



A la découverte de gedit

Le 6 septembre 2008 à 13:44.

Après avoir migré de KDE à Gnome [1] j'ai longtemps recherché un équivalent à Quanta ou plus généralement à Kate. Après avoir testé pas mal d'environnement plus ou moins correctement intégrés au bureau, j'ai fini par me rabattre sur Bluefish, sans grande conviction, je dois bien l'avouer. Le seul outil que j'avais écarté de mon étude était ce pauvre gedit que je prenais pour un équivalent du simplissime kedit, le "notepad" de kde. Grosse erreur...

Pour les développements "durs", j'utilise et continuerais à utiliser eclipse [2]. Cependant lorsqu'il s'agit de modifier de hacker rapidement de petits projets, coder un bout de script, ou encore d'éditer un billet en HTML, cette plate-forme devient pour le moins contraignante.

A l'époque KDE, j'utilisais le couple Kate/Quantum en préférant souvent Kate pour sa légèreté, son terminal intégré, son navigateur de symboles. Quantum était plus pratiques les pages PHP surtout par son auto-complétion. Gnome ne manque en soit pas de plate-forme de développement mais soit elles sont mal intégrés au bureau (Code::Blocks), soit elle manque des fonctionnalités les plus basiques (à peur près tous), soit elles rament (Typiquement Geany), soit elles plantent constamment (j'ai nommé Screem). Avec Bluefish, ce fût pour ainsi dire le choix du "moins pire". Pas brillant mais stable, pas très évolué mais rapide.

Et ce n'est que tout dernièrement, que je suis tombé par hasard sur le menu "greffons" des préférences de GEdit. J'ai commencé un peu surpris à en activer quelques uns, puis je me suis rendu compte que les dépôts en contenait encore beaucoup d'autres... Et après quelques temps, parti de ce que je croyais être un simple "notepad", je me suis retrouvé avec un éditeur aussi puissant que Kate et Quantum réunis tout en étant d'une rapidité et d'une stabilité que Quanta et Bluefish n'ont jamais réussis à atteindre.

Toutes les fonctions d'un éditeur pour développeur y sont : indentation automatique, numérotation des lignes, marge droite, bascule espaces/tabulation, sur-lignage des accolades fermantes.

Un très bon support de la coloration syntaxique.

La possibilité de changer de thèmes de couleurs, j'aime beaucoup celui appelé *oblivion*, très reposant.

Il ne souffre pas du bug du retour à la ligne particulièrement insupportable sur Bluefish. En effet une longue ligne (typiquement un paragraphe HTML) avec la césure automatique activée, s'étend logiquement sur plusieurs lignes à l'écran. Le souci sur Bluefish est que toute pression sur la touche `end` entraîne le curseur à la fin de la ligne réelle, soit à la fin du paragraphe...

Une véritable auto-complétion rapide et efficace. En fonction du type de code édité (css, php, java, c, etc.), un simple `CTRL+ESPACE` donne accès à une liste de commandes disponibles dans le contexte. Et ce, sans mouliner pendant 3 secondes comme Quanta.

Enfin le support ces fameux greffons, et il y en a des dizaines de disponibles....

Les greffons

En standard gedit est fourni avec un certain nombre de ces extensions. Dans ma distribution (Mandriva), un paquet supplémentaire en rajoute encore plus. Enfin, vous pouvez trouver ce qui vous manque encore ici [3]. Je ne présente ici que ceux qui m'ont permis de retrouver l'aisance que j'avais perdu en quittant KDE.

D'un point de vue général, les greffons s'active dans la boîte des préférences, onglet *greffons*. Là en sélectionnant une extension, vous pouvez soit l'activer/désactiver, soit, si cela existe, afficher le panneau de configuration.

Correcteur orthographique

Il est fourni en standard et permet la correction syntaxique temps-réel avec colorisation dans le texte et menu contextuel de suggestions. Un grand plus par rapport à Bluefish qui ramène sur cet aspect gedit.

