Quelles solutions pour Linux embarqué ?

Pierre Ficheux (pierre.ficheux@openwide.fr)

Décembre 2005

Résumé

Le but de cet article est de replacer Linux dans le contexte des systèmes industriels et embarqués. Le précédent article « Introduction aux systèmes embarqués » a permis de définir la terminologie et le champ d'application de cette technologie. De notre côté, nous nous attacherons à expliquer brièvement quels sont les avantages de Linux dans cet environnement ainsi que les composants logiciels disponibles (compilateurs, débogueurs, extensions temps-réel, etc.)

Nous effectuerons également un tour d'horizon des autres solutions disponibles tout en positionnant Linux parmi cette liste et ce en s'aidant de quelques données statistiques. Les références à différents articles, site web ou ouvrages traitant plus précisément des sujets cités sont données en bibliographie.

Logiciel libre et systèmes embarqués

La qualification de logiciels libre nécessite de satisfaire à un certain nombre de critères. La liste complète est disponible auprès du site http://www.opensource.org mais nous pouvons retenir trois critères fondamentaux pour le logiciel embarqué:

- La disponibilité du code source
- La possibilité de réaliser des travaux dérivés
- La redistribution sans *royalties*

La disponibilité du code source est un critère fondamental car contrairement au logiciel classique (comme le logiciel bureautique), la durée de vie d'un logiciel embarqué est particulièrement longue car elle est liée à la durée de vie de l'équipement matériel qui l'héberge. Des contraintes économiques et légales font que certains biens de consommation doivent être maintenus au moins 10 ans. Cette durée est parfois beaucoup plus longue dans le cas de matériel militaire ou scientifique. De ce fait il sera nécessaire de faire évoluer ce logiciel sur un matériel considéré comme obsolète et ce indépendamment des aléas économiques comme par exemple la disparition d'un éditeur de logiciel. Certaines licences associés aux logiciels libres (comme la GPL ou la LGPL) imposent la disponibilité du code source « ad vitam aeternam » et ce dernier ne pourra donc être séquestré même pour de sombres raisons légales ou financières.

La réalisation de travaux dérivés est un avantage compétitif certain. Il paraît absurde de nos jours de développer une bibliothèque de traitement JPEG ou XML. Ce n'était pas forcément le cas il y a encore quelques années, ou bon nombre de petites entreprises , par ignorance ou par obstination, se lançaient dans de coûteux développements sans considérer l'existant déjà disponible à l'époque au travers du logiciel libre (je parle malheureusement en connaissance de cause !). Le problème des licences est à considérer avec soin dans le cas du travail dérivé mais cela n'a rien d'insurmontable et des règles simples découlant du bon sens suffisent largement au respect de licences comme la GPL ou la LGPL.

La redistribution sans royalties est un atout économique évident dans le cas de la diffusion en masse d'un équipement. Avec le critère de disponibilité du code source (lui-même inspiré par des

contraintes économiques) c'est certainement un des principaux arguments motivant l'adoption des logiciels libres en remplacement de solutions propriétaires.

Et Linux dans tout ça ?

Le système Linux bénéficie d'un « héritage multiple » le positionnant comme favori dans la course aux logiciels embarqués.

- C'est un système UNIX. Son image est donc associée à une réputation de fiabilité, de robustesse et de professionnalisme. Il est d'ailleurs important de noter que plusieurs systèmes d'exploitation concurrents (dans le monde industriel) sont des systèmes plus ou moins « UNIX-like ». Nous pouvons citer VxWorks, pSOS, QNX, LynxOS, OS/9, les trois derniers étant réellement compatibles UNIX, les autres moins. Il également noter que les principaux éditeurs propriétaires du domaine ont désormais – de gré ou de force - une offre Linux à leur catalogue!
- Il bénéficie de l'expérience assez longue de son utilisation pour des applications serveurs qui nécessitent également une grande fiabilité et robustesse. Des expériences réussies de migration vers Linux étaient donc disponibles avant l'utilisation dans les applications embarquées.
- Il profite de la méfiance (certains diront de l'ostracisme) des utilisateurs et développeurs du domaine embarqué pour les solutions de type Microsoft (exception faite des PDA, mais nous sommes alors à la frontière entre l'embarqué et le PC)
- Il est relativement simple à adapter à des architectures très variées et a de plus profité du rouleau compresseur Intel et assimilés de ces dernières années. La société Intel est d'ailleurs fortement impliquée dans le support de ses produits pour Linux.
- Linux est un système complexe (donc gourmand en ressources) mais de part l'évolution matérielle (mémoire, puissance de calcul, interchangeabilité du matériel), il est avantageux de l'utiliser dans la majorité des cas.
- De part son développement naturellement réparti, il profite de la mondialisation et il est devenu un système de référence dans les pays asiatiques, premiers producteurs de matériel.
- Il accompagne de la montée en puissance des pays émergents qui de toute façon ne peuvent pas payer les licences propriétaires.
- De part ses capacités d'inter-opérabilité, il a profité de l'avènement d'Internet et de l'introduction de versions communicantes des objets de la vie courante (automobile, Hi-Fi, téléphonie, domotique, vidéo-sécurité, etc.)

De ce fait, un étude récente (2004) montre qu'un très grand nombre d'utilisateurs de solutions industrielles se sont tournés vers Linux ces deux dernières années, et qu'ils seront probablement encore plus nombreux ces deux prochaines années.

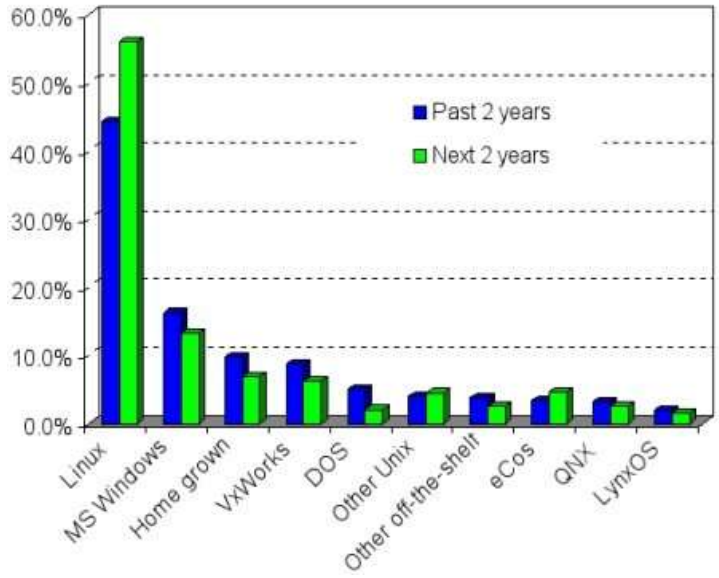

Figure 1: Répartition des systèmes utilisés (source Linuxdevices.com)

L'offre Linux embarqué aujourd'hui

Suite à l'intérêt de bon nombres d'industriels pour les solutions « Linux embarqué », des éditeurs se sont lancés dans la réalisation de distributions spécialisées. Dans les rangs de ces compagnies, on compte des éditeurs de solutions propriétaires ou bien des sociétés nées de l'agitation autour des logiciels libres. La société LynuxWorks, éditeur du système LynxOS, a depuis longtemps considéré Linux avec attention, au point de changer de nom pour une consonance plus « Linux » et éditer une version spécialisée de Linux (appelée *BlueCat*) en plus de son système propriétaire.

