UNIX
ton univers impitoyable

PAR JACQUES VÉTOIS

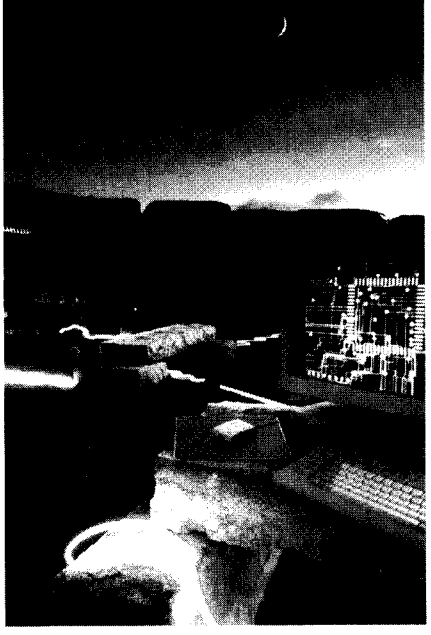
UNIX devient le système d'exploitation standard au bout de vingt années de développement et d'utilisation dans le monde universitaire et scientifique. On le retrouve sur tous les types de machine, du micro ordinateur au mainframe. La leçon devra être méditée par tous les constructeurs. Il devient de plus en plus difficile d'ignorer ce que veulent les utilisateurs : ils ne veulent plus être enchaînés à un seul constructeur et pouvoir modifier à leur guise leur environnement de travail. La micro a déjà imposé un certain type de rapports à la machine. Démarche contagieuse. L'ère des systèmes ouverts commence.

En 1968, deux chercheurs des Laboratoires Bell appartenant au géant des télécom ATT, Ken Thompson et Dennis Ritchie, entreprennent après avoir collaboré au projet de système d'exploitation MULTICS de développer un système multiutilisateur et interactif pour une machine PDP-7 à partir de quelques idées simples: une structure de fichiers hiérarchisée en arborescence sur le disque de la machine, un Shell (ou interpréteur de commandes) programmable.

UN PEU D'HISTOIRE

A l'époque, la plupart des ordinateurs travaillaient en temps différé et les programmeurs soumettaient leurs travaux à la machine à l'aide de cartes perforées. Au départ, UNIX avait été écrit complètement en langage assembleur de l'ordinateur PDP. Le transfert sur une autre machine, un PDP-11 entraîna de nombreuses difficultés. Aussi Ken Thompson et Dennis Ritchie s'orientèrent vers l'utilisation d'un langage évolué . Après plusieurs tentatives, ils créèrent en 1973 le langage C, langage évolué et compilé mais possédant les structures de bas niveaux nécessaires à l'écriture de logiciels de systèmes d'exploitation. Ils réécrivent UNIX en langage C, le rendant portable sur d'autres machines. UNIX était alors utilisé par une vingtaine de chercheurs des Laboratoires Bell. C'était peu. Néanmoins, ce système se diffuse déjà dans quelques laboratoires universitaires. Par contre ATT s'intéressait peu à l'informatique à cette époque. Elle avait le monopole du téléphone aux USA et n'avait pas le droit de commercialiser du matériel informatique.

Pourtant dès 1975, ATT commence à vendre des licences UNIX principalement aux universités et aux centres de recherche. Cette licence procurait une copie du code source et le possesseur de la licence pouvait modifier ce code à volonté et ajouter des commandes et des utilitaires. Ce fut la chance d'UNIX. Il a ainsi été porté sur des machines très diverses ; des versions différentes ont été créées et son développement pris en charge par ceux qui l'utilisaient. L'Université de Berkeley apporta notamment de nouvelles capacités au logiciel de base: un nouveau Shell, le C Shell, des éditeurs pleine page, la mémoire virtuelle et l'ouverture vers le monde des réseaux de télécommunications. UNIX ne serait pas ce qu'il est aujourd'hui si cette période de "développement autogéré" n'avait pas existé. Ainsi, UNIX est le seul système d'exploitation offrant un si grand nombre de possibilités à travers les commandes et les utilitaires disponibles (même si cette richesse est responsable pour une part de la complexité du système). Ce foisonnement a bien sûr engendré des versions incompatibles entre elles,



rendant problématique le portage d'une application d'une machine sur celles ne possédant pas la même source. En 1984, la déréglementation d'ATT brisait son monopole dans le domaine des télécommunications, mais lui laissait en contrepartie les mains libres dans celui de l'informatique. UNIX devenait un produit stratégique pour ATT.

