Rapport final du TIPE:

Linux Embarqué et Système Embarqué

Etudiant : Pham Viet Hung, Promo X-IFI

pvhung@ifi.edu.vn

Hanoï, 15 Juillet 2005

Préface

Ce rapport présente mon travail de TIPE à l'Institut de la Francophonie pour l'Informatique (IFI) pendant deux semestres. J'ai du faire une recherche sur un sujet d'une manière professionnelle telle que la formation mastère avec mes intérêts. Ce document n'est qu'une introduction pour ceux qui s'intéresse parce que le domaine d'application est très vaste et la période de travail relativement courte.

Le but de ce travail est d'abord d'étudier les sujets embarqués en général. Ensuite, on s'intéresse à l'utilisateur de Linux pour construire un système embarqué. Ce rapport aborde donc des caractéristiques de systèmes embarqués et se décrit des méthodes de base, techniques et étapes nécessaires pour la construction d'un système embarqué basé sur le noyau LINUX. Ceci a permis la réalisation d'un petit système Linux Embarqué.

Remerciement

Je remercie le professeur qui est responsable de mon TIPE, Monsieur Nguyen Hong Quang pour sa disponibilité durant deux semestres. J'adresse également mes remerciements à d'autres professeurs de l'IFI qui me donnent des conseils.

Table des matières

Table des figures

Chapitre 1 : Introduction

Les progrès techniques de ces dernières années ont permis de faciliter l'utilisation de systèmes à microprocesseurs. Ils font parti de notre vie de tous les jours. Grâce à eux, l'électronique est de plus en plus présente de varié forme: Les téléphones portables apparaissent par tout. Ces machines à microprocesseurs sont utilisées dans le monde de l'industrie afin de réguler un processus chimique ou d'automatiser une chaîne de montage…etc. Même les machines utilisées à la maison ont également les microprocesseurs. Les exemples concrets pourraient remplir plusieurs pages. Mais, Un point commun : plus petit, plus puissant et moins cher. Sur ce point, l'informatique puisera l'électronique en remplaçant des systèmes matériels par solution logicielle.

Comme on la sait certainement, une solution logicielle a forcément besoin d'un composant logiciel essentiel: C'est un système d'exploitation. Aujourd'hui, Linux considère comme un OS alternative d'OS propriété, dans l'informatique générale, grâce à sa stabilité et sa gratuité, Malgré son jeune age dans le monde embarqué, mais linux embarqué a montre sa puissance, et sa disponibilité pour la construction des système embarqué.

1.1 Définitions

Système Embarqué

En effet, le système embarqué s'applique dans de nombreux domaine. Difficile à définir ce système de manière précise. Mais quelques définitions sont extrait du livre et des articles, qui nous aident à le comprendre:

1. Un Système Embarqué : C'est une combinaison de matériels et logiciels permettant de remplir une ou plusieurs fonctions spécifiques avec des contraintes plus ou moins sévères tel que la consommation, la température, la taille, les performances... et. [Patrice Kadionik, 2004]

2. Il y a beaucoup de caractéristiques des systèmes embarqués, cela dépende du but d'utilisation d'un système. Alors, on peut définir par des caractéristiques communes.

Généralement, Un système embarqué :

- Dispose de ressources limitées.

- Ne possède pas toujours de système de fichiers.
- Doit être le moins cher possible
- Ne doit pas consommer d'énergie inutilement
- Exécute un logiciel dédié aux fonctionnalités spéciales

3. Un système embarqué est susceptible d'être utilisé dans un environnement matériel de faibles performances (si l'on compare au PC de bureau d'aujourd'hui). Si l'ajout de quelques Mo de mémoire sur un PC de bureau n'a pas une grosse influence sur le budget d'un utilisateur, le gain de quelques Mo sur un produit de grande consommation (téléphone, équipement auto mobile, organiseur personnel) a une énorme influence sur le coût final [Pierre Ficheux, 2003]

Qu'est ce que Linux ?

Linux est un système d'exploitation libre de type UNIX lancé par le finlandais Linus Torvalds en 1991 avec l'assistance de milliers de développeurs dans le monde pour son évolution. Son succès tient au fait qu'il est développé sous licence GPL[1], ce qui signifie que le code source Linux est disponible à tous le monde et gratuit.

- Linux est stable et robuste.
- Linux tourne originellement sur plateforme i386 et supérieure avec 8 Mode RAM.
- Linux est complète des outils/logiciels GNU.
- Linux est disponible sous forme de distributions.
- Linux est utilisé avec une interface graphique comparable à Microsoft Windows : Gnome, KDE.

Linux Embarqué

A partir de ces deux définitions, On donne une définition sur Linux embarqué: C'est une adaptation du noyau Linux à un système embarqué. Suivant les capacités du système, on ne retrouve qu'une partie des fonctionnalités du noyau:

- Moins de services disponibles en général
- Moins de mémoire requise (<8Mo).
- Boot depuis une mémoire ROM.
- Pas de clavier ou de souris requis.

[1] GPL : General Public License

- Logiciels spéciaux pour piloter les périphériques du système Les dispositifs peut être souvent l'écran LCD, Flashdisk, DiskOnChip, et touchscreen....

1.2 Panorama du marché de l'embarqué et le futur de Linux Embarqué

Tendances à l'embarqué

Le champ d'application des systèmes embarqués est très vaste. Le fait est d'ailleurs de plus en plus large. Car, tout d'abord, beaucoup de fonctions autrefois réalisé par des systèmes mécaniques ou analogiques sont aujourd'hui remplacées par des composants électroniques pilotés par des logiciels.

De plus, grâce à la diffusion des téléphones portables, des assistants personnels de poche (PDA : Pocket Digital Assistant) et baladeurs MP3. L'embarqué paraît aujourd'hui indispensable dans le domaine portable. Les jeux en mobile sont téléchargés par des millions d'utilisateurs, on peut surfer sur le net, ou consulter des mails et utiliser des logiciels de comptabilité et de lecture multimédia. On peut dire que la mis d'un système embarqué facilite à réaliser ces services.

Linux embarqué devient la principale plateforme de l'embarqué

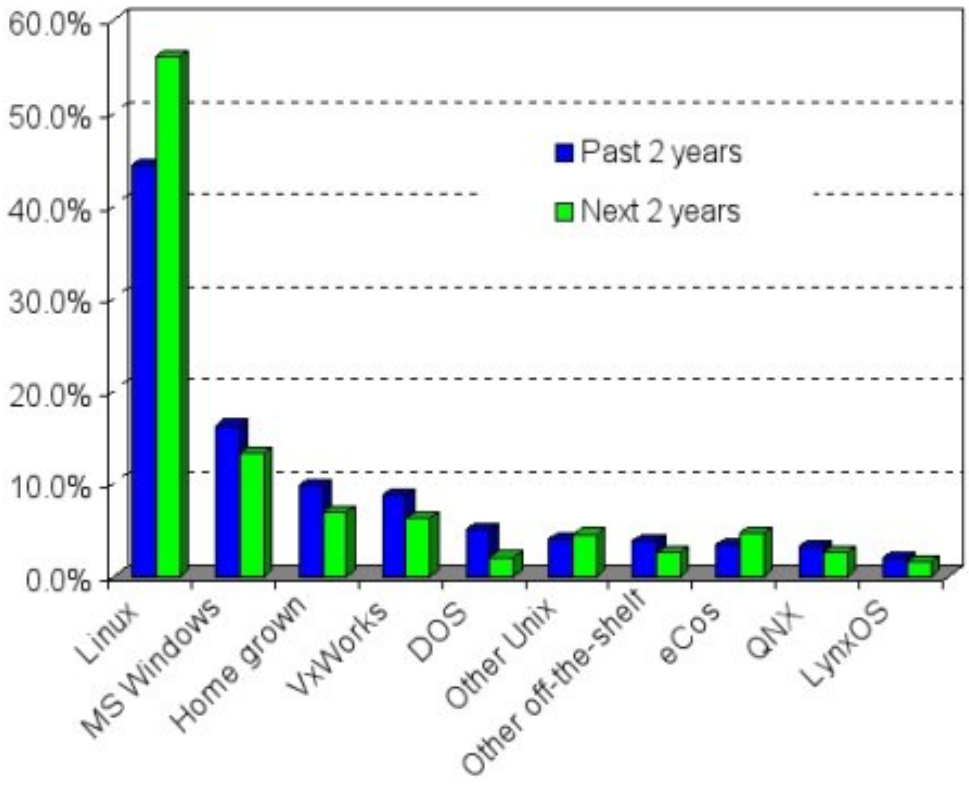


Figure 1 : Le développement de OSs Embarqués, En 2004

(http://www.linuxdevices.com/articles/AT4036830962.html)

Selon VDC (www.vdc.com), qui a étudié sur des plateformes de matériel, des logiciels, Linux, Java, et logiciel ouvert de source, des types d'applications, et des outils de développement. Il conclut que Linux Embraqué se développe très rapidement. On utilise de plus en plus linux pour la stratégie de développement de ses produits. Et Linux embarqué, aujourd'hui, est un premier choix.

1.3 Le but de recherche

Ce sujet est très large. Parce que un système embarque concerne le matériel et le logiciel. Lors de la conception, tout d'abord, on conçoit les composants matériels, ensuite, construire les composants logiciels basant sur les matériels du système. Le temps consacré à ma recherche est limité, donc je me concentre aux solutions logicielles sur Linux et les choses concernées. Ces objectifs en détaille comprend :

- Rappel des parties matérielles qui sont souvent utilisé dans l'embarqué
- Maîtrise du système Linux dans le cadre au sujet de OS Embarqué
- Recherche sur quelques solutions logicielles disponibles
- Présentation des quelques ressources très actives
- Utilisation d'outils disponible pour Embarquer Linux.

