Le top 10 des pièges des logiciels embarqués open source

Attention à l'utilisation d'un système d'exploitation temps réel (RTOS) ou d'un autre logiciel open source dans un produit commercial. Certains problèmes associés sont potentiellement à prendre en compte et un examen attentif peut convaincre les entreprises envisageant d'utiliser un RTOS open source de revoir leur copie et d'opter plutôt pour un RTOS commercial. Explications de l'éditeur américain Express Logic, connu pour son système d'exploitation temps réel ThreadX.

AUTEUR

John Carbone, directeur et membre du bureau exécutif d'Express Logic.

Le logiciel open source a été une aubaine pour l'industrie informatique, tant au niveau de l'entreprise qu'au sein du monde de l'embarqué. Les contributions de la communauté open source ne peuvent pas être ignorées tant au niveau des outils de développement qu'à celui des systèmes d'exploitation en passant par les applications. En règle générale, ce type de logiciel est libre d'utilisation sous une licence telle que la Gnu Public License (GPL). Mais l'utilisation de l'open source dans les domaines de l'embarqué et de l'IoT nécessite des précautions particulières de la part des équipementiers. Nous nous concentrerons ici sur les systèmes d'exploitation en temps réel (Real-Time Operating System, RTOS). Un RTOS est utilisé dans de nombreux produits embarqués ou IoT comme les équipements domotiques, les dispositifs portés sur soi, les instruments médicaux et les systèmes de contrôle industriels. Dans de tels produits, un RTOS exécute les logiciels applicatifs écrits par le développeur du produit en permettant le multitâche, la communication, la gestion de la mémoire et d'autres fonctionnalités qui profitent à l'application et facilitent le développement de produits complexes.

Lors de la sélection d'un RTOS pour un nouveau projet, les développeurs peuvent choisir parmi trois alternatives générales : le développement en interne (DIY, Do It Yourself), la licence d'un RTOS commercial ou la licence d'un RTOS open source. Nous n'allons pas explorer la première option qui est très rarement employée par les développeurs de produits actuels. Le choix du « fait maison » est généralement limité à des cas d'usage relativement simples ou à des systèmes historiques pour lesquels la migration peut nécessiter un effort important. Bien que le DIY puisse fonctionner correctement dans certains cas, il est plus probable que les développeurs choisiront entre les deux autres alternatives. Ce document se concentrera donc sur les problèmes liés à l'utilisation d'un RTOS open source dans un produit commercial embarqué / IoT.

Voici ce que nous considérons comme les dix principaux pièges de l'utilisation d'un RTOS open source dans des produits commerciaux.

● Un RTOS est utilisé dans de nombreux produits embarqués ou IoT comme les équipements domotiques, les dispositifs portés sur soi, les instruments médicaux et les systèmes de contrôle industriels. Dans de tels produits, le RTOS exécute les logiciels applicatifs écrits par le développeur du produit en permettant le multitâche, la communication, la gestion de la mémoire et d'autres fonctionnalités qui profitent à l'application et facilitent le développement de produits complexes.

● 1 - Fiabilité

La fiabilité d'un RTOS est critique pour la fiabilité du produit final au sein duquel il est intégré. Si le RTOS tombe en panne ou se comporte de façon inattendue, le produit lui-même tombera probablement en panne ou se comportera de manière imprévisible. C'est l'une des raisons pour lesquelles les RTOS utilisés dans les systèmes liés à la sécurité doivent être certifiés par des organismes de réglementation. Les RTOS open source ne sont généralement pas certifiés et ne peuvent donc pas être utilisés dans des systèmes liés à la sécurité. Mais les développeurs de tout produit devraient se préoccuper de l'utilisation d'un RTOS non certifié pour la sécurité, car même si un dysfonctionnement du RTOS n'entraînera pas forcément de blessure ou de mort pour l'utilisateur d'un tel produit, il pourrait conduire probablement à ce que le produit échoue à effectuer la tâche prévue. À tout le moins, cela mènerait probablement à l'insatisfaction des clients, à de mauvaises ventes et peut-être même à un rappel de produits. La certification pour la sécurité est un moyen pour tout fabricant de produits d'avoir la certitude que le RTOS a été soigneusement évalué, testé et prouvé pour fonctionner comme prévu. En fait, un RTOS open source populaire comme FreeRTOS a dû être réécrit pour la sécurité et proposé en tant que variante commer-