Bataille pour un standard

ATT tenta d'utiliser sa position de force en tant que "père" d'UNIX : d'une part en imposant des conditions draconiennes aux sociétés qui achetaient la licence (droit de regard sur les produits commercialisés et l'interdiction d'utiliser le nom UNIX par exemple) ; d'autre part en prétendant définir seule la norme UNIX, dite "System V" et en se réservant également le droit de décerner des brevets de conformité aux autres versions d'UNIX. ATT publia en 1985 un document, le SVID (System V Interface Définition), formalisant les interfaces de programmation que doit posséder un système pour obtenir le label System V. Dans le même temps, elle commercialisa un logiciel SVVS chargé de vérifier cette conformité. Comme de plus les premières "releases" de System V (System V.O,System V.2,System V.3) offraient moins de possibilités que les versions diffusées par l'Université de Berkeley (4.3 dernière en date), les prétentions d'ATT suscitèrent une levée de boucliers tant de la part des autres constructeurs que des associations d'utilisateurs d'UNIX.

Huit grands constructeurs (IBM, DEC,HP, Bull, Apollo, Siemens, Nixdorf, Hitachi) se regroupèrent dans un front anti-ATT en une fondation "Open Software Fondation" chargée de maintenir l'ouverture du système UNIX menacée par ATT. Ils adoptèrent une série de normes déjà existantes ou en développement rassemblées dans un "guide de portabilité" qui décrit les appels système,les librairies,les commandes mais aussi les normes concernant l'interface graphique (XWindow 11), l'accès aux réseaux...

Le principal groupe des utilisateurs aux USA, l'usr/group veut, lui, définir une norme complète des fonctions minimales d'un système UNIX, Posix et la faire valider comme norme internationale par les organismes de normalisation : l'ANSI et l'ISO. Si cette démarche aboutit, UNIX sera le premier système d'exploitation normalisé. Devant l'impact de ses travaux auprès des utilisateurs (et clients), ATT et l'OSF se sont engagés à respecter les spécifications de Posix. ATT ne tarda pas à riposter à la création de l'OSF en se cherchant des alliés notamment avec SUN, le leader sur le marché des stations de travail.

D'une certaine façon, la situation s'éclaircit même s'il reste deux concurrents de poids au titre de standard d'UNIX: System V (release4) d'ATT et la version OSF basée sur AIX, l'UNIX d'IBM. Les utilisateurs n'en souffriront pas trop, les deux protagonistes s'étant engagés à se conformer à la norme Posix et ayant abandonné, provisoirement du moins, leur volonté d'imposer leur propre norme. La lutte entre les différents constructeurs s'est quelque peu déplacée, en particulier dans le secteur en pleine expansion des stations de travail et des micros haut de gamme du système d'exploitation vers l'interface graphique qui est devenue une nécessité devant le manque total de convivialité des Shell UNIX. X-Window semble s'imposer, face à SunView car elle laisse libre chaque constructeur, voire chaque utilisateur de définir son propre style. UNIX est donc en voie de normalisation. Cette tendance qui ne s'est pas développée jusqu'ici aux dépens des possibilités techniques, est fondamentale pour les utilisateurs. Ils disposent maintenant d'un quasi standard, indépendant de la machine sur laquelle ils travaillent.

L'existence d'UNIX est un progrès indéniable pour l'informatique. Cependant ce standard, si standard il y a, est loin d'être le meilleur possible.

Convivialité zéro

UNIX a introduit dans les années 70 plusieurs idées nouvelles : sa structure hiérarchisée de fichiers, son langage de commande programmable, l'ouverture par l'écriture dans un langage évolué de l'essentiel du système devenu indépendant du système hôte. Mais d'une certaine façon, UNIX porte la marque de son époque de naissance: la grosse informatique régnait en maître, la programma-

Bibliographie

Sur la philosophie d'UNIX, son évolution

- *La vague UNIX* par Martin Leclerc et Brigitte de La Rochelle, éditeur Eyrolles. Une bonne présentation de la philosophie et des possibilités du système UNIX pour des utilisateurs novices. Un seul défaut: ce livre défend inconditionnellent UNIX System V et ATT.
- *Spécial UNIX - La longue Marche* Dossier Le Monde Informatique (21 novembre 1988). La bataille autour du standard UNIX et les évolutions futures du système.
- *C du charabia !* par John Colibri-Revue Pascalissime N°27 (février 1989). Le point de vue d'un spécialiste du langage Pascal sur C .

