

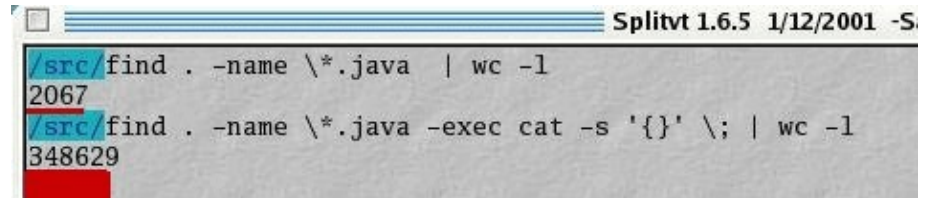


par Gerrit Renker
<gerrit.renker(at)gmx.de>

L'auteur:
Gerrit avait horreur des ordinateurs avant qu'il ne connaisse le C et Linux.

Traduit en Français par:
Laurent Richard
<kouran(at)linuxmail.org>

Naviguer avec snavigator



```
Splitvt 1.6.5 1/12/2001 -S
/src/ find . -name \*.java | wc -l
2067
/src/ find . -name \*.java -exec cat -s '{}' \; | wc -l
348629
```

Résumé:

Cet article va vous présenter snavigator, un puissant outil d'analyse de code, de référencement croisé et de re-engineering qui se trouve être vraiment indispensable pour gérer la complexité de la maintenance de logiciels d'une certaine taille ainsi que la gestion des paquets d'une manière efficace.

Motivation

Un vieux proverbe dit qu'il ne faut pas juger un livre à sa couverture. Un parallèle peut être fait avec le code ouvert. Néanmoins, on ne peut pas comparer du code ouvert à de la documentation libre. La lecture devient de plus en plus complexe à mesure que la taille des fichiers sources s'agrandissent. J'ai récemment dû programmer à l'aide d'un programme qui avait une demi page html en guise de documentation. En comparaison au plus de 348.000 lignes de code ouvert écrit en Java et divisé en pas moins de 2060 fichiers (voir figure). Quand on arrive à de telles chiffres, l'orientation électronique, le reverse engineering et les outils d'analyse deviennent indispensable comme le navigateur de code source Red Hat présenté dans cet article. Cet outil automatise beaucoup de tâches que l'on devrait réaliser en utilisant des (c)tags, grep, search ou replace. Mais il réalisera cela de manière plus précise et plus conviviale dans une interface graphique facile d'utilisation. Jetez un coup d'oeil aux captures d'écran pour vous en convaincre.

Installation sous Debian

Sous Debian, vous pouvez obtenir le tout via un simple

```
apt-get install sourcnav sourcnav-doc
```

Cela récupérera en même temps la documentation. Le navigateur de sources résidera dans `/usr/lib/sourcnav/`. Vous pouvez appeler le programme principal via `/usr/lib/sourcnav/bin/snavigator` (voir l'astuce à propos des liens symboliques ci-dessous). La documentation peut être trouvée dans `/usr/share/doc/sourcnav/html/`.

Installation à partir des sources

L'URL de la page d'accueil du navigateur de source est <http://sourcnav.sourceforge.net/>, les téléchargements se font en fait depuis [ici \(sourceforge.net/project/showfiles.php?group_id=51180\)](http://sourceforge.net/project/showfiles.php?group_id=51180). Obtenez le dernier tarball `sourcnav-xx.xx.tar.gz`. Lors du téléchargement, essayez de faire autre chose, en attendant, vu que les sources font environs 55 Mo. Cela a quand même un coté positif car après cela, le paquet n'a besoin de rien d'autre pour être utilisé. Même s'il fait un grand usage d'autres bibliothèques comme Tcl/Tk, Tix et Berkeley DB, les versions correctes de ces paquets sont également incluses. Afin d'éviter des conflits avec d'autres versions de Tcl/Tk, etc. sur votre système, cela me semble une bonne idée de faire l'installation dans un répertoire séparé, par ex. `/opt/sourcnav`. Les instructions suggèrent en plus d'utiliser un *répertoire de construction* séparé; cela se déroule comme suit. Après avoir décompressé, exécutez les commandes suivantes dans le répertoire contenant les sources décompressées :

```
mkdir snbuild; cd snbuild
../sourcnav-*/configure --prefix=/opt/sourcnav
make                               ## cela prend un peu de temps ...
make install                       ## vous devez devenir surper-utilisateur d'abord
```

