans éclat (le paradoxe Python <http://www.paulgraham.com/pypar.html>).

Je pense que, dans le choix d'un langage de programmation pour une tâche **pratique** (i.e. écrire un programme), on ne prend pas en compte que le langage lui-même mais aussi ce que j'appelle la « mentalité de la communauté ». Certes, cette mentalité n'est pas obligatoire (on peut choisir le langage et ignorer cette mentalité) mais elle influence fortement la documentation, les programme écrits par les autres (qu'on lira et dont on récupérera du code), l'aide qu'on peut avoir sur des listes de diffusion, etc.

Cette mentalité est la **culture** du langage. Si on ne travaille pas tout seul, si on est dans un environnement de logiciel libre, il faut bien en tenir compte. Par exemple, je conseille à tous les débutants en programmation de **lire** des programmes. Avec Perl, cela va les amener à lire du code illisible même si « on peut programmer proprement en Perl » (on peut, de même qu'on peut construire soi-même le bateau avec lequel on va traverser l'Atlantique).

Donc, en pratique, c'est surtout la « mentalité de la communauté » qui me tient à l'écart de PHP et Java, bien plus que les problèmes de ces langages. Par exemple, la culture PHP est de bricolage, sans réfléchir, sans faire attention aux problèmes de sécurité. Bien sûr qu'on peut écrire des programmes sûrs en PHP. Mais la culture du monde PHP ne s'y intéresse pas et donc ne le fait pas. Inversement, j'apprécie beaucoup la mentalité des Pythoniens ou des Haskelliens, qui peuvent discuter de questions avancées **et** répondre aux questions des débutants. Ah, et pour ajouter une note humoristique, la fameuse « hiérarchie des programmeurs <https://lispmachine.files.wordpress.com/2013/03/chart-luke-welling.jpg> » contient beaucoup d'éléments de vérité...

Pour finir sur une note concrète, un excellent article <http://wiki.mozilla.org/Bugzilla:Languages> de l'équipe de Bugzilla illustre très bien une bonne méthode de choix de langage. Contrairement à beaucoup de cas de choix d'un langage, ici les auteurs savent de quoi ils parlent et connaissent bien tous les langages impliqués.

---