Pour approfondir UNIX et le langage C

- *Conception du système UNIX* de M.J. Bach éditeur Masson
- *La programmation sous UNIX* de J.M. Rifflet éditeur Mc Graw Hill
- *Le langage C* de Kernighan et Ritchie éditeur Masson

Quelques Termes-clés

Système d'exploitation
Ensemble de programmes complétant les fonctions du matériel et assurant la communication entre les programmes des utilisateurs et les ressources physiques de la machine . Peut permettre en particulier à plusieurs programmes d'utiliser un même ordinateur en apparence (système multiutilisateur comme UNIX). Gère l'accès à la mémoire, aux périphériques...

Disque
Mémoire auxiliaire de grande capacité (par rapport à la mémoire centrale de la machine), mais à temps d'accès beaucoup plus long: de l'ordre de la milliseconde .

Fichier
Ensemble structuré d'informations reliées logiquement entre elles et rangées en général en mémoire auxiliaire (disques ou bandes magnétiques)

Mémoire virtuelle
Caractéristique présente sur certains ordinateurs permettant aux programmes d'utiliser une zone de mémoire apparemment plus grande que la taille de la mémoire centrale qui leur est attribuée. Une partie des objets du programme sont en mémoire centrale et le reste sur disque.

Compilateur
Programme de traduction d'un langage évolué (Fortran, C, Pascal...) en langage assembleur (langage de base de la machine).

Code
Ensemble d'instructions d'un programme

Editeur de textes
Traitement de textes spécialisé dans l'écriture de programmes .

Librairie

Ensemble de procédures ou de fonctions fournies par le système d'exploitation, les compilateurs ou les logiciels et utilisables par le programmeur lors du développement d'une application.

tion était une affaire de spécialistes. Ken Thompson et Dennis Ritchie étaient obsédés par l'efficacité et la concision du code, bien souvent aux dépens de la clarté.

Le développeur de logiciels y trouve une mine de commandes et d'utilitaires astucieux et efficaces qui lui permettent de créer des applications rapidement (des générateurs d'analyseurs syntaxiques comme yacc et lex par exemple), mais le non spécialiste s'y perd un peu et n'utilise en fait que 10% des capacités du système. La syntaxe de ces commandes a évolué au fil des versions et ne possède pas une structure uniforme. De nombreuses fonctions ont été progressivement intégrées au fil des années. Certaines sont redondantes, d'autres dépassées et délicates à manier parce qu'elles font appel à une connaissance approfondie et une bonne pratique du langage C. Cette complexité nuit à l'efficacité globale d'UNIX. Ces défauts seraient mineurs si ne s'y ajoutait pas un, celui-là fondamental: le rôle prépondérant du langage C dont la philosophie et la syntaxe guident tous ceux qui font évoluer UNIX.

C du Charabia !

Je n'irai pas par quatre chemins: en général,un programme C est, sauf si son auteur a fait des efforts méritoires, illisible. C représente une régression par rapport aux langages de programmation qui ont vu le jour à la fin des années 60, Algol 68 et Pascal, et ont permis le développement de la programmation structurée. Il ne possède pas tous les types de données fondamentaux pour une bonne programmation (en particulier le type booléen) et certaines constructions effectuent plusieurs opérations simultanément ce qui ne facilite pas la compréhension des programmes.

Le laxisme des compilateurs C est légendaire. Par exemple, ils ne vérifient ni le type, ni le nombre des paramètres des fonctions définies par le programmeur, ni le type du résultat. D'où une source d'erreurs sans noms lors de l'écriture de grosses applications : la cohérence des interfaces entre des fonctions développées et utilisées par des personnes différentes ne peut être vérifiée par le compilateur et doit être recherchée grâce à un "debuggeur" lors de l'exécution des programmes. La maintenance des logiciels n'en est pas facilitée. La redéfinition d'une fonction ne sera pas remarquée par le compilateur et entraînera de nouvelles erreurs en cascade.