L'option `--prefix` est là pour spécifier le répertoire d'installation. Lorsque le « configure » tourne, on a déjà une idée sur l'étendue des langages que `snavigator` peut gérer. Il est également possible d'ajouter des parsers supplémentaires pour les langages de votre choix ou d'en créer. Une fois que l'installation est terminée via `make install`, `snavigator` est prêt à fonctionner et il peut être lancé par `/opt/sourcnav/bin/snavigator`. Au lieu d'étendre votre `PATH` à ce nouveau répertoire, vous pouvez comme je vous le suggère plutôt de créer un *lien symbolique*, par ex. vers `/usr/local/bin`.

```
ln -s /opt/sourcnav/bin/snavigator /usr/local/bin
```

L'exécutable principal est un script shell qui a besoin de connaître son répertoire. Donc, il est un peu perdu si on l'appelle via un lien symbolique. Cela peut être résolu en changeant les lignes suivantes dans `/opt/sourcnav/bin/snavigator`; au lieu de

```
snbindir=`dirname $0`
```

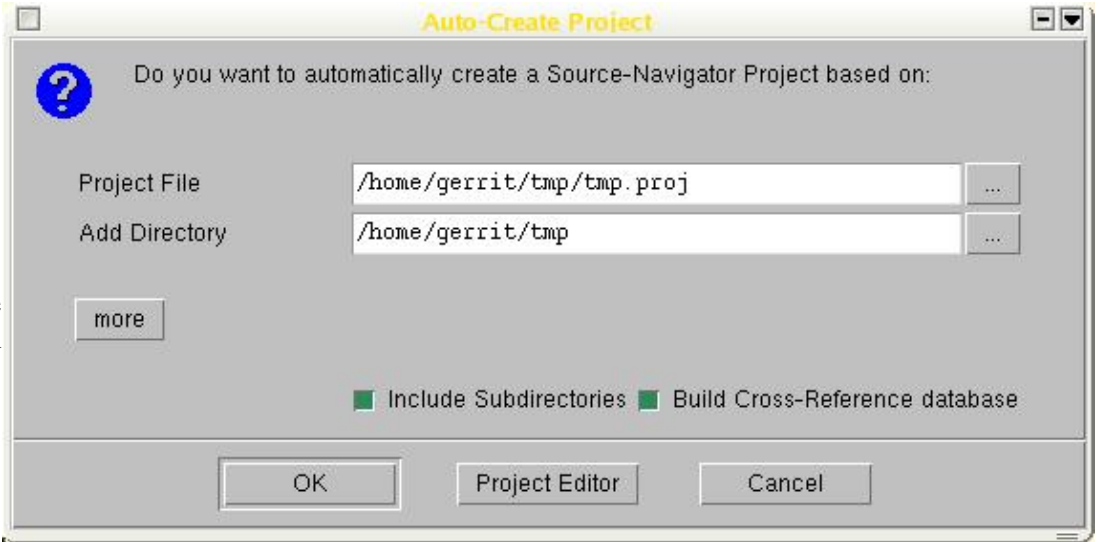
utilisez

```
prog=`readlink -f $0`
snbindir=`dirname $prog`
```

L'option `-f` de **readlink(1)** crée une représentation du nom de chemin canonique. Cela signifie que cela va fonctionner même si on accède au fichier via une longue suite de liens symboliques consécutifs.