D'autres sociétés comme le leader des systèmes embarqués et temps-réel Wind River ont récemment rejoint le mouvement en proposant des environnements de développement utilisables pour Linux. On peut raisonnablement se demander si c'est fait à contre-coeur ou simplement dans le but de capter quelques clients hésitants en leur faisant profiter à la fois d'une offre propriétaire classique et rassurante tout en proposant une ouverture vers Linux et ce au sein d'un même environnement de développement intégré. Un dirigeant de Wind River (Narem Gupta) est même entré récemment au comité de direction de Red Hat!

L'éditeur MontaVista a de son coté une approche différente, puisqu'il propose uniquement des solutions Linux incluant le support de nombreuses cibles matérielles (x86, PowerPC, MIPS, ARM, etc.) et un modèle économique basé sur du support technique mais aussi du service et du développement spécifique.

Cependant, des études récentes (2004) ont montré que les distributions Linux embarqué des éditeurs représentaient une faible part du pourcentage du choix des utilisateurs, soit 8% pour le leader MontaVista. La majorité des utilisateurs préfèrent construire une distribution « maison » (*home grown* en anglais) ou adapter des distributions classiques comme la Debian, la Mandriva ou la Red Hat. Le graphique ci-dessous montre la répartition des choix des utilisateurs.

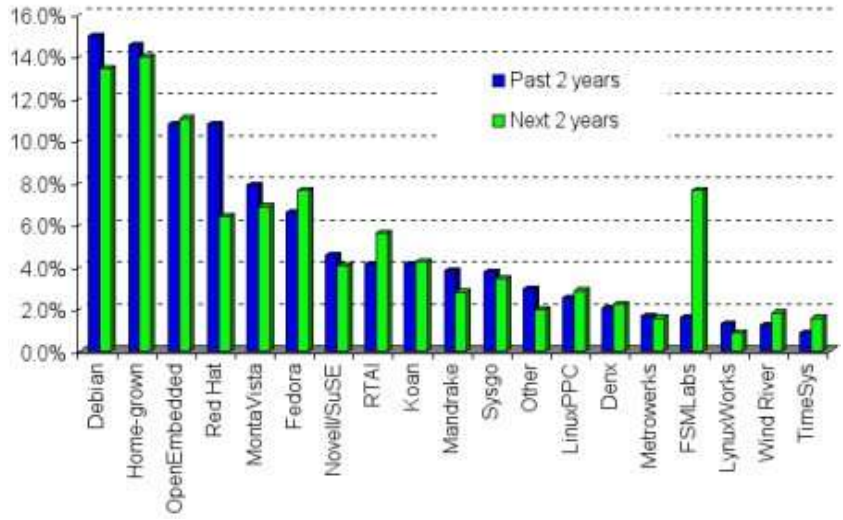

Figure 2: Répartition des distributions « Linux embarqué » (source Linuxdevices.com, 2004)

Les solutions et outils disponibles

La raison du choix des utilisateurs est assez évidente: il existe à ce jour un grand nombre de solutions et de composants logiciels disponibles. Une société désirant effectuer une migration stratégique vers Linux ne va donc pas forcément s'orienter vers un produit d'éditeur mais plutôt souvent acquérir la connaissance en interne afin de maintenir sa propre distribution. Par contre, dans le cas d'une première utilisation, le choix d'une solution éditeur est intéressante car on aura à disposition des composants sous forme de paquetages prêt à l'emploi ainsi que des outils graphiques qui pourront aider les habitués de Windows à s'acclimater à ce nouvel environnement.

Les chaînes de compilation

La chaîne de compilation (compilateur, assembleur, débogueur) est un ensemble de composants indispensables au développement. Les distributions Linux sont livrées avec la chaîne GNU (gcc, gdb, binutils) mais dans le cas d'un système embarqué il sera souvent nécessaire de mettre en place une chaîne de compilation « croisée » permettant de développer sur un PC (Linux ou Windows) des applicatifs pour le système cible (exemple: système avec processeur ARM tournant sous Linux, ou système eCOS).

La mise au point des programmes

La chaîne GNU fournit le débogueur symbolique `gdb`. Son interface est assez rustique mais il peut être piloté par diverses applications plus ou moins graphiques (GNU-Emacs, X-Emacs, Kdevelop, etc.) afin de faciliter son utilisation.

Dans le cas d'une cible avec une faible empreinte mémoire (peu de flash et peu de mémoire vive) il est très intéressant de pouvoir mettre au point l'application directement sur la cible. Pour ce faire, la chaîne GNU permet d'utiliser un agent appelé `gdbserver`. Il est beaucoup plus petit que `gdb` (quelques dizaines de Ko) et permettra de faire le lien entre le programme `gdb` exécuté sur le système de développement et l'application exécutée sur la cible.

Coté cible on pourra exécuter l'application de la manière suivante.

```
$ gdbserver 192.168.1.1:4444 mon_application
```

L'adresse 192.168.1.1 correspond au système de développement, la valeur 4444 à un port TCP disponible pour le dialogue réseau entre `gdb` et `gdbserver`.

Coté poste de développement, on utilisera la syntaxe suivante.

```
$ arm-linux-gdb mon_application

(gdb) target remote 192.168.3.2:4444
(gdb) b main
(gdb) continue
```

L'adresse IP 192.168.3.2 correspond à celle de la cible. Notez que dans notre cas, la cible utilise un processeur de type ARM. Le programme `gdb` à utiliser sur le poste de développement (PC x86) n'est pas la version *native x86* mais un débogueur croisé (*cross debugger)* intégré à la chaîne de compilation croisée, soit `arm-linux-gdb`.

Le principal problème en programmation C/C++ (surtout en C) est celui des « dépassements de tableau ». En général on alloue un espace de mémoire grâce à l'appel système `malloc` puis on dépasse joyeusement cet espace alloué ce qui a pour effet de provoquer une « fuite de mémoire », ce qui a le désagrément de n'être pas détectable immédiatement et provoque donc une érosion des performances du système. Il existe divers outils open source ou propriétaires permettant de traquer ce type de problème.

Le plus simpliste est *Electric Fence*, écrit par le célèbre Bruce Perens. C'est une simple ré-écriture de la fonction `malloc`, remplacée alors par une version beaucoup plus lente mais ne tolérant aucun dépassement de mémoire. Cette solution peut s'avérer suffisante pour un programme de taille raisonnable comportant des erreur de mémoire flagrantes. Considérons le petit programme suivant, qui contient une grossière erreur de dépassement de tableau.

```
#include <stdio.h>
#include <stdlib.h>

main (int ac, char **av)
{
  register int i = 0;
  int *tab;

  /* on alloue l'espace pour 10 entiers */
  tab = (int*)malloc (10 * sizeof(int));

  /* on stocke finalement 20 entiers ! */
  while (i < 20) {
    *(tab+i) = i;
    i++;
  }
}
```

On compile deux versions du programme, une avec la fonction `malloc` standard, l'autre avec de la bibliothèque Electric Fence.

```
$ make
gcc -g   -c -o bad.o bad.c
gcc -g -o bad bad.o
gcc  -g -o bad_ef bad.o -lefence
```

Lorsqu'on exécute le programme standard `bad`, le débogueur `gdb` n'y voit que du feu.

```
$ gdb bad
...
(gdb) run
..
Program exited with code 0170.
```

Par contre, la version `bad_ef` compilée avec Electric Fence s'arrête exactement au dépassement avec une erreur de violation de partage (SIGSEGV), soit pour la valeur 10 de la variable `i`. Le débogueur indique également la ligne provoquant le signal SIGSEGV. Dans le cas d'une pile d'appels de fonctions, on pourra utiliser les fonctions *info stack* ou *backtrace* de `gdb` pour revenir à l'origine de l'erreur.

```
(gdb) run
  Electric Fence 2.2.0 Copyright (C) 1987-1999 Bruce Perens <bruce@perens.com>

Program received signal SIGSEGV, Segmentation fault.
0x0804847b in main (ac=1, av=0xbffff894) at bad.c:11
(gdb) p i
$1 = 10
```

Cet exemple est cependant assez simpliste et il existe également des outils plus évolués, mais aussi plus complexes à mettre en oeuvre comme *Valgrind* ou *Mpatrol*.