Enfin, Je recherche sur une étude de cas: «uClinux », une distribution Linux dédiée aux systèmes embarqués.

Chapitre 2 : Système Embarqué et ses applications

2.1 Types de système embarqué

On utilise un système embarque dans le domaine traditionnel tel que l'industrie aérospatial, automobile, électronique, télécommunication, et portable…etc. Mais cela fournit un peu d'informations ce qui concerne comment un système serait conçu. Il est donc difficile de classifier précisément le système embarqué. Pour bien classifier, on présente quelques critères que peuvent fournir l'information actuelle sur la structure du système : la taille, contrainte de temps, capacité du réseau, et degré de l'interaction d'utilisateurs [Karim Yaghmour 2003].

La taille

La taille d'un système embarqué est déterminé par la taille d'attribut de composant électronique du système, comme le puissance de calcul, la capacité de la mémoire (RAM) et la taille de stockage permanent. En terme de la taille, On a trois grandes catégories de système : petite, moyenne, grande taille.

La contrainte de temps

Il y a deux types de contraintes de temps pour les systèmes embarque : douce ou dure.

- Une contrainte **douce** (système temps réel doux) est moins contraignante qu'elle permet une erreur raisonnable par rapport au moment exact où le processus aurait dû s'exécuter.

- Par opposition, la contrainte **dure** ne permet aucune erreur sur le moment où le processus aurait dû s'exécuter. Sinon, c'est la catastrophe!

La capacité du réseau

La capacité de réseaux définit si un système peut être relié à un réseau. Maintenant, nous voulons tous les dispositifs soit accessible par le réseau. Donc c'est un facteur pour que on choisisse Linux comme OS embarqué à grâce de ses possibilités de gestion de réseau.

Il y a 2 technologies de connexion qui sont supportés par système embarqué : le Bluetooth, et la carte de San-fil.

L'interaction d'utilisateurs

Le degré d'interaction d'utilisateur est considérablement un critère pour classifier un système à l'autre. Le degré d'interaction d'utilisateur change considérablement d'un système à l'autre. Quelques systèmes, tels que PDAs, sont concentré sur l'interaction d'utilisateur, alors que d'autres, tel que des systèmes de commande de processus industriel, pourraient seulement avoir des LED et des boutons pour l'interaction. Quelques autres systèmes, n'ont aucune interface utilisateur quelque. Par exemple, quelques composants d'un système de pilote automatique dans domaine de contrôle de l'avion.

2.2 Exemples d'applications

(Selon les articles sur www.linuxdevices.com)

Calcul normal (limite de matériels)

- Application similaire à une application de bureau mais empaquetée dans un système embarqué.
- Les jeux de vidéo, set- top box, et TV Box.

Les systèmes de contrôle

- Contrôle de systèmes en Temps Réel.
- Moteur d'automobile, traitement chimique, traitement nucléaire, système de navigation aérien.

Traitement de signal

- Calcul sur de grosses quantités de données.
- Le radar et sonar, le dispositif de compression vidéo.

Télécommunications & Réseau

- Transmission d'information et commutation.
- Téléphone portable, Dispositifs de l'Internet.

Quelques OS Embarqué très connu :

Palm OS pour le *handheld*
Ses applications comprends
- Gestion de la mémoire simplifiée,
- Gestion de bases de données et de l'écran,
- Fournissement de la bibliothèque mathématique
- Fournissement des applications minimalistes.

Symbian OS pour la téléphonie,
- Gestion des contacts,
- gestion de réseaux divers(SMS,Blue-Tooth,GSM,TCP/IP),
- gestion multimedia
- Supporte la technologie Java (JavaPhone)

Windows Embarqué :
- Les applications dérivée du Windows

2.3 Caractéristiques d'un système embarqué

Il y a cinq caractéristiques générales du système embarqué :

Faible coût

Un assez grand nombre de produits d'embarqué sont sur les marchés où l'utilisateur ne veut pas payer le supplémentaire pour la performance ou fonctionnalité de plus, Les concepteurs ont donc dû concevoir avec des rapports optimaux entre le prix et la performance Le résultat de ceci est que les ressources disponibles sont minimaux possibles. C'est pour quoi un système embarqué a rarement plus de quelques Mega octets de mémoire disponible.

Faible consommation

La minimisation de la consommation est essentielle pour les systèmes autonomes afin de maximiser l'autonomie des batteries. Une consommation excessive augmente le prix de revient du système embarqué car il faut alors des batteries de forte capacité.

Faible encombrement et faible poids

Ils doivent cohabiter sur une faible surface électronique analogique, électronique numérique. Notamment, c'est très important pour les applications portables où l'on doit minimiser la taille et les poids.

Fonctionnement en Temps Réel (Réponse de temps)

Les applications embarquées, comme des applications de système de contrôle, sont événement conduit et doivent répondre rapidement à ces événements.

Peut être, Un système embarque a besoin des opérations de calcul doivent être faites en réponse à un événement extérieur. C'est une caractéristique pour quelques domaines spéciaux.

Environnement

Ils sont soumis à de nombreuses contraintes dictées par l'environnement telles que la température, l'humidité, les vibrations, les chocs, les variations d'alimentation, les interférences RF, la corrosion, l'eau, le feu, les radiations…etc.

2.4 Architecture générale (composants possibles)

Comme vous savez l'architecture de système normal, un ordinateur se compose de trois couches: Application, Système d'exploitation, et Matériel. De même, un système embarque dispose de 3 couches. Chaque couche a la même fonctionnalité qu'un système normal. Mais, Il y a des différences de sous composants du système. Deux premières couches, Il s'agit du logiciel. Il est possible de modifier ces composants logiciels, et aussi ajouter ou supprimer ses modules au besoin dépendant du but de système. Cependant, Ce n'est pas un système qui contient tous les composants comme le système complet. Car le but de conception est de servir quelques tâches spécifiques, et de concentrer à un unique travail. Le système d'exploitation est une couche logicielle sur laquelle on va se placer l'ensemble des applications lancées par les utilisateurs. Il comprend les librairies pour le développement, des drivers permettant aux applications d'accéder à des périphériques évolués, peut-être les interfaces pour contrôler les éléments. La conception du OS se base forcément sur le matériel visant à optimiser la performance… etc.

La dernière couche, Il s'agit du matériel. C'est-à-dire qu'un ensemble des éléments physiques employés pour le traitement de données. Les composants matériels sont limités et peut-être il y aurait un composant dédié aux tâches spécifiques.

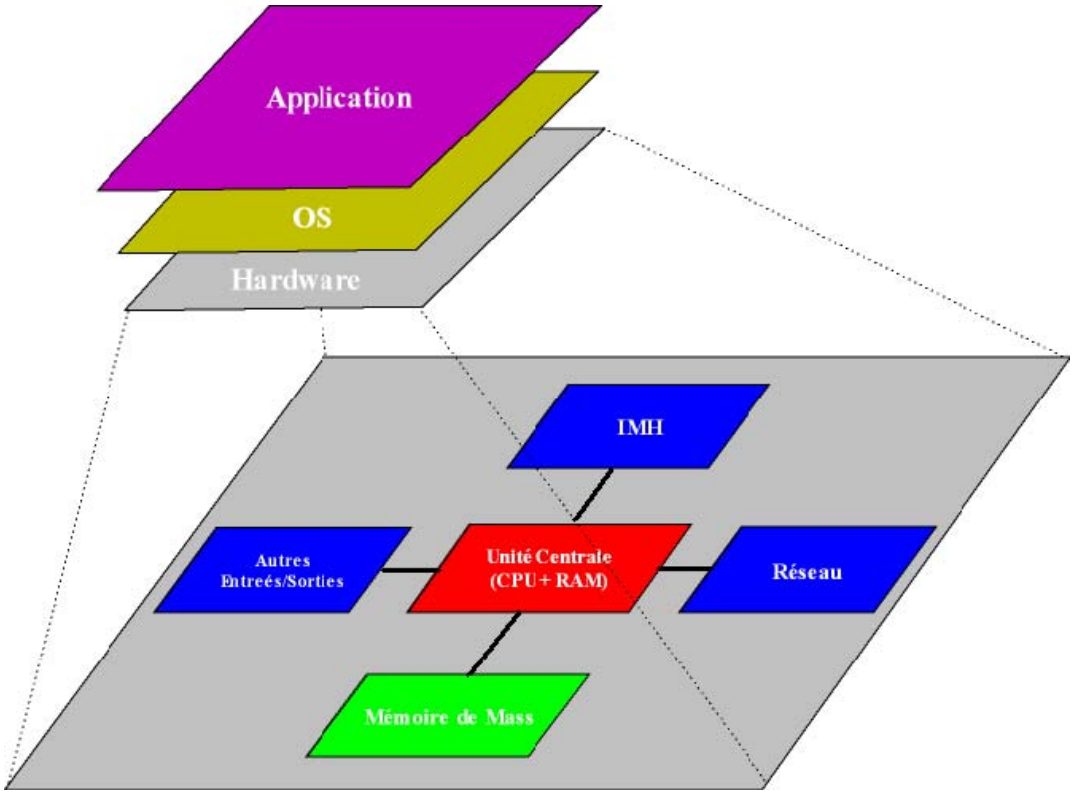


Figure 2: Topologie d'un Système Embarqué

Dans le matériel, l'exception des composants nécessaires qui s'appellent les composants permanent comme CPU, la mémoire vivre RAM,.On peut équiper plusieurs composants supplémentaires. Ca peut être la carte de dédie aux applications spéciale, la mémoire de Mass (DiskOnChip, FlashDisk, CDROM) le réseau, les entrées ou sorties spéciaux. Ci-dessous est quelques dispositifs:

Entrées:

- Les capteurs/convertisseurs (pression, audio, température...)
- Le clavier, boutons poussoirs ou télécommandes (infrarouge, bluetooth, radio...)
- Le lecteurs de tags ou de codes barres.

