Linux embarqué

I. Introduction [1]

Un système Linux est composé d'un nombre potentiellement important de composants logiciels provenant de différentes sources :

- Un chargeur de démarrage, généralement U-Boot, chargé d'effectuer une initialisation minimale du matériel, de charger le noyau Linux et de le démarrer.
- Le noyau Linux lui-même, qui implémente des fonctionnalités telles que la gestion des processus, la gestion de la mémoire, le planificateur, les systèmes de fichiers, la pile réseau et bien sûr tous les pilotes de périphériques pour votre plate-forme matérielle.
- Bibliothèques de l'espace utilisateur et applications issues de la communauté open-source : outils de ligne de commande, bibliothèques graphiques, bibliothèques de mise en réseau, bibliothèques cryptographiques, etc.
- Bibliothèques et applications de l'espace utilisateur développées en interne, mettant en œuvre la «logique métier» du système embarqué.

II. Pourquoi Linux est-il adapté à l'embarqué ? [2]

- **Fiabilité** : Linux est réputé pour sa fiabilité et l'on peut affirmer que cette réputation n'est pas usurpée. Par exemple le nombre de plantage inexpliqué est très limité.
- **Faible coût** : Linux est non seulement exempt de royalties mais les outils de développement sont également disponibles sous GPL. Le seul effort financier nécessaire à l'adoption de Linux se situe sur la formation – souvent indispensable – et le support technique.
- **Performances** : Les performances de Linux ne sont plus à prouver. De nombreux tests comparatifs (benchmarks) entre Linux et d'autres systèmes concurrents comme Windows NT ont démontré la supériorité de Linux.
- **Portabilité et adaptabilité** : Linux est aujourd'hui porté sur un très grand nombre de processeurs et d'architectures matérielles (x86, ARM, Power PC, MIPS…), y compris des processeurs de faible puissance, comme le démontre le projet µClinux (http://www.uclinux.org).
- **Ouverture** : De par sa conception Open Source, Linux est ouvert. Ce concept d'ouverture se situe à deux niveaux. Le premier niveau concerne l'interopérabilité de Linux avec d'autres systèmes d'exploitation. Le second niveau d'ouverture concerne l'adoption dans Linux des nouvelles technologies, pour peu que celles-ci constituent des standards ouverts.

III. Les systèmes d'exploitation basés sur Linux

Il existe d'ores et déjà un grand nombre de distributions et composants embarqués basés sur Linux dont la majorité est constituée de projets Open Source. Parmi les quels :

- **MontaVista Linux** : Développé par la société Monta Vista (http://www.mvista.com), MontaVista Linux est le leader des solutions Linux embarqué commerciales. MontaVista est à l'origine des modifications du noyau Linux afin d'améliorer la préemption de ce dernier et donc les fonctionnalités de temps réel « mou ».
- **BlueCat Linux** : Ce produit est édité par LynuxWorks, créateur et éditeur du système temps réel LynxOS. La nouvelle version BlueCat Linux 5.0 est basée sur le noyau 2.6 et profite donc de la fonction de noyau préemptif de ce dernier.

- **µClinux** : µClinux est un portage du noyau Linux pour micro-contrôleurs et processeurs dépourvus de MMU (Memory Management Unit). De ce fait, il n'y a pas de protection mémoire d'un programme à l'autre et un programme peut planter un autre programme ou le noyau lui-même. Très ouvert, le système µClinux est disponible pour un grand nombre de processeurs comme le ColdFire Motorola, la famille 68xxx, ARM, Intel i960, Axis ETRAX. Les nouvelles versions du noyau 2.6 sont très rapidement intégrées dans le projet.
- **RTLinux** : RTLinux est un projet Open Source qui a pour but d'ajouter à un noyau Linux standard un noyau temps réel préemptif et à priorités fixes. Ce petit noyau temps réel, chargeable dynamiquement dans le système, considère le noyau Linux comme la tâche de plus faible priorité.