Du coté des outils propriétaires, on peut citer *Purify* ou *Insure++* .

L'instrumentation des applications (profiling)

Même si une application ne comporte pas d'erreurs pouvant provoquer une fuite de mémoire ou un plantage, il peut être très intéressant de disposer d'un outils (profiler) permettant d'optimiser le code: détection du code mort, optimisation des fonctions les plus utilisées.

Les développeurs Linux connaissent depuis longtemps le profiler fourni par la chaîne GNU soit *gprof* héritier de l'outil UNIX *prof* datant de 1979. Cet outil nécessite l'utilisation de l'option de compilation `-p` et est quelque peu dépassé par des outils plus récents et non intrusifs.

L'outil *Kcachegrind* est une interface graphique utilisable avec le module *Cachegrind* de mesure de performances livré avec *Valgrind*. Dans le même ordre d'idée, nous pouvons citer les outils *OPprofile* et *PCL*.

L'outil *LTT* (Linux Trace Toolkit) permet d'effectuer un instrumentation fine au niveau du noyau pour prendre en compte les contraintes temporelles et la synchronisation entre plusieurs sections de codes ce qui est très intéressant pour les application embarquées utilisant du temps réel.

La mise au point du noyau et pilotes

La mise au point du noyau et des pilotes de périphériques est plus complexe car elle s'effectue dans l'espace noyau (kernel space) et non dans l'espace utilisateur. Dans cet espace mémoire, un problème de programmation peut être fatal à la survie de la session, et donc d'autant plus difficile à traquer. Les deux principales solutions disponibles dans le monde open source sont *KDB* et *KGDB*.

Le composant KDB est développée par la société SGI (ex Silicon Graphics) qui depuis son virage Intel s'est concentrée sur le logiciel et a donc abandonné son système UNIX propriétaire (*IRIX*, celui que l'on voit dans « Jurassic Park ») pour Linux. SGI a contribué à quelques développements très

intéressants comme le portage de son système de fichier journalisé XFS aujourd'hui en standard dans le noyau 2.6. KDB se présente comme un « patch » du noyau standard. Dans des conditions particulières (séquence de touches ou plantage) on peut accès à l'invité `kdb>` qui permet de visualiser, modifier des variables du noyau, exécuter en pas à pas, etc.

Le composant KGDB a une approche différente qui s'apparente plus à celle de l'agent `gdbserver` décrit précédemment. Un agent KGBD est installé sur le système cible et l'on peut utiliser `gdb` depuis un autre système connecté via un lien série RS-232.

Les éditeurs propriétaires comme Wind River proposent également des fonctionnalités équivalentes, parfois supérieures, ou du moins annoncées comme telles. Notez également que le débogueur `gdb` (et l'environnement GNU en général) est souvent intégré à des produits propriétaires destinés à la mise au point « bas niveau » comme Lauterbach *TRACE32* et Abatron *BDI2000*.

Les environnements de développement (IDE)

Les aficionados de Linux et d' UNIX en général ont l'habitude d'utiliser la chaîne de développement GNU dans sa version originale, soit en mode texte. C'est une approche assez austère pour les développeurs habitués aux environnements graphiques comme ceux de Windows ou MacOS. De ce fait, il existe pas mal de solutions permettant d'agrémenter l'interface des outils GNU.

Emacs/XEmacs propose des interfaces permettant d'utiliser les outils de compilation et le débogueur `gdb`. C'est pratique pour les habitués mais le taux de bouton au mètre carré reste faible.

Kdevelop est un composant livré avec l'interface graphique KDE. Le résultat est très proche de Windows, soit fenêtres, boutons, souris et menus à profusion. Même si ce n'est pas un outil spécifique à Linux embarqué, il peut être configuré facilement pour gérer une chaîne de développement croisée.

Eclipse est aujourd'hui de plus en plus utilisé pour les outils de développement Linux. Initialement prévu pour le développement Java, Eclipse a été enrichi de nombreuses extensions et il fait maintenant partie de l'offre Linux des principaux éditeurs (MontaVista, WindRiver, LynuxWorks, Altera, etc.).

L'éditeur Wind River propose depuis peu son *Workbench* permettant de développer code compatible avec Linux et VxWorks via un IDE. L'environnement inclut un débogueur en mode utilisateur et noyau.

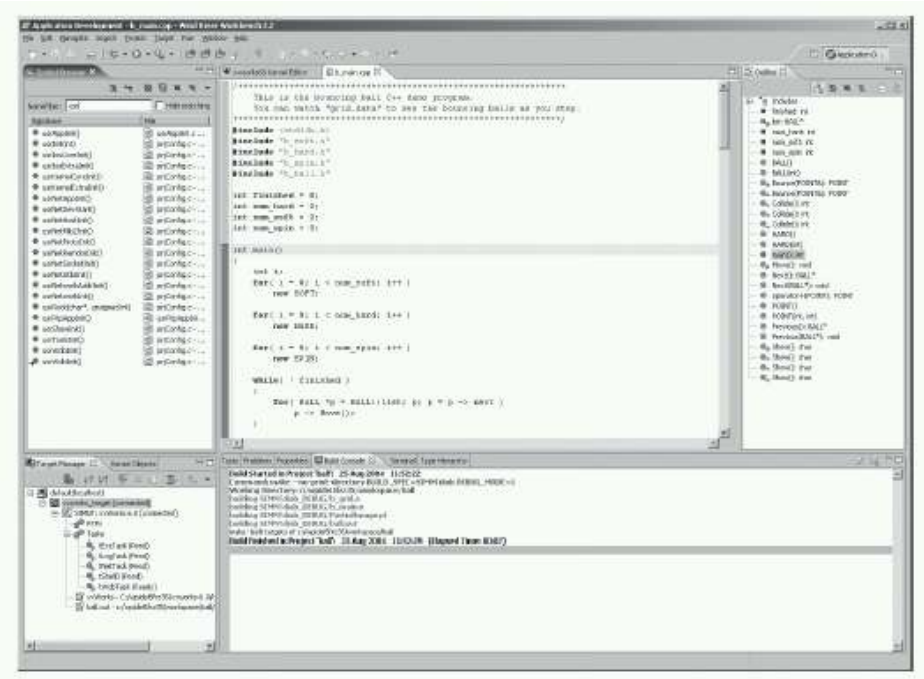

Figure 2: Le Workbench de Wind River, basé sur Eclipse

Le leader des distributions Linux embarqué MontaVista propose un produit similaire appelé *Devrocket* présenté sur la figure suivante.