Sorties:

- L'écrans et afficheurs LCD.
- Le Système d'alarme ou synthèse vocale.
- L'imprimante en tous genres comme papier, étiquettes, photos...

Mémoire de mass : (La détaille dans la section $4.3.1)

- Le disque dur Ex: microdrive à la taille environ 2,5-3,5 inches.

- La mémoire flash Ex: FlashDisk, CompactDisk, DiskOnChip, SDCard, Memoirestick, clésUSB.
- L'utilisation de ROM Ex: Disque virtuel CD, DVD, disquette.
- Le disque à distance Ex: NFS, TFTP.

IHM:

Normalement, Il est un dispositif qui sert à communiquer entre l'humain et la machine. Un exemple réaliste de IHM est l'ecran avec les dispositifs «TouchScreen» qui est visé aux *PDAs* ou *Handheld.*

Chapitre 3 : LINUX comme OS Embarqué

3.1 Pourquoi Linux ?

1. Prix

Tout d'abord, La raison économique est souvent tenue compte, et puis on n'a pas peur de dire le prix. Linux est gratuit et sans royalties à payer pour chaque produit vendu à base de Linux, ce qui est important pour des systèmes destinés à être fabriqués en grande série, et où chaque coût supplémentaire doit être évité. Toutes les distributions Linux sont disponibles gratuitement au téléchargement par Internet, D'ailleurs, les outils de développement d'un Linux comme le compilateurs, le IDE, la bibliothèque…etc sont disponibles à faible coût ou gratuits parce que fonctionner en terme de GNU. Donc les coûts de mise en œuvre de Linux sont réduits. Un avantage de prix par rapport aux autres OS.

2. "Source code" disponible et gratuit

Linux est un logiciel libre, Donc le code source est disponible au public. Cela donne le pouvoir aux utilisateurs d'utiliser ce logiciel comme ils l'entendent. On peut ainsi voir directement à travers les fichiers sources ce que fait le noyau Linux voire modifier son comportement au besoin. C'est très bien pour la construction d'un Linux embarqué.

3. Beaucoup de pilotages et outils de développement disponibles

Les pilots sont importants dans une situation où les composants électroniques évoluent rapidement, les constructeurs n'ont pas le temps de fournir les pilotes logiciels pour tous les systèmes d'exploitation existants. Maintenant, Linux offre l'avantage nombreux pilotes existent souvent en modèle "OpenSource", et sont donc facilement modifiables ou adaptables, si le besoin s'en faisait sentir. Pour développer une application, Il est nécessaire d'avoir les outils comme le compilateurs, IDE, et le déboguer. Linux supporte le GCC, GDB four déboguer et beaucoup de IDE. Pour le développement de OS Embarqué, la chaîne de développement qui s'appelle "cross-compilation" en anglais (compilation pour une machine cible différente de la machine qui sert à générer un programme) est très importante. La communauté de Linux Embarqué fournit suffisamment les outils de développement nécessaires (voir la détaille $4.1.1).

4. Plein de d'application

L'application étant supporté par LINUX est beaucoup diverse, On peut chercher une application adaptée pour votre domaine. D'ailler, l'utilisation est aussi gratuite.

5. Support de réseau

La capacité de réseau sera indispensable pour la connectivité IP dans l'embarqué. Linux le très bon support en réseaux. Il supporte le plus nombreux de protocoles possibles de réseau testés été prouvés depuis des années comme suit. Cela permet a votre système d'être interopérabilité.

- TCP/IP réseau.
- Routage / Firewalling.
- Serveur de Telnet
- SMB, NFS.
- Protocoles WAN:X.25,AX.25,HDLC et ATM.

6. Fiable et robuste

La fiabilité de Linux peut être pour le système configuration minimale ou bien celle de très grand (128KbdeROMet512KbdeRAM)-(2MbdeROM, et 4Gb de RAM avec mutiprocesseurs) Donc la taille du noyau est modeste compatible avec les tailles de mémoires utilisées dans un système embarqué(<500Ko). On peut aussi trouver une application correspond à son besoin.

D'ailleurs, Il y a quelques inconvénients. Linux n'est pas adapté pour les systèmes de quelques dizaines de kilo octets par exemple: l'électroménager "classique", HiFi, télécommande. Parce qu'ils ont besoin de la taille très très petite tandis que Linux n'a que le minimum de128Kb.

3.2 Architecture matériel supportée par Linux

Les composants matériels sont importants pour la conception de logiciels. Bien que je me concentre aux solutions logicielles, je survole des architectures de processeurs ce qui concerne le système Linux. Linux fonctionne sur un grand nombre d'architecture, mais non toutes les architectures sont utilisées dans le linux embarqué. Cette section va aborder les architectures de processeur assez connues et souvent employé le mode de l'embarqué et les supports par Linux de chaque architecture.

X86

Intel est toujours le plus grand distributeur des processeurs de cette famille et quelques constructeurs aussi donnent leurs produits de famillex86. Elle est la plus populaire, linux supporte alors complètement l'architecture de x86. Dans le monde de Linux, il y avait plein de distributions supportant x86. C'est un grand avantage pour la décision du choix d'une plateforme. Malgré le grand support, il représente une petite partie du marché de systèmes embarqués. Dans la plupart des cas, les concepteurs préfèrent les processeurs ARM, MIPS, et PowerPC à i386 en raisons de la complexité et du coût.

ARM

Actuellement, le ARM sont fabriqués par Intel, Toshiba, Samsung, et beaucoup d'autres. L'architecture de ARM est très populaire dans plusieurs domaines d'application et il y a des centaines producteurs fournissant des produits et des supportes de Linux. il est fiable pour l'embarqué, car son ensemble d'instructions est assez petite Maintenant, Linux soutient environ 10 différent ARMs, et 16 flatteries différentes. Vous pouvez consulter la liste de ARM systèmes supporté sur le site Web http://www.arm.linux.org.uk/developer/machines/. Un produit de l'Intel qui est très utilisé dans le PDA et téléphone portable. C'est que le StrongARM. Il a le meilleur supporte de documents, de driver et de noyau. Selon la statistique de VDC en 2004, L'architecture ARM est le plus utilisé dans l'embarqué. Notamment, il est souvent pour les petits dispositifs comme microcontrôle, PDA, portable… etc.

MIPS : Microprocessor without Interlocked Pipeline Stages

MIPS est une architecture de microprocesseur de RISC développée par MIPS Computer Systems. Il y a 2 types de MIPS. Le puce à 32 bits de MIPS son fournis par les producteurs principaux : IDT, Toshiba et la société LSI. Et une autre de 64bits MIPS par ceux de IDT, LSI, NEC, QED, SandCraft, et Toshiba. Tous les deux sont déjà portés en Linux Embarque par Debian et RedHat. Un exemple d'applications de MIPS, On peut le trouver dans Sony Play StationI/II (très conne), Cisco routeurs et également dans presque beaucoup de PDA, et systèmes embarqués de la petite taille. Pour plus d'information concernant le portage de MIPS de Linux en général, accédez sur http://www.linux-mips.org/.

PowerPC

Le PowerPC est très connu par son utilisation dans la machine Apple, et il est également développé par IBM dans la famille des serveurs. Le système embarqué basé Power PC est paru dans quelques applications comme le système. Cependant, Il peut être employé pour le linux embarqué de grande taille TiVo. Le site Web de PPCLinux sur http://penguinppc.org/ contient la documentation et les liens. Il y a également autre site http://www.linuxppc.org/, qui aborde les distributions de LinuxPPC. C'est comme l'architecturex86, un peu appliqué dans l'embarqué.

Les autres:

HitachiSuperH, Motorola, Intel 960, et ColdFire ..