IV. Quelques exemples de produits utilisant Linux [2]

Linux est présent dans divers secteurs de l'industrie grand public. On peut citer :

- les assistants personnels ou PDA (Personal Digital Assistant) ;
- les consoles multimédias et tablettes Internet (set top boxes et web pads) ;
- les magnétoscopes numériques.

Dans des domaines plus professionnels, on peut citer :

- les routeurs ;
- les équipements de téléphonie ;
- les caméras IP.

V. Structure de Linux [2]

À quelques exceptions près, la structure du système est calquée sur les autres systèmes Unix libres ou propriétaires à savoir :

- un noyau ou kernel réalisant les fonctions essentielles comme la gestion des tâches et de la mémoire, ainsi que l'interfaçage entre le matériel et les applicatifs grâce entre autres aux pilotes de périphériques ou en anglais device drivers ;
- une bibliothèque principale appelée libc ou glibc (GNU libc) sous Linux et contenant les fonctions de base utilisées par les applicatifs ;
- les applicatifs eux-mêmes, appelés également commandes, soit livrés en standard avec le système ou bien développés pour des besoins spécifiques.

À ces éléments standards il faut ajouter un programme de démarrage ou bootstrap comme LILO (LInux LOader). À la différence des autres composants du système, le programme de démarrage ou chargeur est très dépendant du matériel utilisé. Le schéma de démarrage d'un système Linux est relativement simple et on peut le décomposer comme suit :

1- Chargement du système par LILO ou un programme équivalent, comme GRUB.
2- Chargement du noyau Linux. Le chargement s'accompagne de l'initialisation des périphériques matériels indispensables au démarrage, et donc au chargement des pilotes de périphériques associés. Le noyau Linux tente également de charger sa partition principale ou root partition sur laquelle il ira chercher les éléments nécessaires à la suite du lancement du système.
3- Lorsque le noyau Linux est chargé, il exécute un programme d'initialisation qui, par défaut, correspond à la commande /sbin/init. Cependant, on peut indiquer facilement au noyau de passer la main à un autre programme.
4- Dans le cas de l'utilisation du programme init standard, ce dernier explore le fichier de configuration /etc/inittab qui contient le chemin d'accès à un script de démarrage.

VI. Le noyau Linux [2]

Le noyau est l'élément principal du système, et ce pour plusieurs raisons. La première raison est historique puisque ce noyau fut initialement conçu par Linus Torvalds, créateur du système Linux, le reste du système étant constitué de composants provenant en majorité du projet GNU de Richard Stallman. L'autre raison est technique et tient en l'occurrence à la structure monolithique du noyau qui en fait l'interface unique entre le système et le matériel dans la quasi-totalité des cas de figure. [2]

Le noyau est le cœur du système, c'est lui qui s'occupe de fournir aux logiciels une interface de programmation pour utiliser le matériel. Le noyau Linux a été créé en 1991 par Linus Torvalds pour les compatibles PC. Initialement conçu pour l'architecture de processeur x86, il a ensuite été porté sur de nombreuses autres, dont m68k, PowerPC, ARM, SPARC, MIPS et Architecture RISC-V. Il s'utilise dans une très large gamme de matériel, des systèmes embarqués aux superordinateurs, en passant par les téléphones mobiles et ordinateurs personnels. [3]

Ses caractéristiques principales sont d'être multitâche et multi-utilisateur. Il respecte les normes POSIX ce qui en fait un digne héritier des systèmes UNIX. Au départ, le noyau a été conçu pour être monolithique. Ce choix technique fut l'occasion de débats enflammés entre Andrew S. Tanenbaum, professeur à l'université libre d'Amsterdam qui avait développé Minix, et Linus Torvalds. Andrew Tanenbaum arguant que les noyaux modernes se devaient d'être des micro-noyaux et Linus répondant que les performances des micro-noyaux n'étaient pas bonnes. Depuis sa version 2.0, le noyau, bien que n'étant pas un micro-noyau, est modulaire, c'est-à-dire que certaines fonctionnalités peuvent être ajoutées ou enlevées du noyau à la volée (en cours d'utilisation). [3]