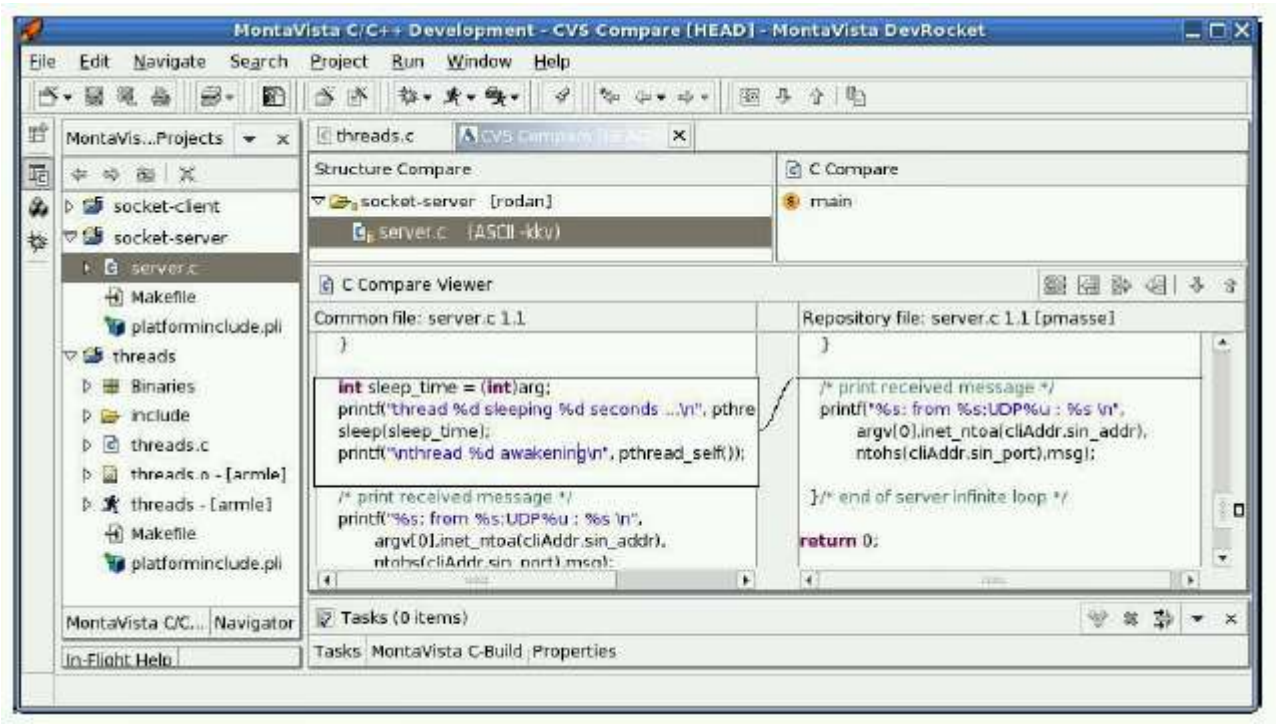

Figure 3: Devrocket de Montavista, basé sur Eclipse

Enfin, LynuxWorks propose deux outils de développement intégrés: *VisualLynux*, fonctionnant sous Windows et s'installant au dessus de Microsoft Visual Studio 6.0 et *Luminosity*, un IDE basé encore une fois sur Eclipse.

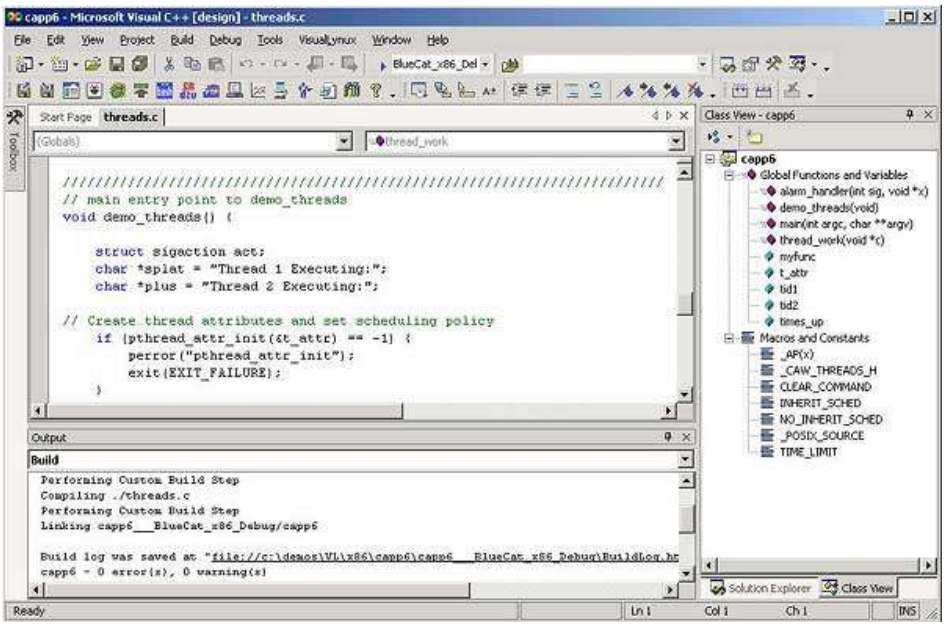

Figure 4: VisualLynux de LynuxWorks

CYGWIN, le développement Linux sous Windows

Ce n'est pas forcément la meilleure solution mais c'est parfois la seule lorsque le responsable du service informatique est définitivement rétif aux PC Linux!

La société CYGNUS, acteur historique du monde de l'open source (Cygnus, Your GNU Support) développe depuis longtemps CYGWIN, un environnement complet permettant de *compiler* des applications Linux sous Windows (attention, ce n'est pas un émulateur). Cela fonctionne très bien même si c'est en moyenne 30% moins efficace en temps de compilation qu'un développement natif sous Linux. La quasi-totalité des composants Linux sont disponibles sous CYGWIN: chaîne GNU, X11, KDE, etc. La figure ci-dessous montre l'utilisation de CYGWIN dans un environnement Windows 2000.

