EMBEDDED LINUX

By Gaspoz Frédéric

Diplôme 2001-2002

HEVS Sion section électricité-infotronique

Gaspoz Frédéric 2001-2002

PC-104 et MicroLinux

Objectif
Le but de ce travail de diplôme est d'optimiser le système d'exploitation Linux en vue de l'intégrer dans un système embarqué du type compatible x86 au format PC-104.

Résultats
L'analyse de l'Os a permis de décrire les parties qui doivent être optimisées. Diverses solutions ont été examinées et les meilleures ont été testées. Le Linux optimisé a pu être expérimenté sur plusieurs plate-formes compatible x86. Cependant, le portage sur le Pc-104 de la Hevs n'a pas été possible.

Mots-clés
Linux, système d'exploitation, Os, Shell, compatible x86,

Ziel
Das Ziel dieser Diplomarbeit war die Optimierung des Betriebssystems Linux. Dieses optimierte System sollte dann auf ein Embedded System x86 Pc-104 portiert werden.

Resultate
Die Analyse des Betriebssystem hat erlaubt die Optimierung des Teilkomponenten des Os. Verschiedene Lösungen wurden untersucht und getestet worden. Das optimierte System wurde auf einem kompatiblen Computer installiert und getestet. Die Portierung auf ein Pc-104 System war wegen Hardwareproblemen.

Schlüsselwörter
Linux, Betriebssystem, Os, Shell , kompatiblen x86

Table des matières

TABLE DES MATIÈRES **3**
TABLE DES ILLUSTRATIONS ET DES TABLEAUX **5**
PRÉFACE **6**
A QUI S'ADRESSE CE RAPPORT ? 6
VUE D'ENSEMBLE 6
CONVENTIONS UTILISEES 7
REMERCIEMENTS 7
1 INTRODUCTION **8**
1.1 APPLICATIONS CONCRETES 8
2 APPROCHE ET CAHIER DES CHARGES **10**
2.1 CAHIER DES CHARGES 10
2.2 APPROCHE 10
3 L'OS LINUX **11**
3.1 INTRODUCTION 11
3.2 STRUCTURE DE L'OS 12
3.3 LES PERMISSIONS 16
3.4 LE DEMARRAGE 17
3.5 LINUX EN TANT QU'OS TEMPS REEL 18
4 LINUX, OS EMBARQUE **20**
4.1 CHOISIR LINUX 20
4.2 LE HARDWARE 22
4.3 PORTABILITE DU CODE SUR DIFFERENTES ARCHITECTURES 26
5 LINUX OPTIMISE POUR X86 ..... **29**
5.1 GENERALITES 29
5.2 L'OPTIMISATION ? 29
5.3 SYSTEME DE FICHIERS 30
5.4 SUPPRESSION DU SWAPPING 33
5.5 OPTIMISATION DE LA PROCEDURE DE DEMARRAGE 35
5.6 OPTIMISATION DES LIBRAIRIES C 37
5.7 SHELL ET COMMANDES MINIMALES 40
5.8 OPTIMISATION DU NOYAU 43
5.9 SIMPLIFICATION DU NOMBRE DE FICHIERS 47
5.10 COMPILATION 49
5.11 OPTIMISATION DES APPLICATIONS 52
5.12 AFFICHAGE 55
5.13 APPLICATIONS MINIMALES POUR LINUX EMBARQUE 59
5.14 CONCLUSION 61
6 PROCESSUS DE CONSTRUCTION D'UN SYSTEME LINUX EMBARQUE **62**
6.1 INTRODUCTION 62

6.2 METHODE ET PLATE-FORME DE DEVELOPPEMENT 62
6.3 CONSTRUCTION DU SYSTEME 63
6.4 CONCLUSION 104

7 TESTS DU SYSTEME OPTIMISE 105

7.1 MEMOIRE RAM MINIMALE 105
7.2 PLACE EN MEMOIRE DE MASSE 105
7.3 LE SYSTEME D'EXPLOITATION EN GENERALE 105
7.4 LA PROCEDURE DE DEMARRAGE 105
7.5 ARCHITECTURE X86 105
7.6 MODULE CHARGEABLES 106
7.7 LA CONNEXION PAR TERMINAL SERIE 106
7.8 L'AFFICHAGE 107
7.9 LE RESEAU ETHERNET 107

8 SYNTHESE 108

8.1 PROBLEMES RENCONTRES 108
8.2 EVOLUTION DU PROJET 110

9 CONCLUSION 112

10 ANNEXES 113

10.1 BIBLIOGRAPHIE ET LIENS UTILES 114
10.2 COMPLÉMENTS SUR LINUX 117
10.3 AFFICHEUR LCD 181
10.4 CODES SOURCES ET SCRIPTS UTILISÉS 194
10.5 DESCRIPTION DES COMMANDES BUSYBOX 231
10.6 DESCRIPTION DES COMMANDES TINYLOGIN 270
10.7 OPTIONS DE COMPILATION DU NOYAU 272
10.8 LICENCE GPL 315
10.9 TABLE DES MATIÈRES DÉTAILLÉE 320
10.10 GLOSSAIRE 326
10.11 CD-ROM ET DISQUETTE 334

Table des illustrations et des tableaux

FIGURE 1 : PDA HNT EXELIEN LINUX 9
FIGURE 2 : VR LINUX 9
FIGURE 3 : APPROCHE 10
FIGURE 4 : STRUCTURE DE LINUX 13
FIGURE 5 : EXEMPLE D'ARBORESCENCE 14
FIGURE 6 : BITS DE PERMISSION 17
FIGURE 7 : OPTIMISATION DE LINUX 30
FIGURE 8 : PARTITION EXT2 SUR DISQUE DUR 48
FIGURE 9 : MODE XIP 54
FIGURE 10 : ECRAN LCD 56
FIGURE 11 : UTILITAIRE CFDISK 65
FIGURE 12 : ARBORESCENCE MINIMALE 69
FIGURE 13 : MAKE CONFIG 74
FIGURE 14 : MAKE MENUCONFIG 74
FIGURE 15 : MAKE XCONFIG 74
FIGURE 16 : OPTION DE COMPILATION POUR INITRD 76
FIGURE 17 : PRINCIPE DE BUSYBOX 79
FIGURE 18 : CONSOLE SERIE 88
FIGURE 19 : OPTIONS DE COMPILATION POUR LA CONSOLE SERIE 89
FIGURE 20 : OPTIONS DE CONFIGURATION POUR L'ACCES AU RESEAU 90
FIGURE 21 : OPTIONS DE CONFIGURATION POUR L'ACCES AU RESEAU 91
FIGURE 22 : OPTIONS DE COMPILATION DE LA CARTE ETHERNET 91
FIGURE 23 : OPTIONS DE COMPILATION POUR LES MODULES 92
FIGURE 24 : SCHEMA LCD-PORT PARALLELE 99
FIGURE 25 : AFFICHAGE TEXTE EN MODE VGA 100
FIGURE 26 : OPTIONS DE NANO-X 101
FIGURE 27 : OPTION DE COMPILATION POUR FRAMEBUFFER 1 101
FIGURE 28 : OPTIONS DE COMPILATION POUR FRAMEBUFFER 2 102
FIGURE 29 : NANO-X 103
FIGURE 30 : TERMINAL SERIE SOUS WINDOWS 2000 106
FIGURE 31 : LCD HD44680 SUR PORT PARALLELE 107
FIGURE 32 : BOA WEBSERVER 107
FIGURE 33: ARBORESCENCE DES REPERTOIRES 118

TABLEAU 1: TAILLE DES LIBRAIRIES 39
TABLEAU 2: TAILLE DE BUSYBOX 42
TABLEAU 3: TAILLE DE TINYLOGIN 42
TABLEAU 4 : TAILLE DES DIFFERENTES OPTIONS DE COMPILATION 45
TABLEAU 5 : COMPARAISON ZIMAGE BZIMAGE 46
TABLEAU 6 : EFFICACITE DES OPTIONS –OX 51
TABLEAU 7 : PORT PARALLELE 182

Préface

Ce rapport est le fruit de mon travail de diplôme à l'école d'ingénieur de Sion en Valais. En l'espace de 12 semaines, il a fallu traiter un sujet d'une manière professionnelle telle que la formation ingénieur nous l'a appris. La matière traitée est très vaste et la période de travail relativement courte.

Le but de ce travail de diplôme est d'optimiser Linux pour l'intégrer dans un système embarqué du type compatible x86[1]. Le travail d'un ingénieur ne se bornant pas à une description purement théorique, il a fallut dans un second temps appliquer les concepts théoriques à un système embarqué réel. Ceci a permis la réalisation d'une mini-distribution Linux.

A qui s'adresse ce rapport ?

Ce rapport s'adresse en premier lieu à mon professeur de travail de diplôme Monsieur Weber Peter et à mon expert Monsieur Baillifard Marc-André qui doivent me juger pour le travail accompli durant ces 12 semaines.

En outre, quiconque souhaitant doter un système embarqué aux ressources minimal d'un système d'exploitation Linux ou simplement souhaitant comprendre le fonctionnement de l'Os Linux en partant d'un système Linux simple, peut trouver dans ce rapport des informations utiles.

Vue d'ensemble

Le chapitre 1, *Introduction*, présente un aperçu général de ce que contient ce rapport en introduisant le sujet de ce travail de diplôme, et en le situant dans le contexte actuel du marché au niveau technologique.

Le second chapitre expose le cahier des charges et la méthode de conception du projet.

Le chapitre 3, *L'OS Linux*, propose un rapide survole du système d'exploitation Linux ceci afin d'en comprendre le fonctionnement, pour permettre une meilleure compréhension des points abordés dans les chapitres qui suivent.

Les problèmes de la portabilité de Linux et du hardware, utilisé pour un système embarqué intégrant cet Os, sont décrits dans le chapitre 4.

La théorie concernant l'optimisation d'un Linux embarqué est traitée dans le chapitre 5.

[1] Ordinateur équipé d'un processeur 80x86, nommé aussi compatible IBM

Le chapitre 6, *Processus de Construction d'un système Linux minimal*, décrit avec précision l'installation d'un système Linux minimal sur un système embarqué de type Pc industriel compatible x86. Différentes configurations y seront abordées.

Le dernier chapitre, C*onclusion,* passe en revue les points importants. Avant de conclure, une validation des objectifs atteints, pour permettre de fixer des objectifs futurs, est proposée au lecteur.

La dernière partie de ce rapport est constitué des annexes qui sont très importants pour comprendre le corps du rapport. Ces annexes peuvent paraître, de prime abord, d'une taille exagérée. Après maintes réflexions, j'ai cependant décidé d'y mettre l'ensemble des informations utiles dans le cadre d'une continuation ou de l'utilisation du projet. De ce fait, tous les programmes et documents de ce rapport sont disponible sur le Cd-Rom se trouvant en dernière page.

Conventions utilisées

Voici la liste des conventions typographiques utilisées dans ce rapport.

```
TheFonctionCode
```

Cette police est utilisée pour l'ensemble des codes sources, fichiers scripts, et fichiers de configuration.

Le texte en italique
Il est utilisé pour le nom des répertoires, des fichiers, des programmes, ainsi que pour mettre en évidence certains termes.

Remerciements

Je remercie mon professeur de travail de diplôme, Monsieur Weber Peter, pour sa disponibilité durant ces 12 semaines. J'adresse également mes remerciements à mon expert, Monsieur Marc-André Baillifard, sans lequel je n'aurais pas pu obtenir le matériel hardware qui fut nécessaire pour valider les théories exposées dans le cadre de l'optimisation de Linux pour les systèmes embarqués.

Enfin, j'adresse mes remerciements à l'ensemble de la communauté Linux d'Internet. L'optimisation de l'Os Linux fait appelle à un grand nombre de connaissances. Il m'a fallut quelques fois solliciter les programmeurs du monde Linux en utilisant les groupes de discutions et l'e-mail. Et c'est avec étonnement que j'ai constaté, qu'à chaque fois, mes questions ont abouti à une réponse. Ce qui m'a permis dans certains cas de progresser rapidement.

1 Introduction

Les progrès techniques de ces dernières années ont permis de démocratiser l'utilisation de systèmes à microprocesseurs. Ils font parti de notre vie de tous les jours. Souvent, sans s'en rendre compte, les individus interagissent avec ces merveilles de la technologie moderne. L'industrie de l'automobile les utilise à foison pour piloter le mécanisme de freinage ou pour interagir avec le conducteur au moyen d'un écran graphique. Ces machines à microprocesseurs sont également utilisées dans le monde de l'industrie afin de réguler un processus chimique ou d'automatiser une chaîne de montage… Les exemples concrets pourraient aisément remplir plusieurs pages A4. Jusqu'il y a quelques années, chaque solution possédait une architecture logiciel spécifique. Ainsi, pour tous ces systèmes, il fallait former du personnel et écrire des applications propriétaires.

Durant ces années de progrès technique, naquit un système d'exploitation libre destiné à équiper de simples Pc de bureau classiques. Cette Os dénommé Linux, du nom de son créateur, fit rapidement parler de lui de part sa stabilité et sa gratuité. Les systèmes embarqués devenant de plus en plus puissant, un groupe de développeurs eut l'idée d'y inclure Linux. Sa popularité, sa libre distribution, sa conception généreuse et tous les avantages liés à l'emploi d'un Os multi-tâches firent de cette combinaison une solution géniale. Cependant, ce système d'exploitation n'était pas prévu, lors de son développement, pour équiper de telles machines. En outre, les années aidant, il prit un sérieux embonpoint, suivant dans leur escalade de puissance les Pc qui trônent fièrement sur les bureaux.

Adapter un Linux moderne aux contraintes d'un système embarqué n'est plus une mince affaire. Ce rapport a pour but de décrire les différentes manipulations à effectuer afin d'optimiser Linux pour qu'il puisse être intégré à un système minimal compatible x86. Il convient avant de traiter du sujet proprement dit, d'apporter quelques informations sur le fonctionnement de Linux. Ensuite, il sera possible d'exposer précisément et de façon théorique les parties qui ont fait l'objet d'une optimisation. Afin, l'application pratique des conseils sera présentée. Ce dernier point, débouchera sur une mini distribution, pour systèmes embarqués, dénommées ELISE[2].

1.1 Applications concrètes

L'utilisation d'un système d'exploitation, dans un système embarqué du type compatible x86, offre une multitude d'avantages. En premier lieu, n'importe quel programmeur peut développer des applications pour un Os comme Linux. La programmation se faisant principalement en langage C, il n'est plus nécessaire de devoir former du personnel pour chaque plate-forme hardware. Actuellement, bien des systèmes embarqués disponibles sur le marché sont au standard compatible x86. L'emploi de tels systèmes facilite le développement des applications. Bien que le standard x86 cumul plusieurs défauts, hérités du mode de compatibilité ascendante, il est assurément l'architecture la mieux connue sur toute la planète. Linux est de plus en plus présent dans les systèmes de petites tailles. Il se

[2] ELISE : Embedded LInux SEtup

retrouve dans une gamme importante d'agenda numérique tel que celui visible sur la figure 1. Ses capacités intrinsèques pour les applications réseau en font un Os idéal pour les petits serveurs HTTP[3]. Linux profite également d'outils de téléphonie très élaborés.

Figure 1 : PDA HNT Exelien Linux

En outre, les capacités multi-tâches de l'Os sont très utiles pour des applications d'automatisation ou de régulation complexes. Il possède également des aptitudes graphiques excellentes. En fait, Linux est une très bonne solution tant au niveau technique qu'au niveau commercial.

A l'instar du fabricant d'automobile Citroën qui produisit une automobile badgée Windows CE (faisant référence au système d'exploitation pilotant les différents accessoires de l'habitacle) ne serait-il pas flatteur d'apercevoir une grosse berline portant l'inscription *Powered by Linux* ?

Figure 2 : VR Linux

[3] hypertext transfert protocol

2 Approche et Cahier des charges

2.1 Cahier des charges

- Définir les éléments de Linux qui peuvent être optimisés
- Donner des solutions pour chaque élément
- Tester les solutions proposées
- Expliquer chaque modifications dans leur application concrète
- Créer une mini distribution Linux pour systèmes embarqués x86
- Proposer des applications

2.2 Approche

Il a fallut environ dix ans pour arriver à la version de Linux connue aujourd'hui. La quantité du code, la complexité du système peuvent sembler être, au premier abord, une montagne dont le franchissement est insurmontable. Cependant, ce projet prouve le contraire. Linux peut être représenté comme un assemblage de divers éléments tels que le noyau de l'Os, les librairies du système, les applications… Si le problème est pris comme un tout, alors il se peut qu'il soit insurmontable. Il faut pour en arriver à bout, déterminer l'ensemble de ces différentes pièces de puzzle et opérer sur chacune un traitement particulier.

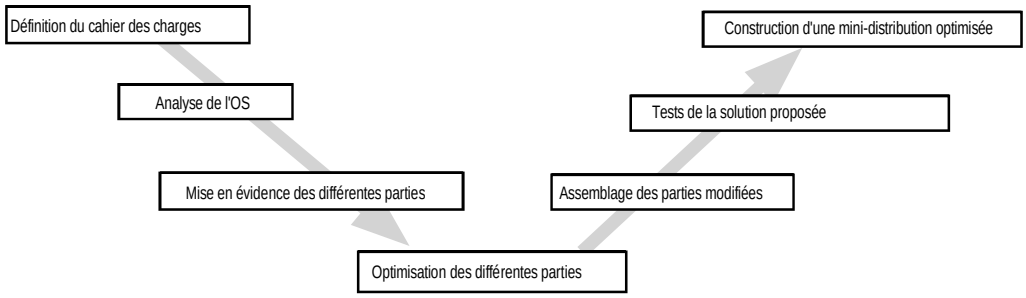


Figure 3 : Approche

Au final, cette immense construction pourra être à nouveau assemblée, afin de produire un système d'exploitation conventionnel mais optimisé pour les systèmes embarqués.

3 L'Os Linux

«Le système d'exploitation ou logiciel-système est l'ensemble des programmes assurant la gestion des tâches élémentaires qui constituent la base du fonctionnement de tout ordinateur. Il gère l'ensemble des ressources de l'ordinateur (RAM[4], mémoire de masse, processeur central, périphériques,...) qu'il met à la disposition des programmes directement accessibles par l'utilisateur. »

3.1 Introduction

Linus B.Torvald est l'inventeur de ce système d'exploitation entièrement gratuit. Au début des années 90, il voulait mettre au point son propre système d'exploitation pendant ses loisirs. Linus Thorvald avait pour intention de développer une version d'Unix pouvant être utilisé sur une architecture de type 80386.

Le premier clone d'Unix, fonctionnant sur Pc, a été Minix, écrit par Andrew Tannenbaum. Linus Thorvald décida d'étendre les possibilités de Minix, en créant ce qui allait devenir : Linux. Amusées par cette initiative, de nombreuses personnes ont contribué à aider Linus Thorvald dans la réalisation de ce système, si bien qu'en 1991 une première version de l'Os a vu le jour. C'est en mars 1992 qu'a été diffusée la première version ne comportant quasiment aucun *bug*.

Avec le nombre croissant de développeurs travaillant sur ce système, celui-ci a rapidement pu intégrer tous les outils présents sous Unix. De nouveaux outils pour Linux apparaissent désormais à une vitesse vertigineuse.

L'originalité de ce système réside dans le fait qu'il n'a pas été développé dans un but commercial. En effet, aucune ligne de code n'a été copiée des systèmes Unix originaux (Linux s'inspire de nombreuses versions d'Unix commerciales: BSD UNIX, System V). Ainsi, tout le monde, depuis sa création, est libre de l'utiliser mais aussi de l'améliorer.

3.1.1 Les capacités de l'Os

Linux est un système multitâche, Multi-Utilisateur, comme toutes les autres versions de Unix. Il est conforme aux standards IEEE[5] POSIX.1, System V et BSD. Plusieurs personnes peuvent être *loggées* sur la même machine en même temps et exécuter des programmes simultanément de surcroît. Linux supporte également les systèmes à processeurs multiples (qui deviennent de plus en plus courant). Linux est, bien sûr, compatible avec une gamme importante de standards Unix. Le code source du kernel, les pilotes, les programmes d'utilisateur, etc. sont distribués gratuitement. Linux prend

[4] Random Access Memory
[5] L'IEEE (www.ieee.org) est un organisme de normalisation

également en charge le principe des *consoles virtuelles*, qui permet de simuler la présence de plusieurs terminaux sur un même ordinateur. Linux peut être utilisé comme serveur, et c'est pourquoi ce genre d'option est inclus dans le kernel.

Le protocole TCP/IP est pris en charge par le kernel. Cela inclut les pilotes pour une multitude de cartes Ethernet. Tous les clients et serveurs sont également disponibles sous Linux, que ce soit FTP, Telnet, SNMP, SMTP, SLIP, PPP, NFS, etc.

Le kernel[6] est le cœur du système d'exploitation. C'est le code qui contrôle l'interface entre les applications d'utilisateur et les pilotes de périphériques, le *scheduling* des processus pour le multitâche, et beaucoup d'autres aspects du système. Le noyau n'est pas un processus séparé des autres. En fait, le kernel est un ensemble de routine constamment en mémoire, dont chaque processus a accès.

Linux peut également effectuer émuler un FPU[7] 387 (coprocesseur arithmétique). Cette particularité permet d'effectuer certains calculs qui seraient autrement impossibles sur d'anciennes machines. Le noyau a été développé afin d'utiliser le mode de mémoire protégé qu'offrent les processeurs x86 d' Intel. C'est ce qui fait que Linux est un système multitâche très efficace.

Linux permet également d'utiliser de la mémoire virtuelle, ce qui donne la chance a certaines machines possédant peu de RAM d'exécuter des applications de taille importante en utilisant de l'espace de disque dur comme de la mémoire RAM physique.

3.2 Structure de l'OS

Linux ce n'est pas seulement un noyau, c'est aussi des librairies, et des applications. La figure ci-dessous expose la structure de l'Os.

[6] Autre nom donné au noyau
[7] Floating Point Unit

Figure 4 : Structure de Linux

Linux est dérivé des Os du monde Unix. Il en conserve le système de fichiers qui est nécessaire à son exécution.

3.2.1 Architecture du noyau

La plus grande partie des noyaux Unix sont monolithiques : « *Chaque couche du noyau est intégrée dans le programme que constitue globalement ce dernier et s'exécute dans le mode noyau pour le compte du processus en cours.* » Par opposition, les systèmes d'exploitation à base de micro-noyau nécessitent un kernel ne possédant que quelques fonctionnalités. En général, il s'agit de quelques primitives de synchronisation, d'un Ordonnanceur simple et d'un mécanisme de communication entre processus. Différents processus systèmes, s'exécutant au-dessus du micro-noyau, fournissent les autres fonctions d'un système d'exploitation, comme l'allocation mémoire, les gestionnaires de périphériques, les gestionnaires d'appels système, etc.

Actuellement, les recherches s'orientent vers les micro-noyaux. Ceux-ci sont, en général, plus lents que les systèmes monolithiques. En effet, la transmission des messages entre les différentes couches du système d'exploitation demande plus de ressources. Cependant les micro-noyaux présentent des avantages théoriques sur les systèmes monolithiques. Du fait du nombre restreint de fonctionnalités d'un micro-noyau, le portage vers diverses architectures se voit simplifié. En outre, quiconque souhaite adapter un périphérique ou ajouter une fonctionnalité au noyau, peut le faire sans avoir à modifier le code du kernel.

Linux est catalogué dans la catégorie des noyaux monolithiques. Il profite cependant de certains avantages d'un micro-noyau grâce à l'utilisation des *modules*. Un *module* est un fichier objet dont le code peut être lié au kernel (et en être supprimé) en cours d'exécution. Ce fichier implémente un système de fichier, un pilote de périphériques ou une autre fonctionnalité de haut niveau. Néanmoins, contrairement au micro-noyau, le *module* ne s'exécute pas dans un processus spécifique. Il est exécuté en mode noyau comme toutes

les fonctionnalités qui y sont liées statiquement. C'est pourquoi, il est possible de nommer le noyau Linux de *« noyau monolithique modulaire »*

3.2.2 Système de fichiers

Linux, comme tous les systèmes d'exploitation du type Unix, est indissociable de son système de fichiers. Celui-ci fait partie intégrante du processus de fonctionnement de l'Os. Un rapide survole des caractéristiques importantes de ce dernier est réalisé dans le chapitre qui suit.

Fichiers

Un fichier est un conteneur d'informations dont la structure est une séquence d'octets, le noyau n'interprétant pas le contenu des fichiers. L'interprétation du contenu des fichiers est réalisée par les librairies.

Du point de vue de l'utilisateur, les fichiers sont organisés selon un arbre comme illustré dans la figure ci-dessous. L'arborescence détaillée est visible en annexe *L'arborescence FHS des répertoires* page 118.

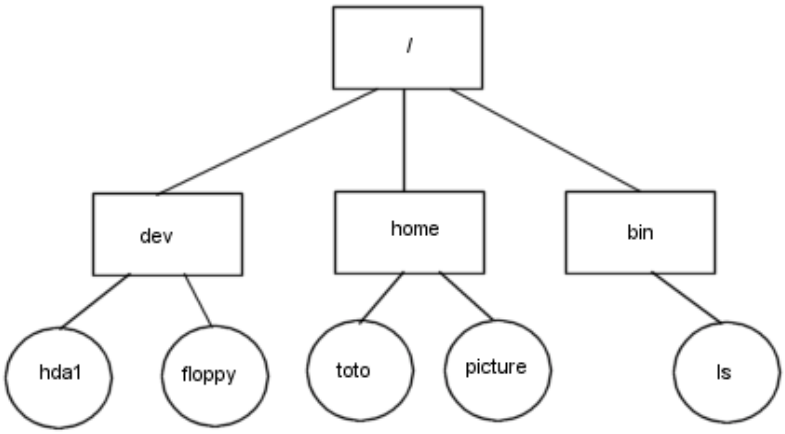


Figure 5 : Exemple d'arborescence

Chaque nœud de l'arbre représente un répertoire. Un nœud de type répertoire contient des informations à propos des fichiers et des répertoires situés sous ce nœud. Le répertoire correspondant à la racine de l'arbre est nommé *ROOT* et sa notation est faite par une barre oblique, le slache : /.

Liens du système et liens symboliques

Un nom de fichier, contenu dans un répertoire, est nommé *lien système vers un fichier*. Le même fichier peut avoir plusieurs liens dans le même répertoire, et dans d'autres répertoires et donc plusieurs noms de fichier. Les liens systèmes possèdent deux problèmes. Premièrement, ils ne peuvent pas être créés pour des répertoires. Dans le cas contraire, cela pourrait entraîner la création de boucle dans la structure des répertoires

rendant impossible le repérage d'un fichier. Enfin, les liens peuvent être créés seulement sur la même partition. Ce qui pose problème sur les systèmes modernes possédant souvent plusieurs partitions différentes.

Pour palier à ces désavantages, il existe le lien symbolique. Le lien symbolique est en fait un petit fichier qui contient le chemin vers un autre fichier.

Types de fichiers

Les fichiers peuvent être de huit types différents :

- Fichier régulier
- Répertoire
- Lien symbolique
- Fichier de périphérique orienté bloc
- Fichier périphérique orienté caractère
- Tube ou pipe
- Socket

Les trois premiers types sont des parties d'un système de fichiers comme le système de fichier Ext2[8], par exemple. Les fichiers de périphériques sont liés aux périphériques d'entrées/sorties et à leurs pilotes. Lorsqu'un programme accède à un périphérique, il dialogue directement avec le fichier correspondant. Les tubes et les sockets sont des fichiers spéciaux utilisés pour la communication entre les processus.

Inode

Le monde Unix distingue un fichier d'un descripteur de fichier. Tous les fichiers, à l'exception des fichiers de périphériques et des fichiers spéciaux, sont constitués d'une chaîne de caractères. Ils ne possèdent aucune information de contrôle telles que la longueur ou un marqueur de fin de fichier (EOF). Tous les renseignements nécessaires au système de fichier pour la gestion des fichiers sont contenus dans une structure de données nommée *inode*. Chaque fichier possède sa propre *inode*.

L'*inode* contient l'information de droit d'accès du fichier par les groupes et les utilisateurs. Ceci permet une sécurisation des données au niveau de leur système de stockage.

[8] Extended 2 : format de partition Linux

3.3 Les permissions

3.3.1 Généralités

Les permissions attribuées aux fichiers sont essentielles à la sécurité du système. Elles se rapportent aux opérations applicables à un fichier et sont au nombre de trois sous Unix :

- Lecture : la permission de lecture autorise la consultation du contenu d'un fichier.
- Ecriture : la permission d'écriture autorise à consulter le contenu d'un fichier.
- Exécution : la permission d'exécution autorise à exécuter un fichier en tant que programme, ou bien d'accéder à un répertoire.

Lorsqu'un fichier est créé, le système lui attribue des permissions par défaut qui , la plupart du temps, font l'affaire. D'ordinaire, il octroie les droits de lecture et d'écriture, alors que tous les autres utilisateurs ne pourront que lire le fichier.

Dans le cas d'un répertoire, ces permissions ont des implications différentes :

- Lecture : la permission de lecture autorise à consulter le contenu d'un répertoire
- Ecriture : la permission d'écriture autorise à ajouter ou à supprimer des fichiers dans le répertoire.
- Exécution : la permission d'exécution autorise à se déplacer dans ce répertoire

3.3.2 Propriétaires et groupes

Pour permettre le partage de la machine, Linux découpe la population des utilisateurs en trois catégories : le propriétaire, le groupe, et les autres. Ces derniers sont les utilisateurs qui ont accès au système et ne sont ni propriétaire du fichier, ne membre du groupe auquel le propriétaire appartient. Les groupes permettent de créer des ensembles d'utilisateurs jouissant d'un accès commun à des données.

A chaque fichier sont associés un propriétaire et un groupe. Le propriétaire est généralement l'utilisateur qui est affecté à ses fichiers.

3.3.3 Représentation des droits d'accès

L'administrateur doit avoir la possibilité de gérer les droits d'accès. Les commandes *chown* et *chgrp* changent respectivement le propriétaire et le groupe associé au fichier. La modification des permissions est plus compliquée. Elle est effectuée à l'aide de l'utilitaire *chmod*. Celui-ci reçoit en paramètre un *mode absolu*. Pour comprendre ce qu'est le mode absolu, il faut penser en terme de numérotation octale. Un mode typique contient trois caractères, correspondant aux trois niveaux de permission. Dans chaque niveau, trois bits permettent de représenter les permissions de lecture, d'écriture et d'exécution.

Utilisateurs			Groupe			Autres		
lecture	écriture	éxecution	lecture	écriture	éxecution	lecture	écriture	éxecution
400	200	100	40	20	10	4	2	1

Figure 6 : Bits de permission

Pour donner la permission de lecture à tout un chacun, il faut associer le bit adéquat à chaque niveau : 400 , 40, 4 .

```
chmod 444 MyProg
```

3.4 Le démarrage

Le démarrage ou « boot » d'un système d'exploitation est une phase délicate. Il s'agit de mettre en service un programme relativement capable, le noyau, à partir d'un autre extrêmement limité, le BIOS[9]. Il est intéressant de connaître quelques étapes du lancement de Linux. Ayant été développé pour des systèmes x86, la description du « boot » est faite pour ce genre d'architecture.

3.4.1 Procédure de boot

La première étape consiste en l'exécution du code BIOS. Celui-ci est indépendant de tout système d'exploitation. En général, un Pc commence par l'instruction située à l'emplacement mémoire 0x0F0000h – 0x0FFFFFh, qui contient le BIOS. Ce code effectue une recherche des différents périphériques tels que la carte graphique, le contrôleur de disque dur etc. Puis, il effectue des tests sur certains circuits vitaux comme le contrôleur DMA[10], la mémoire RAM et le contrôleur d'interruption. Ensuite, le système initialise la table des vecteurs d'interruption dans le premier segment de la mémoire.

Ensuite, grâce à l'information du support de « boot » contenue dans la configuration du BIOS (disque dur, lecteur de disquettes, mémoire Flash), celui-ci sait sur quel support se trouve le système d'exploitation. Quel que soit le support de démarrage, le premier secteur est lu. Il s'agit du MBR[11] dans le cas d'un disque dur et du secteur d'amorçage pour une disquette. Le code situé dans le MBR ou le secteur d'amorçage est chargé à l'emplacement mémoire 0x07C000h. Ce mini programme, chargé de lancer le kernel, est appelé gestionnaire de « boot ». Les plus connu du monde Pc sont *Lilo* et *SysLinux*. Dans le monde Linux le BIOS n'est présent qu'au démarrage du système. En effet, le noyau utilise ses propres pilotes de périphériques.

[9] Basic Input Output System
[10] Direct Memory Access
[11] Master Boot Record

3.4.2 Démarrage de l'OS

Dans les premiers temps de l'histoire de Linux le noyau était stocké sans être compressé. Il était chargé tel quel en mode réel, c'est-à-dire dans les 640 premiers kilos octets, et c'était une limite pour sa taille.

Assez rapidement, (au alentour de la version 0.99) il a été mis au point un mécanisme de chargement plus fin. Le noyau était toujours chargé dans ces 640 [ko], mais compressé. L'image commence par un code capable de passer du mode réelle au mode protégé, donc d'utiliser la mémoire haute (au-delà du premier méga octet) puis de réaliser la décompression. Il s'agit de la *zImage*. Avec cette innovation la machine doit disposer d'au moins deux mégas octets de mémoire RAM. Il faut un méga au-dessus du premier méga pour accueillir le noyau décompressé.

La limite de taille pour un noyau *zImage* est de 508 [ko]. La différence pour atteindre les 640 [ko] est utilisée pour le code de décompression. Le kernel Linux est un logiciel et comme tous les logiciels, son évolution l'a poussé à prendre du volume. Les 508 [ko] qui semblaient suffisant se sont vite avérés trop limitatifs. A ce sujet, dans les sources du noyau */linux/arch/i386/boot/bootsect.S* , il est possible de lire ceci : «currently system is at most (8*65536-4096)bytes long. This should be no problem, even in the future. I want to keep it simple. This 508 kB kernel size should be enough »; comme quoi Linus Torvalds peut faire la même bévue que Bill Gates, dix ans plus tard.

Pour palier à cette limite, il a été inventé une variante de la *zImage*, la *bzImage* Là encore, le noyau est compressé, mais prévu pour être chargé directement au-delà du premier méga octet. Cela suppose une *intelligence* supérieure du chargeur car il doit pouvoir faire la transition du mode réel au mode protégé. De plus, la taille de la mémoire au-delà du méga octet doit être au minimum égale à : *noyau compressé + noyau décompressé*. Il est noté que l'image *bzImage* est compressée avec le programme *bgzip* plus efficace que *gzip* utilisé pour la *zImage*.

3.5 Linux en tant qu'OS temps réel

« Se dit d'un système devant répondre aux sollicitations de son environnement physique dans des délais précis ».

Il n'est pas possible de qualifier Linux d'Os temps réel. Cependant, la nécessité de disposer d'un système d'exploitation temps réel a poussé de nombreux développeurs à modifier Linux pour le doter des attributs des Os temps réel.

Un système Linux classique utilise un système de temps partagé. Ce système ne convient donc pas pour le temps réel. Un Linux spécifique a donc été créé : *RT Linux.* Dans le principe, pour rendre un système temps réel, il faudrait modifier le kernel Linux pour y intégrer la notion de restriction temporelle afin de le rendre "prévisible". La solution choisie par l'équipe de développement de *RT Linux* consiste à construire un petit noyau (indépendant du kernel Linux), intégrant un gestionnaire de tâches. Le noyau de Linux tourne au-dessus de celui-ci, partageant le temps processeur avec les autres tâches temps réel. Le noyau Linux devient alors une tâche du système temps réel. Dans la pratique, le

gestionnaire de tâche est un *module*. Il est donc possible de le charger ou le décharger à volonté comme n'importe quel *module* du système.

Pour ajouter les fonctionnalités temps réel à un système Linux, il faut appliquer un patch aux sources du noyau. Cette manipulation n'est pas destinée à inclure les notions temps réel dans le kernel, mais simplement à ajouter un *module*. Une fois compilé, le noyau est plus ou moins identique à la version non *patchée*, si ce n'est qu'il est ensuite prêt à accueillir le support temps réel. Il existe pour l'instant des patchs pour les versions 2.0.x, 2.2.x et 2.4.x du noyau Linux. Destiné au système embarqué, *RT Linux* fonctionne avec quartes mégas octets de mémoire RAM. *RT Linux* n'a malheureusement pas été testé. Pour de plus amples renseignements, il convient de consulter le site Internet de la société qui développe *RT Linux* : http://www.FSMLabs.com. Le patch y est disponible en libre téléchargement.

4 Linux, OS embarqué

4.1 Choisir Linux

Pourquoi intégrer Linux dans un système embarqué quant il existe une multitude d'autres systèmes d'exploitation pour ce genre de matériel ? Ce chapitre apporte les éléments de réponse à cette question.

4.1.1 Avantages

Avantage d'un système d'exploitation

Linux, comme un grand nombre d'autres systèmes d'exploitation, est multi-tâches. Il est possible d'y exécuter plusieurs processus qui se partagent les ressources[12] de la machine. De plus, Linux possède également l'attribut *préemptif*. Il peut, en effet, stopper, à tous moments, n'importe quelle application pour passer la main à la suivante. Et enfin, Linux est multi-utilisateurs.

Coût du software

Linux ne coûte rien ! Et c'est bien là son principal avantage : rien pour acheter le système d'exploitation, aucunes royalties à verser en cas de commercialisation. Les systèmes d'exploitation embarqués commerciaux de WindRiver ou QNX entraînent un coût important seulement pour pouvoir commencer le développement (achat des outils de développement et des licences) En outre, ces sociétés obligent le payement de royalties en cas de vente du produit modifié.

Stabilité

Tout le monde a déjà entendu des anecdotes au sujet de la stabilité de Linux comparé au système de Microsoft, Windows NT. Des programmeurs du Net ont créé *l'Uptime Project*[13] qui consiste à mesurer la fréquence de « plantage » de divers systèmes d'exploitation tels que Linux, Windows NT. Ils en tirent un *Uptime Number* qui est en fait le nombre de jours pendant lesquels le système d'exploitation a fonctionné correctement Après plusieurs années d'observation, il a été constaté que :

- Windows NT obtient un Uptime de 76 jours
- Linux obtient un Uptime de 575 jours
- BSD obtient un Uptime de 1411 jours

Windows avec son *Uptime* de 76 jours se place en position 277, bien en dessous de la troisième place de Linux.

[12] cpu, mémoires,…
[13] www.uptimes.net

Portabilité

Linux a été porté vers des dizaines de plate-formes différentes. Les sources de Linux, en libre diffusion, intègrent déjà du code pour plusieurs architectures de processeurs. Le chapitre *Portabilité du code sur différentes architectures* en page 26 décrit plus en détail le problème de la portabilité.

Un marché

Les technologies actuelles évoluent d'une façon fulgurante. Le développement d'un projet ne doit pas durer des années, sinon le produit est totalement dépassé lors de sa disponibilité sur le marché. En choisissant Linux, un concepteur dispose d'un système d'exploitation qui évolue constamment. Là où QNX doit employer des dizaines de développeurs pour maintenir leur produit à jour, Linux dispose de milliers de programmeurs motivés. De plus, Linux ayant pris une importance énorme, les nouvelles technologies sont souvent conçues sur ce système d'exploitation. Linux peut donc en disposer en premier. Enfin, la conception d'applications destinées à des systèmes utilisant Linux peut être faite, sur la plate-forme de développement, sous Linux.

4.1.2 Désavantages

Linux ne collectionne pas seulement les avantages. Fruit du travail de programmeurs perfectibles, il contient quelques désavantages qu'il est judicieux de connaître.

Sécurité

Amélioré par de nombreux développeurs, Linux semble être devenu un système d'exploitation sécurisé au maximum. En fait, la disponibilité des sources fournit à une personne malintentionnée le moyen de trouver une faille dans le système d'exploitation.

Taille de l'OS

Linux n'a pas été conçu pour les systèmes embarqués. Il était destiné lors de sa conception, au début des années 90, à équiper des Pc de bureau de type AT 386 compatible x86, disposant d'un disque dur et de quatre mégas octets de mémoire RAM. En outre, Linux exige un système de fichiers. Cette caractéristique est héritée du monde Unix. Son noyau n'est pas vraiment compact. Il pèse au minimum 400 [ko], s'agissant de la dernière version 2.4.0. Comparé aux 80 [ko] du noyau du système d'exploitation embarqué *Ecos*, Linux n'est pas un poids plume. De plus, lorsque Linux est décompressé en mémoire, il utilise facilement un méga octet de mémoire RAM !

Open source

L'open source ou GPL[14] n'est pas qu'un avantage. Certes, cela permet de pouvoir jouir de softwares gratuitement et même de pouvoir les distribuer en y retirant des profits. Cependant, tous les programmes soumis à la GPL doivent toujours être distribués sous licence GPL. Ainsi, une entreprise n'a pas le droit de récupérer Linux, d'y apporter certaines modifications puis de le revendre sous une licence plus restrictive. Tout programme dérivé du source Linux doit être diffusé selon la GPL. Le portage de Linux pour une plate-forme spécifique est un travail important, qui ne peut pas être protégé par une licence spéciale. Chacun pourra distribuer l'Os, le modifier et même le vendre !

[14] General Public License

4.2 Le Hardware

« A la conception d'un système embarqué, le but est d'obtenir une configuration de machine la plus petite possible permettant de faire marcher les applications désirées. Si aucun problème de taille (mémoire, stockage, …) n'est rencontré, c'est que le système choisi n'est pas le bon. »

Ce rapport traite principalement du système d'exploitation Linux et de son optimisation en vue d'être intégré à un système embarqué aux ressources limitées. Lors de la conception d'un tel produit, il faut avoir à l'esprit que le hardware et le software sont intimement liés. C'est pourquoi, ce chapitre apporte quelques informations quant au choix des composants hardware pour un système destiné à recevoir Linux.

4.2.1 Systèmes finis

Le marché des systèmes embarqués a pris, depuis quelques années, une grande importance. Il existe aujourd'hui un grand nombre de petits ordinateurs au ressources limitées et à la taille réduite. Ce hardware spécialisé est utilisé dans de multiples applications toutes plus différentes les unes que les autres : robotique, régulation, télécommunication…

Dans le monde des compatibles x86, la première place est détenue par le standard Pc-104 qui détermine les caractéristiques d'un système x86 aux dimensions moindres. Ce standard est peu à peu remplacé par la norme Pc-104+ qui impose de doter les cartes d'un bus PCI[15]. D'autres systèmes de compatibles x86 miniaturisés existent, tels que les Pc industriels. L'utilisation d'un système x86 peut être motivée par le fait que sont disponibles, pour ce système, un grand nombre de périphériques, d'applications et de personnel formé.

De nombreuses autres architectures systèmes sont également disponibles (StrongArm, PowerPC…). Aujourd'hui, la plus grande partie des solutions embarquées récentes supportent Linux. L'Os est souvent livré avec l'ordinateur. Si ce n'est pas le cas, les drivers, pour les périphériques exotiques, sont généralement inclus. Cependant, rien n'empêche un développeur de concevoir son propre système embarqué.

4.2.2 Processeurs

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. Le portage de Linux ayant été fait pour pratiquement toutes les familles de processeurs, il n'est pas nécessaire de partir de zéro.

Il faut cependant remarquer qu'il est bien plus facile de travailler sur un processeur équipé d'une MMU[16]. 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.

Le noyau *uClinux* permet l'utilisation de Linux sur des processeurs dépourvus de MMU. Les applications, qui fonctionnent parfaitement avec un noyau standard, risquent de

15 Peripheral Communications Interconnect
16 Memory Management Unit

rencontrer certains problèmes. Le mécanisme de protection de la mémoire étant fait au niveau logiciel, une application peut écrire dans les parties de la mémoire réservées au kernel. Il en résulte un blocage total du système. Et enfin, le système d'exploitation *uClinux* est encore à son stade de développement. Il convient donc d'être très prudent si son utilisation est envisagée.

4.2.3 Mémoire RAM et mémoire de masse

Linux ne sera probablement jamais autant compact qu'un système d'exploitation commercial conçu spécialement pour les systèmes embarqués. Avec la diminution du prix des composants de mémoires, il est actuellement réaliste d'envisager l'utilisation de Linux en tant qu'OS embarqué. Le chapitre suivant apporte quelques informations quant au choix de la mémoire RAM et de la mémoire de masse.

Mémoire RAM

« Mémoire non permanente, rapide, servant à stocker plus ou moins temporairement les informations, le code ou les données »

Il serait trop long de disserter sur quel type de mémoire RAM utiliser. L'information utile pour un développeur se trouve dans la quantité de mémoire nécessaire pour faire fonctionner Linux. Répondre à cette question d'une manière précise est très difficile voir même impossible. En effet, comme décrit dans la suite de ce rapport, la quantité de mémoire RAM demandée par Linux dépend de la configuration de l'Os et de son environnement. Par exemple, un système n'exigeant aucune possibilité d'affichage graphique est bien moins gourmant en mémoire qu'un système à fenêtre. L'utilisation d'un certain type de systèmes de fichiers demande une taille de mémoire RAM, ou de mémoire de masse plus faible qu'un autre système. Mais, le facteur le plus important, influençant la taille de mémoire, se situe au niveau de l'application qui fonctionne sur la machine. Par exemple, un serveur *http* qui permet la connexion de dizaine d'usagers, nécessite beaucoup plus de ressources mémoire qu'un programme de régulation.

Les chiffres

Il faut savoir, que le noyau Linux, au moment du démarrage, sur une machine x86, se décompresse en mémoire RAM pour occuper environ 900 [ko] pour un noyau 2.4.0 minimal de 450 [ko] compressé. A cela il faut ajouter 500 [ko] réservés pour les données du noyau (buffer, stack,…). La taille d'un noyau 2.4.0 décompressé, pour un système de bureau multimédia, peut facilement atteindre les trois mégas octets.

Sur un système x86, le noyau 2.4.0 est décompressé en mémoire à l'adresse 0x00100000, soit à partir du premier méga octet. C'est pourquoi, il faut disposer de plus de deux méga octets pour lancer Linux. A cela, il faut ajouter l'espace mémoire nécessaire au chargement des applications spécifiques. La barrière des trois mégas octets est rapidement atteinte. La communauté Linux semble être d'accord pour affirmer qu'une taille de 4 [Mo] suffit pour exploiter un Linux minimal, d'une manière confortable.

Il est possible pour diminuer la taille en mémoire RAM, d'utiliser un noyau plus ancien comme 2.2.x , 2.0.x ou encore un noyau 0.x.x. qui compte tenu de sa petite taille, peut-être décompressé dans les premiers 640 [ko]. Ces possibilités n'ont pas était examinées dans ce rapport. En effet, ces noyaux plus anciens qui apporterait un gain de place en mémoire, ne

profitent pas des dernières mises à jour. Si certaines options du noyau 2.4.x ne sont pas indispensables pour un système embarqué, d'autres ajouts, pour améliorer la sécurité, sont très utiles. En outre, le noyau 2.4.x intègre la prise en charge des disques et composants Flash dont l'emploi dans un système embarqué semble obligatoire.

Mémoire de masse

« Ensemble de tous les supports de stockage autonomes et statiques, c'est-à-dire disposant d'une interface normalisée et capable de préserver les données même sans apport d'énergie. »

La mémoire de masse sur un Pc de bureau désigne le disque dur, le cd-rom et le lecteur de disquettes. Il s'agit en fait d'un espace mémoire de stockage, plus important, mais aussi plus lent en lecture et écriture que la mémoire RAM. Il est important de retenir, que Linux, comme tous les Os Unix, est indissociable de sa structure de fichiers et nécessite pour cela l'emploi d'une unité de stockage. Celle-ci mémorise aussi le système d'exploitation et les applications qui seront chargés en mémoire vive lors de l'exécution. Il existe plusieurs moyens pour offrir un espace de stockage à un système embarqué.

- *L'utilisation d'un disque dur classique 2,5''* : Malgré qu'il s'agisse de la solution la moins coûteuse, dans un système embarqué l'emploi d'un disque dur n'est pas approprié. Sa consommation trop importante, son bruit, sa faible résistance aux chocs aux rayonnements électromagnétiques et aux températures extrêmes limitent son emploi.

- *L'utilisation d'un disque Flash IDE*[17]: Cette solution est avantageuse car elle utilise la même interface physique qu'un disque dur classique, sans posséder les inconvénients de la fragilité et du bruit. Cependant, le coût au méga octet est bien supérieur à celui d'un disque classique. Un autre problème, très peu médiatisé par les fabricants de ce type de périphériques, peut apparaître lors de coupures intempestives d'énergie. Au mieux, seuls quelques blocs seront inutilisables même après un formatage, au pire c'est tout le disque Flash qui deviendra inutilisable… La solution de facilité n'est souvent pas la meilleure !

- *L'utilisation d'un disque Flash non-IDE* : Les fonctionnalités sont les mêmes qu'un disque IDE, seule la connectique change. Le *DiskOnChip* de chez M-Systems, utilisant une interface propriétaire, est livré avec des drivers pour Linux ainsi qu'une documentation détaillée pour son installation. Ce composant électronique est fabriqué à partir de puces de mémoires Flash, avec une logique intégrée de contrôle, qui permet de garantir une bonne résistance en cas de coupure d'alimentation. Ces composants intègrent aussi un BIOS qui permet de *booter* directement avec la mémoire Flash, lorsqu'ils sont utilisés sur une architecture Pc classique. Mais attention, les drivers du fabricant ne sont pas distribués en licence GPL. Pour palier à cet inconvénient, il est possible de se tourner vers un driver OpenSource capable de gérer ce composant. Il s'agit du driver Linux MTD[18] intégré au noyau depuis la version 2.4.X.

[17] Intelligent Device Electronics
[18] Memory Technology Device

- *La dernière solution consiste à utiliser une mémoire Flash brute*. Une mémoire Flash brute ne possède aucun dispositif électronique masquant les détails de fonctionnement. L'utilisation de tels composants est loin d'être simple et il faut dès lors faire appelle à un driver. En effet, la gestion de ces mémoires impose de nombreuses restrictions : elles s'effacent par blocs relativement gros et le nombre d'effacements de blocs est limité[19]. Le MTD cité plus haut gère ce type de composant. Le coût étant bien moindre, par rapport aux solutions Flash IDE ou DiskOnChip, cette solution est idéale pour les systèmes embarqués.

Les chiffres
Choisir une technologie n'est pas tout, il faut encore définir la taille de mémoire de masse nécessaire. Ici aussi, il est difficile de donner un chiffre précis. En effet, la taille pour le stockage du système complet dépend de l'emploi final du système embarqué.

La taille de stockage du noyau 2.4.0 varie entre 400 [ko] à plus d'un méga octet. La grandeur des librairies partagées obligatoires se situe entre quelques kilos octets et plusieurs mégas octets. A cela, il faut ajouter la taille du système de fichiers, les fichiers de configuration, les commandes minimales et le programme d'amorçage. Ceci fournit un système Linux fonctionnel, mais qui ne comporte encore aucune application. Dans le chapitre *Linux optimisé pour x86* page 29 tous ces aspects sont analysés avec plus de détail et différentes solutions sont proposées.

Remarque : Une liste de fabricant de périphériques de stockage est disponible dans la bibliographie page 114.

4.2.4 Périphériques d'entrées/sortie

Le choix des périphériques d'entrées/sorties est immense. Les fonctionnalités désirées détermineront avec exactitude le type d'entrées/sorties à implémenter. De nombreux programmeurs développent et affinent depuis plusieurs années des drivers pour une gamme pratiquement infinie de périphériques. L'exemple le plus parlant est la multitude de pilotes de contrôleurs réseau supportés par Linux. Si un périphérique spécifique, construit dans un chip existe, alors il y a toutes les chances que le driver pour Linux ait déjà été réalisé par le constructeur lui-même ou alors par un programmeur du monde Linux. Pour une économie de temps, d'argent et de soucis choisir un périphérique déjà supporté par l'Os est une solution judicieuse.

4.2.5 Affichage

Linux n'impose pas un système de visualisation strict et peut bien entendu s'en passer. Cette dernière solution est à nuancer. En effet, pour administrer le système, il est plus facile de jouir d'un système d'affichage. Dans ce but, un simple PC peut aisément jouer le rôle d'un terminal par le biais des ports séries. (Comme décrit plus loin, cette option est également forte utile pour l'administration du système)

Pour des applications qui nécessitent une interface graphique, il existe plusieurs solutions de visualisation. La solution la plus économique est l'utilisation d'un écran LCD[20] ou VF[21]

[19] Cette taille peut varier de 100'000 à 1'000'000 de cycle d'effacement
[20]

sur port parallèle ou port série, pour lesquels sont disponibles toute une pléiade de drivers et d'exemples. Les systèmes plus exigeants en terme d'affichage auront recours aux services de contrôleurs graphiques ressemblant aux cartes graphiques présentes dans les PC de bureau. Le sous chapitre *Affichage sous Linux* du chapitre *Linux optimisé pour x86* page 55 expose de manière succincte les différentes solutions logicielles pour permettre l'affichage soit de texte soit de graphisme.

4.2.6 Conclusion

Linux est un système d'exploitation exigeant par rapport à d'autres Os pour systèmes embarqués[22]. Le choix des composants doit se faire en ayant pris connaissance de ces exigences. Le développeur s'évitera beaucoup de nuits blanches en dimensionnant sa carte pour Linux et en choisissant du matériel compatible Linux (drivers). Il veillera à calculer avec le plus d'exactitude possible la quantité de mémoire RAM et de mémoire de masse dont il a besoin. Un léger surdimensionnement n'est pas un luxe dans le cas d'un système destiné à évoluer. Enfin, le processeur choisi possèdera, si possible, une MMU.

4.3 Portabilité du code sur différentes architectures

« Linux is not portable and it probably never will support any thing other than AT-hard disk as that's all I have» LINUS TORVALDS, 25 août 1991

«Linux is tied fairly closely to the 80x86. Not the way to go. Linux is obsolete.» ANDREW TANENBAUM 29 janvier 1992

4.3.1 La portabilité aujourd'hui

Aujourd'hui, Linux a atteint nombre des buts de conception dont certains supposaient à l'origine qu'ils ne pourraient être atteints que sur une architecture micro-noyau.

En construisant un modèle général de noyau fondé sur des éléments communs à des architectures répandues, le noyau Linux a obtenu beaucoup des avantages de la portabilité qui auraient autrement demandé l'ajout d'un niveau d'abstraction, et ce sans souffrir de la même perte de performances que les micro-noyaux.

En autorisant les modules noyau, le code spécifique au matériel peut souvent être confiné, en laissant ainsi le code du noyau hautement portable. Les pilotes de périphériques sont un bon exemple de l'usage effectif de modules noyau pour confiner les spécificités matérielles aux modules. C'est un bon compromis entre mettre toutes les spécificités matérielles dans le noyau de base (ce qui le rend rapide mais non portable) et les mettre dans l'espace utilisateur (ce qui donne un système lent ou instable, voire les deux).

Mais l'approche de la portabilité adoptée par Linux a été aussi bénéfique à tout son environnement de développement. Les décisions qui ont motivé la portabilité ont aussi permis à un large groupe de personnes de travailler simultanément sur des parties de

[21] Vacum Fluorecent
[22] Ecos, ucOS, vxWorks, Qnx …sont des OS destinés aux systèmes embarqués occupant très peu de place comparés à Linux

Linux. Et les modules noyau fournissent aux programmeurs une solution évidente pour travailler indépendamment sur des parties du système qui se doivent d'être indépendantes.

4.3.2 Linux sur différentes architectures

Linux fut développé initialement pour des architectures x86. Rapidement, l'engouement des millions de développeurs pour cet Os *open source* entraîna son portage vers toutes sortes de plate-formes. La migration de Linux fut facilitée par la libre distribution de son code source, par la disponibilité de compilateur *GNU*[23] gcc pour un grand nombre de processeurs, et par son noyau *monolithique* modulaire.

Il est cependant naïf de croire qu'une simple compilation du noyau à l'aide de *gcc*[24] pour le processeur ou l'architecture hardware voulu suffit pour pouvoir l'exécuter. Bien que Linux soit en partie modulaire, son noyau monolithique intègre un grand nombre de pilotes de périphériques. Une simple comparaison entre les sources d'un Linux pour système x86 classique (un PC de bureau) et un ancien système obsolète tel que l'Amiga 4000 à base d'un processeur Motorola 680x0, souligne de grandes différences au niveau du code du noyau. Ces disparités sont dues aux processeurs et à l'architecture des deux machines (bus, gestion de la mémoire, chipset…). Les éléments, qui ne sont pas censés changer entre des Linux pour des machines différentes, sont situés au niveau des appels système du noyau. (API) Quant au drivers pour les périphériques, les bus, et autres matériels, ils doivent être réécrits en totalité. Le but de ce projet n'étant pas de décrire en détail le code source du noyau Linux, les travaux de portage du kernel sont laissés aux spécialistes.

Divers portages

En 1992, Digital présenta une architecture basée sur RISC[25], 64 bits, baptisée Alpha. Le portage de Linux sur celle-ci ne se fit pas du jour au lendemain. Il commença par un simple patch du noyau x86 financé par Digital. C'est avec l'apparition du noyau 2.1.x que Linux Alpha put jouir des avantages du 64 bits au niveau des pilotes et du noyau.

Linux PPC est le portage du système d'exploitation Linux sur le processeur PowerPC. C'est en 1995 que Gary Thomas et Paul Mackerras se sont attelés à porter Linux sur le matériel Power Macintosh. Linux fonctionne actuellement sur presque tous les processeurs PowerPC différents. Il s'est énormément répandu ces dernières années, les utilisateurs de Mac OS cherchant des alternatives à leur système d'exploitation.

Le portage du système d'exploitation Linux sur les processeurs Motorola 680x0 (m68k) se nomme Linux/m68k. Il s'agit du premier projet de portage de Linux sur un processeur Non-Intel. Créé en 1993 par Hamish MacDonald et Greg Harp il vu son noyau adapté pour l'Amiga puis pour les autres ordinateurs 32 bits, tels que Atari ST, Sun et NeXT.

Linux a également été porté sur la célèbre console de jeu de Sony, la PlayStation. Les exemples de portage de Linux pourraient aisément remplir des pages entières. Un détour par Internet confirmera mon affirmation. Le paragraphe qui suit, recense quelques architectures connues supportées. Cette liste évolue de jour en jour.

[23] Libre de droit
[24] Compilateur C GNU
[25] Reduced Instruction Set Computing

Architectures connues supportées

DEC Alpha
http://www.alphalinux.org

Processeur ARM et StrongARM
http://www.arm.linux.org.uk
http://www.uclinux.org

Hitachi SuperH
http://www.m17n.org/linux-sh

IA-64
http://www.linuxia64.org

i386 (x86)
http://www.linux.org
http://www.suse.com
http://www.redhat.com

Motorola 68K
http://www.linux-m68k.org
http://www.uclinux.org

Processeur MIPS
http://www.oss.sgi.com/mips

Motorola PowerPC
http://www.linuxppc.org

Sparc (32 et 64 bits)
http://www.ultralinux.org

4.3.3 Conclusion

La popularité de Linux a entraîné son portage vers une quantité impressionnante d'architectures toutes plus différentes les unes que les autres. Avant de se lancer dans un travail titanesque, afin de l'adapter à son hardware, une petite recherche sur Internet suffira, dans la majeure partie des cas, pour trouver un noyau fonctionnel. Quelques modifications au niveau des drivers valent mieux qu'une gigantesque transformation de son code source qui ne compte pas moins de 4500 fichiers de C et d'assembleur contenus dans près de 270 sous-répertoire ; cela représente quelques deux millions de lignes de code qui occupent plus de 58 [Mo] d'espace disque. (Il s'agit du code source pour toutes les architectures supportées.)

5 Linux optimisé pour x86

5.1 Généralités

Comme décrit précédemment, il existe un grand nombre d'architectures capables d'utiliser Linux en tant que système d'exploitation. Dans le cadre de ce projet le choix s'est porté les compatible x86. Ce choix fut motivé par des contraintes économiques et de temps. En effet, la validation des théories d'optimisation se doit de passer par l'application sur un système existant. Celui-ci n'étant pas bon marché, le budget du projet étant réduit à son minimum, il a fallut trouver un système embarqué sans rien débourser. Les propositions n'étant pas légion, le système généreusement prêté ne pouvait qu'être satisfaisant.