Si comme moi vous avez des problèmes avec les apostrophes, il s'agit à l'évidence d'un problème avec la librairie *enchant* dans sa dernière version. J'ai recompilé et installé la 1.4.3 et tout est revenu à la normale...

Volet de navigation dans les fichiers

Ajoute un panneau latéral permettant de naviguer dans les fichiers. Rien de bien sorcier mais c'est juste indispensable. Je regrette juste qu'il n'y ait pas un paramétrage pour synchroniser le dossier courant avec celui du document en cours d'édition.

Outils externes

Ce greffon ajoute la gestion des outils externes permettant de lancer des commandes avec possibilité de modifier de manière intelligente le contenu en cours d'édition. Pratique pour lancer des formateurs comme *tidy* ou tester une page avec navigateurs externes.

Enregistreur de sessions

Permet de faire comme avec FireFox, des sessions des onglets ouverts.

Sélecteur de couleurs

Une outil d'une bêtise ahurissante qui me manquait sur plein d'éditeur offrant la simple possibilité d'insérer une couleur au format HTML via un sélecteur visuel. Le nombre de fois où j'ai du lancer Gimp simplement pour cela...

Volet Terminal

Voilà bien quelque chose qui me manquait de Kate, un simple volet "console" en bas de l'écran d'édition pour lancer des commandes ou tester le script en cours de rédaction.

Extraits de code

Ce greffon à lui tout seul vaut le détour car c'est en réalité lui qui gère l'auto-complétion. Il vous permet par langage d'ajouter une série de fragments de code permettant d'automatiser son écriture. La syntaxe est aussi puissante qu'avec Eclipse et permet la rédaction de macros complexes. Par exemple, j'ai un tag perso sous Drupal qui me permet de créer un lien vers une page externe. Il s'écrit `<external href="http:// site-externe">le lien</external>`. Sous Bluefish je n'ai pas trouvé de moyen d'automatiser cette syntaxe un peu particulière. Avec les extraits de code j'ai pu saisir ceci :

```
|| <external href="{1:http://somesite.com/}">${2:$GEDIT_SELECTED_TEXT}</external>
```

Le premier paramètre, je lui donne une valeur par défaut, et le second se base sur le texte sélectionné au moment d'insérer mon fragment. A cela je peux ajouter soit un mot clef qui sert de déclencheur d'insertion en tapant `déclencheur` suivi de la touche Tab, ou encore le début d'un déclencheur, suivi de CTRL-SPACE pour avoir la liste. Il est aussi possible d'ajouter en plus un raccourci clavier.

Navigateur de symboles

Celui là, il faut se l'installer à la mano mais c'est très bien expliqué [sur le site de son auteur](#) [4]. Il permet d'ajouter un volet qui affiche la liste des symboles détectés dans le document en cours d'édition. Indispensable pour une navigation rapide dans le code.

Fonctions avancées

Sauvegarde distante

Il est possible via [GIO](#) [5] d'accéder à des systèmes de fichiers distants (webdav, ftp, sftp, smb, etc.). Cela peut se faire soit en utilisant `Fichier/Ouvrir un emplacement`, avec Nautilus en naviguant sur un serveur et en faisant un `glisser-déposer` sur gedit, ou enfin par le volet gauche de navigation (si le greffon est activé) en passant par les [signet](#) [6] définis dans Nautilus.

Dans tous les cas, le fichier distant devrait être lu et édité sans problème. Le souci est au niveau de la sauvegarde. En effet, sauvegarder un fichier distant marche très bien avec les protocoles `sftp` ou encore `smb`, mais pas du tout avec `ftp`. Ce n'est pas un bug, c'est une option...