Utiliser `snavigator`

La première fois que snavigator est lancé, il va demander à connaître les répertoires contenant les fichiers sources comme le montre la capture d'écran suivante. Les langages qui sont inclus mais qui ne sont pas limitatifs sont Java, C, C++,



Tcl, Fortran, COBOL et l'assembleur. Une fois les détails sur la localisation du code source donné, il va construire, indépendamment, une base de donnée du projet en incluant l'information de référencement, les hiérarchies de classes, les inter-dépendances entre les fichier et bien plus encore. La construction prend un certain temps dépendant bien sûr de la taille de votre projet. Une fois réalisé, la base de donnée peut être consultée et des informations supplémentaires peuvent être demandées à propos du code. Je ne fais que mettre en avant quelques possibilités du programme pour vous donner une idée globale. Un *guide de l'utilisateur* illustré de même qu'un manuel de référence sont inclus dans le répertoire `html` de l'installation.

Gestion du projet

Une partie du programme est un *éditeur* avec colorisation de la syntaxe pouvant être aussi utilisé pour des fichiers à imprimer. La capture d'écran suivante montre la fenêtre principale de l'éditeur. Elle ressemble vraiment à un environnement de développement et fournit des outils comme une usine à déboguer, des commandes de construction de projet, un contrôle de version et ainsi de suite.

The screenshot shows the Source-Navigator IDE window titled "Source-Navigator [tmp] Main.java". The menu bar includes File, Edit, Search, Tools, History, Windows, and Help. The toolbar contains navigation arrows, a dropdown menu showing "emit_parser(md) Main", and icons for search, refresh, and other actions. Below the toolbar are tabs for Edit, Hierarchy, Class, Xref, Include, Retriever, and Grep. The main editor area displays the following Java code:

```
*/
protected static void emit_summary(boolean output_produced)
{
    final_time = System.currentTimeMillis();

    if (no_summary) return;

    System.err.println("----- " + version.title_str +
        " Parser Generation Summary -----");

    /* error and warning count */
    System.err.println(" " + lexer.error_count + " error" +
        plural(lexer.error_count) + " and " + lexer.warning_count +
        " warning" + plural(lexer.warning_count));

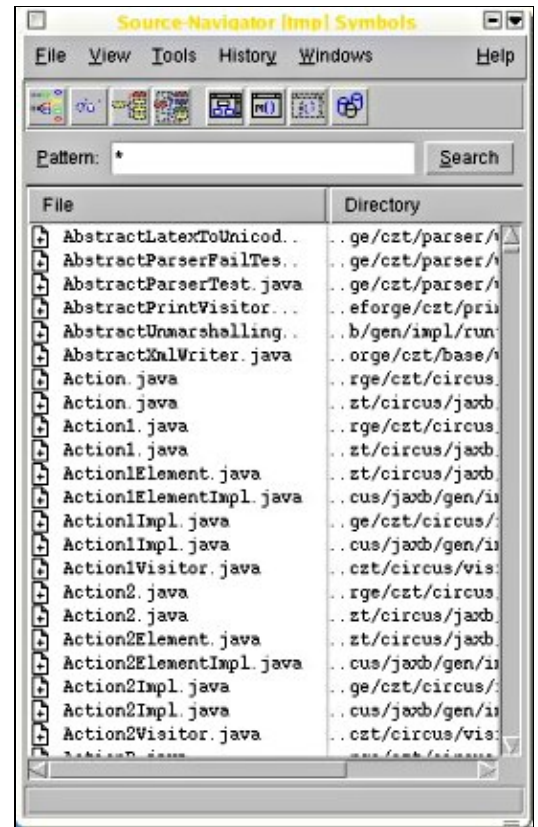
    /* basic stats */
    System.err.print(" " + terminal.number() + " terminal" +
        plural(terminal.number()) + ", ");
    System.err.print(non_terminal.number() + " non-terminal" +
        plural(non_terminal.number()) + ", and ");
    System.err.println(production.number() + " production" +
        plural(production.number()) + " declared, ");
    System.err.println(" producing " + lalr_state.number() +
        " unique parse states.");

    /* unused symbols */
    System.err.println(" " + emit.unused_term + " terminal" +
        plural(emit.unused_term) + " declared but not used.");
    System.err.println(" " + emit.unused_non_term + " non-terminal" +
        plural(emit.unused_term) + " declared but not used.");
}
```

At the bottom of the editor, there are buttons for "Reuse" and "Context".