Le chapitre suivant décrit de manière théorique les points importants de l'optimisation de Linux. Ces informations peuvent être appliquées pour des systèmes autres que x86. L'application pratique des conseils d'optimisation, qui suivent, est traitée dans le chapitre *Processus de création d'un système Linux*.

5.2 L'optimisation ?

La construction d'un système Linux embarqué passe par l'analyse du système Linux standard. Le chapitre *l'OS Linux* propose un survole rapide du système d'exploitation. Cette analyse a permis de déterminer les parties modifiables afin d'optimiser le système d'exploitation. Linux étant destiné, lors de sa création, à équiper des machines à base de processeurs Intel 80386, l'optimisation n'ira pas dans le sens des performances (vitesse d'exécution), mais dans celui de la taille occupée en mémoire vive et en mémoire de masse. En effet, à l'heure actuelle bien des systèmes embarqués dépassent largement les performances d'un vieux Pc 386. Par contre, le coût encore prohibitif de la mémoire (RAM et de masse) impose de trouver des solutions quant à la diminution de la taille de stockage de l'Os. Du fait du nombres de fonctionnalités présentes dans Linux, la diminution de la quantité de mémoire RAM nécessaire à son exécution semble être difficile en dessous de 3 [Mo]. Certes, il existe des minis distributions Linux tournant sur des machines équipées de 2 [Mo] de RAM. Ces systèmes exotiques ne possèdent souvent pas toutes les fonctionnalités d'un Linux standard. Certaines distributions ont subit également les affres d'habiles bidouilleurs qui souvent oublient certaines conventions et règles propres à un système Unix.

La réduction du système passe par l'optimisation des différents blocs qui constituent le système d'exploitation Linux. Il s'agit d'opérer, une transformation du processus de démarrage, la réduction du nombre de fichiers de configuration, des commandes et des librairies C. L'utilité de la partition de *swap* peut judicieusement être remise en question. Et enfin, le noyau peut subir une optimisation en vue d'être débarrassé des pilotes de périphériques inutiles.

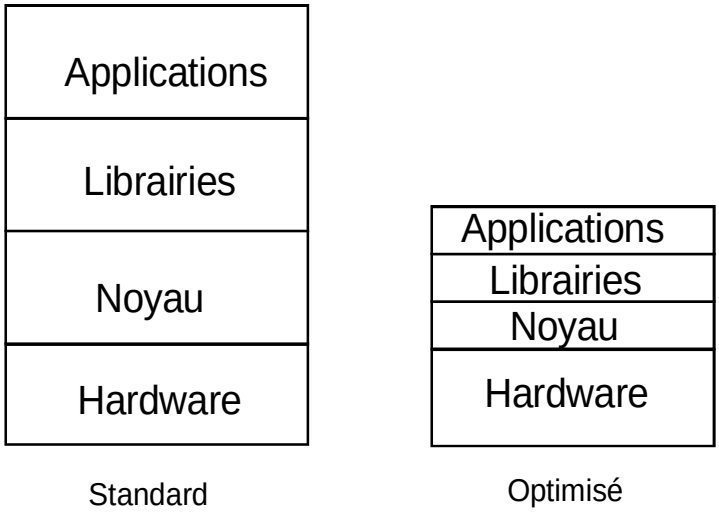


Figure 7 : Optimisation de Linux

Linux n'était pas destiné lors de sa conception aux systèmes embarqués. Immense fourtout informatique, il a été petit à petit adapté à toutes les sauces. Certaines particularités de fonctionnement se doivent d'être adaptées au milieu des machines embarquées. C'est le cas du concept de système de fichiers qui est traité dans les lignes qui suivent. Enfin, cette partie théorique se terminera sur une succincte analyse des possibilités d'affichage graphique n'utilisant que très peu de ressources.

5.3 Système de fichiers

Un système de fichiers est un format de stockage des fichiers sur des supports tels que le disque dur, la disquette, le cd-rom et autres mémoires de masses. Les systèmes de fichiers les plus connus sur les architecture x86 sont FAT16, FAT32, NTFS (formats natifs des divers Windows), et *Ext2* pour Linux ou encore ISO 9660 (le format des cd-rom). Si Windows ne reconnaît que son propre système de fichier, en plus du système des cd-rom, Linux a la possibilité de lire et écrire sur un grand nombre de système de fichiers provenant d'Os multiples. Un tableau avec une liste des systèmes de fichiers compatibles avec Linux est visible en annexe *Systèmes de fichier supporté par Linux* page 123.

Lorsque l'on travaille avec un système d'exploitation de type Unix comme Linux, on ne peut pas supprimer son système de fichier. Celui-ci fait intégralement parti du processus de fonctionnement de l'ensemble de l'Os.

5.3.1 Généralités

Discuter du choix d'un système de fichier et du logiciel permettant au système d'exploitation de le gérer peut paraître inutile lors d'une configuration standard d'un Pc de bureau. La configuration par défaut installée dans les distributions du grand commerce suffit généralement. Lorsqu'on parle d'un système embarqué, il faut avoir à l'esprit, qu'il évolue dans des conditions très rudes. Ce que la plus grande partie des systèmes de fichiers supportent très mal, ce sont les coupures intempestives de l'alimentation. Si le système n'est pas éteint dans les règles de l'art, des données peuvent être perdues. Cela vient leur mode de fonctionnement. Pour résumer, lorsque des données sont inscrites, effacées ou encore modifiées, le noyau n'accèdera pas dans tous les cas à la mémoire de masse. Pour en réduire les accès, et augmenter les performances générales du système, il stockera ces

modifications en cache dans la mémoire RAM. Si la machine est stoppée d'une façon brutale, il se peut que les données n'aient pas encore effectivement été écrites en mémoire de masse. Elles seront donc perdues. Linux supportant un nombre impressionnant de systèmes de fichiers, une rapide description des plus intéressants d'entre eux justifiera le choix final.

5.3.2 Ext2

Il s'agit du système de fichier le plus utilisé actuellement sous Linux. Son utilisation sur des stations de travail ou des serveurs ne pose aucun problème. En utilisant un système de redondance des structures vitales, il permet souvent de retrouver un système cohérent après un arrêt forcé, grâce à un utilitaire semi-automatique du nom de *e2fsck*. Cette procédure de restauration nécessite l'intervention d'un administrateur ce qui n'est pas concevable dans le cas d'un système embarqué.

5.3.3 ReiserFs

Le système ReierFs est relativement récent. Il semble qu'il soit le successeur de *Ext2*. On le retrouve comme système de fichier par défaut dans un grand nombre de distribution grand publique récentes. Ce système de fichier a été créé par Monsieur Reiser Hans, qui suite à son travail de thèse sur les *filesystems*, s'aperçu que tous les systèmes de fichiers reposait sur les mêmes concepts archaïques. Sans entrer dans les détails de son implémentation il faut savoir que *ReiserFs* utilise un système de journalisation, légèrement pénalisant au point de vue des performances mais bien utile en cas de coupure de l'alimentation. Ce système de fichier malgré certains avantages ne convient pas au systèmes embarqués. Certes sa structure permet une économie de 5% d'espace disque et sa journalisation[26] élimine le problème d'arrêts intempestifs. Malheureusement, son implémentation très lourde pénalise les systèmes aux performances moindres. De plus, la taille des partitions *ReiserFs* doivent être au minimum de 32 [Mo]. Ce dernier point exclue son utilisation dans un système disposant d'une mémoire de stockage limitée.

5.3.4 Initrd

L'une des options précieuses de Linux consiste dans la possibilité de créer de petite image de boot en utilisant le système de fichiers *initrd*. Initialement destiné à simplifier la procédure d'installation, le système de fichier *initrd* permet de charger un petit système de fichiers de la même manière que le noyau se place de lui-même en mémoire vive depuis son lieu de stockage. Ce système de fichier est compressé au format *gzip*. Le noyau le décompresse en RAM Disk [27]au moment du démarrage. Dans l'installation de distributions standards, ce mécanisme permet d'éviter de devoir choisir au démarrage entre plusieurs noyau utilisant des drivers de disques durs différents. Avant la création du système *initrd* il était impossible de démarrer Linux d'un support quelconque sans que le driver pour ce support n'ait été préalablement compilé avec le noyau. En utilisant un contrôleur IDE, ce problème ne se pose pas. Les contrôleurs SCSI employant chacun des drivers différents, l'utilisateur devait pouvoir choisir dans une liste d'une douzaine de noyaux différents

[26] Tant que le fichier n'est pas totalement écrit sur le support de stockage le journal en garde une trace.
[27] Le RAM Disk permet d'obtenir une partition similaire à celle rencontrée sur une mémoire de masse en mémoire vive.

avant même de pouvoir démarrer le système. Avec *initrd* un petit système de fichiers contenant les drivers essentiels et les scripts de démarrage est chargé avec le noyau. Ce dernier décompresse le système de fichiers et exécute un fichier nommé */linuxrc*. Ce script met en place les drivers manquants pour compléter la procédure de démarrage. Lorsque le script se termine le système de fichiers *initrd* est démonté, la mémoire occupée est libérée et la procédure d'initialisation du système se poursuit d'une façon normale.

Il est alors concevable de détourner le système *initrd* afin qu'ils deviennent le système de fichier d'une machine embarquée. Plusieurs avantages se dégagent d'une telle configuration :

- Comme le noyau, le système de fichier *initrd* est compressé et donc ne prend pas beaucoup de place en stockage.
- Le système de fichier et le noyau peuvent être fusionnés pour ne former plus qu'un seul gros fichier.
- La mémoire de masse, par exemple une mémoire Flash, peut être réduite à son minimum.

Malheureusement, il n'y a pas que des avantages :

- Le système de fichier étant stocké en RAM, lors de chaque coupure d'alimentation toutes les modifications, apportées aux fichiers de configurations par exemple, sont perdues. Il faut modifier le système de fichiers, créer l'*initrd* puis l'injecter dans la Flash
- L'exécution d'une application charge celle-ci en mémoire RAM hors comme le système de fichiers se trouve en mémoire RAM, L'application sera présente en double. Il faut encore ajouter le système de fichiers complet avec les fichiers de configuration. Ceci veut dire qu'il faut beaucoup plus de mémoire RAM.
- Le gain obtenu en pouvant diminuer la mémoire de stockage est à nuancer. En effet, il faut augmenter la mémoire RAM qui doit contenir le système de fichier.

Ce système de fichier est utilisé dans certain systèmes miniaturisés comme les lecteurs de mp3. Le système de fichier est charger lors de l'initialisation de la mémoire Flash dans une mémoire RAM qui est alimentée par batterie pour éviter la perte des données.

Remarque : Pour utiliser *initrd,* le noyau doit-être compilé avec les options pour le RAM disque.

5.3.5 JFFS

Le *Journaling Flash File System* fut développé à l'origine pour les mémoires Flash qui, comme décrit auparavant, imposent quelques contraintes de gestion. Pour s'accommoder au mieux de la limite du nombre de modifications d'une zone de données, ce système de fichiers ne réécrit jamais les données au même emplacement, et garde toujours les versions antérieures cohérentes des données en cours de modification. Et, cerise sur le gâteau, ce système de fichier fut développé pour des systèmes embarqués aux ressources limitées. Il est inclus dans le module MTD cité auparavant lui-même intégré au noyau 2.4.x. La version *JFFS2* est en cours de développement. Cette évolution apportera encore plus d'avantages car il possèdera un système de compression du système de fichiers.

Remarque : Le programme chargé de lancer le noyau (cas des x86) doit pouvoir supporter ce type de système de fichiers.

5.3.6 Conclusion

Il est évident que le système de fichiers *JFFS* présenté apporte toutes les satisfactions dans le cadre d'une utilisation pour un système embarqué. Le système *Initrd* n'est pas à écarter. Il peut convenir dans certaines applications. Il est à noter que l'évolution de JFFS, JFFS2 est en court de finition, au moment de la rédaction de ce rapport.

5.4 Suppression du swapping

5.4.1 Généralités

On alloue sous Linux un espace mémoire nommé *zone de swap*. Cet espace est réservé en mémoire de masse, le disque dur dans le cas d'un système de bureau.

Le swapping s'acquitte de deux fonctions majeures :

- Augmenter l'adressage réellement utilisable par un processus ;
- Augmenter la quantité de mémoire vive dynamique (ce qui reste de mémoire vive une fois que le code du noyau et les structures de données statiques ont été initialisées) pour charger les processus

On s'en doute, l'espace de swap ne saurait pallier un manque réel de mémoire vive car les temps d'accès des mémoires de masse sont en générale beaucoup plus élevé. Le swapping est surtout utile à l'exécution simultanée de multiples programmes, là où la mémoire physique ne suffirait plus. Si aucune zone de swap n'est active, le système pourrait refuser des requêtes pour lancer une nouvelle application. En revanche, une zone de swap permet au noyau de la lancer, puisque la mémoire peut être libérée au détriment de quelques processus déjà existant, sans les tuer.

Le swap est utile lorsque le noyau manque dangereusement de mémoire. Le noyau garde toujours une petite réserve de pages mémoire libres qui ne peuvent être utilisées que par les fonctions les plus critiques. Ceci s'avère essentiel pour éviter les pannes système qui pourrait se produire lorsqu'une routine du noyau appelée en vue de libérer des ressources n'est pas en mesure d'obtenir la zone de mémoire dont elle a besoin pour réaliser sa tâche.

Linux reconnaît deux formes de zones de swap : une partition dédiée dans le cas d'un disque dur, ou bien un quelconque fichier. Pour assurer un fonctionnement optimal de Linux, il est conseillé d'opter pour une zone de swap d'une taille double par rapport à la quantité de mémoire vive disponible. Un système Linux opérationnel s'exécutant avec un minimum de mémoire vive de quatre mégas octets, la taille du swap est alors portée à huit mégas octets. La gestion détaillée de la zone de swap ne sera pas abordée ici. En effet, décrire avec précision le mécanisme de swapping nécessiterait de nombreuses pages.

5.4.2 Suppression du swapping

A la lecture du chapitre précédant, parler de suppression de la zone de swap pourrait être, de prime abord, une totale hérésie. Cependant, il ne faut pas oublier que les systèmes abordés dans ce rapport sont des systèmes embarqués disposant de ressources limitées. Sur un ordinateur de bureau jouir de quantités gigantesques tant au niveau de la mémoire de masse que de la mémoire vive est actuellement monnaie courante. Les systèmes embarqués sont quant à eux plus exigeants. L'optimisation de la taille du système d'exploitation soulève la question de l'utilité de la zone de swap. La suppression de celle-ci entraînerait un gain de place appréciable dans le cadre d'un projet embarqué. L'utilisation de la mémoire vive sans swapping augmente également les performances du système en terme de rapidité. Enfin, certains périphériques de mémoire de masse sont bien plus gourment en consommation d'énergie, les disques durs par exemple. Et enfin, si le designer à recours à une mémoire de stockage de type Flash, il doit être conscient que ce type de composant ne supporte pas un nombre d'écriture illimitée. Le swapping n'est donc pas du tout adapté à ce type de configuration et il voudra mieux s'en passer. La suppression de la zone de *swap* ne doit évidemment pas aboutir à un manque de stabilité (blocage du système par manque de mémoire).

Lors du design d'un système embarqué, les fonctionnalités exactes sont définies dans la première phase d'un projet. Connaissant avec précision les applications qui s'exécutent sur le système, le concepteur peut prévoir une quantité de mémoire suffisante sans devoir recourir aux services du swapping.

5.4.3 Comment supprimer le swapping ?

La zone de swap est activée au moment de l'initialisation du système d'exploitation au démarrage. L'initialisation de Linux est réalisée à l'aide de scripts de démarrage. La commande *swapon –a* active la partition de swap. La suppression de cette commande dans les fichiers script de démarrage interdira au noyau les fonctionnalités du swapping. Dans les distributions Red Hat, il s'agit du fichier */etc/rc.d/rc.sysinit* pour la distribution Suse il s'agit de */sbin/init.d/boot*. De plus, Linux utilise le fichier de configuration */etc/fstab* pour décrire le partitionement du disque. L'entrée propre à la zone de swap doit être effacée. Certaine distribution opère également un contrôle de la validité du système de fichier à l'aide du script *checkfs*. Lors de ce contrôle la partition de swap est aussi vérifiée. La modification de ce script de démarrage est donc aussi nécessaire en supprimant la commande *swapon –a*.

Remarques : Les fichiers de démarrage sont détaillés dans le chapitre *Processus de création d'un système Linux*.

5.4.4 Conclusion

Définir une taille de mémoire optimale en vue de supprimer le swapping est très difficile. En effet, chaque système embarqué aura ses propres fonctionnalités et donc utilisera des applications spécifiques. Le type de mémoire de masse utilisé, le prix de la combinaison mémoire vive – mémoire de masse, les performances escomptées, et les fonctionnalités exigées sont les points qui motiveront le concepteur à utiliser ou non la zone de swap.

5.5 Optimisation de la procédure de démarrage

5.5.1 Généralités

Dans le cas d'une carte personnalisée, tout est possible, il « suffit » après avoir initialisé le hardware de passer la main au noyau contenu sur le support de stockage. Il est donc inutile de s'étendre plus à fond sur ce sujet qui dépend de la carte utilisée.

Le processus de démarrage sert à initialiser le hardware puis à passer la main au système d'exploitation. Lorsqu'il s'agit d'un x86 la procédure peut-être résumée[28] par :

- Le chargeur est démarré par le BIOS
- Le chargeur charge le noyau Linux avec l'aide du BIOS, transmet les paramètres nécessaires et démarre le noyau.
- Le noyau initialise tous les pilotes qu'il contient directement.
- Le noyau monte le système de fichiers racine (root) et y lance le processus *init*.
- Lecture du fichier */etc/inittab* par le processus *init* qui contient en général le nom du script de démarrage
- Le script en question poursuit l'initialisation du système

En utilisant un disque dur classique ou de la mémoire les émulant, le problème du démarrage est le même que celui des Pc de bureau. En revanche, le problème devient plus délicat si l'unité de stockage est équipée de mémoire Flash. Il faut dès lors posséder une « extension BIOS » qui sera reconnue par le BIOS du système x86. Cette extension BIOS doit être présente lors du « boot » du système soit en ROM[29] ou dans le composant lui-même. C'est le cas des *DiskOnChip* qui possèdent leur propre BIOS. Il est tout à fait possible d'écrire soi-même cette extension. Mais il faudra trouver ensuite un emplacement pour la stocker.

5.5.2 Chargeurs Linux

Un système x86 impose l'utilisation d'un chargeur. Celui-ci peut-être contenu sur le support de stockage ou dans la mémoire non-volatile (ROM) qui contient également le BIOS. Le projet *LinuxBIOS* propose de remplacer le BIOS des cartes mères actuelles par un programme se chargeant d'initialiser le matériel et de lancer le système d'exploitation Linux. L'installation d'un chargeur logicielle étant la procédure la plus simple, elle sera traitée dans les lignes qui suivent.

Syslinux

Syskinux est un chargeur très facile à utiliser puisqu'il s'agit d'une application Ms-Dos. Il prend en argument un fichier de configuration et une image du noyau Linux, stocké sous forme d'une partition *Dos*. Si le démarrage fait appel à une extension du BIOS ne possédant qu'une comptabilité avec *Dos*, à l'exclusion de tout autre système de fichiers, ce chargeur est particulièrement recommandé. Il faut dès lors partitionner une partie de la mémoire Flash en une petite partition *Dos* et le reste comme partition Linux.

[28] Le détail de procédure de
[29] Read Only Memory

Lilo

Lilo est le chargeur par défaut de Linux. Une partie de son code est stocké sur le *Master Boot Record*[30]. Lors de la phase de démarrage d'un système x86, ce bout de code est le premier à être exécuté. Son chargement en mémoire vive est effectué par le BIOS. Puis, Lilo continue la phase de démarrage en lançant le noyau. Il reconnaît un grand nombre de système de fichiers dont *Ext2* et *JFFS* pour les *DiskOnChip*. Pour le système de fichiers *JFFS*, il convient d'utiliser le patch *mtd-lilo*.

Les autres

Il existe de nombreux chargeur pour les systèmes x86 comme *Grub, LoadLin*. Il n'est pas utile de les décrire tous ici, leur tâche principale étant similaire.

5.5.3 Démarrage du noyau

Lors du démarrage de Linux, le noyau affiche une quantité impressionnante de messages statuant sur l'initialisation de différents périphériques. Ces messages de boot peuvent être, dans le cas d'une utilisation avec console de visualisation, trop verbeux. Il est possible de les supprimer en ajoutant simplement quelques lignes de code dans le fichier */kernel/printk.c*. Le détail de ce code est visible dans le chapitre *Processus de création d'un système Linux* sous chapitre *Modification du code source du noyau* page 71. De plus, le noyau initialise un grand nombre de périphériques. Il tente également d'initialiser les périphériques installer dans le noyau mais qui ne sont pas présents. La suppression de ces drivers, non utilisés par le noyau, accélère le processus de démarrage. Ce dernier point est détaillé à la page 43.

5.5.4 Fichiers d'initialisation

Lorsque le noyau a fini sa phase d'initialisation, il lance sa première tâche utilisateur En générale il s'agit d'un programme nommé *init* placé dans le répertoire */sbin* .Il est possible de passer en paramètre au noyau un autre emplacement de stockage.

Init « System V »

Les distributions standards possèdent généralement un système d'initialisation hérité des versions Unix System V, basé sur le lancement de service en fonction du niveau d'exécution : mono-utilisateur, multi-utilisateur, multi-utilisateur avec réseau, multi-utilisateur avec réseau et interface graphique, reboot, arrêt. A chaque niveau est associé un répertoire */etc/rc.d/rc.x* [31]qui contient des scripts en *Shell* organisé par priorité. Pour chaque niveau, les scripts sont chargés de lancer ou d'arrêter les services assurant le fonctionnement de la machine pour ce niveau d'exécution. Ce système a le gros avantage de définir proprement la localisation des différents scripts de démarrage dans des sous-répertoires correspondants aux niveaux d'exécution. La gestion de ce système est cependant complexe pour un système embarqué comprenant en règle générale très peu de service à démarrer. De plus, la présence d'un interpréteur de commande est presque obligatoire.

30 Le MBR se situe dans les 512 premiers octets d'un disque
31 Où x est le niveau en question

Init personnalisé

Il est tout à fait possible de lancer un programme personnalisé afin de réaliser l'initialisation du système. Ce programme aura pour charge de lancer toutes les tâches du système. Ce programme peut-être généré d'une façon statique et dans ce cas le programme et indépendant. Si par contre, il utilise des bibliothèques dynamiques il faudra veiller à la présence de celle-ci.

Il est possible d'être encore plus radical en intégrant la procédure d'initialisation au noyau.. En effet, le code du fichier source *linux/init/main.c* s'occupe de lancer le programme *init*. L'ajout de quelques lignes de codes à la fin de ce fichier permet de se passer du programme *init*.

Ces solutions sont acceptables dans la mesure ou le système embarqué a une utilité spécifique. Il manquera cependant de beaucoup de souplesse. En cas de changement de configuration il faut recompiler le programme d'initialisation voire même le noyau !!

Init avec configuration

La solution utilisant un script de configuration sans pour autant reprendre toute la procédure de démarrage du type Système V paraît être la meilleure. En effet, bien qu'un peu plus complexe que la solution précédente, elle est tout de même intéressante dans un système embarqué, car elle ne nécessite que très peu de ressources supplémentaires. Sur les systèmes Unix le script */etc/inittab* définit les différentes tâches à lancer comme les différents terminaux virtuels, le serveur graphique X[32] … De plus, il est beaucoup plus facile d'en modifier la configuration, un simple éditeur de texte suffisant.

5.5.5 Conclusion

Dans le cas d'une architecture x86, le démarrage de Linux nécessite l'utilisation d'un chargeur. Lilo semble, de part son utilisation répandue et sa taille respectable en mémoire, être un choix judicieux. La configuration avec un fichier *inittab* est également conseillée car l'administration du système embarqué s'en trouvera amplement facilitée. Et enfin, l'optimisation du noyau peu également alléger la procédure de démarrage. Ce dernier point fait l'objet d'un chapitre complet dans les pages qui suivent.

5.6 Optimisation des librairies C

5.6.1 Généralités

Les librairies aident le noyau à faire fonctionner les programmes. Le noyau Linux utilise en générale un ensemble de bibliothèques « linkées » qui permettent son accès. La plus grande de ces bibliothèques est la *Libc* 6 qui contient la majorité des fonctions POSIX. Par exemple, elles fournissent le code pour des fonctions comme : *open(), close(), read(), printf()* …Sur les distributions standards, ces bibliothèques prennent un espace considérable en mémoire de masse : quelques quatre méga bytes pour la *Libc6* standard. C'est l'optimisation des librairies C qui permet un gain de place énorme pour un Linux embarqué!!

32 X est le nom donné à l'application de la fondation Xfree servant à l'affichage de graphiques évolués

5.6.2 Statique vs Dynamique

Les bibliothèques peuvent être « linkées » à un programme de deux manières différentes : statiquement ou dynamiquement. Dans le cas d'une liaison statique, chaque programme possède dans son code la copie du code contenu dans la bibliothèque utilisée. Dans le cas d'un lien dynamique, le programme utilise le code de la bibliothèque nécessaire, qu'au moment ou il en a besoin. Ainsi, plusieurs programmes peuvent se partager le même morceau de code. Sur un système exécutant de multiples applications, cette solution est la plus judicieuse. En effet, elle permet de réduire la taille des programmes exécutable, leur temps en chargement, et de réduire leur place en mémoire en partageant ce qui peut l'être entre les différents programmes.

Par contre, sur un système embarqué il peut être utile de savoir si l'utilisation de programmes « linkés » d'une manière statique est plus opportun. En effet, sur un système embarqué, la plupart du temps, seuls quelques programmes s'exécutent simultanément. Dans ce cas, l'utilisation de liens statiques économise de l'espace en mémoire. Cependant si le système ce doit d'être évolutif et comporter plusieurs programmes, l'emploi de lien dynamiques est la meilleure solution.

5.6.3 Eliminer les librairies inutiles

Toutes les bibliothèques installées ne sont pas utile et peuvent dès lors être éliminée. En effet, elles sont composées de modules distincts. Sous Linux, la commande *ldd*, avec comme paramètre le nom d'une application, indique les dépendances de ce programme vis à vis des diverses librairies. Pour simplifier le recensement des bibliothèques nécessaires, le script *mklibs.sh* utilisé par la distribution *Debian* est fort intéressant. Le principe est de créer un graphique des dépendances d'une arborescence d'exécutables et de bibliothèque. Il crée ensuite une version des bibliothèques débarrassée des modules inutiles. Cette approche est cependant relativement spécifique à un système donné. L'ajout d'autres applications entraînera la modification des librairies. Il est donc fortement conseiller d'appliquer cette technique en phase finale, lorsque le produit est totalement stable. Le script *mklibs.sh* est visible en annexe *Réduction des librairies C* page 211. Il est également disponible sur le Cd-Rom.

Actuellement la librairie standard se trouve être la *Libc 6* ou *Glibc*. La librairie précédente la *Libc 5* semble être moins volumineuse et peut dans la plus grande partie des cas remplacer la *Libc 6*.

5.6.4 Librairies alternatives

La boulimie de la *Libc 6* a poussé un certain nombre de développeurs à construire de petites librairies, dont le but est de fournir une partie des fonctionnalités de la bibliothèque originale avec une taille réduite. La *ucLibc* est la plus connue. Elle fut implémentée en premier lieu dans le cadre du projet *ucLinux*. Rapidement cette bibliothèque fut adaptée pour des noyaux classiques pour architecture x86. De plus, un grand nombre de programmes ont vu leur fichier de compilation adapté pour l'utilisation de la librairie *ucLibc*. Il s'agit entre autre des programmes comme *BusyBox* et *TinyLogin* qui sont détaillé dans le chapitre suivant. D'autres libraires réduites sont aussi disponibles. Il s'agit de *NewLibc* et *diet-Libc*. Ces librairies sont, en générale, limitées et ne conviennent pas pour toutes les applications mais en contre partie elles offrent un gain de place énorme.

Librairies	Taille en kilo bytes
Libc 6	1500 [ko]
ucLibc	380 [ko]
NewLibc	269[ko]
Diet-Libc	800[ko]

Tableau 1: Taille des librairies

Linkage statique à l'aide de la librairie ucLibc

Comme décrit précédemment, il est possible de linké un programme d'une manière statique. Il est intéressant de comparer la différence de taille entre un programme linké de manière statique avec la librairie standard *Libc6* et la *ucLibc*. Le programme, choisi pour ce test, est le suivant :

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

int main(void)
{
  printf("hello world\n");
  exit(42);
}
```

La compilation est réalisée avec l'outil GNU *gcc*. Les options de compilations utilisées sont similaires seules les libraires diffèrent. Le résultat obtenu après compilation est le suivant :

Compilation avec la *Libc 6* ou *Glibc*

```
gcc -Wall -Os -fomit-frame-pointer -c hello.c -o hello_glibc.o
gcc -s --static  --static hello_glibc.o -o hello_glibc
strip --remove-section=.note --remove-section=.comment hello_glibc
ldd hello_glibc
  statically linked (ELF)
ls –sh hello_glibc
192k hello_glibc
./hello
hello world
```

Compilation avec la *ucLibc*

```
gcc -Wall -Os -fomit-frame-pointer -fno-builtin -nostdinc -
I../include -I/usr/include/linux -c hello.c -o hello.o
gcc -nostdlib -s -gc-sections hello.o -o hello ../libc.a
strip --remove-section=.note --remove-section=.comment hello
ldd hello
  statically linked (ELF)
ls –sh hello
4.0k hello
./hello
hello world
```

La commande *ls –sh hello* affiche la taille du fichier. A la vue des chiffres, il n'a aucun commentaire à faire.

5.6.5 Conclusion

Les librairies occupent une grande place en mémoire de masse. Leur optimisation est absolument nécessaire pour une utilisation dans un système embarqué. L'emploi de librairies réduites permet de diminuer au maximum l'espace de stockage. En contre partie, tous les programmes ne pourront pas être portés sans modifications. Pour des systèmes évolutifs, il est plus judicieux d'opter pour une librairie standard telle que la *Libc6* débarrassée des modules inutiles. Si le système fini, exécute un nombre restreint d'application il peut être judicieux de comparer la taille des applications linkées dynamiquement aux applications linkées statiquement.

5.7 Shell et commandes minimales

5.7.1 Généralités

Linux comme tous les systèmes Unix utilise un interpréteur de commande pour dialoguer avec le système. L'opérateur y écrit des commandes qui y sont traduites. Ces commandes ainsi que l'interpréteur offre un certain de nombre de fonctionnalités intéressantes pour gérer, administrer et configurer le système. Ces programmes prennent malheureusement de la place sur le support de stockage. C'est pourquoi leur optimisation doit faire l'objet d'une analyse complète.

5.7.2 Le Shell

Le Shell est l'interpréteur, en mode console, qui permet de communiquer avec le système en lui passant des commandes. Sous Linux, comme sur l'ensemble des Os du type Unix, il existe deux groupes de *Shell*. Le premier groupe descend du *Bourne Shell* et le second du C *Shell*. La différence entre ces deux types d'interpréteur de commande se situe au niveau de la syntaxe. Le monde Linux utilise en générale le *bash* [33]. Il est conforme POSIX, compatible avec le *Bourne Shell*, créé et distribué par le projet GNU. Cependant, pour qui le *bash* ne convient pas, il en existe bien d'autres: *csh, ksh, sh, tcsh, zsh*,… Chaque interpréteur offre des avantages et des désavantages.

Contrairement aux ordinateurs de bureau, la présence d'un interpréteur de commandes Shell dans un système embarqué n'est pas une nécessité. En effet, ces systèmes ne sont souvent pas destinés à interagir avec un utilisateur tapant des commandes au clavier qui bien souvent n'est pas présent. Cependant, la présence du *Shell* peut s'avérer importante. En effet, les scripts de démarrages et un grand nombre d'exécutables utilise des scripts *Shell*. Ceux-ci sont souvent plus petits qu'un programme en C. De plus, leur édition se fait en utilisant un simple éditeur de texte. Il n'est donc pas nécessaire de compiler le fichier à chaque fois. L'interpréteur de commande *bash* cité plus haut peut être employé dans un système embarqué, sa taille étant de 80 [ko]. Pour réduire encore la taille de l'interpréteur de commande il est possible d'utiliser une version plus légère dénommée *ash*. Cet interpréteur ne pèse que 12 [ko] et profite d'une totale compatibilité avec son grand frère le *bash*. [34]

5.7.3 Optimisation des commandes standards

Les distributions Linux du marché installent toute une pléiade de commandes qui sont simplement de petits programmes. Leur utilité sur un poste de bureau ou sur un serveur peut se justifier. Sur un système embarqué, il n'est pas indispensable par exemple de pouvoir formater un disque. Il est opportun d'effacer les commandes inutiles. Le choix des commandes dépend directement de l'application installée sur le système. Elles sont regroupées sous quatre répertoires : */sbin* , */bin*, /*usr/sbin* et */usr/bin.*

Lors du choix des commandes à conserver, il faut avoir à l'esprit que les scripts de configuration des applications et les scripts d'initialisation du système les utilisent également. De plus, la nécessité d'une connectivité distante ou locale obligera de garder un certain nombres de commandes supplémentaires telles que *agetty, passwd, login*.

5.7.4 Alternatives aux commandes standards

L'emploi des commandes standards, tout comme les librairies habituelles, est relativement lourd en espace mémoire pour un système embarqué. De plus, certaines options de ces commandes ne sont jamais utilisées. Dans le cadre de projet comme *Lineo* ou *LEM*, il a été implémenté des programmes dont le but est de réduire de façon drastique la place en mémoire et de fournir les fonctionnalités des commandes standards. Le projet le plus aboutit est le *BusyBox* complété par *TinyLogin*. C'est deux programmes sont sous licence *GPL*, ce qui constitue un avantage non négligeable dans le cas ou le système complet serait commercialisé.

---

[33] Bourne Again Shell
[34] Avantage non négligeable dans le cas de portages de scripts existants

BusyBox

BusyBox combine des versions "allégées" de nombreux utilitaires Unix en un exécutable unique et de petite taille. En effet, chaque fichier exécutable individuel contient un en-tête de chargement, dont la taille est loin d'être négligeable par rapport aux instructions du programme proprement dit. 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 ou systèmes embarqués. *BusyBox* a été écrit avec pour objectif une optimisation de taille et une utilisation de ressources limitées.

Commandes standards	**BusyBox**
1500[ko] Commandes utiles :	190[ko] busybox*
1500[ko] total	190[ko] total

Tableau 2: Taille de BusyBox

Malheureusement, les commandes *Shell* incorporée ne possède pas toutes les options des commandes externes équivalentes. Cependant, elles s'avèrent plus que suffisantes. L'outil est aussi extrêmement 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 est d'approximativement 200 [ko]. Les commandes standards prennent plus de 3,5 [Mo]. Les détails d'installation sont visibles dans le chapitre *Processus de création d'un système Linux* page 78. En annexe *Description des commandes Busybox* page 231 se trouve la description de toutes les commandes

TinyLogin

TinyLogin fonctionne selon le même principe que *BusyBox*. Il complète généralement ce dernier, en fournissant des commandes *Shell* complémentaires pour la gestion des connexions utilisateurs. Il s'agit des commandes *addgroup, adduser, getty, login, passwd, su, sulogin*.

Commandes standards	**TinyLogin**
0 /usr/sbin/addgroup	40[ko] ./tinylogin*
24[ko] /usr/sbin/adduser	
16[ko] /sbin/getty	
36[ko] /bin/login	
28[ko] /usr/bin/passwd	
24[ko] /bin/su	
12[ko] /sbin/sulogin	
140[ko] total	40[ko] total

Tableau 3: Taille de TinyLogin

La configuration de *TinyLogin* est faite dans le chapitre *Processus de création d'un système Linux.* De plus, l'ensemble des commandes est décrit en annexe *Description des commandes TinyLogin* page 270.

5.7.5 Conclusion

L'élimination d'une partie des commandes standards jugées inutiles, ne permet pas un gain de place optimal. Il est préférable d'avoir recourt à un programme tel que *BusyBox*. La nécessité d'une connectivité distante obligera l'emploi de *TinyLogi*n. Ces deux programmes jouissent d'une fiabilité suffisante et sont utilisés dans de nombreuses distributions de petite taille.

5.8 Optimisation du noyau

5.8.1 Généralités

Le noyau Linux constitue le cœur du système. Il faut donc apporter une attention toute particulière à sa configuration. Le noyau Unix joue le rôle d'intermédiaire entre les programmes et le matériel. En premier lieu, il gère la mémoire pour tous les programmes en cours d'exécution, et s'assure qu'ils occupent tous une part équitable du temps processeur. De plus, il fournit une interface simple à utiliser aux programmes pour communiquer avec le matériel. On parle d'appels système. Bien sûr, c'est un petit peu plus compliqué que ça, mais ces fonctions de bases sont les plus importantes à connaître

Son architecture *monolithique* modulaire permet de le configurer d'une manière optimale. Une simple compilation permet d'en réduire la taille. Il est également possible d'optimiser le noyau en supprimant ou en modifiant le code source.

5.8.2 Choix de la version du noyau

Le noyau Linux est en constante évolution. Il existe plusieurs sources du noyau disponibles, des premières versions 0.x.x à la dernière version 2.4.x. L'utilisation d'une version plus ancienne peut-être utile car elle est souvent bien moins exigeante en espace mémoire RAM et mémoire de stockage. Un noyau 0.x.x est environ de 200 [ko] plus petit que le noyau 2.4.x.(Lorsqu'ils sont compressés). Le chargement en mémoire du noyau 0.x.x demande également moins de place. Ainsi il peut être chargé dans les premiers 640 [ko] sur un système x86 soit à l'adresse 0x0001000. Les noyaux des générations suivantes dont la taille, lorsqu'ils sont décompressés, est plus importante, sont chargés en mémoire à l'adresse 0x00100000 soit au premier méga octet. Un noyau 2.4.4 demande environ un méga octet de mémoire dans une configuration allégée. Il peut atteindre plusieurs méga octets avec certaines options de compilation.

Le choix du noyau c'est porté pour ce projet sur le noyau 2.4.4. . Il s'agit actuellement du dernier noyau disponible. Bien que plus volumineux que ces prédécesseurs, il intègre certaines fonctionnalités utiles aux systèmes embarqués. Ce noyau permet la gestion des disques et mémoires Flash. Cette gestion était rendue possible sur les noyaux de type 2.2.x par l'ajout d'un module chargeable. De plus, comme chaque évolution, certains problèmes de stabilités sont solutionnés. Le gain de quelques kilo octets ne justifie pas de rattacher le système à un noyau obsolète tel que le 1.x.x .

5.8.3 Compilation du noyau

La compilation du noyau est une spécificité des systèmes libres, qu'il n'est possible que parce que l'on dispose des sources du noyau. Contrairement aux ordinateurs de bureau, les systèmes embarqués ont une configuration matérielle quasi-statique. Ceci permet d'éliminer, sans aucune arrière pensée, tous les drivers inutiles. Un système embarqué n'aura pas besoin de drivers SCSI, USB, de support Cd-Rom ou DVD, périphériques de pointages. Le noyau Linux intégrant un grand nombre de driver leur suppression entraîne évidemment un gain de place. En plus, le noyau Linux permet de charger certain driver par le biais de modules qui ne sont appelés que lors de leur utilisation. Il est donc envisageable d'éliminer certain driver du noyau pour les intégrer à l'aide de modules afin de limiter la taille du noyau en mémoire car les modules ne sont chargé que lorsque le noyau en à besoin.

Sous Linux, il est très courent de compiler son noyau. Au lancement de la compilation l'utilisateur choisit les drivers qu'il désire garder. On peut avec cette technique obtenir un noyau fonctionnel de 350 [ko]. Mais il faut compter environ 450 [ko] pour un noyau intégrant les fonctionnalité du réseau ethernet. Il s'agit du noyau compressé. La compilation du noyau nécessite de posséder les sources du noyau et un compilateur. Dans les distributions du commerce tel que Suse ou Red Hat, le compilateur GNU *gcc* est installé par défaut.

La configuration du noyau est effectué grâce à un menu graphique avec les commandes :

```
make menuconfig

ou

make xconfig  (sous XWindows)
```

En mode console l'accès au option de configuration du noyau ce fait avec la commande

```
make config
```

Afin d'avoir un aperçu sur la taille de certaines options du noyau, plusieurs compilations ont été effectuées avec un noyau du type 2.4.4. . Les options utiles mais discutables pour un système embarqué ont été analysées. Le noyau a été compilé avec l'option pour processeur 80386 qui permet de parvenir à un code plus petit. Les différences de taille ont été constatées sur des noyaux en fin de compilation donc des noyaux compressé au format bzImage. Il est évident que la taille des options de compilation lorsque le noyau est décompressé, est encore plus importante.

Option de compilation du noyau	Taille en kilo octet
Affichage VGA	63[ko]
Console sur port série	43[ko]
Eléments minimaux pour le réseau	102[ko]
Frame buffer pour affichage graphique	42[ko]
Bus PCI	61[ko]
Disques et mémoires Flash	4[ko]
Emulation du coprocesseur	28[ko]
Module chargeable	18[ko]
Option pour cpu 486	1[ko]
Option pour cpu 586	1[ko]

Tableau 4 : Taille des différentes options de compilation

La procédure détaillée de compilation est visible dans le chapitre *Processus de création d'un système Linux.* En outre, l'annexe *Options de compilation du noyau* page 272 décrit les principales options de compilation.

5.8.4 zImage ou bzImage

Pour lancer la compilation du noyau destiné à une machine x86, il existe deux commandes possibles :

```
make zImage

ou

make bzImage
```

Comme décrit dans le chapitre *L'OS Linux* page 11, le taux de compression n'est pas le même pour une *zImage* et une *bzImage*. La *zImage* possède un taux de compression inférieur. En outre, la taille du noyau compressé doit être inférieure à 508 [ko]. En contre partie, l'image compressée peut être chargée dans les 640 premiers [ko]. La *bzImage* jouit d'un taux de compression supérieure. Cependant, l'image compressée doit être chargée au-delà du premier méga octet. Dans les deux cas le noyau est décompressé à partir du premier méga octet soit à l'adresse 0x100000. Dans le cas de la *bzImage* il faut disposer au-delà du méga octet d'une taille de mémoire équivalent à : *noyau compressé* + *noyau décompressé*. Tandis que pour une *zImage,* il est nécessaire de disposer de la place pour l'image décompressée uniquement.

Le noyau 2.4.4 a été compressé en zImage et bzImage, en conservant des options de configuration identiques. Le résultat est visible dans le tableau ci-dessous :

Noyau	Taille
zImage	508 [ko]
bzImage	501 [ko]

Tableau 5 : Comparaison zImage bzImage

5.8.5 Modification du code source

Il est possible d'optimiser encore plus le noyau car tout le code inutile à un système embarqué ne figure pas dans le menu de configuration. A titre d'exemple, le code pour la gestion des lecteurs de disquette 360 [ko] n'est pas une option et donc intègre le noyau sans qu'il soit possible de le supprimer. De plus, Linux possède un grand nombre de constantes qui définissent la taille de certaines ressources. Ces constantes peuvent être diminuées sans conséquences néfastes pour un système embarqué. Un exemple d'une telle optimisation est réalisé par *ETLinux*. Il serait préférable d'appliquer ces modifications sous forme de patch, de manière à pouvoir les utiliser facilement en cas de changement du noyau. Ces modifications sont bénéfiques tant sur le plan de l'occupation de la mémoire de masse que sur le plan de l'utilisation de la mémoire RAM. Les différentes modifications qui peuvent être opérée sur le code source sont traitées dans le chapitre *Processus de construction d'un système Linux embarqué* page 71.

5.8.6 Taille du noyau décompressé et de ses structures de données

Le noyau Linux est présent sous une forme compressée lorsqu'il est stocké en mémoire de masse. C'est seulement au démarrage de l'Os qu'il est décompressé en mémoire RAM. La taille du noyau décompressé varie énormément, en fonction des options de compilations sélectionnées. De plus, il réserve une certaine quantité de mémoire RAM pour ses structures de données. Il est difficile de donner avec précision le taux de compression tant celui-si est dépendant du contenu du code. Cependant après plusieurs expérimentations, il s'est avéré que le taux de compression oscillait au alentour de 1,7. Un noyau de 450 [ko] compressé prendra environ 900 [ko] en mémoire RAM. A cela, il faut encore ajouter un espace alloué pour les différentes structures de données. Par structures de données il faut comprendre la mémoire réservée pour le stack TCP/IP, pour pouvoir lancer un nombre minimal de tâches, ou pour l'exploitation de différentes consoles. Cette allocation de mémoire est en partie réalisée en fonction du total de mémoire RAM disponible. Ainsi un système disposant de suffisamment de mémoire verra la taille de ses structures de données augmentées. Sans entrer dans les détails du code source et par simple observation du système, un noyau minimal de 450 [ko] sur un système minimal équipé de 3 [Mo] de mémoire RAM demande environ 200 [ko] pour les structures de données. Ce noyau est compilé avec les options pour le réseau ainsi que pour l'affichage sur la console VGA. Il est cependant très difficile, sans décortiquer le code source de Linux, de pouvoir donner une taille exacte pour ses structures. Une observation du système en place semble être la seule solution.

5.8.7 Drivers chargeables ou modules

Traditionnellement, les pilotes de périphériques intègrent le noyau, ce pour plusieurs raisons. En premier lieu, la plupart ont besoin d'interagir de façon spécifique avec le matériel, ce que seul le noyau permet. Ce système pose évidemment certains problèmes. Tout d'abord, à chaque ajout de nouveau matériel l'administrateur doit, comme décrit précédemment, reconstruire le noyau pour y inclure les nouveaux périphériques. De plus,

cette méthode n'incite pas les développeurs à programmer. Pour y remédier, Linux reconnaît les pilotes chargeables, ajoutés ou supprimés de la mémoire lors du fonctionnement du système. Ces pilotes, qui constituent toujours une partie du noyau, sont compilés séparément et valider qu'au moment de leur utilisation. L'écriture d'un module est en générale relativement propre car il doit respecter certaines règles. Pour que le noyau soit à même d'utiliser les modules, une option de compilation spécifique doit être sélectionnée. Si cela est possible, il est préférable dans le cadre d'un système embarqué de se passer des modules. Ceci n'est malheureusement pas toujours possible. En effet, tous les périphériques ne sont pas inclus dans le code du noyau et doivent donc être accédés à l'aide de pilotes de périphériques chargeables. Un module n'est en fait qu'un fichier objet contenant l'intégralité du code du pilote. Il faut savoir que certains drivers ne peuvent pas être utilisés sous forme de module. Il s'agit plus particulièrement des pilotes de périphériques utiles au démarrage du système. (Le pilote du contrôleur IDE dans le cas d'un système x86.)

5.8.8 Conclusion

La taille du noyau dépend de l'utilité du système final. C'est en compilant le noyau avec les options désirées que le volume précis peut être défini. Le noyau Linux gardera malgré toutes les optimisations possibles une taille respectable se situant au alentour des 400 [ko] pour la version 2.4.4 compressée. Des modifications plus radicales afin d'en diminuer encore plus la taille risqueraient de poser des problèmes de compatibilité. Le but étant de disposer d'un système Linux « standard » de telles modifications sont vivement déconseillées.

5.9 Simplification du nombre de fichiers

5.9.1 Généralité

Un système Linux standard possède un grand nombre de fichiers de configuration réparti dans autant de répertoire. Ces fichiers de configuration ont été mis en place dans un souci de clarté afin de facilité l'administration du système et la standardisation des configurations. Un système embarqué n'a pas les mêmes exigences en terme d'administration système qu'un serveur, par exemple.

5.9.2 Facilité de l'administration

Comme décrit auparavant, le démarrage standard de Linux est conforme à la norme UNIX System V. Cette norme initialise le système selon six niveaux d'exécution. A chaque niveau d'exécution correspond un répertoire qui contient les fichiers chargés de lancer les applications ou de les stopper. Par exemple, le niveau 6 correspond au redémarrage du système. Ainsi, les scripts contenus dans le répertoire du niveau 6 sont exécutés à l'appelle de celui-ci. Il s'agit en grande partie d'ordre d'arrêt des applications. L'utilisation de la norme System V n'est pas adapté au système embarqué. La multiplication des fichiers ne facilite pas l'administration de petit système. C'est pourquoi, il est plus commode d'utiliser une méthode intermédiaire. Celle-ci a déjà été présentée dans le chapitre *Optimisation de la procédure de démarrage* page 35. Par contre, l'administration de machines plus importantes chargées d'exécuter un grand nombre de programmes impose l'emploi d'une telle norme.

5.9.3 Economie en mémoire de masse

Un des buts de la simplification du nombre de fichier est bien entendu l'économie de place en mémoire de masse. Chaque fichier de configuration est constitué de caractères qui sont inscrits en mémoire de stockage. Comme décrit au chapitre *L'OS Linux* page 11, pour chaque fichier, il existe un *inode*. Celui-ci fait parti de la base de donnée permettant aux systèmes de fichiers de trouver un fichier sur le support de stockage. Cette *inode* occupe de la place en mémoire. Pour être précis, une *inode* occupe 128 octets dans le cas d'un système de fichiers *Ext2*. De plus, les *inodes* contenu dans cette base de donnée sont regroupée en blocs contenant de 8 à 32 *inodes.* Les données sont stockées à la suite de cette base de donnée par bloc de un à quatre kilos octet. L'opération est répétée pour chaque groupe bloc-données. Si un bloc ne contient qu'une seule *inodes*, il occupera une place en mémoire plus important que la taille de l'inode. En effet, il faut rajouter diverses balises pour chaque bloc.

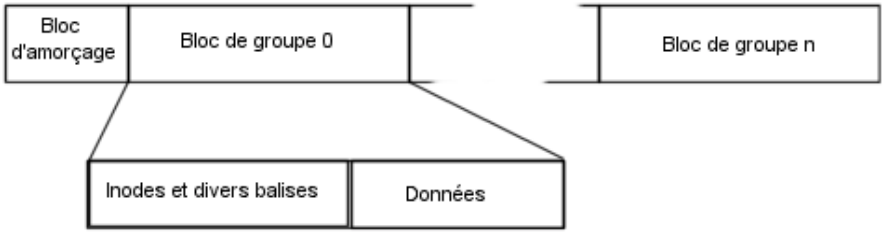


Figure 8 : Partition Ext2 sur disque dur

Les données sont écrites par morceau d'un à quatre kilos octets[35]. Un fichier de configuration n'occupe souvent que quelques octets. Cependant physiquement c'est au minimum 1024 octets qui lui seront alloués. C'est pourquoi il est conseillé de regrouper les fichiers de configurations.

Ces fichiers de configuration contiennent la plupart de temps une quantité non négligeable de commentaire décrivant l'utilité du script. Eliminer les commentaires des fichiers de configuration permet de gagner quelques kilos octets. On veillera cependant à fournir une documentation externe en contre partie des commentaires éliminés.

La minimisation des répertoires est aussi une solution d'optimisation de l'occupation du support de stockage. Un répertoire est un fichier possédant une *inode* et une suite de caractères. Un répertoire vide occupe au minimum 1024 octets plus la taille de l'*inode* soit 128 octets. Plus le nombre de fichier contenu dans un répertoire est important plus son inscription en mémoire de masse est importante. Pour chaque répertoire éliminé, il résulte un gain de place d'un minimum de 1024 [ko]. Ce conseil d'optimisation est à nuancer. En effet, Linux impose la présence de certains répertoires. Les éliminer ferait s'écarter Linux du format standard. (Le but est de produire un système Linux conforme.)

Remarque : D'un point de vue technique, il est tout à fait possible de se passer des fichiers de configuration, des répertoires et autres fioritures propre à Linux en modifiant le noyau en y incluant le code d'initialisation et même celui des applications. Mais dans ce cas, Linux n'est plus Linux.

35 Cette option est paramétrable avec Ext2

5.9.4 Conclusion

L'optimisation du nombre de fichier peut dans le cas d'un système minimal faciliter l'administration. Elle aide également à gagner de la mémoire de masse. Cependant, cette simplification ne doit pas devenir trop radicale. En effet, le gain de quelques kilos octets de mémoire peut rendre le système beaucoup plus compliqué. Il faut avoir à l'esprit que l'utilisateur et le développeur du système sont dans la plupart du temps des personnes totalement différentes.

5.10 Compilation

5.10.1 Généralité

La compilation d'un programme est un ensemble de traitements successifs sur un fichier source dont l'étape ultime est de générer un fichier exécutable par la machine. Elle se divise en quatre étapes :

- *Le pré-processing* consiste à substituer toutes les macros présentes dans le code par leur valeur: #ifdef, #define, #include, etc.

- *La compilation* : Cette étape travaille sur le fichier résultat du pré-processing, et produit un fichier texte contenant du code en langage d'assemblage spécifique à la machine sur laquelle à lieu la compilation

- L'étape d'*assemblage* prend le fichier précédent, et génère du code machine. Le fichier produit est appelé *fichier objet*, et se reconnaît en général par son extension.

- L'*édition de liens* prend un ensemble de fichiers objets pour produire un programme exécutable.

Sous Linux le langage le plus usité reste encore le langage C (sur l'ensemble des systèmes UNIX également). Ceci peut-être parce que le système lui-même est écrit en C. La compilation d'un programme entraîne évidemment l'emploi d'un compilateur. Le compilateur C de GNU, *gcc*, est l'un des plus performants et des plus polyvalents qui soient. Outil GNU gratuit, *gcc* est livré avec toutes les distributions du commerce telles que Suse, Red Hat, Mandrake… Il est également possible de télécharger les sources de *gcc* livré avec un compilateur minimal chargé de lancé la procédure de compilation. Compiler *gcc* est une procédure très longue. Pour de plus ample information il convient de se référer au site Internet *http://www.gnu.org/software/gcc/gcc.html*. *Gcc* étant le compilateur le plus utilisé dans le monde Linux, les autres applications du même genre ne seront pas examinées.

5.10.2 Options de compilations

Le compilateur GNU *gcc* fournit plusieurs « options d'optimisation ». Optimisation fait référence soit à la rapidité d'exécution soit à la réduction du code afin de prendre moins de place sur le support de stockage.

Sur les Pc de bureau actuel possédant une quantité indécente d'espace mémoire sur le support de stockage, l'optimisation ira bien entendu dans le sens de la vitesse d'exécution. Sur des systèmes embarqués où la taille de la mémoire de masse se compte en kilo octets, minimiser la taille des applications est souvent prioritaire face à la vitesse d'exécution.

Pour compiler un programme C contenu par exemple dans le fichier *hello.c* à l'aide de *gcc* il faut exécuter la ligne de commande suivante :

```
gcc –O hello hello.ch
```

Le paramètre *–O* est une option de configuration qui en l'occurrence spécifie de n'utiliser aucune optimisation. Pour compiler un programme, le développeur écrit généralement un script de configuration appelé *Makefile*. Ce ficher de compilation contient l'ensemble des options de compilation. Si une application nécessite la compilation de plusieurs fichiers, le *Makefile* en a connaissance et peut automatiser la compilation. De plus, lors de compilations successives le *Makefile* sait quel fichier a été modifié. Lorsqu'un seul des fichiers sources est modifié, il n'est pas forcément utile de recompiler *tous* les fichiers sources pour régénérer l'application. On gagnera du temps à ne recompiler que les fichiers sources sur lesquels la modification du code effectuée a un réel impact.

Options –Ox

Il s'agit des options d'optimisation les plus simples à mettre en œuvre. Pour être plus précis, *-Ox* regroupe plusieurs options de compilations de *gcc* afin de fournir une optimisation d'un certain type.

- *-O0* : aucune optimisation
- *-O2* : l'optimisation du code sera faite au niveau de la rapidité d'exécution. Si le code contient des boucles, elles ne seront pas optimisées.
- *-O3* : l'optimisation du code sera faite au niveau de la rapidité mais d'une manière encore plus radicale que l'option précédente. (Les boucles sont optimisées) Il en résulte une augmentation de la place occupée en mémoire
- *-Os* : cette option est nécessaire pour optimiser un programme en réduisant sa taille. Bien entendu, la rapidité d'exécution du programme ne sera pas prise en compte.

Afin de se rendre compte de l'efficacité de ces options dans le cas d'une optimisation de taille plusieurs compilations successives sur un même code en C ont été effectuées. Le seul paramètre changeant d'une compilation à une autre se situe au niveau de l'option de compilation *–Ox*. Ce code comporte une boucle pour permettre de profiter de l'optimisation de l'option *–O3*

```
int Var1
char Char1

void ProcTest(int *IntTest)
{
  int IntStop ;
  int EnumStop ;