ciale. Cette variante, SafeRTOS, a le même modèle fonctionnel que le noyau FreeRTOS, mais a été reconstruite pour plus de sécurité. On peut donc se demander ce que cela implique sur la sécurité et la fiabilité du noyau open source FreeRTOS...
Une autre façon de s'assurer qu'un RTOS est fiable est de regarder où il est utilisé, et quelle est la popularité des produits l'utilisant. Si un RTOS est utilisé dans des produits à faible volume, ou juste pour la recherche ou l'évaluation, difficile de tabler sur sa fiabilité. Cela pourrait être le cas, mais cela n'est pas démontré par un usage aussi limité. Par contre, un RTOS largement utilisé dans de nombreux produits très performants – a fortiori issus des leaders du marché – a plus de chance d'être fiable. Les larges déploiements de RTOS sont un indicateur fort de la fiabilité du système d'exploitation, de sa facilité d'utilisation et de son succès global. En général, de par leur fiabilité non prouvée et pour les raisons que nous discuterons ci-dessous, les RTOS open source ne sont pas utilisés dans les produits populaires et à fort volume. Aucun RTOS open source ne revendique par exemple des déploiements de plus d'un milliard d'unités. Si un tel OS est utilisé dans des produits commerciaux, les développeurs doivent fournir des efforts supplémentaires en matière de débogage et de développement applicatif pour s'assurer d'un fonctionnement correct. Mais le temps passé et les surcoûts sont-ils indispensables pour fabriquer un produit fiable, alors qu'un RTOS fiable et éprouvé est disponible par ailleurs ?

2 - Sécurité

La sécurité de l'open source représente un véritable défi. Dans l'open source, le code source, par définition ouvert, est librement disponible pour que quiconque puisse l'examiner et, éventuellement, concevoir un moyen de le subvertir. Sa popularité est son principal danger, et, s'il est utilisé dans un produit commercial à succès, la motivation des pirates est accrue, que ce soit pour l'extorquer ou pour causer une perturbation maximale de type cyber-terrorisme. En outre, les composants open source peuvent contenir des failles de sécurité qui pourraient être exploitées pour menacer n'importe quel produit dans lequel ils sont utilisés.
Par exemple, mbed TLS est un produit open source qui prend en charge les architectures Arm et fait partie de l'environnement Arm mbed OS. mbed TLS était auparavant connu sous le nom de PolarSSL, un produit open source, avant d'être inclus dans mbed OS par Arm. PolarSSL est un exemple de composant de sécurité open source avec des vulnérabilités connues publiquement.
Les logiciels commerciaux ont de bien meilleures chances d'éviter de telles vulnérabilités et donc de se défendre contre des attaques néfastes. Les éditeurs de logiciels commerciaux emploient en interne des développeurs de logiciels qualifiés et encadrés, et n'utilisent aucun logiciel externe de pedigree inconnu (Software Of Unknown Pedigree, SOUP – voir le point 9a ci-dessous). En procédant ainsi, le fournisseur a un contrôle total sur chaque ligne de code de ses produits et peut empêcher plus facilement toute introduction involontaire ou même intentionnelle de code qui pourrait aider un pirate informatique. Avec un RTOS open source, on connaît peu les auteurs du code.
Les logiciels de communication du commerce, comme les piles TCP/IP, peuvent être sécurisés par l'absence de points d'entrée involontaires (en tant que système fermé). Cela rend impossible la prise de contrôle à distance ou l'attaque par déni de service. En outre, la sécurité des données peut être assurée avec les protocoles IPsec, SSL, TLS et DTLS qui utilisent un cryptage puissant pour empêcher la prise en main non autorisée du trafic de données à travers les réseaux, y compris Internet. Avec l'open source, il y a beaucoup plus d'inconnues au sujet du code. Est-ce du logiciel SOUP qui est innocemment apporté comme contribution, mais éventuellement avec des défauts ? Est-ce que ce code est issu d'une personne mal intentionnée qui cherche à saboter un produit ? Les risques sont réels et le seul recours pour l'open source consiste à s'appuyer sur l'examen de la communauté pour bloquer un tel code avant qu'il ne devienne une version déployée. C'est un risque, et pour de nombreuses entreprises, celui-ci est trop grand.