3.3 Architecture logicielle d'un Linux Embarqué

A partir de ces caractéristiques et l'architecture du LINUX en général, je donne un exemple sur l'architecture logicielle d'un Linux Embarque pour décrire

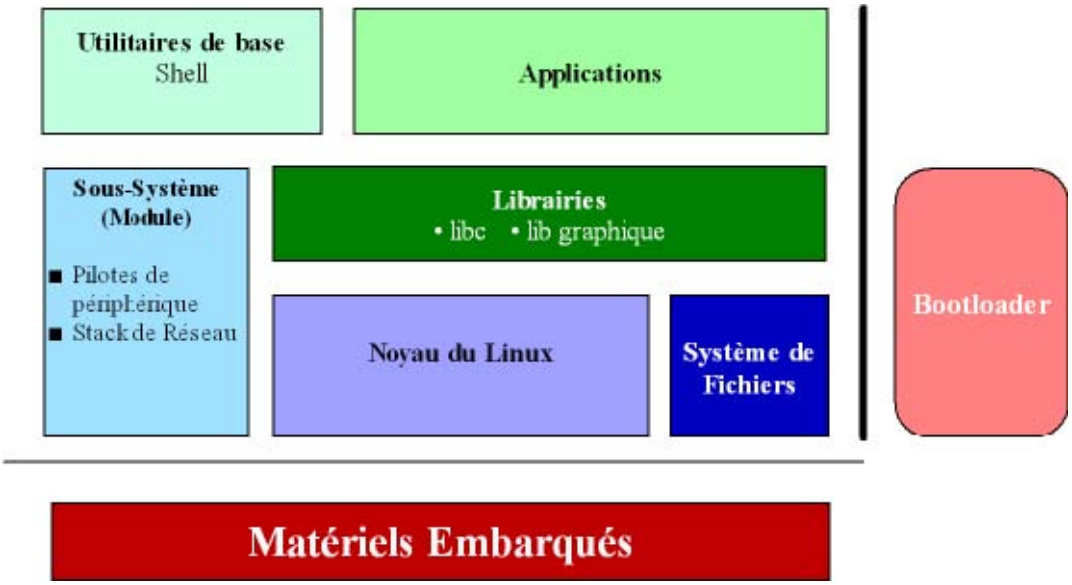


Figure 3 Un exemple des composants logiciels

ses composants possibles. Ces composants Peut-être sont présentés ou non dans les différents systèmes. Cela dépend de quel objectif de système.

Le noyau est le composant le plus important du système d'exploitation. Son but est de contrôler le matériel d'une façon logique et il fournit les services au bas niveau aux utilisateurs au haut niveau. Comme avec autre Unix, Linux conduit des dispositifs, contrôle des accès d'I/O, gestion de processus, gestion de la mémoire, manipule la distribution des signaux.

Les applications utilisent un ensemble de l'APIs fourni par le noyau, Donc le noyau de linux va être porté avec un peu ou pas de changements. Cela permet de réutiliser les logiciels existence pour Linux embarqué :

Le noyau doit pouvoir charger et/ou accéder à un Rootfilesystem via une certaine forme de stockage permanent ou stockage en réseau. Un Système de Fichiers doit être présent dans Linux embarqué.

Pourtant, les services exportés par le noyau sont souvent incapables pour être employés directement par des applications. Au lieu de cela, les applications se basent sur des bibliothèques qui fournir les APIs familiers et services abstraits qui communique avec le noyau pour obtenir la fonctionnalité désirée. La bibliothèque principale employée par la plupart des applications de Linux est la bibliothèque de GNU C. Pour les systèmes embarqués de Linux, La librairie GNU peut être remplacé en autre bibliothèque (voir $4.1.1)

3.4 Systèmes Linux Embarqué existants

Distribution de Linux Embarqué:

Voici, la liste de distributions très active que j'ai consulté les articles concernant le « Embedded Linux » dans le site Web http://www.linuxdevices.com/articles/

Lineo Embedix

Lineo développe et vend des composants logiciels et des solution de système embarqué en interaction avec Internet. Cette société possède 3 famille de produits : Embedix (Système d'exploitation embarqué), Embrowser (embedded web browser) et de nombreuses technologies de l'embarqué. Elle aussi développe les applications du temps réel à base de Linux.

- Architecture de processeurs supports :
  Motorola PowerPC, x86, MIPS, and SuperH.
- Caractéristiques de Système d'exploitation :
  Temps réel mou (Soft real time)
  Temps réel dur (Hard real time)

MontaVista Hard Hat Linux

Hard Hat Linux est un produit commercial de la société Monta Vista qui s'attache à faciliter le développement de système embarqué. Elle se concentre aux travaux pour améliorer les performances temps réel du noyau Linux.

- Architecture de processeurs supports : Intel et Motorola PowerPC
- Caractéristiques de Système d'exploitation:
  Noyau et Command de base sont inférieur à 500k.
  Temps réel (très bien)
- Application :
  Support des librairies graphique Microwindows
  ViewML Browser
  Application en temps réel.

uClinux

Une version libre de Linux qui ne possède pas l'unité de gestion de mémoire (voir le chapitre 5).

LEM (un peu utilisé):

Linux EMbedded est une petite distribution basée sur la Mandrake. Elle fait tenir sur moins de 8Mo un système complet contenant même XWindow. On peut ajouter le navigateur Netscape 4 et un serveur web.

Chapitre 4 : Processus de construction d'un Linux Embarqué

4.1 La construction de système

Avant de discuter la conception, on parlera de la terminologie qui concerne le développement .Il s'agit d'un environnement de développement croisé avec deux entités à distinguer :

La cible (Target-System) est la plate-forme matérielle qui va accueillir l'OS et le ou les applicatifs embarques. En fait, La cible est un système cible sur lequel On veut mis en place les OS, ou bien l'application. Elle est caractérisée par l'architecture des processeurs (voir la section"Architecture matériel supportée par Linux"). Cela peut être IntelX86, MIPS, ARM, SuperHitachi-SH…etc.

L'hôte de développement (Host-Development) est la plate-forme de développement sur la quelle sont mis au point les différentes parties logicielles de la cible. En fait, C'est une machine hôte (host) pour le développement et la mise au point. Simplement, lors de la création d'un OS Embarqué ou d'une application embarquée, On écrit "code source" et compile directement les sources sur ce host de développement. Donc, Il faut y installer avant les outils nécessaires de développement (voir4.1.2) comme la compilation croisée, et peut-être un IDE pour faciliter l'intégration des ressources.

La construire ici, Ce sont, tout d'abord, de compiler de manière croisée sur l'hôte pour avoir un code objet exécutable par le processeur de la cible. De plus, On va peut-être tester l'application sur la même plateforme (PC standard) avec les outils logiciels de développement afin de faciliter le débouger.

4.1.1 Outils du développement

Comme le développement de logiciel ordinaire, les développeurs de système embarqué ont besoin des compilateurs, des interpréteurs, les linkers, les IDE, et d'autres outils de développement. Les outils pour les développeurs de système embarqués ont différents. Car ils fonctionnent typiquement sur une plateforme lors de la construction des applications pour le système cible. C'est pourquoi ces outils s'appellent souvent des chaîne de compilation croisée en anglais "cross-platform development tools" ou "cross-development tools". Cette section discutera la chaîne de compilation croisée de GNU qui est très connu dans le monde de Linux, et IDE.

Les chaînes de compilation croisée (GNU)

Elles sont très importants pour développer Linux Embarqué, Donc, tout d'abord, Je les rappelle de manière définitive. La compilation croisée (cross-compile) est un processus de création des logiciels pour le système (microprocesseurs) cible en utilisant un machine hôte avec les outils tel que compilateur, bibliothèque C, ce qui manipuler le fichiers binaires.

GNU fournit un chaîne de compilation croisée qui est très suivant utilisé pour la construction d'un Linux Embarqué. Vous pouvez télécharger le composant du toolchain de GNU sur ftp://ftp.gnu.org/gnu/

Elle se compose des composants suivants:

1. *binutils*: un ensemble d'outils de manipulation de fichiers binaires comme : ld, gas, etar..
2. *compilateur* C/C++:GCC et débogueur GDB
3. *bibliotheque C*
   - Glibc,
   - Dietlibc
   - uClibc

La bibliothèque C

Généralement, c'est la base d'écrire l'application. On tient donc compte le choix de bibliothèque C. GNU Toolchain supporte complètement librairie glibc, mais on peut ajouter les librairies afin des applications et fonctionnalités spécifiques, Donc je parlera les librairies possibles pour Linux Embarque et aussi leurs supports dans la suite. Un Linux Embarque dispose ses contraintes et sa limitation, la taille de la bibliothèque glibc dévalorise un candidat pour l'usage sur le système cible. Au lieu de cela, nous devons rechercher une bibliothèque de C qui aura à la fois la fonctionnalité suffisante et la taille petite relativement. Un certain nombre de bibliothèque sont été implémenté dans Linux Embarque, Je choisis 2 librairies très utilisées pour présenter : uClibc et Diet libc. Pour chaque bibliothèque, je fournirai l'information et ses caractéristiques.

uClibc : http://www.uclibc.org/

La bibliothèque d'uClibc vient du projet d'uClinux, qui fournit un Linux qui fonctionne sur des processeurs de sans-MMU. Cependant, elle supporte maintenant processeurs qui ont MMU et beaucoup d'architectures. La uClibc peut être employé comme bibliothèque partagée sur toutes ces architectures. La taille < 900kb.

Bien que l'uClibc ne supporte pas complètement comme glibc. Il fournit la plupart de la même fonctionnalité. Voici les supportes de base de uClibc, si vous en savez plus, consultez sur le site ci-dessus et les documents des applications que vous voulez porter.

- Librairie partagée
- Supporte de API comme gLibc
- Supporte de réseau
- Supporte de toutes les architectures
- Supporte de Debug

Aujourd'hui, un grand nombre de programme sont recompilés pour adapter à l'utilisation de la librairie ucLibc, existant sur l'Internet. Donc, je recommande qu'on l'installe sur votre système, c'est facile de trouver une application embarquée et profiter de ses supportes de programmation.

Diet libc

Il est écrit à partir de"scratch" (construire à nouveaux et ne se base pas n'import quelle librairie) pour réduire la taille au minimum et optimiser la performance. Le Diet libc ne soutient pas toutes les architectures de processeur discutées. Il soutient ARM, les MIPS, le x86, et PPC. Le Diet libc peux être employé comme bibliothèque partagée sur quelques plateformes, mais il est souvent utilisé comme bibliothèque statique.