Pour autoriser la sauvegarde des fichiers via ftp, il faut d'abord aller dans le menu `Gnome Système/Préférences/Avancées/Éditeur de configuration`. Là il faut rechercher le chemin `/apps/gedit-2/preferences/editor/save`. Là vous trouvez une variable `writable_vfs_schemes` contenant une liste de protocoles à laquelle il manque... `ftp`. Double-cliquez dessus et ajoutez `ftp`. Validez et relancez gedit. Vous devriez maintenant pouvoir sauvegarder vos fichiers sur un serveur FTP sans problèmes.

Sauvegarde automatique

Ne fermez pas l'éditeur de configuration car il y a une autre option intéressante dans ce dossier `save`. Il s'agit de la sauvegarde automatique que vous pouvez activer en cochant `auto_save` et éventuellement en spécifiant la valeur `auto_save_interval` pour définir un temps de sauvegarde en minutes. Il est aussi possible ici de ne pas créer de copies de sauvegarde (ces fichiers terminés par des `~`) en décochant `create_backup_copy`.

Encodage des caractères

Un éditeur de texte est souvent pratique pour gérer les problèmes d'encodage. Avec gedit, il est possible de forcer la lecture dans un format précis dans la boîte de dialogue d'ouverture d'un fichier par la liste déroulante `Encodage des caractères` (Par défaut gedit détecte automatiquement l'encodage).

Mais plus intéressant, il est aussi possible lors de la sauvegarde de spécifier dans la boîte `enregistrer` un encodage des caractères différent de celui du fichier d'origine et ce toujours par la liste `Encodage des caractères`.

Maintenant cette liste est étrangement réduite. Si par exemple vous avez besoin de sauver un texte UTF-8 en ISO-8859-3, il va vous manquer quelque chose. Pour arranger cela, utilisez encore l'éditeur de configuration mais allez cette fois dans le dossier `/apps/gedit-2/preferences/encodings`. Vous avez alors deux listes, la première `auto_detected` indique les encodages que gedit va chercher à trouver automatiquement en lecture d'un fichier, la seconde, `shown_in_menu` indique les encodages qui apparaîtront en plus d'UTF-8 dans la liste déroulante des boîtes de dialogue.

Le bug du MCP

Droop [7] a attiré mon attention sur le problème de non conformité du comportement click central avec gedit. Traditionnellement, une sélection sous X11 stocke le texte dans un tampon accessible ensuite par un simple click-central. Or avec gedit, cela ne fonction *que* si le texte reste sélectionné. Ce problème est un bug référencé [8] qui semble déclencher la passion entre les "geek" et les "adorateurs de la grand-mère [9]".

En attendant que tout ce beau monde trouve une solution et que cette solution se retrouve, comme d'hab, dans `gconf-edit`, le contournement simple est d'utiliser l'applette Glipper en sélectionnant dans les préférences `Select - Mark/middle mouse button`.

Conclusion

En pensant que gedit était un simple équivalent à kedit, j'étais bien loin du compte. Cet outil est en fait ultra-représentatif de la raison pour laquelle je suis passé à Gnome : simple, rapide, stable, facile à compiler soit même, extensible facilement. Et aujourd'hui, avec l'ensemble des extensions que j'ai rajouté, je dispose d'un éditeur qui, au regard de mon usage évidemment, est totalement équivalent à Kate ou Quanta en était tellement plus léger.

*<http://artisan.karma-lab.net/node/1619>
(C) artisan numerique - CC BY-SA*

Liens:

- [1] <http://artisan.karma-lab.net/node/1323>
- [2] <http://artisan.karma-lab.net/eclipse.org>
- [3] <http://live.gnome.org/Gedit/Plugins>
- [4] <http://www.micahcarrick.com/11-14-2007/gedit-symbol-browser-plugin.html>
- [5] <http://artisan.karma-lab.net/node/1323%23GVFS>
- [6] <http://artisan.karma-lab.net/node/1323%23signets>
- [7] <http://artisan.karma-lab.net/node/1619%232956>
- [8] http://bugzilla.gnome.org/show_bug.cgi?id=333514
- [9] <http://artisan.karma-lab.net/node/1125>