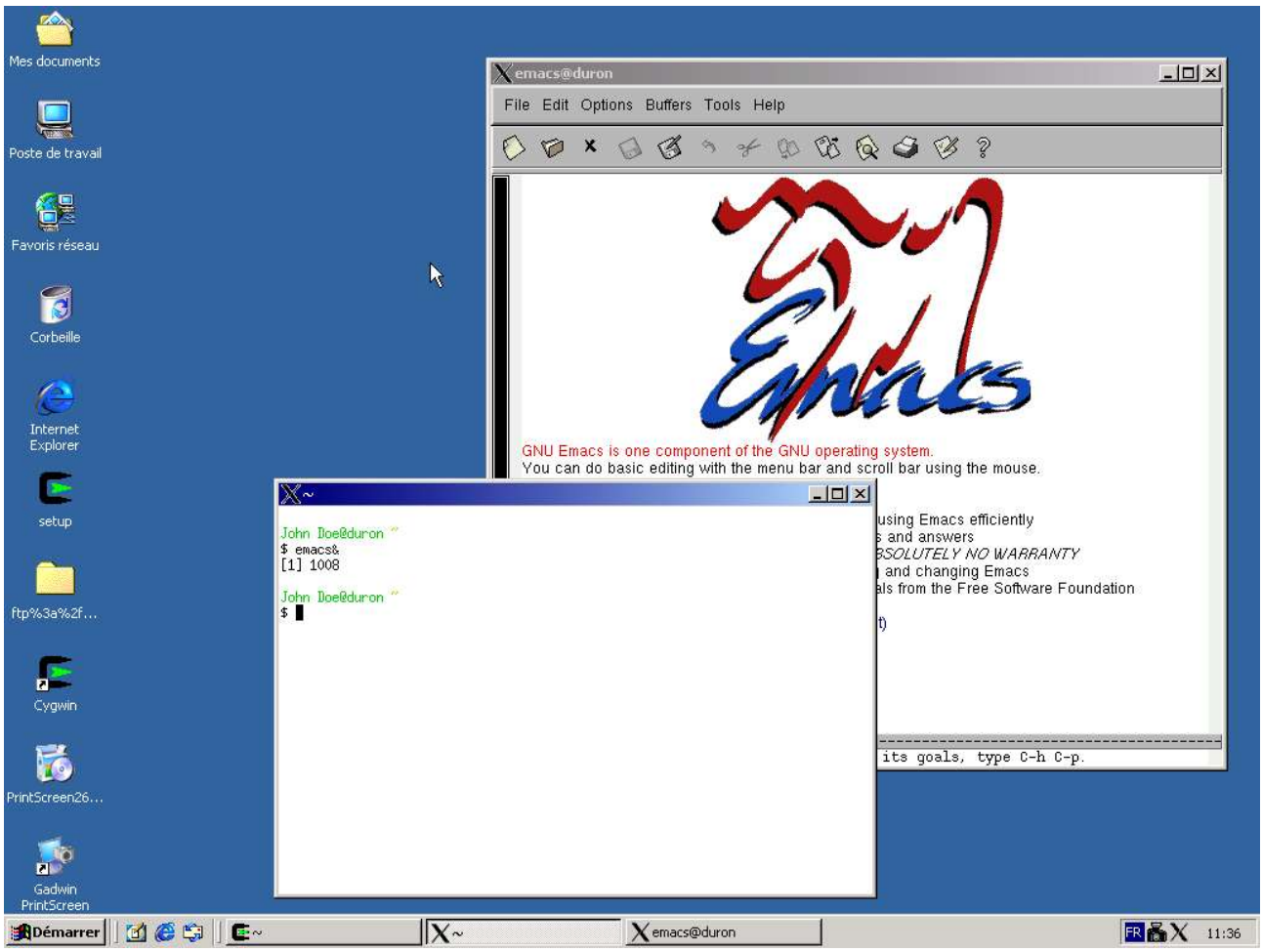

Figure 5: Environnement CYGWIN (Emacs et xterm) sous Windows 2000

Les interfaces homme machines (IHM) pour Linux embarqué

L'environnement graphique *X Window System* (X11) domine depuis longtemps le marché des interfaces graphiques sous UNIX. Ce choix n'est pas forcément le plus judicieux car X11 est complexe, lourd et souvent sous utilisé dans le cas des applications embarquées. Depuis quelque temps l'utilisation du *frame-buffer Linux* se répand de plus en plus. Ce mode de fonctionnement correspond au pilotage du contrôleur graphique directement à partir du noyau Linux (fichier spécial `/dev/fb0`). Il existe aujourd'hui plusieurs bibliothèques graphiques de haut niveau pouvant utiliser le frame-buffer comme alternative à la couche X11. Cela permet de construire des applications graphiques portables, plus facile à mettre au point tout en restant compatible avec les contraintes de Linux embarqué.

La bibliothèque *Qt/Embedded* éditée par *Trolltech* est certainement la plus utilisée pour les applications graphiques complexes. Elle est compatible avec *Qt* et diffusée sous double licence GPL et propriétaire. La bibliothèque permet également de développer du code portable sur Windows et MacOS.

La bibliothèque *GTK/Embedded* correspond à une version frame-buffer de *GTK+*. Elle est un peu moins lourde que Qt/Embedded mais ne dispose pas du même niveau de finition ni d'un support commercial. Elle est diffusée sous LGPL.

La bibliothèque *Nano-X Window* (ex-MicroWindows) est beaucoup plus légère mais elle est réservée aux applications graphiques simples (voire simplistes). Elle est diffusée sous LGPL.

Dans le cas d'applications moins évoluées graphiquement (utilisant des afficheurs LCD de petite taille, souvent monochromes), on peut utiliser les composants *LCDProc* et *LCD4Linux*.

Les extensions temps réel

Linux fut développé sur les concepts d' UNIX, ce n'est donc pas un système temps réel, contrairement à la plupart des systèmes embarqués propriétaires comme VxWorks, QNX ou LynxOS. Au niveau de Linux, il existe divers composants permettant de mettre en place des systèmes temps réel et ce à plusieurs niveaux de contraintes (temps réel *mou* ou *dur*).

Le noyau Linux 2.6 inclut un nouvel ordonnanceur permettant d'améliorer les temps de latence du système. La configuration est accessible dans le menu *Processor types and features/Preemption Model* de la configuration du noyau. Un tel noyau permet d'obtenir des performances satisfaisantes dans le cas d'applications de temps réel *mou*.

Dans le cas de temps réel dur, il est nécessaire d'utiliser des extensions. Concrètement, cela correspond à un noyau temps réel auxiliaire chargé dans l'espace du noyau Linux sous forme de modules.

Le composant *Xenomai 2*, dérivé du projet RTAI permet de développer des programmes temps réel dur dans l'espace utilisateur (et non dans l'espace noyau), ce qui est un avantage notoire. Un article consacré aux solutions temps réel et présenté dans ce même numéro donnera des détails plus pratiques sur ce composant et sur les performances du noyau 2.6.

Dans le monde propriétaire, nous pouvons citer *RTLinux* de *FSMLabs* et *Jaluna 2* de *JALUNA*. Le premier (américain) était initialement un projet open source mais a définitivement viré du côté « obscur » en ne maintenant plus que la version propriétaire de sa solution (sans oublier un dépôt de brevet logiciel). Le deuxième (français) a une architecture un peu différente et le noyau temps réel utilisé n'est autre que C5 (Chorus 5) qui n'est pas un logiciel libre mais qui est bien connu des spécialistes du domaine.

Le matériel pour Linux embarqué

Il n'existe bien évidement pas de matériel spécifique à Linux, cependant l'apparition du logiciel libre dans les applications embarquées a permis le développement de produits à des coûts plus intéressants puisque les composants logiciels de la cible et de l'environnement de développement (SDK: *Software Development Kit*) sont exempts de coûts de licences. Nous allons dans ce paragraphe décrire brièvement quelques composants matériels fréquemment utilisés avec Linux embarqué.