Les autres :

Newlib :http://sources.redhat.com/newlib/developpéparREDHAT.
MiniCLib :http://www.cs.vu.nl/ast/minix.html

Librairie graphique

Un système embarqué peut avoir besoin d'un affichage performant. Un écran LCD, par exemple, exige des performances graphiques. Une librairie graphique doit s'installer si le système visé à « l'utilisation d'interaction » (user interaction). C'est pourquoi, quelques solutions existantes seront présentées brièvement

Xfree86

Xfree86 est le gestionnaire d'affichage graphique le plus commun sous Linux. Il est comparable à Windows pour pouvoir afficher du graphisme à l'écran, permettre de contrôler les fenêtres... Sa version actuelle est devenue énormément lourde au niveau du stockage en mémoire de masse et surtout au niveau du besoin en mémoire RAM. Il est simplement adapté au Pc actuel qui sont avant tout multimédia. Pourtant, n'est pas envisageable dans un système embarqué aux ressources limitées. La librairie *XLib* fournie par Xfree86 dispose de 4 à 12 Mo.

Nano-X (Microwindows)

Nano-X reprend le principe comme **Xfree86**. Il n'est cependant pas autant encombrant que Xfree. En effet, Nano-X est développé à l'intention des systèmes minimaux. C'est pourquoi sa taille n'excède guère les 250Ko. En outre, pour faciliter le développement des applications. Donc, Nano-X est utilisable sous l'environnement graphique X dans le système embarqué. Les applications sont totalement testables sur l'environnement de développement.

Les autres

Il existe encore une multitude d'applications permettant l'affichage de graphisme pour des systèmes embarqués minimaux fonctionnant à l'aide de Linux. Pour plus savoir l'informations voici une petite liste sur les autres possibilités: *PicoGUI, QtEmbedded, OpenGui, PicoTk, MiniGui, PocketLinux …*

L'environnements intégrés de développement

Beaucoup d'environnements intégrés de développement (IDEs) sont disponibles pour Linux. La plupart de ces IDEs sont habituellement employées pour développer des applications natives. Néanmoins, elles peuvent être personnalisé pour la compilation croisée en plaçant un compilateur appropriés dans la configuration de l'IDE. Voici la liste de source ouverte IDEs possible pour Linux Embarqué : ECLISP, Anjuta, KDevelop.

4.1.2 Méthodologie de développement

Dans cette section, Je vais indiquer la méthodologie à suivre pour construire Linux Embarqué. On commence par les ressources qui sont toujours disponible dans le monde de Linux. C'est-à-dire que les outils de développement et les codes sources sont téléchargeables, complètement gratuits et ouverts à développer. Tous les liens concernent les ressources ou des options et alternatives possibles pour votre choix sont listées.

En cherchant sur l'Internet, il est facile de trouver de distributions Linux pour votre système adaptatif. Je ne vais pas aborder leur utilisation dans ce chapitre. Pour élaborer un système Linux, On a deux solutions sont possibles.

- Construire un système A Partir d'une distribution Linux libre existante comme Mandrake ou RedHat.

- Construire un système en compilant et en portant le noyau officiel (parexemplesurwww.kernel.org), par choix et intégration des composants logiciels.

Les noyaux disponibles dans les différentes distributions Linux sont souvent patchés afin d'offrir le support de périphériques et sont applicatif pour son objectif. Le but de ce chapitre est de parler comment construire en général, donc c'est la deuxième solution qui est retenue. Cette approche nécessite un investissement intellectuel. Mais la construction d'un système Linux nous permet de connaître tous les aspects du système d'exploitation et de la structure du système Linux. Théoriquement, si l'on fait l'étude de cette construction, Cela nous apporte la connaissance de base et vous être capable de réaliser n'importe quel système. Vous pouvez rencontrer même problème lors de la conception d'un système concret.

Un système d'exploitation Linux cible est créé en configurant et en combinant ensemble les composants appropriés du système. Les aspects de développement sont discutés de façon un peu détaillée. Mais, je parle comment concevoir le Système Linux Embarqué, Plus précisément, le processus se compose de combien d'étapes principales ? Et quels sont les ressources et les outils de développement nécessaires dans chaque étape ? Quelles sont les tâches à faire.

Les étapes principales de la construction (selon le livre «Building Embedded Linux System). Il y a 5 étapes à créer un Linux Embarqué.

1. Déterminer les composants

La détermination des composants matériels est la première étape et il n'est pas possible d'effectuer de manière parallèle avec les autres étapes. Ces composants comprennent une architecture de processeurs, des stockages, et des entrées/sorties qui sont abordés dans la partie $2.4 et $3.3. Le choix est très important, vous devez choisir les composants qui assurer d'être intégré aux autres au niveau physique tandis chaque composant est supporté par le noyau de linux et son pilots est développé.

2. Installation du bootloader.

Le démarrage ou «boot» d'un système d'exploitation est une phase d'initialisation. Il s'agit de mettre en service un programme relativement capable, le noyau, à partir d'un autre extrêmement limité, le BIOS. Il est intéressant de connaître quelques étapes du lancement de Linux. Ayant été développé pour des systèmes linux, L'installation du boot est faite dépendant de 3 facteurs : l'architecture de processeurs, types de démarrage et types de stockage.

3. Portage du noyau Linux.

D'abord, Il faut choisir le noyau à porter correspondant au type de processeur de la cible. Ensuite, vous faites la configuration du noyau pour retenir les sous composants nécessaires qui supporte à vos fonctionnalités, pour optimiser et réduire la taille de noyau. Enfin compiler le noyau.

4. Création du Système de fichier de racine. (RootFileSystems)

Les librairies utiles seront configurées. Les commandes et utilitaires de Linux pourront être installés sur la mémoire de stockage du système embarqué.

5. Développement des drivers spécifiques et de l'application finale.

Il est souhaitable de ajouter une fonctionnalité qui n'est pas supportés par le noyau de Linux lors de la phase de construction du système. On doit écrire un nouveau pilotage.

Pour un matériel spécifique, On doit écrire un module (sous-système) pour fournir les nouveaux services ou nouvelle fonctionnalité à la couche d'application.

Concernant le développement d'application apres avoir l'environnement Linux Embarqué. Il est important de voir s'il n'existe pas déjà une application existante collant à son besoin. Dans le cas de ne pas trouver, c'est à vous-même qui développez, Il faut attention à la limitation de la mémoire et la manque de fonctions par rapport du système complet. Dans le meilleur des cas, vous devez les traduire par une recompilation croisée, ensuite intégrer tous les composants logiciels enfin faire teste totalement son fonctionnement .

4.2 Préparation et établissement de l'environnement de développement

Maintenant, on parle des types de host qui est la plus utilisé. Et puis classifier ces types d'hôte. Voici, les types très utilisés:

4.2.1 Choisir la plate-forme de développement

Linux ou Unix Workstation

Si vous développez un Linux Embarqué, Il vaut mieux choisir Linux pour construire un Linux Embarque. Parce que le développement Linux Embarqué exige que vous vous familiarisiez assez avec Linux et il n'y a aucune meilleure manière de faire ceci comme l'utilisation de Linux pour votre travail de tous les jours. Vous pouvez utiliser l'import quelle distribution Linux telle que Mandrake, Redhat et

Debian sur votre système hôte de développement. La liste des outils est mentionne dans la section 4.1.1.

Windows (XP/2000/NT/98)

Avec la famille de Windows, il y a plein de produit commercial. On peut utiliser des IDE commerciaux, par exemple: Code Warrior de Metrowerks qui fonctionne avec les versions Linux embarqué de LynuxWorks(BlueCat), Lineo/Metrowerks/ Motorola et Montavista, ou bien MicrosoftVisualStudio avec le plug-in de LynuxWorks qui fonctionne avec la version Linux embarqué…etc.

Heureusement, Les outils supportés par Linux, qui peut fonctionner sur Windows à condition d'installer l'environnent Cygwin. Vous pouvez le utiliser aussi.

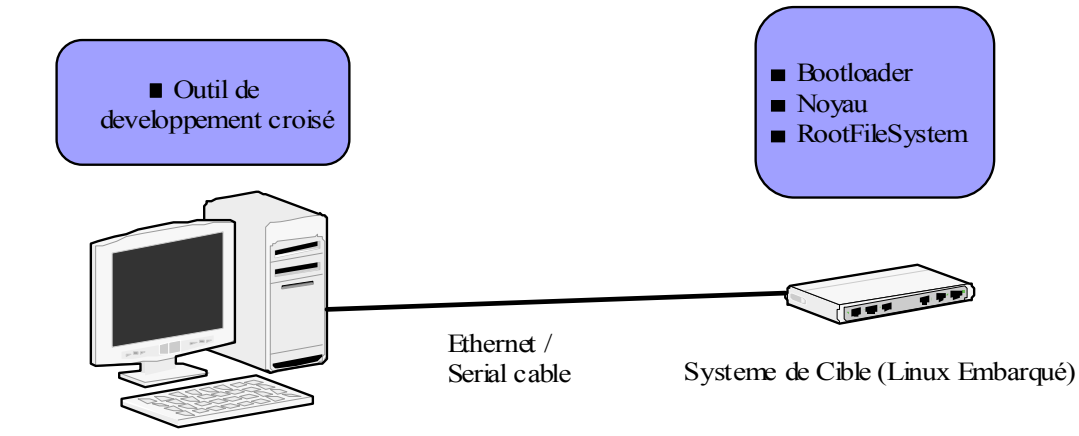
4.2.2 Méthodes d'accès entre l'hôte et la cible

Le problème lors du développement de l'embarqué, Comment faire la liaison entre l'hôte et la cible ?. Il y a trois méthodes différentes beaucoup employées de hôte-cible sont pour le développement de l'embarqué:

- Développement connecté en anglais "Linked Setup".
- Développement par stockage amovible en anglais "Removable Storage Setup".
- Développement sur cible en anglais "Standalone Setup".