En particulier, les grandes flèches vertes sur le menu fonctionnent de la même manière qu'un navigateur web. L'éditeur de projet permet de contrôler l'information de la base de données. Par ex : si un fichier vient d'être mis-à-jour, l'ajout ou la suppression de fichiers de la liste ainsi que d'autres tâches de gestion. Tous les fichiers sont traités comme un grand projet. Dès lors, si des modifications sont effectuées, vous pouvez mettre à jour l'information de la base de données via *Refresh Project* ou *Reparse Project*.

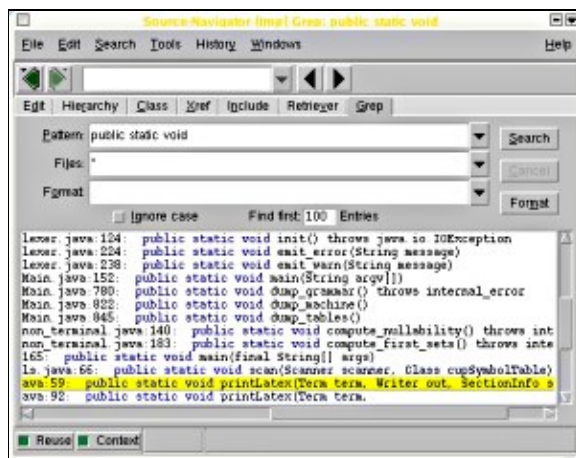
Lorsque la fenêtre de l'éditeur met en évidence quelque chose comme le nom d'une fonction comme celle qui est représentée ici en jaune, vous pouvez faire un clic-droit avec votre souris afin de pouvoir choisir de faire une déclaration de ce qui a été mis en évidence (comme un fichier d'en-tête), d'implémenter le symbole mis en évidence (ex : un fichier .cpp) ainsi que quelques autres options utiles.



Navigateur de symboles

Voici la première fenêtre qui s'ouvre après avoir constitué la base de données du projet. Généralement, elle contient les noms de fichiers mais peut également afficher les méthodes de classe, les symboles de fonctions et ainsi de suite. Lorsque on clique sur un nom de fichier, l'éditeur sera ouvert avec ce fichier.