  IntStop = *IntTest + 10 ;
  for( ;;)
  {
        if (Char1 == 'A')
        {
              --IntStop;
              *IntTest = IntStop – Var1;
              EnumStop = 1;
        {
        if (EnumStop == 1) break;
  }
}
```

Les résultats de l'optimisation son visible dans le tableau ci-dessous:

Option de compilation	Taille après compilation
-O (aucune optimisation)	80 Bytes
-O2	56 Bytes
-O3	56 Bytes
-Os	48 Bytes

Tableau 6 : Efficacité des options –Ox

Il faut remarquer que les optimisations en vitesse réduisent également la taille de l'exécutable.

-i386

Cette option de *gcc* permet de s'assurer que le programme compile peut s'exécuter sur une ancienne machine Intel 80386. En effet les processeurs des générations suivantes possèdent un jeu d'instructions plus importantes que les vieux processeurs ne connaissent pas. L'utilisation de cette option se révèle également utile afin de diminuer la taille du code. Une application compilée avec un code pour 386 a une taille inférieure que si elle

était compilée avec l'option pour le type de processeur x86 exacte. En contre partie l'application ne s'exécutera pas avec une vitesse optimale.

-g

Certain *Makefile* intègre l'option de compilation *–g* . Cette option dit au compilateur d'intégrer des symboles pour le « debuging ». Il possible dès lors de suivre le déroulement du code à l'aide d'un « debugger » GNU. Le code rajouté avec cette option est approximativement de la même taille que le code de départ. C'est pourquoi il convient de ne pas sélectionner cette option de compilation. En effet, lorsque l'application est prête pour être intégrée au système embarqué, il n'est pas utile de pouvoir la « debugger » sur celui-ci.

-static

L'option *–static* indique au compilateur que les librairies doivent être linkées d'une manière statique. Cette option grossit considérablement la taille de l'application. Le code de la librairie est en fait recopié dans l'application. Pour plus de détail sur le linkage des librairies il faut se référer au chapitre *Optimisation des librairies C* en page 37.

Remarque : Lors de la compilation du noyau, l'emploi des options de configuration semble être la seule option de compilation. En fait, une modification des paramètres de compilation de *gcc* permet un gain de quelques kilos octets sur l'image du noyau compressé (aussi lorsqu'il est décompressé). Il faut pour cela modifier le fichier *Makefile* dans le répertoire */linux* des sources du noyau. Cette optimisation n'est pas d'une grande efficacité car il ne faut pas l'oublier car le noyau après compilation est compressé. Les tests ont été effectués sur un noyau compressé avec *bzImage*. La taille sans modification est de 512 [ko]. La taille avec l'option *–Os* est de 504 [ko].

5.10.3 Conclusion

Lors de la compilation de programmes, pour une utilisation dans un système embarqué, il faut absolument exclure l'option *–g* qui entraîne une augmentation considérable de la taille de l'exécutable. Avec l'option *–Os* il est également possible de réduire la taille de certaines applications.

5.11 Optimisation des applications

5.11.1 Généralités

Un système d'exploitation opérationnel ne sert à rien sans applications. Elles déterminent les fonctionnalités du système embarqué. Ces applications occupent de la place en mémoire de masse. C'est pourquoi certaines optimisations en taille peuvent être judicieuses.

5.11.2 Compilation

Sans reprendre la totalité du chapitre qui parle des options de compilations, il est intéressant de savoir que la plupart des applications disponibles pour Linux sont livrées avec leurs codes sources. Appliquer les conseils du chapitre précédent est donc tout à fait

possible. Les modifications doivent généralement être effectuées dans le fichier *Makefile*. Il ne faut cependant pas s'attendre à une diminution de taille gigantesque du programme.

5.11.3 Compression des exécutables

Un programme est stocké dans la mémoire de masse. Lors de son exécution, le code de celui-ci est chargé en mémoire RAM. Il est possible de comprimer le programme pour en diminuer la taille de stockage. Au lancement de l'application un petit script sera chargé de décompresser le programme et de le charger RAM. La compression est faite à l'aide d' programme de compression. Le choix du programme de compression n'est pas figé. Le code est très facilement modifiable afin d'employer un autre software de compression. Le script le plus populaire qui effectue la transformation de l'application se nomme BZEXE. Le code du script est visible en annexe *Script BZEXE* page 206.

Lorsque l'application est compressée à l'aide ce script, il n'est pas nécessaire pour pouvoir disposer de BZEXE. Par contre, il est évident que le logiciel de compression choisi doit être présent sur la machine. De plus, la compression d'une application oblige la décompression avant exécution, ce qui ralenti le système.

Il est à remarquer que BZEXE utilise, temporairement, de la mémoire soit RAM soit de stockage, pour effectuer la décompression. Un système qui ne contient que très peu d'application n'a pas intérêt à faire appelle aux services de BZEXE.

5.11.4 XIP :Execute in place

XIP ou exécuter sur place permet une économie de mémoire RAM par l'utilisation de la mémoire de masse. Au lieu de charger le programme en RAM pour permettre son exécution, il suffit de l'exploiter depuis le support de stockage (Flash ou ROM) sans chargement en mémoire RAM. Les mémoires de masse de type Flash ou ROM, sont souvent directement adressable par le microprocesseur, à la différence des disques durs. Les systèmes embarqués délaissent l'usage du disque dur pour des périphériques moins sensibles tels que les mémoires Flash ou les ROM.

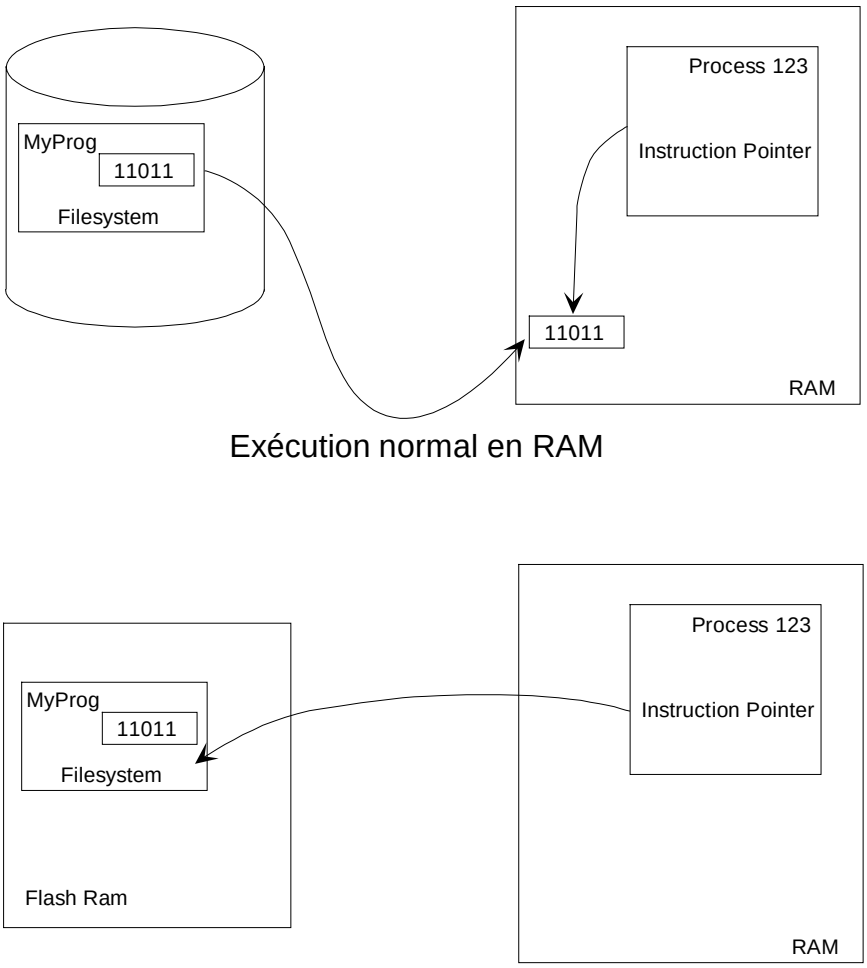


Figure 9 : Mode XIP

Le driver *MTD* qui prend en charge les mémoires Flash et la gestion du système de fichiers *JFFS* supporte, dans une version encore en développement, le mode d'exécution sur place. Cette solution d'économie de ressources n'a malheureusement pas pu être testée. Elle semble cependant forte intéressante car elle évite une double utilisation de la mémoire.

5.11.5 Ecriture des applications

Ce n'est pas dans ce rapport que doit figurer un cours sur la programmation d'applications. Cependant, une sensibilisation du développeur aux contraintes d'un système embarqué semble être de bon alois. Un système embarqué possède des ressources minimales tant au niveau de la mémoire RAM que de la mémoire de masse. Un programme écrit pour une telle machine devra être encore plus propre que pour une machine normale. De plus, une définition précise des fonctionnalités du logiciel évitera l'ajout d'options inutiles. En outre, la maîtrise parfaite du hardware permet une utilisation optimale de toutes les ressources.

5.11.6 Conclusion

Les fonctionnalités d'un système embraqué se trouvent dans les applications qu'il contient. L'optimisation en taille est réalisable lors de la compilation et par la compression des exécutables. Cette dernière méthode bien que très efficace augmente d'une façon significative le temps d'accès.

5.12 Affichage

5.12.1 Généralités

Le PDA[36] est l'exemple le plus accompli d'un système embarqué possédant un afficheur graphique de bonne qualité. Tous les systèmes embarqués n'ont pas l'utilité d'un affichage graphique. Cependant, si l'application n'utilise pas spécifiquement d'affichage, l'administration du système peut en être grandement facilitée. Ce chapitre, propose différentes solutions d'affichage tant au niveau graphique qu'au niveau texte.

5.12.2 Console VGA

Il s'agit du mode d'affichage le plus connu sous Linux. Il nécessite de posséder une carte à la norme VGA. Le code de gestion de la console VGA est inclus dans le code source du noyau. Du fait de sa taille relativement importante, il peut être exclu lors de la compilation. De plus, la gestion de l'affichage par la console demande une quantité de mémoire RAM plus importante. Il est évident que la carte graphique seule ne suffit pas. Il faut encore ajouter un écran LCD ou CRT[37]. L'utilisation des modes graphique, qui sont présentés dans les lignes qui suivent, oblige d'inclure la gestion de la console VGA.

5.12.3 Console série

Linux est un descendant du monde Unix dont les unités de travail étaient, il y a quelques années, de simples terminaux connecter à une unité centrale commune. Linux possède dans son noyau la faculté de permettre des connexions au moyen de terminaux série. L'accès par terminal est identique à un accès direct. Cette option peut s'avérer bien utile pour la maintenance d'un système embarqué. Pour autant que le système embarqué dispose d'une connexion série, il est possible de se connecter au moyen d'un terminal présent sous forme d'un software dans n'importe quel ordinateur portable. Le noyau doit être compilé avec cette option. De plus, certains fichiers de configuration devront être modifiés pour permettre la connexion au moyen d'un port série et d'un câble *nullmodem*. Les détails de configuration sont abordés dans le chapitre *Processus de construction d'un système Linux embarqué* page 88.

5.12.4 Ecran LCD en mode texte

Pour afficher un nombre limité d'information, l'écran LCD à caractères s'avère amplement suffisant. Ce périphérique d'un coup modeste peut être aisément interfacé à un port série, parallèle ou à un bus propriétaire.

[36] personal digital assistant
[37] cathode ray tube

Application propriétaire

Afin de démontrer les possibilités d'interfaçage d'un écran LCD à caractère, une petite application a été écrite pour ce genre de matériel. Le type d'écran choisi est un LCD 2x16 caractères possédant un contrôleur Hitachi HD44780 qui est pratiquement devenu un standard. La liaison avec le hardware a été faite par le biais du port parallèle courant. L'utilisation du port série est également envisageable. Cependant, cette dernière solution est légèrement plus fastidieuse à mettre en œuvre. Les détails de l'interfaçage de l'écran ainsi que le code de l'application sont visibles dans le chapitre *Processus de construction d'un système Linux embarqué* page 98.

LCDproc

LCDproc est un pilote de périphérique pour écran LCD. Il fonctionne sur le principe client-serveur cher au monde Linux et dérivé de XWindows. *LCDproc* est en fait le programme serveur qui s'exécute en tâche de fond. Pour afficher des informations par le biais de ce pilote, il faut y accéder en temps que client. Ce type de dialogue bien que plus lourd en terme de ressource reste bien plus souple, facilitant l'écriture des applications.

Figure 10 : Ecran LCD

LCDproc offre également un support pour plusieurs types d'écran LCD. Il supporte naturellement toute la gamme des écrans à base du contrôleur Hitachi 44780 de 1x8 à 4x20 caractères, interfacés sur port parallèle ou port série. Les écrans des chez CrystalFonz et Matrix-Orbital peuvent également être utilisés sans aucun problème. Sa petite taille (100 [ko]) fait de ce programme un candidat idéal pour les systèmes embarqués. Il faut cependant préciser, qu'il n'a pas était possible de le compiler à l'aide de la librairie *ucLibc*.

5.12.5 Graphique

Un système embarqué peut avoir besoin d'un affichage performant. Un agenda électronique, par exemple, exige des performances graphiques élevées. La programmation complète d'un environnement graphique dépasserait le cadre de ce travail. C'est pourquoi, quelques solutions existantes testées seront brièvement présentées.

Xfree86 3.x.x

Xfree86 est le gestionnaire d'affichage graphique le plus commun sous Linux. Dénommé serveur X, X11 ou XWindow, Xfree86 fonctionne sur le principe de l'architecture client-serveur. Le serveur, Xfree86, fournit « des services graphiques » à des clients qui sont des programmes qui se connectent en utilisant un protocole de communication. Xfree86 ne fournit pas seulement des « services graphiques », il prend également en charge le support de la souris et du clavier. Ainsi un gestionnaire de fenêtres comparable à Windows devra dialoguer avec le serveur X (Xfree86) pour pouvoir afficher du graphisme à l'écran, permettre le contrôle par le biais d'un pointeur,... Le client enverra des requêtes au serveur : trace une ligne, prend en compte l'entrée au clavier… Ce mode de hiérarchie

(client-serveur) peut paraître trop gourment en ressources. Il est cependant tout à fait adapter à un système d'exploitation comme Linux qui se veut orienter réseau. En effet, ce système permet d'utiliser une machine en mode graphique au travers d'un réseau, le client dialoguant avec le serveur X au travers d'Internet par exemple. Xfree86 dans sa version actuelle est devenu énormément lourd au niveau du stockage en mémoire de masse et surtout au niveau du besoin en mémoire RAM. Il est simplement adapter au Pc actuel qui sont avant tout multimédia. Ainsi il supporte pratiquement toutes les cartes et les chipsets graphiques du marché. Son utilisation, adaptée au Pc boulimique qui trônent les places de travail de chacun, n'est pas envisageable dans un système embarqué aux ressources limitées. La librairie *XLib* fournie par Xfree86 pèse de quatre à douze méga octets.

Nano-X ou Microwindows

Nano-X reprend le principe du client-serveur expliqué ci-dessus. Il n'est cependant pas autant encombrant que Xfree. En effet, Nano-X fut développé à l'intention des systèmes minimaux. C'est pourquoi son poids n'excède guère les 200 [ko] pour Nano-X et 256 [ko] pour la librairie minimale souvent appelée *Xlib-like*. De plus, il fut parallèlement développé pour le système d'exploitation Windows CE de Microsoft sous le nom de Microwindows. Les API fournis par les librairies du software sont identiques pour Linux et pour Windows CE. La version pour Linux permet de choisir entre plusieurs solutions graphiques qui sont : l'utilisation de la célèbre librairie graphique *SVGAlib* ou du mode FrameBuffer. En outre, pour faciliter le développement des applications, il est possible de configurer Nano-X pour qu'il vienne se greffer sur X11. Ainsi, Nano-X est utilisable sous X11. Les applications sont donc totalement testables sur l'environnement de développement.

L'emploi de Nano-X à l'aide de la librairie *SVGAlib* n'est pas la meilleure des solutions. En effet, cette librairie n'est pas des plus performantes en terme d'affichage graphique. Elle reste cependant tout à fait acceptable pour des applications graphiques simples. L'option la plus intéressante se situe dans l'utilisation du mode FrameBuffer. Le FrameBuffer du noyau Linux est une solution d'affichage graphique se situant entre l'utilisation de la SVGAlib trop lente, et XWindow trop gourment en ressources. Il permet d'utiliser les modes graphiques de la carte vidéo pour l'affichage de la console. Cela autorise donc de mélanger texte et graphique, pour une consommation mémoire très réduite.

De plus, Nano-X supporte un grand nombre de pointeur graphique tel que la souri et le TouchScreen. Et enfin, Nano-X peut être compilé à l'aide de la librairie minimal *ucLibc*. L'installation ainsi que la configuration sont visibles dans le chapitre *Processus de construction d'un système Linux embarqué* page 100.

PicoGui

PicoGui est un petit client-serveur graphique utilisé dans un grand nombre de projet exploitant Linux. L'entreprise *SmartData*, implantée à Martigny, participe activement au développement de PicoGui intégré dans leur PDA. Ce client-serveur graphique est encore moins lourd que l'application Nano-X. Pour arriver à ce résultat, il a fallut intégrer certains éléments au serveur. Ainsi, les polices d'écriture, les *widgets*[38], et tous les éléments de l'interface graphique, sont inclus dans le serveur. Il n'est donc pas aisé de modifier

[38] Les widegets ,*window – gadget,* englobent les éléments d'une interface graphique tels que boutons, cadres, pointeur…

l'apparence d'un gestionnaire de fenêtre sans s'attaquer au serveur PicoGui. Cependant, la modification de l'apparence graphique peut être réalisée avant la compilation du programme grâce à un système de thème de bureau. PicoGui utilise la librairie SVGlib ou SDLlib. Cette dernière n'est pas indépendante. Elle nécessite l'installation de X11.

Ce manque de modularité est compensé par un gain énorme en mémoire RAM et en mémoire de stockage. En effet, compilé, PicoGui ne prend que 100 [ko]. La librairie graphique pour les applications ne pèse que 6 [ko]. De plus, les besoins en mémoire RAM sont également minuscules. Au lancement de PicoGui, sans aucune application, 5 [ko] de mémoire RAM sont accaparés. En lançant un programme de démonstration, avec un thème, le besoin en mémoire passe à 42 [ko]. Si l'application est lancée sans le thème, l'ensemble n'occupe que 13 [ko] de mémoire RAM. PicoGui supporte également souris, touchscreen et clavier. Et enfin, un très bon tutorial, disponible sur le site http://picogui.org/wiki/ , explique le principe de programmation d'application cliente et expose le mode de fonctionnement détaillé du serveur. L'installation ainsi que la configuration sont visibles dans le chapitre *Processus de construction d'un système Linux embarqué* page 104.

TinyX

TinyX est une version allégée de Xfree86 qui fut mise au point pour des distributions Linux légères. Il peut s'employer sans problème dans des configurations qui possèdent 4 [Mo] de mémoire RAM. Mais attention, car avec une telle contrainte de place en mémoire RAM TinyX doit disposer de mémoire de masse pour effectuer du swapping. Les développeurs du programme préconisent 12 [Mo] pour le swap. Cependant TinyX utilise beaucoup plus de place en mémoire de stockage, environ 3 [Mo].

Il supporte pratiquement tous les chipsets graphiques pris en charge par Xfree86. De plus, son installation sur un système x86 est relativement simple et s'apparente énormément à Xfree86.En outre, la plus grande partie des API standards de Xfree86 sont présents pour faciliter la portabilité des applications. Et enfin, TinyX peut être compilé à l'aide de ucLibc. De part la nécessité de disposer d'une zone de swapping, ce serveur graphique semble plus approprié à une utilisation sur de vieux Pc dont la capacité au niveau de la mémoire RAM est insuffisante pour exécuter Xfree86. Son emploi dans un système embarqué est possible si celui-ci est équipé d'un disque dur.

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. Tous n'ont pas pu être testé. Pour de plus amples informations voici une petite liste sur les autres possibilités: *QtEmbedded, OpenGui, PicoTk, MiniGui, PocketLinux …*

Remarque : Comme indiqué auparavant, les applications ont été testées sur un système x86. La compatibilité avec d'autres architectures n'a pas pu être vérifiée pour cause de disponibilité du matériel et pour cause de temps.

5.12.6 Conclusion

Toutes les applications ne demandent pas forcement des possibilités d'affichage. Cependant, pour la maintenance, il est vivement conseiller de proposer, une connexion par

terminal série, l'augmentation de la taille du noyau ainsi que la place utilisée en mémoire n'étant pas titanesque. L'utilisation d'un écran LCD à caractères peut s'avérer être une solution envisageable pour divulguer un nombre restreint d'information. Les systèmes plus exigeants auront recours à un gestionnaire graphique de petite taille tel que Nano-X ou PicoGui. Ce dernier étant moins gourment en mémoire, il paie cet avantage par un manque de modularité et une plus grande difficulté de programmation des applications clientes.

5.13 Applications minimales pour Linux embarqué

5.13.1 Généralités

L'utilisation d'un système d'exploitation aussi populaire que Linux offre une bibliothèque d'applications énorme. Pourquoi écrire un programme lorsqu'il existe déjà ? Ce petit chapitre présente quelques softwares divers, de petites tailles et libres de droits.

5.13.2 E3

Ce petit programme de 12 [ko], remplit le rôle d'un éditeur de texte en mode console. Il peut-être très utile dans un système embarqué. En effet, un grand nombre de fichiers de configuration est édité en scripts *Shell* pour faciliter la mise à jour et l'administration. Ils peuvent dès lors être modifié par le logiciel e3.

5.13.3 Boa Webserver

Comme son nom l'indique, BOA est un programme implémentant un serveur *HTTP* minimal supportant le *CGI*. De part sa petite taille et sa stabilité, il est utilisé dans un grand nombre d'applications Linux embarquées. BOA peut-être compilé sans aucun problème à l'aide de la librairie *ucLibc* citée précédemment, offrant ainsi un gain de place appréciable. La compilation et l'installation de ce software sont expliqué en détail dans le chapitre *Processus de construction d'un système Linux embarqué* page 96.

5.13.4 Thttpd

Thttpd est un serveur *HTTP* légèrement plus encombrant que BOA en terme de taille. Cependant, outre les fonctionnalités *HTTP*, il supporte le CGI. Par contre, il n'a pas été possible de le compiler pour une utilisation avec *ucLibc*. Il accepte les librairies telles que la *Libc6* ou la *Libc5* standards et peut naturellement être *linké* statiquement.

5.13.5 Ash

Il s'agit d'un interpréteur de commandes de petite taille. *Ash* implémente la plus grande partie des fonctionnalités de *Bash*[39] et garde une entière compatibilité avec ce dernier, un avantage pour l'utilisation des scripts qui peuplent le monde linuxien. De plus, *Ash* est intégré depuis la version 0.8 au programme *BusyBox* décrit précédemment.

5.13.6 Pcmcia-cs

Une grande partie de Pc industriels et de cartes Pc-104 dispose d'un port au standard *PCMCIA*. Card Service pour Linux est un paquet offrant un support des périphériques au standard *PCMCIA* 2.1. Il se charge comme module. Ce pilote prend en charge la plus

[39] Bourn Again Shell : interpréteur de commande principal sous Linux

grande partie des cartes Ethernet, modem, cartes séries, plusieurs adaptateurs *SCSI*, la plus grande partie des contrôleurs ATA/IDE, et certaines cartes mémoires *SRAM* ou Flash. De plus, il est possible de compiler ce logiciel à l'aide de la librairie minimale *ucLibc*.

5.13.7 MAD : Mpeg Audio Decoder

MAD et un décodeur audio *MPEG* de grande qualité. Il supporte *MPEG-1* et *MPEG-2*, les trois layers audio dont le bien connu *MP3*. Sa distribution se fait sous licence GNU. De plus, il offre l'avantage de pouvoir être compilé à l'aide de la librairie ucLibc. Son implémentation particulière le rend fonctionnel sur des architectures dépourvues de *FPU* comme les chips *ARM*.

5.13.8 LSH

LSH est une implémentation minimale du protocole *SSH*. *SSH* est un protocole pour réaliser des connexions cryptées entre un serveur *SSH* et un client *SSH*. Cela ressemble au protocole telnet, sauf que ce dernier ne crypte pas la communication. Ce logiciel est distribué sous licence GNU. Il peut également être compilé à l'aide de la librairie *ucLibc*.

5.13.9 Pppd

Pppd est un module implémentant le protocole *Point to Point Protocol*. Ce programme peut également être compilé à l'aide de la librairie *ucLibc*. La distribution est libre, se faisant sous licence GNU.

5.13.10 OpenSSL

OpenSSL est un ensemble d'outils complet et robuste offrant les avantages du protocole sécurisé SSL. De plus, OpenSSL met à disposition une librairie d'outils cryptographiques. Cette application peut être compilée à l'aide de la librairie minimale *ucLibc*.

5.13.11 BlueZ Bluetooth stack

BlueZ est une implémentation du stack Bluetooth pour Linux. A l'origine, BlueZ fut développé par Qualcomm. Il fut ensuite abandonné par Qualcomm pour devenir un projet Open source. BlueZ Bluetooth peut être compilé à l'aide de la librairie *ucLibc*

5.13.12 uDHCP

uDHCP est une petite application qui implémente les fonctionnalités d'un client ou d'un serveur DHCP. Il peut être compilé à l'aide de la librairie *ucLibc*.

Remarque : Tous les programmes cités ici sont disponibles sur le CD-Rom. De plus, les adresses Internet des sites dédiés à ces applications sont visibles dans la bibliographie en page 114.

5.13.13 Conclusion

Cette liste de programmes pourrait être complétée indéfiniment ou presque. Il existe une multitude d'applications en libre diffusion sur Internet. Certains programmes, cités auparavant, possèdent l'avantage de pouvoir être compilés à l'aide de la librairie optimisée *ucLibc*. Pour les autres, à moins d'opérer des modifications dans les sources du code, il faudra se contenter d'utiliser les librairies standards. Les programmes peuvent naturellement être compilés statiquement, au cas ou l'ensemble du système utilise la *ucLibc*. Cette solution entraîne une augmentation de la taille des exécutables.

5.14 Conclusion

Ce chapitre a permis de mettre en évidence les différentes parties de Linux qui peuvent être optimisées en vue d'une intégration à un système embarqué x86. L'optimisation s'est faite au niveau de l'espace mémoire RAM et de l'espace mémoire de stockage. En effet, la puissance de calcul des systèmes embarqués actuels ne nécessite pas d'opérer des modifications dans le but d'accélérer l'exécution des applications.

Les points qui ont fait l'objet d'une optimisation sont :

- Suppression de l'utilisation du swapping
- Choix d'un système de fichier adapté aux contraintes des systèmes embarqués : JFFS
- Amélioration de la procédure de démarrage et de l'initialisation
- Diminution de la taille de la librairie Libc6 ou alors remplacement la librairie minimale *ucLibc*
- Utilisation du programme BusyBox et TinyLogin afin de remplacer les commandes Linux
- Compression possible de certaines applications

Lors de l'optimisation du système d'exploitation Linux, il ne faut pas tomber dans une radicalisation extrême. De trop importantes modifications risquent d'entraîner un manque de modularité, voire même des problèmes de compatibilité !!!

6 Processus de construction d'un système Linux embarqué

6.1 Introduction

Le chapitre précédent expose de façon théorique les parties pouvant être optimisées. Cette partie applique les conseils d'optimisation. Il peut être utilisé comme un document *HowTo* du monde Linux. Ces recommandations permettent, s'elles sont suivies scrupuleusement, de configurer un Os minimal.

Après une préparation de la mémoire de stockage de la cible, l'arborescence des répertoires sera éditée. La compilation, et le choix des options du noyau feront l'objet d'un sous chapitre. Puis, les librairies utiles seront configurées. Les commandes et utilitaires de Linux pourront être installés sur la mémoire de stockage du système embarqué. L'ajout des scripts de configuration et du programme d'amorçage permettra d'obtenir un Os fonctionnel. La fin du chapitre traitera de diverses options du système d'exploitation telles que l'accès au réseau Ethernet, les solutions d'affichage et l'utilisation d'un terminal série.

6.2 Méthode et plate-forme de développement

6.2.1 Plate-forme de développement

La construction d'un système Linux se fait à l'aide d'un système hôte. Pour ce projet, un Pc de bureau classique compatible x86 équipé d'un processeur Intel Pentium III cadencé à 700 [Mhz] et disposant de 100 [Mo] d'espace disque libre, servit de plate-forme de développement. Une distribution Linux classique, la Suse 7.1, fut installée sur ce dernier. Afin de faciliter la gestion des fichiers, Xfree86 et le gestionnaire de fenêtre KDE2 entrèrent également dans la configuration de la machine. Ils ne sont cependant pas nécessaires, le mode texte, par le biais de la console, étant totalement suffisant.

Certaines applications doivent être présentes sur la machine hôte :

- Il faut sur le système hôte un noyau 2.4.0 ou supérieure
- Une librairie récente comme la Libc6
- Le compilateur GNU *gcc* pour x86 version 2.95
- L'outil *Autoconf-Automake* livré avec la plus grande partie des distributions grand public
- Le chargeur d'amorce Lilo doit être également présent dans sa version installée.

- Un éditeur de texte quelconque

Toutes les sources, les scripts, ainsi que les utilitaires utilisés sont disponibles sur le Cd-Rom en dernière page. Il n'est pas obligatoire d'utiliser la distribution Suse 7.1. Plus la machine hôte est puissante moins la compilation d'applications prendra du temps. A titre d'exemple, la compilation d'un noyau Linux dure environ 45 minutes sur un Pentium 133 [Mhz] contre 10 minutes sur un Pentium III 700[Mhz]. La machine destinée à recevoir le système optimisé, étant un compatible x86 au format réduit, il n'est pas nécessaire de configurer *gcc* pour effectuer du *cross-developpement,* l'architecture étant identique à celle de la machine hôte. Et enfin, la construction du système, sur la machine de développement, doit être réalisée en mode administrateur. Il faut se *loguer* à l'aide du nom d'utilisateur *ROOT*.

6.2.2 Méthode de développement

En parcourant Internet, il est aisé de trouver de petites distributions pour systèmes minimaux. Leur utilisation ne fut pas envisagée dans ce projet : le but étant d'optimiser Linux et non d'avoir recours à un Linux déjà optimisé.

Pour élaborer un système Linux, en tenant compte des conseils théoriques du chapitre précédent, deux solutions sont possibles.

- Partir d'une distribution Linux existante, comme la Suse ou la Red Hat

- Construire un système en partant de zéro.

Le titre de ce sous chapitre étant *construire* et non *détruire*, c'est naturellement la deuxième solution qui fut retenue. La construction d'un système Linux permet de connaître tous les aspects du système d'exploitation. Théoriquement l'étude de n'importe quel système, n'apporte pas autant que la conception de ce même système. De plus, les distributions actuelles incluent une multitude d'options inutiles et utilisent des conventions propriétaires.

6.3 Construction du système

Les pages qui suivent, expliquent la construction d'un Linux optimisé pour les machines aux ressources limitées. Malheureusement, toutes les solutions n'ont pas pu être testées car le matériel spécifique n'a pas pu être trouvé. Elles ont été remplacées par des choix moins judicieux mais entièrement fonctionnels, pour permettre la mise en place du système d'exploitation. Ce système se nomme ELISE : *Embedded LInux SEtup.*

Remarque : Le système construit dans ce document est sous licence GPL. Pour plus d'information, il faut ce référer à l'annexe *Licence GPL* en page 315.

6.3.1 Installation du système de fichiers

Généralités

Le système de fichiers le mieux adapté aux machines embarquées est le *JFFS*. Il n'a cependant pas était possible de le tester. En effet, son utilisation requiert la présence d'un périphérique de mémoire de masse de type Flash. En l'absence de ce dernier, l'emploi d'un disque dur IDE classique et du système de fichier *Ext2* semble être une solution avantageuse. Le disque dur est installé sur un port IDE libre. Sous Linux, les périphériques portent le nom de *hda* pour l'IDE primaire maître, *hdb* pour l'IDE primaire esclave, *hdc* pour l'IDE secondaire maître, *hdd* pour l'IDE secondaire esclave. Puis, pour chaque disque, il est possible de générer des partitions qui sont numérotées de 1 à … Ils sont situés dans le répertoire */dev*. La quatrième partition d'un disque dur, connecté sur le port IDE secondaire en temps qu'esclave, a comme dénomination *hdc4*. Lorsque la configuration est terminée, le disque peut être connecter à la machine cible. Ceci est possible car le système hôte et le système embarqué sont tous les deux des compatibles x86.

Création de la partition

La création de la partition destinée à recevoir Linux est faite à l'aide de l'utilitaire *cfdisk*. Le disque dur est connecté au port IDE secondaire en temps que maître. Il se dénomme donc *hdc*.

```
cfdisk hdc
```

Il n'est pas nécessaire de faire plusieurs partition. Premièrement le swapping n'est pas utilisé. Deuxièmement, un système embarqué n'en a pas l'utilité. Avant de créer l'unique subdivision, il faut supprimer celles existantes. Les données, présentes sur le disque, sont dés lors perdues !!

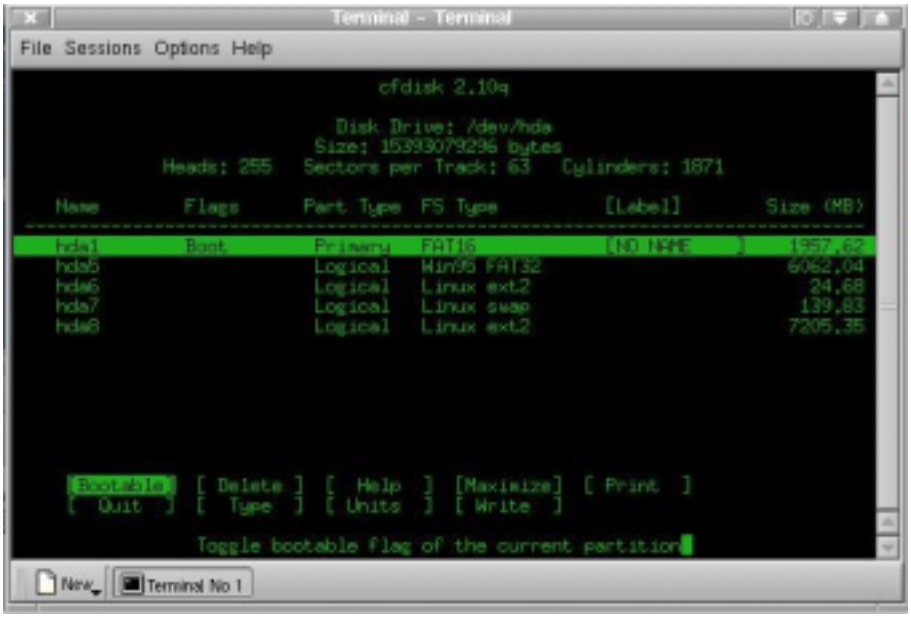


Figure 11 : Utilitaire cfdisk

La partition créée doit être du type LINUX..

Implantation du système de fichiers

La création du système de fichiers entraîne le formatage de la partition sélectionnée. Lors de la création du celui-ci, il est nécessaire d'indiquer la taille d'un bloc en octets. Cette taille doit impérativement être un multiple de la taille d'un secteur du support physique de données, parce que les blocs ne peuvent contenir qu'un nombre entier de secteurs. Pour un disque dur, la taille des secteurs est fixée à 512 octets. Ext2 autorise des bloc de 1024 à 4096 octets. De même, il faut spécifier le nombre d'inode de la partition. Il est possible de spécifier ce nombre soit directement, ou d'indiquer seulement le taux du nombre d'octets par inode de la partition. Il faut savoir que le nombre maximum d'inode possible est bien entendu le nombre total de blocs, puisque tout fichier non vide requiert au moins un bloc Le choix s'étant fait sur le système *Ext2*, il faut utiliser l'utilitaire *mktfs :*

```
mke2fs hdc1
```

L'absence d'options oblige l'utilitaire à déterminer des valeurs optimales en fonctions de la place sur la partition. La syntaxe de la commande *mke2fs* est donnée ci-dessous :

```
mke2fs -b Taille Fichier
```

Taille représente la taille d'un bloc en octets. De même, le nombre d'octets par inode peut être précisé avec l'une des options *–i :*

```
mke2fs -i Octets Fichier
```

Octet est le rapport de la taille de la partition en octets par le nombre d'inodes à créer. Il est possible d'indiquer directement ce nombre avec la commande suivante :

```
mke2fs -N nombre fichier
```

Le disque dur est prêt à recevoir les différents éléments du système d'exploitation Linux. Il est maintenant possible de *monter*[40] le périphérique pour l'inclure au système de fichiers du système hôte afin d'y avoir accès en lecture et en écriture. Pour ce faire, il faut créer un répertoire qui recevra le système monté.

```
mkdir  –p /mnt/ELISE
```

Le répertoire */mnt/ELISE* sera pour le reste de la construction du système le répertoire cible. Il convient d'être attentif à chaque fois qu'un élément doit être copié, supprimé ou créé dans ce répertoire car une erreur dans l'écriture du chemin peut entraîner des pertes de données sur la machine hôte.

Le périphérique peut y être monté :

```
mount –t ext2 /dev/hdc1 /mnt/ELISE
```

Ainsi, le disque dur peut être accédé par l'intermédiaire du répertoire */mnt/ELISE*

Remarques : Les outils de formatage et de création du système de fichiers *JFFS* sont présents sur le Cd-Rom. Ils sont accompagnés de divers outils de contrôle pour les mémoires Flash.

[40] monter : rendre un média accessible en lecture et en écriture

6.3.2 Image Initrd ou fichier binaire contenant l'OS

Généralités

Linux offre la possibilité de créer un disque compressé qui sera chargé en mémoire RAM. Ce chapitre décrit la procédure à suivre pour concevoir une image du système d'exploitation. Celle-ci est encapsulée dans un fichier. Il est donc envisageable d'utiliser le code binaire du fichier pour implémenter une mémoire non volatile.

Pré requis

Pour créer cette image dans un fichier il faut utiliser un périphérique de boucle (loopback). Il permet de faire croire au système d'exploitation que le fichier est un périphérique normal. Si le périphérique de boucle *(/dev/loop0* ou */dev/loop1*,…) n'est pas présent, sur la machine hôte, il faut l'installer. En premier lieu, il s'agit de le créer à l'aide de la commande :

```
mknod /dev/loop0 b 7 0
```

Ensuite, il est nécessaire de mettre à jour les utilitaires *mount* et *umount* . Ils sont disponibles sur le Cd-Rom dans le paquet *util-linux*.

Il faut comme précédemment, créer un répertoire destiné à recevoir le système monté.

```
mkdir  –p /mnt/ELISE
```

Création du fichier de stockage

Il est maintenant possible de créer le fichier qui recevra l'ensemble de l'Os.

```
dd if=/dev/zero of=/tmp/MyFile bs=1k count=3000
```

La commande *dd* initialise à zéro l'ensemble des bits du nouveau fichier */tmp/MyFile* afin que la compression soit plus efficace. Ce fichier possède une taille 3000 [ko]. (1[ko] x 3000).

La prochaine étape consiste à construire le système de fichiers. Le noyau Linux ne sait charger que deux types de systèmes de fichiers : *Minix* et *Ext2*. Pour créer un fichier binaire destiné à une mémoire Flash, il est possible d'utiliser un autre système de fichiers tel que *JFFS*.

```
mke2fs –i 120 /tmp/MyFile
```

L'option *–i 120* permet de définir le nombre de *inode*. Dès lors le fichier peut être monter afin qu'il puisse être accédé en lecture et écriture.

```
mount –o loop –t ext2 /tmp/MyFile /mnt/ELISE
```

L'option *–o loop* est fondamental lorsqu'il s'agit de périphérique de boucle. Le fichier est maintenant accessible comme n'importe quel média de stockage. De ce fait, les manipulations qui suivent, peuvent être appliquées comme pour le disque dur. Lorsque le système sera terminé, il faudra le compresser à l'aide de *gzip*.

```
umount /mnt/ELISE

dd if=/tmp/MyFile  bs=1k  |  gzip –v9  initrd.gz
```

L'image compressée porte le nom de *initrd.gz*.

6.3.3 Création des répertoires nécessaires

Création des répertoires

Linux ne peut pas fonctionner sans une arborescence de répertoires minimale. La structure proposée ici n'est pas conforme au standard FHS[41]trop lourd et pas adapté aux systèmes embarqués.

[41] Filesystem Hierarchy Standard www.pathname.com/fhs

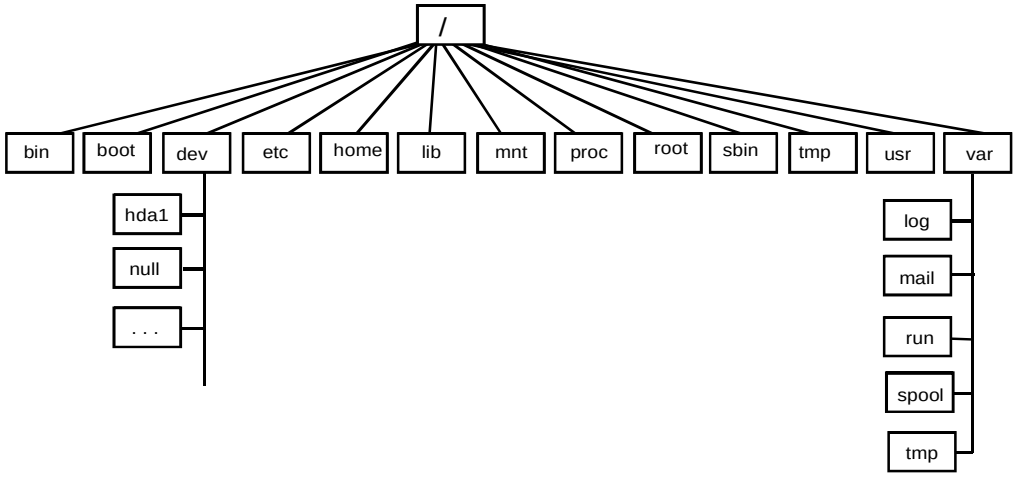


Figure 12 : Arborescence minimale

```
mkdir –p bin boot dev/pts etc home lib mnt proc root sbin tmp usr var

cd /mnt/ELISE/var

mkdir log mail run spool tmp
```

Si le système cible est destiné à être compressé, pour réaliser une *Initrd*, le répertoire *boot* n'est pas nécessaire. Le noyau, chargé de décompresser le système de fichiers, ne peut naturellement pas être inclus dans celui-ci.

Normalement, les répertoires sont créés avec le mode permission 755[42], ce qui n'est pas souhaitable pour tous les répertoires. Le premier changement est à appliquer au répertoire */mnt/ELISE/root* en lui attribuant la permission 750. Ceci pour s'assurer que tout le monde ne peut pas entrer dans le répertoire */root*. La deuxième modification est le mode 1777 pour les répertoires *tmp*. De cette façon, tous les utilisateurs peuvent écrire dans le répertoire */tmp*, mais ne peuvent pas supprimer les fichiers des autres utilisateurs.

```
cd /mnt/ELISE &&

chmod 0750 root &&

chmod 1777 tmp var/tmp
```

L'arborescence au standard FHS est visible en annexe *Arborescence FHS des répertoires* en page 118. Elle possède également la définition de chaque répertoire.

42 Le système de permission est détaillé dans le chapitre L'OS Linux

Création des fichiers périphériques du répertoire /dev

Comme évoqué au chapitre *L'OS Linux* page 11, les systèmes d'exploitation de type UNIX sont basés sur la notion d'un fichier, qui n'est rien d'autre qu'un conteneur d'informations structurées comme une séquence d'octets. Ainsi, les périphériques d'entrées/sorties sont traités comme des fichiers. De ce fait, les *API* utilisés pour interagir avec des fichiers standard peuvent être appliqués sur les périphériques.

Les fichiers de périphérique servent à représenter la plupart des périphériques d'entrées/sorties pris en charge par Linux. Chaque fichier possède en plus du nom trois attributs :

- *Type bloc ou caractère* : Un périphérique bloc peut transférer un bloc de donnée de taille fixe en une opération. Il s'agit des disques durs, lecteurs de disquettes, de Cd-Rom… Un périphérique caractère transfert généralement des caractères séquentiellement. Il s'agit de la console, du port série…

- *Major Number* : Nombre compris entre 1 et 255 qui identifie le type de périphérique.

- *Minor Number* : Nombre qui identifie un périphérique spécifique parmi un groupe de périphériques qui se partagent le même *Major Number*

La commande *mknod* sert à créer des fichiers périphériques. Elle reçoit le nom du fichier de périphérique, la permission d'accès en mode absolu, son type et les *Major* et *Minor Number* en tant que paramètre. Les lignes qui suivent, montrent le code pour la création de *hda, hda1, hdc1, et hdd1*. Le script complet est visible en annexe *Script de création des fichiers périphériques* en page 195. De plus, il est disponible sur le Cd-Rom.

```
Cd /mnt/ELISE/dev
mknod -m 660 hda b 3 0
mknod -m 660 hda1 b 3 1
mknod -m 660 hdc1 b 22 1
mknod -m 660 hdd1 b 22 61
```

Remarque : Lors de la création d'une système pour disquette, il faut ajouter le périphérique lecteur de disquette *fd0* dans */mnt/ELISE/dev*.

```
cd /mnt/ELISE/dev
mknod 660 fd0 b 2 0
```

Le script présenté crée de multiples périphériques qui ne sont pas tous utiles. Il est judicieux de supprimer ceux qui sont superflus.

Particularité de la carte réseau

Certains périphériques d'entrées/sorties ne comportent pas de fichier de périphérique correspondant. Les cartes réseau en sont l'exemple le plus significatif. Chaque carte réseau se voit attribuer un nom allant de *eth0* à *ethx*. Cependant ce nom ne correspond à aucun fichier de périphérique et ne comporte aucun inode correspondant.

Remarque : La liste de l'ensemble des fichiers de périphériques, avec leurs attributs respectifs, est visible en annexe *Fichiers périphériques* page 125.

6.3.4 Modification du code source du noyau

Généralités

Ce chapitre présente quelques modifications du code source du noyau Linux. Elles ne sont pas importantes et ne permettent pas d'obtenir un gain en mémoire RAM conséquent. De plus, la place économisée sur le noyau compressé est imperceptible. Cependant, il s'agit ici de démontrer la liberté qu'offre Linux au développeur. Le but de ce projet étant de fournir une distribution minimale complète, il n'a pas était possible d'explorer en détail le code du kernel pour l'optimiser au maximum.

Installation des sources sur la machine hôte

Pour modifier le noyau il faut disposer des sources de celui-ci. Elles sont disponibles sur le Cd-Rom pour la version 2.4.4. Il suffit de décompressé le paquet dans le répertoire */usr/src/linux2.4.4* sur la machine hôte bien entendu. Les modifications exposées dans les lignes qui suivent n'ont pas pu être testées sur un noyau d'une version différente.

Modification de certaines constantes du code sources

Limitation du nombre de système de fichiers
If faut modifier la constante *NR_SUPER 256* contenu dans le fichier source */include/linux/major.h*

```
NR_SUPER 64
```

Le gain en mémoire RAM est d'environ 10 [ko]

Limiter le nombre de consoles
If faut modifier les constantes *MAX_NR_CONSOLES 63 et MAX_NRE_USER_CONSOLES 63* situées dans le fichier source */include/linux/tty.h*.

```
MAX_NR_CONSOLES 3
```

```
MAX_NRE_USER_CONSOLES 3
```

Le système ne peut ensuite utiliser que quatre consoles (0-3). Le gain en mémoire RAM est d'environ 5 [ko].

Réduire la taille du buffer pour les messages affichés par le noyau
If faut modifier la constante *LOG_BUF_LEN 4096* contenue dans le fichier */kernel/printk.c*

```
LOG_BUF_LEN 1024
```

Cette modification n'est pas pénalisante pour l'affichage des messages. Le buffer est suffisamment grand pour contenir 80 caractères. La place économisée en mémoire est de 3[ko].

Suppression des messages aux démarrages

Les messages de boot statuant sur l'initialisation du système sont, pour une grande partie d'individus, totalement incompréhensibles. Ainsi, en cas de disponibilité d'une console de visualisation, il peut être intéressant de les supprimer.

Ceci peut être fait très facilement en ajoutant le code suivant au sommet de la fonction *printk()* définie dans les sources du noyau dans *kernel/printk.c*.

```
switch(myAnim) {
case 0 :
  (*console_print_proc)("\\");
  break

case 1 :
  (*console_print_proc)("|");
  break

case 2 :
  (*console_print_proc)("/");
  break

default :
  (*console_print_proc)("-");
```

```
  break
}

if (myAnim == 3)
  myAnim = 0;
else
  myAnim++;
(*console_print_proc)("\r");
return 0 ;
```

6.3.5 Compilation du noyau

Généralités

La compilation du noyau pour Linux est une phase obligatoire dans la construction d'un système Linux optimisé. Au premier abord, cela peut paraître compliqué. Il n'en est rien. Linux étant un logiciel libre, il se doit d'offrir un système de compilation facile à mettre en œuvre.

Sélection des options de configuration

Il faut en premier lieu se diriger dans le répertoire des sources de Linux.

```
cd /usr/src/linux2.4.4/linux
```

Linux offre trois outils pour la configuration du noyau.

```
make config

make menuconfig

make xconfig
```

La première ligne de commande autorise la configuration de Linux en mode console. La deuxième solution offre un menu de configuration à l'aide de boites de dialogue, toujours en mode console. La dernière possibilité, la plus confortable, permet de paramétrer le noyau sous XWindow. Toutes les options de compilation pour le noyau 2.4.x, sont décrites en détail dans l'annexe *Options de compilation du noyau* en page 272. Il est important de n'inclure que les pilotes nécessaires pour une optimisation maximale.

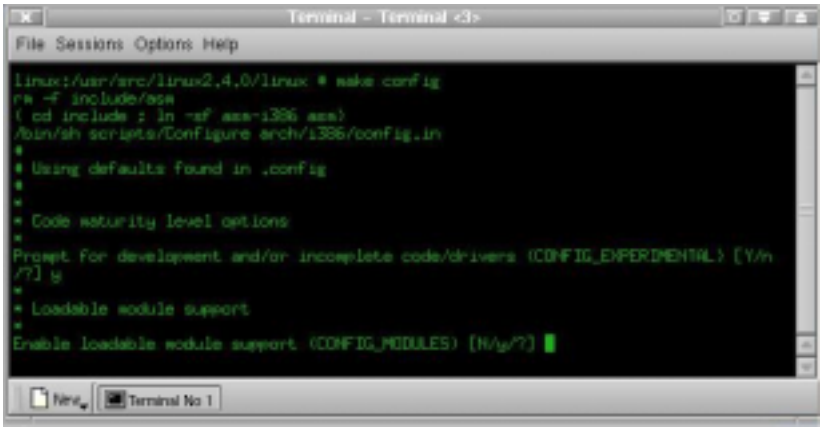


Figure 13 : make config

Figure 14 : make menuconfig

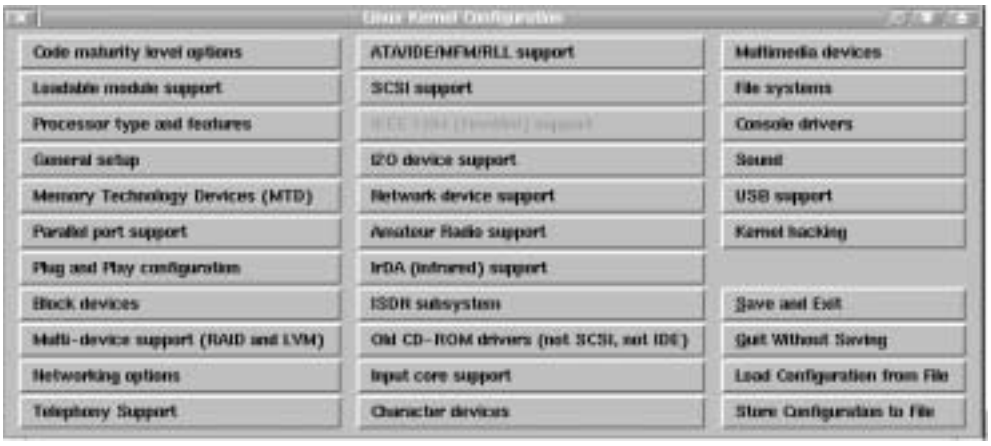


Figure 15 : make xconfig

Le *Makefile* principal, situé dans le répertoire */usr/src/linux2.4.4/linux*, peut être modifié. Il est possible d'y ajouter l'option *–Os*.

```
HPATH   = $(TOPDIR)/include
FINDHPATH     =     $(HPATH)/asm     $(HPATH)/linux     $(HPATH)/scsi
$(HPATH)/net

HOSTCC  = gcc
HOSTCFLAGS    = -Wall -Wstrict-prototypes -Os -fomit-frame-pointer

CROSS_COMPILE       =

#
# Include the make variables (CC, etc...)
```

Le choix des options terminé, il est possible de compilé le noyau grâce aux commandes :

```
make dep

make clean

make bzImage
```

Make dep va réaliser les dépendances correctes pour la compilation des sources. C'est une étape très importante dans la suite de la procédure de compilation du noyau. *make clean*, détruit tous les fichiers objets et autres fichiers que le noyau compile. Puis *make bzImage* compile le noyau et compresse le code. Le noyau est placé dans le répertoire : */usr/src/linux2.4.4/linux/arch/i386/boot.* La durée de la compilation dépend de la plate-forme de développement. Elle peut durer quelques minutes sur un Pentium III, jusqu'à quelques jours sur un vieux 386. Si la compilation échoue, il y a de forte chance pour qu'il y ait un problème au niveau du hardware de la machine hôte.

Le noyau ainsi compilé peut-être copié dans le répertoire de boot du système cible.

```
cp  /usr/src/linux2.4.4/linux/arch/i386/boot/bzImage  /mnt/ELISE/boot
```

Si le système cible est destiné à être compressé pour réaliser une *Initrd*, le répertoire *boot* n'existe pas.

Options de compilation pour l'utilisation d'une initrd

L'utilisation d'un système de fichier compressé qui est stocké et utilisé en mémoire RAM oblige le noyau à être compilé avec certaines options.

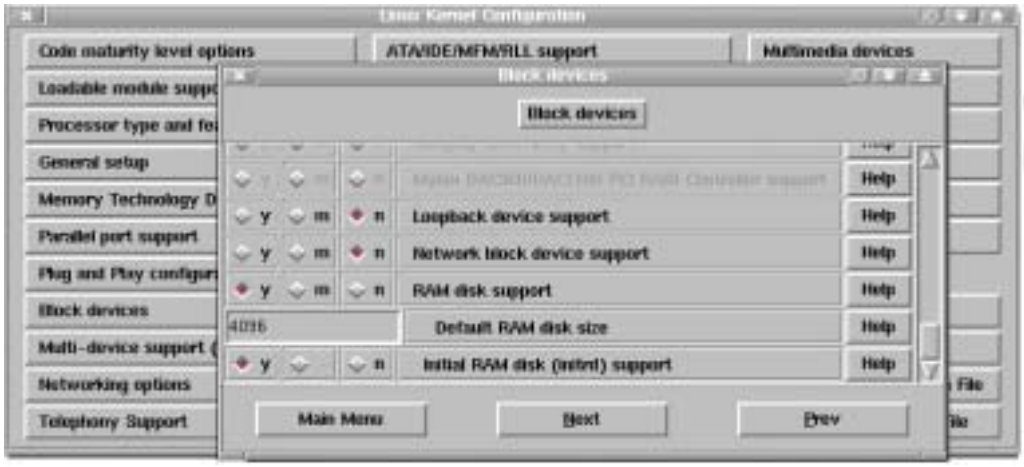


Figure 16 : Option de compilation pour initrd

6.3.6 Installation des librairies

Généralités

Les librairies sont des éléments encombrants dans un système d'exploitation. L'approche utilisée ici consiste soit à remplacer purement et simplement la librairie standard par une librairie minimale soit à ne garder que les éléments essentiels.

Utilisation de la librairie classique optimisée

La distribution du système hôte étant récente, elle utilise la *Libc6* ou *Glibc*. Toutefois, il est tout à fait possible d'appliquer ce qui suit pour une librairie antérieure telle que la Libc5. Sa taille étant moins importante, il s'agit d'une solution envisageable. Il faut être attentif qu'une application faite pour être compilée à l'aide de la *Libc5* pourra sans aucun problème utiliser la *Libc6*, le contraire n'étant pas toujours vrai.

Les librairies principales se trouvent dans le répertoire */lib*. Pour déterminer quels éléments doivent être inclus dans le système Linux minimal, il faut, en premier lieu, disposer de toutes les applications qui doivent s'exécuter sur le système embarqué. Il ne faut rien oublier : commandes, programme d'initialisation, applications, etc. Ensuite, à l'aide de l'utilitaire *ldd, il* est possible de déterminer, pour chaque application, quelle librairie est utilisée. Par exemple, en tapant sur la machine hôte :

```
ldd /usr/bin/cat
```

le système renvoie l'information suivante:

```
```

```
libc.so.6 => /lib/libc.so.6    (0x40007000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2    (0x00000000)
```

L'utilitaire *cat* a besoin de la librairie *libc.co.6* et de la librairie *ld-linux.so.2*. La flèche indique que la librairie possède un lien auquel se réfère le programme. Ainsi, si le système utilise *cat* il faudra inclure *libc.so.6* et *ld-linux.so.2* dans le répertoire */mnt/ELISE/lib*.

Le nombre de commandes Linux étant importantes, opérer une telle sélection manuellement est un travail fastidieux. Toutefois, il existe un script qui effectue un trie parmi la librairie, et en recompose une contenant seulement les éléments nécessaires. Ce script *mklibs.sh* est visible en annexe *Réduction des librairies C* page 211.

Installation de la librairie ucLibc

La librairie *ucLibc* version 0.9.8 se trouve sur le Cd-Rom. Il est cependant tout à fait possible de se rendre sur le site Internet[43] des développeurs pour télécharger la dernière version. D'abord, le paquet doit être décompressé dans un répertoire de la machine hôte, */usr/src/uclibc*, par exemple. Le fichier *Config* doit être édité pour indiquer le chemin d'installation sur la machine hôte : */usr/i386-linux-uclibc-2.4.4*.

Puis, il ne reste plus qu'à faire confiance au concepteur de la librairie qui fournissent les scripts d'installation. Pour compiler la librairie il faut simplement exécuter la commande *make* depuis le répertoire d'installation. Si un message d'erreur s'affiche durant cette procédure, il ne sert à rien de la continuer. Le *make clean* sert à effacer les fichiers objets d'une précédente compilation.

```
make clean

make
```

UcLibc peut être installé sur le système embarqué, mais également sur la plate-forme hôte pour permettre de compiler les applications destinées au Linux utilisant la *ucLibc*.

```
make install
```

```
make PREFIX=/mnt/ELISE install_target
```

[43] http://uclibc.org

La première ligne de commande installe *ucLibc* sur l'ordinateur de développement. La deuxième met en place *ucLibc* sur le système cible *: /mnt/ELISE*. L'établissement de la librairie sur la machine hôte permet la compilation des applications à l'aide de *ucLibc*. Celle-ci fournit un *wrapper* qui comporte en fait les indications pour que *gcc* utilise cette bibliothèque minimale. Pour utiliser la *ucLibc* , il faut, dans le fichier *Makefile* de l'application à compiler, ajouter ou remplacer l'option CC par:

```
CC=/usr/i386-linux-uclibc-2.4.4/usr/bin/i386-uclibc-gcc
```

Dans le répertoire */usr/i386-linux-uclibc-2.4.4/usr/bin*, il existe plusieurs *wrapper* pour autant de compilateurs différents.

La librairie *ucLibc* est considérablement allégée par rapport à une bibliothèque classique telle que la *Libc6*. C'est pourquoi, toutes les applications écrites pour une librairie traditionnelle ne pourront pas être compilées à l'aide de la *ucLibc*. Quelques modifications mineures suffiront, peut-être, pour permettre la compilation. Cependant, bien souvent il faudra se résigner à utiliser la librairie classique ou alors il faudra simplement linké de manière statique l'application, au risque de voir sa taille grandir.

Remarque : Comme indiqué au début de ce chapitre, le système hôte doit posséder un noyau 2.4.x pour pouvoir compiler des applications utilisables sur le système embarqué intégrant un noyau 2.4.4. L'emploi d'un noyau 2.2.x, par exemple, nécessitera d'installer un noyau 2.2.x sur le système hôte et de lui adjoindre *gcc*. Dans le cas contraire, l'exécution d'une application compilée sur une machine à kernel 2.4.4 et exécutée sur un système à noyau 2.2.x ne pourra pas être possible et se soldera par l'affichage du message d'erreur : *kernel too old*

6.3.7 Installation des commandes minimales

Généralités

Les commandes Linux servant à l'administration de Linux ne sont pas utilisables dans leur forme originale, pour intégrer un système embarqué. En effet, leur taille est bien trop imposante. Elles sont donc remplacées par deux applications chargées de fournir tous les utilitaires indispensables, pour l'administration et le fonctionnement d'un Linux.

Configuration de BusyBox

BusyBox réunit dans une même application plusieurs utilitaires Linux. Le paquet se trouvant sur le disque dur doit être décompressé sur la machine hôte dans le répertoire */usr/src/busybox*, par exemple. Il s'agit de la version 0.60.2 de *BusyBox*.

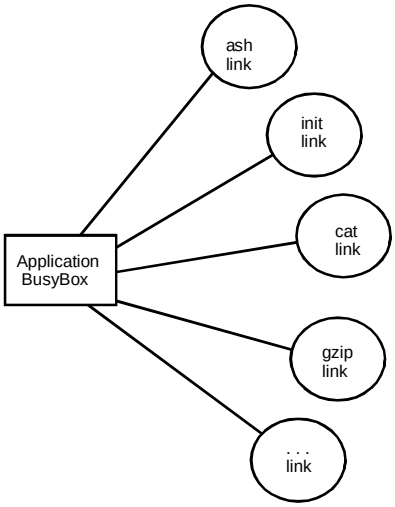


Figure 17 : Principe de BusyBox

La compilation et l'installation de l'application sont grandement facilitées. En effet, avant la compilation il convient d'éditer à l'aide d'un simple éditeur de texte le fichier *Config* présent dans le répertoire de *BusyBox*. Celui-ci possède pour chaque utilitaire une entrée qui peut être mise en commentaire à l'aide du signe # dans le but de ne pas inclure l'utilitaire lors de la compilation. Il ne sera bien entendu pas disponible. *BusyBox* propose plusieurs Shell comme *msh* ou *ash*. *Msh* est une version ultra-dépouillée. C'est pourquoi, pour un confort d'utilisation plus important, il convient d'utiliser *ash*. De plus, *BusyBox* remplace également le logiciel d'initialisation *init*. Celui-ci comporte certaines restrictions. En effet, il ne supporte pas le démarrage au standard System V avec les niveaux d'exécution.

Il reste à choisir la librairie. Si le choix se porte sur la librairie standard, il ne faut plus rien modifier. Dans le cas ou la librairie *ucLibc* doit être utilisée, il suffit d'ouvrir le fichier *makefile*. Ensuite, la mise en commentaire du code, destiné à permettre la compilation à l'aide de *ucLib, c* doit être effacée.

Il est maintenant envisageable d'achever la compilation et l'installation du programme. La compilation ce fait à l'aide de :

```
make clean
make
```

Ensuite, un utilitaire d'installation se charge de copier le programme sur la cible, d'éditer les liens pour chaque application, et de placer les liens dans les bons sous répertoires */bin*, */sbin.*

```
make PREFIX=/mnt/ELISE install
```

De nombreux scripts utilisent, pour accéder au *Shell*, le lien *sh*. Comme le choix c'est porté sur un interpréteur de commande autre que celui édité par défaut, il faut réaliser ce lien dans le répertoire */mnt/ELISE/bin*.