Avant de la conception d'un Linux Embarqué. C'est a vous qui choisit un type de communication de hôte-cible selon votre méthodologie de développement et sur les contraintes de stockage, de performances et d'accessibilité du système cible.

Développement connecté



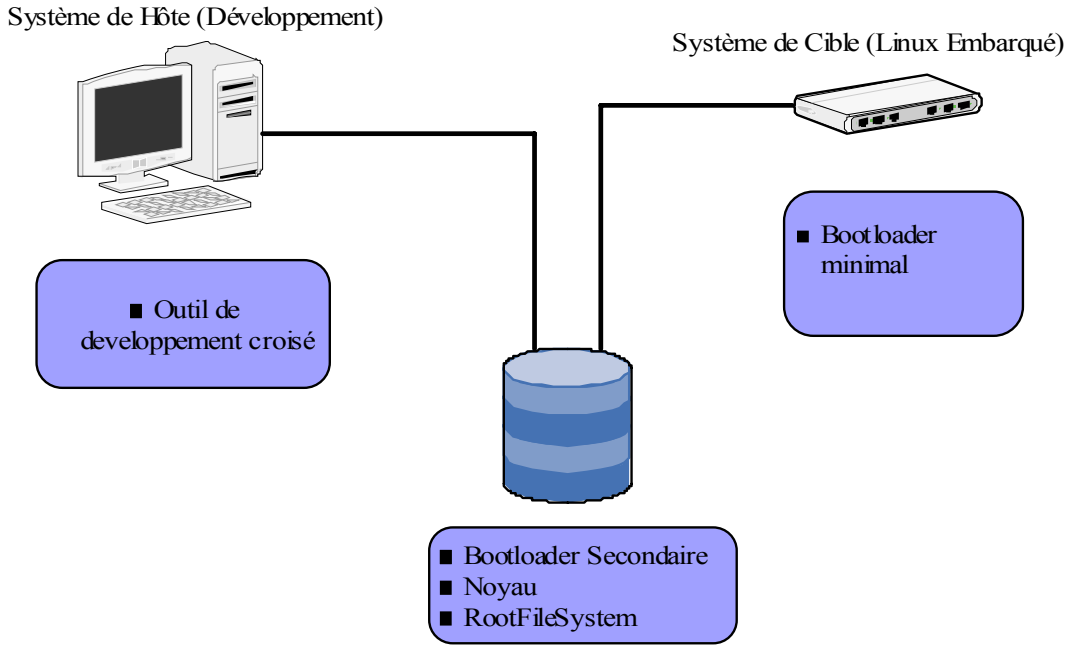
Dans cette installation, la cible et l'hôte serveur sont de manière permanente reliés par un lien physique. Ce lien est typiquement un câble de série ou un lien d'Ethernet. La propriété principale de ce type est qu'aucun device matériel de stockage physique n'est transféré entre la cible et l'hôte. Tous les transferts se produisent par l'intermédiaire du lien.

Le lien permet à l'hôte de mettre facilement à jour la cible à distance. A cote de la cible, elle peut récupérer dynamiquement le noyau via TFTP à l'extérieur. Le rootfs peut être également un NFS qui est montré à distance au lieu de l'utilisation de stockage local.

En plus, le lien physique peut également être employé pour le but du débogage, Il est, cependant, ceci dépende de quel câble physique que vous employez. Par exemple, Dans cette installation, le lien d'Ethernet est souvent employé pour télécharger l'exécutable, le noyau, le système de fichier de racine, alors que le lien RS232 est employé pour le débogage. Si Il y a encore le lien spécial pour seulement déboguer, telle que la carte de JTAG et BDM.

En résume, la configuration la plus rencontrée dans le développement de l'embarqué.

Développement par stockage amovible

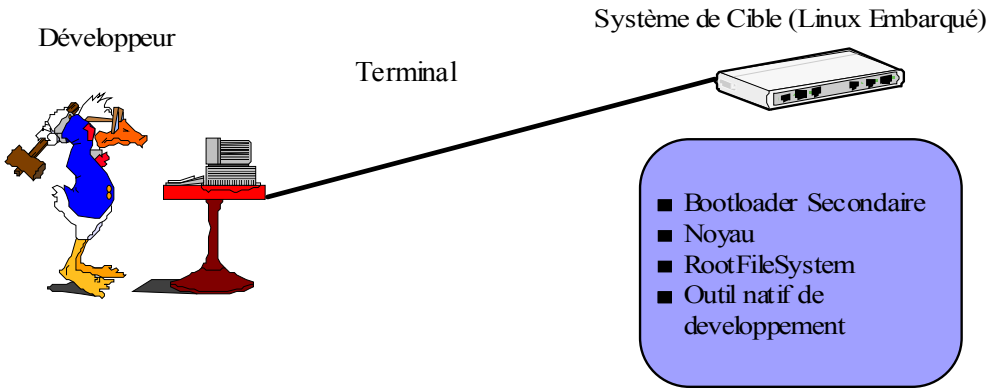


Il n'y a aucun lien physique direct entre le hôte et la cible. Au lieu de cela, un dispositif de stockage est écrit par l'hôte, est puis transféré dans la cible, et est utilisé pour booter ce dispositif.

Comme avec le type précédent, l'hôte contient l'outil de développement croisé. Cependant, la cible comporte seulement un bootloader minimal. Le développeur place le noyau et le rootfs sur le stockage amovible tels que le CompactFlash, FlashDisk et EEPROM. Ils sont programmées sur l'hôte et chargées par le bootloader minimal de la cible lors du démarrage après avoir été installé dans la cible.

Cette installation est beaucoup populaire pendant les phases initiales du développement inclus de système. Parce qu'elle permet de facilement créer un émulateur de ROM pour le faire ressembler à un développement connecte.

Développement sur cible



La cible est un système de développement avec ses outils de construction des composantes nécessaires. Actuellement, cette installation est comme le travail dans le Workstation et n'a pas l'environnement de développement croisé, tous les outils de développement fonctionnent dans leur environnement natif.

La cible comporte ses propres outils de développement natifs comme éditeur, compilateur, débogueur. Le développeur doit accéder à la cible soit directement, à l'aide d'un clavier et d'un écran, soit par le réseau à partir de l'hôte en utilisant le ssh, telnet...

Le problème, C'est que les systèmes embarqués disposant de suffisamment d'espace de stockage et de mémoire pour pouvoir faire fonctionner un compilateur. Donc, Ce type d'installation est assez populaire pour construire les systèmes embarqués basé sur PC (PC-based embedded systems).

4.3 Device de Stockage et le bootloader

4.3.1 Device de Stockage

Autrement dit qu'il est la mémoire de masse. Il est certain que vous aurez besoin de ceux-ci pour stocker des informations de manière permanente: votre programme d'application, des données, des paramètres...En général, Dans un système Linux, la mémoire de masse dispose deux fonctionnalités principales:

1. Héberger le système de fichiers de Root de Linux. L'existence de ce système de fichiers est indispensable au bon fonctionnement du système.
2. Mettre le bootloader dans le premier secteur du périphérique concerné pour initialiser le systeme.

Le choix de device de stockage a l'influence sur l'installation du bootloader. Car le bootloader dépende souvent de le device de stockage principal installé dans le système. Donc, Il faut choisir une solution convenable pour le stockage. Je parlera de la création de bootloader dans le section suivant. Mais, d'abord quelques types de stockage bien supportés par Linux sont abordés.

Il ne suffit pas de dire tout les devices existant. Mais la description souvent nous aide à connaître de ses touts et limites. Voici les solutions possibles :

Disque Dur

C'est la solution de stockage la moins coûteuse, et celle offrant les plus grandes capacités. Mais les disques durs magnétiques sont souvent peu recommandés pour des applications embarquées. Car ils sont encombrants (la grande taille), bruyants, et peut-être sensibles aux chocs, à la température et Ils ont besoin de beaucoup d'énergie. De plus, une coupure d'alimentation au milieu d'une écriture peut avoir des conséquences désastreuses. Bref, je pense que les disques durs convient petit peu de l'embarque. Si les contraintes ne sont pas trop fortes, il vaut donc mieux les oublier.

CompactFlash

Les compact Flash sont des composants électroniques sans aucune mécanique, basés sur la technologie de mémoire Flash, Lors de la mis en oeuvre ce device dans le linux embarqué, Il y a plusieurs avantages que le disque dur. On en trouve aujourd'hui beaucoup dans les appareils numériques. Ils disposent des caractéristiques : petits, silencieux, résistants aux chocs, et économe d'énergie. Ils semblent donc posséder toutes les qualités requises...cependant en cas de coupure d'alimentation, ils ont encore la limitation sur la probabilité de perte des données comme disque dur. Les CompactFlash, un candidat pour le stockage de l'application embarqué.

Figure 4: Adaptateur PC-Card pour CompactFlash et le CompactFlash

Au niveau des systèmes informatiques embarqués, elle est le plus souvent utilisée de plusieurs manières : Comme un disque IDE (cas le plus courant) soit directement soit à travers un adaptateur PC-Card.. Via un adaptateur USB

Les mémoires DiskOnChip (DOC)

Le DoC est une mémoire FLASH intelligente (contenant un BIOS) développée par la société MSystems(http://www.m-sys.com).