VII. Solutions pour utiliser Linux embarqué [1]

Afin d'assembler un système Linux avec tous ces composants logiciels, on a généralement trois solutions :

- Utiliser une distribution binaire, comme Debian , Ubuntu ou Fedora . Plusieurs de ces distributions prennent en charge l'architecture ARMv7. Le principal avantage de cette solution est qu'elle est simple : ces distributions binaires sont familières à la plupart des utilisateurs de Linux, elles ont un système de gestion de paquets agréable et facile à utiliser, tous les paquets sont pré-compilés donc il est très rapide de générer un Système Linux. Cependant, les systèmes Linux générés de cette manière sont généralement difficiles à personnaliser (les composants logiciels sont pré-construits, vous ne pouvez donc pas facilement adapter leur configuration à vos besoins) et difficiles à optimiser (en termes d'encombrement ou de temps de démarrage).
- Construire un système Linux entier **"à la main"** en téléchargeant les sources des applications, des librairies, du noyau... et gérer toutes les **dépendances**, ce qui revient à créer un « **Linux From Scratch ».**
- Utiliser un système de construction, comme Buildroot ou OpenEmbedded . Ces systèmes de construction construisent un système Linux entier à partir du code source, ce qui signifie qu'il peut être hautement personnalisé et optimisé en fonction de vos besoins. Bien sûr, c'est moins simple que d'utiliser une distribution binaire et comme vous construisez tous les composants à partir du code source, une quantité non négligeable de temps CPU sera consacrée à la compilation du code.

VIII. Chaîne de compilation croisée

Une chaîne de compilation (en anglais : « toolchain ») désigne l'ensemble des paquets utilisés dans le processus de compilation d'un programme, pour un processeur donné. Le compilateur n'est qu'un élément de cette chaîne, laquelle varie selon l'architecture matérielle cible.

La chaîne de compilation GNU (en anglais : « GNU toolchain ») utilisée dans le monde du logiciel libre comprend les éléments suivants :

- **binutils** : une collection d'utilitaires utilisés pour le traitement des fichiers binaires ; ce paquetage contient notamment l'assembleur qui transforme le pseudo-code généré par la compilation en instructions comprises par le processeur cible, et l'éditeur des liens pour lier les bibliothèques utilisées par un programme.
- **gcc** : les compilateurs C et C++ de la collection ;
- **glibc**, bibliothèque C du système utilisée pour les appels au noyau et le traitement des processus de bas-niveau ;
- **les en-têtes du noyau** requis par la bibliothèque glibc (linux-headers dans le cas du noyau Linux) ;
- **gdb**, composant optionnel pour le débogage d'une chaîne binaire déjà compilée ;

On parle de **compilation native** lorsque la chaîne est compilée sur une machine à la fois hôte et cible. Les éléments de la chaîne sont installés dans des paquetages pour les distributions dites binaires (par exemple, deb pour Debian, rpms pour Red Hat), ou sont compilés à partir des sources pour les distributions sources. La chaîne de compilation génère des exécutables sur la même architecture matérielle pour laquelle elle a été compilée.

Une chaîne de compilation croisée est une chaîne compilée pour fonctionner sur l'architecture de processeurs de la machine hôte, mais qui va compiler des logiciels pour une architecture cible différente. Dans ce cas, il est nécessaire de compiler la chaîne à partir du code source.

Les composants de la chaîne en environnement GNU/Linux utilisent les mécanismes de construction du système GNU. Le script configure permet de préciser les architectures hôte et cible (paramètres host et target). Les distributions peuvent fournir une couche d'abstraction pour en simplifier la gestion (ebuild pour Gentoo, Cookutils pour SliTaz2), ou des outils comme Crosstool, Buildroot pour les systèmes embarqués.

Références

[1] https://bootlin.com/blog/building-a-linux-system-for-the-stm32mp1-basic-system/

[2] Linux embarqué - 2ème Edition, Pièrre Ficheux

[3] https://fr.wikipedia.org/wiki/Noyau_Linux

[4] https://fr.wikipedia.org/wiki/Cha%C3%AEne_de_compilation