```
ln /bin/ash /bin/sh
```

La description de toutes les commandes *BusyBox* est visible en annexe *Description des commandes BusyBox* page 231.

Configuration de TinyLogin

TinyLogin fonctionne sur le même principe que *BusyBox*. Il ne s'agit en fait que d'un programme complémentaire, qui contient les utilitaires d'accès multi-utilisateurs. Le paquet est disponible sur le Cd-Rom et doit être décompressé sur la machine de développement, pour être utilisé. La version disponible porte le numéro 0.80.

La configuration du logiciel est semblable à *BusyBox*. Il faut également éditer le fichier *Config* pour choisir quels utilitaires doivent être compilés. Cependant, pour compiler *TinyLogin* à l'aide de la *ucLibc*, il faut utiliser le *wrapper* créé à l'installation du *ucLibc* sur la machine hôte. Dans le fichier makefile il faut ajouter :

```
CC=/usr/i386-linux-uclibc-2.4.4/usr/bin/i386-uclibc-gcc
```

Il est maintenant possible de compilé, est d'installer *TinyLogin,* à l'instar du programme *BusyBox* :

```
make clean

make

make PREFIX=/mnt/ELISE install
```

TinyLogin n'est pas essentiel dans tous les types de systèmes embarqués. En effet, une application dont le but et la régulation d'un processus ou l'automatisation d'une chaîne de montage n'utilise pas forcement les options du mode multi-utilisateurs.

La description de toutes les commandes *TinyLogin* est visible en annexe *Description des commandes TinyLogin* page 270.

6.3.8 Configuration des scripts d'initialisation minimaux

Généralités

Les programmes indispensables étant installés, il faut maintenant éditer un certain nombre de fichiers de configuration, utiles au moment de l'initialisation du système. Ces fichiers sont situés dans le répertoire */mnt/ELISE/etc*.

Le fichier inittab

Au moment du démarrage, le noyau va chercher le programme *init* dans les répertoires */bin* et */sbin*. Ce dernier a recours au fichier *inittab*, pour initialiser le système. Le fichier *inittab*, employé ici, n'est pas conforme au standard de démarrage System V.

Pour concevoir ce fichier, il suffit de fabriquer un nouveau fichier texte, à l'aide d'un éditeur de texte, dans le répertoire */mnt/ELISE/etc*, et de le nommer *inittab* (en minuscule).

```
#etc/initab for ELISE by GF
#config file launch by init at the start

::sysinit:/etc/init.d/rcS

#::respawn:/sbin/getty 9600 ttyS0
tty0::respawn:/bin/login
tty1::askfirst:-/bin/login

::ctrlaltdel:/bin/umount -a –r
```

La ligne *::respawn:/sbin/getty 9600 ttyS0* mise en commentaire permet l'emploi d'un terminal en mode console, sur le port série 1 : *ttyS0*.

Cette configuration permet d'utiliser deux consoles (*tty0*-*tty1*). De plus, ce fichier lance le script */etc/init.d/rcS* qui se charge du lancement de diverses applications. La dernière ligne définit les commandes à lancer en cas d'interruption du système.

Le fichier fstab

Le script *fstab* est exécuté dans le but de monter automatiquement certain périphérique. Ainsi, il ne sera pas nécessaire, sur un ordinateur équipé de plusieurs partitions, de les monter à chaque fois, l'opération se faisant automatiquement. Le fichier doit également être créé dans le répertoire */mnt/ELISE/etc* et il doit se nommer *fstab*.

```
#etc/fstab for ELISE by GF
#indicate the partition that must be mounted
```

```
#
/dev/hda1     /     ext2  defaults 1 1
proc          /proc proc  defaults 0 0
```

Dans le cas d'un système *initrd* sur disquette, le fichier *fstab* se présente comme ceci :

```
#etc/fstab for ELISE on RAM by GF
#indicate the partition that must be mounted
#
/dev/ram0     /     ext2  defaults 1 1
proc          /proc proc  defaults 0 0
```

Le fichier rcS

Dans le système Linux le fichier *rcS* sert à lancer des applications, au moment du démarrage. Il peut s'agir de programmes servant à initialiser le réseau, d'application spécifique au système embarqué, etc. Ce fichier est généré dans un sous répertoire de */mnt/ELISE/etc*, le répertoire */mnt/ELISE/etc/init.d*. Ceci pour permettre d'y ajouter d'autres fichiers de configurations qui accompagnent certaines applications.

```
mkdir /mnt/ELISE/etc/init.d
```

Le fichier peut maintenant être créé, à l'aide d'un éditeur de texte. Il doit porter le nom de *rcS*.

```
#! /bin/sh
#/etc/init.d/rcS for ELISE by GF

PATH=/bin:/sbin:/usr/bin:/usr/sbin

echo "Setting ELISE System"

#mount system for read and write

echo "Setting up system for read and write...."
```

```
/bin/mount -n -o remount,rw /

#mount all dev locate in fstab
/bin/mount –a
```

Les fichiers passwd et group

Ces deux fichiers sont utiles par le programme *login* inclus dans *TinyLogin*. Ils contiennent les noms et les mots de passes cryptés des utilisateurs, ainsi que les différents groupes du système. Les deux fichiers doivent être construits dans le répertoire */mnt/ELISE/etc*. Le fichier *passwd* se présente sous cette forme :

```
#passwd file for ELISE by GF
root::0:0:Super User:/:/bin/ash
```

Pour permettre l'accès au Linux embarqué, lors de la première connexion, il n'est pas possible d'inscrire un mot de passe crypté pour l'utilisateur ROOT. Au premier démarrage du système, il faudra introduire un mot de passe pour l'administrateur à l'aide de l'utilitaire *passwd*. Le fichier group se présente de la manière suivante :

```
#group file for ELISE by GF
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tty:x:4:
tape:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
```

Il est nécessaire d'y inscrire des groupes utiles pour certaines applications : bin, sys, kmem, tty, etc. Si le système n'utilise pas *TinyLogin*, les fichiers *passwd* et *group* ne sont pas nécessaires.

Le fichier profile

Le fichier profile est consulté par plusieurs applications Linux pour obtenir la valeur de certaines constantes, comme les chemins des commandes par exemple.

```
#Debut/root/.profile for ELISE by GF

#DON'T change this file. Just add a file name etc/profile.local with
#your configuration's choice

PS1='elise# '
PS2='elise# '
PATH=/bin:/usr/bin:/sbin:/usr/sbin

export PS1 PS2 PATH

#test if user add his own config file
test -e /etc/profile.local && . /etc/profile.local
#Fin
```

Le fichier utmp

Le fichier *utmp* doit être présent dans le répertoire */mnt/ELISE/var/run* pour que l'utilitaire *login* fonctionne parfaitement.

```
touch /mnt/ELISE/var/run/utmp
```

Il contient une trace de chaque connexion. Ce fichier est créé sans contenu. Il faut être attentif au fait qu'a chaque connexion sont rajoutées, dans *utmp*, des entrées. Si le fichier n'est pas effacé à intervalles réguliers, il risque de prendre une place considérable. C'est pourquoi, Il faut ajouter au fichier */mnt/ELISE/etc/init.d/rcS* la ligne suivante, chargée de supprimer le fichier à chaque démarrage du système.

```
touch /var/run/utmp
```

6.3.9 Paramètres de démarrage

Généralités

Ce chapitre aborde le dernier point essentiel, afin d'obtenir un Linux totalement indépendant. Il s'agit de l'amorçage du système qui, sur une architecture x86, n'est pas une mince affaire. Lilo étant devenu un standard pour ce type de machine, il fut naturellement utilisé. En outre, le noyau Linux possède la faculté de se décompresser et de s'exécuter. Ce point est également abordé dans les lignes qui suivent.

Configuration de Lilo pour les systèmes à disque dur ou les émulants

La construction du système oblige d'utiliser Lilo, car il emploie un disque dur. Lilo est copié en partie dans le MBR du disque dur. Une deuxième tranche du code est placée sur le disque car le MBR est d'une taille réduite.

Lilo doit être installé sur la machine hôte, si ce n'est pas le cas, il convient de le faire immédiatement, les sources étant disponibles sur le Cd-Rom. Il n'est pas nécessaire d'installer le chargeur d'amorce sur la machine cible. Néanmoins, cette solution oblige d'user d'une machine hôte à chaque changement de noyau. En effet, la deuxième partie de Lilo, copiée sur la mémoire de masse, permet de trouver le kernel sans que le système de fichier ne soit disponible. Ainsi, une modification, même minime de la taille de celui-ci, nécessite l'édition d'un nouveau code Lilo.

La première manipulation consiste à copier le code Lilo standard dans le répertoire */mnt/ELISE/boot*. Ce programme dénommé *boot.b* est disponible soit sur la machine hôte dans */boot* soit sur le Cd-Rom dans le répertoire */lilo/boot*.

```
cp  /boot/boot.b  /mnt/ELISE/boot
```

Puis, il faut éditer un fichier texte dénommé *lilo.conf* dans le répertoire */mnt/ELISE/etc*. Ce fichier contient les paramètres nécessaires à l'installation du chargeur d'amorce. Ce fichier doit contenir le code suivant :

```
#lilo configuration file for ELISE by GF
#say to lilo who is the boot media
boot=/dev/hdc
disk=/dev/hdc
bios=0x80
#give the name of the second part of lilo
install=/boot/boot.b
#give where must be install the map file
map=/boot/map
#give the delay to stay in the lilo menu
```

```
delay = 10
#give where is the kernel code
image=/boot/bzImage
#give where is the root media
root=/dev/hda1
#give a label in case of use of multiple OS
label=Linux
```

Si le disque n'est pas sur *hdc*, il convient de modifier le script. Le champs *map* indique à Lilo ou écrire le fichier contenant les informations pour trouver le noyau sur le média de stockage. Il est maintenant possible de lancer le programme Lilo comme ceci :

```
lilo  –C  etc/lilo.conf  –r  /mnt/ELISE
```

Cette ligne de commande lance l'application Lilo de la machine hôte, en opérant un *chroot* sur la cible. Il s'agit d'utiliser l'application du système hôte, en utilisant la cible comme disque principal.

Le disque dur peut dés lors être branché sur la machine cible. Il devrait, si tout a été appliqué scrupuleusement démarrer sans aucun problème.

Remarque : Le Cd-Rom contient une version de Lilo modifiée, nommée *Lilo-mtd*. Elle a été implémentée pour servir de gestionnaire de démarrage sur des système possédant une mémoire de stockage de type Flash. Il n'a malheureusement pas pu être testé au cours de ce projet.

Configuration de démarrage pour un système sans Lilo

Toutes les architectures hardware n'ont pas l'obligation d'utiliser un chargeur d'amorçage comme Lilo. En effet, le noyau compressé possède du code pour permettre sa décompression et son chargement. La disquette est l'exemple parfait d'un média de stockage ne nécessitant pas l'emploi d'un chargeur d'amorce.

Copie du noyau sur le média de stockage
Dans le cas d'une disquette, le noyau doit être copié à partir du secteur zéro car il s'agit du secteur visité en premier par le BIOS. La commande suivante copie le kernel sur la disquette :

```
dd if=NOYAU of=/dev/fd0 bs=1k
```

NOYAU doit être remplacé par le chemin exact du noyau fraîchement compilé. La console renvoie des informations sur la place occupée par le kernel. :

```
353+1 records in
353+1 records out
```

Dans cet exemple, le noyau occupe 353 enregistrements complets, plus 1 partiel, ce qui signifie que le noyau occupe les 354 premiers blocs de la disquette. Cette valeur sera nommée *BlocsNoyau*. Elle est utilisée dans les lignes qui suivent.

Il faut indiquer au noyau que le périphérique racine doit être la disquette et que le kernel doit être chargé en lecture/écriture. A l'aide de l'utilitaire *rdev* :

```
rdev /dev/fd0 /dev/fd0
rdev -R /dev/fd0 0
```

Mise en place du mot disque mémoire

Le *mot disque mémoire* situé dans l'image du noyau permet de spécifier où se trouve le système racine, ainsi que d'autres options. Le mot peut être lu et modifié avec la commande `rdev`, et sa valeur s'interprète de la manière suivante :

- bits 0-10 : Décalage jusqu'au début du disque mémoire, en blocs de 1024 octets
- bits 11-13 : Inutilisé
- bit 14 : Drapeau indiquant s'il faut charger un disque mémoire
- bit 15 : Drapeau pour faire une pause avant de charger le système racine

Si le bit 15 est mis à 1, le noyau demandera au moment du démarrage de changer la disquette dans le lecteur. C'est nécessaire si le système possède deux disques. Avec un disque unique, le système racine compressé sera placé juste après le noyau, et donc le décalage sera le premier bloc libre (qui doit être au même endroit que `BlocsNoyau`). Le bit 14 sera mis à 1, et le bit 15 à 0. Par exemple, il faut imaginer un système à disque unique dont le système racine doit commencer au bloc 253 (valeur décimale). Le mot disque mémoire devrait valoir 253 (toujours en décimal) avec le bit 14 à 1 et le bit 15 à 0. Pour calculer sa valeur il faut simplement additionner les valeurs décimales. 253 + (2^14) = 253 + 16384 = 16637.

Après avoir bien calculé la valeur du mot disque mémoire, il faut l'écrire avec *rdev -r*. *Attention à utiliser la valeur* décimale.

```
rdev -r /dev/fd0 VALEUR_DECIMALE_CALCULEE
```

Transfert du système racine
La dernière étape concerne le transfert du système racine *initrd.gz*. Le système racine étant placé sur le même disque que le noyau, if faut le transférer avec *dd* et son option *seek*, qui indique combien de blocs sauter (pour ne pas écraser le kernel):

```
dd if=initrd.gz of=/dev/fd0 bs=1k seek=BlocsNoyau
```

La disquette peut maintenant être testée.

6.3.10 Activation du mode console par terminal série

Généralités

Le mode console par terminal série permet d'accéder au système embarqué à l'aide d'un terminal par le biais du port série.

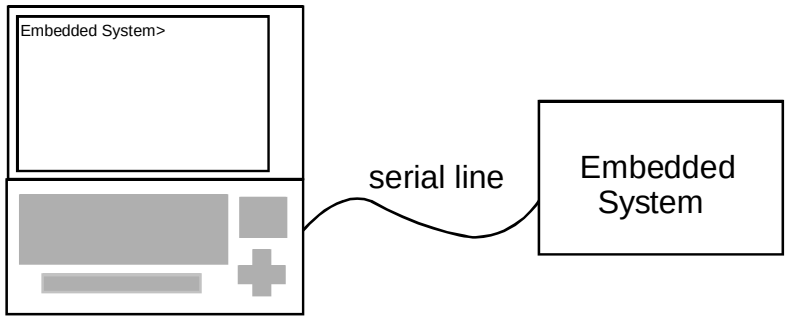


Figure 18 : Console série

Cette option peut-être nécessaire pour l'administration d'un système ne possédant pas d'affichage ou possédant un affichage réduit.

Options du noyau nécessaires

Dans les options de compilation, il est possible de distinguer deux niveaux pour l'affichage en mode console. Le premier consiste à permettre l'emploi du port série. Le deuxième ajoute à cela, l'envoi des messages du noyau sur le port série. Si cette dernière est choisie, le programme *init* ne permet pas l'affichage sur la console VGA. Il s'agit là d'une particularité de la fonction *init* de *BusyBox*, qui détecte l'émission des messages du noyau vers un port série.

Figure 19 : Options de compilation pour la console série

L'options pour activer l'envoi des messages du noyau sur le port série est : *Support for console on serial port.*

Modification des fichiers de configuration

Il faut ajouter une ligne au fichier */mnt/ELISE/etc/inittab* pour indiquer qu'il faut activer une console sur le port série, *ttySx*.

```
::sysinit:/etc/init.d/rcS

::respawn:/sbin/getty 9600 ttyS0
tty0::respawn:/bin/login
```

De plus il est nécessaire d'ajouter le fichier *securetty* dans le répertoire */mnt/ELISE/etc/* pour permettre la connexion en tant qu'administrateur ROOT au port série.

```
#securetty file for ELISE by GF
tty0
tty1
ttyS0
```

Terminal hardware

Il existe des terminaux série dédiés à cette unique fonction. Cependant, actuellement, un ordinateur portable peut remplir ce rôle. Equipé d'un programme de terminal comme

HyperTerminal ou *TeraTerm* un Pc portable peut faire office de terminal. Il devra être connecté au système embarqué à l'aide d'un simple câble *NullModem*.

6.3.11 Configuration Ethernet

Généralités

L'accès au réseau suppose que le système embarqué possède un contrôleur Ethernet. Si c'est le cas, il est faut le configurer pour pouvoir y accéder sous Linux. Cela implique l'ajout de fichiers de configuration, la présence de certains utilitaires de *BusyBox* et la compilation d'options spécifiques du noyau.

Options du noyau obligatoires

Afin de pouvoir utiliser le réseau dans sa forme minimale mais suffisante pour permettre d'exécuter un serveur *http*. Il faut choisir les options suivantes :

Figure 20 : Options de configuration pour l'accès au réseau

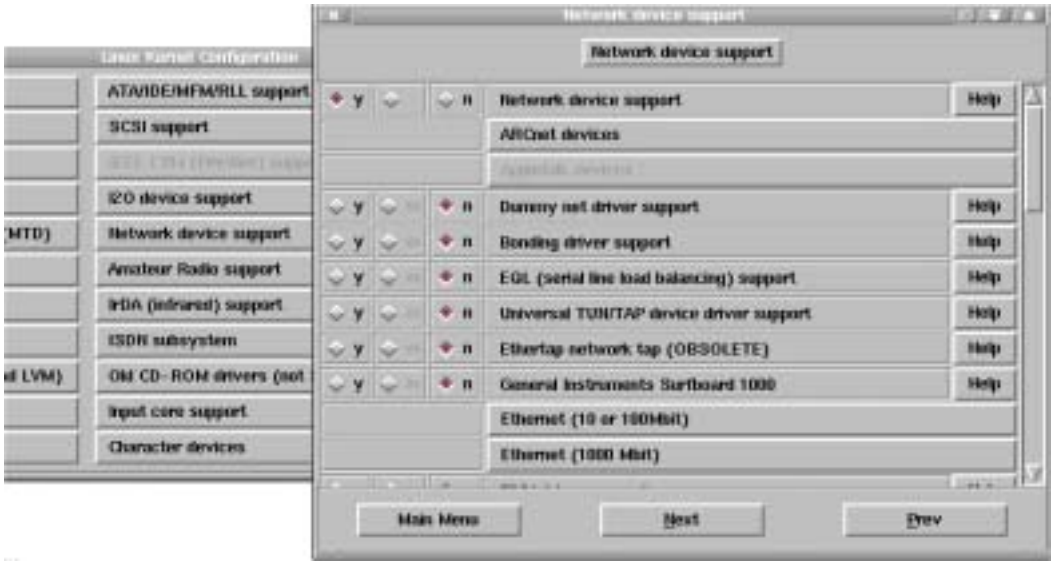


Figure 21 : Options de configuration pour l'accès au réseau

La carte réseau doit également être sélectionnée.

Figure 22 : Options de compilation de la carte Ethernet

Utilisation d'un module chargeable

Si le contrôleur Ethernet ne figure pas dans la liste des périphériques supportés par le noyau il faut recourir au module chargeable. Pour cela, il faut en premier lieu obtenir le driver du contrôleur sous forme du code source. Le code source doit préalablement être compilé avec comme option le chemin des sources du noyau Linux du système en construction. Ces changements doivent avoir lieu dans le fichier *Makefile* accompagnant les sources. Par exemple, dans le cas d'un driver Ethernet, la ligne suivante a du être modifiée en y indiquant le chemin exact des sources */usr/src/linux2.4.4/linux/include*. En omettant ces modifications, l'exécution du pilote risque d'entraîner un blocage du système.

```
CC= gcc
```

```
CFLAGS  = -DMODULE -D__KERNEL__ -I/usr/src/linux2.4.4/linux/include -
Wall -Wstrict-prototypes -O6 -c
MFLAGS  = -DMODVERSIONS
```

Les modules doivent être placés dans un nouveau répertoire nommé */mnt/ELISE/lib/modules*.

Le noyau utilisé doit prendre en charge les pilotes de périphériques chargeables. Il faut donc que les options de configurations suivantes soient validées :

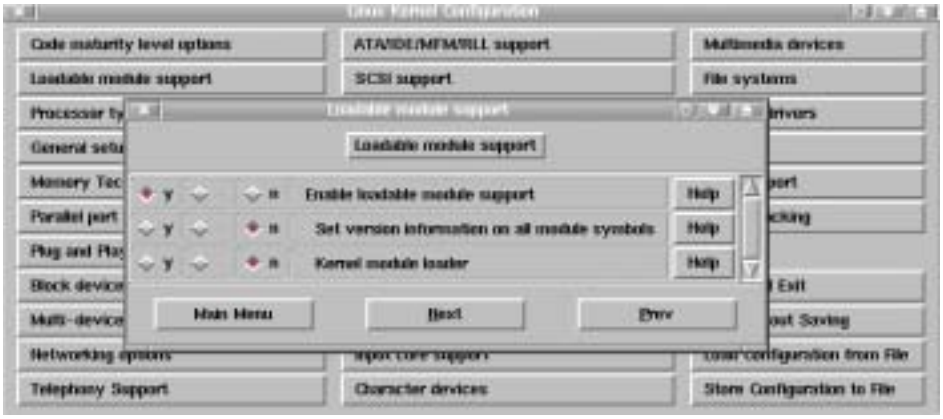


Figure 23 : Options de compilation pour les Modules

Il ne reste plus qu'a modifié le fichier de configuration */mnt/ELISE/etc/init.d/rcS*. Le driver est chargé grâce à l'utilitaire *insamod*. A l'aide d'un éditeur de texte il faut ajouter la ligne suivante :

```
insmod  PathOfMyModule
```

Remarque : Il est possible d'indiquer au noyau de charger un module à l'aide d'un autre utilitaire : *modprob*. De plus, ils peuvent être activés automatiquement. Ces options n'ont pas été abordées dans ce rapport, car elles sont plus gourmande en ressources.

Commandes Linux requises

Pour utiliser Linux avec un réseau Ethernet, il est nécessaire de posséder plusieurs utilitaires fournis par *BusyBox*. Comme décrit dans le chapitre *Installation des commandes minimales* page 78, la sélection des commandes désirées se fait en éditant le fichier *Config*. En supprimant la mise en commentaire des utilitaires, ils sont ensuite compilés.

Les utilitaires nécessaire sont :

- *hostname* : Cet utilitaire attribue à l'ordinateur un nom pour qu'il soit identifiable sur un réseau

- *ifconfig* : Il permet de configurer l'interface réseau: netmask, gatway…
- *route*: Cette application sert à paramétrer le routage.
- *ping* : Ce programme permet d'envoyer et de recevoir des requêtes au travers du réseau.

Scripts de configurations

Afin de faciliter l'administration du système, deux fichiers de configuration sont à ajouter dans le répertoire */mnt/ELISE/etc/init.d*. De plus, il faut modifier le fichier de configuration */mnt/ELISE/etc/init.d/rcS* afin que l'initialisation du système puisse être lancée.

Il faut modifier le fichier *rcS* en y ajoutant les lignes suivantes :

```
/bin/mount -n -o remount,rw /

#mount all dev locate in fstab
/bin/mount -a

#NETWORK option for rcS

#Setting loop interface
echo "Setting up interface lo: "
/sbin/ifconfig lo up 127.0.0.1

#Setting hostname
echo "Setting up hostname: "
/bin/hostname linux

#run ethernet script
/etc/init.d/./rc.net
```

Le fichier script *rc.net* doit être créé dans le répertoire */mnt/ELISE/etc/initd*. Celui-ci aurait pu être intégré au script *rcS*. Cependant par soucis de clarté, il a été édité séparément.

```
#!/bin/sh
#rc.net file for ELISE by GF
# config: /etc/network.d/interface.[devname]
```

```
# global functions

PATH=/bin:/sbin:/usr/bin:/usr/sbin

echo "Setting up ethernet interface"

for i in /etc/init.d/interface.*
do
  if [ -r "$i" ]; then
       echo -n "Setting up interface $INTERFACE: "
       /sbin/ifconfig lo 127.0.0.1 # setup loopback device

       /sbin/route add -net 127.0.0.0 netmask 255.0.0.0 lo

       /sbin/ifconfig $INTERFACE $IPADDRESS netmask $NETMASK broadcast
$BROADCAST up

       # add gateway for external gateway
       /sbin/route add default gw ${GATEWAY} netmask 0.0.0.0 metric 1

  fi
done
```

Il faut encore ajouter le script dénommé *interface.eth0* dans le répertoire /mnt/ELISE/etc/rcS. Ce script initialise la carte réseau présente *sur eth0* pour l'accès au réseau à l'aide de divers utilitaires Linux :

```
#etc/init.d/interface.eth0 for ELISE by GF

# network device name
INTERFACE="eth0"

# interface settings

# IP address
IPADDRESS="153.109.5.134"

# netmask
```

```
NETMASK="255.255.255.000"

# broadcast address
BROADCAST="153.109.5.255"

# gateway
GATEWAY="153.109.5.1"
```

Les valeurs, entrées pour les champs BROADCAST, GATWAY, NETMASK, IPADDRESS, … doivent être adaptées avec un contenu spécifique au réseau utilisé.

Script pour la configuration du DNS

Certaines applications ont besoins des fonctionnalité DNS. Par exemple, un navigateur Internet doit pouvoir accéder à un serveur de DNS pour faire correspondre une adresse littérale à une adresse IP. La configuration du DNS se fait très facilement sous Linux. En effet, lorsqu'un programme requiert l'emploi d'un DNS, le noyau se réfère au fichier *resolv.conf* contenu dans le répertoire */etc* . Il faut donc éditer un fichier texte nommé *resolv.conf* dans le répertoire */mnt/ELISE/etc*.

```
#resolv.conf file for ELISE by GF
#DNS configuration

nameserver 153.109.10.11
nameserver 153.109.10.17
search eiv.ch
```

Autre script nécessaire

Certaines applications réseau ont besoin du fichier *protocols* dans le répertoire */etc*. Celui-ci contient la liste des différents protocoles en correspondance avec leur port par défaut. Il peut être copié depuis la machine hôte ou depuis le Cd-Rom.

```
cp /etc/protocols /mnt/ELISE/etc
```

Le fichier *protocols* est visible en annexe *Fichier protocols* page 199.

6.3.12 Installation du serveur http BOA

Compilation et installation

Les sources du programme BOA dans sa version 0.94 sont disponible sur le Cd-Rom. Après décompression dans un répertoire de la machine hôte comme */usr/src/boa/*, il faut éditer le fichier *Makefile* dans le cas d'une compilation avec la librairie *ucLibc*, en ajoutant la ligne suivante :

```
CC=/usr/i386-linux-uclibc-2.4.4/usr/bin/i386-uclibc-gcc
```

Une compilation à l'aide de la librairie standard ne demande aucune modification. La compilation peut être lancée avec la commande :

```
make
```

L'application compilée doit être copiée vers la cible :

```
cp /usr/src/boa/boa-0.94.12pre1/src/mnt/ELISE/bin
```

Fichiers nécessaires

Le serveur http demande quelques fichiers de configuration. Il faut en premier lieu, copier le fichier */usr/src/boa/ boa-0.94.12pre1/debian /boa.conf* vers la cible dans /*mnt/ELISE/etc.*

```
cp /usr/src/boa/boa-0.94.12pre1/debian/boa.conf /mnt/ELISE/etc
```

Le fichier *boa.conf* contient les paramètres du serveur http. Il est nécessaire d'ajouter le fichier *mime.types*. Celui-ci est disponible soit sur le système hôte soit sur le Cd-Rom. De plus, la création de répertoire est obligatoire. Ces deux fichiers sont visibles en annexe *Fichiers de configuration de Boa* page 200. Ainsi, il faut ajouter le répertoire */mnt/ELISE/var/log/boa* qui contient un journal des accès au serveur. Ce fichier doit être effacé régulièrement car il devient vite important en taille. Et enfin, le répertoire */mnt/ELISE/var/www* doit être créé. Celui-ci contient les pages, au format html, vues depuis Internet. Il convient d'attribuer des droits suffisant à ce répertoire. Dans le cas contraire son accès est interdit et le visiteur ne trouvera rien.

Pour que le serveur soit lancé au démarrage du système, il reste à ajouter une ligne au script de démarrage */mnt/ELISE/etc/init.d/rc.net*.

```
  fi
done

echo -n "Setting up BOA web server: "
/bin/./boa
```

6.3.13 Compression des exécutables à l'aide de BZEXE

La compression des exécutables est réalisable à l'aide du script BZEXE. Il utilise l'utilitaire *bzip2*. Celui-ci n'étant pas disponible dans *BusyBox*, il a fallut modifier le script pour inclure *gzip*. De plus, d'autres adaptations ont eu lieu afin de garantir la concordance avec les commandes de *BusyBox*. (Tout en gardant une compatibilité avec un Linux conventionnel) Une application compressée à l'aide de BZEXE est totalement indépendante. En effet, un code permettant la décompression est ajouté avant l'application compressée.

```
!/bin/sh
skip=22
tmpfile=`mktemp /tmp/bz$$.XXXXXX` || {
        echo 'cannot create a temporary file' >&2
        exit 1
}
if /usr/bin/tail -n +$skip $0 | "/bin"/gzip -d > $tmpfile; then
  /bin/chmod 700 $tmpfile
  prog="`echo $0 | /bin/sed 's|^.*/||'`"
  if /bin/ln $tmpfile "/tmp/$prog" 2>/dev/null; then
    trap '/bin/rm -f $tmpfile "/tmp/$prog"; exit $res' 0
    (/bin/sleep 5; /bin/rm -f $tmpfile "/tmp/$prog") 2>/dev/null &
    /tmp/"$prog" ${1+"$@"}; res=$?
  else
    trap '/bin/rm -f $tmpfile; exit $res' 0
    (/bin/sleep 5; /bin/rm -f $tmpfile) 2>/dev/null &
    $tmpfile ${1+"$@"}; res=$?
  fi
else
  echo Cannot decompress $0; exit 1
fi; exit $res
```

Pour utiliser BZEXE, il suffit de :

```
./bzexe MyProg
```

Lors du lancement du programme, la décompression se déroule dans un fichier temporaire etpar conséquent, occupe de la place sur le média de stockage. Il est possible d'indiquer d'opérer l'opération en mémoire RAM.

Le script modifié est disponible sur le Cd-Rom et visible en annexe *Script BZEXE* page 206.

6.3.14 Affichage à l'aide d'un LCD à caractères

Application propriétaire

Le port série et le port parallèle peuvent servir à interfacer facilement un écran LCD à caractères. La solution utilisant le port parallèle demande moins de logique externe. C'est pourquoi, elle est présentée dans les pages qui suivent. L'écran choisi répond aux caractéristiques suivantes :

- 16 caractères par ligne, 2 lignes
- contrôleur Hitachi HD44780 dont les caractéristiques détaillées sont visibles en annexe *LCD HD44780* page 184

Le contrôleur est connecté sur le port parallèle. Les niveaux logiques étant identiques, il n'est pas nécessaire d'utiliser de la logique externe. L'interfaçage physique est identique au schéma ci-dessous.

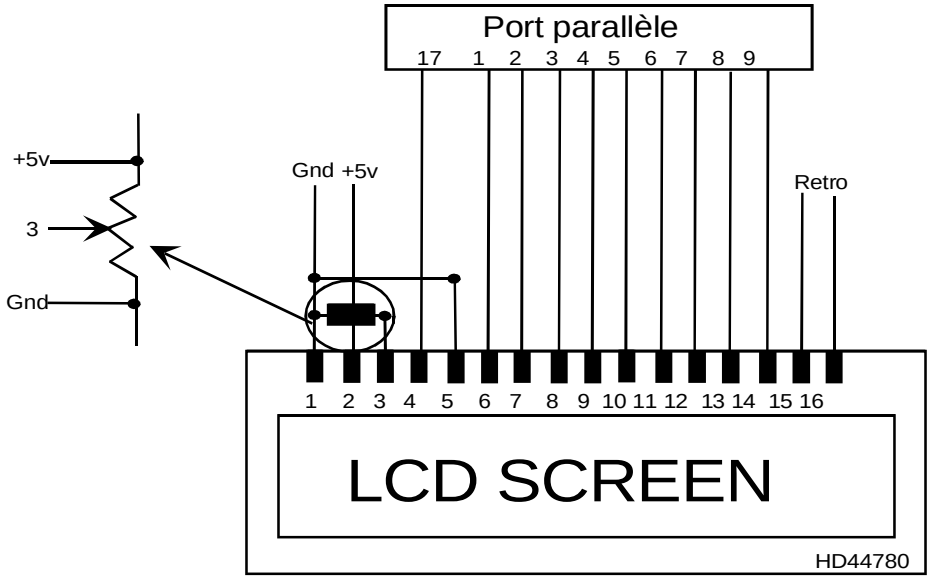


Figure 24 : Schéma LCD-port parallèle

Le brochage des afficheurs basés sur un HD44780 est toujours standardisé, comme suit :

- 1 – Ground
- 2 – Vcc
- 3 – Vo
- 4 – Sélection de registre
- 5 – Lecture/écriture
- 6– Activation du signal
- à 14 Entrée des données
- 15 Ground du rétro-éclairage
- 16 Vcc du rétro-éclairage

Les broches 1,2 et 7 à 14 sont simple d'utilisation. La broche 3 permet le contrôle du contraste de l'affichage selon le schéma. Le signal Sélection du registre (4) permet de choisir entre ce qui s'appelle le mode de commandes et le mode données. Selon l'état de cette broche, il est possible de contrôler l'afficheur (effacement, positionnement du curseur, etc.) ou simplement d'envoyer des caractères destinés à être affichés. Le signal lecture/écriture permet, comme son nom l'indique, de choisir entre la réception des informations et l'envoi de données au HF44780. Dans cet exemple, cette broche est directement reliée à la masse car il est seulement nécessaire d'écrire sur le composant. Le signal envoyé sur la broche 6 peut être comparé au fonctionnement de la touche ENTER d'un clavier. En effet, pour afficher un caractère, il faut envoyer les données sur les broches 7 à 14, puis activer brièvement la broche 6. Et ce n'est qu'a ce moment que le caractère apparaît sur l'écran. Il en va de même pour le mode commande, une commande ne sera validée que par ce signal.

Le programme écrit en C est en annexe *Programme pour LCD HD44780* page 185.

LCDproc

Le programme client/serveur *LCDproc* n'a pas été testé. Cependant, il a été compilé avec les librairies standards sans aucun problème. Le paquet *LCDproc* contient une

documentation complète quant à la programmation d'application et à l'interfaçage physique d'un écran LCD.

6.3.15 Affichage en mode texte VGA

L'affichage texte en mode VGA n'est pas obligatoire pour faire fonctionner Linux. Il nécessite la compilation d'une, voir deux options du noyau.

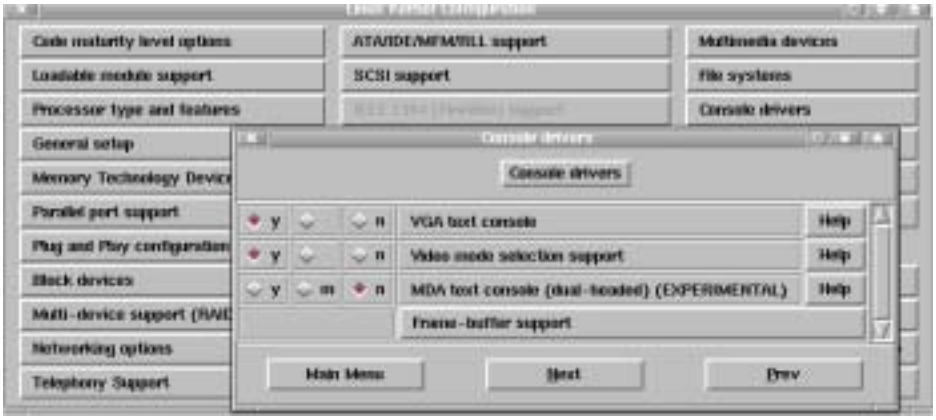


Figure 25 : Affichage texte en mode VGA

VGA text console est obligatoire pour l'affichage. *Video mode selection support* rajoute la possibilité de sélectionner plusieurs mode d'affichage. Cette option n'est pas indispensable.

6.3.16 Affichage graphique

Nano-X - Microwindows

Après avoir décompressé le paquet *Microwindows* version 0.88 dans le répertoire */usr/src/microwindows*, il faut éditer le fichier *Config* afin d'y inscrire la librairie utilisée pour la compilation et diverses autres options. Pour utiliser un outils plus facile d'emploi, il faut lancer :

```
cd usr/src/microwindows/src
make xconfig
```

La fenêtre qui apparaît, facilite la configuration de *Nano-X*. Elle permet de choisir les polices, le type d'affichage (FrameBuffer, SVGALib, X11), les commandes utilisées (clavier, souris, touchscreen).

Figure 26 : Options de Nano-X

FrameBuffer

En plus de la sélection dans la configuration de Nano-X, il faut que l'option *FrameBuffer* soit installée sur le noyau de la machine embarquée. Pour une utilisation confortable, le *FrameBuffer* requiert environ 1 [Mo]de mémoire RAM supplémentaire.

La première option de compilation se situe dans l'activation de :

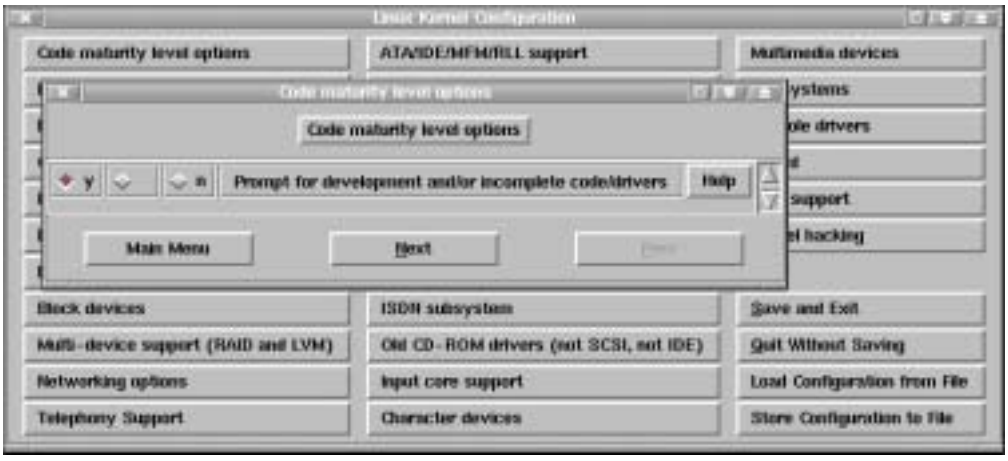


Figure 27 : Option de compilation pour FrameBuffer 1

Ensuite il faut définir plus précisément le type de driver. L'utilisation du pilote VESA permet d'employer le FrameBuffer sur toutes les cartes répondant à la norme VESA 2.0 .

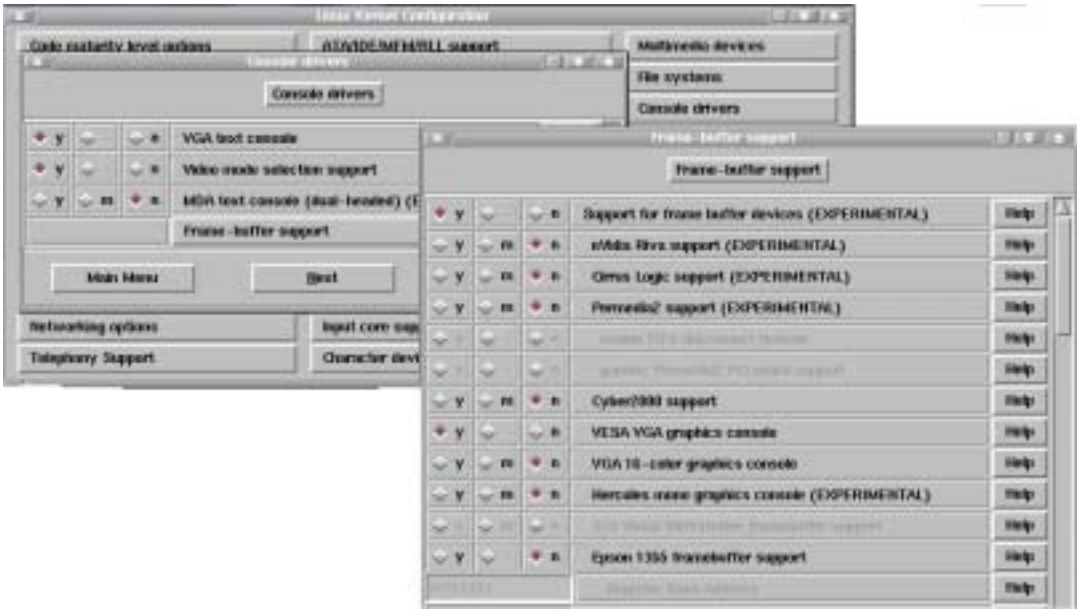


Figure 28 : Options de compilation pour FrameBuffer 2

Il faut, en plus, ajouter le périphérique propre au *FrameBuffe*r sous */mnt/ELISE/dev* :

```
mknod –m 660 /mnt/ELISE/dev c 29 0
```

Et enfin, il faut passer au noyau le paramètre pour indiquer la résolution du FrameBuffer. LA modification est faite dans le fichier de configuration de Lilo, */etc/lilo.conf.*

```
vga=0x301
```

SVGALib

La compilation de Nano-X pour une utilisation avec la SVGAlib, impose que cette librairie soit installée sur la machine hôte et sur la cible. La bibliothèque est disponible sur le Cd-Rom. Il suffit de copié, après la compilation de Nano-X, la librairie qu'il utilise. Pour cela *ldd* peut être très utile :

```
ldd /usr/src/microwindows/src/bin/nano-X
```

Ce qui donne :

```
```

```
libvga.so.1 => /usr/lib/libvga.so.1 0x40023000
```

Il faut donc copier *libvga.so.1* dans le répertoire */mnt/ELISE/usr/lib* :

```
cp /usr/lib/libvga.so.1  /mnt/ELISE/lib
```

Remarque : Nano-X ne peut pas être compilé à l'aide de la ucLibc dans le cas de l'utilisation de la librairie SVGAlib.

Sous XWindow

La possibilité de compiler Nano-X pour être utilisé sous XWindow est très utile pour la conception des applications. Il n'est pas nécessaire de les tester sur le système embarqué.

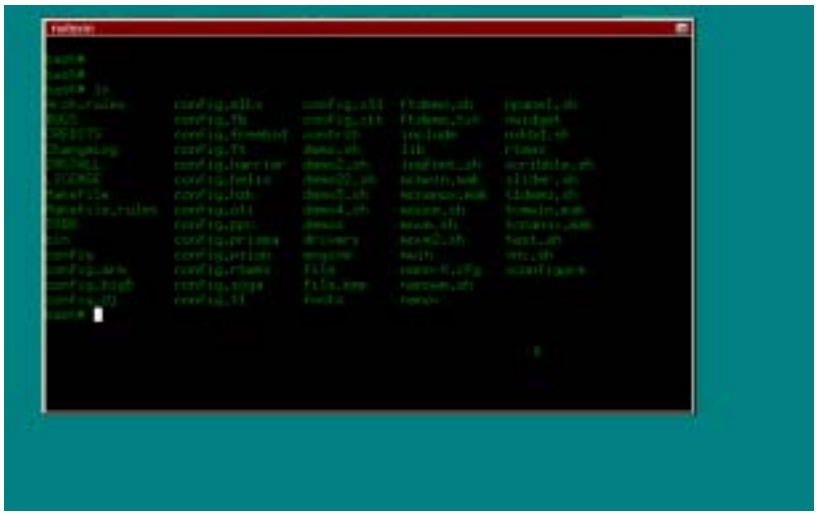


Figure 29 : Nano-X

Utilisation

Nano-X est lancé comme une simple application. Il faut qu'il soit exécuté en tâche de fond :

```
./nano-x &&
```

Il est livré avec une multitude de programmes de tests tels qu'un gestionnaire de fenêtres, un terminal en mode graphique, une horloge, etc.

PicoGui

Pour utiliser PicoGui version 0.20, le paquet présent sur le Cd-Rom doit être décompressé dans le répertoire */usr/src/picogui*. Celui-ci utilise également la librairie graphique SVGAlib. L'utilitaire *automake* est employé par PicoGui pour générer les fichiers de *makefile*. Sur la machine hôte, il n'a pas encore été possible de compiler PicoGui sans erreur. Cependant sur une autre machine présentant les mêmes caractéristiques, PicoGui se compile parfaitement. Le problème n'a pu encore être découvert mais semble provenir de l'utilitaire *automake*.

6.4 Conclusion

En suivant scrupuleusement toutes les indications de ce chapitre, il est possible de concevoir un Linux minimal : création du système de fichier, modification et compilation du noyau, installation des librairies, choix des commandes minimales et créations des scripts de démarrage. De plus, l'Os peut recevoir plusieurs options supplémentaires telles qu'interface réseau ou affichage graphique.

Une configuration ne doit pas se cantonner aux indications données ici. Chaque système embarqué étant unique, il faut adapter les conseils d'optimisation, les modifier pour sa propre machine.

7 Tests du système optimisé

7.1 Mémoire RAM minimale

Afin de tester le besoin du système en mémoire RAM, un calcul, sur la base de la consommation de chaque élément, n'a pas été suffisant. Les besoins exacts du système en mémoire RAM sont difficiles à déterminer avec précision. Le Linux optimisé minimal (sans affichage graphique) fonctionne sur un système x86 équipé de 3[Mo] de mémoire RAM. Néanmoins, cette valeur est un peu trop petite. En effet, l'exécution de quelques applications montre rapidement les limites du système. Le confort d'utilisation est atteint avec 4 [Mo] de RAM.

7.2 Place en mémoire de masse

Ce rapport a prouvé qu'il était ardu de calculer avec exactitude la taille minimale d'un Linux pour système embarqué. Celle-ci dépend directement de l'application finale de la machine cible. Néanmoins, il a été possible de faire fonctionner sur un x86, un Os Linux optimisé dont la taille totale n'excédait pas 1,2[Mo], noyau inclus. (Le mode *initrd* n'a pas été utilisé.)

7.3 Le système d'exploitation en générale

Il n'est pas facile de vérifier un système d'exploitation. Les modifications, effectuées sur le noyau, ne sont pas dangereuses pour la stabilité de celui-ci. Les applications comme *BusyBox* ou *TinyLogin* fonctionnent parfaitement. De plus, le serveur web BOA est resté en fonction durant une semaine (accès fréquent au serveur).

Il n'a pas été possible d'implémenter un banc de test spécifique. Il s'agit de la meilleure méthode de test possible.

7.4 La procédure de démarrage

Le chargement du système d'exploitation à l'aide de Lilo est possible sur les différentes machines cibles testées. Certains problèmes sont apparus sur d'anciennes plate-formes. Ce point est expliqué plus en détail dans le chapitre suivant *Synthèse* page 108. De plus, l'utilisation du noyau pour le démarrage a aussi été testée par la création d'un exemple sur disquette. Celle-ci est disponible en dernière page du rapport.

7.5 Architecture x86

Le système d'exploitation a été testé, en mode console, sur diverses architectures compatibles x86. Il fonctionne parfaitement sur :

- Pc de bureau AMD Athelon 1,4 [Ghz] 512 [Mo] RAM

- Pc de bureau Pentium III 700 [Mhz] 128 [Mo] RAM
- Pc industriel Pentium 233[Mhz] MMX 64 [Mo] RAM
- Pc de bureau Pentium 133 [Mhz] 16 [Mo] RAM
- Pc portable AST 80486 DX2 66[Mhz] 8[Mo] RAM
- Pc portable Compaq 80386 SX 25[Mhz] et coprocesseur arithmétique 20 [Mo] RAM
- Pc de bureau 80386 SX 16 [Mhz] 4 [Mo] RAM

Remarque : Les instructions de certains programmes ne sont pas reconnues avec le 80386 SX 16. Il s'agit en fait d'une erreur au niveau du compilateur gcc 2.95. L'utilisation d'une version antérieure fait disparaître cet inconvénient.

7.6 Module chargeables

La carte Ethernet installée dans la machine cible ne dispose pas d'un pilote intégré au noyau. Il a fallut utiliser un module à l'aide de la commande *insmod*. L'ensemble fonctionne parfaitement puisque le serveur *http*, utilisant le réseau, est accessible depuis Internet. L'image de la page Internet est visible sur la page suivante.

7.7 La connexion par terminal série

La connexion par terminal série ne pose aucun problème particulier. La figure ci-dessous montre un terminal réalisé avec le programme HyperTerminal sous Windows2000.

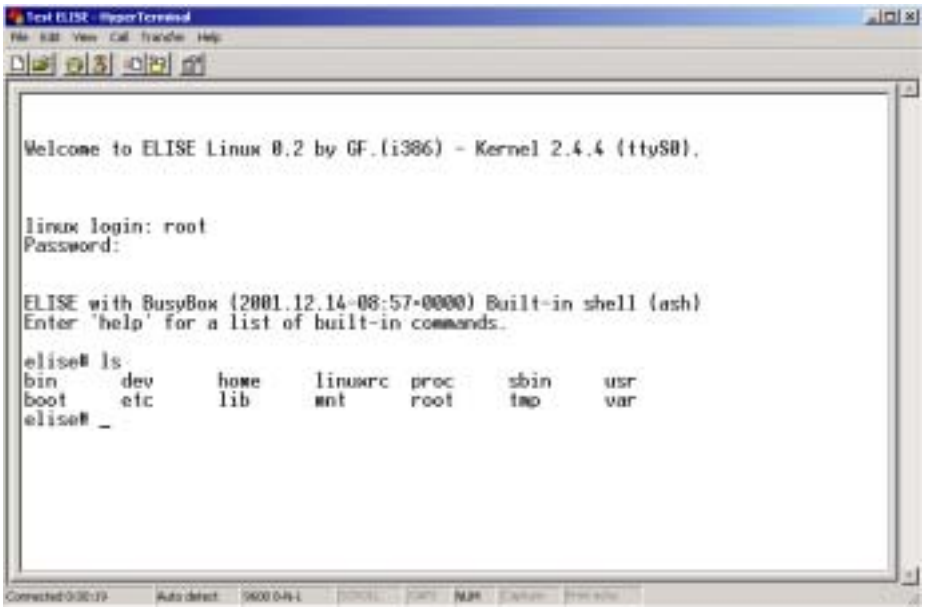


Figure 30 : Terminal série sous Windows 2000

7.8 L'affichage

L'affichage sur console VGA n'a posé aucun problème. La bonne marche de Nano-X a été confirmée par l'emploi des programmes de démonstration, plus précisément le gestionnaire de fenêtres et le terminal en mode graphique visible sur la figure 27 .

L'application propriétaire fonctionne parfaitement comme le montre la figure suivante :

Figure 31 : LCD HD44680 sur port parallèle

7.9 Le réseau Ethernet

La configuration du réseau a été validée dans un premier temps à l'aide de l'utilitaire *ping*. En effet, une machine tiers était capable de vérifier la présence du système embarqué. Une deuxième confirmation a eu lieu avec la mise en place du serveur http BOA. En effet, une page web était visible depuis Internet :

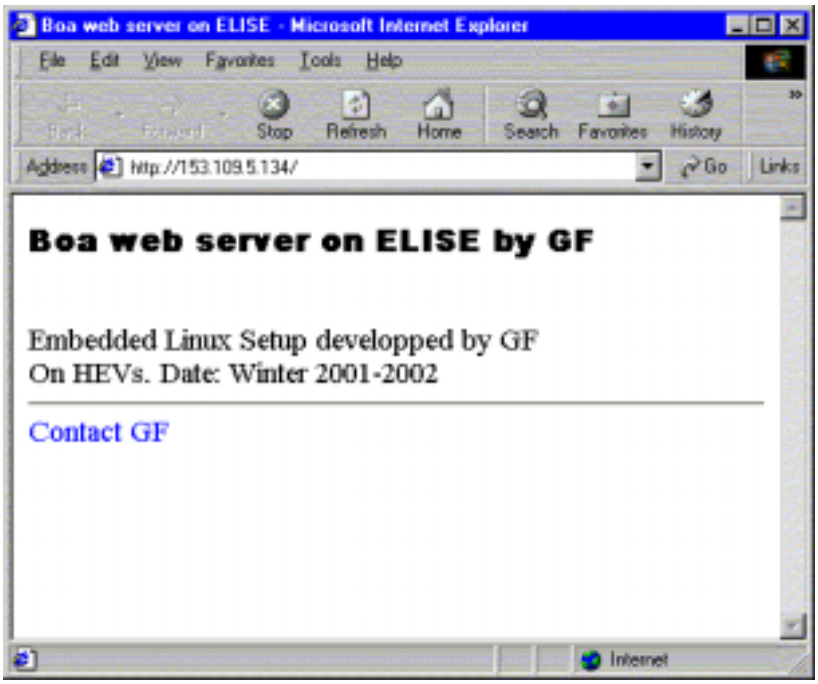


Figure 32 : BOA webserver

8 Synthèse

8.1 Problèmes rencontrés

8.1.1 Compilation à l'aide de la ucLibc

Problème

Bien des programmes ne peuvent pas être compilés à l'aide de cette librairie minimale. L'exécution du *make* après modification du *makefile* se solde souvent par un message d'erreur.

Solution

La première solution consiste à lier l'application de manière statique à la librairie standard. Cette solution est à envisager seulement dans le cas ou il faut absolument garder la ucLibc. Sinon, il est conseillé de changer de librairie pour adopter la bibliothèque standard.

Il est également possible de modifier le code source du programme. Si une erreur survient lors de la compilation, *gcc* indique ou se trouve la faute. Il s'agit, dans la plupart des cas, d'une fonction qui manque ou alors qui est simplifiée.

8.1.2 Compilation du noyau

Problème

La compilation du noyau est une tâche qui demande énormément de ressource. *Gcc* utilise activement la mémoire RAM et les ressources du processeur. Sur une machine qui fonctionnait sans problèmes apparents sous Linux, ne parvenait pas à compiler le noyau. Celui-ci stoppait sur une erreur de type *signal 11*.

Solution

Après maintes réflexions, il s'est avéré que le problème se situait au niveau de la mémoire RAM. Le remplacement des barrettes de mémoire du Pc incriminé résolut l'obstacle. En règle générale, si un problème de compilation du kernel est répétitif, il est probable que la mémoire ait un défaut.

8.1.3 Problème du noyau 2.4.0

Problème

Une première version du système embarqué a été réalisée avec le noyau 2.4.0. Néanmoins, au moment de la configuration de la carte réseau, la détection par les pilotes du kernel se soldait par un message d'erreur du type : *could not reserve irq0*. (Ceci sur certaines machines seulement) L'emploi d'un module chargeable ne posait aucun problème.

Solution

Le noyau 2.4.0 fut remplacé par la version 2.4.4. Celle-ci corrige certains *bug* de la version 2.4.0.

8.1.4 Utilisation des modules chargeables

Problème

L'utilisation des modules chargeables se soldait par un blocage total du système et l'affichage d'un message d'erreur du noyau.

Solution

Il faut être attentif au fait que les modules doivent être compilés en utilisant les sources exactes du noyau qui doit les faire fonctionner. Pour cela, il faut modifier le *makefile* du code source du module, afin de lui indiquer le bon chemin.

8.1.5 Géométrie du disque dur et Lilo

Problème

Lors de tests sur d'anciennes machines, Lilo n'a pas pu lancer, dans tous les cas, Linux. Un message d'erreur du type *L 04 04 04 04 04* ….s'affichait de manière répétitive. Ce message d'erreur indique que la partie de Lilo située dans le MBR n'a pas pu trouver la seconde partie se trouvant sur le disque dur.

Solution

La machine de développement est une machine récente. Son BIOS est capable de détecter toutes les sortes de disques durs. Les vieux Pc sont plus limités. Certains ne reconnaissent que quelques types de géométries prés-enregistrés. Au moment de la configuration de Lilo sur la machine hôte, tout fonctionne à merveille. Néanmoins, au moment du démarrage, Lilo utilise le BIOS pour connaître cette fameuse géométrie. Les indications s'y trouvant étant erronées, Lilo n'est pas en mesure de localiser sa deuxième moitié. Pour résoudre ce problème, il suffit d'ajouter dans le fichier */mnt/ELIDE/etc/lilo.conf* des lignes concernent la géométrie du disque. Il faut se référer aux entrées prés-définies dans le BIOS.

