Libroscope, plus que des logiciels libres, des hommes libres !

-- Réflexions --

Réflexions

La valse des versions, la fatigue de l'utilisateur

Raphaël RousseauJulien Tayon

mardi 15 juillet 2003

Interrogé sur son opinion concernant le logiciel libre, un administrateur système répondait : « *ils n'ont donc aucune pitié des admistrateurs systèmes : ils sortent de nouvelles versions tout le temps, et nous avons du mal à suivre* »

Cette remarque peut s'appliquer à tous les types de logiciels en fait : nous nous retrouvons à subir une valse des versions et des déclinaisons de produits qui nous fait tourner la tête. Rien que sous Linux, nous ne rien que de :
- au moins une dizaine de gestionnaires de fenêtres, et certains comme blackbox ont donné lieu à pas moins de 5 variantes (dont Fluxbox, Waimea, Hackedbox, Openbox) ;
- un choix dans les interprêteurs de commande (shells) : bash, ksh, zsh
- plusieurs environnements de bureaux, principalement GNOME et KDE ;
- une foule de traitements de texte : OpenOffice, KOffice, AbiWord ;
- une avalanche de langages de programmation : Perl, Python, PHP, Ruby, Objective C, Lisp...

Cette tendance à la multiplication des logiciels et des versions n'est pas sans rappeler ce qui se passe dans le domaine de la vente des ordinateurs où rien que dans le domaine des PCs nous pouvons dénombrer une grande diversité de processeurs, de bus, de mémoire sans que nous soyons assurés que cette diversité nous profite *vraiment*.

Dans le logiciel non libre, cela correspond à l'ajout de fonctionnalités sensées vous convaincre de renouveler votre parc, mais dans le logiciel libre, qu'en est-il ?

Corrections de bugs et finalisation de fonctionnalités

L'exemple des numéros de version du noyau Linux.

Il est numéroté avec trois numéros : majeur.mineur.infra -> ex 2.6.1 ou 1.2.8
- le premier correspond à la version *majeure*. Un changement de ce numéro correspond à une véritable évolution dans l'architecture du noyau, son organisation interne et ses principaux mécanismes ;
- le second numéro correspond à la version *mineure*. Elle apporte de nouvelles fonctionnalités, parfois importantes, mais ne remet pas en cause la structure du noyau.

Les versions mineures ont une signification particulière selon que le numéro est pair ou impair.
- Un **numéro impair** signifie que cette version n'est pas encore testée/aboutie ; on dit qu'elle est *en développement*. Elle est destinée à ceux qui sont courageux et/ou compétents et/ou exigents et qui veulent tester le noyau. On parle parfois de version *instable*.
- Un **numéro pair** désigne une version de production, jugée assez testée et éprouvée pour être mise sur des machines sensibles. Cela ne signifie pas qu'aucun bug ne sera découvert, puisque c'est l'objet du numéro suivant, mais les craintes de problèmes graves sont généralement éloignées. On parle de version *stable*.
- le troisième numéro est destiné aux optimisations diverses et aux corrections des bogues constatés dans les versions antérieures.

La multiplication des versions mineures n'est pas un problème, cela signifie juste que les développeurs prennent soin de laisser aux utilisateurs une version à jour au niveau de la sécurité et de la correction de bugs. En matière de sécurité, il est important de fournir une version à jour au plus tôt. **La multiplication des mises à jour de sécurité n'est pas un mal en soi, c'est un mal nécessaire**. Il est parfois juste énervant de voir des erreurs classiques de programmation (*buffer overflow* par exemple) revenir régulièrement alors que les techniques pour les prévenir sont

enseignées aux débutants en informatique (la solution est le "blindage des données").

La multiplication des numéros de version majeures, elle, peut poser un problème si les auteurs remettent en cause leurs fonctionnalités : les logiciels libres sont interdépendants.

Quoi qu'il advienne, la multiplication des mises à jour ne fait que souligner un fait ; l'ordinateur qui marche à la sortie de son carton, sans qu'on y touche jamais est un mythe : il faut, quelque soit le parc d'ordinateur, quelque soit le système d'exploitation **remettre à jour ses logiciels** [1] : si vous avez Internet Explorer 5.0 sur un ordinateur professionnel, ou si vous avez Netscape 4.0, alors il y a un risque : ces deux logiciels ont des trous de sécurité. Avec Windows, vous devez appliquer des *Service Packs* qui peuvent rendre votre ordinateur inutilisable. Avec les système Libres (*BSD, GNU/Linux) vous avez la possibilité de les remettre à jour aisément sans remettre en cause l'intégration de vos logiciels.

Les forks et processus darwiniens de sélection

La coopétition de projet

Des problématiques similaires dans des domaines différents apparaissent parfois simultanément. Il arrive souvent que deux projets similaires émergent. C'est très bien, cela laisse le choix. Bien fol qui s'en plaindrait, il y a plus de chance que l'un aboutisse, et que l'autre apprennent des erreurs du concurrent.