La fenêtre grep

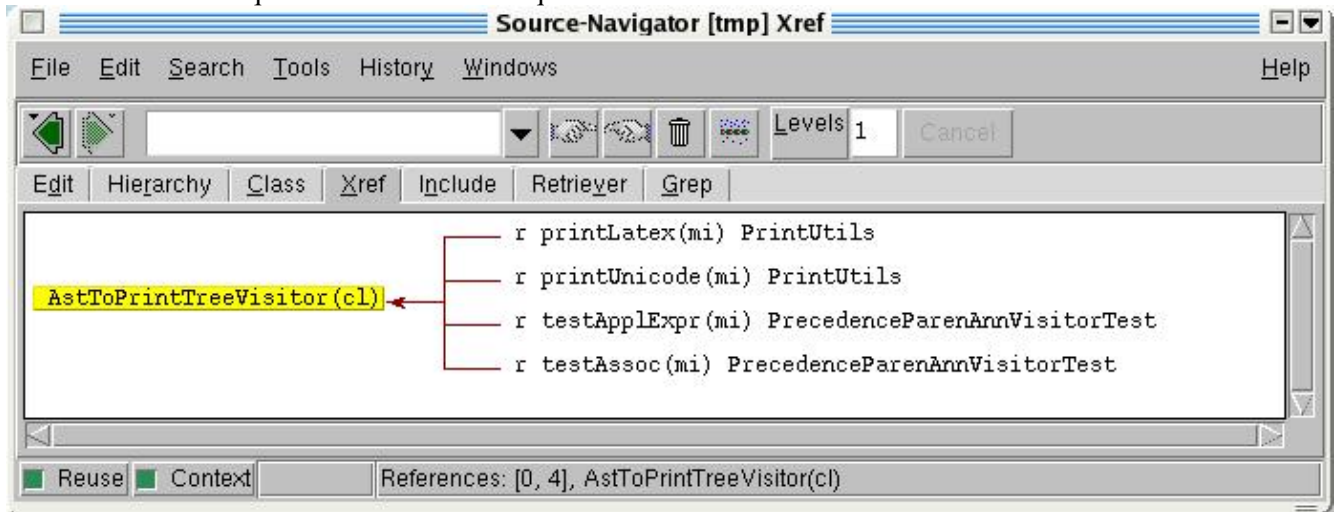


Cela ouvre, comme le dit si bien le titre, une interface graphique conviviale pour « grepper » dans l'ensemble des fichiers de code source concernés. Les entrées correspondantes seront mises en évidence et cliquables. Le code source peut dès lors faire l'objet d'une navigation comme si vous étiez sur une page web. Comme le montre la capture d'écran, le fichier ainsi que sa localisation respective peuvent être sélectionnés et par simple clic, vous entrez dans l'éditeur qui sera positionné à l'endroit exact dans le code. (Cette recherche sur la base d'un terme a donné des résultats positifs dans beaucoup de fichiers Java :)

Fenêtre Xref

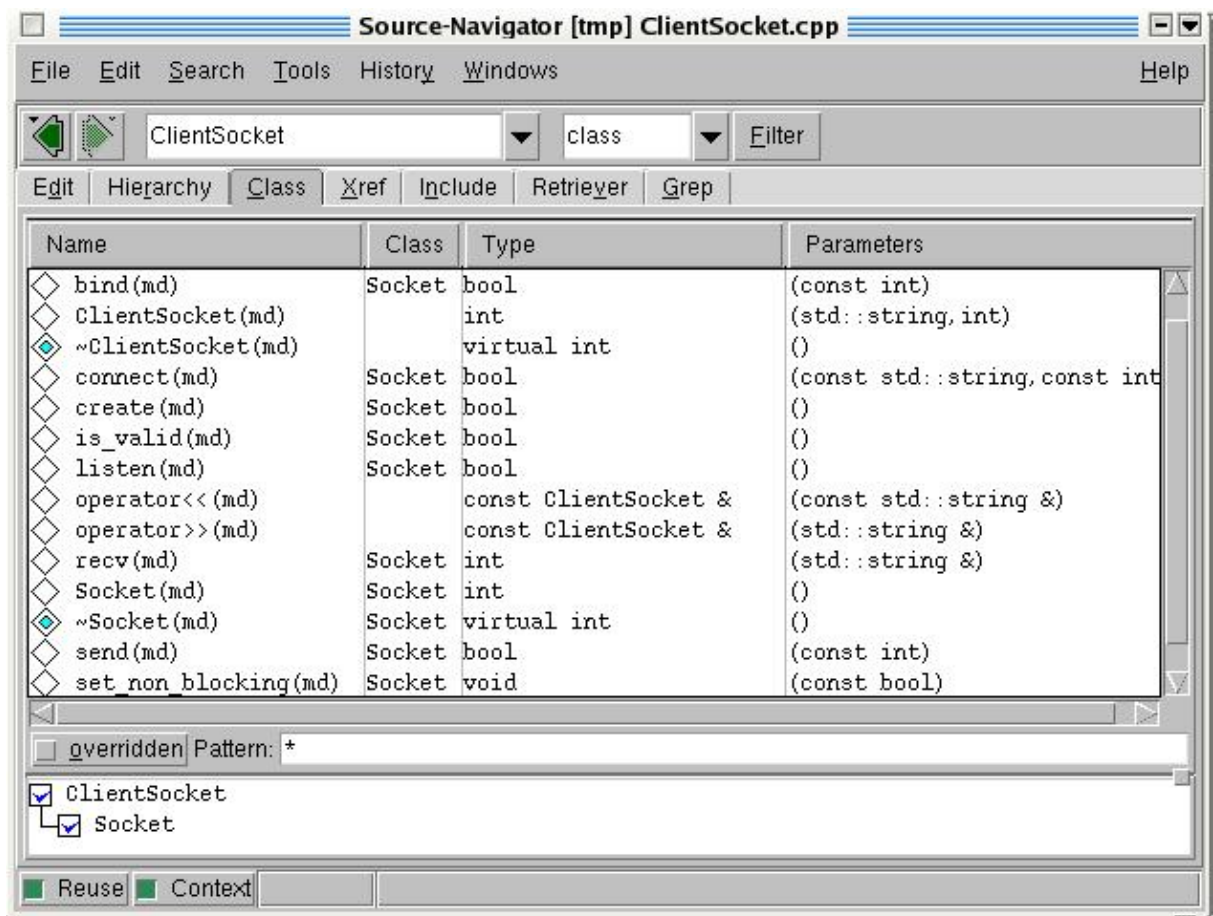
Ici, nous avons une liste de références croisées de tous les symboles. En particulier, on peut constater quelles méthodes lisent (r), écrivent (w), ... et sur quelles données et voir les relations entre les symboles représentées