```
sectors= NumberOfSectors
heads=NumberOfHeads
cylinders=NumberOfCylinders
```

8.1.6 Notion de temps en programmation

Problème

Lors de la programmation de l'application de démonstration pour l'écran LCD, la fonction C *ussleep()* censée attendre 1 [us] ne fonctionne pas. En effet, le temps d'attente est aléatoire.

Solution

Linux n'étant pas temps réel, il est difficile de pouvoir garantir un *timing* de petit ordre dans un programme. Il faut pour cela utiliser un patch qui permet de rendre Linux temps réel.

8.1.7 Compilation de PicoGui

Problème

La compilation de PicoGui, sur la machine hôte, n'a pas encore pu être effectuée sans erreur. Néanmoins, elle s'est déroulée sans problème sur un système possédant la même distribution. L'erreur provient de l'outil *automake*. Malgré plusieurs tentatives d'installation de l'utilitaire, le problème ne disparaît pas.

Solution

La solution n'est malheureusement pas encore connue !

8.1.8 Incompatibilité avec la carte Pc-104 de la Hevs

Problème

Le système Linux n'est malheureusement pas exécutable sur la carte Pc-104 développée au sein de la *Hevs*. Il semble qu'il ait un problème au moment de l'accès au contrôleur du disque dur. Celui-ci est pourtant entièrement supporté par Linux.

Solution

La pièce maîtresse de la carte n'étant plus produite, il n'a pas été jugé nécessaire de dépenser du temps dans la recherche du problème.

8.2 Evolution du projet

8.2.1 Stockage en Flash

La construction du système Linux embarqué s'est faite autour d'un disque dur classique. Il est évident qu'il ne s'agit pas d'une bonne solution pour un système embarqué. L'issue la plus réaliste se trouve dans l'emploi de mémoire Flash. Bien que l'ensemble des outils pour la gestion de tels périphériques soient sur le Cd-Rom, la non-disponibilité de celui-ci a empêché tout test. Ce problème mérite d'être exploré en profondeur.

8.2.2 Les autres architectures

L'optimisation de Linux, pour les systèmes embarqués, est un sujet très vaste. Dans le cadre de ce rapport, il n'a pas été possible d'effectuer un portage sur une machine autre qu'un x86, ceci pour une question de temps. Comme expliqué précédemment, il existe de nombreux portage de Linux pour une plate-formes autres que x86. Ainsi, il serait intéressant de tester Linux pour une autre architecture telle que les processeurs ARM, par exemple.

8.2.3 Linux temps réel

Pour une utilisation industrielle, il est intéressant de pouvoir bénéficier des avantages d'un Os temps réel. Les patchs ont été trouvés mais n'ont pas pu être testés. C'est pourquoi, il serait intéressant d'expérimenter *RT Linux*.

8.2.4 Solutions de mises à jour

Tous les systèmes embarqués ne sont pas figés. Ils doivent pouvoir évoluer en fonction des désirs du client. Il n'a pas été possible, dans ce projet d'examiner des solutions de mises à jour de la machine Linux. Il serait intéressant d'explorer diverses possibilités hardware et software.

9 Conclusion

La plus grande partie des objectifs inscrits dans le cahier des charges ont été atteint. Ce rapport a permis, en premier lieu, de décomposer Linux en différentes pièces de puzzle. Ce fractionnement a permis de faciliter l'analyse des options d'optimisation applicables sur les parties qui sont :

- Le noyau
- Le système de fichier
- Les librairies
- Les utilitaires de base
- Les fichiers de configuration
- Les applications spécifiques

En outre, il a été possible de donner quelques conseils quant au choix du hardware d'un système embarqué devant accueillir Linux. Il convient de préciser, pour la dernière fois, que les optimisations ont été faites dans le sens d'un gain de place en mémoire de stockage et en mémoire RAM, et dans le sens d'une adaptation aux contraintes des systèmes embarqués.

Dans un deuxième temps, l'analyse a aboutit sur la mise en pratique des options théoriques pour donner naissance à un système d'exploitation fonctionnel destiné au compatible x86. Ce mode d'emploi pourrait être intitulé : *HOW-TO Embedded Linux*

Sion, le 24 janvier 2002 Gaspoz Frédéric

10 Annexes

10.1 BIBLIOGRAPHIE ET LIENS UTILES 114
10.1.1 Généralités 114
10.2 COMPLÉMENTS SUR LINUX 117
10.2.1 L'arborescence FHS des répertoires 118
10.2.2 Systèmes de fichiers supportés par Linux 123
10.2.3 Fichiers périphériques 125
10.3 AFFICHEUR LCD 181
10.3.1 Le port parallèle 182
10.3.2 LCD HD44780 184
10.3.3 Programme pour LCD HD44780 185
10.4 CODES SOURCES ET SCRIPTS UTILISÉS 194
10.4.1 Script de création des fichiers périphériques 195
10.4.2 Fichier protocols 199
10.4.3 Fichier de configuration de BOA 200
10.4.4 Script BZEXE 206
10.4.5 Réduction des Librairies C 211
10.5 DESCRIPTION DES COMMANDES BUSYBOX 231
10.6 DESCRIPTION DES COMMANDES TINYLOGIN 270
10.7 OPTIONS DE COMPILATION DU NOYAU 272
10.8 LICENCE GPL 315
10.8.1 Preamble 315
10.8.2 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION 316
10.9 TABLE DES MATIÈRES DÉTAILLÉE 320
10.10 GLOSSAIRE 326
10.11 CD-ROM ET DISQUETTE 334

10.1 Bibliographie et liens utiles

Analyser et optimiser un noyau Linux a nécessité une documentation gigantesque. Plusieurs sources d'informations ont été utilisées. Ce chapitre présente les principales sources qui ont permis la rédaction de ce texte.

10.1.1 Généralités

Embedded Linux par **John Lombardo** édition New Riders

Le Noyau Linux par **Bovet et Cesati** édition O'Reilly

Le système Linux édition O'Reilly

Linux Kernal Internals de **Tigran Aivazian**

Concrete Architecture of the Linux Kernel **de Ivan Bowman et Saheem Sidiqi**

Guide d'installation de Linux de **Christian Casteyde**

Bible du cracker par Falcon

PC programmation système par

Kid Operating System par **Thomas Petazzoni**

Informations générales

http://www.linuxdevices.com

http://www.AllLinuxDevices.com

http://www.Embedded-linux.org

http://www.linuxdocs.org/

http://www.freenix.org

Linux France Magazine édition Diamond Editions

Login magazine édition Diamond Editions

Système fini

PC-104 http://www.pc104.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

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 http://www.uclinux.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

Software

Codes sources du noyau Linux	http://www.kernel.org
Système de fichiers JFFS	http://www.developer.axis.com
Driver MTD pour mémoire Flash	http://www.linux-mtd.infradead.org
BusyBox	http://busybox.lineo.com
Tinylogin	http://tinylogin.lineo.com
Nano-x et MicroWindows	http://www.microwindows.org
PicoGui	http://www.picogui.org
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/
Lilo	http://freshmeat.net/projects/lilo/
BlueZ Bluetooht	http://bluez.sourceforge.net/
Thttpd	http://www.acme.com/software/thttpd/
E3	http://www.sax.de/~adlibit/
Gcc	http://www.gnu.org/software/gcc/gcc.html.

Librairies

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

10.2 Compléments sur Linux

10.2.1 L'arborescence FHS des répertoires

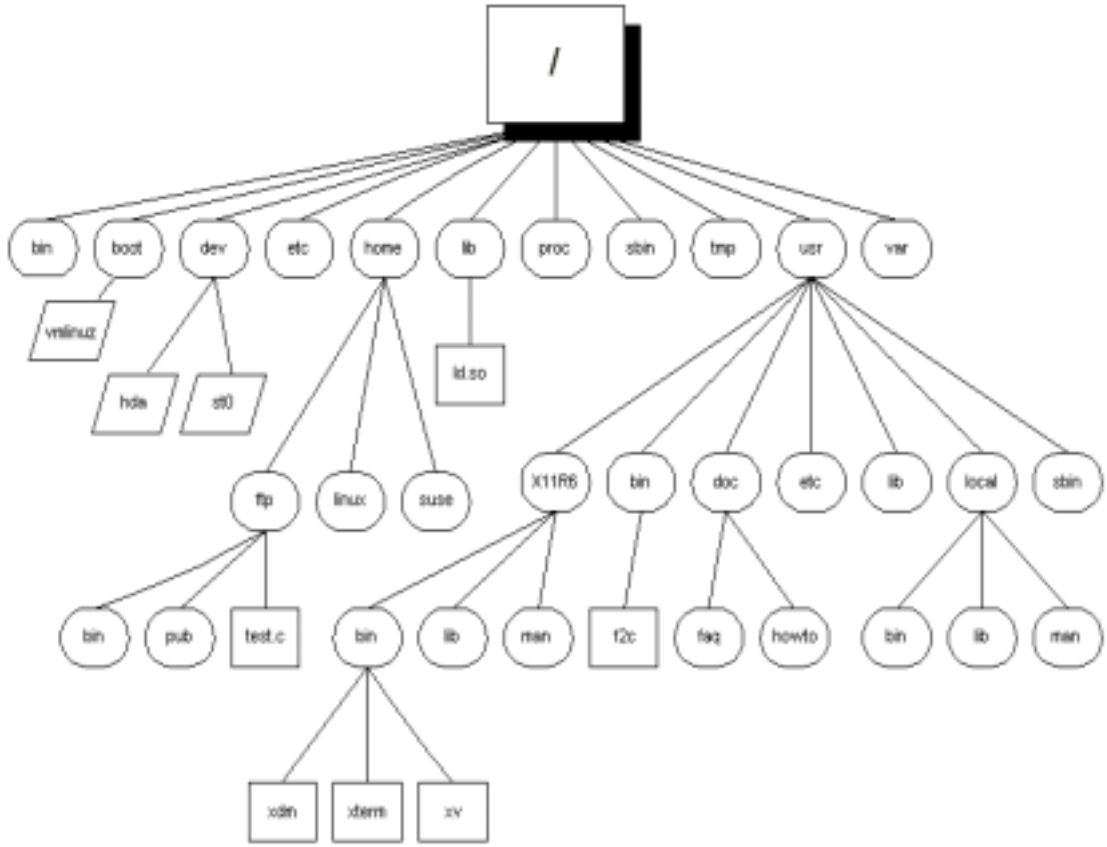


Figure 33: Arborescence des répertoires

Répertoire	Signification
/	Répertoire racine. Point de départ de toute la hiérarchie du système de fichiers. Le système de fichiers contenant ce répertoire est monté automatiquement par le noyau pendant l'amorçage du système. Ce système de fichiers est appelé système de fichiers racine (« root » en anglais).
/boot/	Répertoire contenant le noyau de Linux et ses informations de symboles.
/boot/vmlinuz	Noyau comprimé de Linux. Sur certains systèmes, le noyau est encore placé dans le répertoire racine de la partition root.
/dev/	Répertoire contenant tous les fichiers spéciaux permettant d'accéder aux périphériques. Sous Linux, la plupart des

Répertoire	Signification
	périphériques sont accessibles au travers de fichiers spéciaux, grâce auxquels l'envoi et la réception des données vers les périphériques peuvent être réalisés de manière uniforme. Il existe un tel fichier pour chaque périphérique, et ils sont tous placés dans ce répertoire. Les distributions installent souvent dans ce répertoire les fichiers spéciaux pour la plupart des périphériques, même s'ils ne sont pas physiquement présents dans la machine. Dans ce cas, les opérations sur les fichiers spéciaux des périphériques non installés seront tout simplement refusées par le noyau. Une autre possibilité est d'utiliser un système de fichiers virtuels, géré directement par le noyau. Le répertoire `/dev/` ne contient dans ce cas que les fichiers spéciaux des périphériques pour lesquels le noyau dispose d'un gestionnaire intégré ou chargé dynamiquement. Quelle que soit la technique utilisée, ce répertoire doit être impérativement placé dans le système de fichiers racine.
/sbin/	Répertoire contenant les commandes systèmes nécessaires à l'amorçage et réservées à l'administrateur. Ce répertoire doit être impérativement placé dans le système de fichiers racine. En général, seul l'administrateur utilise ces commandes.
/bin/	Répertoire contenant les commandes systèmes générales nécessaires à l'amorçage. Ce répertoire doit être impérativement placé dans le système de fichiers racine. Tous les utilisateurs peuvent utiliser les commandes de ce répertoire.
/lib/	Répertoire contenant les bibliothèques partagées (« DLL » en anglais, pour « Dynamic Link Library ») utilisées par les commandes du système des répertoires `/bin/` et `/sbin/`. Ce répertoire doit être impérativement placé dans le système de fichiers racine.
/lib/modules/	Répertoire contenant les modules du noyau. Ce répertoire contient les modules additionnels du noyau. Ces modules sont des composants logiciels du noyau, mais ne sont pas chargés immédiatement pendant l'amorçage. Ils peuvent en revanche être chargés et déchargés dynamiquement, lorsque le système est en fonctionnement. Il est fortement recommandé que ce répertoire soit placé dans le système de fichiers racine.
/etc/	Répertoire contenant tous les fichiers de configuration du système. Ce répertoire doit être impérativement placé dans le système de fichiers racine.
/etc/X11/	Dans certaines distributions, les fichiers de configuration de

Répertoire	Signification
	XWindow sont placés directement dans le répertoire `/etc/X11/`.
/etc/opt/	Répertoire contenant les fichiers de configuration des applications.
/tmp/	Répertoire permettant de stocker des données temporaires. En général, `/tmp/` ne contient que des données très éphémères. Il est préférable d'utiliser le répertoire `/var/tmp/`. En effet, le répertoire `/tmp/` ne dispose pas nécessairement de beaucoup de place disponible.
/usr/	Répertoire contenant les fichiers du système partageables en réseau et en lecture seule.
/usr/bin/	Répertoire contenant la plupart des commandes des utilisateurs.
/usr/sbin/	Répertoire contenant les commandes systèmes non nécessaires à l'amorçage. Ces commandes ne sont normalement utilisées que par l'administrateur système.
/usr/lib/	Répertoire contenant les bibliothèques partagées de tous les programmes de `/usr/bin/` et `/usr/sbin/` et les bibliothèques statiques pour la création de programmes.
/usr/include/	Répertoire contenant les fichiers d'en-têtes du système pour le compilateur C/C++. Les fichiers de ce répertoire sont utilisés pour réaliser des programmes dans les langages de programmation C et C++.
/usr/X11R6/	Répertoire contenant X11R6 et ses applications. Ce répertoire contient des sous-répertoires `bin/`, `lib/` et `include/`, où se trouvent les exécutables de XWindow, les bibliothèques et les fichiers d'en-têtes pour créer des programmes pour XWindow en C et C++.
/usr/src/	Répertoire contenant les fichiers sources du noyau et des applications de la distribution. Normalement, ce répertoire ne doit contenir que le code source des applications dépendantes de la distribution que vous utilisez.
/usr/src/linux/	Sources du noyau de Linux. Il est vivement recommandé de conserver les sources du noyau de Linux sur son disque, afin de pouvoir changer la configuration du système à tout moment.
/usr/local/	Répertoire contenant les programmes d'extension du système indépendants de la distribution. Ce n'est pas le

Répertoire	Signification
	répertoire d'installation des applications. « local » ne signifie pas ici que les programmes qui se trouvent dans ce répertoire ne peuvent pas être partagés sur le réseau, mais plutôt que ce sont des extensions du système qu'on ne trouve donc que localement sur un site donné. Ce sont donc les extensions qui ne font pas partie de la distribution de Linux utilisée, et qui doivent être conservées lors des mises à jour ultérieures de cette distribution. Ce répertoire contient les sous-répertoires `bin/`, `lib/`, `include/` et `src/`, qui ont la même signification que les répertoires du même nom de `/usr/`, à ceci près qu'ils ne concernent que les extensions locales du système, donc indépendantes de la distribution.
/var/	Répertoire contenant toutes les données variables du système. Ce répertoire contient les données variables qui ne pouvaient pas être placées dans le répertoire `/usr/`, puisque celui-ci est normalement accessible en lecture seule.
/var/tmp/	Répertoire contenant les fichiers temporaires. Il est préférable d'utiliser ce répertoire plutôt que le répertoire `/tmp/`.
/var/opt/	Répertoire contenant les données variables des applications.
/var/log/	Répertoire contenant les traces de tous les messages système. C'est dans ce répertoire que l'on peut consulter les messages d'erreurs du système et des applications.
/var/spool/	Répertoire contenant les données en attente de traitement. Les travaux d'impression en cours, les mails et les fax en attente d'émission, les travaux programmés en attente d'exécution sont tous stockés dans ce répertoire.
/var/locks/	Répertoire contenant les verrous sur les ressources système. Certaines ressources ne peuvent être utilisées que par une seule application (par exemple, un modem). Les applications qui utilisent de telles ressources le signalent en créant un fichier de verrou dans ce répertoire.
/var/cache/	Répertoire contenant les données de résultats intermédiaires des applications. Les applications qui doivent stocker des résultats intermédiaires doivent les placer dans ce répertoire.
/opt/	Répertoire contenant les applications. C'est dans ce répertoire que les applications qui ne font pas réellement partie du système doivent être installées. Les applications graphiques devraient être installées dans ce répertoire. C'est en particulier le cas des gestionnaires de bureau. Les seules applications graphiques considérées comme faisant partie du

Répertoire	Signification
	système sont les applications de X11, qui sont donc stockées dans `/usr/X11R6/`. Il est recommandé que ce répertoire soit placé sur un système de fichiers en lecture seule, et que les applications utilisent le répertoire `/var/opt/` pour travailler.
/home/	Répertoire contenant les répertoires personnels des utilisateurs.
/root/	Répertoire contenant le répertoire personnel de l'administrateur. Il est donc recommandé que le répertoire personnel de l'administrateur soit placé en dehors de `/home/` pour éviter qu'un problème sur le système de fichiers des utilisateurs ne l'empêche de travailler. Toutefois, il est important que l'administrateur puisse travailler même si les répertoires `/root/` et `/home/root/` ne sont pas présents. Dans ce cas, son répertoire personnel devra être le répertoire racine.
/mnt/	Répertoire réservé au montage des systèmes de fichiers non-permanents (CD-ROM, disquettes, etc.). Ce répertoire peut contenir plusieurs sous-répertoires pour chaque périphérique amovible, afin de permettre d'en monter plusieurs simultanément. Notez qu'il est assez courant de disposer de liens symboliques dans la racine référençant les principaux systèmes de fichiers, afin d'en simplifier l'accès. Par exemple, il est courant d'avoir un répertoire `/floppy/` référençant le lecteur de disquette et un répertoire `/cdrom/` référençant le lecteur de CD-ROM.
/proc/	Répertoire contenant le pseudo système de fichiers du noyau. Ce pseudo système de fichiers contient des fichiers permettant d'accéder aux informations sur le matériel, la configuration du noyau et sur les processus en cours d'exécution.

10.2.2 Systèmes de fichiers supportés par Linux

Système de fichiers	Identification	Description
Etendu version 2	*ext2*	Le plus courant sous Linux
/proc	*proc*	Système de fichiers virtuel, pour infos noyau
ADFS	*adfs*	Système de fichiers Acorn
Amiga FFS	*affs*	Amiga
Apple Mac FS	*hfs*	Apple Macintosh
Coherent	*coherent*	Celui de l'OS Coherent
DOS-FAT	*msdos*	Celui de Ms-Dos pour accéder à ces fichiers
HPFS	*hpfs*	Partitions HPFS en lecture
ISO 9660	*iso9660*	Destiné à la plupart des cd-rom
JFFS	*jffs*	Système de fichiers pour système embarqué à disque Flash, utilise la journalisation
Joliet	*iso9660*	Extention de l'ISO supportant les fichiers Unicode
Minix	*minix*	Le tout premier, celui de Minix
NFS network file system	*NFS*	Pour l'accès aux fichiers distants via le réseau
Novell	*ncpfs*	Réservé au serveur Novell
NT	*ntfs*	Permet d'accéder aux fichiers Windows NT
QNX4	*qnx4*	Système QNX4
ReiserFS	*reiserfs*	Système de fichiers avec la journalisation
ROM	*romfs*	Un mini système de fichiers en lecture seule utilisé pour les disques RAM
SMB	*smbfs*	Accès au fichiers d'un serveur Windows Workgroups ou Windows NT
System V	*sysv*	Accès aux fichiers de différents System V
UFS	*ufs*	Système de fichiers Sun ou BSD
UMSDOS	*umsdos*	Permet d'installer Linux sur une partition Ms-Dos
VFAT	*vfat*	Windows 95798 VFAT32 géré
Xenix	*xenix*	Celui de l'OS Xenix
XFS	*xfs*	Système de fichiers développé par SGI pour les gros serveurs

Remarque : Cette liste n’est pas exhaustive et peu bien entendu être incomplète. La communauté Linux regorgeant de programmeurs et de développeurs motivés, son contenu varie très rapidement.

10.2.3 Fichiers périphériques

LINUX ALLOCATED DEVICES
Maintained by H. Peter Anvin

Last revised: 3 June 2001

This list is the Linux Device List, the official registry of allocated device numbers and /dev directory nodes for the Linux operating system.

The latest version of this list is available from http://www.lanana.org/docs/device-list/ or ftp://ftp.kernel.org/pub/linux/docs/device-list/. This version may be newer than the one distributed with the Linux kernel.

The LaTeX version of this document is no longer maintained.

This document is included by reference into the Filesystem Hierarchy Standard (FHS). The FHS is available from http://www.pathname.com/fhs/.

Allocations marked (68k/Amiga) apply to Linux/68k on the Amiga platform only. Allocations marked (68k/Atari) apply to Linux/68k on the Atari platform only.

The symbol {2.6} means the allocation is obsolete and scheduled for removal once kernel version 2.6 (or equivalent) is released.

This document is in the public domain. The author requests, however, that semantically altered versions are not distributed without permission of the author, assuming the author can be contacted without an unreasonable effort.

In particular, please don't sent patches for this list to Linus, at least not without contacting me first.

I do not have any information about these devices beyond what appears on this list. Any such information requests will be deleted without reply.

**** DEVICE DRIVERS AUTHORS PLEASE READ THIS ****

THE DEVICE REGISTRY IS OFFICIALLY FROZEN FOR LINUS TORVALDS' KERNEL
TREE. At Linus' request, no more allocations will be made official for Linus' kernel tree; the 3 June 2001 version of this list is the official final version of this registry. At Alan Cox' request, however, the registry will continue to be maintained for the -ac

series of kernels, and registrations will be accepted.

To have a major number allocated, or a minor number in situations where that applies (e.g. busmice), please contact me with the appropriate device information. Also, if you have additional information regarding any of the devices listed below, or if I have made a mistake, I would greatly appreciate a note.

I do, however, make a few requests about the nature of your report. This is necessary for me to be able to keep this list up to date and correct in a timely manner. First of all, *please* send it to the correct address... . I receive hundreds of email messages a day, so mail sent to other addresses may very well get lost in the avalanche. Please put in a descriptive subject, so I can find your mail again should I need to. Too many people send me email saying just "device number request" in the subject.

Second, please include a description of the device *in the same format as this list*. The reason for this is that it is the only way I have found to ensure I have all the requisite information to publish your device and avoid conflicts.

Third, please don't assume that the distributed version of the list is up to date. Due to the number of registrations I have to maintain it in "batch mode", so there is likely additional registrations that haven't been listed yet.

Finally, sometimes I have to play "namespace police." Please don't be offended. I often get submissions for /dev names that would be bound to cause conflicts down the road. I am trying to avoid getting in a situation where we would have to suffer an incompatible forward change. Therefore, please consult with me *before* you make your device names and numbers in any way public, at least to the point where it would be at all difficult to get them changed.

Your cooperation is appreciated.

0	Unnamed devices (e.g. non-device mounts)	
	0 = reserved as null device number	
1 char	Memory devices	
	1 = /dev/mem	Physical memory access
	2 = /dev/kmem	Kernel virtual memory access
	3 = /dev/null	Null device
	4 = /dev/port	I/O port access
	5 = /dev/zero	Null byte source
	6 = /dev/core	OBSOLETE - replaced by /proc/kcore
	7 = /dev/full	Returns ENOSPC on write
	8 = /dev/random	Nondeterministic random number gen.

```
                9 = /dev/urandom    Faster, less secure random number gen.
               10 = /dev/aio        Asyncronous I/O notification interface
block          RAM disk
                0 = /dev/ram0           First RAM disk
                1 = /dev/ram1           Second RAM disk
                  ...
              250 = /dev/initrd     Initial RAM disk {2.6}

              Older kernels had /dev/ramdisk (1, 1) here.
              /dev/initrd refers to a RAM disk which was preloaded
              by the boot loader; newer kernels use /dev/ram0 for
              the initrd.

2 char        Pseudo-TTY masters
                0 = /dev/ptyp0      First PTY master
                1 = /dev/ptyp1      Second PTY master
                  ...
              255 = /dev/ptyef      256th PTY master

              Pseudo-tty's are named as follows:
              * Masters are "pty", slaves are "tty";
              * the fourth letter is one of pqrstuvwxyzabcde indicating
                the 1st through 16th series of 16 pseudo-ttys each, and
              * the fifth letter is one of 0123456789abcdef indicating
                the position within the series.

              These are the old-style (BSD) PTY devices; Unix98
              devices are on major 128 and above and use the PTY
              master multiplex (/dev/ptmx) to acquire a PTY on
              demand.

block         Floppy disks
                0 = /dev/fd0        Controller 0, drive 0, autodetect
                1 = /dev/fd1        Controller 0, drive 1, autodetect
                2 = /dev/fd2        Controller 0, drive 2, autodetect
                3 = /dev/fd3        Controller 0, drive 3, autodetect
              128 = /dev/fd4        Controller 1, drive 0, autodetect
              129 = /dev/fd5        Controller 1, drive 1, autodetect
              130 = /dev/fd6        Controller 1, drive 2, autodetect
              131 = /dev/fd7        Controller 1, drive 3, autodetect

              To specify format, add to the autodetect device number:
                0 = /dev/fd?        Autodetect format
                4 = /dev/fd?d360    5.25"  360K in a 360K  drive(1)
               20 = /dev/fd?h360    5.25"  360K in a 1200K drive(1)
               48 = /dev/fd?h410    5.25"  410K in a 1200K drive
               64 = /dev/fd?h420    5.25"  420K in a 1200K drive
               24 = /dev/fd?h720    5.25"  720K in a 1200K drive
               80 = /dev/fd?h880    5.25"  880K in a 1200K drive(1)
                8 = /dev/fd?h1200   5.25" 1200K in a 1200K drive(1)
```

```
 40 = /dev/fd?h1440   5.25" 1440K in a 1200K drive(1)
 56 = /dev/fd?h1476   5.25" 1476K in a 1200K drive
 72 = /dev/fd?h1494   5.25" 1494K in a 1200K drive
 92 = /dev/fd?h1600   5.25" 1600K in a 1200K drive(1)

 12 = /dev/fd?u360    3.5"   360K Double Density(2)
 16 = /dev/fd?u720    3.5"   720K Double Density(1)
120 = /dev/fd?u800    3.5"   800K Double Density(2)
 52 = /dev/fd?u820    3.5"   820K Double Density
 68 = /dev/fd?u830    3.5"   830K Double Density
 84 = /dev/fd?u1040   3.5"  1040K Double Density(1)
 88 = /dev/fd?u1120   3.5"  1120K Double Density(1)
 28 = /dev/fd?u1440   3.5"  1440K High Density(1)
124 = /dev/fd?u1600   3.5"  1600K High Density(1)
 44 = /dev/fd?u1680   3.5"  1680K High Density(3)
 60 = /dev/fd?u1722   3.5"  1722K High Density
 76 = /dev/fd?u1743   3.5"  1743K High Density
 96 = /dev/fd?u1760   3.5"  1760K High Density
116 = /dev/fd?u1840   3.5"  1840K High Density(3)
100 = /dev/fd?u1920   3.5"  1920K High Density(1)
 32 = /dev/fd?u2880   3.5"  2880K Extra Density(1)
104 = /dev/fd?u3200   3.5"  3200K Extra Density
108 = /dev/fd?u3520   3.5"  3520K Extra Density
112 = /dev/fd?u3840   3.5"  3840K Extra Density(1)

 36 = /dev/fd?CompaQ          Compaq 2880K drive; obsolete?

(1) Autodetectable format
(2) Autodetectable format in a Double Density (720K) drive only
(3) Autodetectable format in a High Density (1440K) drive only

NOTE: The letter in the device name (d, q, h or u)
signifies the type of drive: 5.25" Double Density (d),
5.25" Quad Density (q), 5.25" High Density (h) or 3.5"
(any model, u).          The use of the capital letters D, H
and E for the 3.5" models have been deprecated, since
the drive type is insignificant for these devices.

3 char      Pseudo-TTY slaves
  0 = /dev/ttyp0         First PTY slave
  1 = /dev/ttyp1         Second PTY slave
    ...
255 = /dev/ttyef         256th PTY slave

These are the old-style (BSD) PTY devices; Unix98
devices are on major 136 and above.

 block      First MFM, RLL and IDE hard disk/CD-ROM interface
  0 = /dev/hda           Master: whole disk (or CD-ROM)
 64 = /dev/hdb           Slave: whole disk (or CD-ROM)
```

For partitions, add to the whole disk device number:

0 = /dev/hd?	Whole disk
1 = /dev/hd?1	First partition
2 = /dev/hd?2	Second partition
...	
63 = /dev/hd?63	63rd partition

For Linux/i386, partitions 1-4 are the primary partitions, and 5 and above are logical partitions. Other versions of Linux use partitioning schemes appropriate to their respective architectures.

4 char TTY devices

0 = /dev/tty0	Current virtual console
1 = /dev/tty1	First virtual console
...	
63 = /dev/tty63	63rd virtual console
64 = /dev/ttyS0	First UART serial port
...	
255 = /dev/ttyS191	192nd UART serial port

UART serial ports refer to 8250/16450/16550 series devices.

Older versions of the Linux kernel used this major number for BSD PTY devices. As of Linux 2.1.115, this is no longer supported. Use major numbers 2 and 3.

5 char Alternate TTY devices

0 = /dev/tty	Current TTY device
1 = /dev/console	System console
2 = /dev/ptmx	PTY master multiplex
64 = /dev/cua0	Callout device for ttyS0
...	
255 = /dev/cua191	Callout device for ttyS191

(5,1) is /dev/console starting with Linux 2.1.71. See the section on terminal devices for more information on /dev/console.

6 char Parallel printer devices

0 = /dev/lp0	Parallel printer on parport0
1 = /dev/lp1	Parallel printer on parport1
...	

Current Linux kernels no longer have a fixed mapping between parallel ports and I/O addresses. Instead, they are redirected through the parport multiplex layer.

7 char Virtual console capture devices

0 = /dev/vcs	Current vc text contents
1 = /dev/vcs1	tty1 text contents
...	
63 = /dev/vcs63	tty63 text contents
128 = /dev/vcsa	Current vc text/attribute contents
129 = /dev/vcsa1	tty1 text/attribute contents
...	
191 = /dev/vcsa63	tty63 text/attribute contents

NOTE: These devices permit both read and write access.

block Loopback devices

0 = /dev/loop0	First loopback device
1 = /dev/loop1	Second loopback device
...	

The loopback devices are used to mount filesystems not associated with block devices. The binding to the loopback devices is handled by mount(8) or losetup(8).

8 block SCSI disk devices (0-15)

0 = /dev/sda	First SCSI disk whole disk
16 = /dev/sdb	Second SCSI disk whole disk
32 = /dev/sdc	Third SCSI disk whole disk
...	
240 = /dev/sdp	Sixteenth SCSI disk whole disk

Partitions are handled in the same way as for IDE disks (see major number 3) except that the limit on partitions is 15.

9 char SCSI tape devices

0 = /dev/st0	First SCSI tape, mode 0
1 = /dev/st1	Second SCSI tape, mode 0
...	
32 = /dev/st0l	First SCSI tape, mode 1
33 = /dev/st1l	Second SCSI tape, mode 1
...	
64 = /dev/st0m	First SCSI tape, mode 2
65 = /dev/st1m	Second SCSI tape, mode 2
...	
96 = /dev/st0a	First SCSI tape, mode 3
97 = /dev/st1a	Second SCSI tape, mode 3
...	
128 = /dev/nst0	First SCSI tape, mode 0, no rewind
129 = /dev/nst1	Second SCSI tape, mode 0, no rewind
...	
160 = /dev/nst0l	First SCSI tape, mode 1, no rewind
161 = /dev/nst1l	Second SCSI tape, mode 1, no rewind

```
   ...
192 = /dev/nst0m      First SCSI tape, mode 2, no rewind
193 = /dev/nst1m      Second SCSI tape, mode 2, no rewind
   ...
224 = /dev/nst0a      First SCSI tape, mode 3, no rewind
225 = /dev/nst1a      Second SCSI tape, mode 3, no rewind
   ...

"No rewind" refers to the omission of the default
automatic rewind on device close.  The MTREW or MTOFFL
ioctl()'s can be used to rewind the tape regardless of
the device used to access it.

block   Metadisk (RAID) devices
  0 = /dev/md0        First metadisk group
  1 = /dev/md1        Second metadisk group
   ...

The metadisk driver is used to span a
filesystem across multiple physical disks.

10 char Non-serial mice, misc features
  0 = /dev/logibm     Logitech bus mouse
  1 = /dev/psaux      PS/2-style mouse port
  2 = /dev/inportbm   Microsoft Inport bus mouse
  3 = /dev/atibm      ATI XL bus mouse
  4 = /dev/jbm        J-mouse
  4 = /dev/amigamouse       Amiga mouse (68k/Amiga)
  5 = /dev/atarimouse Atari mouse
  6 = /dev/sunmouse   Sun mouse
  7 = /dev/amigamouse1      Second Amiga mouse
  8 = /dev/smouse     Simple serial mouse driver
  9 = /dev/pc110pad   IBM PC-110 digitizer pad
 10 = /dev/adbmouse   Apple Desktop Bus mouse
 11 = /dev/vrtpanel   Vr41xx embedded touch panel
 13 = /dev/vpcmouse   Connectix Virtual PC Mouse
 14 = /dev/touchscreen/ucb1x00  UCB 1x00 touchscreen
 15 = /dev/touchscreen/mk712    MK712 touchscreen
128 = /dev/beep            Fancy beep device
129 = /dev/modreq    Kernel module load request {2.6}
130 = /dev/watchdog  Watchdog timer port
131 = /dev/temperature     Machine internal temperature
132 = /dev/hwtrap    Hardware fault trap
133 = /dev/exttrp    External device trap
134 = /dev/apm_bios  Advanced Power Management BIOS
135 = /dev/rtc       Real Time Clock
139 = /dev/openprom  SPARC OpenBoot PROM
140 = /dev/relay8    Berkshire Products Octal relay card
141 = /dev/relay16   Berkshire Products ISO-16 relay card
142 = /dev/msr             x86 model-specific registers {2.6}
```

143 = /dev/pciconf PCI configuration space
144 = /dev/nvram Non-volatile configuration RAM
145 = /dev/hfmodem Soundcard shortwave modem control {2.6}
146 = /dev/graphics Linux/SGI graphics device
147 = /dev/opengl Linux/SGI OpenGL pipe
148 = /dev/gfx Linux/SGI graphics effects device
149 = /dev/input/mouse Linux/SGI Irix emulation mouse
150 = /dev/input/keyboard Linux/SGI Irix emulation keyboard
151 = /dev/led Front panel LEDs
153 = /dev/mergemem Memory merge device
154 = /dev/pmu Macintosh PowerBook power manager
155 = /dev/isictl MultiTech ISICom serial control
156 = /dev/lcd Front panel LCD display
157 = /dev/ac Applicom Intl Profibus card
158 = /dev/nwbutton Netwinder external button
159 = /dev/nwdebug Netwinder debug interface
160 = /dev/nwflash Netwinder flash memory
161 = /dev/userdma User-space DMA access
162 = /dev/smbus System Management Bus
163 = /dev/lik Logitech Internet Keyboard
164 = /dev/ipmo Intel Intelligent Platform Management
165 = /dev/vmmon VMWare virtual machine monitor
166 = /dev/i2o/ctl I2O configuration manager
167 = /dev/specialix_sxctl Specialix serial control
168 = /dev/tcldrv Technology Concepts serial control
169 = /dev/specialix_rioctl Specialix RIO serial control
170 = /dev/smapi IBM Thinkpad SMAPI
171 = /dev/srripc QNX4 API IPC manager
172 = /dev/usemaclone Semaphore clone device
173 = /dev/ipmikcs Intelligent Platform Management
174 = /dev/uctrl SPARCbook 3 microcontroller
175 = /dev/agpgart AGP Graphics Address Remapping Table
176 = /dev/gtrsc Gorgy Timing radio clock
177 = /dev/cbm Serial CBM bus
178 = /dev/jsflash JavaStation OS flash SIMM
179 = /dev/xsvc High-speed shared-mem/semaphore service
180 = /dev/vrbuttons Vr41xx button input device
181 = /dev/toshiba Toshiba laptop SMM support
182 = /dev/perfctr Performance-monitoring counters
183 = /dev/intel_rng Intel i8x0 random number generator
184 = /dev/cpu/microcode CPU microcode update interface
186 = /dev/atomicps Atomic shapshot of process state data
187 = /dev/irnet IrNET device
188 = /dev/smbusbios SMBus BIOS
189 = /dev/ussp_ctl User space serial port control
190 = /dev/crash Mission Critical Linux crash dump facility
191 = /dev/pcl181
192 = /dev/nas_xbus NAS xbus LCD/buttons access
193 = /dev/d7s SPARC 7-segment display
194 = /dev/zkshim Zero-Knowledge network shim control

195 = /dev/elographics/e2201 Elographics touchscreen E271-2201
198 = /dev/sexec Signed executable interface
199 = /dev/scanners/cuecat :CueCat barcode scanner
200 = /dev/net/tun TAP/TUN network device
201 = /dev/button/gulpb Transmeta GULP-B buttons
204 = /dev/video/em8300 EM8300 DVD decoder control
205 = /dev/video/em8300_mv EM8300 DVD decoder video
206 = /dev/video/em8300_ma EM8300 DVD decoder audio
207 = /dev/video/em8300_sp EM8300 DVD decoder subpicture
208 = /dev/compaq/cpqphpc Compaq PCI Hot Plug Controller
209 = /dev/compaq/cpqrid Compaq Remote Insight Driver
210 = /dev/impi/bt IMPI coprocessor block transfer
211 = /dev/impi/smic IMPI coprocessor stream interface
212 = /dev/watchdogs/0 First watchdog device
213 = /dev/watchdogs/1 Second watchdog device
214 = /dev/watchdogs/2 Third watchdog device
215 = /dev/watchdogs/3 Fourth watchdog device
216 = /dev/fujitsu/apanel Fujitsu/Siemens application panel
217 = /dev/ni/natmotn National Instruments Motion
218 = /dev/kchuid Inter-process chuid control
219 = /dev/modems/mwave MWave modem firmware upload
220 = /dev/mptctl Message passing technology (MPT) control
221 = /dev/mvista/hssdsi Montavista PICMG hot swap system driver
222 = /dev/mvista/hasi Montavista PICMG high availability
240-255 Reserved for local use

11 char Raw keyboard device
0 = /dev/kbd Raw keyboard device

The raw keyboard device is used on Linux/SPARC only.

block SCSI CD-ROM devices
0 = /dev/sr0 First SCSI CD-ROM
1 = /dev/sr1 Second SCSI CD-ROM
...

The prefix /dev/scd instead of /dev/sr has been used
as well, and might make more sense.

12 char QIC-02 tape
2 = /dev/ntpqic11 QIC-11, no rewind-on-close
3 = /dev/tpqic11 QIC-11, rewind-on-close
4 = /dev/ntpqic24 QIC-24, no rewind-on-close
5 = /dev/tpqic24 QIC-24, rewind-on-close
6 = /dev/ntpqic120 QIC-120, no rewind-on-close
7 = /dev/tpqic120 QIC-120, rewind-on-close
8 = /dev/ntpqic150 QIC-150, no rewind-on-close
9 = /dev/tpqic150 QIC-150, rewind-on-close

The device names specified are proposed -- if there

are "standard" names for these devices, please let me know.

block MSCDEX CD-ROM callback support {2.6}
 0 = /dev/dos_cd0 First MSCDEX CD-ROM
 1 = /dev/dos_cd1 Second MSCDEX CD-ROM
 ...

13 char Input core
 0 = /dev/input/js0 First joystick
 1 = /dev/input/js1 Second joystick
 ...
 32 = /dev/input/mouse0 First mouse
 33 = /dev/input/mouse1 Second mouse
 ...
 63 = /dev/input/mice Unified mouse
 64 = /dev/input/event0 First event queue
 65 = /dev/input/event1 Second event queue
 ...

Each device type has 5 bits (32 minors).

block 8-bit MFM/RLL/IDE controller
 0 = /dev/xda First XT disk whole disk
 64 = /dev/xdb Second XT disk whole disk

Partitions are handled in the same way as IDE disks
(see major number 3).

14 char Open Sound System (OSS)
 0 = /dev/mixer Mixer control
 1 = /dev/sequencer Audio sequencer
 2 = /dev/midi00 First MIDI port
 3 = /dev/dsp Digital audio
 4 = /dev/audio Sun-compatible digital audio
 6 = /dev/sndstat Sound card status information {2.6}
 7 = /dev/audioctl SPARC audio control device
 8 = /dev/sequencer2 Sequencer -- alternate device
 16 = /dev/mixer1 Second soundcard mixer control
 17 = /dev/patmgr0 Sequencer patch manager
 18 = /dev/midi01 Second MIDI port
 19 = /dev/dsp1 Second soundcard digital audio
 20 = /dev/audio1 Second soundcard Sun digital audio
 33 = /dev/patmgr1 Sequencer patch manager
 34 = /dev/midi02 Third MIDI port
 50 = /dev/midi03 Fourth MIDI port
block BIOS harddrive callback support {2.6}
 0 = /dev/dos_hda First BIOS harddrive whole disk
 64 = /dev/dos_hdb Second BIOS harddrive whole disk
 128 = /dev/dos_hdc Third BIOS harddrive whole disk
 192 = /dev/dos_hdd Fourth BIOS harddrive whole disk

Partitions are handled in the same way as IDE disks
(see major number 3).

15 char Joystick
0 = /dev/js0 First analog joystick
1 = /dev/js1 Second analog joystick
...
128 = /dev/djs0 First digital joystick
129 = /dev/djs1 Second digital joystick
...
block Sony CDU-31A/CDU-33A CD-ROM
0 = /dev/sonycd Sony CDU-31a CD-ROM

16 char Non-SCSI scanners
0 = /dev/gs4500 Genius 4500 handheld scanner
block GoldStar CD-ROM
0 = /dev/gscd GoldStar CD-ROM

17 char Chase serial card
0 = /dev/ttyH0 First Chase port
1 = /dev/ttyH1 Second Chase port
...
block Optics Storage CD-ROM
0 = /dev/optcd Optics Storage CD-ROM

18 char Chase serial card - alternate devices
0 = /dev/cuh0 Callout device for ttyH0
1 = /dev/cuh1 Callout device for ttyH1
...
block Sanyo CD-ROM
0 = /dev/sjcd Sanyo CD-ROM

19 char Cyclades serial card
0 = /dev/ttyC0 First Cyclades port
...
31 = /dev/ttyC31 32nd Cyclades port
block "Double" compressed disk
0 = /dev/double0 First compressed disk
...
7 = /dev/double7 Eighth compressed disk
128 = /dev/cdouble0 Mirror of first compressed disk
...
135 = /dev/cdouble7 Mirror of eighth compressed disk

See the Double documentation for the meaning of the
mirror devices.

20 char Cyclades serial card - alternate devices
0 = /dev/cub0 Callout device for ttyC0

...
31 = /dev/cub31 Callout device for ttyC31

block Hitachi CD-ROM (under development)
0 = /dev/hitcd Hitachi CD-ROM

21 char Generic SCSI access
0 = /dev/sg0 First generic SCSI device
1 = /dev/sg1 Second generic SCSI device
...

Most distributions name these /dev/sga, /dev/sgb...; this sets an unnecessary limit of 26 SCSI devices in the system and is counter to standard Linux device-naming practice.

block Acorn MFM hard drive interface
0 = /dev/mfma First MFM drive whole disk
64 = /dev/mfmb Second MFM drive whole disk

This device is used on the ARM-based Acorn RiscPC. Partitions are handled the same way as for IDE disks (see major number 3).

22 char Digiboard serial card
0 = /dev/ttyD0 First Digiboard port
1 = /dev/ttyD1 Second Digiboard port
...

block Second IDE hard disk/CD-ROM interface
0 = /dev/hdc Master: whole disk (or CD-ROM)
64 = /dev/hdd Slave: whole disk (or CD-ROM)

Partitions are handled the same way as for the first interface (see major number 3).

23 char Digiboard serial card - alternate devices
0 = /dev/cud0 Callout device for ttyD0
1 = /dev/cud1 Callout device for ttyD1
...

block Mitsumi proprietary CD-ROM
0 = /dev/mcd Mitsumi CD-ROM

24 char Stallion serial card
0 = /dev/ttyE0 Stallion port 0 card 0
1 = /dev/ttyE1 Stallion port 1 card 0
...
64 = /dev/ttyE64 Stallion port 0 card 1
65 = /dev/ttyE65 Stallion port 1 card 1
...
128 = /dev/ttyE128 Stallion port 0 card 2
129 = /dev/ttyE129 Stallion port 1 card 2

```
                ...
              192 = /dev/ttyE192    Stallion port 0 card 3
              193 = /dev/ttyE193    Stallion port 1 card 3
                ...
  block       Sony CDU-535 CD-ROM
               0 = /dev/cdu535      Sony CDU-535 CD-ROM

25 char       Stallion serial card - alternate devices
               0 = /dev/cue0              Callout device for ttyE0
               1 = /dev/cue1              Callout device for ttyE1
                ...
              64 = /dev/cue64       Callout device for ttyE64
              65 = /dev/cue65       Callout device for ttyE65
                ...
             128 = /dev/cue128      Callout device for ttyE128
             129 = /dev/cue129      Callout device for ttyE129
                ...
             192 = /dev/cue192      Callout device for ttyE192
             193 = /dev/cue193      Callout device for ttyE193
                 ...
  block       First Matsushita (Panasonic/SoundBlaster) CD-ROM
               0 = /dev/sbpcd0      Panasonic CD-ROM controller 0 unit 0
               1 = /dev/sbpcd1      Panasonic CD-ROM controller 0 unit 1
               2 = /dev/sbpcd2      Panasonic CD-ROM controller 0 unit 2
               3 = /dev/sbpcd3      Panasonic CD-ROM controller 0 unit 3

26 char       Quanta WinVision frame grabber {2.6}
               0 = /dev/wvisfgrab   Quanta WinVision frame grabber
  block       Second Matsushita (Panasonic/SoundBlaster) CD-ROM
               0 = /dev/sbpcd4      Panasonic CD-ROM controller 1 unit 0
               1 = /dev/sbpcd5      Panasonic CD-ROM controller 1 unit 1
               2 = /dev/sbpcd6      Panasonic CD-ROM controller 1 unit 2
               3 = /dev/sbpcd7      Panasonic CD-ROM controller 1 unit 3

27 char       QIC-117 tape
               0 = /dev/qft0        Unit 0, rewind-on-close
               1 = /dev/qft1        Unit 1, rewind-on-close
               2 = /dev/qft2        Unit 2, rewind-on-close
               3 = /dev/qft3        Unit 3, rewind-on-close
               4 = /dev/nqft0       Unit 0, no rewind-on-close
               5 = /dev/nqft1       Unit 1, no rewind-on-close
               6 = /dev/nqft2       Unit 2, no rewind-on-close
               7 = /dev/nqft3       Unit 3, no rewind-on-close
              16 = /dev/zqft0       Unit 0, rewind-on-close, compression
              17 = /dev/zqft1       Unit 1, rewind-on-close, compression
              18 = /dev/zqft2       Unit 2, rewind-on-close, compression
              19 = /dev/zqft3       Unit 3, rewind-on-close, compression
              20 = /dev/nzqft0      Unit 0, no rewind-on-close, compression
              21 = /dev/nzqft1      Unit 1, no rewind-on-close, compression
              22 = /dev/nzqft2      Unit 2, no rewind-on-close, compression
```

```
                23 = /dev/nzqft3      Unit 3, no rewind-on-close, compression
                32 = /dev/rawqft0     Unit 0, rewind-on-close, no file marks
                33 = /dev/rawqft1     Unit 1, rewind-on-close, no file marks
                34 = /dev/rawqft2     Unit 2, rewind-on-close, no file marks
                35 = /dev/rawqft3     Unit 3, rewind-on-close, no file marks
                36 = /dev/nrawqft0    Unit 0, no rewind-on-close, no file marks
                37 = /dev/nrawqft1    Unit 1, no rewind-on-close, no file marks
                38 = /dev/nrawqft2    Unit 2, no rewind-on-close, no file marks
                39 = /dev/nrawqft3    Unit 3, no rewind-on-close, no file marks
  block         Third Matsushita (Panasonic/SoundBlaster) CD-ROM
                 0 = /dev/sbpcd8      Panasonic CD-ROM controller 2 unit 0
                 1 = /dev/sbpcd9      Panasonic CD-ROM controller 2 unit 1
                 2 = /dev/sbpcd10     Panasonic CD-ROM controller 2 unit 2
                 3 = /dev/sbpcd11     Panasonic CD-ROM controller 2 unit 3

28 char         Stallion serial card - card programming
                 0 = /dev/staliomem0  First Stallion card I/O memory
                 1 = /dev/staliomem1  Second Stallion card I/O memory
                 2 = /dev/staliomem2  Third Stallion card I/O memory
                 3 = /dev/staliomem3  Fourth Stallion card I/O memory
  char Atari SLM ACSI laser printer (68k/Atari)
                 0 = /dev/slm0            First SLM laser printer
                 1 = /dev/slm1            Second SLM laser printer
                   ...
  block         Fourth Matsushita (Panasonic/SoundBlaster) CD-ROM
                 0 = /dev/sbpcd12     Panasonic CD-ROM controller 3 unit 0
                 1 = /dev/sbpcd13     Panasonic CD-ROM controller 3 unit 1
                 2 = /dev/sbpcd14     Panasonic CD-ROM controller 3 unit 2
                 3 = /dev/sbpcd15     Panasonic CD-ROM controller 3 unit 3
  block         ACSI disk (68k/Atari)
                 0 = /dev/ada         First ACSI disk whole disk
                16 = /dev/adb         Second ACSI disk whole disk
                32 = /dev/adc         Third ACSI disk whole disk
                   ...
               240 = /dev/adp             16th ACSI disk whole disk

                Partitions are handled in the same way as for IDE
                disks (see major number 3) except that the limit on
                partitions is 15, like SCSI.

29 char         Universal frame buffer
                 0 = /dev/fb0         First frame buffer
                 1 = /dev/fb1         Second frame buffer
                   ...
                31 = /dev/fb31            32nd frame buffer

                For backwards compatibility {2.6} the following
                progression is also handled by current kernels:
                 0 = /dev/fb0
                32 = /dev/fb1
```

...
224 = /dev/fb7

block Aztech/Orchid/Okano/Wearnes CD-ROM
0 = /dev/aztcd Aztech CD-ROM

30 char iBCS-2 compatibility devices
0 = /dev/socksys Socket access
1 = /dev/spx SVR3 local X interface
2 = /dev/inet/arp Network access
2 = /dev/inet/icmp Network access
2 = /dev/inet/ip Network access
2 = /dev/inet/udp Network access
2 = /dev/inet/tcp Network access

Additionally, iBCS-2 requires /dev/nfsd to be a link to /dev/socksys, and /dev/X0R to be a link to /dev/null.

block Philips LMS CM-205 CD-ROM
0 = /dev/cm205cd Philips LMS CM-205 CD-ROM

/dev/lmscd is an older name for this device. This driver does not work with the CM-205MS CD-ROM.

31 char MPU-401 MIDI
0 = /dev/mpu401data MPU-401 data port
1 = /dev/mpu401stat MPU-401 status port
block ROM/flash memory card
0 = /dev/rom0 First ROM card (rw)
...
7 = /dev/rom7 Eighth ROM card (rw)
8 = /dev/rrom0 First ROM card (ro)
...
15 = /dev/rrom7 Eighth ROM card (ro)
16 = /dev/flash0 First flash memory card (rw)
...
23 = /dev/flash7 Eighth flash memory card (rw)
24 = /dev/rflash0 First flash memory card (ro)
...
31 = /dev/rflash7 Eighth flash memory card (ro)

The read-write (rw) devices support back-caching written data in RAM, as well as writing to flash RAM devices. The read-only devices (ro) support reading only.

32 char Specialix serial card
0 = /dev/ttyX0 First Specialix port
1 = /dev/ttyX1 Second Specialix port

...

block Philips LMS CM-206 CD-ROM
0 = /dev/cm206cd Philips LMS CM-206 CD-ROM

33 char Specialix serial card - alternate devices
0 = /dev/cux0 Callout device for ttyX0
1 = /dev/cux1 Callout device for ttyX1
...

block Third IDE hard disk/CD-ROM interface
0 = /dev/hde Master: whole disk (or CD-ROM)
64 = /dev/hdf Slave: whole disk (or CD-ROM)

Partitions are handled the same way as for the first interface (see major number 3).

34 char Z8530 HDLC driver
0 = /dev/scc0 First Z8530, first port
1 = /dev/scc1 First Z8530, second port
2 = /dev/scc2 Second Z8530, first port
3 = /dev/scc3 Second Z8530, second port
...

In a previous version these devices were named /dev/sc1 for /dev/scc0, /dev/sc2 for /dev/scc1, and so on.

block Fourth IDE hard disk/CD-ROM interface
0 = /dev/hdg Master: whole disk (or CD-ROM)
64 = /dev/hdh Slave: whole disk (or CD-ROM)

Partitions are handled the same way as for the first interface (see major number 3).

35 char tclmidi MIDI driver
0 = /dev/midi0 First MIDI port, kernel timed
1 = /dev/midi1 Second MIDI port, kernel timed
2 = /dev/midi2 Third MIDI port, kernel timed
3 = /dev/midi3 Fourth MIDI port, kernel timed
64 = /dev/rmidi0 First MIDI port, untimed
65 = /dev/rmidi1 Second MIDI port, untimed
66 = /dev/rmidi2 Third MIDI port, untimed
67 = /dev/rmidi3 Fourth MIDI port, untimed
128 = /dev/smpte0 First MIDI port, SMPTE timed
129 = /dev/smpte1 Second MIDI port, SMPTE timed
130 = /dev/smpte2 Third MIDI port, SMPTE timed
131 = /dev/smpte3 Fourth MIDI port, SMPTE timed

block Slow memory ramdisk
0 = /dev/slram Slow memory ramdisk

36 char Netlink support

```
                0 = /dev/route          Routing, device updates, kernel to user
                1 = /dev/skip           enSKIP security cache control
                3 = /dev/fwmonitor      Firewall packet copies
               16 = /dev/tap0                   First Ethertap device
                  ...
               31 = /dev/tap15          16th Ethertap device
  block        MCA ESDI hard disk
                0 = /dev/eda            First ESDI disk whole disk
               64 = /dev/edb            Second ESDI disk whole disk
                  ...

                Partitions are handled in the same way as IDE disks
                (see major number 3).

37 char         IDE tape
                0 = /dev/ht0            First IDE tape
                1 = /dev/ht1            Second IDE tape
                  ...
              128 = /dev/nht0                   First IDE tape, no rewind-on-close
              129 = /dev/nht1                   Second IDE tape, no rewind-on-close
                  ...

                Currently, only one IDE tape drive is supported.

  block         Zorro II ramdisk
                0 = /dev/z2ram          Zorro II ramdisk

38 char         Myricom PCI Myrinet board
                0 = /dev/mlanai0        First Myrinet board
                1 = /dev/mlanai1        Second Myrinet board
                  ...

