De Linux et de l'opportunité d'une synchronisation des distributions

Voici une traduction[1] un peu *technique* mais qui illustre bien la problématique de la *démocratisation* de GNU/Linux.

Elle fait suite à la proposition récente de Mark Shuttleworth (Monsieur Ubuntu) de synchroniser les cycles et donc les sorties des principales distributions Linux (outre Ubuntu il cite celles de Red Hat, Novell et Debian ainsi que le noyau, GNOME/KDE, X et OpenOffice.org). Histoire que tout ce petit monde avance groupés, ce qui d'après lui simplifierait la vie de tout le monde à commencer par celle des utilisateurs.

Mais Ryan Paul du site *Ars Technica* n'est visiblement pas tout à fait de cet avis. Au *ce serait bien si…* de Mark Shuttleworth il répond avec des arguments précis qui évoquent souvent le quotidien collaboratif d'un développeur de logiciels libres (en particulier tout ce qui touche à la gestion de versions). Et lorsque l'on est, comme moi, utilisateur mais non développeur de logiciels libres, c'est culturellement fort enrichissant.

On notera qu'à la suite de sa proposition et du vif débat suscité, Shuttleworth a précisé voire nuancé son propos quelques jours plus tard sur son blog.

Pourquoi Linux n'est pas encore prêt pour des cycles de parution synchronisés

[Why Linux isn't yet ready for synchronized release cycles]

Ryan Paul – 21 mai 2008 – ars technica

Le fondateur d'Ubuntu, Mark Shuttleworth a répété son appel aux développeurs des principaux logiciels libres et des distributions Linux pour une synchronisation des développements et des cycles de publication. Il avance que l'adhésion fidèle et universelle à un modèle de parution régulier encouragerait la collaboration entre les projets, assurerait aux utilisateurs l'accès aux dernières nouveautés des applications populaires et ferait de la plateforme Linux une cible plus stable et prévisible pour les vendeurs de logiciels.

Shuttleworth souhaite organiser les principales sorties en trois vagues distinctes, chacune formant un ensemble cohérent. La première vague concernerait les composants fondamentaux comme le noyau Linux, le compilateur GCC, les boîtes à outils graphiques comme GTK+ et les plateformes de développement comme Python et Java. La deuxième vague apporterait les environnements graphiques et les applications tandis que la troisième vague serait composée des distributions.

Bien qu'un cycle de sortie unifié rendrait plus aisée la

création d’une distribution Linux, ce concept apporte d’importantes difficultés et n’est que peu gratifiant pour les développeurs de logiciels. Pour parvenir à une synchronisation à grande échelle comme Shuttleworth le souhaite, certains logiciels libres devraient radicalement changer leur modèle de développement actuel et adopter une nouvelle approche qui ne sera pas viable pour nombreux d’entre eux.

Comprendre les cycles de sorties réguliers

Un cycle de sorties régulier nécessite de sortir de nouvelles versions à une fréquence donnée. Le processus de développement pour les projets qui emploient ce modèle implique en général une planification des fonctionnalités prévues et ensuite une implémentation maximale jusqu’à ce que le projet gèle le code lorsque l’échéance approche. À partir de ce moment là, les fonctionnalités qui ne sont pas terminées sont reportées. On se concentre alors sur la correction des bogues et sur l’assurance qualité jusqu’à la date butoir, quand le logiciel est officiellement sorti.

Ce modèle fonctionne bien pour de nombreux projets, en particulier pour l’environnement GNOME. Mais, une conséquence de ce modèle est que les développeurs doivent travailler par incrémentation et il décourage les modifications de grande ampleur, celles qui nécessiteraient plus de temps que n’en offre le cycle. Parfois cet intervalle n’est simplement pas suffisant pour ajouter au code principal et tester des changements d’architecture importants qui sont incubés en parallèle en dehors de l’arbre principal du code.