3 - Indépendance

Un véritable RTOS open source repose sur la communauté et ne dépend d'aucune organisation. Les RTOS open source sont parfois pris en charge et sous-licenciés par des organisations commerciales, ce qui en fait des offres pseudo-commerciales. D'autres RTOS open source peuvent être contrôlés, ou « gérés » par une organisation commerciale, apportant une valeur ajoutée aux utilisateurs des produits commerciaux de cette organisation. Il n'est pas clair si une telle gestion inclut une modification du RTOS open source lui-même, mais il semble probable que cela soit bien le cas. Si tel est le cas, le RTOS open source prend un aspect pseudo-commercial et le support ne peut provenir que de la société qui le gère. En tant que tel, il devient alors la propriété ou au moins bénéficie du soutien d'une seule et unique organisation, faisant cesser son indépendance et sa gestion par la communauté. Par exemple, mbed OS n'est disponible

Selon Express Logic, éditeur du RTOS ThreadX, dix pièges peuvent guetter l'utilisateur d'un système d'exploitation open source dans des produits commerciaux.

que pour les processeurs Arm, donc son utilisation est un verrouillage efficace de la part d'Arm. Ceci limite les options pour une utilisation future sur un microprocesseur différent.
Le manque d'indépendance suscite aussi des inquiétudes quant à l'utilisation d'un RTOS open source dans un environnement incompatible avec l'organisme le gérant. A titre d'exemple, maintenant qu'Amazon AWS est devenu responsable de FreeRTOS, il pourrait y avoir des inquiétudes quant à l'utilisation de FreeRTOS avec Google Cloud ou un autre service IoT dans le nuage...

● 4 - Performances

En fonction de l'application, la rapidité avec laquelle les services RTOS peuvent être exécutés peut faire la différence dans les performances et la fiabilité d'un produit. Des services plus rapides signifient moins d'overhead au niveau du RTOS, plus de temps CPU disponible pour l'application et une plus grande marge de sécurité dans les environnements exigeants (par exemple, en cas d'interruptions rapides nécessitant un changement de contexte fréquent).
La performance de tout RTOS peut être mesurée et quantifiée. La suite de tests Thread-Metric d'Express Logic par exemple mesure la performance de n'importe quel RTOS en termes de temps d'exécution pour chacun des services RTOS de base: traitement des interruptions, préemption, commutation de contexte, sémaphores, mutexes, transmission de messages et gestion de la mémoire.
Grâce à la mesure réelle des performances d'un RTOS – open source ou commercial – les développeurs sont mieux informés sur la capacité de celui-ci à atteindre les objectifs de performance du produit. La plupart des RTOS commerciaux publient des rapports de performance pour les services de base ou rendent ces données disponibles pour l'évaluation. Cependant, les développeurs doivent prendre garde à « comparer des produits comparables ». Différents fabricants peuvent effectivement mesurer des fonctionnalités de nature différente, mais se référer à celles-ci sous couvert d'un même nom, ce qui rend une comparaison directe trompeuse. En utilisant le même code de référence pour la mesure, ce problème est évité et une véritable comparaison peut être faite.
De même, la taille du code du RTOS doit être prise en compte. Une taille de code plus petite permet l'utilisation de microprocesseurs moins coûteux, moins de mémoire et laisse plus de place au code de l'application. La taille du code est simple à mesurer et n'est pas aussi sujette aux variations que pour les performances publiées. Cependant, il y a quelques variables à considérer, à savoir les options du compilateur, le compilateur lui-même et les dépendances. Il est préférable de faire une mesure précise de l'empreinte du code pour un environnement de développement spécifique, en utilisant les outils qui seront utilisés pour le développement réel du produit afin de mesurer toutes les alternatives.
La plupart des RTOS commerciaux ont des avantages significatifs en termes de performances par rapport à l'open source puisqu'ils sont en général de 4 à 8 fois plus rapides. De même, en ce qui concerne la taille, la connectivité au cloud peut être fournie en moins de 30 Ko, tandis que les composants open source peuvent prendre plus de 200 Ko. De nombreux composants de type open source sont en fait écrits en C++, bien que la cible prévue soit un capteur qui nécessite un encombrement réduit. L'utilisation du C++ a non seulement un impact négatif sur la quantité utilisable de mémoire flash, mais elle a un impact encore plus important sur la RAM et la complexité en général. L'open source peut ne pas être un bon choix pour les petites cibles du type Cortex-M ou autres microcontrôleurs.