Les cartes mères

D'après une étude de *Linuxdevices.com*, les plate-formes matérielles les plus utilisées en 2004 pour Linux embarqué sont dans l'ordre l'ARM (32 %), le x86 (29 %) et le PowerPC (15 %). Bien que Linux provienne initialement du monde x86, l'article d'introduction aux systèmes embarqués a montré que l'architecture PC x86 n'était pas toujours bien adaptée à certaines applications du domaine, en particulier dans le cas de forte intégration, de faible consommation, ou d'empreinte mémoire réduite. Cependant l'architecture x86 occupe une place importante dans un certain type

d'application assez proche du PC industriel « durci ». De plus, la mise en place sur le marché de processeurs compatibles x86 à « faible » consommation (VIA C3) permet d'envisager une utilisation avantageuse dans certains cas:

- Maquettage avant développement d'une solution à grande échelle. On se rapproche de la configuration finale au niveau de l'empreinte mémoire sans investir dans un matériel spécialisé.
- Production d'un faible volume. Dans ce cas la, des solutions comme les cartes Mini-ITX/Nano-ITX (basées sur des processeurs VIA) ou des PC/104 sont intéressantes. Dans le cas du PC/104, le coût est assez élevé mais on arrive à une solution modulaire du fait de la structure mécanique (cartes enfichables sous formes de pile). Il faut également noter que dans beaucoup d'applications industrielles ou militaires le coût d'un tel matériel sera négligeable par rapport à l'enveloppe globale.

  Dans le cas d'un besoin d'intégration important, la plate forme x86 n'est pas non plus inaccessible et nous pouvons citer le *DilNet PC* diffusé par la société allemande *SSV Embedded*.
- Peu ou pas de connaissance spécifiques en interne. La plate-forme x86 est un terrain connu de tous et beaucoup d'industriels on déjà réalisé des applications embarquées dans un environnement PC x86 sous DOS. La migration vers Linux n'est pas immédiate mais elle n'est pas un saut technologique insurmontable et ouvre de larges horizons (réseau, etc.). N'oublions pas non plus qu'une application développée sous Linux est peu dépendante de la plate forme. A terme, une migration vers un processeur plus spécialisé sera toujours possible, parfois sous la forme d'une simple « recompilation ».

Les figures ci-dessous présentent les solutions de carte mère les plus fréquemment utilisées soit le Mini-ITX/Nano-ITX, le PC/104 ainsi que le DilNet PC.

Figure 6: Carte mère de type Nano-ITX

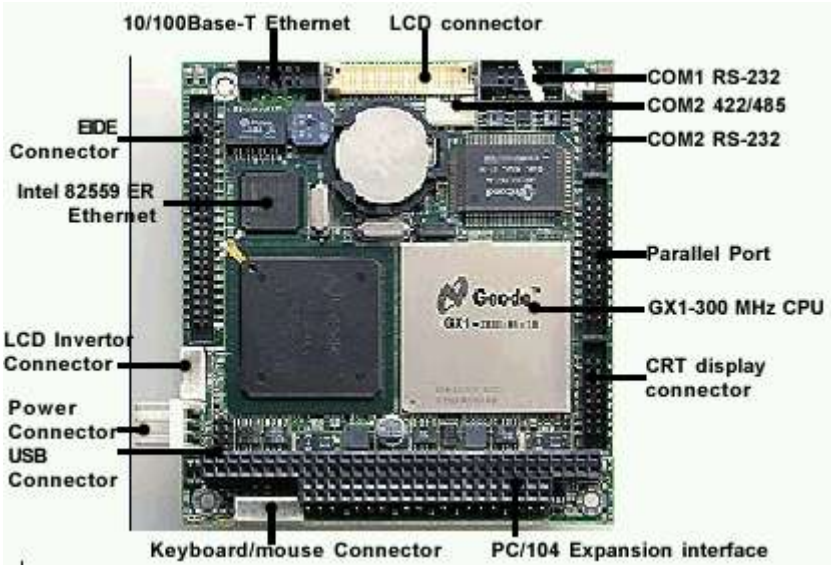

Figure 7: Carte mère de type PC/104

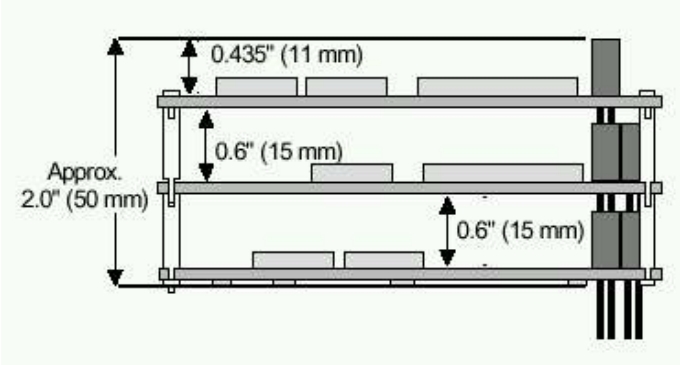

Figure 8: Principe d'empilage des modules PC/104

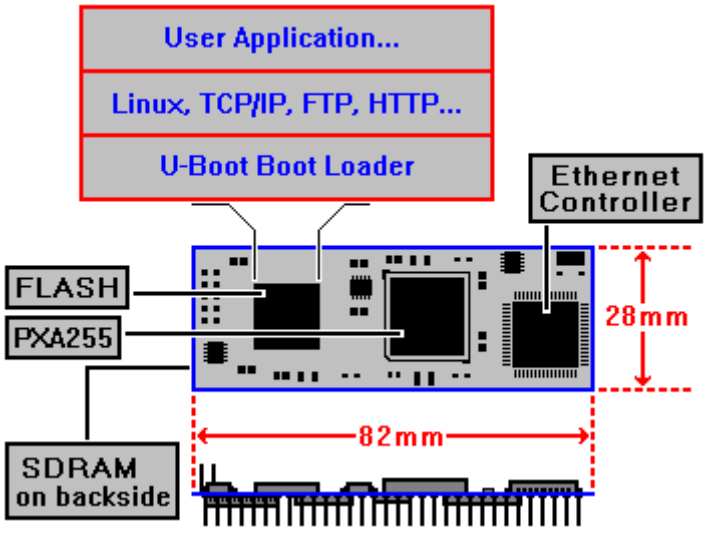

Figure 9: Schéma d'un DilNet PC de chez SSV Embedded

Dans le cas de contraintes incompatibles avec le x86, l'utilisateur pourra se tourner vers des solutions plus intégrées comme l'ARM. Longtemps cantonné aux « petites » applications tournant sur des versions de Linux adaptées aux processeurs sans MMU (µCLinux) et autre micro-contrôleurs, l'ARM est aujourd'hui très utilisé dans les applications tournant sur un « vrai » noyau Linux (avec MMU). Au niveau des fabricants nous pouvons citer les modèles *AT91RM9200* de chez ATMEL ou *XScale* de chez Intel qui bénéficient d'un excellent support Linux. Un autre avantage est la forte intégration: dans le cas du processeur ATMEL, la puce intègre de base un contrôleur Ethernet, un contrôleur USB « host » et un contrôleur USB « device ».

Au niveau des cartes mères disponibles il y a deux approches:

- La carte dite d'*évaluation* est fournie par le fabricant de processeur. Elle fait office de référence , bénéficie normalement d'un très bon support technique (paquetages logiciels, schémas) mais elle est souvent très onéreuse (1500€ ou plus). Le but est d'utiliser cette carte comme base pour la conception d'une carte plus compacte et optimisée, produite en grand nombre. La figure ci-dessous présente la carte d'évaluation du processeur AT91RM9200 fournie par ATMEL.