                This device is used for status query, board control
                and "user level packet I/O."  This board is also
                accessible as a standard networking "eth" device.

  block         Reserved for Linux/AP+

39 char         ML-16P experimental I/O board
                0 = /dev/ml16pa-a0      First card, first analog channel
                1 = /dev/ml16pa-a1      First card, second analog channel
                  ...
               15 = /dev/ml16pa-a15             First card, 16th analog channel
               16 = /dev/ml16pa-d       First card, digital lines
               17 = /dev/ml16pa-c0      First card, first counter/timer
               18 = /dev/ml16pa-c1      First card, second counter/timer
               19 = /dev/ml16pa-c2      First card, third counter/timer
               32 = /dev/ml16pb-a0      Second card, first analog channel
               33 = /dev/ml16pb-a1      Second card, second analog channel
                  ...
```

47 = /dev/ml16pb-a15 Second card, 16th analog channel
48 = /dev/ml16pb-d Second card, digital lines
49 = /dev/ml16pb-c0 Second card, first counter/timer
50 = /dev/ml16pb-c1 Second card, second counter/timer
51 = /dev/ml16pb-c2 Second card, third counter/timer
...

block Reserved for Linux/AP+

40 char Matrox Meteor frame grabber {2.6}
0 = /dev/mmetfgrab Matrox Meteor frame grabber

block Syquest EZ135 parallel port removable drive
0 = /dev/eza Parallel EZ135 drive, whole disk

This device is obsolete and will be removed in a
future version of Linux. It has been replaced with
the parallel port IDE disk driver at major number 45.
Partitions are handled in the same way as IDE disks
(see major number 3).

41 char Yet Another Micro Monitor
0 = /dev/yamm Yet Another Micro Monitor

block MicroSolutions BackPack parallel port CD-ROM
0 = /dev/bpcd BackPack CD-ROM

This device is obsolete and will be removed in a
future version of Linux. It has been replaced with
the parallel port ATAPI CD-ROM driver at major number 46.

42 Demo/sample use

This number is intended for use in sample code, as
well as a general "example" device number. It
should never be used for a device driver that is being
distributed; either obtain an official number or use
the local/experimental range. The sudden addition or
removal of a driver with this number should not cause
ill effects to the system (bugs excepted.)

IN PARTICULAR, ANY DISTRIBUTION WHICH CONTAINS A
DEVICE DRIVER USING MAJOR NUMBER 42 IS NONCOMPLIANT.

43 char isdn4linux virtual modem
0 = /dev/ttyI0 First virtual modem
...
63 = /dev/ttyI63 64th virtual modem

block Network block devices
0 = /dev/nb0 First network block device
1 = /dev/nb1 Second network block device
...

Network Block Device is somehow similar to loopback devices: If you read from it, it sends packet accross network asking server for data. If you write to it, it sends packet telling server to write. It could be used to mounting filesystems over the net, swapping over the net, implementing block device in userland etc.

44 char isdn4linux virtual modem - alternate devices
- 0 = /dev/cui0 Callout device for ttyI0
- ...
- 63 = /dev/cui63 Callout device for ttyI63

block Flash Translatio Layer (FTL) filesystems
- 0 = /dev/ftla FTL on first Memory Technology Device
- 16 = /dev/ftlb FTL on second Memory Technology Device
- 32 = /dev/ftlc FTL on third Memory Technology Device
- ...
- 240 = /dev/ftlp FTL on 16th Memory Technology Device

Partitions are handled in the same way as for IDE disks (see major number 3) expect that the partition limit is 15 rather than 63 per disk (same as SCSI.)

45 char isdn4linux ISDN BRI driver
- 0 = /dev/isdn0 First virtual B channel raw data
- ...
- 63 = /dev/isdn63 64th virtual B channel raw data
- 64 = /dev/isdnctrl0 First channel control/debug
- ...
- 127 = /dev/isdnctrl63 64th channel control/debug

- 128 = /dev/ippp0 First SyncPPP device
- ...
- 191 = /dev/ippp63 64th SyncPPP device

- 255 = /dev/isdninfo ISDN monitor interface

block Parallel port IDE disk devices
- 0 = /dev/pda First parallel port IDE disk
- 16 = /dev/pdb Second parallel port IDE disk
- 32 = /dev/pdc Third parallel port IDE disk
- 48 = /dev/pdd Fourth parallel port IDE disk

Partitions are handled in the same way as for IDE disks (see major number 3) except that the partition limit is 15 rather than 63 per disk.

46 char Comtrol Rocketport serial card
- 0 = /dev/ttyR0 First Rocketport port
- 1 = /dev/ttyR1 Second Rocketport port
- ...

block Parallel port ATAPI CD-ROM devices

0 = /dev/pcd0 First parallel port ATAPI CD-ROM
1 = /dev/pcd1 Second parallel port ATAPI CD-ROM
2 = /dev/pcd2 Third parallel port ATAPI CD-ROM
3 = /dev/pcd3 Fourth parallel port ATAPI CD-ROM

47 char Comtrol Rocketport serial card - alternate devices
0 = /dev/cur0 Callout device for ttyR0
1 = /dev/cur1 Callout device for ttyR1
...

block Parallel port ATAPI disk devices
0 = /dev/pf0 First parallel port ATAPI disk
1 = /dev/pf1 Second parallel port ATAPI disk
2 = /dev/pf2 Third parallel port ATAPI disk
3 = /dev/pf3 Fourth parallel port ATAPI disk

This driver is intended for floppy disks and similar devices and hence does not support partitioning.

48 char SDL RISCom serial card
0 = /dev/ttyL0 First RISCom port
1 = /dev/ttyL1 Second RISCom port
...

block Mylex DAC960 PCI RAID controller; first controller
0 = /dev/rd/c0d0 First disk, whole disk
8 = /dev/rd/c0d1 Second disk, whole disk
...
248 = /dev/rd/c0d15 32nd disk, whole disk

For partitions add:
0 = /dev/rd/c?d? Whole disk
1 = /dev/rd/c?d?p1 First partition
...
7 = /dev/rd/c?d?p7 Seventh partition

49 char SDL RISCom serial card - alternate devices
0 = /dev/cul0 Callout device for ttyL0
1 = /dev/cul1 Callout device for ttyL1
...

block Mylex DAC960 PCI RAID controller; second controller
0 = /dev/rd/c1d0 First disk, whole disk
8 = /dev/rd/c1d1 Second disk, whole disk
...
248 = /dev/rd/c1d15 16th disk, whole disk

Partitions are handled as for major 48.

50 char Reserved for GLINT

block Mylex DAC960 PCI RAID controller; third controller
0 = /dev/rd/c2d0 First disk, whole disk

8 = /dev/rd/c2d1 Second disk, whole disk
...
248 = /dev/rd/c2d15 16th disk, whole disk

51 char Baycom radio modem
0 = /dev/bc0 First Baycom radio modem
1 = /dev/bc1 Second Baycom radio modem
...
block Mylex DAC960 PCI RAID controller; fourth controller
0 = /dev/rd/c3d0 First disk, whole disk
8 = /dev/rd/c3d1 Second disk, whole disk
...
248 = /dev/rd/c3d15 16th disk, whole disk

Partitions are handled as for major 48.

52 char Spellcaster DataComm/BRI ISDN card
0 = /dev/dcbri0 First DataComm card
1 = /dev/dcbri1 Second DataComm card
2 = /dev/dcbri2 Third DataComm card
3 = /dev/dcbri3 Fourth DataComm card
block Mylex DAC960 PCI RAID controller; fifth controller
0 = /dev/rd/c4d0 First disk, whole disk
8 = /dev/rd/c4d1 Second disk, whole disk
...
248 = /dev/rd/c4d15 16th disk, whole disk

Partitions are handled as for major 48.

53 char BDM interface for remote debugging MC683xx microcontrollers
0 = /dev/pd_bdm0 PD BDM interface on lp0
1 = /dev/pd_bdm1 PD BDM interface on lp1
2 = /dev/pd_bdm2 PD BDM interface on lp2
4 = /dev/icd_bdm0 ICD BDM interface on lp0
5 = /dev/icd_bdm1 ICD BDM interface on lp1
6 = /dev/icd_bdm2 ICD BDM interface on lp2

This device is used for the interfacing to the MC683xx microcontrollers via Background Debug Mode by use of a Parallel Port interface. PD is the Motorola Public Domain Interface and ICD is the commercial interface by P&E.

block Mylex DAC960 PCI RAID controller; sixth controller
0 = /dev/rd/c5d0 First disk, whole disk
8 = /dev/rd/c5d1 Second disk, whole disk
...
248 = /dev/rd/c5d15 16th disk, whole disk

Partitions are handled as for major 48.

54 char Electrocardiognosis Holter serial card
0 = /dev/holter0 First Holter port
1 = /dev/holter1 Second Holter port
2 = /dev/holter2 Third Holter port

A custom serial card used by Electrocardiognosis SRL
to transfer data from Holter
24-hour heart monitoring equipment.

block Mylex DAC960 PCI RAID controller; seventh controller
0 = /dev/rd/c6d0 First disk, whole disk
8 = /dev/rd/c6d1 Second disk, whole disk
...
248 = /dev/rd/c6d15 16th disk, whole disk

Partitions are handled as for major 48.

55 char DSP56001 digital signal processor
0 = /dev/dsp56k First DSP56001
block Mylex DAC960 PCI RAID controller; eigth controller
0 = /dev/rd/c7d0 First disk, whole disk
8 = /dev/rd/c7d1 Second disk, whole disk
...
248 = /dev/rd/c7d15 16th disk, whole disk

Partitions are handled as for major 48.

56 char Apple Desktop Bus
0 = /dev/adb ADB bus control

Additional devices will be added to this number, all
starting with /dev/adb.

block Fifth IDE hard disk/CD-ROM interface
0 = /dev/hdi Master: whole disk (or CD-ROM)
64 = /dev/hdj Slave: whole disk (or CD-ROM)

Partitions are handled the same way as for the first
interface (see major number 3).

57 char Hayes ESP serial card
0 = /dev/ttyP0 First ESP port
1 = /dev/ttyP1 Second ESP port
...

block Sixth IDE hard disk/CD-ROM interface
0 = /dev/hdk Master: whole disk (or CD-ROM)
64 = /dev/hdl Slave: whole disk (or CD-ROM)

Partitions are handled the same way as for the first
interface (see major number 3).

58 char Hayes ESP serial card - alternate devices
0 = /dev/cup0 Callout device for ttyP0
1 = /dev/cup1 Callout device for ttyP1
...
block Reserved for logical volume manager

59 char sf firewall package
0 = /dev/firewall Communication with sf kernel module

block Generic PDA filesystem device
0 = /dev/pda0 First PDA device
1 = /dev/pda1 Second PDA device
...

The pda devices are used to mount filesystems on
remote pda's (basically slow handheld machines with
proprietary OS's and limited memory and storage
running small fs translation drivers) through serial /
IRDA / parallel links.

NAMING CONFLICT -- PROPOSED REVISED NAME /dev/rpda0 etc

60-63 LOCAL/EXPERIMENTAL USE
Allocated for local/experimental use. For devices not
assigned official numbers, these ranges should be
used, in order to avoid conflicting with future assignments.

64 char ENskip kernel encryption package
0 = /dev/enskip Communication with ENskip kernel module

65 char Sundance "plink" Transputer boards
0 = /dev/plink0 First plink device
1 = /dev/plink1 Second plink device
2 = /dev/plink2 Third plink device
3 = /dev/plink3 Fourth plink device
64 = /dev/rplink0 First plink device, raw
65 = /dev/rplink1 Second plink device, raw
66 = /dev/rplink2 Third plink device, raw
67 = /dev/rplink3 Fourth plink device, raw
128 = /dev/plink0d First plink device, debug
129 = /dev/plink1d Second plink device, debug
130 = /dev/plink2d Third plink device, debug
131 = /dev/plink3d Fourth plink device, debug
192 = /dev/rplink0d First plink device, raw, debug
193 = /dev/rplink1d Second plink device, raw, debug
194 = /dev/rplink2d Third plink device, raw, debug
195 = /dev/rplink3d Fourth plink device, raw, debug

This is a commercial driver; contact James Howes
for information.

block SCSI disk devices (16-31)
0 = /dev/sdq 16th SCSI disk whole disk
16 = /dev/sdr 17th SCSI disk whole disk
32 = /dev/sds 18th SCSI disk whole disk
...
240 = /dev/sdaf 32nd SCSI disk whole disk

Partitions are handled in the same way as for IDE
disks (see major number 3) except that the limit on
partitions is 15.

66 char YARC PowerPC PCI coprocessor card
0 = /dev/yppcpci0 First YARC card
1 = /dev/yppcpci1 Second YARC card
...

block SCSI disk devices (32-47)
0 = /dev/sdag 33th SCSI disk whole disk
16 = /dev/sdah 34th SCSI disk whole disk
32 = /dev/sdai 35th SCSI disk whole disk
...
240 = /dev/sdav 48nd SCSI disk whole disk

Partitions are handled in the same way as for IDE
disks (see major number 3) except that the limit on
partitions is 15.

67 char Coda network file system
0 = /dev/cfs0 Coda cache manager

See http://www.coda.cs.cmu.edu for information about Coda.

block SCSI disk devices (48-63)
0 = /dev/sdaw 49th SCSI disk whole disk
16 = /dev/sdax 50th SCSI disk whole disk
32 = /dev/sday 51st SCSI disk whole disk
...
240 = /dev/sdbl 64th SCSI disk whole disk

Partitions are handled in the same way as for IDE
disks (see major number 3) except that the limit on
partitions is 15.

68 char CAPI 2.0 interface
0 = /dev/capi20 Control device
1 = /dev/capi20.00 First CAPI 2.0 application

```
  2 = /dev/capi20.01   Second CAPI 2.0 application
    ...
 20 = /dev/capi20.19   19th CAPI 2.0 application
```

ISDN CAPI 2.0 driver for use with CAPI 2.0 applications; currently supports the AVM B1 card.

block SCSI disk devices (64-79)

```
  0 = /dev/sdbm              64th SCSI disk whole disk
 16 = /dev/sdbn              65th SCSI disk whole disk
 32 = /dev/sdbo              66th SCSI disk whole disk
    ...
240 = /dev/sdcb              80th SCSI disk whole disk
```

Partitions are handled in the same way as for IDE disks (see major number 3) except that the limit on partitions is 15.

69 char MA16 numeric accelerator card

```
  0 = /dev/ma16              Board memory access
```

block SCSI disk devices (80-95)

```
  0 = /dev/sdcc         81st SCSI disk whole disk
 16 = /dev/sdcd              82nd SCSI disk whole disk
 32 = /dev/sdce              83th SCSI disk whole disk
    ...
240 = /dev/sdcr              96th SCSI disk whole disk
```

Partitions are handled in the same way as for IDE disks (see major number 3) except that the limit on partitions is 15.

70 char SpellCaster Protocol Services Interface

```
  0 = /dev/apscfg    Configuration interface
  1 = /dev/apsauth   Authentication interface
  2 = /dev/apslog    Logging interface
  3 = /dev/apsdbg    Debugging interface
 64 = /dev/apsisdn   ISDN command interface
 65 = /dev/apsasync  Async command interface
128 = /dev/apsmon    Monitor interface
```

block SCSI disk devices (96-111)

```
  0 = /dev/sdcs         97th SCSI disk whole disk
 16 = /dev/sdct              98th SCSI disk whole disk
 32 = /dev/sdcu              99th SCSI disk whole disk
    ...
240 = /dev/sddh             112nd SCSI disk whole disk
```

Partitions are handled in the same way as for IDE disks (see major number 3) except that the limit on

partitions is 15.

71 char Computone IntelliPort II serial card

0 = /dev/ttyF0	IntelliPort II board 0, port 0
1 = /dev/ttyF1	IntelliPort II board 0, port 1
...	
63 = /dev/ttyF63	IntelliPort II board 0, port 63
64 = /dev/ttyF64	IntelliPort II board 1, port 0
65 = /dev/ttyF65	IntelliPort II board 1, port 1
...	
127 = /dev/ttyF127	IntelliPort II board 1, port 63
128 = /dev/ttyF128	IntelliPort II board 2, port 0
129 = /dev/ttyF129	IntelliPort II board 2, port 1
...	
191 = /dev/ttyF191	IntelliPort II board 2, port 63
192 = /dev/ttyF192	IntelliPort II board 3, port 0
193 = /dev/ttyF193	IntelliPort II board 3, port 1
...	
255 = /dev/ttyF255	IntelliPort II board 3, port 63

block SCSI disk devices (112-127)

0 = /dev/sddi	113th SCSI disk whole disk
16 = /dev/sddj	114th SCSI disk whole disk
32 = /dev/sddk	115th SCSI disk whole disk
...	
240 = /dev/sddx	128th SCSI disk whole disk

Partitions are handled in the same way as for IDE disks (see major number 3) except that the limit on partitions is 15.

72 char Computone IntelliPort II serial card - alternate devices

0 = /dev/cuf0	Callout device for ttyF0
1 = /dev/cuf1	Callout device for ttyF1
...	
63 = /dev/cuf63	Callout device for ttyF63
64 = /dev/cuf64	Callout device for ttyF64
65 = /dev/cuf65	Callout device for ttyF65
...	
127 = /dev/cuf127	Callout device for ttyF127
128 = /dev/cuf128	Callout device for ttyF128
129 = /dev/cuf129	Callout device for ttyF129
...	
191 = /dev/cuf191	Callout device for ttyF191
192 = /dev/cuf192	Callout device for ttyF192
193 = /dev/cuf193	Callout device for ttyF193
...	
255 = /dev/cuf255	Callout device for ttyF255

block Compaq Intelligent Drive Array, first controller

0 = /dev/ida/c0d0 First logical drive whole disk
16 = /dev/ida/c0d1 Second logical drive whole disk
...
240 = /dev/ida/c0d15 16th logical drive whole disk

Partitions are handled the same way as for Mylex DAC960 (see major number 48) except that the limit on partitions is 15.

73 char Computone IntelliPort II serial card - control devices
0 = /dev/ip2ipl0 Loadware device for board 0
1 = /dev/ip2stat0 Status device for board 0
4 = /dev/ip2ipl1 Loadware device for board 1
5 = /dev/ip2stat1 Status device for board 1
8 = /dev/ip2ipl2 Loadware device for board 2
9 = /dev/ip2stat2 Status device for board 2
12 = /dev/ip2ipl3 Loadware device for board 3
13 = /dev/ip2stat3 Status device for board 3

block Compaq Intelligent Drive Array, second controller
0 = /dev/ida/c1d0 First logical drive whole disk
16 = /dev/ida/c1d1 Second logical drive whole disk
...
240 = /dev/ida/c1d15 16th logical drive whole disk

Partitions are handled the same way as for Mylex DAC960 (see major number 48) except that the limit on partitions is 15.

74 char SCI bridge
0 = /dev/SCI/0 SCI device 0
1 = /dev/SCI/1 SCI device 1
...

Currently for Dolphin Interconnect Solutions' PCI-SCI bridge.

block Compaq Intelligent Drive Array, third controller
0 = /dev/ida/c2d0 First logical drive whole disk
16 = /dev/ida/c2d1 Second logical drive whole disk
...
240 = /dev/ida/c2d15 16th logical drive whole disk

Partitions are handled the same way as for Mylex DAC960 (see major number 48) except that the limit on partitions is 15.

75 char Specialix IO8+ serial card
0 = /dev/ttyW0 First IO8+ port, first card
1 = /dev/ttyW1 Second IO8+ port, first card

...
8 = /dev/ttyW8 First IO8+ port, second card
...

block Compaq Intelligent Drive Array, fourth controller
0 = /dev/ida/c3d0 First logical drive whole disk
16 = /dev/ida/c3d1 Second logical drive whole disk
...
240 = /dev/ida/c3d15 16th logical drive whole disk

Partitions are handled the same way as for Mylex DAC960 (see major number 48) except that the limit on partitions is 15.

76 char Specialix IO8+ serial card - alternate devices
0 = /dev/cuw0 Callout device for ttyW0
1 = /dev/cuw1 Callout device for ttyW1
...
8 = /dev/cuw8 Callout device for ttyW8
...

block Compaq Intelligent Drive Array, fifth controller
0 = /dev/ida/c4d0 First logical drive whole disk
16 = /dev/ida/c4d1 Second logical drive whole disk
...
240 = /dev/ida/c4d15 16th logical drive whole disk

Partitions are handled the same way as for Mylex DAC960 (see major number 48) except that the limit on partitions is 15.

77 char ComScire Quantum Noise Generator
0 = /dev/qng ComScire Quantum Noise Generator

block Compaq Intelligent Drive Array, sixth controller
0 = /dev/ida/c5d0 First logical drive whole disk
16 = /dev/ida/c5d1 Second logical drive whole disk
...
240 = /dev/ida/c5d15 16th logical drive whole disk

Partitions are handled the same way as for Mylex DAC960 (see major number 48) except that the limit on partitions is 15.

78 char PAM Software's multimodem boards
0 = /dev/ttyM0 First PAM modem
1 = /dev/ttyM1 Second PAM modem
...

block Compaq Intelligent Drive Array, seventh controller
0 = /dev/ida/c6d0 First logical drive whole disk
16 = /dev/ida/c6d1 Second logical drive whole disk
...
240 = /dev/ida/c6d15 16th logical drive whole disk

Partitions are handled the same way as for Mylex DAC960 (see major number 48) except that the limit on partitions is 15.

79 char PAM Software's multimodem boards - alternate devices
0 = /dev/cum0 Callout device for ttyM0
1 = /dev/cum1 Callout device for ttyM1
...

block Compaq Intelligent Drive Array, eigth controller
0 = /dev/ida/c7d0 First logical drive whole disk
16 = /dev/ida/c7d1 Second logical drive whole disk
...
240 = /dev/ida/c715 16th logical drive whole disk

Partitions are handled the same way as for Mylex DAC960 (see major number 48) except that the limit on partitions is 15.

80 char Photometrics AT200 CCD camera
0 = /dev/at200 Photometrics AT200 CCD camera

block I2O hard disk
0 = /dev/i2o/hda First I2O hard disk, whole disk
16 = /dev/i2o/hdb Second I2O hard disk, whole disk
...
240 = /dev/i2o/hdp 16th I2O hard disk, whole disk

Partitions are handled in the same way as for IDE disks (see major number 3) except that the limit on partitions is 15.

81 char video4linux
0 = /dev/video0 Video capture/overlay device
...
63 = /dev/video63 Video capture/overlay device
64 = /dev/radio0 Radio device
...
127 = /dev/radio63 Radio device
192 = /dev/vtx0 Teletext device
...

223 = /dev/vtx31 Teletext device
224 = /dev/vbi0 Vertical blank interrupt
...
255 = /dev/vbi31 Vertical blank interrupt

block I2O hard disk
0 = /dev/i2o/hdq 17th I2O hard disk, whole disk
16 = /dev/i2o/hdr 18th I2O hard disk, whole disk
...
240 = /dev/i2o/hdaf 32nd I2O hard disk, whole disk

Partitions are handled in the same way as for IDE disks (see major number 3) except that the limit on partitions is 15.

82 char WiNRADiO communications receiver card
0 = /dev/winradio0 First WiNRADiO card
1 = /dev/winradio1 Second WiNRADiO card
...

The driver and documentation may be obtained from http://www.proximity.com.au/~brian/winradio/

block I2O hard disk
0 = /dev/i2o/hdag 33rd I2O hard disk, whole disk
16 = /dev/i2o/hdah 34th I2O hard disk, whole disk
...
240 = /dev/i2o/hdav 48th I2O hard disk, whole disk

Partitions are handled in the same way as for IDE disks (see major number 3) except that the limit on partitions is 15.

83 char Teletext/videotext interfaces {2.6}
0 = /dev/vtx Teletext decoder
16 = /dev/vttuner TV tuner on teletext interface

Devices for the driver contained in the VideoteXt package. More information on http://home.pages.de/~videotext/

block I2O hard disk
0 = /dev/i2o/hdaw 49th I2O hard disk, whole disk
16 = /dev/i2o/hdax 50th I2O hard disk, whole disk
...
240 = /dev/i2o/hdbl 64th I2O hard disk, whole disk

Partitions are handled in the same way as for IDE disks (see major number 3) except that the limit on partitions is 15.

84 char Ikon 1011[57] Versatec Greensheet Interface
0 = /dev/ihcp0 First Greensheet port
1 = /dev/ihcp1 Second Greensheet port

block I2O hard disk
0 = /dev/i2o/hdbm 65th I2O hard disk, whole disk
16 = /dev/i2o/hdbn 66th I2O hard disk, whole disk
...
240 = /dev/i2o/hdcb 80th I2O hard disk, whole disk

Partitions are handled in the same way as for IDE disks (see major number 3) except that the limit on partitions is 15.

85 char Linux/SGI shared memory input queue
0 = /dev/shmiq Master shared input queue
1 = /dev/qcntl0 First device pushed
2 = /dev/qcntl1 Second device pushed
...

block I2O hard disk
0 = /dev/i2o/hdcc 81st I2O hard disk, whole disk
16 = /dev/i2o/hdcd 82nd I2O hard disk, whole disk
...
240 = /dev/i2o/hdcr 96th I2O hard disk, whole disk

Partitions are handled in the same way as for IDE disks (see major number 3) except that the limit on partitions is 15.

86 char SCSI media changer
0 = /dev/sch0 First SCSI media changer
1 = /dev/sch1 Second SCSI media changer
...

block I2O hard disk
0 = /dev/i2o/hdcs 97th I2O hard disk, whole disk
16 = /dev/i2o/hdct 98th I2O hard disk, whole disk
...
240 = /dev/i2o/hddh 112th I2O hard disk, whole disk

Partitions are handled in the same way as for IDE disks (see major number 3) except that the limit on partitions is 15.

87 char Sony Control-A1 stereo control bus
0 = /dev/controla0 First device on chain
1 = /dev/controla1 Second device on chain
...

block I2O hard disk
0 = /dev/i2o/hddi 113rd I2O hard disk, whole disk
16 = /dev/i2o/hddj 114th I2O hard disk, whole disk
...
240 = /dev/i2o/hddx 128th I2O hard disk, whole disk

Partitions are handled in the same way as for IDE disks (see major number 3) except that the limit on partitions is 15.

88 char COMX synchronous serial card
0 = /dev/comx0 COMX channel 0
1 = /dev/comx1 COMX channel 1
...

block Seventh IDE hard disk/CD-ROM interface
0 = /dev/hdm Master: whole disk (or CD-ROM)
64 = /dev/hdn Slave: whole disk (or CD-ROM)

Partitions are handled the same way as for the first interface (see major number 3).

89 char I2C bus interface
0 = /dev/i2c-0 First I2C adapter
1 = /dev/i2c-1 Second I2C adapter
...

block Eighth IDE hard disk/CD-ROM interface
0 = /dev/hdo Master: whole disk (or CD-ROM)
64 = /dev/hdp Slave: whole disk (or CD-ROM)

Partitions are handled the same way as for the first interface (see major number 3).

90 char Memory Technology Device (RAM, ROM, Flash)
0 = /dev/mtd0 First MTD (rw)
1 = /dev/mtdr0 First MTD (ro)
...
30 = /dev/mtd15 16th MTD (rw)
31 = /dev/mtdr15 16th MTD (ro)

block Ninth IDE hard disk/CD-ROM interface
0 = /dev/hdq Master: whole disk (or CD-ROM)
64 = /dev/hdr Slave: whole disk (or CD-ROM)

Partitions are handled the same way as for the first interface (see major number 3).

91 char CAN-Bus devices
0 = /dev/can0 First CAN-Bus controller

1 = /dev/can1 Second CAN-Bus controller
...

block Tenth IDE hard disk/CD-ROM interface
0 = /dev/hds Master: whole disk (or CD-ROM)
64 = /dev/hdt Slave: whole disk (or CD-ROM)

Partitions are handled the same way as for the first interface (see major number 3).

92 char Reserved for ith Kommunikationstechnik MIC ISDN card

block PPDD encrypted disk driver
0 = /dev/ppdd0 First encrypted disk
1 = /dev/ppdd1 Second encrypted disk
...

Partitions are handled in the same way as for IDE disks (see major number 3) except that the limit on partitions is 15.

93 char IBM Smart Capture Card frame grabber {2.6}
0 = /dev/iscc0 First Smart Capture Card
1 = /dev/iscc1 Second Smart Capture Card
...
128 = /dev/isccctl0 First Smart Capture Card control
129 = /dev/isccctl1 Second Smart Capture Card control
...

block NAND Flash Translation Layer filesystem
0 = /dev/nftlaFirst NFTL layer
16 = /dev/nftlb Second NFTL layer
...
240 = /dev/nftlp 16th NTFL layer

94 char miroVIDEO DC10/30 capture/playback device {2.6}
0 = /dev/dcxx0 First capture card
1 = /dev/dcxx1 Second capture card
...

95 char IP filter
0 = /dev/ipl Filter control device/log file
1 = /dev/ipnat NAT control device/log file
2 = /dev/ipstate State information log file
3 = /dev/ipauth Authentication control device/log file

block IBM S/390 DASD block storage
0 = /dev/dasd0 First DASD device, major
1 = /dev/dasd0a First DASD device, block 1
2 = /dev/dasd0b First DASD device, block 2

3 = /dev/dasd0c First DASD device, block 3
4 = /dev/dasd1 Second DASD device, major
5 = /dev/dasd1a Second DASD device, block 1
6 = /dev/dasd1b Second DASD device, block 2
7 = /dev/dasd1c Second DASD device, block 3
...

96 char Parallel port ATAPI tape devices
0 = /dev/pt0 First parallel port ATAPI tape
1 = /dev/pt1 Second parallel port ATAPI tape
...
128 = /dev/npt0 First p.p. ATAPI tape, no rewind
129 = /dev/npt1 Second p.p. ATAPI tape, no rewind
...

block IBM S/390 VM/ESA minidisk
0 = /dev/msd0 First VM/ESA minidisk
1 = /dev/msd1 Second VM/ESA minidisk
...

97 char Parallel port generic ATAPI interface
0 = /dev/pg0 First parallel port ATAPI device
1 = /dev/pg1 Second parallel port ATAPI device
2 = /dev/pg2 Third parallel port ATAPI device
3 = /dev/pg3 Fourth parallel port ATAPI device

These devices support the same API as the generic SCSI devices.

block Packet writing for CD/DVD devices
0 = /dev/pktcdvd0 First packet-writing module
1 = /dev/pktcdvd1 Second packet-writing module
...

98 char Control and Measurement Device (comedi)
0 = /dev/comedi0 First comedi device
1 = /dev/comedi1 Second comedi device
...

See http://stm.lbl.gov/comedi or http://www.llp.fu-berlin.de/.

block User-mode virtual block device
0 = /dev/ubd0 First user-mode block device
1 = /dev/ubd1 Second user-mode block device
...

This device is used by the user-mode virtual kernel port.

99 char Raw parallel ports
0 = /dev/parport0 First parallel port

1 = /dev/parport1 Second parallel port
...

block JavaStation flash disk
0 = /dev/jsfd JavaStation flash disk

100 char Telephony for Linux
0 = /dev/phone0 First telephony device
1 = /dev/phone1 Second telephony device
...

101 char Motorola DSP 56xxx board
0 = /dev/mdspstat Status information
1 = /dev/mdsp1 First DSP board I/O controls
...
16 = /dev/mdsp16 16th DSP board I/O controls

block AMI HyperDisk RAID controller
0 = /dev/amiraid/ar0 First array whole disk
16 = /dev/amiraid/ar1 Second array whole disk
...
240 = /dev/amiraid/ar15 16th array whole disk

For each device, partitions are added as:
0 = /dev/amiraid/ar? Whole disk
1 = /dev/amiraid/ar?p1 First partition
2 = /dev/amiraid/ar?p2 Second partition
...
15 = /dev/amiraid/ar?p15 15th partition

102 char Philips SAA5249 Teletext signal decoder {2.6}
0 = /dev/tlk0 First Teletext decoder
1 = /dev/tlk1 Second Teletext decoder
2 = /dev/tlk2 Third Teletext decoder
3 = /dev/tlk3 Fourth Teletext decoder

block Compressed block device
0 = /dev/cbd/a First compressed block device, whole device
16 = /dev/cbd/b Second compressed block device, whole device
...
240 = /dev/cbd/p 16th compressed block device, whole device

Partitions are handled in the same way as for IDE
disks (see major number 3) except that the limit on
partitions is 15.

103 char Arla network file system
0 = /dev/xfs0 Arla XFS

Arla is a free clone of the Andrew File System, AFS.

Any resemblance with the Swedish milk producer is coincidental. For more information about the project, write to or subscribe to the arla announce mailing list by sending a mail to .

block Audit device
0 = /dev/audit Audit device

104 char Flash BIOS support

block Compaq Next Generation Drive Array, first controller
0 = /dev/cciss/c0d0 First logical drive, whole disk
16 = /dev/cciss/c0d1 Second logical drive, whole disk
...
240 = /dev/cciss/c0d15 16th logical drive, whole disk

Partitions are handled the same way as for Mylex DAC960 (see major number 48) except that the limit on partitions is 15.

105 char Comtrol VS-1000 serial controller
0 = /dev/ttyV0 First VS-1000 port
1 = /dev/ttyV1 Second VS-1000 port
...

block Compaq Next Generation Drive Array, second controller
0 = /dev/cciss/c1d0 First logical drive, whole disk
16 = /dev/cciss/c1d1 Second logical drive, whole disk
...
240 = /dev/cciss/c1d15 16th logical drive, whole disk

Partitions are handled the same way as for Mylex DAC960 (see major number 48) except that the limit on partitions is 15.

106 char Comtrol VS-1000 serial controller - alternate devices
0 = /dev/cuv0 First VS-1000 port
1 = /dev/cuv1 Second VS-1000 port
...

block Compaq Next Generation Drive Array, third controller
0 = /dev/cciss/c2d0 First logical drive, whole disk
16 = /dev/cciss/c2d1 Second logical drive, whole disk
...
240 = /dev/cciss/c2d15 16th logical drive, whole disk

Partitions are handled the same way as for Mylex DAC960 (see major number 48) except that the limit on partitions is 15.

107 char 3Dfx Voodoo Graphics device
0 = /dev/3dfx Primary 3Dfx graphics device

block Compaq Next Generation Drive Array, fourth controller
0 = /dev/cciss/c3d0 First logical drive, whole disk
16 = /dev/cciss/c3d1 Second logical drive, whole disk
...
240 = /dev/cciss/c3d15 16th logical drive, whole disk

Partitions are handled the same way as for Mylex DAC960 (see major number 48) except that the limit on partitions is 15.

108 char Device independent PPP interface
0 = /dev/ppp Device independent PPP interface

block Compaq Next Generation Drive Array, fifth controller
0 = /dev/cciss/c4d0 First logical drive, whole disk
16 = /dev/cciss/c4d1 Second logical drive, whole disk
...
240 = /dev/cciss/c4d15 16th logical drive, whole disk

Partitions are handled the same way as for Mylex DAC960 (see major number 48) except that the limit on partitions is 15.

109 char Reserved for logical volume manager

block Compaq Next Generation Drive Array, sixth controller
0 = /dev/cciss/c5d0 First logical drive, whole disk
16 = /dev/cciss/c5d1 Second logical drive, whole disk
...
240 = /dev/cciss/c5d15 16th logical drive, whole disk

Partitions are handled the same way as for Mylex DAC960 (see major number 48) except that the limit on partitions is 15.

110 char miroMEDIA Surround board
0 = /dev/srnd0 First miroMEDIA Surround board
1 = /dev/srnd1 Second miroMEDIA Surround board
...

block Compaq Next Generation Drive Array, seventh controller
0 = /dev/cciss/c6d0 First logical drive, whole disk
16 = /dev/cciss/c6d1 Second logical drive, whole disk
...
240 = /dev/cciss/c6d15 16th logical drive, whole disk

Partitions are handled the same way as for Mylex DAC960 (see major number 48) except that the limit on partitions is 15.

111 char Philips SAA7146-based audio/video card {2.6}
0 = /dev/av0 First A/V card
1 = /dev/av1 Second A/V card
...

block Compaq Next Generation Drive Array, eigth controller
0 = /dev/cciss/c7d0 First logical drive, whole disk
16 = /dev/cciss/c7d1 Second logical drive, whole disk
...
240 = /dev/cciss/c7d15 16th logical drive, whole disk

Partitions are handled the same way as for Mylex DAC960 (see major number 48) except that the limit on partitions is 15.

112 char ISI serial card
0 = /dev/ttyM0 First ISI port
1 = /dev/ttyM1 Second ISI port
...

There is currently a device-naming conflict between these and PAM multimodems (major 78).

block IBM iSeries virtual disk
0 = /dev/iseries/vda First virtual disk, whole disk
8 = /dev/iseries/vdb Second virtual disk, whole disk
...
200 = /dev/iseries/vdz 26th virtual disk, whole disk
208 = /dev/iseries/vdaa 27th virtual disk, whole disk
...
240 = /dev/iseries/vdaf 32nd virtual disk, whole disk

Partitions are handled in the same way as for IDE disks (see major number 3) except that the limit on partitions is 7.

113 char ISI serial card - alternate devices
0 = /dev/cum0 Callout device for ttyM0
1 = /dev/cum1 Callout device for ttyM1
...

block IBM iSeries virtual CD-ROM

0 = /dev/iseries/vcda First virtual CD-ROM
1 = /dev/iseries/vcdb Second virtual CD-ROM
...

114 char Picture Elements ISE board
  0 = /dev/ise0 First ISE board
  1 = /dev/ise1 Second ISE board
  ...
128 = /dev/isex0 Control node for first ISE board
129 = /dev/isex1 Control node for second ISE board
  ...

The ISE board is an embedded computer, optimized for image processing. The /dev/iseN nodes are the general I/O access to the board, the /dev/isex0 nodes command nodes used to control the board.

115 char Console driver speaker
  0 = /dev/speaker Speaker device file

Plays music using IBM BASIC style strings.

116 char Advanced Linux Sound Driver (ALSA)

117 char COSA/SRP synchronous serial card
  0 = /dev/cosa0c0 1st board, 1st channel
  1 = /dev/cosa0c1 1st board, 2nd channel
  ...
 16 = /dev/cosa1c0 2nd board, 1st channel
 17 = /dev/cosa1c1 2nd board, 2nd channel
  ...

118 char Solidum ???
  0 = /dev/solnp0
  1 = /dev/solnp1
  ...
128 = /dev/solnpctl0
129 = /dev/solnpctl1
  ...

119 char VMware virtual network control
  0 = /dev/vnet0 1st virtual network
  1 = /dev/vnet1 2nd virtual network
  ...

120-127 LOCAL/EXPERIMENTAL USE

128-135 char Unix98 PTY masters

These devices should not have corresponding device nodes; instead they should be accessed through the /dev/ptmx cloning interface.

136-143 char Unix98 PTY slaves
0 = /dev/pts/0 First Unix98 pseudo-TTY
1 = /dev/pts/1 Second Unix98 pesudo-TTY
...

These device nodes are automatically generated with the proper permissions and modes by mounting the devpts filesystem onto /dev/pts with the appropriate mount options (distribution dependent, however, on *most* distributions the appropriate options are "mode=0620,gid=".)

144 char Encapsulated PPP
0 = /dev/pppox0 First PPP over Ethernet
...
63 = /dev/pppox63 64th PPP over Ethernet

This is primarily used for ADSL.

The SST 5136-DN DeviceNet interface driver has been relocated to major 183 due to an unfortunate conflict.

145 char SAM9407-based soundcard
0 = /dev/sam0_mixer
1 = /dev/sam0_sequencer
2 = /dev/sam0_midi00
3 = /dev/sam0_dsp
4 = /dev/sam0_audio
6 = /dev/sam0_sndstat
18 = /dev/sam0_midi01
34 = /dev/sam0_midi02
50 = /dev/sam0_midi03
64 = /dev/sam1_mixer
...
128 = /dev/sam2_mixer
...
192 = /dev/sam3_mixer
...

Device functions match OSS, but offer a number of addons, which are sam9407 specific. OSS can be operated simultaneously, taking care of the codec.

146 char SYSTRAM SCRAMNet mirrored-memory network
0 = /dev/scramnet0 First SCRAMNet device
1 = /dev/scramnet1 Second SCRAMNet device
...

147 char Aueral Semiconductor Vortex Audio device
0 = /dev/aureal0 First Aureal Vortex

```
                1 = /dev/aureal1      Second Aureal Vortex
                  ...

148 char        Technology Concepts serial card
                0 = /dev/ttyT0        First TCL port
                1 = /dev/ttyT1        Second TCL port
                  ...

149 char        Technology Concepts serial card - alternate devices
                0 = /dev/cut0         Callout device for ttyT0
                1 = /dev/cut0         Callout device for ttyT1
                  ...

150 char        Real-Time Linux FIFOs
                0 = /dev/rtf0         First RTLinux FIFO
                1 = /dev/rtf1         Second RTLinux FIFO
                  ...

151 char        DPT I2O SmartRaid V controller
                0 = /dev/dpti0        First DPT I2O adapter
                1 = /dev/dpti1        Second DPT I2O adapter
                  ...

154 char        Specialix RIO serial card
                0 = /dev/ttySR0       First RIO port
                  ...
              255 = /dev/ttySR255     256th RIO port

155 char        Specialix RIO serial card - alternate devices
                0 = /dev/cusr0        Callout device for ttySR0
                  ...
              255 = /dev/cusr255      Callout device for ttySR255

156 char        Specialix RIO serial card
                0 = /dev/ttySR256     257th RIO port
                  ...
              255 = /dev/ttySR511     512th RIO port

157 char        Specialix RIO serial card - alternate devices
                0 = /dev/cusr256      Callout device for ttySR256
                  ...
              255 = /dev/cusr511      Callout device for ttySR511

158 char        Dialogic GammaLink fax driver
                0 = /dev/gfax0        GammaLink channel 0
                1 = /dev/gfax1        GammaLink channel 1
                  ...

159             RESERVED
```

160 char General Purpose Instrument Bus (GPIB)
0 = /dev/gpib0 First GPIB bus
1 = /dev/gpib1 Second GPIB bus
...

161 char IrCOMM devices (IrDA serial/parallel emulation)
0 = /dev/ircomm0 First IrCOMM device
1 = /dev/ircomm1 Second IrCOMM device
...
16 = /dev/irlpt0 First IrLPT device
17 = /dev/irlpt1 Second IrLPT device
...

162 char Raw block device interface
0 = /dev/rawctl Raw I/O control device
1 = /dev/raw/raw1 First raw I/O device
2 = /dev/raw/raw2 Second raw I/O device
...

163 char Radio Tech BIM-XXX-RS232 radio modem
0 = /dev/bimrt0 First BIM radio modem
1 = /dev/bimrt1 Second BIM radio modem
...

164 char Chase Research AT/PCI-Fast serial card
0 = /dev/ttyCH0 AT/PCI-Fast board 0, port 0
...
15 = /dev/ttyCH15 AT/PCI-Fast board 0, port 15
16 = /dev/ttyCH16 AT/PCI-Fast board 1, port 0
...
31 = /dev/ttyCH31 AT/PCI-Fast board 1, port 15
32 = /dev/ttyCH32 AT/PCI-Fast board 2, port 0
...
47 = /dev/ttyCH47 AT/PCI-Fast board 2, port 15
48 = /dev/ttyCH48 AT/PCI-Fast board 3, port 0
...
63 = /dev/ttyCH63 AT/PCI-Fast board 3, port 15

165 char Chase Research AT/PCI-Fast serial card - alternate devices
0 = /dev/cuch0 Callout device for ttyCH0
...
63 = /dev/cuch63 Callout device for ttyCH63

166 char ACM USB modems
0 = /dev/ttyACM0 First ACM modem
1 = /dev/ttyACM1 Second ACM modem
...

167 char ACM USB modems - alternate devices
0 = /dev/cuacm0 Callout device for ttyACM0

1 = /dev/cuacm1 Callout device for ttyACM1
...

168 char Eracom CSA7000 PCI encryption adaptor
0 = /dev/ecsa0 First CSA7000
1 = /dev/ecsa1 Second CSA7000
...

169 char Eracom CSA8000 PCI encryption adaptor
0 = /dev/ecsa8-0 First CSA8000
1 = /dev/ecsa8-1 Second CSA8000
...

170 char AMI MegaRAC remote access controller
0 = /dev/megarac0 First MegaRAC card
1 = /dev/megarac1 Second MegaRAC card
...

171 char Reserved for IEEE 1394 (Firewire)

172 char Moxa Intellio serial card
0 = /dev/ttyMX0 First Moxa port
1 = /dev/ttyMX1 Second Moxa port
...
127 = /dev/ttyMX127 128th Moxa port
128 = /dev/moxactl Moxa control port

173 char Moxa Intellio serial card - alternate devices
0 = /dev/cumx0 Callout device for ttyMX0
1 = /dev/cumx1 Callout device for ttyMX1
...
127 = /dev/cumx127 Callout device for ttyMX127

174 char SmartIO serial card
0 = /dev/ttySI0 First SmartIO port
1 = /dev/ttySI1 Second SmartIO port
...

175 char SmartIO serial card - alternate devices
0 = /dev/cusi0 Callout device for ttySI0
1 = /dev/cusi1 Callout device for ttySI1
...

176 char nCipher nFast PCI crypto accelerator
0 = /dev/nfastpci0 First nFast PCI device
1 = /dev/nfastpci1 First nFast PCI device
...

177 char TI PCILynx memory spaces

0 = /dev/pcilynx/aux0 AUX space of first PCILynx card
...
15 = /dev/pcilynx/aux15 AUX space of 16th PCILynx card
16 = /dev/pcilynx/rom0 ROM space of first PCILynx card
...
31 = /dev/pcilynx/rom15 ROM space of 16th PCILynx card
32 = /dev/pcilynx/ram0 RAM space of first PCILynx card
...
47 = /dev/pcilynx/ram15 RAM space of 16th PCILynx card

178 char Giganet cLAN1xxx virtual interface adapter
0 = /dev/clanvi0 First cLAN adapter
1 = /dev/clanvi1 Second cLAN adapter
...

179 char CCube DVXChip-based PCI products
0 = /dev/dvxirq0 First DVX device
1 = /dev/dvxirq1 Second DVX device
...

180 char USB devices
0 = /dev/usb/lp0 First USB printer
...
15 = /dev/usb/lp15 16th USB printer
16 = /dev/usb/mouse0 First USB mouse
...
31 = /dev/usb/mouse15 16th USB mouse
32 = /dev/usb/ez0 First USB firmware loader
...
47 = /dev/usb/ez15 16th USB firmware loader
48 = /dev/usb/scanner0 First USB scanner
...
63 = /dev/usb/scanner15 16th USB scanner
64 = /dev/usb/rio500 Diamond Rio 500

181 char Conrad Electronic parallel port radio clocks
0 = /dev/pcfclock0 First Conrad radio clock
1 = /dev/pcfclock1 Second Conrad radio clock
...

182 char Picture Elements THR2 binarizer
0 = /dev/pethr0 First THR2 board
1 = /dev/pethr1 Second THR2 board
...

183 char SST 5136-DN DeviceNet interface
0 = /dev/ss5136dn0 First DeviceNet interface
1 = /dev/ss5136dn1 Second DeviceNet interface
...

This device used to be assigned to major number 144.
It had to be moved due to an unfortunate conflict.

184 char Picture Elements' video simulator/sender
0 = /dev/pevss0 First sender board
1 = /dev/pevss1 Second sender board
...

185 char InterMezzo high availability file system
0 = /dev/intermezzo0 First cache manager
1 = /dev/intermezzo1 Second cache manager
...

See http://www.inter-mezzo.org/ for more information.

186 char Object-based storage control device
0 = /dev/obd0 First obd control device
1 = /dev/obd1 Second obd control device
...

See ftp://ftp.lustre.org/pub/obd for code and information.

187 char DESkey hardware encryption device
0 = /dev/deskey0 First DES key
1 = /dev/deskey1 Second DES key
...

188 char USB serial converters
0 = /dev/ttyUSB0 First USB serial converter
1 = /dev/ttyUSB1 Second USB serial converter
...

189 char USB serial converters - alternate devices
0 = /dev/cuusb0 Callout device for ttyUSB0
1 = /dev/cuusb1 Callout device for ttyUSB1
...

190 char Kansas City tracker/tuner card
0 = /dev/kctt0 First KCT/T card
1 = /dev/kctt1 Second KCT/T card
...

191 char Reserved for PCMCIA

192 char Kernel profiling interface
0 = /dev/profile Profiling control device
1 = /dev/profile0 Profiling device for CPU 0
2 = /dev/profile1 Profiling device for CPU 1
...

193 char Kernel event-tracing interface
 0 = /dev/trace Tracing control device
 1 = /dev/trace0 Tracing device for CPU 0
 2 = /dev/trace1 Tracing device for CPU 1
 ...

194 char linVideoStreams (LINVS)
 0 = /dev/mvideo/status0 Video compression status
 1 = /dev/mvideo/stream0 Video stream
 2 = /dev/mvideo/frame0 Single compressed frame
 3 = /dev/mvideo/rawframe0 Raw uncompressed frame
 4 = /dev/mvideo/codec0 Direct codec access
 5 = /dev/mvideo/video4linux0 Video4Linux compatibility

 16 = /dev/mvideo/status1 Second device
 ...
 32 = /dev/mvideo/status2 Third device
 ...
 ...
240 = /dev/mvideo/status15 16th device
 ...

195 char Nvidia graphics devices
 0 = /dev/nvidia0 First Nvidia card
 1 = /dev/nvidia1 Second Nvidia card
 ...
255 = /dev/nvidiactl Nvidia card control device

196 char Tormenta T1 card
 0 = /dev/tor/0 Master control channel for all cards
 1 = /dev/tor/1 First DS0
 2 = /dev/tor/2 Second DS0
 ...
 48 = /dev/tor/48 48th DS0
 49 = /dev/tor/49 First pseudo-channel
 50 = /dev/tor/50 Second pseudo-channel
 ...

197 char OpenTNF tracing facility
 0 = /dev/tnf/t0 Trace 0 data extraction
 1 = /dev/tnf/t1 Trace 1 data extraction
 ...
128 = /dev/tnf/status Tracing facility status
130 = /dev/tnf/trace Tracing device

198 char Total Impact TPMP2 quad coprocessor PCI card
 0 = /dev/tpmp2/0 First card
 1 = /dev/tpmp2/1 Second card
 ...

199 char Veritas volume manager (VxVM) volumes
0 = /dev/vx/rdsk/*/* First volume
1 = /dev/vx/rdsk/*/* Second volume
...
block Veritas volume manager (VxVM) volumes
0 = /dev/vx/dsk/*/* First volume
1 = /dev/vx/dsk/*/* First volume
...

The namespace in these directories is maintained by the user space VxVM software.

200 char Veritas VxVM configuration interface
0 = /dev/vx/config Configuration access node
1 = /dev/vx/trace Volume i/o trace access node
2 = /dev/vx/iod Volume i/o daemon access node
3 = /dev/vx/info Volume information access node
4 = /dev/vx/task Volume tasks access node
5 = /dev/vx/taskmon Volume tasks monitor daemon

201 char Veritas VxVM dynamic multipathing driver
0 = /dev/vx/rdmp/* First multipath device
1 = /dev/vx/rdmp/* Second multipath device
...
block Veritas VxVM dynamic multipathing driver
0 = /dev/vx/dmp/* First multipath device
1 = /dev/vx/dmp/* Second multipath device
...

The namespace in these directories is maintained by the user space VxVM software.

202 char CPU model-specific registers
0 = /dev/cpu/0/msr MSRs on CPU 0
1 = /dev/cpu/1/msr MSRs on CPU 1
...

203 char CPU CPUID information
0 = /dev/cpu/0/cpuid CPUID on CPU 0
1 = /dev/cpu/1/cpuid CPUID on CPU 1
...

204 char Low-density serial ports
0 = /dev/ttyLU0 LinkUp Systems L72xx UART - port 0
1 = /dev/ttyLU1 LinkUp Systems L72xx UART - port 1
2 = /dev/ttyLU2 LinkUp Systems L72xx UART - port 2
3 = /dev/ttyLU3 LinkUp Systems L72xx UART - port 3
4 = /dev/ttyFB0 Intel Footbridge (ARM)
5 = /dev/ttySA0 StrongARM builtin serial port 0
6 = /dev/ttySA1 StrongARM builtin serial port 1

7 = /dev/ttySA2	StrongARM builtin serial port 2
8 = /dev/ttySC0	SCI serial port (SuperH) - port 0
9 = /dev/ttySC1	SCI serial port (SuperH) - port 1
10 = /dev/ttySC2	SCI serial port (SuperH) - port 2
11 = /dev/ttySC3	SCI serial port (SuperH) - port 3
12 = /dev/ttyFW0	Firmware console - port 0
13 = /dev/ttyFW1	Firmware console - port 1
14 = /dev/ttyFW2	Firmware console - port 2
15 = /dev/ttyFW3	Firmware console - port 3
16 = /dev/ttyAM0	ARM "AMBA" serial port 0
...	
31 = /dev/ttyAM15	ARM "AMBA" serial port 15
32 = /dev/ttyDB0	DataBooster serial port 0
...	
39 = /dev/ttyDB7	DataBooster serial port 7

205 char Low-density serial ports (alternate device)

0 = /dev/culu0	Callout device for ttyLU0
1 = /dev/culu1	Callout device for ttyLU1
2 = /dev/culu2	Callout device for ttyLU2
3 = /dev/culu3	Callout device for ttyLU3
4 = /dev/cufb0	Callout device for ttyFB0
5 = /dev/cusa0	Callout device for ttySA0
6 = /dev/cusa1	Callout device for ttySA1
7 = /dev/cusa2	Callout device for ttySA2
8 = /dev/cusc0	Callout device for ttySC0
9 = /dev/cusc1	Callout device for ttySC1
10 = /dev/cusc2	Callout device for ttySC2
11 = /dev/cusc3	Callout device for ttySC3
12 = /dev/cufw0	Callout device for ttyFW0
13 = /dev/cufw1	Callout device for ttyFW1
14 = /dev/cufw2	Callout device for ttyFW2
15 = /dev/cufw3	Callout device for ttyFW3
16 = /dev/cuam0	Callout device for ttyAM0
...	
31 = /dev/cuam15	Callout device for ttyAM15
32 = /dev/cudb0	Callout device for ttyDB0
...	
39 = /dev/cudb7	Callout device for ttyDB7

206 char OnStream SC-x0 tape devices

0 = /dev/osst0	First OnStream SCSI tape, mode 0
1 = /dev/osst1	Second OnStream SCSI tape, mode 0
...	
32 = /dev/osst0l	First OnStream SCSI tape, mode 1
33 = /dev/osst1l	Second OnStream SCSI tape, mode 1
...	
64 = /dev/osst0m	First OnStream SCSI tape, mode 2
65 = /dev/osst1m	Second OnStream SCSI tape, mode 2
...	

```
 96 = /dev/osst0a          First OnStream SCSI tape, mode 3
 97 = /dev/osst1a          Second OnStream SCSI tape, mode 3
   ...
128 = /dev/nosst0          No rewind version of /dev/osst0
129 = /dev/nosst1          No rewind version of /dev/osst1
   ...
160 = /dev/nosst0l         No rewind version of /dev/osst0l
161 = /dev/nosst1l         No rewind version of /dev/osst1l
   ...
192 = /dev/nosst0m         No rewind version of /dev/osst0m
193 = /dev/nosst1m         No rewind version of /dev/osst1m
   ...
224 = /dev/nosst0a         No rewind version of /dev/osst0a
225 = /dev/nosst1a         No rewind version of /dev/osst1a
   ...
```

The OnStream SC-x0 SCSI tapes do not support the standard SCSI SASD command set and therefore need their own driver "osst". Note that the IDE, USB (and maybe ParPort) versions may be driven via ide-scsi or usb-storage SCSI emulation and this osst device and driver as well. The ADR-x0 drives are QIC-157 compliant and don't need osst.

207 char Compaq ProLiant health feature indicate

```
 0 = /dev/cpqhealth/cpqw   Redirector interface
 1 = /dev/cpqhealth/crom   EISA CROM
 2 = /dev/cpqhealth/cdt    Data Table
 3 = /dev/cpqhealth/cevt   Event Log
 4 = /dev/cpqhealth/casr   Automatic Server Recovery
 5 = /dev/cpqhealth/cecc   ECC Memory
 6 = /dev/cpqhealth/cmca   Machine Check Architecture
 7 = /dev/cpqhealth/ccsm   Deprecated CDT
 8 = /dev/cpqhealth/cnmi   NMI Handling
 9 = /dev/cpqhealth/css    Sideshow Management
10 = /dev/cpqhealth/cram   CMOS interface
11 = /dev/cpqhealth/cpci   PCI IRQ interface
```

208 char User space serial ports

```
 0 = /dev/ttyU0            First user space serial port
 1 = /dev/ttyU1            Second user space serial port
   ...
```

209 char User space serial ports (alternate devices)

```
 0 = /dev/cuu0             Callout device for ttyU0
 1 = /dev/cuu1             Callout device for ttyU1
   ...
```

210 char SBE, Inc. sync/async serial card

```
 0 = /dev/sbei/wxcfg0      Configuration device for board 0
```

1 = /dev/sbei/dld0 Download device for board 0
2 = /dev/sbei/wan00 WAN device, port 0, board 0
3 = /dev/sbei/wan01 WAN device, port 1, board 0
4 = /dev/sbei/wan02 WAN device, port 2, board 0
5 = /dev/sbei/wan03 WAN device, port 3, board 0
6 = /dev/sbei/wanc00 WAN clone device, port 0, board 0
7 = /dev/sbei/wanc01 WAN clone device, port 1, board 0
8 = /dev/sbei/wanc02 WAN clone device, port 2, board 0
9 = /dev/sbei/wanc03 WAN clone device, port 3, board 0
10 = /dev/sbei/wxcfg1 Configuration device for board 1
11 = /dev/sbei/dld1 Download device for board 1
12 = /dev/sbei/wan10 WAN device, port 0, board 1
13 = /dev/sbei/wan11 WAN device, port 1, board 1
14 = /dev/sbei/wan12 WAN device, port 2, board 1
15 = /dev/sbei/wan13 WAN device, port 3, board 1
16 = /dev/sbei/wanc10 WAN clone device, port 0, board 1
17 = /dev/sbei/wanc11 WAN clone device, port 1, board 1
18 = /dev/sbei/wanc12 WAN clone device, port 2, board 1
19 = /dev/sbei/wanc13 WAN clone device, port 3, board 1
...

Yes, each board is really spaced 10 (decimal) apart.

211 char Addinum CPCI1500 digital I/O card
0 = /dev/addinum/cpci1500/0 First CPCI1500 card
1 = /dev/addinum/cpci1500/1 Second CPCI1500 card
...

216 char USB BlueTooth devices
0 = /dev/ttyUB0 First USB BlueTooth device
1 = /dev/ttyUB1 Second USB BlueTooth device
...

217 char USB BlueTooth devices (alternate devices)
0 = /dev/cuub0 Callout device for ttyUB0
1 = /dev/cuub1 Callout device for ttyUB1
...

218 char The Logical Company bus Unibus/Qbus adapters
0 = /dev/logicalco/bci/0 First bus adapter
1 = /dev/logicalco/bci/1 First bus adapter
...

219 char The Logical Company DCI-1300 digital I/O card
0 = /dev/logicalco/dci1300/0 First DCI-1300 card
1 = /dev/logicalco/dci1300/1 Second DCI-1300 card
...

220 char Myricom Myrinet "GM" board
0 = /dev/myricom/gm0 First Myrinet GM board

1 = /dev/myricom/gmp0 First board "root access"
2 = /dev/myricom/gm1 Second Myrinet GM board
3 = /dev/myricom/gmp1 Second board "root access"
...

221 char VME bus
0 = /dev/bus/vme/m0 First master image
1 = /dev/bus/vme/m1 Second master image
2 = /dev/bus/vme/m2 Third master image
3 = /dev/bus/vme/m3 Fourth master image
4 = /dev/bus/vme/s0 First slave image
5 = /dev/bus/vme/s1 Second slave image
6 = /dev/bus/vme/s2 Third slave image
7 = /dev/bus/vme/s3 Fourth slave image
8 = /dev/bus/vme/ctl Control

It is expected that all VME bus drivers will use the same interface. For interface documentation see http://www.vmelinux.org/.

224 char A2232 serial card
0 = /dev/ttyY0 First A2232 port
1 = /dev/ttyY1 Second A2232 port
...

225 char A2232 serial card (alternate devices)
0 = /dev/cuy0 Callout device for ttyY0
1 = /dev/cuy1 Callout device for ttyY1
...

226 char Direct Rendering Infrastructure (DRI)
0 = /dev/dri/card0 First graphics card
1 = /dev/dri/card1 Second graphics card
...

227 char IBM 3270 terminal block-mode access
0 = /dev/3270/tub Controlling interface
1 = /dev/3270/tub1 First 3270 terminal
2 = /dev/3270/tub2 Second 3270 terminal
...

228 char IBM 3270 terminal Unix tty access
1 = /dev/3270/tty1 First 3270 terminal
2 = /dev/3270/tty2 Seconds 3270 terminal
...

229 char IBM iSeries virtual console
0 = /dev/iseries/vtty0 First console port
1 = /dev/iseries/vtty1 Second console port
...

230 char IBM iSeries virtual tape

0 = /dev/iseries/vt0	First virtual tape, mode 0
1 = /dev/iseries/vt1	Second virtual tape, mode 0
...	
32 = /dev/iseries/vt0l	First virtual tape, mode 1
33 = /dev/iseries/vt1l	Second virtual tape, mode 1
...	
64 = /dev/iseries/vt0m	First virtual tape, mode 2
65 = /dev/iseries/vt1m	Second virtual tape, mode 2
...	
96 = /dev/iseries/vt0a	First virtual tape, mode 3
97 = /dev/iseries/vt1a	Second virtual tape, mode 3
...	
128 = /dev/iseries/nvt0	First virtual tape, mode 0, no rewind
129 = /dev/iseries/nvt1	Second virtual tape, mode 0, no rewind
...	
160 = /dev/iseries/nvt0l	First virtual tape, mode 1, no rewind
161 = /dev/iseries/nvt1l	Second virtual tape, mode 1, no rewind
...	
192 = /dev/iseries/nvt0m	First virtual tape, mode 2, no rewind
193 = /dev/iseries/nvt1m	Second virtual tape, mode 2, no rewind
...	
224 = /dev/iseries/nvt0a	First virtual tape, mode 3, no rewind
225 = /dev/iseries/nvt1a	Second virtual tape, mode 3, no rewind
...	

"No rewind" refers to the omission of the default automatic rewind on device close. The MTREW or MTOFFL ioctl()'s can be used to rewind the tape regardless of the device used to access it.

231-239 UNASSIGNED

240-254 LOCAL/EXPERIMENTAL USE

255 RESERVED

This major is reserved to assist the expansion to a larger number space. No device nodes with this major should ever be created on the filesystem.

**** ADDITIONAL /dev DIRECTORY ENTRIES

This section details additional entries that should or may exist in the /dev directory. It is preferred that symbolic links use the same form (absolute or relative) as is indicated here. Links are classified as "hard" or "symbolic" depending on the preferred type of link; if possible, the indicated type of link should be used.

Compulsory links

These links should exist on all systems:

/dev/fd	/proc/self/fd	symbolic	File descriptors
/dev/stdin	fd/0	symbolic	stdin file descriptor
/dev/stdout	fd/1	symbolic	stdout file descriptor
/dev/stderr	fd/2	symbolic	stderr file descriptor
/dev/nfsd	socksys	symbolic	Required by iBCS-2
/dev/X0R	null	symbolic	Required by iBCS-2

Note: /dev/X0R is --.

Recommended links

It is recommended that these links exist on all systems:

/dev/core	/proc/kcore	symbolic	Backward compatibility
/dev/ramdisk	ram0	symbolic	Backward compatibility
/dev/ftape	qft0	symbolic	Backward compatibility
/dev/bttv0	video0	symbolic	Backward compatibility
/dev/radio	radio0	symbolic	Backward compatibility
/dev/i2o*	/dev/i2o/*	symbolic	Backward compatibility
/dev/scd?	sr?	hard	Alternate SCSI CD-ROM name

Locally defined links

The following links may be established locally to conform to the configuration of the system. This is merely a tabulation of existing practice, and does not constitute a recommendation. However, if they exist, they should have the following uses.

/dev/mouse	mouse port	symbolic	Current mouse device
/dev/tape	tape device	symbolic	Current tape device
/dev/cdrom	CD-ROM device	symbolic	Current CD-ROM device
/dev/cdwriter	CD-writer	symbolic	Current CD-writer device
/dev/scanner	scanner	symbolic	Current scanner device
/dev/modem	modem port	symbolic	Current dialout device
/dev/root	root device	symbolic	Current root filesystem
/dev/swap	swap device	symbolic	Current swap device

/dev/modem should not be used for a modem which supports dialin as well as dialout, as it tends to cause lock file problems. If it exists, /dev/modem should point to the appropriate primary TTY device (the use of the alternate callout devices is deprecated).

For SCSI devices, /dev/tape and /dev/cdrom should point to the ``cooked" devices (/dev/st* and /dev/sr*, respectively), whereas /dev/cdwriter and /dev/scanner should point to the appropriate generic

SCSI devices (/dev/sg*).

/dev/mouse may point to a primary serial TTY device, a hardware mouse device, or a socket for a mouse driver program (e.g. /dev/gpmdata).

Sockets and pipes

Non-transient sockets and named pipes may exist in /dev. Common entries are:

/dev/printer	socket	lpd local socket
/dev/log	socket	syslog local socket
/dev/gpmdata	socket	gpm mouse multiplexer

Mount points

The following names are reserved for mounting special filesystems under /dev. These special filesystems provide kernel interfaces that cannot be provided with standard device nodes.

/dev/pts	devpts	PTY slave filesystem
/dev/shm	shmfs	POSIX shared memory maintenance access

**** TERMINAL DEVICES

Terminal, or TTY devices are a special class of character devices. A terminal device is any device that could act as a controlling terminal for a session; this includes virtual consoles, serial ports, and pseudoterminals (PTYs).

All terminal devices share a common set of capabilities known as line diciplines; these include the common terminal line dicipline as well as SLIP and PPP modes.

All terminal devices are named similarly; this section explains the naming and use of the various types of TTYs. Note that the naming conventions include several historical warts; some of these are Linux-specific, some were inherited from other systems, and some reflect Linux outgrowing a borrowed convention.

A hash mark (#) in a device name is used here to indicate a decimal number without leading zeroes.

Virtual consoles and the console device

Virtual consoles are full-screen terminal displays on the system video monitor. Virtual consoles are named /dev/tty#, with numbering starting at /dev/tty1; /dev/tty0 is the current virtual console. /dev/tty0 is the device that should be used to access the system video card on those architectures for which the frame buffer devices (/dev/fb*) are not applicable. Do not use /dev/console

for this purpose.

The console device, /dev/console, is the device to which system messages should be sent, and on which logins should be permitted in single-user mode. Starting with Linux 2.1.71, /dev/console is managed by the kernel; for previous versions it should be a symbolic link to either /dev/tty0, a specific virtual console such as /dev/tty1, or to a serial port primary (tty*, not cu*) device, depending on the configuration of the system.

Serial ports

Serial ports are RS-232 serial ports and any device which simulates one, either in hardware (such as internal modems) or in software (such as the ISDN driver.) Under Linux, each serial ports has two device names, the primary or callin device and the alternate or callout one. Each kind of device is indicated by a different letter. For any letter X, the names of the devices are /dev/ttyX# and /dev/cux#, respectively; for historical reasons, /dev/ttyS# and /dev/ttyC# correspond to /dev/cua# and /dev/cub#. In the future, it should be expected that multiple letters will be used; all letters will be upper case for the "tty" device (e.g. /dev/ttyDP#) and lower case for the "cu" device (e.g. /dev/cudp#).

The names /dev/ttyQ# and /dev/cuq# are reserved for local use.

The alternate devices provide for kernel-based exclusion and somewhat different defaults than the primary devices. Their main purpose is to allow the use of serial ports with programs with no inherent or broken support for serial ports. Their use is deprecated, and they may be removed from a future version of Linux.

Arbitration of serial ports is provided by the use of lock files with the names /var/lock/LCK..ttyX#. The contents of the lock file should be the PID of the locking process as an ASCII number.

It is common practice to install links such as /dev/modem which point to serial ports. In order to ensure proper locking in the presence of these links, it is recommended that software chase symlinks and lock all possible names; additionally, it is recommended that a lock file be installed with the corresponding alternate device. In order to avoid deadlocks, it is recommended that the locks are acquired in the following order, and released in the reverse:

1. The symbolic link name, if any (/var/lock/LCK..modem)
2. The "tty" name (/var/lock/LCK..ttyS2)
3. The alternate device name (/var/lock/LCK..cua2)

In the case of nested symbolic links, the lock files should be installed in the order the symlinks are resolved.

Under no circumstances should an application hold a lock while waiting for another to be released. In addition, applications which attempt to create lock files for the corresponding alternate device names should take into account the possibility of being used on a non-serial port TTY, for which no alternate device would exist.

Pseudoterminals (PTYs)

Pseudoterminals, or PTYs, are used to create login sessions or provide other capabilities requiring a TTY line dicipline (including SLIP or PPP capability) to arbitrary data-generation processes. Each PTY has a master side, named /dev/pty[p-za-e][0-9a-f], and a slave side, named /dev/tty[p-za-e][0-9a-f]. The kernel arbitrates the use of PTYs by allowing each master side to be opened only once.

Once the master side has been opened, the corresponding slave device can be used in the same manner as any TTY device. The master and slave devices are connected by the kernel, generating the equivalent of a bidirectional pipe with TTY capabilities.

Recent versions of the Linux kernels and GNU libc contain support for the System V/Unix98 naming scheme for PTYs, which assigns a common device, /dev/ptmx, to all the masters (opening it will automatically give you a previously unassigned PTY) and a subdirectory, /dev/pts, for the slaves; the slaves are named with decimal integers (/dev/pts/# in our notation). This removes the problem of exhausting the namespace and enables the kernel to automatically create the device nodes for the slaves on demand using the "devpts" filesystem.

10.3 Afficheur LCD

10.3.1 Le port parallèle

Le port parallèle d'un Pc actuel cumule les appellations EPP et ECP. Ce port ayant avant tout été pensé pour communiquer avec une imprimante, la plupart de ses signaux ont un rapport avec ce périphérique. Il possède 17 broches utilisables, ainsi que 8 broches reliées à la masse.

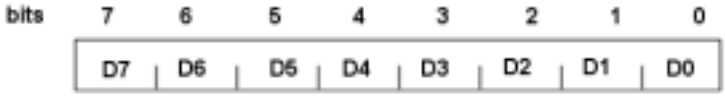
Pin	Nom		Pin	Nom	
1	/STROBE	I/O	14	/AUTOFEED	I/O
2	D0	I/O	15	/ERROR	I
3	D1	I/O	16	/INIT	I/O
4	D2	I/O	17	/SELECT IN	I/O
5	D3	I/O	18	GROUND	
6	D4	I/O	19	GROUND	
7	D5	I/O	20	GROUND	
8	D6	I/O	21	GROUND	
9	D7	I/O	22	GROUND	
10	/ACK	I	23	GROUND	
11	BUSY	I	24	GROUND	
12	PE	I	25	GROUND	
13	SELECT	I			

Tableau 7 : Port parallèle

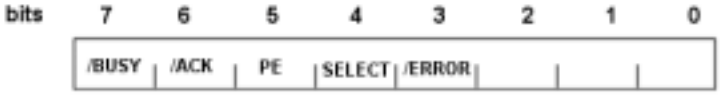
STROBE: cette ligne active basse indique à l'imprimante que des données sont présentes sur les lignes D0 à D7 et qu'il faut les prendre en compte.
D0 à D7: c'est le bus de données sur lequel véhicule la valeur du caractère à imprimer. On ne peut écrire sur ce port, à moins d'avoir un port parallèle étendu (c'est le cas pour les ports de type ECP/EPP).
/ACK: l'imprimante met à 0 cette ligne pour indiquer à l'ordinateur qu'elle a bien reçu le caractère transmis et que ce dernier peut continuer la transmission.
BUSY: cette ligne est mise à 0 par l'imprimante lorsque son buffer de réception est plein. L'ordinateur est ainsi averti que celle-ci ne peut plus recevoir de données. Il doit attendre que cette ligne revienne à 1 pour recommencer à émettre.
PE: signifie " paper error ". L'imprimante indique par cette ligne à l'ordinateur que l'alimentation en papier a été interrompue.
SELECT: cette ligne indique à l'ordinateur si l'imprimante est "on line" ou "off line".
/AUTOFEED: lorsque ce signal est à 1, l'imprimante doit effectuer un saut de ligne à chaque caractère "return" reçu. En effet, certaines imprimantes se contentent d'effectuer un simple retour du chariot en présence de ce caractère.
/ERROR: indique à l'ordinateur que l'imprimante a détecté une erreur.
/INIT: l'ordinateur peut effectuer une initialISAtion de l'imprimante par l'intermédiaire de cette ligne.
/SELECT IN: l'ordinateur peut mettre l'imprimante hors ligne par l'intermédiaire de ce signal.

Pour utiliser le port parallèle, il faut accéder à 3 registres de 8 bits situés aux adresses 0x378h, 0x379h et 0x37Ah.

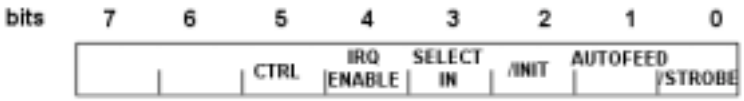
- **0x378h** :L'écriture d'un octet dans ce registre place immédiatement les lignes D0-D7 aux niveaux inscrit. Il peut également être lu.



- **0x379h :** Ce registre, accessible uniquement en lecture, est l'image de l'état des lignes ERROR, SELECT, PE, ACK et BUSY



- **0x37Ah :** Ce dernier registre est accessible à la fois en lecture et en écriture. Les 4 premiers bits sont l'image des lignes STROBE, AUTOFEED, INIT et SELECT IN. Le bit 4 autorise ou non le déclenchement d'une interruption lorsque ACK passe à l'état bas (c'est à dire quand l'imprimante valide la réception d'un caractère).En mettant le bit CTRL à 1, le port est accessible en écriture.



10.3.2 LCD HD44780

10.3.3 Programme pour LCD HD44780

Fichier main.c