● Un RTOS open source populaire comme FreeRTOS a dû être réécrit pour la sécurité et proposé en tant que variante commerciale. Cette variante, SafeRTOS, a le même modèle fonctionnel que le noyau FreeRTOS, mais a été reconstruite pour plus de sécurité. On peut donc se demander ce que cela implique sur la sécurité et la fiabilité du noyau open source FreeRTOS...

● 5 - Manque de fonctionnalités avancées

Les RTOS open source tendent à être simples, conçus pour exécuter des services RTOS de base qui permettent le fonctionnement d'un dispositif embarqué IoT. Des fonctionnalités telles que l'ordonnancement préemptif sont généralement fournies, ainsi que la gestion des interruptions, les sémaphores, la transmission de messages, etc. Les RTOS commerciaux offrent souvent des fonctionnalités supplémentaires, reposant sur une technologie plus avancée, en partie pour donner à ces RTOS plus de valeur ajoutée par rapport à la concurrence. De telles caractéristiques à valeur ajoutée comprennent l'ordonnancement avec seuil de préemption (Preemption-Threshold Scheduling), la trace en temps réel intégrée, les modules d'application téléchargeables, la protection de la mémoire, le chaînage d'événements, entre autres fonctionnalités. Ces caractéristiques avancées fournissent aux développeurs d'applications des outils supplémentaires qui accélèrent le fonctionnement de leur logiciel et facilitent également le développement et le débogage. Le résultat est un produit IoT embarqué plus efficace et plus performant, qui arrive plus rapidement sur le marché, et qui connaît une meilleure réussite tout au long de son cycle de vie.

● 6 - Middleware

La plupart des RTOS open source consistent en un simple noyau. Alors que les RTOS commerciaux peuvent également être décrits comme comportant seulement un noyau, la différence est qu'ils sont aussi entourés d'un middleware commercial supplémentaire. Ce middleware peut inclure un système de gestion de fichiers, une pile de réseau TCP/IP, la prise en charge de l'USB hôte/périphérique, un environnement graphique, des interfaces aux services IoT dans le nuage… Bien que chacun de ces composants de