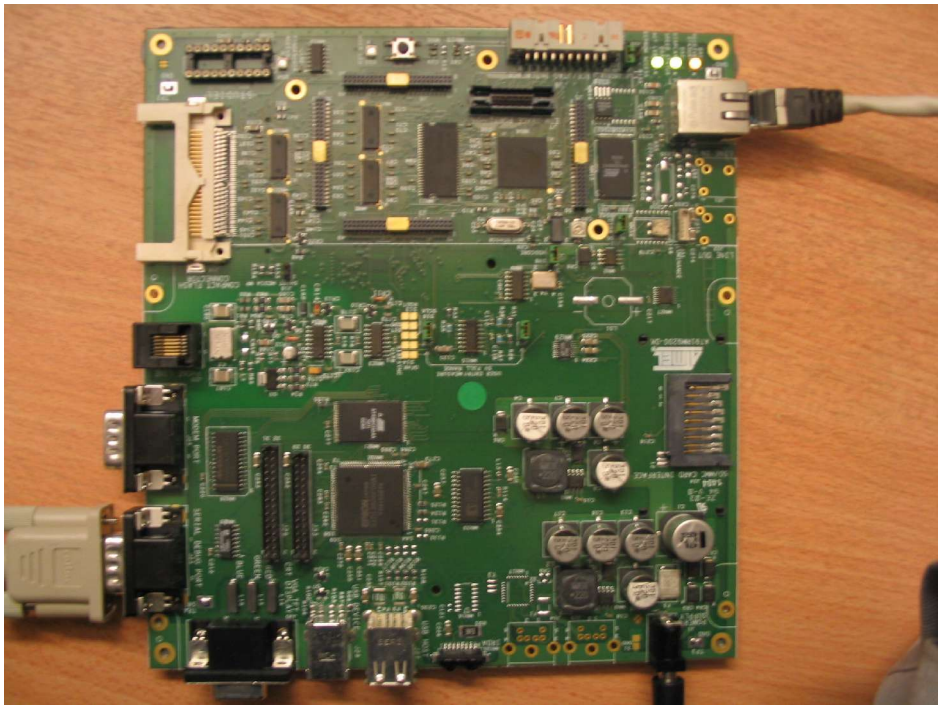

Figure 10: Carte d'évaluation ATMEL

- Le module (carte fille) est une solution utilisable dans le produit final. En phase de développement, il est installé sur une carte mère contenant les interfaces (série, Ethernet, USB, etc.). Le coût d'une telle solution est inférieur car elle est commercialisée comme un produit à part entière (et non une référence). Il conviendra cependant de vérifier la qualité du support Linux: SDK fourni ou non, qualité de la documentation, compétences du fabricant sur la cible Linux, conformité par rapport à la licence GPL (fourniture des sources). La figure suivante présente une solution commercialisée par la société *EUKREA*.

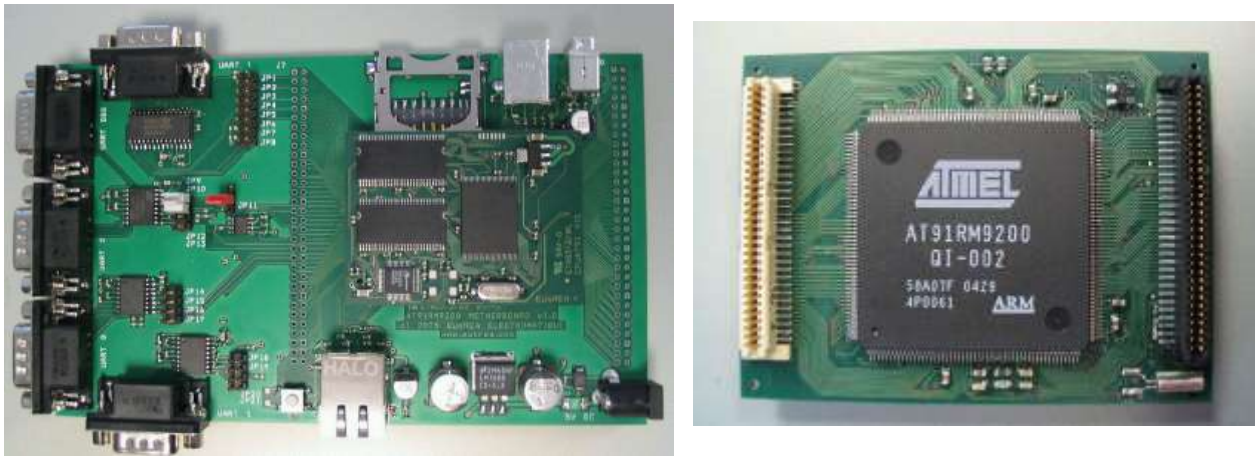

Figure 11: Interface et module ARM AT91RM9200 (EUKREA)

La plate forme *PowerPC* est très présente dans l'environnement embarqué mais elle a tendance à être sérieusement concurrencée par l'ARM dont le rapport prix/performances est souvent meilleur. Cependant des sociétés comme *Freescale* (ex-Motorola) sont toujours très présentes sur le marché avec des processeurs performants comme l'*i.MX21* ou le *MPC5200*. Notons également que l'éditeur *Metrowerks*, fournisseur d'outils de développement comme *CodeWarrior* fait partie du groupe Freescale.

Le site *Linuxdevices.com* dispose d'une liste de fournisseurs de cartes mères spécialisées, nous pouvons entre-autres citer les sociétés suivantes.

- La société EUKREA (France) spécialisée dans les cartes ARM. Une article du co-fondateur d' EUKREA concernant l'architecture ARM est présenté dans ce numéro.
- La société ACME Systems (Italie)
- La société COGENT Corp. (USA)

Les périphériques de stockage

Les cartes mères pré-citées supportent un certain nombre de périphériques de stockage utilisables dans l'environnement Linux. L'article « Embarquez Linux! ou Linux Everywhere » paru dans Linux

Magazine étudie en détails la mise en oeuvre de ces périphériques. Nous nous limiterons ici à un rapide tour d'horizon.

- Les cartes *Compact Flash* sont très répandues dans le monde embarqué soit sous forme de périphériques visibles sur le bus IDE (cas le plus courant) ou sur des bus plus spécifiques comme le SPI ou le PC-Card (PCMCIA). En toute rigueur elle ne sont pas réellement dédiées au stockage d'un système d'exploitation (flash NAND, donc destinée aux données) mais en pratique la solution est viable. Si la flash est vue sur le bus IDE, son utilisation sous Linux est triviale car il n'y a pas de pilote à ajouter.

Figure 12: Carte Compact Flash dans un adaptateur PC-Card

- Les boitiers *Disk On Module* (DOM) permettent d'utiliser le connecteur IDE standard. Le disque dur est alors remplacé par une mémoire flash de la taille du connecteur.

Figure 13: Boitier DOM

- Les boitiers *Disk On Chip* de chez M-Systems sont des mémoires flash « intelligentes » intégrant un « firmware » gérant l'utilisation optimale des blocs de données. Elles nécessitent un pilote spécial disponible sous Linux via le pilote MTD du noyau 2.6 ou bien le pilote propriétaire disponible chez M-Systems.

Figure 14: Boitier Disk On Chip

- Les *clés USB* ont l'avantage d'être très banalisées et donc peu onéreuses. Sous réserve de modification mineure du noyau Linux, la clé USB peut constituer un bon périphérique de stockage.