Pourtant C posséderait deux avantages indéniables selon ses thuriféraires. D'abord la vitesse du code généré par les compilateurs C . En fait, C est un langage de même niveau et de même complexité que Pascal et Fortran. La qualité du code généré dépend avant tout du compilateur et non du langage lui-même. Si certaines constructions de C étaient adaptées au langage machine du PDP- 11 sur lequel il a été développé la première fois et générait donc du code efficace pour cette machine, ce n'est sûrement plus vrai pour les ordinateurs actuels (du PC au mainframe).

De plus, les différences de temps d'exécution entre C, Pascal, Fortran, sont minimes (de l'ordre de 5 à 10%) et peuvent être négligées dans la plupart des applications .

L'autre argument en faveur de C est celui de la portabilité d'une application d'une machine à une autre. En fait, le manuel initial du langage *(The C programming language de Kernighan et Ritchie)* n'est pas complet et laisse des imprécisions dans certaines constructions . Une norme ANSI existe, mais elle n'est pas respectée par tous les compilateurs, loin de là.

Passage au C++ ?

Ceux qui essayent de porter des logiciels écrits en C d'une machine UNIX à une autre en font parfois la dure expérience. Les Entrées/Sorties ne font pas partie du langage mais sont fournies sous forme de librairies. Rien ne garantit que le programme à transférer n'appelle pas des fonctions non disponibles dans celles du système de votre machine. Tout le monde s'accorde plus ou moins sur les limites du langage C. Il est de plus en plus question que la future "release" de System V soit réécrite en C++ qui en apportant les techniques de la programmation-objet corrige les imperfections les plus criantes de son prédécesseur.

UNIX a ouvert une ère nouvelle pour les systèmes d'exploitation. Celle des systèmes propriétaires, liée à un constructeur, est révolue (cela n'empêche pas IBM et Microsoft de vouloir en imposer un sur les micros haut de gamme OS2, avec peu de succès semble-t-il). C'est une bonne chose. Mais UNIX, malgré sa normalisation n'est qu'un point de départ car il ne prend pas en compte les problèmes du temps réel. Le parallélisme, l'informatique répartie, commencent à en poser de nouveaux; l'usage des supports multimédias également. UNIX devra évoluer où laissera la place à d'autres systèmes.

■

What's GNU ? Gnu's Not Unix

GNU est le nom d'un ensemble de logiciels entièrement compatibles avec le système UNIX développé par la Free Software Fondation créée par Richard Stallman du MIT. Richard Stallman est connu dans les milieux informatiques comme le créateur de l'éditeur de textes EMACS, qui devient la référence comme éditeur de texte sous UNIX. La Free Software Fondation entend réécrire et diffuser largement et gratuitement une version d'UNIX avec de nombreux utilitaires. Des compilateurs C et C++, un éditeur de liens, un "debugger" ont déjà vu le jour et tout le monde s'accorde sur la qualité de ces logiciels.

Au moment où les éditeurs de logiciels entament une véritable campagne policière contre la copie des logiciels avec des arguments sur la propriété intellectuelle qui dissimulent mal leurs véritables raisons, sonnantes et trébuchantes, il nous a semblé opportun de publier des extraits du "Manifeste de GNU" qui affirme d'abord que "copier tout ou partie d'un programme est aussi naturel pour un programmeur que respirer". Qui a pratiqué un tant soit peu la programmation ne peut qu'être d'accord avec celà. Les propositions de Richard Stallman feront sans doute sourire. Mais c'est oublier que des idéalistes comme lui ont plus apporté à l'informatique que ceux qui engagent des procédures judiciaires pour défendre leur droit de propriété sur le "look" d'une interface qu'ils n'ont même pas inventée.

J.V.

Pour contacter la Free Software Foundation, écrire à : Free Software Foundation 675 Massachussetts Avenue CAMBRIDGE MA 02139

■ "POURQUOI TOUS LES UTILISATEURS DE LOGICIELS EN BÉNÉFICIERONT ?"

Lorsque le GNU sera disponible, tout le monde sera en mesure d'obtenir de bons logiciels gratuitement, comme l'air. Non seulement cela permettra d'économiser à tout le monde le prix d'une redevance Unix, mais cela évitera la duplication peu rentable du système de programmation. Au lieu de cela l'effort pourra porter sur l'amélioration du système.

Les programmes sources seront disponibles pour tout le monde. En conséquence un utilisateur désireux d'effectuer des changements dans le système sera libre de le faire lui-même, ou de faire appel à n'importe quel programmeur ou entreprise. Les utilisateurs ne seront plus à la merci d'un programmeur ou d'une entreprise qui détiendrait les programmes sources...