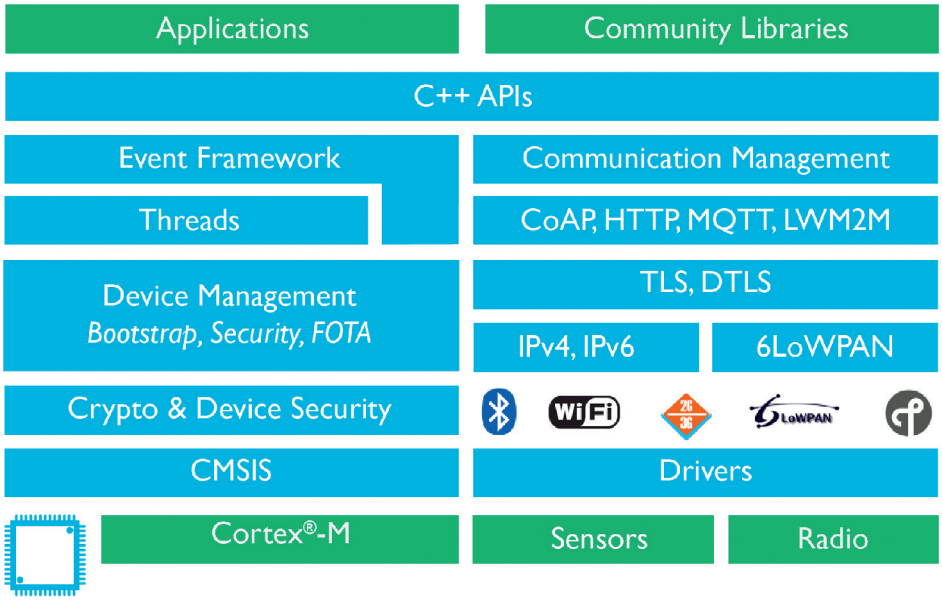
middleware puisse être disponible pour une utilisation avec un RTOS open source, ils ne sont pas toujours intégrés ni pris en charge par une seule organisation responsable. Cette intégration, que nous appelons « Integration Gap », doit être réalisée par quelqu'un – généralement le développeur du produit final. Cela entraîne des délais, des coûts et des risques d'erreurs supplémentaires. C'est l'une des raisons pour lesquelles les RTOS commerciaux ont plus de succès, car ils permettent à leurs utilisateurs d'achever le projet à temps, voire même en avance. Ce succès des RTOS commerciaux est mis en évidence par les données de l'enquête réalisée auprès des développeurs par Embedded Market Forecasters *(http://www.newelectronics.co.uk/electronics-technology/an-industrial-grade-rtos-could-get-products-to-market-more-quickly/155137/)*.

Un RTOS commercial avec middleware est généralement préintégré et pris en charge par une seule entité. Cette entité a tout intérêt à faire fonctionner tous les composants de manière harmonieuse, efficace et sans conflit dans un système IoT/embarqué. Cela minimise les surprises et le travail d'intégration supplémentaire pendant le développement, ce qui rend plus probable l'achèvement d'un projet en temps voulu.

Certaines des solutions open source pour l'IoT se sont développées en prenant l'hypothèse que la tâche la plus importante de l'application est d'échanger avec le cloud. En conséquence, l'accent a été mis sur la connectivité au cloud, avec peu de considération pour les autres attentes liées au RTOS d'un appareil IoT. En réalité, un produit IoT a généralement beaucoup plus de fonctionnalités, et la connectivité au cloud n'en est qu'une petite partie.

1 ARCHITECTURE DE L'ENVIRONNEMENT ARM MBED OS

mbed TLS est un produit open source qui prend en charge les architectures Arm et fait partie de l'environnement Arm mbed OS. mbed TLS était auparavant connu sous le nom de PolarSSL, un produit open source, avant d'être inclus dans mbed OS par Arm. PolarSSL est un exemple de composant de sécurité open source avec des vulnérabilités connues publiquement.



● 7 - Support

7+-L'une des préoccupations les plus importantes liées à l'utilisation d'un RTOS open source dans un produit commercial est peut-être la question du support. Les RTOS open source n'ont pas de support professionnel et commercial. Ils s'appuient sur une communauté qui peut fournir des suggestions et des conseils pour les questions qu'un développeur pourrait se poser. Parce qu'ils sont gratuits, les RTOS open source ne permettent pas de financer une équipe de support disponible pour les développeurs. De son côté, un RTOS commercial est généralement disponible avec le soutien commercial complet des auteurs et des responsables de la maintenance du produit, moyennant un coût. Ces frais rémunèrent le personnel de support et permettent de répondre avec réactivité aux besoins des développeurs.