Figure 5: Le Disk On Chip de M-System

Les DoC utilisent une interface propriétaire (ni IDE, ni SCSI) et leur utilisation nécessite une configuration spéciale du noyau Linux.Il y a deux méthodes possibles:

- Utiliser le pilote propriétaire fourni par M-Systems(un patch du noyau)
- Utiliser le pilote MTD du noyau Linux.

4.3.2 Le BootLoader

Le bootloader est principalement responsable de charger le noyau de linux, il est un composant très important de système. L'installation d'un bootloader est une tâche commune à tous les systèmes de Linux. Pourtant, pour le système embarqué, C'est une tâche spéciale. Parce que le bootloader utilisés dans les systèmes embarqués sont complètement différents de ceux utilisés dans les systèmes communs: Je me découpe en trois types de bootloader selon les interfaces de devices.

- Amorçage-chargeur à partir de IDE et SCSI: Disque et CompactFlash.
- Amorçage-chargeur à partir du réseau en fonction des protocole BOOTP/DHCP, et TFTP.
- Amorçage-chargeur spécial pour DOC.

Voici, les sources de bootloader pour Linux.

LILO

Ceci est développé pour x386, et implémenté dans beaucoup de distributions de Linux. LILO peut démarrer à partir du périphérique IDE et SCSI. Il est bien implémente si vous avez un disque dur ou CompactFlash avec l'architecture x86. Sa limite est de n'être pas capable de marcher sur les autres architectures.
LILO avait été soutenu DOC, mais il faut utiliser une version modifiée de LILO pour pouvoir démarrer sur ce support. Cette version adaptée (soit lilo-mtd) est disponible dans le répertoire patches des sources MTD chargées à partir du site.

GRUB: Grand Unified Bootloader

Grub remplit la même fonction que lilo. Il est une version avancée de LILO, Sa limite est même que LILO. Il fonctionne seulement sur le plat-forme x86. Cependant, Un aspect des avantages du GRUB que vous pouvez trouver utile pendant le développement, est sa capacité de booter via le réseau en utilisant TFTP, et BOOTP ou DHCP. En plus, GRUB soutient complètement les DiskOnChips.
En résumé, Si vous avez envie de développez un Linux Embarque sur le plate-forme x86, GRUB est une meilleure solution pour le bootloader.

Logiciel	Architecture de processeurs	Types d'amorçage	Description
LILO	x86	- Disque Dur - Mémoire de Flash	Le bootloader principal pour le Disque commun sous Linux
GRUB	x86	- Disque Dur - Mémoire de Flash - DOC - TFTP, NFS	Une version avancée de LILO
U-BOOT	Tous	Tous les types	le bootloader universel basé sur ARM-Boot, pour presque tous les processeurs embarqué :
REDBOOT	Tous	Tous	Developpé par RedHat

4.4 Portage du noyau Linux

Choisir un noyau approprié

Comme je dis dans la section $4.1.2. On va construire un Linux Embarque à partir du noyau original sans prendre distribution de Linux. On compilera directement le code source officielle du noyau pour le système Linux Embarque. Ce code source est disponible dans le site web http://www.kernel.org. Avant de compiler, Il faut choisir le noyau correspondant au type de processeur de la cible. Cela est très important, car le noyau est écrit pour quelques architectures de processeurs. Il est donc nécessaire de sélectionner un noyau approprié à votre système cible. Je ne parle pas des systèmes spéciaux parce qu'ils nécessitent le noyau spécifique supporté par ses producteurs, cependant les architectures très connus supporte par linux (voir la section $3.2). Malheureusement, le lien ci-dessus n'est pas suffisamment pour la compilation parce que le développement de Linux pour ces architectures n'est pas synchronisé avec les noyaux principaux lancés par l'organisme de kernel.org. Par exemple, pour l'architecture de ARM avec sa fonctionnalité spéciale, Nous téléchargeons le noyau dans ce site et devons utiliser le "*patch*" ayant le support cette fonctionnalité, à partir site de ARM Linux http://www.arm.linux.org.uk/. En combinant ce patch dans le noyau principal, On obtient le noyau complet qui contient tous les caractéristiques requises pour ce système. Voici les endroits contenant les noyaux appropriés pour des architectures de processeur.

Processeur	Site Web
x86	http://www.kernel.org/
ARM	http://www.arm.linux.org.uk/developer
PowerPC	http://penguinppc.org/
MIPS	http://www.linux-mips.org
SuperH	http://linuxsh.sourceforge.net/
M68k	http://www.linux-m68k.org/

Bien sur que Le type d'application, le prix, la consommation, les performances désirées sont généralement les facteurs qui influencent la décision du développeur dans le choix d'un processeur. Mais il y a un choix de composant matériel aussi important que ceux d'un processeur, Est-ce que l'architecture de votre système supporte MMU[2] ou pas le MMU ?

Une MMU est un élément hardware qui fournit différents emplacements mémoire protégés pour le noyau et les processus, renforce la protection de la mémoire et facilite l'exploitation de la mémoire virtuelle. Sans MMU, ces options ne sont tout simplement pas accessibles. Il faut donc remarquer qu'il est bien plus facile de travailler sur un processeur équipé d'une MMU.

Cependant, dans le développement de l'embarque, Les système à la petite taille n'ont pas souvent le MMU. Parce que leurs applications sont simples et petites, il n'est pas nécessaire de gestion de mémoire au niveau virtuel et de protection.

[2] Memory Management Unit

Beaucoup de Linux embarque sont développés sur l'architecture sans MMU, c'est la raison pour laquelle je mentionne dans cette section.

Le noyau uClinux permet l'utilisation de Linux sur des processeurs sans MMU. Les applications, qui fonctionnent parfaitement avec un noyau standard, peuvent écrire dans les parties de la mémoire réservées au kernel. Mais Elles risquent de rencontrer certains problèmes. Le système d'exploitation uClinux est encore à son stade de développement. Il est le plus utilisé pour Linux Embarqué. Voir la détaille dans chapitre 5.

Configurer le noyau associé

Nous devons configurer le noyau avant de commencer à le compiler. Pendant la phase de configuration, On sélectionne les composants qui feront partie de notre futur noyau. Imaginant par exemple qu'on utilise le système de fichiers EXT3. Il nous faudra alors choisir la prise en charge du système de fichiers EXT3 au moment de configurer le noyau. La liste d'options principales de composants disponibles à presque touts les Linux: (Voici un exemple si vous en savez plus, consultez sur le document de support de Linux).

- Power management supporte
- ATA/ATAPI/MFM/RLL supporte
- Périphérique USB
- Bluetooth et Infrarouge supporte
- Système de Fichiers
- Réseau
- MTD Driver supporte

Concernant la taille du système largement réduite, on peut s'attaquer à l'optimisation du noyau en fonction des besoins du système. Le gaine n espace disque ne sera pas énorme mais le temps de chargement du noyau sera réduit et son fonctionnement optimisé. L'optimisation et la réduction sont faits dans cette phase de configuration.

Enfin, la compilation à l'aide de la command *make, make modules, make mrpropre* ..

4.5 Création de Système de Fichier du Root

Choisir Système de fichiers pour le ROOT

Je mentionne les systèmes de ficher souvent supportés pour les devices de Linux Embarqué.

JFFS

Un système de fichiers sur les mémoires Flash et Il a les spécificités des supports de stockage mémoire via la couche MTD de Linux exporté par le noyau. Il peut donc être implémente pour le device de stockage de DiskOnChip. La version deuxième ajoute le support de la compression et quelques améliorations.

ROMFS

Un système de fichiers est le device de stockages virtuel qui fonctionne sur la mémoire RAM avec des fonctionnalités comme le disque normale. Il est implemeté sur le système avec le microcontrôleur.

CRAMFS

Il a même but que ROMFS mais avec plus d'optimisations et compression des fichiers. Il est assez utilisé dans l'embarqué pour construire le disque virtuelle.

EXT2 et EXT3

Un standard de Linux, plein de supports pour le Disque Dur, Compact Disk.

Contenu de RootFileSystem (\)

Comme le système normal, RootFileSystem contient les pilotages, les fichiers de configuration et les applications. L'exception de l'optimisation des fichiers et répertoires nécessaires, vous pouvez supprimer quelques fonctionnalités tel que les permissions et droite d'accès.

Utilitaires de base et commande minimale

Linux comme tous les systèmes Unix utilise un interpréteur de commande pour dialoguer avec le système. L'opérateur écrit des commandes qui sont traduites parle système. Ces commandes fournissent à l'utilisateur un certain de nombre de fonctionnalités intéressantes pour gérer, administrer et configurer le système. Ces programmes sont malheureusement la taille assez grande. C'est pour quoi leur optimisation doit être considéré. Je parle donc deux outils très utilisés lors de la construction de Linux Embarqué.

BusyBox

Il combine des versions de nombreux utilitaires Unix en un exécutable unique et de petite taille. BusyBox permet de remplacer la plupart des utilitaires qui se trouvent dans les paquets *fileutils, shellutils, findutils, textutils, modutils, grep, gzip, tar, etc*. Il représente un environnement POSIX assez complet pour petits systèmes sou systèmes embarqués. BusyBox a été écrit avec pour objectif une optimisation détaille et une utilisation de ressources limitées. Cet outil est modulaire, ainsi il est

facile d'inclure ou d'exclure des commandes à la compilation. Cela permet de l'adapter facilement à chaque type de système embarqué. Sa taille plus petite est d'approximativement 200 Ko.