de manière hiérarchique. Les entrées sont cliquables.



Fenêtre Class

Cette interface regroupe toutes les informations utiles que l'on désire connaître sur les classes dans un langage orienté objet. En particulier, les super- et les sous-classes sont affichées ainsi que les noms des attributs et des méthodes accompagnées de leurs paramètres. Pour changer, la fenêtre en dessous montre une classe ClientSocket en C++ qui hérite de Socket et qui a quelques méthodes. Encore une fois, en cliquant sur une des entrées, vous pouvez ouvrir une fenêtre d'édition positionnée à l'endroit adéquat.



Autres alternatives

`cscope` est un navigateur interactif, basé sur la console de code source C (il peut faire également du C++). Il a certaines fonctionnalités de `snavigator`. Une capture d'écran se trouve [ici](#). En fait, il est beaucoup plus vieux et a été utilisé dans pas mal de très gros projets. Sa page d'accueil est <http://cscope.sourceforge.net/>. Mais vous n'avez pas besoin de vous y rendre car il est incorporé directement dans **vim** et peut être utilisé de la même manière que `(g)vim` en combinaison avec les `tags`. Tapez simplement

```
:help cscope
```

dans votre session vim pour vérifier les options disponibles. Il existe certains dérivés de `cscope`. [Freescope](#) est un clone de `cscope` qui a quelques fonctionnalités supplémentaires telles que la complétion de symboles. Il y a maintenant également une interface graphique sous KDE pour `cscope` qui s'appelle `kscope` et qui peut être trouvée sur <http://kscope.sourceforge.net/>.

Conclusions

Pour quiconque impliqué au moins en partie dans du re-engineering ou de l'intégration de code source, `snavigator` est un outil puissant et très utile. J'avais avant une vieille application Qt qui ne fonctionne malheureusement plus avec la version actuelle de la bibliothèque Qt. En regardant les messages d'erreurs et en naviguant un peu avec `snavigator`, j'ai trouvé rapidement que seule la liste de paramètre d'une des fonctions devait être changées. En utilisant la fonctionnalité de localisation par clic, il fut possible de rendre le logiciel complet à jour en à peine quelques minutes.

Site Web maintenu par l'équipe d'édition LinuxFocus © Gerrit Renker "some rights reserved" see linuxfocus.org/license/ http://www.LinuxFocus.org	Translation information: en --> -- : Gerrit Renker <gerrit.renker(at)gmx.de> en --> fr: Laurent Richard <kouran(at)linuxmail.org>
--	---