Les écoles seront en mesure de fournir un environnement plus pédagogique en encourageant les étudiants à étudier et à améliorer le code du système. Le laboratoire informatique de Harvard avait la politique suivante : aucun programme ne pouvait être mis en place sur le système si ses sources n'étaient pas accessibles publiquement, et il appliqua cette politique en refusant de mettre en place certains programmes. Je m'en suis beaucoup inspiré.

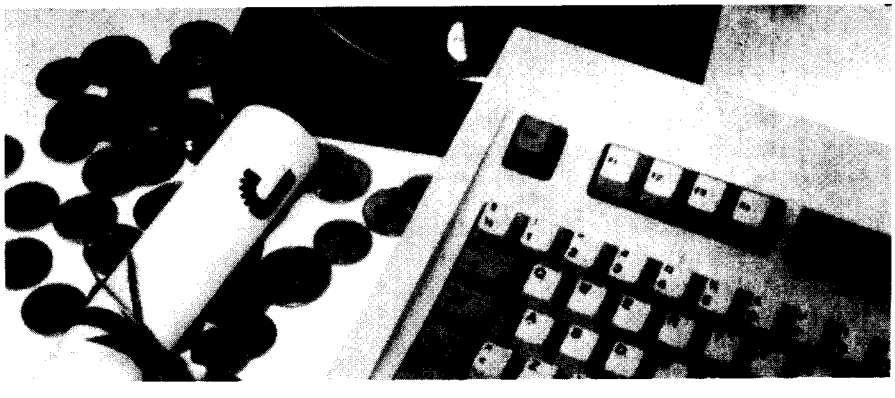
Finalement la question de savoir qui possède les logiciels du système, et ce que l'on peut faire avec, ne se posa plus...

Nous devons distinguer entre le support sous forme de travail de programmation réelle et la simple maintenance. Si votre entreprise a besoin d'un support, la seule façon de l'avoir est de posséder les sources et les outils nécessaires. Vous pourrez alors faire appel à n'importe qui pour régler votre problème, vous ne serez plus à la merci d'un seul individu. Chez Unix, le prix des sources rend cette solution impossible pour la plupart des entreprises. Avec GNU, cela sera facile. Il est possible qu'il n'y ait pas de personne compétente disponible, mais ce problème ne peut être imputé aux conditions de distribution. GNU n'élimine pas tous les problèmes du monde mais seulement quelques uns...

■ "LES PROGRAMMEURS NE MÉRITENT-ILS PAS UNE RÉCOMPENSE POUR LEUR CRÉATIVITÉ ?"

Si quelque chose mérite une récompense, c'est la contribution sociale. La créativité peut être une contribution sociale dans la mesure où la société est libre d'en utiliser les résultats. Récompensés pour avoir créé des programmes innovateurs, les programmeurs méritent aussi d'être punis s'ils limitent l'utilisation de ces programmes. Il est normal de vouloir que son travail soit rétribué, ou de tenter de maximaliser ses revenus, dans la mesure où l'on n'utilise pas de moyens destructeurs. Mais les moyens utilisés dans le domaine des logiciels sont basés sur la destruction.

Extorquer de l'argent aux utilisateurs d'un programme en limitant son usage, est une démarche destructrice car ces limitations réduisent les champs d'applications et la richesse que l'humanité pui-



se dans ce programme. Lorsqu'il y a une volonté délibérée de limitation, les conséquences néfastes sont délibérément destructrices...

■ "Les gens n'ont-ils pas le droit de controler l'utilisation de leur créativité?"

"Le contrôle de l'utilisation de ses idées" constitue un contrôle de la vie des autres gens ; et il est généralement destiné à leur rendre la vie plus difficile. Les gens qui ont étudié le problème des droits de propriété intellectuelle (les avocats par exemple) affirment qu'il n'existe pas de droit intrinsèque à la propriété intellectuelle. Les types de droits de propriété intellectuelle reconnus par le gouvernement furent créés par des actes législatifs spécifiques, dans des buts spécifiques...

L'idée du copyright n'existait pas autrefois, lorsque les auteurs copiaient d'autres auteurs (dans le détail) dans la littérature non romanesque. Cette pratique était utile et explique pourquoi les œuvres de nombreux auteurs ont survécu, ne serait-ce qu'en partie. Le système de copyright fut créé expressément dans le but d'encourager la paternité littéraire. Dans le domaine littéraire - les livres pouvaient être reproduits de façon économique par l'imprimerie - le système de copyright ne fut pas dommageable et il ne fut pas un obstacle à la lecture des livres. Tous les droits de propriété intellectuelle ne sont que des licences accordées par la société car l'on pensait, à tort ou à raison, que la société toute entière en tirerait profit.