TinyLogin

TinyLogin fonctionne selon le même principe que BusyBox. Il fournit des commandes Shell complémentaires pour la gestion des connexions utilisateurs. Il s'agit des commandes : *addgroup, adduser, getty, login, passwd, su, sulogin*. Et la taille est aussi petite.

Un autre avantage de 2 ces outils, Ils sont compatible et fonctionnent bien avec la librairie uClinux.

Chapitre 5 : PROJET µClinux et la simulation Skyeye

5.1 Présentation de µClinux

Parmi l'ensemble des distributions Linux embarqué, j'ai choisi le projet µClinux parce qu'il est de plus en plus utilisé de nos jours pour l'étude de cas. Il s'agit d'un portage Linux sur des processeurs qui ne possède pas d'unité de gestion mémoire MMU. Ma démo consiste au portage de ce système sur la plate-forme de développement LINUX (Mandrake 10 ou Fedora 2) et à sa mise en oeuvre au la machine virtuel de l'architecture ARM.

Il y a beaucoup de produits matériels sans MMU qui est généralement bon marché et plus petit. Tandis que beaucoup de supports sur le processeur tel que Motorola 68k, Coldfire, Intel i960, ARM, MIPS…etc.

L'absence de MMU implique des différences entre µClinux et l'environnement Linux. En effet, bien que µClinux soit un véritable système Linux multitâches, quelques considérations sont tout de même à prendre en compte lors de la conception d'un programme. Ses limitations d'usage sont : (extrait du « Linux embarqué : le projet µClinux, Patrice KADIONIK,»)

- L'absence de mémoire virtuelle et de protection de la mémoire. L'espace protégé est inexistant. Et le processus parent est suspendu jusqu'à ce que le processus fils appelle *exec()* ou *exit().*
- L'appel fork() non supporté, mais il est possible d'utiliser l'appel système vfork()
- L'image binaire d'un processus < 256Ko. L'appel système exec() ne peut charger actuellement une image binaire supérieure à 256Ko.
- La taille de la pile est fixe pour chaque processus.

La distribution µClinux utilisée est la version à partir d'émulateur Skyeye. Elle est dérivée du noyau Linux 2.4 et permet le portage sur le processeur l'ARM. On peut également trouver dans cette distribution, des outils de développement (GNU gcc, gdb…) ainsi que le portage de quelques applications sous licence GPL (serveurs Web, shells…).

5.2 Caractéristique de uClinux

Il possède les caractéristiques suivantes :

- la taille du noyau est inférieure à 512Ko
- la taille du noyau et des commandes Linux inférieures à 900Ko
- une pile TCP/IP complète assurant une connectivité IP au système embarqué sous µClinux
- µClinux supporte les systèmes de fichier NFS, SMB, ext2, MS-DOS, FAT16/32 et flash disk JFFS2 (MTD).

5.3 Outils, ressources et simulateur Skyeye pour uClinux

Les ressources :

uClinux-dist-200305522.tar.gz : source code du noyau de uClinux
uclinux4skyeye-v0.2.3.tgz : patch pour la simulation Skyeye
skyeye-binary-testutils-1.0.7.tar.gz : contient les rootfs disponible

Les outils :
Les packages de la compilation croisée pour l'ARM

arm-elf-gcc-2.95.3-4.i386.rpm
arm-elf-gcc-2.95.3-4.i386.rpm
arm-elf-binutils-2.11-5.i386.rpm

Le simulateur :

Skyeye-0.8.0.rpm

5.3 Processus de création d'un uClinux.

Le fait :

1. Installation «compilation croisée »

```
#rpm –i arm-elf-gcc-2.95.3-4.i386.rpm
 #rpm –i arm-elf-gcc-2.95.3-4.i386.rpm
 #rpm –i arm-elf-binutils-2.11-5.i386.rpm
```

2. Installation de Skyeye
Avant d'installer skyeye, il faut installer GTK > 2.0.

```
#rpm –i skyeye-0.8.0.rpm
```

3. Création de uClinux

```
#tar zxfv uClinux-dist-200305522
#cp <path>/ uclinux4skyeye <path>/ uClinux-dist-200305522
```

Sélectionner les composants du noyau

```
#make menuconfig
#make dep
#make
```

La sortie est un fichier d'image du noyau « linux »
Récupérer un fichier rootfs a partir skyeye-binary-testutils, et puis mettre sur même répertoire avec «linux», ajouter la configuration skyeye.conf

```
#cp <choisir un fichier> ./
```

Lancer la simulation avec uClinux :

```
#skyeye linux
(Skyeye) target sim
(Skyeye) load
(Skyeye) run
```

Chapitre 6 : Conclusion

Ce rapport est insuffisant pour le développeur de LE, mais donne la connaissance de Linux, et Linux Embarque sur des technologies les plus utilisées, l'analyse de composants, les méthodes, les outils et ressources nécessaire à construire un tel système.

LE se développe très vite (300% selon VDC) et Il peut devenir le standard industriel dans l'avenir. Linux embarqué est prometteur. On peut signaler particulièrement un point très intéressant. C'est l'utilisation de Linux embarqué sur les ordinateurs de poche, PDA, téléphone portable… Ce domaine est très actif avec des offres Linux sur les applications portables.

Personnellement, J'obtient quelques expériences qui me permet comprendre profondément le système Linux et d'utiliser les différentes fonctionnalités associées

Annexes

Processeurs

DEC Alpha : http://www.alphalinux.org
Processeur ARM et StrongARM : http://www.arm.linux.org.uk
Hitachi SuperH : http://www.m17n.org/linux-sh
IA-64 : http://www.linuxia64.org
i386 (x86) : http://www.linux.org
Motorola 68K : http://www.linux-m68k.org
Processeur MIPS : http://www.oss.sgi.com/mips
Motorola PowerPC : http://www.linuxppc.org
Sparc (32 et 64 bits) : http://www.ultralinux.org
Processeur sans MMU : http://www.ucLinux.org

Mémoire de masse

SanDisk qui est le leader sur le marché : http://www.sandisk.com
M-Systems : produit IDE : http://www.m-sys.com
PQI propose des solutions IDE 2,5'' : http://www.pqi.com.tw
Driver MTD pour mémoire Flash : http://www.linux-mtd.infradead.org

Compilateur croisée

Gcc : http://www.gnu.org/software/gcc/gcc.html.

Librairies C

ucLibc http://uclibc.org/
Diet-libc http://www.fefe.de/dietlibc/
NewLibc http://sources.redhat.com/newlib/
MiniClib : www.miniclib.com/

Librairies graphiques

Nano-x et MicroWindows : http://www.microwindows.org
PicoGui : http://www.picogui.org
Mini-Gui : http://www.minigui.com
QtEmbedded : www.trolltech.com/products/embedded/
OpenGui : www.tutok.sk/fastgl/
PicoTk : picotk.sourceforge.net/

Code source du noyau Linux

Linux : http://www.kernel.org
uClinux : http://www.uclinux.org

Système de Fichiers

JFFS : http://www.developer.axis.com
ROMFS : http://romfs.sourceforge.net/
CRAMFS : http://sourceforge.net/projects/cramfs/

Bootloader

Lilo : http://freshmeat.net/projects/lilo/
Gnub : www.gnu.org/software/grub/

IDE

ECLIPSE : http://www.eclipse.org/
Anjuta : http://anjuta.sourceforge.net/
KDevelop : http://www.kdevelop.org/

Logiciels embarqués

BusyBox : http://busybox.lineo.com
Tinylogin : http://tinylogin.lineo.com
Pcmcia-cs: http://pcmcia-cs.sourceforge.net/
uDHCP : http://udhcp.busybox.net/udhcp
MAD mpeg decoder : http://www.mars.org/home/rob/proj/mpeg/
LSH : http://www.net.lut.ac.uk/psst/
PppD : http://www.samba.org/ppp/
BOA webserver : http://www.boa.org/
OpenSSL : http://www.openssl.org/
BlueZ Bluetooht : http://bluez.sourceforge.net/
Thttpd : http://www.acme.com/software/thttpd/
E3 : http://www.sax.de/~adlibit/

Bibliographie

Livres:

[1] Building Embedded Linux System, KarimYaghmour , 2003.

[2] Linux for Embedded and Real-time Applications - DougAbbott.

[3] The Concise Handbook of Linux for Embedded Real-Time Systems – www.timesystem.com.

[4] Embedded Linux -Hardware, Software, and Interfacing - Craig Hollabaugh.

Les articles:

[5] EMBEDDED LINUX - Gaspoz Frédéric en 2002.

[6] Linux embarqué : le projet uClinux - Patrice Kadionik, 2003

[7] Embarquez Linux! ou Linux Everywhere - Pierre Ficheux, 2004

[8] Les Systèmes Embarqués Linux pour l'embarqué - Patrice Kadionik.

[9] Embedding Linux in a Commercial Product, Joel R.Williams

[10] Survey of Embedded OS - Catherine Lingxia Wang, Bo Yao, Yang Yang, Zhengyong Zhu.

Les sites Web:

[11] Le site Linux Embarqué page par PatriceKadionik http://www.enseirb.fr/~kadionik/cours.html

[12] The µClinux http://uclinux.home.at/.

[13] www.linuxdevices.com

[14] Site de référence : http://www.handhelds.org

[15] Distribution Linux Familiar http://familiar.handhelds.org