Quand cela se produit, les développeurs doivent se demander si les avantages de la nouvelle fonctionnalité compensent les effets néfastes de la régression (comme avec l’adoption de GVFS dans GNOME 2.22 par exemple). Ils doivent parfois décider de retirer des fonctionnalités à la dernière minute ou de repousser la date de sortie pour améliorer la stabilité. Ce sont des choix difficiles à prendre et, comme le reconnaît

Shuttleworth lui-même, faire ces choix demande beaucoup de discipline.

Même si des cycles réguliers peuvent convenir à certains projets, tenter d'imposer l'adoption de cette approche à tous les projets et ensuite les faire correspondre universellement pourrait gravement endommager le processus de développement. Si les projets deviennent dépendants de la synchronisation, alors un retard à n'importe quelle étape aurait des conséquences sur toutes les autres étapes. Chaque projet subirait alors une pression énorme pour tenir les délais et ce serait néfaste pour le programme et ses utilisateurs finaux.

L'utilisation des branches pour faciliter des sorties régulières

D'après Shuttleworth, de bons outils, en particulier des systèmes de contrôle de version possédant de bonnes capacités de création de branches et de fusion, peuvent rendre ce problème obsolète. Il se réfère spécifiquement à Bazaar, un système de contrôle de version mis au point par Canonical qui s'intègre à la plateforme de développement Launchpad de l'entreprise. J'ai beaucoup testé Bazaar durant ces deux dernières semaines en cherchant des technologies de contrôle de version distribuées et je ne peux qu'être d'accord avec l'argument de Shuttleworth.

Bazaar rend très facile le portage du flot continu de petits changements, du tronc vers les branches, où les fonctionnalités importantes sont développées, afin que ces fonctionnalités puissent être fusionnées sans accroc dans la branche principale quand elles sont achevées. En utilisant cette approche, où la majeure partie du développement est faite dans des branches, le code du tronc est naturellement et systématiquement plus robuste qu'il ne le serait autrement. Shuttleworth va même plus loin encore et théorise que lorsque cette approche est employée en parallèle à des tests automatisés le code du tronc est toujours prêt à être sorti à

n’importe quel moment.

« Un ensemble de tests complet … vous permet d’être plus ouvert aux gros ajouts au tronc parce que les tests assurent les fonctionnalités que les gens ont avant l’ajout. Un ensemble de tests agit comme un champ de force, il protège l’intégrité du code dont le comportement était connu le jour précédent face au changement perpétuel. » écrivait ainsi Shuttleworth sur son blog.

« La plupart des projets que je finance maintenant ont adopté une politique de tests avant ajout. Les ajouts au tronc sont gérés par un robot qui refuse de valider l’ajout s’il ne satisfait pas à tous les tests. Vous ne pouvez pas discuter avec un robot ! Ce que je trouve beau là-dedans c’est que le tronc est toujours dans un état publiable. Ce n’est pas complètement vrai ; on peut toujours faire un peu plus d’assurance qualité avant de sortir quelque chose, mais vous avez cette garantie que l’ensemble de tests est toujours satisfait. Toujours. »

Les ensembles de tests et les très bons systèmes de contrôle de version peuvent simplifier le développement et améliorer la qualité du code, mais ils ne sont pas la panacée. Shuttleworth surestime largement la capacité de ces outils à pallier aux problèmes associés aux sorties régulières. Des bogues surgiront toujours quand de grosses nouveautés sont fusionnées au code existant et parfois ces bogues nécessitent un report de la date de sortie. Si les développeurs ne peuvent ou ne veulent pas faire cela, la qualité du logiciel s’en retrouvera forcément affectée.

Ubuntu 8.04 est le parfait exemple de la voie à ne pas suivre

Pas besoin de chercher très loin pour constater la baisse de qualité résultante d’un engagement sans compromis à un cycle de sorties régulières. Prenez l’exemple de la dernière version

d’Ubuntu. Shuttleworth vante Ubuntu 8.04 comme l’exemple d’une gestion plus intelligente des sorties et soutient que cela démontre la capacité des développeurs à s’en tenir à un programme strict.