Le cas des programmes actuels est très différent de celui des livres il y a un siècle. Etant donné que la solution la plus simple pour dupliquer un programme est de le passer d'un ordinateur à celui d'à côté, qu'un programme s'utilise par nécessité et non par plaisir, tout ceci crée une situation où une personne qui applique la loi du copyright nuit à la société à la fois matériellement et spirituellement. On ne devrait pas agir ainsi ! peu importe que la loi le permette ou non...

■ "La concurrence rend les choses mieux faites."

La concurrence peut être comparée à une course : en récompensant le vainqueur, on encourage tout le monde à courir plus vite. Lorsque le capitalisme fonctionne ainsi, il fait du bon travail, mais ses défenseurs ont tort de prétendre qu'il fonctionne toujours de cette façon. Si les coureurs oublient pourquoi la récompense est offerte et tentent de gagner par n'importe quel moyen, ils peuvent trouver d'autres stratégies comme par exemple attaquer les autres coureurs. Si les coureurs s'empoignent, ils finiront tous en retard. Les logiciels secrets et de marque déposée sont les équivalents moraux des coureurs engagés dans une empoignade...

■ "Tout le monde ne va-t-il pas cesser la programmation si la motivation financière est supprimée ?"

En fait, beaucoup de gens font de la programmation sans aucune incitation d'ordre financier. La programmation exerce une fascination irrésistible sur certaines personnes, principalement sur les gens qui excellent. On ne manque pas de musiciens professionnels même si ceux-ci n'ont aucun espoir de gagner leur vie en faisant de la musique...

Mais en fait cette question, bien que très communément posée, n'est pas très pertinente par rapport à la situation. Le salaire des programmeurs ne sera pas supprimé, il sera simplement réduit. La question qu'il faut se poser est donc : les gens continueront-ils à programmer avec une incitation financière moindre ? mon expérience me conduit à répondre oui...

■ "Les programmeurs ont besoin de gagner leur vie d'une façon ou d'une autre."

A court terme, c'est vrai. Cependant les programmeurs ont de multiples façons de gagner leur vie sans vendre le droit d'utiliser un programme... En voici quelques unes :

Un fabriquant qui met un nouvel ordinateur sur le marché supportera le coût du port des systèmes d'exploitation sur le nouveau matériel. Les services d'enseignement et de maintenance pourraient également employer des programmeurs. Des gens ayant le sens de l'innovation pourraient distribuer des programmes gratuitement, en demandant des donations aux utilisateurs satisfaits ou en vendant des services de maintenance. J'ai rencontré des gens qui travaillaient de cette façon avec succès...

Toutes sortes d'aménagements peuvent être envisagés avec une taxe sur les logiciels : supposez que tous les gens qui achètent un ordinateur doivent payer x pour cent du prix pour la taxe sur les logiciels. Le gouvernement donnerait cet argent à une agence comme la NSF afin de contribuer au développement des logiciels.

Mais si l'acheteur d'ordinateur fait une donation de lui-même pour le développement des logiciels, il bénéficiera d'un crédit de taxe. Il pourra faire une donation pour le projet de son choix - qu'il choisira parce qu'il espère bénéficier des résultats de la recherche. Il peut bénéficier d'un crédit pour une donation de n'importe quel montant et pouvant égaler le montant total de la taxe qu'il doit payer...

■ "Quelles en seront les conséquences ?"

- La communauté utilisatrice d'ordinateurs supportera le coût du développement des logiciels.
- Cette communauté décidera quel est le niveau de support nécessaire.
- Les utilisateurs qui le souhaitent pourront choisir d'acquitter une taxe pour le projet de leur choix.

A long terme la gratuité des programmes constitue une étape vers une société d'abondance où personne ne devra travailler beaucoup pour gagner sa vie. Les gens seront libres de se consacrer à des activités d'agrément telles que la programmation après avoir passé une dizaine d'heures sur des tâches obligatoires telles que la législation, la concertation familiale, la réparation des robots et la recherche d'astéroïdes. Il ne sera plus nécessaire de programmer pour gagner sa vie.