```
/***************************************************************************
                          main.c  -  description
                             -------------------
    begin                : Thu Dec 13 19:51:23 CET 2001
    copyright            : (C) 2001 by

***************************************************************************/



#ifdef HAVE_CONFIG_H
#include <config.h>

#include "LCD_function.h"
#include "utilities.h"


int main(int argc, char *argv[])
{
/*Define variables                                                  */
  int count, counttest;
  char s[16];
  int ret;

/*ask for permission                                                */
  printf("ELISE Embedded Linux Setup by GF\n");
  if(ioperm(PORTADDRESS,3,1))
/*ask the permission to use lpt*/
  {
        perror("ioperm");
        exit(1);
  }
/*Initialisation of the LCD                                         */
  initlcd();
  cursor(0,0);
  clearscreen();
  movelcd(0x00);
```

```
  sendlcd("Loading");
  movelcd(0x40);
  sendlcd("ELISE Linux");
  usleep(2000000);
  clearscreen();
  cursor(0,0);
  movelcd(0x00);
  sendlcd("Demo Text by ");
  movelcd(0x40);
  sendlcd("Gaspoz Fred");
  usleep(2000000);
  clearscreen();
  cursor(0,0);

  /* Display the date and the time               */
  givetime();

  /* exit the application                        */
  return EXIT_SUCCESS;
}
```

Fichier utilities.h

```
/***************************************************************
                     utilities.h  -  description
                        -------------------
    begin                : Mon Dec 17 2001
    copyright            : (C) 2001 by
    email                :
***************************************************************/

#ifndef HAVE_LCD_UTILITY
#define HAVE_LCD_UTILITY


#include "LCD_function.h"

void givetime(void);
void readkey(void);
#endif
```

Fichier utilities.c

```
/***************************************************************************
                      utilities.c  -  description
                          -------------------
    begin                : Mon Dec 17 2001
    copyright            : (C) 2001 by
    email                :
***************************************************************************/

#include "utilities.h"



/***************************************************************************
 *   Function givetime                                                     *
 *                                                                         *
 *   send ste to LCD 16x2                                                  *
 ***************************************************************************/

void givetime(void)
{
  time_t intps;
  int count;
  char date[16];
  char clock[16];
  char all[26];
  intps = time(NULL);

  /*initialise date variable*/
  for (count = 0; count < 16; count++)
  {
        date[count] = NULL;
  }
  /*initialise clock variable*/
  for (count = 0; count < 16; count++)
  {
        clock[count] = '-';
  }
```

```
  /*loop*/
  while(1){
  strcpy(all, ctime(&intps));
  for (count = 0; count < 4; count++)
  {
        date[count] = all[count];
  }

  for (count = 0; count < 3; count++)
  {
        date[4+count] = all[8+count];
  }

  for (count = 0; count < 4; count++)
  {
        date[7+count] = all[4+count];
  }

  for (count = 0; count < 4; count++)
  {
        date[11+count] = all[20+count];
  }
  strcat(date, "\0");
  movelcd(0x00);
  sendlcd(date);
  movelcd(0x40);


    intps = time(NULL);
        strcpy(all, ctime(&intps));
  for (count = 0; count < 8; count++)
        {
              clock[4+count] = all[11+count];
        }
  strcat(clock, "\0");
        movelcd(0x40);
    sendlcd(clock);

  }
}
```

Fichier LCD_function.h

```
/***************************************************************************
                      LCD_function.h  -  description
                          -------------------
    begin                : Thu Dec 13 2001
    copyright            : (C) 2001 by
    email                :
***************************************************************************/

#ifndef HAVE_LCD_FUNCTION
#define HAVE_LCD_FUNCTION
#define PORTADDRESS 0x378       /*parellele port adress*/
#define DATA PORTADDRESS+0      /*data port*/
#define STATUS PORTADDRESS+1    /*status port*/
#define CONTROL PORTADDRESS+2   /*control port*/

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <asm/io.h>
#include <time.h>


void initlcd(void);
void sendlcd(char text[16]);
void movelcd(int position);
int busylcd(void);
void clearscreen(void);
void cursor(int onoff, int blink);
void waittime(void);
#endif
```

Fichier LCD_function.c

```
/*********************************************************************
 *   Function waittime                                               *
 *                                                                   *
 *   wait for valide data                                            *
 *********************************************************************/
void waittime(void)
{
  int temp;

  for (temp = 0; temp <= 7500; temp++)
  {}

}

/*********************************************************************
 *   Function clearscreen                                            *
 *                                                                   *
 *   clear screen                                                     *
 *********************************************************************/
void clearscreen(void)
{
  outb(inb(CONTROL) | 0x08, CONTROL);      /*command mode*/
  outb(0x01, DATA);
  outb(inb(CONTROL) | 0x01, CONTROL);
  waittime();
  outb(inb(CONTROL) & 0xFE, CONTROL);
  //*waittime();
  outb(inb(CONTROL) & 0xF7, CONTROL);      /*stop command mode*/
}

/*********************************************************************
 *   Function cursor                                                 *
 *                                                                   *
 *   set cursor => cursor on/off   1/0                               *
 *                 blink on/off    1/0                               *
 *                                                                   *
 *********************************************************************/
void cursor(int onoff, int blink)
```

```
{
  int data_control;
  data_control = 0x0C + (onoff*2) + blink;
  outb(inb(CONTROL) | 0x08, CONTROL);      /*command mode*/
  outb(data_control, DATA);
  outb(inb(CONTROL) | 0x01, CONTROL);
  usleep(2);
  outb(inb(CONTROL) & 0xFE, CONTROL);
  usleep(2);
  outb(inb(CONTROL) & 0xF7, CONTROL);      /*stop command mode*/

}

/*********************************************************************
 *   Function Init                                                   *
 *                                                                   *
 *   Initialisation of the LCD                                       *
 *********************************************************************/
void initlcd(void)
{

  char init[10];               /*array of initialization*/
  int count;                   /*variable for loop*/

  init[0] = 0x0F;              /*reset LCD*/
  init[1] = 0x01;              /*clear LCD*/
  init[2] = 0x38;              /*8bits mode+2 lines*/

  printf("Initialize LCD \n");

  outb(inb(CONTROL) & 0xDF, CONTROL);      /*port control init*/
  outb(inb(CONTROL) | 0x08, CONTROL); /*command mode with pin 17 */

/*loop to send data on command register*/
  for (count = 0; count <= 2; count++)
        {
        outb(init[count], DATA);           /*send on data*/
        /*validate strobe=enable*/
        outb(inb(CONTROL) | 0x01, CONTROL);
```

```
        usleep(1);                          /*1ms pause*/
    outb(inb(CONTROL) & 0xFE, CONTROL);
  }

  outb(inb(CONTROL) & 0xF7, CONTROL);       /*stop command mode*/

}


/*********************************************************************
 *   Function sendlcd                                                *
 *                                                                   *
 *   send text to LCD                                                *
 *********************************************************************/
void sendlcd(char text[16])
{
  int len;
  int count;

  len = strlen(text);                       /*give length of text*/

  for (count = 0; count < len; count++)
  {
        outb(text[count], DATA);            /*send one char*/
        outb(inb(CONTROL) | 0x01, CONTROL); /*validation*/
        waittime();                         /*wait 1ms*/
        outb(inb(CONTROL) & 0xFE, CONTROL);
  }
}

/*********************************************************************
 *   Function movelcd                                                *
 *                                                                   *
 *   move cursor in LCD                                              *
 *********************************************************************/
void movelcd(int position)
{
  outb(inb(CONTROL) | 0x08, CONTROL);       /*command mode*/
  outb(128+position, DATA);                 /*set position 128=0x80*/
```

```
  outb(inb(CONTROL) | 0x01, CONTROL);
  usleep(2);
  outb(inb(CONTROL) & 0xFE, CONTROL);
  usleep(2);
  outb(inb(CONTROL) & 0xF7, CONTROL);      /*stop command mode*/
}
```

10.4 Codes sources et scripts utilisés

10.4.1 Script de création des fichiers périphériques

```
#!/bin/sh

#########################################################################
#Creates device for ELISE Embedded Linux Setup by GF
#########################################################################

# we do our work in the dev/ directory
if [ -z "$1" ]; then
  echo "usage: `basename $0` path/to/dev/dir"
  exit 1
fi

cd $1

#########################################################################
#Define the permission's variable
#########################################################################
  public=666
 private=600
  system=660
    kmem=640
     tty=666
    cons=622
     vcs=600
 dialout=660
   mouse=660
 printer=660
  floppy=660
    disk=660
    scsi=660
   cdrom=660
    tape=660
   audio=660
   video=660
   ibcs2=666
 scanner=666
```

```
    coda=600
   ipsec=200
readable=444


#########################################################################
#standarddevices
#########################################################################
mknod -m $cons console c 5 1
mknod -m $public full c 1 7
mknod -m $kmem kmem c 1 2
mknod -m $kmem mem c 1 1
mknod -m $public null c 1 3
mknod -m $kmem port c 1 4
mknod -m $readable random c 1 8
mknod -m $readable urandom c 1 9
mknod -m $readable aio c 1 10
mknod -m $public zero c 1 5
ln -s /proc/kcore core



#########################################################################
#IDE HD devs
#########################################################################

 # hda devs
 mknod -m $disk hda b 3 0
 for i in `seq 1 8`; do
   mknod -m $disk hda$i b 3 $i
 done


 # hdb devs
 mknod -m $disk hdb b 3 60
 for i in `seq 1 7`; do
   mknod -m $disk hdb$i b 3 `expr $i + 60`
 done

 # hdc devs
```

```
 mknod -m $disk hdc b 22 0
 for i in `seq 1 7`; do
   mknod -m $disk hdc$i b 22 $i
 done

 # hdd devs
 mknod -m $disk hdd b 22 60
 for i in `seq 1 7`; do
   mknod -m $disk hdd$i b 22 `expr $i + 60`
 done

#############################################################################
# loop devs
#############################################################################
for i in `seq 0 7`; do
  mknod -m $disk loop$i b 7 $i
done

#############################################################################
# ram devs
#############################################################################
for i in `seq 0 9`; do
  mknod -m $disk ram$i b 1 $i
done
ln -s ram1 ram

#############################################################################
# ttys
#############################################################################
mknod tty c 5 0
for i in `seq 0 9`; do
  mknod -m $tty tty$i c 4 $i
done

#############################################################################
#Serial port
#############################################################################
mknod -m $dialout ttyS0 c 4 64
mknod -m $dialout ttyS1 c 4 65
```

```
mknod -m $dialout cua1 c 5 64
mknod -m $dialout cua2 c 5 65

#############################################################################
# virtual console screen devs
#############################################################################
for i in `seq 0 9`; do
  mknod -m $vcs vcs$i b 7 $i
done
ln -s vcs0 vcs

#############################################################################
# virtual console screen w/ attributes devs
#############################################################################
for i in `seq 0 9`; do
  mknod -m $vcs vcsa$i b 7 $i
done

ln -s vcsa0 vcsa
```

10.4.2 Fichier protocols

```
#
# protocols   This file describes the various protocols that are
#       available from the TCP/IP subsystem.  It should be
#       consulted instead of using the numbers in the ARPA
#       include files, or, worse, just guessing them.
#

ip      0     IP          # internet protocol v4
icmp          1     ICMP        # internet control message protocol
igmp          2     IGMP        # internet group multicast protocol
ggp           3     GGP         # gateway-gateway protocol
tcp           6     TCP         # transmission control protocol
egp           8     EGP         # exterior gateway protocol
pup           12    PUP         # PARC universal packet protocol
udp           17    UDP         # user datagram protocol
hmp           20    HMP         # host monitoring protocol
idp           22    IDP         # WhatsThis?
rdp           27    RDP         # "reliable datagram" protocol
iso-tp4       29    ISO-TP4     # ISO Transport Protocol Class4
#
# Internet (IPv6) extension headers
#
hopopt        0     HOPOPT            # Hop-by-hop optons for IPv6
ipv6          41    IPv6        # IPv6
ipv6-route    43    IPv6-Route  # Routing Header for IPv6
ipv6-frag     44    IPv6-Frag   # Fragment Header for IPv6
ipv6-crypt    50    IPv6-Crypt  # Encryption Header for IPv6
ipv6-auth     51    IPv6-Auth   # Authentication Header for IPv6
icmpv6        58    IPv6-ICMP ICMPV6 ipv6-icmp icmp6 # ICMP for IPv6
ipv6-nonxt    59    IPv6-NoNxt  # No Next Header for IPv6
ipv6-opts     60    IPv6-Opts   # Destination Options for IPv6
iso-ip        80    ISO-IP            # ISO Internet Protocol
encap         98    ENCAP       # RFC1241 encapsulation
raw           255   RAW         # RAW IP interface

# End.
```

10.4.3 Fichier de configuration de BOA

```
# Boa v0.94 configuration file
# File format has not changed from 0.93
# File format has changed little from 0.92
# version changes are noted in the comments
#
# The Boa configuration file is parsed with a lex/yacc or flex/bison
# generated parser.  If it reports an error, the line number will be
# provided; it should be easy to spot.  The syntax of each of these
# rules is very simple, and they can occur in any order.  Where possible
# these directives mimic those of NCSA httpd 1.3; I saw no reason to
# introduce gratuitous differences.

# $Id: boa.conf,v 1.23 2001/09/25 03:28:30 jnelson Exp $

# The "ServerRoot" is not in this configuration file.  It can be compiled
# into the server (see defines.h) or specified on the command line with
# the -c option, for example:
#
# boa -c /usr/local/boa



# Port: The port Boa runs on.  The default port for http servers is 80.
# If it is less than 1024, the server must be started as root.

Port 80


# Listen: the Internet address to bind(2) to.  If you leave it out,
# it takes the behavior before 0.93.17.2, which is to bind to all
# addresses (INADDR_ANY).  You only get one "Listen" directive,
# if you want service on multiple IP addresses, you have three choices:
#    1. Run boa without a "Listen" directive
#         a. All addresses are treated the same; makes sense if the addresses
#          are localhost, ppp, and eth0.
```

```
#        b. Use the VirtualHost directive below to point requests to different
#              files.   Should be good for a very large number of addresses (web
#          hosting clients).
#      2. Run one copy of boa per IP address, each has its own configuration
#        with a "Listen" directive.  No big deal up to a few tens of addresses.
#       Nice separation between clients.
# The name you provide gets run through inet_aton(3), so you have to use dotted
# quad notation.  This configuration is too important to trust some DNS.


#Listen 192.68.0.5


#  User: The name or UID the server should run as.
# Group: The group name or GID the server should run as.


User root
Group root


# ServerAdmin: The email address where server problems should be sent.
# Note: this is not currently used, except as an environment variable
# for CGIs.


#ServerAdmin root@localhost


# ErrorLog: The location of the error log file. If this does not start
# with /, it is considered relative to the server root.
# Set to /dev/null if you don't want errors logged.
# If unset, defaults to /dev/stderr


ErrorLog /var/log/boa/error_log
# Please NOTE: Sending the logs to a pipe ('|'), as shown below,
#  is somewhat experimental and might fail under heavy load.
# "Usual libc implementations of printf will stall the whole
#  process if the receiving end of a pipe stops reading."
#ErrorLog    "|/usr/sbin/cronolog    --symlink=/var/log/boa/error_log /var/log/boa/error-%Y%m%d.log"
```

```
# AccessLog: The location of the access log file. If this does not
# start with /, it is considered relative to the server root.
# Comment out or set to /dev/null (less effective) to disable
# Access logging.

AccessLog /var/log/boa/access_log
# Please NOTE: Sending the logs to a pipe ('|'), as shown below,
#  is somewhat experimental and might fail under heavy load.
# "Usual libc implementations of printf will stall the whole
#  process if the receiving end of a pipe stops reading."
#AccessLog    "|/usr/sbin/cronolog  --symlink=/var/log/boa/access_log
/var/log/boa/access-%Y%m%d.log"

# UseLocaltime: Logical switch.  Uncomment to use localtime
# instead of UTC time
#UseLocaltime

# VerboseCGILogs: this is just a logical switch.
#  It simply notes the start and stop times of cgis in the error log
# Comment out to disable.

#VerboseCGILogs

# ServerName: the name of this server that should be sent back to
#  clients  if  different  than  that  returned  by  gethostname  +
gethostbyname

#ServerName www.your.org.here

# VirtualHost: a logical switch.
# Comment out to disable.
# Given DocumentRoot /var/www, requests on interface 'A' or IP 'IP-A'
# become /var/www/IP-A.
# Example: http://localhost/ becomes /var/www/127.0.0.1
#
#  Not  used  until  version  0.93.17.2.   This  "feature"  also  breaks
commonlog
# output rules, it prepends the interface number to each access_log
line.
# You are expected to fix that problem with a postprocessing script.
```

```
#VirtualHost

# DocumentRoot: The root directory of the HTML documents.
# Comment out to disable server non user files.

DocumentRoot /var/www

# UserDir: The name of the directory which is appended onto a user's
home
# directory if a ~user request is recieved.

UserDir public_html

# DirectoryIndex: Name of the file to use as a pre-written HTML
# directory index.  Please MAKE AND USE THESE FILES.  On the
# fly creation of directory indexes can be _slow_.
# Comment out to always use DirectoryMaker

DirectoryIndex index.html

# DirectoryMaker: Name of program used to create a directory listing.
# Comment out to disable directory listings.  If both this and
# DirectoryIndex are commented out, accessing a directory will give
# an error (though accessing files in the directory are still ok).

DirectoryMaker /usr/lib/boa/boa_indexer

# DirectoryCache: If DirectoryIndex doesn't exist, and DirectoryMaker
# has been commented out, the the on-the-fly indexing of Boa can be
used
# to generate indexes of directories. Be warned that the output is
# extremely minimal and can cause delays when slow disks are used.
# Note: The DirectoryCache must be writable by the same user/group
that
# Boa runs as.

# DirectoryCache /var/spool/boa/dircache

# KeepAliveMax: Number of KeepAlive requests to allow per connection
# Comment out, or set to 0 to disable keepalive processing
```

```
KeepAliveMax 1000

# KeepAliveTimeout: seconds to wait before keepalive connection times
out

KeepAliveTimeout 10

# MimeTypes: This is the file that is used to generate mime type
pairs
# and Content-Type fields for boa.
# Set to /dev/null if you do not want to load a mime types file.
# Do *not* comment out (better use AddType!)

MimeTypes /etc/mime.types

# DefaultType: MIME type used if the file extension is unknown, or
there
# is no file extension.

DefaultType text/plain

# AddType: adds types without editing mime.types
# Example: AddType type extension [extension ...]

# Uncomment the next line if you want .cgi files to execute from
anywhere
#AddType application/x-httpd-cgi cgi

# Redirect, Alias, and ScriptAlias all have the same semantics --
they
# match the beginning of a request and take appropriate action.  Use
# Redirect for other servers, Alias for the same server, and
ScriptAlias
# to enable directories for script execution.

# Redirect allows you to tell clients about documents which used to
exist in
# your server's namespace, but do not anymore. This allows you to
tell the
# clients where to look for the relocated document.
# Example: Redirect /bar http://elsewhere/feh/bar
```

```
# Aliases: Aliases one path to another.
# Example: Alias /path1/bar /path2/foo

Alias /doc /usr/doc

# ScriptAlias: Maps a virtual path to a directory for serving scripts
# Example: ScriptAlias /htbin/ /www/htbin/

ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
```

10.4.4 Script BZEXE

```
#!/bin/sh
# bzexe: compressor for Unix executables.
# Use this only for binaries that you do not use frequently.
# The compressed version is a shell script which decompresses itself after
# skipping $skip lines of shell commands.  We try invoking the compressed
# executable with the original name (for programs looking at their name).
# We also try to retain the original file permissions on the compressed file.
# For safety reasons, bzexe will not create setuid or setgid shell scripts.
# WARNING: the first line of this file must be either : or #!/bin/sh
# The : is required for some old versions of csh.
# On Ultrix, /bin/sh is too buggy, change the first line to: #!/bin/sh5
# Simply made from gzexe by Sebastien HUET
# Uses bzip2 instead of gzip. V1.0 march 30 1999
# Idea Guillaume Schmid

x=`basename $0`
if test $# = 0; then
  echo compress executables. original file foo is renamed to foo~
  echo usage: ${x} [-d] files...
  echo   "   -d  decompress the executables"
  exit 1
fi

tmp=bz$$
trap "rm -f $tmp; exit 1" 1 2 3 5 10 13 15

decomp=0
res=0
test "$x" = "unbzexe" && decomp=1
if test "x$1" = "x-d"; then
  decomp=1
  shift
```

```
fi

echo hi > zfoo1$$
echo hi > zfoo2$$
if test -z "`(${CPMOD-cpmod} zfoo1$$ zfoo2$$) 2>&1`"; then
  cpmod=${CPMOD-cpmod}
fi
rm -f zfoo[12]$$

tail=""
IFS="${IFS=   }"; saveifs="$IFS"; IFS="${IFS}:"
for dir in $PATH; do
  test -z "$dir" && dir=.
  if test -f $dir/tail; then
    tail="$dir/tail"
    break
  fi
done
IFS="$saveifs"
if test -z "$tail"; then
  echo cannot find tail
  exit 1
fi

for i do
  if test ! -f "$i" ; then
    echo ${x}: $i not a file
    res=1
    continue
  fi
  if test $decomp -eq 0; then
    if sed -e 1d -e 2q "$i" | grep "^skip=[0-9]*$" >/dev/null; then
      echo "${x}: $i is already bzexe'd"
      continue
    fi
  fi
  if ls -l "$i" | grep '^...[sS]' > /dev/null; then
    echo "${x}: $i has setuid permission, unchanged"
    continue
  fi
```

```
  if ls -l "$i" | grep '^......[sS]' > /dev/null; then
    echo "${x}: $i has setgid permission, unchanged"
    continue
  fi
  case "`basename $i`" in
  bzip2 | tail | chmod | ln | sleep | rm)
  echo "${x}: $i would depend on itself"; continue ;;
  esac
  if test -z "$cpmod"; then
    cp -p "$i" $tmp 2>/dev/null || cp "$i" $tmp
    if test -w $tmp 2>/dev/null; then
      writable=1
    else
      writable=0
      chmod u+w $tmp 2>/dev/null
    fi
  fi
  if test $decomp -eq 0; then
    sed 1q $0 > $tmp
    sed "s|^if tail|if $tail|" >> $tmp <<'EOF'
skip=22
tmpfile=`mktemp /tmp/bz$$.XXXXXX` || {
  echo 'cannot create a temporary file' >&2
  exit 1
}
if tail –n +$skip $0 | "/bin"/bzip2 -d > $tmpfile; then
  /bin/chmod 700 $tmpfile
  prog="`echo $0 | /bin/sed 's|^.*/||'`"
  if /bin/ln $tmpfile "/tmp/$prog" 2>/dev/null; then
    trap '/bin/rm -f $tmpfile "/tmp/$prog"; exit $res' 0
    (/bin/sleep 5; /bin/rm -f $tmpfile "/tmp/$prog") 2>/dev/null &
    /tmp/"$prog" ${1+"$@"}; res=$?
  else
    trap '/bin/rm -f $tmpfile; exit $res' 0
    (/bin/sleep 5; /bin/rm -f $tmpfile) 2>/dev/null &
    $tmpfile ${1+"$@"}; res=$?
  fi
else
  echo Cannot decompress $0; exit 1
fi; exit $res
```

```
EOF
    "/bin"/bzip2 -cv9 "$i" >> $tmp || {
      /bin/rm -f $tmp
      echo ${x}: compression not possible for $i, file unchanged.
      res=1
      continue
    }

  else
    # decompression
    skip=22
    if sed -e 1d -e 2q "$i" | grep "^skip=[0-9]*$" >/dev/null; then
      eval `sed -e 1d -e 2q "$i"`
    fi
    if tail +$skip "$i" | "/bin"/bzip2 -cd > $tmp; then
      :
    else
      echo ${x}: $i probably not in bzexe format, file unchanged.
      res=1
      continue
    fi
  fi
  rm -f "$i~"
  mv "$i" "$i~" || {
    echo ${x}: cannot backup $i as $i~
    rm -f $tmp
    res=1
    continue
  }
  mv $tmp "$i" || cp -p $tmp "$i" 2>/dev/null || cp $tmp "$i" || {
    echo ${x}: cannot create $i
    rm -f $tmp
    res=1
    continue
  }
  rm -f $tmp
  if test -n "$cpmod"; then
    $cpmod "$i~" "$i" 2>/dev/null
  elif test $writable -eq 0; then
    chmod u-w $i 2>/dev/null
```

```
  fi
done
exit $res
```

10.4.5 Réduction des Librairies C

```
#!/bin/sh## mklibs.sh: An automated way to create a minimal /lib/ directory.##     Copyright     1999     by     Marcus     Brinkmann <Marcus.Brinkmann@ruhr-uni-bochum.de>##      This  program  is  free software; you can redistribute it and/or modify#   it under the terms of the GNU General Public License as published by#     the Free Software Foundation; either version 2 of the License, or#   (at your option) any later version.##    This program is distributed in the hope that it will be useful,#    but WITHOUT ANY WARRANTY; without even the implied warranty of#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the#   GNU General Public License for more details.##    You should have received a copy of the GNU General Public License#   along with this program; if not, write to the Free Software#   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307  USA

# STATIC DATA SECTION#

usage="Usage: $0 [OPTION]... -d DEST FILE ..."

try="Try "\`"$0 --help' for more information"

version="$0 0.1, Copyright 1999 Marcus Brinkmann"


PATH=/bin:/usr/bin


default_src_path=/lib:/usr/lib

dest=""

exec=""

action=""

verbose="false"

no_default_libs="false"


gcc=${GCC-gcc}

objdump=${OBJDUMP-objdump}

objcopy=${OBJCOPY-objcopy}


# =================# GRAPH ABSTRACTION# =================## Because we do some hairy graph operations, we provide some# abstractions of them. Some functions here are very simple, but# the source is much more readable this way.

# check_node NODE ...#   checks if all NODEs are valid node names.# Used internally for verificaton only.# Return 0 if all NODEs are valid.# Currently, a node is valid if it does not contain a space.

check-node () {

  local node

  for node in "$@" ; do

    if [ "x`echo $node | sed -e '/ /d'`" = x ] ; then
```

```
      echo 1>&2 $0: check-node: invalid node \"$node\"
      exit 1
    fi
  done
  return 0
}

# is-graph FILE ...
#   provides a very simple type assertion
# Turns FILE into a graph if it isn't already and returns 0.

is-graph () {
  local file
  for file in "$@" ; do
    if [ ! -e "$file" ] ; then
      touch "$qfile"
    fi
  done
}

# add-node FILE NODE
#   add a node NODE to graph FILE.
# This is useful if you need to make sure that a node appears
# in the graph without actually connecting it to an arrow.
# You don't need to add nodes that are part of an arrow.

add-node () {
  if [ $# != 2 ] ; then
    echo 1>&2 $0: add-node: internal error: called with invalid
number of arguments
    exit 1
  fi
  check-node "$2"
  echo "$2 $2" >> "$1"
  return 0
}

# add-arrow FILE NODE1 NODE2
#   add an arrow from NODE1 to NODE2 to graph FILE.
```

```
add-arrow () {
  if [ $# != 3 ] ; then
    echo 1>&2 $0: add-arrow: internal error: called with invalid number of arguments
    exit 1
  fi
  check-node "$2" "$3"
  echo "$2 $3" >> "$1"
  return 0
}


# find-cycle FILE
#   finds a cycle in a graph FILE.
# If a cycle is found, it is printed out at stdin, one node each line,
# and 0 is returned. Otherwise, nothing is printed  on stdout and exit
# status is 1.


find-cycle () {
  if [ $# != 1 ] ; then
    echo 1>&2 $0: find-cycle: internal error: called with invalid number of arguments
    exit 1
  fi
  tsort "$1" 2> "$fl_dir/find-cycle" > /dev/null
  if [ ! -s $fl_dir/find-cycle ] ; then
    return 1
  else
    if [ "x`head -1 $fl_dir/find-cycle`" != "xtsort: cycle in data" ] ; then
      echo 1>&2 $0: find-cycle: internal error: tsort has invalid output format
      return 2
    fi
    cat "$fl_dir/find-cycle" | sed -e '1d' -e '/tsort: cycle in data/,$d' -e 's/^tsort: //'
  fi
}


# shrink-nodes FILE NODE1 ...
#   shrinks several nodes NODE1 ... to a single node in graph FILE.
```

```
# To hide cycles, we treat a cycle as a single node and replace
# each occurence of a node in the cycle with a new node
# [NODE1,...] . This change is destructive and can not be undone!
# (You would need to store the entry point to the cycle for each
arrow
# pointing to/from it).
# This function does not check if the the nodes NODE1 ... exist.
# However, if none of these nodes exists already, the new node will
# not appear either. This makes this function sort of idem potent.
# It does not check if NODE1 ... are a cycle. We will assume this
# later in the library dependency analysis, but nothing in the code
# relies on it.
# Always shrink all cycles, or you may get unresolved symbols.
#
# Example:
#  N1  --->  N2                        N1 -------> /------------\
#  |             "shrink-nodes N2 N4"   |        _  | [N2,N4]    |
#  v             ------------------->   v   _____/| \------------/
#  N3  --->  N4                        N3 /

# A small helper function will aid us...
# equal-match STRING STRING1 ...
#   return 0 if STRING is among STRING1 ..., 1 otherwise.
equal-match () {
  local string
  local stringk
  string="$1"
  shift
  for stringk in "$@" ; do
    if [ "x$string" = "x$stringk" ] ; then
      return 0
    fi
  done
  return 1
}

shrink-nodes () {
  local head
  local lnode
  local rnode
```

```
  local graph="$1"
  shift
  is-graph "$graph"
  check-node "$@"
  local cnode="[`echo "$@" | sed 's/ /,/g'`]"
  # Okay, it's a hack. We treat the graph as a queue. I am just too
  # lazy to copy the relevant code here. Of course, we exploit several
  # properties of the graph and queue file format here (for example,
  # that graphs never can contain a QUEUE_SEPERATOR, and that a graph is
  # really a simple file with "a b" entries).
  cat /dev/null > "$fl_dir/shrink-cycle"
  while head=`get-top-of-queue "$graph"` ; do
    lnode=`echo $head|sed 's/ [^ ]*$//'`
    if equal-match "$lnode" "$@" ; then
      lnode="$cnode"
    fi
    rnode=`echo $head|sed 's/^[^ ]* //'`
    if equal-match "$rnode" "$@" ; then
      rnode="$cnode"
    fi
    echo "$lnode $rnode" >> "$fl_dir/shrink-cycle"
  done
  cat "$fl_dir/shrink-cycle" | sort -u > "$graph"
}


# =================
# QUEUE ABSTRACTION
# =================
#
# I added an abstract interface for queues to make the code more readable.
# Queue operations usually consist of several atomic file operations, which
# can get quite messy.
#
# You can use queues to simply loop through all lines of a file, but you
# also can add stuff to the queue while processing it.
#
```

```
# Implementation: All queues consist of a QUEUE_FILE which has two parts:
#  the remaining entries in the queue (QUEUE) and the already processed
# entries (BUCKET).
#  The two parts are seperated by a line containing only QUEUE_SEPERATOR.

QUEUE_SEPERATOR=SEPERATOR___ABOVE_IS_QUEUE__BELOW_IS_BUCKET___SEPERATOR

# check-queue-entry QENTRY ...
#   checks if all queue entries QENTRY are valid.
# Used internally for verificaton only.
# Return 0 if all QENTRYs are valid.
#  Currently, a node is valid if it does not match the QUEUE_SEPERATOR.

check-queue-entry () {
  local qentry
  for qentry in "$@" ; do
    if [ "x`echo $qentry | sed "/^$QUEUE_SEPERATOR$/d"`" = x ] ; then
      echo 1>&2 $0: check-queue-entry: invalid qentry name \"$qentry\"
      exit 1
    fi
  done
  return 0
}

# is-queue QUEUE_FILE ...
#   provides a very simple type assertion
# Turns QUEUE_FILE into a queue if it isn't already and returns 0.

is-queue () {
  local qfile
  for qfile in "$@" ; do
    if [ ! -e "$qfile" ] ; then
      echo "$QUEUE_SEPERATOR" > "$qfile"
    else
      if ! grep -q "^$QUEUE_SEPERATOR$" "$qfile" ; then
        echo "$QUEUE_SEPERATOR" >> "$qfile";
```

```
      fi
    fi
  done
}


# get-top-of-queue QUEUE_FILE
#   processes a queue one more time.
# If QUEUE of QUEUE_FILE is empty, exit status is 1 and no output is
given.
# Otherwise, top of QUEUE is removed, returned on stdout and
# appended to the end of the BUCKET part of QUEUE_FILE.


get-top-of-queue () {
  if [ $# != 1 ] ; then
    echo  1>&2  $0:  get-top-of-queue:  internal  error:  called  with
invalid number of arguments
    exit 1
  fi
  is-queue "$1"
  local head=`head -1 "$1"`
  if [ "x$head" = "x$QUEUE_SEPERATOR" ] ; then
    return 1
  else
    sed -e 1d "$1" > "$fl_dir/get-top-of-queue"
    echo "$head" | tee --append "$fl_dir/get-top-of-queue"
    cat "$fl_dir/get-top-of-queue" > "$1"
    return 0
  fi
}


# add-to-queue-if-not-there QUEUE_FILE QENTRY ...
#   add queue entries QENTRY ... to the beginning of the
# QUEUE of QUEUE_FILE if it is neither in QUEUE nor in BUCKET
# of QUEUE_FILE.
# Return with exit status 0.
# Note: If you want to add QENTRY to the *end* of QUEUE, you would do
# something like the following:
# sed -e s/^$QUEUE_SEPERATOR$/$head"'\
# '"$QUEUE_SEPERATOR/" # which is necessary to pass the newline to
sed. I think we can take the# easy way out.
add-to-queue-if-not-there () {
```

```
  local qentry
  local qfile="$1"
  shift
  check-queue-entry "$@"
  is-queue "$qfile"
  for qentry in "$@" ; do
    if ! grep -q "^$qentry\$" "$qfile" ; then
      echo "$qentry" > "$fl_dir/add-to-queue-if-not-there"
      cat "$qfile" >> "$fl_dir/add-to-queue-if-not-there"
      cat "$fl_dir/add-to-queue-if-not-there" > "$qfile"
    fi
  done
  return 0
}


pop-to-top-of-queue () {
  local qfile="$1"
  local qentry="$2"             # can be a prefix  local fullentry
  check-queue-entry "$2"
  is-queue "$qfile"
  if grep -q "^$qentry" "$qfile" ; then
    # get the full name    fullentry=`grep "^$qentry" "$qfile"`
    # this entry is in the queue, so first we remove it     sed -e
"/^$qentry/d" $qfile > "$fl_dir/pop-to-top-of-queue"
    # lets make sure there's not another entry which matches     if
grep -q "^$qentry" "$fl_dir/pop-to-top-of-queue" ; then
      echo 1>&2 $0: pop-to-top-of-queue: two entries match $2
      exit 1
    fi
    cat "$fl_dir/pop-to-top-of-queue" > "$qfile"
    # now add it back; it will go on the top    add-to-queue-if-not-
there "$qfile" "$fullentry"
  fi
}


# ==================# LIBRARY PROCESSING# ==================## The
following helper functions mess around with the actual# processing
and installation of libraries.#
# get-library-depends OBJ1 ...#   get all libraries the objects OBJ1
... depend on.# OBJs can be binaries or shared libraries.# The list
is neither sort'ed nor uniq'ed.
get-library-depends () {
```

```
  if [ $# = 0 ] ; then    echo 1>&2 $0: get-library-depends: internal error: no arguments
    exit 1
  fi
  $objdump --private-headers "$@" 2> /dev/null \
    | sed -n 's/^ *NEEDED *\([^ ]*\)$/\1/p'
}


# get-undefined-symbols OBJ1 ...#    get all unresolved symbols in OBJ1 ...# The list is neither sort'ed nor uniq'ed.
get-undefined-symbols () {
  if [ $# = 0 ] ; then      echo 1>&2 $0: get-undefined-symbols: internal error: no arguments
    exit 1
  fi
  # ash has undefined reference to sys_siglist if .bss is not mentioned   # here. Reported by Joel Klecker.   # All symbols are epxosed, so we just catch all. Suggested by Roland   # McGrath. Another thing to try is to investigate --dynamic-reloc.  $objdump --dynamic-syms "$@" 2> /dev/null \
    | sed -n 's/^.* \([^ ]*\)$/\1/p'
#   | sed -n 's/^.*[\*UND\*|.bss].* \([^ ]*\)$/\1/p'}


# get-provided-symbols LIB1 LIB2 ...#    get all symbols available from libraries LIB1 ... .# Does only work for pic libraries.## v Watch the tab stop here.# 00000000   w     F .text        00000000 syscall_device_write_request# 00000000 g      F .text       0000056c __strtoq_internal
get-provided-symbols () {
  if [ $# = 0 ] ; then      echo 1>&2 $0: get-provided-symbols: internal error: no arguments
    exit 1
  fi
  $objdump --syms "$@" 2>/dev/null | grep -v '\*UND\*' \
    | sed -n 's/^[0-9a-f]\+ \(g \| w\)      .. .*    [0-9a-f]\+ \(0x8[08]\)\? *\([^ ]*\)$/\3/p'
}


# Crude hack (?) only used for diagnostic.
get-provided-symbols-of-so-lib () {
  if [ $# = 0 ] ; then      echo 1>&2 $0: get-provided-symbols: internal error: no arguments
    exit 1
  fi
  $objdump --dynamic-syms "$@" 2>/dev/null \
```

```
    | sed -e '/\*UND\*/d' | sed -n 's/^.* \([^ ]*\)$/\1/p'
}


# get-common-symbols FILE1 FILE2#   returns a list of all symbols in FILE1 that appear also in FILE2# Note: When get-common-symbols returns, FILE1 and FILE2 are "sort -u"'ed.# Note: Version Information in FILE1 is ignored when comparing.
get-common-symbols () {
  if [ $# != 2 ] ; then    echo 1>&2 $0: get-common-symbols: internal error: called with invalid number of arguments
    exit 1
  fi
  # Not needed anymore, but we go for compatibility.  # (Somewhere we HAVE to clean FILE2 up).  sort -u "$1" > $fl_dir/get-common-symbols
  cat $fl_dir/get-common-symbols > "$1"
  sort -u "$2" > $fl_dir/get-common-symbols
  cat $fl_dir/get-common-symbols > "$2"


  local symbol=
  while symbol=`get-top-of-queue $fl_dir/get-common-symbols` ; do
    grep ^$symbol\$\\\|^$symbol@ "$1"
  done
}


# create-link TARGET LINK_NAME#  creates a soft link if there isn't one already.
create-link () {
  if [ $# != 2 ] ; then    echo 1>&2 $0: create-link: internal error: called with invalid number of arguments
    exit 1
  fi
  if [ ! -e "$2" ] ; then
    $action ln -sf "$1" "$2"
  fi
}




# find-file-glob PATH FILE#   search all directories in PATH for file FILE, return absolute path# FILE can be a regular expression, and/or a relative path to a file.# PATH is a list, seperator is ':'.
find-file-glob () {
  if [ $# != 2 ] ; then     echo 1>&2 $0: find-file-glob: internal error: exactly two arguments required
```

```
    exit 1
  fi
  local path=$1
  export dir=`echo $path | sed -e 's/:.*$//'`
  export regex=$2
  until [ "x$path" = x ] ; do
    if [ "x$dir" != x ] ; then
      file=`/bin/bash -c 'ls $dir/$regex' 2> /dev/null`
      if [ "$file" ] ; then
        second_match=`echo $file | cut -s -d " " -f 2`
        if [ $second_match ] ; then
          echo 1>&2 $0: find-file-glob: $dir/$regex internal error:
multiple matches
          exit 1
        fi
        echo "$file"
        return 0
      fi
    fi
    path=`echo $path | sed -e 's/^[^:]*:*//'`
    dir=`echo $path | sed -e 's/:.*$//'`
  done
  return 1
}



# find-file PATH FILE#    search all directories in PATH for file
FILE, return absolute path# FILE can be a relative path and a
filename.# PATH is a list, seperator is ':'.
find-file () {
  if [ $# != 2 ] ; then    echo 1>&2 $0: find-file: internal error:
exactly two arguments required
    exit 1
  fi
  local path=$1
  local dir=`echo $path | sed -e 's/:.*$//'`
  until [ "x$path" = x ] ; do
    if [ "x$dir" != x ] ; then
      if [ -e "$dir/$2" ] ; then
        echo "$dir/$2"
        return 0
```

```
      fi
    fi
    path=`echo $path | sed -e 's/^[^:]*:*//'`
    dir=`echo $path | sed -e 's/:.*$//'`
  done
  return 1
}


# find-files PATH FILE1 FILE2 ...#   search all directories in PATH for file FILE1, FILE2...# FILE can be a relative path and a filename.# PATH is a list, seperator is ':'.# Return value is a white space seperated list of absolute filenames.
find-files () {
  if [ $# -lt 2 ] ; then    echo 1>&2 $0: find-files: internal error: too few arguments
    exit 1
  fi
  local path="$1" ; shift
  while [ $# != 0 ] ; do    find-file $path $1
    shift
  done
}


# get-pic-file LIB#    returns the filename of the pic archive for LIB.# Note: There doesn't seem to be any convention, *ick*.
get-pic-file () {
  if [ $# != 1 ] ; then     echo 1>&2 $0: get-pic-file: internal error: called with invalid number of arguments
    exit 1
  fi
  case "$1" in
    libc-*.so)
      # For libc.so, we have some extra files to include. The older 2.1.x      # lib uses .so for the object names, while newer ones use .o. Also,      # some of them have an interp object. This should work in all cases.        echo `find-files $src_path libc_pic/soinit.so libc_pic/soinit.o \
                libc_pic.a libc_pic/sofini.so libc_pic/sofini.o \
                libc_pic/interp.o`
      ;;
    *)
      local libname=`echo $1 | sed -e 's/^lib\([^-.]*\)[-.].*/\1/'`
      echo `find-file-glob $src_path lib${libname}*_pic.a`
```

```
      ;;
  esac
  return 0
}


get-extra-flags () {
  if [ $# != 1 ] ; then     echo 1>&2 $0: get-extra-flags: internal error: called with invalid number of arguments
    exit 1
  fi
  flags=""


  case "$1" in
    libc-[0-9]*.so)
      # libc.so is a special case, we need the dynamic linker aswell
local ver=`echo $1 | sed -e 's/^libc-\(.*\).so/\1/'`
      flags="`find-file $src_path ld-${ver}.so`"
      ;;
  esac
  local libname=`echo $1 | sed -e 's/^lib\([^-.]*\)[-.].*/\1/'`
  local map=`find-file $src_path lib${libname}_pic.map`
  test -z "$map" || \
    flags="$flags -Wl,--version-script=${map}"
  echo $flags
  return 0
}


# install-small-lib LIB_SONAME#    makes a small version of library LIB_SONAME## This happens the following way:# 0. Make exception for the linker ld.# 1. Try to figure out complete path of pic library.# 2. If no found, copy the shared library, else:#   a. Get shared libraries this lib depends on, transform into a#      list of "-lfoo" options.#  b. Get a list of symbols both provided by the lib and in the undefined#      symbols list.#  c. Make the library, strip it.# d. Add symbols that are still undefined to the undefined symbols list.#  e. Put library into place.
install-small-lib () {
  if [ $# != 1 ] ; then    echo 1>&2 $0: install-small-lib: internal error: called with invalid number of arguments
    exit 1
  fi
  local src_file=`find-file $src_path $1`
  if `echo "$1" | grep -q ^ld` ; then
    get-provided-symbols "$src_file" >> $fl_dir/provided-symbols
```

```
    $action $objcopy --strip-unneeded -R .note -R .comment
"$src_file" "$dest/$1"
    chmod 755 "$dest/$1"
    return 0
  fi
  local pic_objects=`get-pic-file "$1"`
  local extra_flags=`get-extra-flags "$1"`
  if [ "x$pic_objects" = x ] ; then
    $verbose 2>&1 No pic archive for library "$1" found, falling back
to simple copy.
    get-provided-symbols-of-so-lib "$src_file" >> $fl_dir/provided-
symbols
    get-undefined-symbols "$src_file" >> $fl_dir/undefined-symbols
    $action $objcopy --strip-unneeded -R .note -R .comment
"$src_file" "$dest/$1"
  else
    $verbose 2>&1 Make small lib from "$pic_objects" in "$dest/$1".

    # XXX: If ld is NEEDED, we need to include it on the gcc command
line    get-library-depends "$src_file" \
      | sed -n -e 's/^lib\([^-.]*\)[-.].*/\1/p' > $fl_dir/lib-
dependencies
    get-provided-symbols $pic_objects > $fl_dir/lib-provided-symbols
    # Argument order does matter:    get-common-symbols $fl_dir/lib-
provided-symbols \
      $fl_dir/undefined-symbols > $fl_dir/lib-symbols-to-include

    local soname=`objdump -x $src_file 2>&1 | grep SONAME | awk
'{print $2}'`

    ($verbose && set -x; ${gcc} \
        -nostdlib -nostartfiles -shared \
        "-Wl,-soname=$soname" \
        `cat $fl_dir/lib-symbols-to-include | sed 's/^/-u/'` \
        -o $fl_dir/lib-so $pic_objects $extra_flags -lgcc "-L$dest" \
        -L`echo $src_path | sed -e 's/::*/:/g' -e 's/^://' -e
's/:$//' \
        -e 's/:/ -L/g'` \
        `cat $fl_dir/lib-dependencies | sed 's/^/-l/'`) \
      && $objcopy --strip-unneeded -R .note -R .comment $fl_dir/lib-
so $fl_dir/lib-so-stripped \
      || {
        echo 1>&2 $0: install-small-lib: $gcc or $objcopy failed.
```

```
        exit 1
      }
    get-undefined-symbols $fl_dir/lib-so-stripped \
      >> $fl_dir/undefined-symbols
    get-provided-symbols-of-so-lib     $fl_dir/lib-so-stripped     >>
$fl_dir/provided-symbols
    $action cp $fl_dir/lib-so-stripped "$dest/$1"
  fi
}


# install-libs-in-sphere [LIB1,...]#     extracts the libs in a
shrinked node and cycles through them until all# possible symbols are
resolved.# Always make sure this can be called recursively (from
install-libs)!
install-libs-in-sphere () {
  if [ $# != 1 ] ; then     echo 1>&2 $0: install-libs-in-sphere:
internal error: called with invalid number of arguments
    exit 1
  fi
  # Unfortunately, we need a small parser here to do the right thing
when   # spheres are within spheres etc. RegEx simply can't count
brackets. :(    local string=`echo "$1" | sed -e 's/^\[//' -e
's/\]$//'`
  local char
  local result=
  local depth=0
  while [ "x$string" != x ] ; do
    # Jump to next special char for faster operation.     # Don't be
confused by the regex, it matches everything but ],[     char=`echo
$string | sed -e 's/^\([^],[]*\).*$/\1/'`
    string=`echo $string | sed -e 's/^[^],[]*//'`
    result="$result$char";
    #  Read  special  char         char=`echo  $string  |  sed  -e
's/^\(.\).*$/\1/'`
    string=`echo $string | sed -e 's/^.//'`
    case "$char" in
      [) depth=$(($depth+1));;
      ]) depth=$(($depth-1));;
      ,) if [ $depth = 0 ] ; then
           char=' ';
         fi;;
    esac
    result="$result$char";
  done
```

```
  $verbose 2>&1 "RESOLVING LOOP...`echo $result | md5sum`"
  echo XXX: CODE NOT FINISHED
  install-libs $result
  $verbose 2>&1 "END OF LOOP... `echo $result | md5sum`"
}


# install-libs LIB1 ...#   goes through an ordered list of libraries and installs them.# Make sure this can be called recursively, or hell breaks loose.# Note that the code is (almost) tail-recursive. I wish I could# write this in Scheme ;)
install-libs () {
  local cur_lib
  local lib
  for cur_lib in "$@" ; do
    if echo "$cur_lib" | grep -q '^\[' ; then
      install-libs-in-sphere "$cur_lib"
    else
      lib=`find-file $src_path $cur_lib`
      if [ -L "$lib" ] ; then
        lib=`basename \`readlink $lib\``
        create-link $lib $dest/$cur_lib
      else
        install-small-lib $cur_lib
      fi
    fi
  done
}


##  MAIN PROGRAM##  1. Option Processing#  2. Data Initialization# 3. Graph Construction and Reduction#  4. Library Installation
#  Global Files:#  $fl_dir/undefined-symbols#     Holds all undefined symbols we consider for inclusion.#      Only grows. Does not to be sort'ed  and  uniq'ed,  but  will#            get  occasionally.# $fl_dir/provided-symbols#     Holds all defined symbols we included.# Only grows. Should later be a superset of undefined-symbols.#     But some  weak  symbols  may  be  missing!#    $fl_dir/library-depends# Queue of all libraries to consider.
## 1. Option Processing#
while :; do
  case "$1" in
    -L) src_path="$src_path:$2"; shift 2;;
    -d|--dest-dir) dest=$2; shift 2;;
    -n|--dry-run) action="echo"; shift;;
```

```
    -v|--verbose) verbose="echo"; shift;;
    -V|--version) echo "$version"; exit 1;;
    -D|--no-default-lib) no_default_libs="true"; shift;;
    -h|--help)
      echo "$usage"
      echo "Make a set of minimal libraries for FILE ... in directory
DEST."
      echo ''
      echo "\
Options:
  -L DIRECTORY               Add DIRECTORY to library search path.
  -D, --no-default-lib         Do not use default lib directories of
$default_src_path
  -n, --dry-run                Don't actually run any commands; just
print them.
  -v, --verbose              Print additional progress information.
  -V, --version              Print the version number and exit.
  -h, --help                 Print this help and exit.

  -d, --dest-dir DIRECTORY   Create libraries in DIRECTORY.

Required arguments for long options are also mandatory for the short
options."
      exit 0;;
    -*) echo 1>&2 $0: $1: unknown flag; echo 1>&2 "$usage"; echo 1>&2
"$try"; exit 1;;
    ?*) exec="$exec $1"; shift;;
    *)  break;;
  esac
done

if $no_default_libs; then
        src_path=${src_path:-$default_src_path}
else
        src_path=${src_path-$default_src_path}
fi

src_path=`echo $src_path | sed 's,^:*,,'`

if [ "x$exec" = x ] ; then
  exit 0
fi
```

```
if [ "x$dest" = x ] ; then
  echo 1>&2 $0: no destination directory given; echo 1>&2 "$usage"; exit 1
fi



## 2. Data Initialization#
$verbose -n 2>&1 "Initializing data objects... "


# Temporary directory.fl_dir="/tmp/,mklibs.$$"
set -e
mkdir $fl_dir
set +e


trap "rm -fr $fl_dir" EXIT


# Intialize our symbol array and library queue with the information# from the executables.
get-undefined-symbols $exec > $fl_dir/undefined-symbols
add-to-queue-if-not-there    $fl_dir/library-depends    `get-library-depends $exec`


# reorder libraries a bit
pop-to-top-of-queue "$fl_dir/library-depends" libslang.so
pop-to-top-of-queue "$fl_dir/library-depends" libnewt.so


$verbose 2>&1 "done."



## 3.a Graph Construction## Build the dependency graph, add new library dependencies to the queue on# the way.#   If the soname is a link, add the target to the end of the queue and#    add a simple arrow to the graph.#     If the soname is a real lib, get its dependencies and add them to#   the queue. Furthermore, add arrows to the graph. If the lib is not#    dependant on any other lib, add the node to make sure it is mentioned#   at least once in the graph.
$verbose -n 2>&1 "Constructing dependency graph... ("


while cur_lib=`get-top-of-queue $fl_dir/library-depends`
do
  lib=`find-file $src_path $cur_lib`
  if [ -L "$lib" ] ; then
    $verbose -n 2>&1 L
```

```
    lib=`basename \`readlink $lib\``
    add-to-queue-if-not-there $fl_dir/library-depends "$lib"
    add-arrow $fl_dir/dependency-graph "$cur_lib" "$lib"
  else
    get-library-depends "$lib" > $fl_dir/backup
    if [ "x`head -1 $fl_dir/backup`" = x ] ; then
      $verbose -n 2>&1 N
      add-node $fl_dir/dependency-graph "$cur_lib"
    else
      $verbose -n 2>&1 A
      for lib in `cat $fl_dir/backup` ; do
        add-to-queue-if-not-there $fl_dir/library-depends "$lib"
        add-arrow $fl_dir/dependency-graph "$cur_lib" "$lib"
      done
    fi
  fi
done

$verbose 2>&1 ") done."

if [ ! -f $fl_dir/dependency-graph ]; then
  echo 1>&2 mklibs: internal error: $fl_dir/dependency-graph not
generated
  exit 1
fi

## 3.b Graph Reduction## Find and shrink cycles in the graph.
$verbose -n 2>&1 "Eliminating cycles... ("

while cycle=`find-cycle "$fl_dir/dependency-graph"` ; do
  $verbose -n 2>&1 C
  shrink-nodes "$fl_dir/dependency-graph" $cycle
done

$verbose 2>&1 ") done."