« 8.04 LTS représente pour nous un grand pas en avant dans notre conception de la gestion d’une sortie. Pour autant que je sache, jamais une sortie de cette envergure ne s’est faite exactement le jour prévu jusqu’à maintenant, dans le monde des OS propriétaires ou des OS libres. » commente Shuttleworth sur son blog. « Nous avons non seulement démontré que l’on peut préparer une version LTS dans les 6 mois impartis, mais cela prouve également que l’on peut s’engager par anticipation sur un tel cycle LTS. Félicitations aux preneurs de décisions techniques, aux responsables versions et à toute la communauté qui a calqué nos efforts sur le but fixé. »

Ubuntu 8.04, qui est parue le mois dernier, est une version avec support à long terme (LTS pour Long Terme Support), ce qui signifie qu’elle sera maintenue trois ans pour la version Desktop et 5 ans pour la version serveur. Depuis le début, Shuttleworth affirmait aux utilisateurs que la qualité et la fiabilité seraient les mots d’ordre pour la 8.04 et qu’elle serait faite pour durer. Malheureusement, la version n’a pas atteint ces objectifs et est sortie avec quelques bogues importants. Le problème le plus frustrant que nous avons relevé dans notre test d’Ubuntu 8.04 est la configuration défectueuse de PulseAudio, qui affecte à la fois les fonctionnalités audio et vidéo.

Un léger retard aurait permis de résoudre les problèmes de ce genre avant la sortie, mais ce n’est jamais arrivé, peut-être parce que l’engagement de faire la sortie à temps l’a emporté sur l’engagement de la qualité. Mais certains diront qu’une version défaillante n’est pas un problème parce que les bogues peuvent être réparés par des petites mises à jour après sa sortie.

« Les grands déploiements attendent la première ou la deuxième version consolidée de toute façon » fait noter Shuttleworth en réponse à un commentaire sur ton blog *(NdT : La sortie de Ubuntu 8.04.1 est prevue pour le 3 juillet)*. Je me doute que je ne suis pas seul à avoir pensé aux Service Packs de Microsoft en voyant cette remarque. Mais une version officielle n'est-elle pas censée être un gage de qualité ? Si les sorties sont basées sur des jalons arbitraires posés sur une chronologie plutôt que sur une réelle amélioration, alors elles perdent leur sens ou leur pertinence pour les utilisateurs finaux.

D'autres approches

Les cycles de sortie devraient être flexibles et les développeurs devraient pouvoir en ajuster la durée pour qu'ils collent à leur activité. Selon les projets, la culture de développement et les buts peuvent être très différents, les stratégies de publication sont par conséquent différentes. L'appel de Shuttleworth en faveur d'une synchronisation reflète une forme d'incapacité à reconnaître la valeur et la profondeur de la diversité dans la communauté du logiciel libre. Des distributions qui visent des publics différents et qui ont des priorités différentes pourraient ne pas rentrer dans le même moule que les distributions généralistes comme Ubuntu. On retrouve également des logiciels libres multi-plateformes, comme le navigateur Web Firefox par exemple, qui réunissent beaucoup d'utilisateurs sur d'autres systèmes d'exploitation et qui peuvent avoir d'autres priorités que la fréquence de sortie des distributions Linux.

Je tiens à dire quand même que je ne rejette pas catégoriquement les idées de Shuttleworth. Même si je suis vraiment contre une approche descendante et centralisée de la planification des sorties synchronisées je pense qu'il y pourrait y avoir des bénéfices à tirer d'un meilleur alignement du calendrier de quelques distributions principales qui partagent déjà des buts, une technologie et une

méthodologie similaires.

La simultanéité des sorties est déjà à l'ordre du jour (Fedora 9, Ubuntu 8.04 et OpenSolaris 2008.05 ont toutes vu le jour à quelques semaines d'intervalle) et je suis convaincu que de meilleurs résultats sont atteignables si on laisse cette tendance se développer d'elle-même. Encourager trop d'interdépendance créerait des risques sévères, on parle d'un domaine où une planification consciencieuse et un calendrier gravé dans la roche seraient à l'origine de plus de problèmes qu'ils n'en résolvent.