Les initiatives autour de Linux embarqué

Au début de l' ère *Linuxienne*, utiliser un logiciel libre dans le monde industriel n'était pas chose facile car les motivations et les habitudes des industriels étaient initialement très éloignées de celles du monde du logiciel libre. Si nous mettons de coté l'éditeur Microsoft qui joue sur le volume de ses ventes grand public ou institutionnelles, beaucoup d'éditeurs spécialisés travaillent sur des marchés de niche avec un volume assez faible et sont donc obligés de pratiquer des tarifs élevés tant sur les licences que sur les prestations: support, formation, fourniture de tout ou partie du code source. Ils voyaient donc forcément d'un mauvais l'oeil l'arrivée d'un concurrent « gratuit » et surtout disposant de méthodes de développement classiques garantissant – au grand dam des éditeurs – l'interchangeabilité des prestataires. Outre les problèmes commerciaux, le modèle de développement open source du fait de sa structure décentralisée se prêtait mal au contraintes du monde industriel qui fonctionne autour de normes et et certifications « métier ». Bien que Linux soit basé sur des « normes » informatiques comme POSIX ou les RFC (*Request For Comments*, définissant les protocoles de l'Internet), il est peu envisageable de voir le noyau Linux certifié dans sa globalité pour une norme spécifique à un métier donné (aéronautique, automobile, gestion de processus industriels). Un éditeur spécialisé positionné sur un marché de niche peut investir sur une certification d'autant qu'il est l'unique responsable de l'évolution de son produit. Dans le cas de Linux, qui peut payer ou garantir la conformité des sources du noyau officiel?

L'évolution de cette situation problématique nécessitait l'intervention d'utilisateur de renom, soit les grands groupes industriels. Au tout début de l'histoire (avant 2000), un consortium fut déjà formé autour de Linux, soit ELC pour *Embedded Linux Consortium*. Cependant, les fondateurs du consortium ne constituaient pas réellement un groupe d'utilisateurs mais en majorité de spécialistes du domaine (MontaVista, FSMLabs, IBM, LynuxWorks, Metrowerks, Panasonic, RTI). Certains grands noms de l'électronique (Sony, Alcatel, Motorala, etc.) avaient cependant apporté leur soutien à la création de l'ELC. L'événement et la suite de la montée en puissance du sujet furent fortement relayés par le portail *Linuxdevices.com* toujours très actif aujourd'hui et véritable porte parole de la communauté des utilisateurs et fournisseurs.

En 2003, l'initiative Linux CE ou CELF (*Consumer Electronics Linux Forum*) fut lancée autour de grands noms de l'électronique grand public: Matshushita, Sony, Hitachi, NEC, Philips, Samsung, Sharp, Toshiba. Le but était de créer des composants Linux mieux adaptés aux métiers de ces grands industriels.

Il y a quelques jours (novembre 2005) des grands noms de la téléphonie mobile on annoncé leur intention de « standardiser » Linux pour leurs applications sous le nom de *LiPS*. Parmi eux on compte PalmSource (éditeur du système *PalmOS*), Motorola, Orange, NEC, Panasonic. Le grand

absent est bien entendu Nokia qui a beaucoup investi dans Symbian, l'éditeur d'un système d'exploitation dédié à la téléphonie mobile dont Nokia est actionnaire à 50%, ce qui n'empêche pas Nokia de s'intéresser de près à Linux sur d'autres applications (tablette multimédia). D'ores et déjà, plusieurs *smart-phones* sous Linux existent entre-autres chez Motorola ou des fabricants de *reference design*.

En avance ou en retard ?

La plupart des utilisateurs sont d'accords pour affirmer que Linux est très performant de manière intrinsèque, lorsque qu'il est nécessaire de mettre en place une fonctionnalité donnée, ce grâce à la grande modularité du système et à la richesse des paquetages disponibles.

En contre partie, utiliser Linux et l'open source nécessite un apprentissage car les règles de fonctionnement sont très différentes. Il y a donc un « coût de changement » qui n'est pas à négliger. Même si les communautés de développement sont la pour épauler, il faut apprendre les principes, les langages (au sens propre et figuré), les codes de comportement, les méthodes de recherche de l'information pertinente et souvent se faire aider par des spécialistes afin de gagner du temps. De plus des documentations pédagogiques et *abordables* ne sont pas toujours disponibles avec les composants.

La ou les éditeurs propriétaires proposent de luxueuses documentations, des formations rodées, du marketing, des séminaires et des présentations très efficaces, les utilisateurs de Linux doivent parfois faire l'effort d'aller écouter des développeurs certes compétents, mais à l'aspect « rugueux » et parfois peu pédagogues. Des solutions viables pour Linux embarqué existent d'ores et déjà à la fois au travers d'offres commerciales et de composants libres même si il reste aux développeurs à peaufiner tout cela pour faciliter encore le travail des utilisateurs...et devenir aussi communiquants que Linux lui-même !

Bibliographie

- Page « *Systèmes embarqué* » de Patrice Kadionik sur http://www.enseirb.fr/~kadionik/embedded/embedded.html
- Ouvrage « *Linux embarqué, 2ème édition* » sur http://www.editions-eyrolles.com/Livre/9782212116748/linux-embarque
- Article « *Embarquez Linux! ou Linux Everywhere* » sur http://pficheux.free.fr/articles/lmf/linux_everywhere/linux_everywhere_img_final.pdf
- Etude Linuxdevices « *Embedded Linux market snapshot 2005* » sur http://www.linuxdevices.com/articles/AT4036830962.html
- Société LynuxWorks sur http://www.lynuxworks.com
- Société Montavista sur http://www.mvista.com
- Société Wind River sur http://www.windriver.com
- Projet Eclipse sur http://www.eclipse.org
- Electric Fence sur http://perens.com/FreeSoftware/ElectricFence/
- Valgrind et outils annexes sur http://valgrind.org/ et http://valgrind.org/info/tools.html
- OProfile sur http://oprofile.sourceforge.net
- Linux Trace Toolkit sur http://www.opersys.com/LTT

- KDB sur http://oss.sgi.com/projects/kdb/
- Utilisation de KDB sur http://linuxdevices.com/articles/AT3761062961.html
- KGDB sur http://kgdb.linsyssoft.com/
- Rational Pury sur http://www-306.ibm.com/software/awdtools/purify/unix/
- Insure++ sur http://www.parasoft.com/jsp/home.jsp
- CYGWIN sur http://www.cygwin.com
- Qt/Embedded sur http://www.trolltech.com/products/embedded/
- GTK+ sur http://www.gtk.org/
- GTK on Embedded devices sur http://www.directfb.org/docs/GTK_Embedded/
- Nano-X Window System sur http://www.microwindows.org/
- Projet XENOMAI sur http://www.xenomai.org
- Projet RTAI sur http://www.rati.org
- FSMLbs (RTLinux) sur http://www.fsmlabs.com
- Société Jaluna sur http://www.jaluna.com
- Cartes Mini-ITX sur http://www.mini-itx.com
- Dilnet PC sur http://www.dilnetpc.com/
- Support ARM pour Linux sur http://www.arm.linux.org.uk/
- Support PowerPC pour Linux sur http://www.penguinppc.org/
- Société EUKREA sur http://www.eukrea.com
- Société ACME Systems sur http://www.acmesystems.it/
- Société COGENT Corp. sur http://www.cogentcorp.com
- Société M-Systems sur http://www.m-sys.com
- Embedded Linux Consortium (ELC) sur http://www.embedded-linux.org/
- CE Linux Forum sur http://www.celinuxforum.org/