Il existe également un risque à long terme associé à l'open source. La communauté open source ne garantit généralement aucune rétrocompatibilité (en fait, FreeRTOS a modifié ses API publiées par le passé) et peut changer n'importe quoi en fonction des besoins de la communauté, notamment l'API, les termes de licence, etc. Les fournisseurs de RTOS commerciaux garantissent quant à eux la rétrocompatibilité de l'API, et les conditions de licence sont fixées sous forme contractuelle et ne peuvent être modifiées unilatéralement.

● 8 - Coût

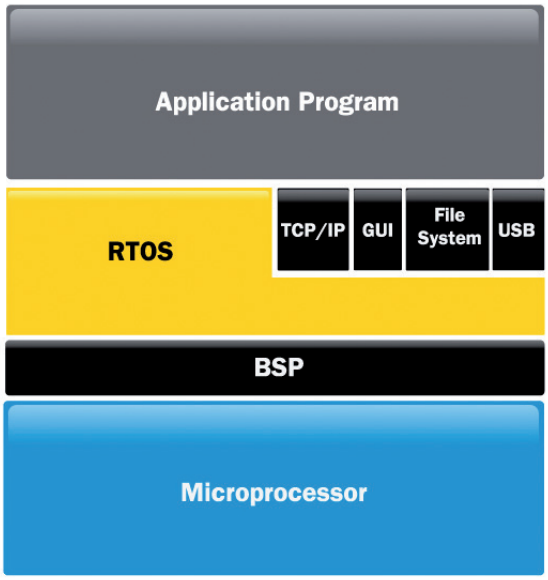
On fait souvent remarquer que les RTOS open source, et l'open source en général, sont « libres ». C'est vrai dans la plupart des cas pour la licence d'utilisation du code lui-même. Mais ce n'est pas le seul coût d'utilisation de l'open source. Comme décrit ci-dessus, le support de la communauté n'est pas satisfaisant pour un projet de développement d'un produit commercial. Par conséquent, certains fabricants devront fournir un support en interne de type DIY. Avec l'accès au code source permis par l'open source, il est en effet possible de mettre en place une équipe d'ingénieurs de support experts dans le logiciel open source, et disponibles pour les utilisateurs en interne. Cependant, un tel personnel a besoin d'une formation et doit être payé pour le temps passé à aider les utilisateurs.

Un autre coût de l'open source « libre » est celui de l'intégration, comme nous l'avons vu plus haut. L'écart d'intégration (Integration Gap) nécessite du temps de la part des développeurs, ce qui se traduit par des coûts, des délais et des risques. Ces coûts doivent être reconnus et, en fin de compte, pèsent sur le budget du projet.

Dans la section suivante, nous discuterons d'un autre coût supplémentaire, celui lié aux aspects juridiques. Ceux-ci incluent la violation des droits de propriété intellectuelle, et l'obligation de partager librement le code propriétaire avec la commu-

2 ENVIRONNEMENT CLASSIQUE D'UN SYSTÈME D'EXPLOITATION TEMPS RÉEL

La plupart des RTOS open source consistent en un simple noyau. Alors que les RTOS commerciaux peuvent également être décrits comme comportant seulement un noyau, la différence est qu'ils sont aussi entourés d'un middleware commercial supplémentaire. Ce middleware peut inclure un système de gestion de fichiers, une pile de réseau TCP/IP, la prise en charge de l'USB hôte/périphérique, un environnement graphique, des interfaces aux services IoT dans le nuage.



nauté. Ces coûts peuvent être substantiels, et les ignorer peut être catastrophique pour une entreprise commerciale.

● 9 - Aspects juridiques

Il existe de nombreuses considérations juridiques à prendre en compte lors de l'utilisation d'un RTOS open source dans un produit commercial. En voici trois qui sont couramment rencontrées :

9a - SOUP

Le logiciel de pedigree inconnu (Software of Unknown Pedigree, SOUP) est un risque. L'utilisation d'un SOUP, à partir d'un téléchargement issu de la communauté, est risquée car même une simple ligne de code peut enfreindre les droits de propriété intellectuelle d'un auteur (copyright ou brevet). Une injonction sur une violation aussi minime pourrait impliquer des coûts juridiques substantiels, de la perte de temps et un jugement obligeant à changer le code du produit pour éviter d'utiliser le code mis en cause. À tout le moins, le paiement d'un droit de licence pourrait être la solution la plus simple pour éviter une telle situation.

