Les moteurs de recherche de code source

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 21 avril 2009. Dernière mise à jour le 20 octobre 2011

https://www.bortzmeyer.org/codesearch.html

Pour le programmeur, la présence d'énormes quantités de codes source librement disponibles sur l'Internet est une bénédiction. Si la documentation n'est pas claire et qu'on cherche à trouver un exemple d'utilisation de la fonction C `gpgme_get_engine_info` ou la définition de la classe Java `BufferedReader`, on a tout sous la main.

Quels outils sont disponibles pour ces recherches ? Il y a les moteurs de recherche classique comme Yahoo ou Exalead. Mais ils ne sont pas spécialisés dans le code source : ils gèrent mal des chaînes de caractères comme `PERL_DL_NONLAZY` (Exalead croit que ce sont trois mots séparés), ils ne peuvent pas restreindre la recherche à un langage particulier, ils n'indexent pas les archives, par exemple tar, etc.

C'est pour cela qu'il existe des moteurs de recherche spécialisés dans le code source. J'en présente quelques uns, en maintenant une liste que je crois complète en <http://delicious.com/bortzmeyer/codesearch>.

La « méthodologie » que je suis pour leur évaluation est la suivante :

— Voir si les codes qui ne sont **pas** dans une archive ou un dépôt d'un VCS sont également indexés. Par exemple, ces moteurs voient-ils (en ligne sur `https://www.bortzmeyer.org/files/traceroute-nanog-xml.c`) ou (en ligne sur `https://www.bortzmeyer.org/files/gpgme-sign.c`) ?
— Voir si on peut trouver des choses qui apparaissent dans des relativement « petits » programmes, des logiciels libres pas trop connus (j'ai choisi echoping <http://echoping.sourceforge.net>, xmldiff <http://www.logilab.org/859/> et darcs <http://darcs.net/>).

Le premier moteur de recherche spécialisé dans le code source est Krugle <http://www.krugle.org/>. Parfois, il ne répond plus et toutes les adresses de courrier de contact échouent. Quand il marche, il ne connait pas les fichiers publiés sur mon blog, il ne connait pas echoping, ni xmldiff, ni darcs. Krugle a une interface simple, qui permet de chercher uniquement dans certaines parties du code (par exemple uniquement dans les commentaires ou, pour les fonctions, de ne chercher que les définitions ou que les appels, ce qui est très pratique).

Le géant de la recherche, Google, avait aussi son moteur de recherche spécialisé dans le code source <http://www.google.com/codesearch>, dont la fin a été récemment annoncée <http://googleblog.blogspot.com/2011/10/fall-sweep.html>. Il dispose d'un langage de requête perfectionné qui permet, par exemple, de spécifier le langage de programmation choisi (avec le qualificateur `lang:`) et de nombreux langages sont reconnus, même parmi les plus exotiques. S'il indexe d'innombrables archives, il ne connait pas les fichiers publiés sur mon blog (alors que mon blog lui-même est très bien indexé par Google).

Outre le qualificateur `lang:`, Google Code Search disposait du très pratique `function:` permet de trouver la définition d'une fonction. Mais il n'existe pas de moyen de trouver l'usage d'une fonction. Plus ennuyeux, si on cherche les usages de la définition `IPV6_UNICAST_HOPS`, on est noyés sous les définitions mais je n'ai pas trouvé de moyen simple de voir les appels de fonctions utilisant ce paramètre.

Google connait les programmes testés, echoping, xmldiff et darcs. Enfin, il dispose d'une API (je ne l'ai pas testée).

Continuous la tournée des moteurs de recherche. Si intéressant qu'il soit, et en logiciel libre, je n'ai pas cherché longtemps avec Gonzui <http://gonzui.sourceforge.net/> car il fonctionne uniquement en local, avec les archives qu'on lui fournit. Cela peut être pratique dans un environnement fermé mais, pour les logiciels librement accessibles, il indexera donc toujours moins que Google.

Enfin, place à Codase <http://www.codase.com/>, le plus récent. Parmi ses propriétés intéressantes, une possibilité de chercher, par le nom d'une fonction, son utilisation (option "*Method Name*") ou sa définition (option "*Method Definition*"). Encore mieux, l'option "*Smart Query*" tient compte de la syntaxe du langage pour proposer le bon résultat. C'est ainsi qu'on peut chercher des fonctions par signature, ce qui est pratique si deux fonctions portent le même nom.

Par contre, le choix des langages est bien plus restreint qu'avec Google : uniquement C, C++ et Java. Et le nombre de programmes indexé est également beaucoup plus faible : ni echoping, ni xmldiff, ni darcs ne sont archivés. C'est logique qu'il ne connaisse pas beaucoup de programmes. Lorsqu'on utilise son interface pour signaler un nouveau projet, on obtient « Bad Gateway The proxy server received an invalid response from an upstream server. Apache/2.0.54 (Fedora) Server at www.codase.com Port 80 ». Même message lorsqu'on essaie d'utiliser leur formulaire de "*feedback*".

Le dernier que j'ai rencontré est Koders <http://www.koders.com/>. Il ne connait pas non plus mes fichiers, ni les trois programmes d'exemple (malgré sa prétention à indexer « "*2,099,323,932 lines of open source code*" »). Il est le seul moteur testé ici qui ne reconnait pas la convention du _ comme partie d'un identificateur et il croit donc que `KEYRING_DIR` est composé de deux mots !

Bref, comme souvent avec les moteurs de recherche <https://www.bortzmeyer.org/moteur-recherche.html>, l'écart des capacités techniques entre Google et ses lointains suivants est énorme. La fin brutale de Google Code Search début 2012 devrait donc dégager un espace pour les concurrents. Un de ces jours, je re-testerai tous ceux de ma liste <http://delicious.com/bortzmeyer/codesearch>.

Mes remerciements à Sébastien Tanguy, Hervé Agnoux et à Kim-Minh Kaplan pour leurs suggestions.