Aaron Seigo, développeur KDE, est l'un des détracteurs ayant exprimé des inquiétudes convaincantes et perspicaces au sujet de la proposition de Shuttleworth. Seigo met à plusieurs reprises en avant que le genre de synchronisation que souhaite Shuttleworth améliore l'efficacité d'intégration au dépend de l'efficacité des développeurs, une concession qu'il décrit comme contre-productive car c'est dans le développement que se trouve le richesse des logiciels.

« Mark parle de processus en flux tendu, mais seulement du point de vue de l'intégration ; il existe aussi des processus en flux tendu dans le développement et définir le cycle de développement à l'aune du cycle de sorties, surtout s'il n'est pas bon, érode la fluidité du flux de développement », écrit Seigo sur son blog. « Il ne faut pas oublier que c'est le processus de développement qui fait toute la valeur d'une distribution Linux. La distribution rend cette valeur accessible à grande échelle et crée un autre type de valeur ajoutée par-dessus (le support, le marketing, etc.) mais c'est le développement, pas l'intégration, qui est la source primaire de valeur. Il devrait alors être évident que le processus de développement n'est pas quelque chose qu'on peut prendre comme ça à la légère. »

Seigo propose une alternative qui faciliterait la synchronisation en aval sans nécessiter de synchronisation ou

de chamboulement en amont. D'après lui, les distributions devraient gérer par elle-mêmes les sorties en créant leurs propres branches et en tenant compte des contraintes de leurs propres cycles.

« Puisqu'il y a cette volonté en aval pour des cycles de parution synchronisés… pourquoi est-ce que l'aval ne prendrait pas en charge les sorties ? Pourquoi attendre que les *tarballs* soient livrées devant leur porte pour mettre en place une équipe de publication ? » s'interroge Seigo. « Pourquoi ne pas demander à la communauté d'intégration (les vendeurs de systèmes d'exploitation en gros) de coordonner leur efforts pour créer une branche en vue d'une sortie à un moment donné, moment qu'ils définissent eux-mêmes, et travailler avec l'amont pour la stabilisation de cette branche ? Plutôt que d'espérer que l'amont fasse ce qu'ils désirent, pourquoi ne peuvent-ils pas regrouper un tas de gars des communautés de chez Novell, Red Hat, Debian, Mandriva, MacOS et Microsoft, de chez Canonical ou encore de chez n'importe qui qui voudrait s'impliquer et offrir un vrai processus sérieux de sortie par lequel l'amont pourrait s'intégrer naturellement ? »

Les suggestions de Seigo sont plus viables que les propositions de Shuttleworth. Elles permettraient aux distributions Linux de bénéficier des avantages pratiques de la synchronisation dont bénéficieraient également les utilisateurs finaux sans avoir à bouleverser ou synchroniser le développement en amont. Cela engendrerait cependant un coût additionnel et un défi nouveau pour les distributeurs et leur ferait porter le poids de la gestion des sorties. Seigo assure que si les distributeurs veulent vraiment des sorties synchronisées en aval autant que ça ils seront prêts à accepter cette charge supplémentaire et trouveront un bon moyen pour y parvenir.

Il est bien probable que cette discussion dure pendant encore quelques temps à mesure que les acteurs principaux pèsent le pour et le contre. La communication a déjà fait avancer le

débat de bien des manières et a déjà fait émerger des alternatives attirantes et des variations de la proposition initiale. Le résultat final pourrait avoir des implications importantes sur la gestion des sorties par les logiciels libres et les distributions, mais pour l'instant aucune des idées proposées n'est suffisamment mature pour être appliquée à grande échelle.

Notes

[1] Traduction : Olivier – Relecture : Daria – Café : Framalang.