9b - Obligation de divulguer le code propriétaire

Certaines licences open source, comme la GPL par exemple, exigent que les utilisateurs de l'open source retournent à la communauté tout logiciel qu'ils écrivent et qui est combiné ou lié (dans certains cas) au logiciel open source lui-même. Cela signifie que le code de l'application propriétaire, qui constitue la valeur ajoutée logicielle rendant un produit commercial compétitif, doit être partagé avec le public – y compris vos concurrents ! C'est une obligation inconfortable à assumer, et une bonne raison de ne pas utiliser l'open source dans un produit commercial. Si vous le faîtes, il est conseillé de lire attentivement le contrat de licence pour voir ce que vous êtes autorisé à faire avec le logiciel sous licence, et ce que vous êtes obligé de faire en retour.

9c – Fiabilité du produit

Des lois sont en train d'être promulguées pour renforcer la responsabilité vis-à-vis des produits défectueux, telle que la directive que la Commission européenne a instaurée en ce sens. Aux USA, la FDA (Food and Drug Administration) a décidé que les CEO des sociétés qui fabriquent des dispositifs médicaux pourraient être tenus responsables du mauvais fonctionnement d'un produit. Que ces lois soient ou non en place, le concepteur d'un produit devrait suivre les « meilleures pratiques » lors de la mise au point dudit produit afin de maximiser sa sûreté et de se prémunir contre les allégations de négligence de sa part. Sur ce point, l'open source présente un énorme problème, car il est difficile d'argumenter que l'utilisation de logiciels libres sur Internet soit la meilleure pratique en la matière.

● 10 - Absence de pression commerciale pour améliorer l'open source

L'open source n'est pas un logiciel commercial. Cela signifie qu'il n'est pas générateur de revenus, mais qu'il nécessite en revanche le développement et la maintenance par la communauté. Un logiciel commercial, lui, est développé pour générer des revenus. C'est son but principal et il est souvent essentiel à la prospérité de l'entreprise qui le développe voire même à sa survie. La pression concurrentielle exercée par les développeurs de logiciels commerciaux génère une motivation pour réussir, ce qui les pousse à améliorer leurs produits au fil du temps. Il n'y a pas de telle pression pour l'open source. Oui, il y a des développeurs communautaires intrépides qui se soucient vraiment d'améliorer le logiciel libre, et il existe un mécanisme permettant à la communauté de demander de nouvelles fonctionnalités et de corriger certains bogues en priorité. Mais en aucun cas le travail d'un membre de la communauté n'est en jeu, pas plus que la survie d'une organisation. Les éditeurs de logiciels commerciaux sont toujours en situation de concurrence. S'il n'y a pas de concurrence, il n'y a pas de marché. Afin de survivre dans un tel environnement, les développeurs de logiciels commerciaux investissent dans des ingénieurs et d'autres employés pour qu'ils travaillent ensemble afin de déterminer les besoins des clients et de les satisfaire de la meilleure façon possible. Les entreprises qui ne réussissent pas dans cette pratique ne survivent pas et finissent par fermer ou par être absorbées par d'autres entreprises pour un prix au rabais. Ceux qui survivent et contrôlent leur avenir investissent des fonds importants dans l'amélioration et l'expansion de leurs produits, qui, espèrent-ils, généreront des revenus en retour. Cette dynamique fondamentale profite au consommateur, lui garantissant l'accès aux meilleurs produits issus d'entreprises qui se sont forgées dans l'adversité. ■

● La performance de tout RTOS peut être mesurée et quantifiée. La suite de tests Thread-Metric d'Express Logic par exemple mesure la performance de n'importe quel RTOS en termes de temps d'exécution pour chacun des services RTOS de base : traitement des interruptions, préemption, commutation de contexte, sémaphores, mutexes, transmission de messages et gestion de la mémoire.