La bêtise/fainéantise humaine

Par besoin de reconnaissance, par divergence de points de vue ou par cruauté bestiale, certains ré-implémentent des projets, au lieu de contribuer à ceux existant. Ces derniers développeurs vont lancer un projet *concurrent*, soit en partant de rien, soit en bifurquant un projet existant.

Le cas du fork (bifucation à partir d'un projet existant) par manque de fonctionnalités est légitime.

Le cas du fork par manque de recherche est particulièrement pénible. Il est d'autant plus pénible que la règle est qu'avant de vous lancer dans un projet vérifier que : un projet similaire n'existe pas ;
votre projet n'est pas une sous-problématique d'un projet existant (ex : faire une gallerie photo de portraits sur un site web, est une sous-problématique d'une gallerie de photo). Les architectures modulaires favorisent l'ajout de fonctionnalités.

Le cas des entreprises qui veulent faire une opération de communication en prenant la tête d'un projet Logiciel Libre est totalement abscons : qui va vouloir coopérer avec eux s'ils ne veulent pas accepter les règles du jeu ? Au mieux, ils font un *logiciel propriétaire sous GPL*, au pire ils font un prototype éternel.

Pour un logiciel libre, tout un chacun peut *"forker"* : il suffit de prendre le code source du logiciel, et de la modifier sans communiquer les corrections aux développeurs originaux, et sans chercher un consensus dans les orientations à prendre.

Cette opportunité fait peur à bien des adoptants potentiels, qui ont peur de se lancer dans l'aventure en installant GNU/Linux sur leurs machines, car ils craignent que n'importe qui forme des versions concurrentes du noyau et que la communauté ne s'émiette. À ce sujet, s'il fallait en rassurer certains, lisez Fear of Forking. Cet essai explique très bien pourquoi un projet, une fois qu'il a atteint un certain nombre d'adoptants et de contributeurs (ce qui est parfois lié directement), a peu de chances

de subir une bifurcation, car cela mettrait les utilisateurs et les contributeurs face à un choix inconfortable : miser sur l'un ou sur l'autre. Les ressources sont potentiellement infinies car tout le monde peut contribuer, certes, mais vous retrouver face à une telle alternative sans que cela se justifie pleinement, voilà un mécontentement auquelle serait confronté le *mutin*.

La diversité, une nécessité ?

L'exemple des langages de programmation

Il est énervant de voir autant de langages. Dans un monde idéal, tout le monde parlerait une novlangue dont le vocabulaire serait réduit afin que tous se comprennent [2]. Cependant, dans la vraie vie, différents besoins donnent différents langages :
- pour faire des programmes bêtes qui doivent aller très vite, ou proches du coeur de l'ordinateur, le C est le langage de prédilection ;
- pour combiner des petits programmes rapides écrit sous Unix rien ne vaut un langage de scripts comme *bash*, *tcsh* ou *ksh* ;
- pour faire en quelques lignes des choses compliquées (au coût des ressources nécessaires et de la rapidité d'exécution) indépendante des plates-formes, rien ne vaut le Perl, le Ruby ou le Python ;
- pour faire des appplications web, rien ne vaut le PHP, le Perl, le Python.

En fonction des problématiques, chaque langage a ses points forts ou faibles, et je dois admettre que les langues informatiques sont comme des langues naturelles : avec certaines, on se sent plus inspiré que par d'autre.

Le choix est trop agréable pour vouloir le renier. Avec le choix viennent des responsabilités : si le choix dans les langages est agréable, un projet utilisant moult langages est une plaie à comprendre et à maintenir. Il est de la responsabilité des utilisateurs de choisir de manière éclairée. Contrairement aux éditeurs de logiciels propriétaires, nous avons la culture du choix ; si elle vous agace, renoncez au logiciel libre.

"They that can give up essential liberty to obtain a little temporary safety deserve neither liberty nor safety." *-- Benjamin Franklin, 1759*

Conclusion

La responsabilité des utilisateurs est importante ; ils sont un élément moteur du libre en tant qu'écosystème. On ne peut concevoir d'utiliser le logiciel libre sans faire une veille technologique. Souvent, la veille se résume à demander à ses proches quels logiciels ils utilisent. Les choix que vous faites influencent les choix des autres. Si vous n'aimez pas la valse des versions, évitez vous-mêmes de valser, et intéressez-vous à l'historique du projet. En utilisant un projet, vous le faites vivre. Faites vivre ceux qui ont un intérêt, ignorez les autres, et vous participerez à ralentir a valse des versions.

http://www.techcentralstation.com/1051/techwrapper.jsp ?PID=1051-250&CID=1051-061902B

[1] Eh oui, les administrateurs systèmes compétents sont nécessaires !

[2] *novlangue* : référence à 1984 de George Orwell .