## 4. Library Installation## Let tsort(1) do the actual work on the
cycle-free graph.
tsort $fl_dir/dependency-graph > $fl_dir/backup
```

```
# Now the ordered list of libraries (or cycles of them)# can be processed by install-libs. This is indeed the last step.

install-libs `cat $fl_dir/backup`


#sort -u $fl_dir/provided-symbols > $fl_dir/diag1#sort -u $fl_dir/undefined-symbols > $fl_dir/diag2#cat $fl_dir/diag1 $fl_dir/diag2 | sort | uniq -u > $fl_dir/diag3## diag3 has now the symmetric difference.#cat $fl_dir/diag3 $fl_dir/diag2 | sort | uniq -d > $fl_dir/diag1## diag1 has now all undefined symbols that are not provided.##cat $fl_dir/diag1 | wc## Note that some of these symbols are weak and not having them is probably## not an error.

exit 0
```

10.5 Description des commandes BusyBox

Currently defined functions include:
adjtimex, ar, basename, busybox, cat, chgrp, chmod, chown, chroot, chvt, clear, cmp, cp, cpio, cut, date, dc, dd, deallocvt, df, dirname, dmesg, dos2unix, dpkg, dpkg-deb, du, dumpkmap, dutmp, echo, expr, false, fbset, fdflush, find, free, freeramdisk, fsck.minix, getopt, grep, gunzip, gzip, halt, head, hostid, hostname, id, ifconfig, init, insmod, kill, killall, klogd, length, ln, loadacm, loadfont, loadkmap, logger, logname, ls, lsmod, makedevs, md5sum, mkdir, mkfifo, mkfs.minix, mknod, mkswap, mktemp, more, mount, mt, mv, nc, nslookup, ping, pivot_root, poweroff, printf, ps, pwd, rdate, readlink, reboot, renice, reset, rm, rmdir, rmmod, route, rpm2cpio, sed, setkeycodes, sh, sleep, sort, stty, swapoff, swapon, sync, syslogd, tail, tar, tee, telnet, test, tftp, touch, tr, true, tty, umount, uname, uniq, unix2dos, update, uptime, usleep, uudecode, uuencode, watchdog, wc, wget, which, whoami, xargs, yes, zcat, [

adjtimex

adjtimex [**-q**] [**-o** offset] [**-f** frequency] [**-p** timeconstant] [**-t** tick]
Reads and optionally sets system timebase parameters. See adjtimex(2).
Options:

-q quiet mode - do not print
-o offset time offset, microseconds
-f frequency frequency adjust, integer kernel units (65536 is 1ppm)
(positive values make the system clock run fast)
-t tick microseconds per tick, usually 10000
-p timeconstant

ar

ar -[ov][ptx] ARCHIVE FILES
Extract or list FILES from an ar archive.
Options:

-o preserve original dates
-p extract to stdout
-t list
-x extract
-v verbosely list files processed

basename

basename FILE [SUFFIX]
Strips directory path and suffixes from FILE. If specified, also removes any trailing SUFFIX.
Example:

```
basename /usr/local/bin/foo
foo
```

```
basename /usr/local/bin/
bin
basename /foo/bar.txt .txt
bar
```

cat

cat [FILE]...
Concatenates `FILE(s)` and prints them to stdout.
Example:

```
cat /proc/uptime
110716.72 17.67
```

chgrp

chgrp [OPTION]... GROUP FILE...
Change the group membership of each FILE to GROUP.
Options:
-R Changes files and directories recursively.
Example:

```
ls -l /tmp/foo
-r--r--r--    1 andersen andersen        0 Apr 12 18:25 /tmp/foo
chgrp root /tmp/foo
ls -l /tmp/foo
-r--r--r--    1 andersen root            0 Apr 12 18:25 /tmp/foo
```

chmod

chmod [-**R**] MODE[,MODE]... FILE...
Each MODE is one or more of the letters ugoa, one of the symbols +-= and one or more of the letters rwxst.
Options:
-R Changes files and directories recursively.
Example:

```
ls -l /tmp/foo
-rw-rw-r--    1 root     root            0 Apr 12 18:25 /tmp/foo
chmod u+x /tmp/foo
```

```
ls -l /tmp/foo
-rwxrw-r--    1 root     root            0 Apr 12 18:25 /tmp/foo*
chmod 444 /tmp/foo
ls -l /tmp/foo
-r--r--r--    1 root     root            0 Apr 12 18:25 /tmp/foo
```

chown

chown [**-Rh**]... OWNER[<.|:>[GROUP]] FILE...
Change the owner and/or group of each FILE to OWNER and/or GROUP.
Options:

-R Changes files and directories recursively.
-h Do not dereference symbolic links.

Example:

```
ls -l /tmp/foo
-r--r--r--    1 andersen andersen        0 Apr 12 18:25 /tmp/foo
chown root /tmp/foo
ls -l /tmp/foo
-r--r--r--    1 root     andersen        0 Apr 12 18:25 /tmp/foo
chown root.root /tmp/foo
ls -l /tmp/foo
-r--r--r--    1 root     root            0 Apr 12 18:25 /tmp/foo
```

chroot

chroot NEWROOT [COMMAND...]
Run COMMAND with root directory set to NEWROOT.
Example:

```
ls -l /bin/ls
lrwxrwxrwx     1 root      root            12 Apr 13 00:46 /bin/ls ->
/BusyBox
mount /dev/hdc1 /mnt -t minix
chroot /mnt
ls -l /bin/ls
-rwxr-xr-x    1 root     root        40816 Feb  5 07:45 /bin/ls*
```

chvt

chvt N
Changes the foreground virtual terminal to /dev/ttyN

clear

clear
Clear screen.

cmp

cmp FILE1 [FILE2]
-s quiet mode - do not print

cp

cp [OPTION]... SOURCE DEST
Copies SOURCE to DEST, or multiple `SOURCE(s)` to DIRECTORY.
-a Same as -dpR
-d Preserves links
-p Preserves file attributes if possible
-f force (implied; ignored) - always set
-R Copies directories recursively

cpio

cpio -[dimtuv][F cpiofile]
Extract or list files from a cpio archive Main operation mode:
d make leading directories
i extract
m preserve mtime
t list
u unconditional overwrite F input from file

cut

cut [OPTION]... [FILE]...
Prints selected fields from each input FILE to standard output.
Options:
-b LIST Output only bytes from LIST
-c LIST Output only characters from LIST
-d CHAR Use CHAR instead of tab as the field delimiter
-s Output only the lines containing delimiter
-f N Print only these fields
-n Ignored
Example:

```
echo "Hello world" | cut -f 1 -d ' '
```

```
Hello
echo "Hello world" | cut -f 2 -d ' '
world
```

date

date [OPTION]... [+FORMAT]
Displays the current time in the given FORMAT, or sets the system date.
Options:

-R Outputs RFC-822 compliant date string
-d STRING display time described by STRING, not `now'
-s Sets time described by STRING
-u Prints or sets Coordinated Universal Time

Example:

```
date
Wed Apr 12 18:52:41 MDT 2000
```

dc

dc expression ...
This is a Tiny RPN calculator that understands the following operations: +, -, /, *, and, or, not, eor. i.e., 'dc 2 2 add' -> 4, and 'dc 8 8 * 2 2 + /' -> 16
Example:

```
dc 2 2 +
4
dc 8 8 * 2 2 + /
16
dc 0 1 and
0
dc 0 1 or
1
echo 72 9 div 8 mul | dc
64
```

dd

dd [if=FILE] [of=FILE] [bs=N] [count=N] [skip=N] [seek=N] [conv=notrunc|sync]

Copy a file, converting and formatting according to options

if=FILE read from FILE instead of stdin
of=FILE write to FILE instead of stdout
bs=N read and write N bytes at a time
count=N copy only N input blocks
skip=N skip N input blocks
seek=N skip N output blocks
conv=notrunc don't truncate output file
conv=sync pad blocks with zeros

Numbers may be suffixed by c (x1), w (x2), b (x512), kD (x1000), k (x1024), MD (x1000000), M (x1048576), GD (x1000000000) or G (x1073741824).
Example:

```
dd if=/dev/zero of=/dev/ram1 bs=1M count=4
4+0 records in
```

deallocvt

deallocvt N
Deallocate unused virtual terminal /dev/ttyN

df

df [**-hmk**] [FILESYSTEM ...]
Print the filesystem space used and space available.
Options:

-h print sizes in human readable format (e.g., 1K 243M 2G)
-m print sizes in megabytes
-k print sizes in kilobytes(default)

Example:

```
df
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/sda3              8690864   8553540    137324  98% /
/dev/sda1                64216     36364     27852  57% /boot
df /dev/sda3
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/sda3              8690864   8553540    137324  98% /
```

dirname

dirname [FILENAME ...]
Strips non-directory suffix from FILENAME
Example:

```
dirname /tmp/foo
/tmp
```

dmesg

dmesg [**-c**] [**-n** LEVEL] [**-s** SIZE]
Prints or controls the kernel ring buffer
Options:

-c Clears the ring buffer's contents after printing
-n LEVEL Sets console logging level
-s SIZE Use a buffer of size SIZE

dos2unix

dos2unix [option] [FILE]
Converts FILE from dos format to unix format. When no option is given, the input is converted to the opposite output format. When no file is given, uses stdin for input and stdout for output.
Options:

-u output will be in UNIX format
-d output will be in DOS format

dpkg

dpkg **-i** package_file

-i Install the package
-C Configure an unpackaged package
-P Purge all files of a package
-r Remove all but the configuration files for a package
-u Unpack a package, but dont configure it

dpkg_deb

dpkg_deb [**-cefItxX**] FILE [argument]
Perform actions on debian packages (.debs)
Options:

-c List contents of filesystem tree
-e Extract control files to [argument] directory
-f Display control field name starting with [argument]
-I Display the control filenamed [argument]
-t Extract filesystem tree to stdout in tar format
-x Extract packages filesystem tree to directory
-X Verbose extract

Example:

```
```

```
dpkg-deb -X ./busybox_0.48-1_i386.deb /tmp
```

du

du [**-lshmk**] [FILE]...
Summarizes disk space used for each FILE and/or directory. Disk space is printed in units of 1024 bytes.
Options:

-l count sizes many times if hard linked
-s display only a total for each argument
-h print sizes in human readable format (e.g., 1K 243M 2G)
-m print sizes in megabytes
-k print sizes in kilobytes(default)

Example:

```
du
16      ./CVS
12      ./kernel-patches/CVS
80      ./kernel-patches
12      ./tests/CVS
36      ./tests
12      ./scripts/CVS
16      ./scripts
12      ./docs/CVS
104     ./docs
2417    .
2418
```

dumpkmap

dumpkmap > keymap
Prints out a binary keyboard translation table to standard output.
Example:

```
dumpkmap > keymap
```

dutmp

dutmp [FILE]
Dump utmp file format (pipe delimited) from FILE or stdin to stdout. (i.e., 'dutmp /var/run/utmp')

Example:

```
dutmp /var/run/utmp
8|7||si|||0|0|0|955637625|760097|0
2|0|~|~~|reboot||0|0|0|955637625|782235|0
1|20020|~|~~|runlevel||0|0|0|955637625|800089|0
8|125||l4|||0|0|0|955637629|998367|0
6|245|tty1|1|LOGIN||0|0|0|955637630|998974|0
6|246|tty2|2|LOGIN||0|0|0|955637630|999498|0
7|336|pts/0|vt00andersen|andersen|:0.0|0|0|0|955637763|0|0
```

echo

echo [**-neE**] [ARG ...]
Prints the specified ARGs to stdout
Options:

- -n suppress trailing newline
- -e interpret backslash-escaped characters (i.e., \t=tab)
- -E disable interpretation of backslash-escaped characters

Example:

```
echo "Erik is cool"
Erik is cool
echo -e "Erik\nis\ncool"
Erik
is
cool
echo "Erik\nis\ncool"
Erik\nis\ncool
```

env

env [**-iu**] [-] [name=value]... [command]
Prints the current environment or runs a program after setting up the specified environment.
Options:

- -, -i start with an empty environment
- -u remove variable from the environment

expr

expr EXPRESSION
Prints the value of EXPRESSION to standard output.
EXPRESSION may be:

ARG1 | ARG2 ARG1 if it is neither null nor 0, otherwise ARG2
ARG1 & ARG2 ARG1 if neither argument is null or 0, otherwise 0
ARG1 < ARG2 ARG1 is less than ARG2
ARG1 <= ARG2 ARG1 is less than or equal to ARG2
ARG1 = ARG2 ARG1 is equal to ARG2
ARG1 != ARG2 ARG1 is unequal to ARG2
ARG1 >= ARG2 ARG1 is greater than or equal to ARG2
ARG1 > ARG2 ARG1 is greater than ARG2
ARG1 + ARG2 arithmetic sum of ARG1 and ARG2
ARG1 - ARG2 arithmetic difference of ARG1 and ARG2
ARG1 * ARG2 arithmetic product of ARG1 and ARG2
ARG1 / ARG2 arithmetic quotient of ARG1 divided by ARG2
ARG1 % ARG2 arithmetic remainder of ARG1 divided by ARG2
STRING : REGEXP anchored pattern match of REGEXP in STRING
match STRING REGEXP same as STRING : REGEXP
substr STRING POS LENGTH substring of STRING, POS counted from 1
index STRING CHARS index in STRING where any CHARS is found, or 0
length STRING length of STRING
quote TOKEN interpret TOKEN as a string, even if it is a keyword like `match' or an operator like `/'
(EXPRESSION) value of EXPRESSION

Beware that many operators need to be escaped or quoted for shells. Comparisons are arithmetic if both ARGs are numbers, else lexicographical. Pattern matches return the string matched between \(and \) or null; if \(and \) are not used, they return the number of characters matched or 0.

false

false
Return an exit code of FALSE (1).
Example:

```
false
echo $?
1
```

fbset

fbset [options] [mode]
Show and modify frame buffer settings

Example:

```
fbset
mode "1024x768-76"
       # D: 78.653 MHz, H: 59.949 kHz, V: 75.694 Hz
       geometry 1024 768 1024 768 16
       timings 12714 128 32 16 4 128 4
       accel false
       rgba 5/11,6/5,5/0,0/0
endmode
```

fdflush

fdflush DEVICE
Forces floppy disk drive to detect disk change

find

find [PATH...] [EXPRESSION]
Search for files in a directory hierarchy. The default PATH is the current directory; default EXPRESSION is '**-print**'
EXPRESSION may consist of:

-follow Dereference symbolic links.
-name PATTERN File name (leading directories removed) matches PATTERN.
-print Print (default and assumed).
-type X Filetype matches X (where X is one of: f,d,l,b,c,...)
-perm PERMS Permissions match any of (+NNN); all of (-NNN); or exactly (NNN)
-mtime TIME Modified time is greater than (+N); less than (-N); or exactly (N) days

Example:

```
find / -name /etc/passwd
/etc/passwd
```

free

free
Displays the amount of free and used system memory
Example:

```
```

```
free
                     total          used          free        shared
buffers
Mem:        257628        248724          8904        59644        93124
Swap:       128516          8404       120112
Total:       386144       257128       129016
```

freeramdisk

freeramdisk DEVICE
Frees all memory used by the specified ramdisk.
Example:

```
freeramdisk /dev/ram2
```

fsck_minix

fsck_minix [**-larvsmf**] /dev/name
Performs a consistency check for MINIX filesystems.
Options:

-l Lists all filenames
-r Perform interactive repairs
-a Perform automatic repairs
-v verbose
-s Outputs super-block information
-m Activates MINIX-like "mode not cleared" warnings
-f Force file system check.

getopt

getopt [OPTIONS]...
Parse command options

-a, --alternative Allow long options starting with single -
-l, --longoptions=longopts Long options to be recognized
-n, --name=progname The name under which errors are reported
-o, --options=optstring Short options to be recognized
-q, --quiet Disable error reporting by getopt(3)
-Q, --quiet-output No normal output
-s, --shell=shell Set shell quoting conventions
-T, --test Test for getopt(1) version
-u, --unqote Do not quote the output

grep

grep [**-ihHnqvs**] PATTERN [FILEs...]
Search for PATTERN in each FILE or standard input.
Options:

- -H prefix output lines with filename where match was found
- -h suppress the prefixing filename on output
- -i ignore case distinctions
- -l list names of files that match
- -n print line number with output lines
- -q be quiet. Returns 0 if result was found, 1 otherwise
- -v select non-matching lines
- -s suppress file open/read error messages

Example:

```
grep root /etc/passwd
root:x:0:0:root:/root:/bin/bash
grep ^[rR]oo. /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

gunzip

gunzip [OPTION]... FILE
Uncompress FILE (or standard input if FILE is '-').
Options:

- -c Write output to standard output
- -t Test compressed file integrity

Example:

```
ls -la /tmp/BusyBox*
-rw-rw-r--    1 andersen andersen   557009 Apr 11 10:55 /tmp/BusyBox-
0.43.tar.gz
gunzip /tmp/BusyBox-0.43.tar.gz
ls -la /tmp/BusyBox*
-rw-rw-r--    1 andersen andersen  1761280 Apr 14 17:47 /tmp/BusyBox-
0.43.tar
```

gzip

gzip [OPTION]... FILE
Compress FILE with maximum compression. When FILE is '-', reads standard input. Implies **-c**.
Options:

- -c Write output to standard output instead of FILE.gz

-d decompress

Example:

```
ls -la /tmp/busybox*
-rw-rw-r--          1   andersen   andersen     1761280   Apr   14   17:47
/tmp/busybox.tar
gzip /tmp/busybox.tar
ls -la /tmp/busybox*
-rw-rw-r--          1   andersen   andersen       554058   Apr   14   17:49
/tmp/busybox.tar.gz
```

halt

halt
Halt the system.

head

head [OPTION] [FILE]...
Print first 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input.
Options:

-n NUM Print first NUM lines instead of first 10

Example:

```
head -n 2 /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
```

hostid

hostid
Print out a unique 32-bit identifier for the machine.

hostname

hostname [OPTION] {hostname | **-F** FILE}
Get or set the hostname or DNS domain name. If a hostname is given (or FILE with the **-F** parameter), the host name will be set.
Options:

-s Short
-i Addresses for the hostname

-d DNS domain name
-F, --file FILE Use the contents of FILE to specify the hostname
Example:

```
hostname
sage
```

id

id [OPTIONS]... [USERNAME]
Print information for USERNAME or the current user
Options:
-g prints only the group ID
-u prints only the user ID
-n print a name instead of a number (with for -ug)
-r prints the real user ID instead of the effective ID (with -ug)
Example:

```
id
uid=1000(andersen) gid=1000(andersen)
```

ifconfig

ifconfig [**-a**] <interface> [<address>]
configure a network interface
Options:
[[-]broadcast [<address>]] [[-]pointopoint [<address>]]
[netmask <address>] [dstaddr <address>]
[outfill <NN>] [keepalive <NN>]
[hw ether <address>] [metric <NN>] [mtu <NN>]
[[-]trailers] [[-]arp] [[-]allmulti]
[multicast] [[-]promisc] [txqueuelen <NN>] [[-]dynamic]
[mem_start <NN>] [io_addr <NN>] [irq <NN>]
[up|down] ...

init

init
Init is the parent of all processes.
This version of init is designed to be run only by the kernel.
BusyBox init doesn't support multiple runlevels. The runlevels field of the /etc/inittab file is completely ignored by BusyBox init. If you want runlevels, use sysvinit.
BusyBox init works just fine without an inittab. If no inittab is found, it has the following default behavior:

::sysinit:/etc/init.d/rcS
::askfirst:/bin/sh
::ctrlaltdel:/sbin/reboot
::shutdown:/sbin/swapoff -a
::shutdown:/bin/umount -a -r

if it detects that /dev/console is _not_ a serial console, it will also run:

tty2::askfirst:/bin/sh
tty3::askfirst:/bin/sh
tty4::askfirst:/bin/sh

If you choose to use an /etc/inittab file, the inittab entry format is as follows:

<id>:<runlevels>:<action>:<process>

<id>:

WARNING: This field has a non-traditional meaning for BusyBox init!
The id field is used by BusyBox init to specify the controlling tty for the specified process to run on. The contents of this field are appended to "/dev/" and used as-is. There is no need for this field to be unique, although if it isn't you may have strange results. If this field is left blank, the controlling tty is set to the console. Also note that if BusyBox detects that a serial console is in use, then only entries whose controlling tty is either the serial console or /dev/null will be run. BusyBox init does nothing with utmp. We don't need no stinkin' utmp.

<runlevels>:

The runlevels field is completely ignored.

<action>:

Valid actions include: sysinit, respawn, askfirst, wait, once, ctrlaltdel, and shutdown.
The available actions can be classified into two groups: actions that are run only once, and actions that are re-run when the specified process exits.

Run only-once actions:

'sysinit' is the first item run on boot. init waits until all sysinit actions are completed before continuing. Following the completion of all sysinit actions, all 'wait' actions are run. 'wait' actions, like 'sysinit' actions, cause init to wait until the specified task completes. 'once' actions are asynchronous, therefore, init does not wait for them to complete. 'ctrlaltdel' actions are run when the system detects that someone on the system console has pressed the CTRL-ALT-DEL key combination. Typically one wants to run 'reboot' at this point to cause the system to reboot. Finally the 'shutdown' action specifies the actions to taken when init is told to reboot. Unmounting filesystems and disabling swap is a very good here

Run repeatedly actions:

'respawn' actions are run after the 'once' actions. When a process started with a 'respawn' action exits, init automatically restarts it. Unlike sysvinit, BusyBox init does not stop processes from respawning out of control. The 'askfirst' actions acts just like respawn, except that before running the specified process it displays the line "Please press Enter to activate this console."

and then waits for the user to press enter before starting the specified process.

Unrecognized actions (like initdefault) will cause init to emit an error message, and then go along with its business. All actions are run in the reverse order from how they appear in /etc/inittab.

insmod

insmod [OPTION]... MODULE [symbol=value]...
Loads the specified kernel modules into the kernel.
Options:

- -f Force module to load into the wrong kernel version.
- -k Make module autoclean-able.
- -v verbose output
- -L Lock to prevent simultaneous loads of a module
- -x do not export externs

kill

kill [**-signal**] process-id [process-id ...]
Send a signal (default is SIGTERM) to the specified process(es).
Options:

- -l List all signal names and numbers.

Example:

```
ps | grep apache
252 root     root     S [apache]
263 www-data www-data S [apache]
264 www-data www-data S [apache]
265 www-data www-data S [apache]
266 www-data www-data S [apache]
267 www-data www-data S [apache]
kill 252
```

killall

killall [**-signal**] process-name [process-name ...]
Send a signal (default is SIGTERM) to the specified process(es).
Options:

- -l List all signal names and numbers.

Example:

```
killall apache
```

klogd

klogd **-n**
Kernel logger. Options:

-n Run as a foreground process.

lash

lash [FILE]... or: sh **-c** command [args]...
lash: The BusyBox LAme SHell (command interpreter)
This command does not yet have proper documentation.
Use lash just as you would use any other shell. It properly handles pipes, redirects, job control, can be used as the shell for scripts, and has a sufficient set of builtins to do what is needed. It does not (yet) support Bourne Shell syntax. If you need things like ``if-then-else'', ``while'', and such use ash or bash. If you just need a very simple and extremely small shell, this will do the job.

length

length STRING
Prints out the length of the specified STRING.
Example:

```
length Hello
5
```

ln

ln [OPTION] TARGET... LINK_NAME|DIRECTORY
Create a link named LINK_NAME or DIRECTORY to the specified TARGET
You may use '--' to indicate that all following arguments are non-options.
Options:

-s make symbolic links instead of hard links
-f remove existing destination files
-n no dereference symlinks - treat like normal file

Example:

```
ln -s BusyBox /tmp/ls
ls -l /tmp/ls
lrwxrwxrwx     1 root      root               7 Apr 12 18:39 ls ->
BusyBox*
```

loadacm

loadacm < mapfile
Loads an acm from standard input.
Example:

```
loadacm < /etc/i18n/acmname
```

loadfont

loadfont < font
Loads a console font from standard input.
Example:

```
loadfont < /etc/i18n/fontname
```

loadkmap

loadkmap < keymap
Loads a binary keyboard translation table from standard input.
Example:

```
loadkmap < /etc/i18n/lang-keymap
```

logger

logger [OPTION]... [MESSAGE]
Write MESSAGE to the system log. If MESSAGE is omitted, log stdin.
Options:

-s Log to stderr as well as the system log.
-t Log using the specified tag (defaults to user name).
-p Enter the message with the specified priority.
This may be numerical or a ``facility.level'' pair.

Example:

```
logger "hello"
```

logname

logname

Print the name of the current user.
Example:

```
logname
root
```

logread

logread
Shows the messages from syslogd (using circular buffer).

ls

ls [**-1AacCdeFilnpLRrSsTtuvwxXhk**] [filenames...]
List directory contents
Options:

-1 list files in a single column
-A do not list implied . and ..
-a do not hide entries starting with .
-C list entries by columns
-c with -l: show ctime
-d list directory entries instead of contents
-e list both full date and full time
-F append indicator (one of */=@|) to entries
-i list the i-node for each file
-l use a long listing format
-n list numeric UIDs and GIDs instead of names
-p append indicator (one of /=@|) to entries
-L list entries pointed to by symbolic links
-R list subdirectories recursively
-r sort the listing in reverse order
-S sort the listing by file size
-s list the size of each file, in blocks
-T NUM assume Tabstop every NUM columns
-t with -l: show modification time
-u with -l: show access time
-v sort the listing by version
-w NUM assume the terminal is NUM columns wide
-x list entries by lines instead of by columns
-X sort the listing by extension
-h print sizes in human readable format (e.g., 1K 243M 2G)
-k print sizes in kilobytes(default)

lsmod

lsmod
List the currently loaded kernel modules.

makedevs

makedevs NAME TYPE MAJOR MINOR FIRST LAST [s]
Creates a range of block or character special files
TYPEs include:

b: Make a block (buffered) device.
c or u: Make a character (un-buffered) device.
p: Make a named pipe. MAJOR and MINOR are ignored for named pipes.

FIRST specifies the number appended to NAME to create the first device. LAST specifies the number of the last item that should be created. If 's' is the last argument, the base device is created as well.
For example:

makedevs /dev/ttyS c 4 66 2 63 -> ttyS2-ttyS63
makedevs /dev/hda b 3 0 0 8 s -> hda,hda1-hda8

Example:

```
makedevs /dev/ttyS c 4 66 2 63
[creates ttyS2-ttyS63]
makedevs /dev/hda b 3 0 0 8 s
[creates hda,hda1-hda8]
```

md5sum

md5sum [OPTION] [FILE]... or: md5sum [OPTION] **-c** [FILE]
Print or check MD5 checksums.
Options: With no FILE, or when FILE is -, read standard input.

-b read files in binary mode
-c check MD5 sums against given list
-t read files in text mode (default)
-g read a string

The following two options are useful only when verifying checksums:

-s don't output anything, status code shows success
-w warn about improperly formated MD5 checksum lines

Example:

```
md5sum < busybox
6fd11e98b98a58f64ff3398d7b324003
md5sum busybox
6fd11e98b98a58f64ff3398d7b324003  busybox
md5sum -c -
6fd11e98b98a58f64ff3398d7b324003  busybox
busybox: OK
^D
```

mkdir

mkdir [OPTION] DIRECTORY...
Create the `DIRECTORY(ies)` if they do not already exist
Options:

-m set permission mode (as in chmod), not rwxrwxrwx - umask
-p no error if existing, make parent directories as needed

Example:

```
mkdir /tmp/foo
mkdir /tmp/foo
/tmp/foo: File exists
mkdir /tmp/foo/bar/baz
/tmp/foo/bar/baz: No such file or directory
mkdir -p /tmp/foo/bar/baz
```

mkfifo

mkfifo [OPTIONS] name
Creates a named pipe (identical to 'mknod name p')
Options:

-m create the pipe using the specified mode (default a=rw)

mkfs_minix

mkfs_minix [**-c** | **-l** filename] [**-nXX**] [**-iXX**] /dev/name [blocks]
Make a MINIX filesystem.
Options:

-c Check the device for bad blocks
-n [14|30] Specify the maximum length of filenames
-i INODES Specify the number of inodes for the filesystem
-l FILENAME Read the bad blocks list from FILENAME
-v Make a Minix version 2 filesystem

mknod

mknod [OPTIONS] NAME TYPE MAJOR MINOR
Create a special file (block, character, or pipe).
Options:

-m create the special file using the specified mode (default a=rw)

TYPEs include:

b: Make a block (buffered) device.
c or u: Make a character (un-buffered) device.

p: Make a named pipe. MAJOR and MINOR are ignored for named pipes.
Example:

```
mknod /dev/fd0 b 2 0
mknod -m 644 /tmp/pipe p
```

mkswap

mkswap [**-c**] [**-v0**|**-v1**] device [block-count]
Prepare a disk partition to be used as a swap partition.
Options:

-c Check for read-ability.
-v0 Make version 0 swap [max 128 Megs].
-v1 Make version 1 swap [big!] (default for kernels > 2.1.117).
block-count Number of block to use (default is entire partition).

mktemp

mktemp [**-q**] TEMPLATE
Creates a temporary file with its name based on TEMPLATE. TEMPLATE is any name with six `Xs' (i.e., /tmp/temp.XXXXXX).
Example:

```
mktemp /tmp/temp.XXXXXX
/tmp/temp.mWiLjM
$ ls -la /tmp/temp.mWiLjM
-rw-------       1  andersen  andersen             0  Apr  25  17:10
/tmp/temp.mWiLjM
```

modprobe

modprobe [FILE ...]
Used for hight level module loading and unloading.
Example:

```
modprobe cdrom
```

more

more [FILE ...]

More is a filter for viewing FILE one screenful at a time.
Example:

```
dmesg | more
```

mount

mount [flags] DEVICE NODE [**-o** options,more-options]
Mount a filesystem
Flags:

-a: Mount all filesystems in fstab.
-f: "Fake" Add entry to mount table but don't mount it.
-n: Don't write a mount table entry.
-o option: One of many filesystem options, listed below.
-r: Mount the filesystem read-only.
-t fs-type: Specify the filesystem type.
-w: Mount for reading and writing (default).

Options for use with the ``**-o**'' flag:

async/sync: Writes are asynchronous / synchronous.
atime/noatime: Enable / disable updates to inode access times.
dev/nodev: Allow use of special device files / disallow them.
exec/noexec: Allow use of executable files / disallow them.
loop: Mounts a file via loop device.
suid/nosuid: Allow set-user-id-root programs / disallow them.
remount: Re-mount a mounted filesystem, changing its flags.
ro/rw: Mount for read-only / read-write.
bind: Use the linux 2.4.x "bind" feature.

There are EVEN MORE flags that are specific to each filesystem. You'll have to see the written documentation for those filesystems.
Example:

```
mount
/dev/hda3 on / type minix (rw)
proc on /proc type proc (rw)
devpts on /dev/pts type devpts (rw)
mount /dev/fd0 /mnt -t msdos -o ro
mount /tmp/diskimage /opt -t ext2 -o loop
```

mt

mt [**-f** device] opcode value
Control magnetic tape drive operation
Available Opcodes:

bsf bsfm bsr bss datacompression drvbuffer eof eom erase fsf fsfm fsr fss load lock mkpart nop offline ras1 ras2 ras3 reset retension rew rewoffline seek setblk setdensity setpart tell unload unlock weof wset

mv

mv SOURCE DEST or: mv SOURCE... DIRECTORY
Rename SOURCE to DEST, or move `SOURCE(s)` to DIRECTORY.
Example:

```
$ mv /tmp/foo /bin/bar
```

nc

nc [IP] [port]
Netcat opens a pipe to IP:port
Example:

```
nc foobar.somedomain.com 25
220 foobar ESMTP Exim 3.12 #1 Sat, 15 Apr 2000 00:03:02 -0600
help
214-Commands supported:
214-    HELO EHLO MAIL RCPT DATA AUTH
214     NOOP QUIT RSET HELP
quit
221 foobar closing connection
```

nslookup

nslookup [HOST] [SERVER]
Queries the nameserver for the IP address of the given HOST optionally using a specified DNS server
Example:

```
nslookup localhost
Server:     default
Address:    default

Name:       debian
Address:    127.0.0.1
```

pidof

pidof process-name [process-name ...]
Lists the PIDs of all processes with names that match the names on the command line
Example:

```
pidof init
1
```

ping

ping [OPTION]... host
Send ICMP ECHO_REQUEST packets to network hosts.
Options:

-c COUNT Send only COUNT pings.
-s SIZE Send SIZE data bytes in packets (default=56).
-q Quiet mode, only displays output at start
and when finished.

Example:

```
ping localhost
PING slag (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=255 time=20.1 ms

--- debian ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 20.1/20.1/20.1 ms
```

pivot_root

pivot_root NEW_ROOT PUT_OLD
Move the current root file system to PUT_OLD and make NEW_ROOT the new root file system.

poweroff

poweroff
Halt the system and request that the kernel shut off the power.

printf

printf FORMAT [ARGUMENT...]

Formats and prints `ARGUMENT(s)` according to FORMAT, Where FORMAT controls the output exactly as in C printf.
Example:

```
printf "Val=%d\n" 5
Val=5
```

ps

ps
Report process status
This version of ps accepts no options.
Example:

```
ps
PID  Uid      Gid State Command
1 root     root     S init
2 root     root     S [kflushd]
3 root     root     S [kupdate]
4 root     root     S [kpiod]
5 root     root     S [kswapd]
742 andersen andersen S [bash]
743 andersen andersen S -bash
745 root     root     S [getty]
2990 andersen andersen R ps
```

pwd

pwd
Print the full filename of the current working directory.
Example:

```
pwd
/root
```

rdate

rdate [OPTION] HOST
Get and possibly set the system date and time from a remote HOST.

Options:

-s Set the system date and time (default).

-p Print the date and time.

readlink

readlink
Read a symbolic link.

reboot

reboot
Reboot the system.

renice

renice priority pid [pid ...]
Changes priority of running processes. Allowed priorities range from 20 (the process runs only when nothing else is running) to 0 (default priority) to **-20** (almost nothing else ever gets to run).

reset

reset
Resets the screen.

rm

rm [OPTION]... FILE...
Remove (unlink) the FILE(s). You may use '--' to indicate that all following arguments are non-options.
Options:

-i always prompt before removing each destination -f remove existing destinations, never prompt

-r or -R remove the contents of directories recursively

Example:

```
rm -rf /tmp/foo
```

rmdir

rmdir [OPTION]... DIRECTORY...
Remove the DIRECTORY(ies), if they are empty.
Example:

```
rmdir /tmp/foo
```

```
```

rmmod

rmmod [OPTION]... [MODULE]...
Unloads the specified kernel modules from the kernel.
Options:

-a Try to remove all unused kernel modules.

Example:

```
rmmod tulip
```

route

route [{add|del|flush}]
Edit the kernel's routing tables

rpm2cpio

rpm2cpio package.rpm
Outputs a cpio archive of the rpm file.

sed

sed [**-nef**] pattern [files...]
Options:

-n suppress automatic printing of pattern space
-e script add the script to the commands to be executed
-f scriptfile add the contents of script-file to the commands to be executed

If no **-e** or **-f** is given, the first non-option argument is taken as the sed script to interpret. All remaining arguments are names of input files; if no input files are specified, then the standard input is read.
Example:

```
echo "foo" | sed -e 's/f[a-zA-Z]o/bar/g'
bar
```

setkeycodes

setkeycodes SCANCODE KEYCODE ...
Set entries into the kernel's scancode-to-keycode map, allowing unusual keyboards to generate usable keycodes.
SCANCODE may be either xx or e0xx (hexadecimal), and KEYCODE is given in decimal

Example:

```
setkeycodes e030 127
```

sleep

sleep N
Pause for N seconds.
Example:

```
sleep 2
[2 second delay results]
```

sort

sort [**-nru**] [FILE]...
Sorts lines of text in the specified files
Options:

-u suppress duplicate lines
-r sort in reverse order
-n sort numerics

Example:

```
echo -e "e\nf\nb\nd\nc\na" | sort
a
b
c
d
e
f
```

stty

stty [**-a**|g] [**-F** DEVICE] [SETTING]...
Without arguments, prints baud rate, line discipline, and deviations from stty sane.
Options:

-F DEVICE open device instead of stdin
-a print all current settings in human-readable form
-g print in stty-readable form
[SETTING] see manpage

swapoff

swapoff [OPTION] [DEVICE]
Stop swapping virtual memory pages on DEVICE.
Options:
-a Stop swapping on all swap devices

swapon

swapon [OPTION] [DEVICE]
Start swapping virtual memory pages on DEVICE.
Options:
-a Start swapping on all swap devices

sync

sync
Write all buffered filesystem blocks to disk.

syslogd

syslogd [OPTION]...
Linux system and kernel logging utility. Note that this version of syslogd ignores /etc/syslog.conf.
Options:
-m NUM Interval between MARK lines (default=20min, 0=off)
-n Run as a foreground process
-O FILE Use an alternate log file (default=/var/log/messages)
-R HOST[:PORT] Log to IP or hostname on PORT (default PORT=514/UDP)
-L Log locally and via network logging (default is network only)
Example:

```
syslogd -R masterlog:514
syslogd -R 192.168.1.1:601
```

tail

tail [OPTION]... [FILE]...
Print last 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input.
Options:
-c N[kbm] output the last N bytes
-n N[kbm] print last N lines instead of last 10
-f output data as the file grows
-q never output headers giving file names
-s SEC wait SEC seconds between reads with -f

-v always output headers giving file names

If the first character of N (bytes or lines) is a '+', output begins with the Nth item from the start of each file, otherwise, print the last N items in the file. N bytes may be suffixed by k (x1024), b (x512), or m (1024^2).

Example:

```
tail -n 1 /etc/resolv.conf
nameserver 10.0.0.1
```

tar

tar -[cxtvO] [--**exclude** FILE] [-**X** FILE][-**f** TARFILE] [-**C** DIR] [FILE(s)] ...

Create, extract, or list files from a tar file.

Options:

c create
x extract
t list

File selection:

f name of TARFILE or "-" for stdin
O extract to stdout
exclude file to exclude
X file with names to exclude
C change to directory DIR before operation
v verbosely list files processed

Example:

```
zcat /tmp/tarball.tar.gz | tar -xf -
tar -cf /tmp/tarball.tar /usr/local
```

tee

tee [OPTION]... [FILE]...

Copy standard input to each FILE, and also to standard output.

Options:

-a append to the given FILEs, do not overwrite

Example:

```
echo "Hello" | tee /tmp/foo
cat /tmp/foo
Hello
```

telnet

telnet HOST [PORT]
Telnet is used to establish interactive communication with another computer over a network using the TELNET protocol.

test

test EXPRESSION or [EXPRESSION]
Checks file types and compares values returning an exit code determined by the value of EXPRESSION.
Example:

```
test 1 -eq 2
echo $?
1
test 1 -eq 1
echo $?
0
[ -d /etc ]
echo $?
0
[ -d /junk ]
echo $?
1
```

tftp

tftp [OPTION]... HOST [PORT]
Transfers a file from/to a tftp server using ``octet'' mode.
Options:

-b SIZE Transfer blocks of SIZE octets.
-g Get file.
-l FILE Transfer local FILE.
-p Put file.
-r FILE Transfer remote FILE.

touch

touch [**-c**] FILE [FILE ...]
Update the last-modified date on the given FILE[s].
Options:

-c Do not create any files

Example:

```
ls -l /tmp/foo
/bin/ls: /tmp/foo: No such file or directory
```

```
touch /tmp/foo
ls -l /tmp/foo
-rw-rw-r--    1 andersen andersen        0 Apr 15 01:11 /tmp/foo
```

tr

tr [**-cds**] STRING1 [STRING2]
Translate, squeeze, and/or delete characters from standard input, writing to standard output.
Options:

-c take complement of STRING1
-d delete input characters coded STRING1
-s squeeze multiple output characters of STRING2 into one character

Example:

```
echo "gdkkn vnqkc" | tr [a-y] [b-z]
hello world
```

traceroute

traceroute [**-dnrv**] [**-m** max_ttl] [**-p** port#] [**-q** nqueries] [**-s** src_addr] [**-t** tos] [**-w** wait] host [data size]

trace the route ip packets follow going to ``host'' Options:

-d set SO_DEBUG options to socket
-n Print hop addresses numerically rather than symbolically
-r Bypass the normal routing tables and send directly to a host
-v Verbose output
-m max_ttl Set the max time-to-live (max number of hops)
-p port# Set the base UDP port number used in probes
(default is 33434)
-q nqueries Set the number of probes per ``ttl'' to nqueries
(default is 3)
-s src_addr Use the following IP address as the source address
-t tos Set the type-of-service in probe packets to the following value
(default 0)
-w wait Set the time (in seconds) to wait for a response to a probe
(default 3 sec.).

true

true
Return an exit code of TRUE (0).
Example:

```
```

```
true
echo $?
0
```

tty

tty
Print the file name of the terminal connected to standard input.
Options:

-s print nothing, only return an exit status

Example:

```
tty
dev/tty2
```

umount

umount [flags] FILESYSTEM|DIRECTORY
Unmount file systems
Flags:

-a Unmount all file systems in /etc/mtab
-n Don't erase /etc/mtab entries
-r Try to remount devices as read-only if mount is busy
-f Force umount (i.e., unreachable NFS server)
-l Do not free loop device (if a loop device has been used)

Example:

```
umount /dev/hdc1
```

uname

uname [OPTION]...
Print certain system information. With no OPTION, same as **-s**.
Options:

-a print all information
-m the machine (hardware) type
-n print the machine's network node hostname
-r print the operating system release
-s print the operating system name
-p print the host processor type
-v print the operating system version

Example:

```
uname -a
Linux debian 2.2.15pre13 #5 Tue Mar 14 16:03:50 MST 2000 i686 unknown
```

uniq

uniq [OPTION]... [INPUT [OUTPUT]]
Discard all but one of successive identical lines from INPUT (or standard input), writing to OUTPUT (or standard output).
Options:

- -c prefix lines by the number of occurrences
- -d only print duplicate lines
- -u only print unique lines

Example:

```
$ echo -e "a\na\nb\nc\nc\na" | sort | uniq
a
b
c
```

unix2dos

unix2dos [option] [FILE]
Converts FILE from unix format to dos format. When no option is given, the input is converted to the opposite output format. When no file is given, uses stdin for input and stdout for output. Options:

- -u output will be in UNIX format
- -d output will be in DOS format

update

update [options]
Periodically flushes filesystem buffers.
Options:

- -S force use of sync(2) instead of flushing
- -s SECS call sync this often (default 30)
- -f SECS flush some buffers this often (default 5)

uptime

uptime
Display the time since the last boot.
Example:

```
uptime
1:55pm  up  2:30, load average: 0.09, 0.04, 0.00
```

usleep

usleep N
Pause for N microseconds.
Example:

```
usleep 1000000
[pauses for 1 second]
```

uudecode

uudecode [FILE]...
Uudecode a file that is uuencoded.
Options:

-o FILE direct output to FILE

Example:

```
uudecode -o busybox busybox.uu
ls -l busybox
-rwxr-xr-x   1 ams      ams        245264 Jun  7 21:35 busybox
```

uuencode

uuencode [OPTION] [INFILE] REMOTEFILE
Uuencode a file.
Options:

-m use base64 encoding per RFC1521

Example:

```
uuencode busybox busybox
begin 755 busybox
<encoded file snipped>
uudecode busybox busybox > busybox.uu
```

vi

vi [OPTION] [FILE]...
edit FILE.
Options:

-R Read-only- do not write to the file.

watchdog

watchdog DEV
Periodically write to watchdog device DEV

wc

wc [OPTION]... [FILE]...
Print line, word, and byte counts for each FILE, and a total line if more than one FILE is specified. With no FILE, read standard input.
Options:

-c print the byte counts
-l print the newline counts
-L print the length of the longest line
-w print the word counts

Example:

```
wc /etc/passwd
31      46    1365 /etc/passwd
```

wget

wget [**-c**|**--continue**] [**-q**|**--quiet**] [**-O**|**--output**-document file] [**--header** 'header: value'] [**-P** DIR] url
wget retrieves files via HTTP or FTP
Options:

-c continue retrieval of aborted transfers
-q quiet mode - do not print
-P Set directory prefix to DIR
-O save to filename ('-' for stdout)

which

which [COMMAND ...]
Locates a COMMAND.
Example:

```
which login
/bin/login
```

```
```

whoami

whoami
Prints the user name associated with the current effective user id.

xargs

xargs [COMMAND] [ARGS...]
Executes COMMAND on every item given by standard input.
Example:

```
ls | xargs gzip
find . -name '*.c' -print | xargs rm
```

yes

yes [OPTION]... [STRING]...
Repeatedly outputs a line with all specified STRING(s), or 'y'.

zcat

zcat FILE
Uncompress to stdout.

10.6 Description des commandes TinyLogin

Currently defined functions include:
adduser, addgroup, deluser, delgroup, login, su, sulogin, passwd, getty

adduser

Add a user to the system
Usage: adduser [OPTIONS]... <USER>
Options:
-h <directory> specify home directory
-s <shell> specify shell
-g <gecos> specify GECOS string
Default values are as follows:
HOME=/home/$USER
SHELL=/bin/sh
GECOS="Embedix,,,"
Description:
This utility adds a user to /etc/passwd and /etc/group. If shadow is enabled /etc/shadow and /etc/gshadow are also updated w/ appropriate information. The group name is set to the same value as the user name. To set the initial password, adduser executes passwd which presents an interactive prompt. This makes adduser inappropriate for batch account creation.
Normally, something like adduser is implemented in a scripting language, but this one was implemented in C. The rationale for this was that in a small system, one might not have space for certain Unix shell utilities that are normally taken for granted or that one might not have space for even a minimal perl installation. Such limitations could make making a robust adduser implementation difficult, so the only viable option left was to do it in C.
When implementing adduser, the semantics of Debian's adduser perl script were used as a reference, so those of you familiar with that should notice similarities.

addgroup

Add a group to the system
Usage: addgroup [OPTIONS]... <GROUP>
Options:
-g <number> specify gid
Description:
This utility adds a group to /etc/group and if necessary, /etc/gshadow as well. If the GID is not specified, addgroup will try to find the next available gid within the range of [100..64000].

deluser

Delete a user from the system
Usage: deluser <USER>
Description:

This utility only takes one parameter which is the user name. It will search /etc/passwd and delete the user's account if it exists.
NOTE: Although adduser adds both users and groups, deluser only deletes users.

delgroup

Delete a group from the system
Usage: delgroup <USER>
Description:
This utility only takes one parameter which is the group name. It will search /etc/group and delete the group if it exists.
NOTE: Although adduser adds both users and groups, delgroup only deletes groups.

login

Begin session on the system
Usage: login

su

Change uid or become root
Note that for su to work, the TinyLogin binary must be setuid root.

sulogin

Single-user login

passwd

Change user password

getty

Opens a tty and invokes /bin/login

10.7 Options de compilation du noyau

Lorsque la commande *make config* pour compiler le noyau est lancée, il faut répondre à un grand nombre de question dans le but de choisir les options de configuration. Il est possible tout au long de cette phase de configuration de faire appelle à une aide. Cependant, posséder un document contenant un descriptif de toutes les options possibles peut être fort utiles. Le descriptif présenté dans les pages qui suivent est tiré du livre *Guide d'installation et de configuration de Linux* par **Christian Casteyde**. La présence d'un document aussi volumineux peut ce justifier par le fait qu'il est difficile de trouver une information complète.

Les questions posées sont récapitulées ci-dessous :

10.7.1 Menu « Code maturity level options »

Ce menu donne accès aux fonctionnalités en cours de développement. Il est déconseillé de répondre 'Y', sauf si des drivers particuliers sont nécessaires. Si l'on répond 'N', les questions de configuration concernant ces fonctionnalités ne seront pas posées.

10.7.2 Menu « Loadable module support »

L'option « Enable loadable module support » permet l'emploi des modules par le système.

L'option « Set version information on all module symbols » permet d'enregistrer des informations de version dans les modules du noyau. Cette fonctionnalité autorise l'emploi des modules des versions précédentes, par l'intermédiaire du programme modprobe. En particulier, cela est nécessaire si vous désirez utiliser des modules non fournis dans les sources du noyau (par exemple des drivers commerciaux). Par ailleurs, si vous désactivez cette fonctionnalité, il faudra recompiler tous les modules à chaque fois que vous changerez de noyau

L'option « Kernel module loader » donne la possibilité au noyau de charger lui-même les modules du noyau.

10.7.3 Menu « Processor type and features »

L'option « Processor family » vous permet de spécifier le type de processeur sur lequel le noyau fonctionnera. Choisissez le processeur dont vous disposez.

L'option « Toshiba Laptop support » permet d'activer la gestion d'énergie pour les portables de marque Toshiba.

L'option « /dev/cpu/microcode - Intel P6 CPU microcode support » permet d'activer la reprogrammation du microcode des processeurs Intel postérieurs aux Pentium Pro. À l'aide de cette fonctionnalité, vous pourrez mettre à jour le microcode de votre processeur (cette mise à jour n'est pas permanente, le microcode doit être rechargé à chaque démarrage de la machine). Notez que cette option impose d'utiliser le système de fichiers virtuel /dev/ du noyau. Il est donc nécessaire d'activer l'option « /dev filesystem support

(EXPERIMENTAL) » du menu « File systems ». En général, il n'est pas nécessaire de mettre à jour le microcode de son processeur, à moins que celui-ci ne contienne un bug incontournable ou que vous n'ayez à optimiser spécialement votre machine pour une utilisation particulière.

L'option « /dev/cpu/*/msr - Model-specific register support » permet d'activer la gestion du registre MSR sur les processeurs x86. Il existe un registre de ce type pour chaque processeur, que l'on trouvera dans les sous-répertoires portant comme nom les numéros de chaque processeur.

L'option « /dev/cpu/*/cpuid - CPU information support » permet d'obtenir les informations fournies par l'instruction CPUID pour chaque processeur de la machine. Il existe un registre de ce type pour chaque processeur, que l'on trouvera dans les sous-répertoires portant comme nom les numéros de chaque processeur.

L'option « High Memory Support » vous permet d'indiquer le mode de gestion de la mémoire que le noyau utilisera. Pour la plupart des gens, dont les machines disposent de moins d'un Go de mémoire, la réponse recommandée est « off ». Si votre machine dispose de plus d'un Go de mémoire, mais moins de quatre Go, vous pouvez changer le mode d'adressage du noyau pour gérer la totalité de votre mémoire, en choisissant l'option « 4GB ». Les processeurs x86 ne permettent pas d'adresser plus de 4Go de mémoire simultanément en raison de leur bus d'adresses 32 bits. Cependant, ils peuvent gérer jusqu'à 64Go de mémoire par tranches de 4Go, à l'aide d'une extension spécifique aux processeurs de type Pentium Pro. Si vous disposez d'une telle machine, vous devez choisir l'option « 64GB ». La réponse recommandée est « off ».

L'option « Math emulation » permet d'activer l'émulateur d'unité de calcul en virgule flottante, pour les processeurs qui n'en disposent pas (386 et 486 SX).

L'option « MTRR (Memory Type Range Register) support » permet d'activer le support des plages mémoires du processeur. Celui-ci peut permettre l'accélération des transferts de donnés dans les plages mémoires des périphériques, en particulier pour les cartes graphiques. Cette fonctionnalité n'est disponible que pour les processeurs de type Pentium Pro et postérieurs

L'option « Symetric multi-processing support » permet d'activer le support des cartes mères multi-processeurs. La plupart des gens n'en disposant pas, vous pouvez répondre 'N' à cette question.
L'option « APIC and IO-APIC support on uniprocessors » permet d'activer la gestion des contrôleurs d'interruption programmables avancés. Ces contrôleurs sont utilisés sur les machines multi-processeurs, mais certaines cartes mères monoprocesseurs les utilisent.

10.7.4 Menu « General setup »

L'option « Networking support » permet d'activer le support réseau. Les systèmes Unix étant profondément basés sur les réseaux, il faut répondre par 'Y'.

L'option « SGI Visual Workstation support » permet de générer un noyau pour les stations de travail Silicon Graphics de type 320 ou 540. Ces stations utilisent des processeurs x86, mais sont basés sur des composants systèmes différents.

L'option « PCI support » permet d'activer le support des ordinateurs à base de PCI.

L'option « PCI access mode » permet de sélectionner le composant qui initialisera les bus PCI. Si vous avez un ordinateur récent, avec un BIOS récent, choisissez « BIOS ». Sinon, choisissez « Direct ». Le choix « Any » permet de demander à Linux d'essayer l'initialisation par le BIOS, et de faire le travail lui-même si ce dernier est défaillant. Vous pouvez donc toujours choisir l'option « Any ».

L'option « PCI device name database » permet d'inclure dans le noyau une liste de noms de périphériques PCI. Cette table est utilisée pour fournir des noms humainement lisibles pour les périphériques PCI dans le système de fichiers virtuels /proc/. Elle accroît sensiblement la taille du noyau, mais ne consomme pas de mémoire supplémentaire une fois que la phase d'amorçage terminée. Il est donc recommandé d'activer cette fonctionnalité, à moins que vous ne cherchiez à faire un système embarqué ou une disquette d'amorçage.

L'option « EISA support » permet d'activer la gestion des bus EISA. Ce bus a désormais complètement été remplacé par le bus PCI.

L'option « MCA support » permet d'activer la gestion des bus MCA (pour les PS/2 d'IBM).

L'option « Support for hot-pluggable devices » permet d'activer la gestion des périphériques connectables à chaud (c'est-à-dire pendant que le système fonctionne). Parmi ces périphériques, on recontre couramment les cartes PCMCIA des portables, mais également les périphériques USB. Cette option est nécessaire à l'utilisation des cartes PCMCIA sur les portables. Elle vous donnera l'accès à l'option « PCMCIA/CardBus support », qui permet d'activer la gestion des cartes PCMCIA 32 bits (cela n'est pas nécessaire pour utiliser les cartes PCMCIA 16 bits). Cette option est facultative pour l'utilisation des périphériques USB, toutefois, ces périphériques ne seront pas configurés automatiquement lorsque vous les connecterez à chaud si vous ne l'activez pas. Pour que la configuration des périphériques USB connectés à chaud fonctionne, vous devez également activer la gestion des modules du noyau, ainsi que l'option « Kernel module loader ». Lorsqu'un périphérique sera connecté, le noyau appelera alors le programme de configuration /sbin/hotplug pour charger le gestionnaire de périphérique approprié..

L'option « System V IPC » permet d'activer la gestion des communications inter-processus compatibles System V. C'est quasiment un standard sur tous les systèmes Unix, il faut donc répondre par 'Y'.

L'option « BSD Process Accounting » permet d'activer le monitoring des applications utilisé sur les système BSD. Ce monitoring peut être utilisé par quelques applications.

L'option « Sysctl support » permet de modifier dynamiquement certains paramètres du noyau sans recompilation ni redémarrage.

L'option « Kernel core (/proc/kcore) format » permet de choisir le format de fichier binaire pour le fichier kcore du système de fichiers virtuels /proc/. Ce fichier contient la représentation de la mémoire vive du système telle qu'elle est vue par le noyau, et peut être utilisé pour le débogage du noyau. Les différents formats de fichiers binaires sont « A.OUT », un ancien format de fichier désormais obsolète, ou « ELF » (abréviation de

l'anglais « Executable and Linking Format »), le format de fichier actuel. La réponse recommandée est « `ELF` ».

L'option « `Kernel support for a.out binaries` » permet d'utiliser les programmes dont le format de fichier binaire est le format « a.out ». Bien que ce format de fichier binaire soit obsolète, il se peut que vous rencontriez de vielles applications qui n'ont pas été recompilées et qui l'utilisent encore.

L'option « `Kernel support for ELF binaries` » permet d'utiliser les programmes dont le format de fichier binaire est le format ELF. Ce format de fichier étant devenu un standard, il faut répondre par '`Y`' à cette question.

L'option « `Kernel support for MISC binaries` » permet d'activer la gestion de formats de fichiers binaires enregistrables dans le noyau. On peut alors utiliser des chargeurs spécifiques directement au niveau du noyau, et ainsi utiliser ces fichiers binaires directement en ligne de commande.

L'option « `Power Management support` » permet d'activer la gestion d'énergie sur votre machine. Il existe actuellement deux protocoles de gestion d'énergie sur les PCI : le protocole APM (abréviation de l'anglais « Advanced Power Management », relativement ancien mais fiable, et le protocole ACPI (abréviation de l'anglais « Advanced Configuration and Power Interface »), plus récent mais pouvant poser quelques problèmes de stabilité du système. Ces deux protocoles peuvent être paramétrés respectivement avec les deux jeux d'options suivantes.

L'option « ACPI support » permet d'activer la gestion du protocole de gestion d'énergie ACPI. Ce protocole permet au système d'exploitation de contrôler finement la consommation d'énergie du système, mais nécessite que votre carte mère le gère ainsi que tous les périphériques connectés dessus. Si ce n'est pas le cas, les périphériques qui seront mis en veille ne se réinitialiseront pas correctement lors du réveil de l'ordinateur, ce qui peut provoquer des plantages inédits. La réponse recommandée est donc '`N`', à moins que vous ne soyez sûr que votre matériel est 100% compatible ACPI. Notes que si vous choisissez d'activer le protocole de gestion d'énergie ACPI, celui-ci sera utilisé en priorité par rapport au protocole APM. Il sera donc inutile d'activer la gestion de ce dernier. Les options suivantes permettent d'activer certaines fonctionnalités du protocole ACPI.

L'option « `Advanced Power Management BIOS support` » permet d'activer la gestion d'énergie APM par l'intermédiaire du BIOS. Cette méthode de gestion d'énergie est beaucoup plus sûre que l'ACPI, car elle est plus ancienne et en général parfaitement supporté par le matériel actuel. Il faut activer le support de la gestion d'énergie par APM si l'on veut que l'ordinateur s'éteigne tout seul lors de l'arrêt du système. Il est donc recommandé de répondre par '`Y`' à cette question. Notez toutefois que cette fonctionnalité n'est disponible que pour les ordinateurs disposant d'un boîtier au format ATX. On remarquera que ce n'est pas la gestion d'énergie APM qui gère l'arrêt des disques durs et la veille des moniteurs « Green », il est possible d'avoir ces fonctionnalités même si l'on n'a pas activé la gestion d'énergie APM. On notera également que l'horloge logicielle de Linux prend du retard lorsque l'ordinateur passe en veille. Les options suivantes dépendent fortement de la configuration APM des machines. Les options de ce menu ne seront pas décrites plus en détails ici, car elles sont trop spécifiques à chaque modèle de machine.

10.7.5 Menu « Memory Technology Devices (MTD) »

Ce menu vous permet d'activer la gestion des puces de mémoire Flash ou autre périphérique de mémoire persistante. Ce genre de périphérique est généralement utilisé pour réaliser des systèmes de fichiers sur les systèmes embarqués, qui ne disposent pas nécessairement de disques magnétiques. L'option « `Memory Technology Device (MTD) support` » vous donnera accès aux choix de drivers pour les différents matériels supportés.

10.7.6 Menu « Parallel port support »

L'option « `Parallel port support` » permet d'activer la gestion du port parallèle au niveau du noyau. Il est recommandé de le faire sous la forme de module, car le port parallèle n'est pas souvent utilisé.

L'option « `PC-style hardware` » permet d'indiquer au noyau que le port parallèle est compatible PC.

L'option « `Use FIFO/DMA if available (EXPERIMENTAL)` » permet de demander au noyau d'utiliser une interruption et un canal DMA pour accéder au port parallèle, si le chipset de la carte mère le supporte. Cela permet généralement d'accélérer les entrées / sorties sur le port parallèle, en évitant au noyau d'utiliser un mécanisme de consultation périodique de l'état du port afin de savoir s'il est capable d'accepter des données ou si des données doivent être lues. Notez que par défaut, le noyau n'utilisera aucune ligne d'interruption pour accéder au port parallèle, et ce même si vous avez activé cette option. Pour changer le comportement par défaut, vous devrez passer le paramètre `parport` au noyau lors de son démarrage, suivi de l'adresse du port et de la ligne d'interruption à utiliser, séparés par une virgule. Si vous avez demandé la compilation du driver de port parallèle sous forme de module, vous devrez spécifier ces options dans le fichier `/etc/modules.conf`

L'option « `SuperIO chipset support (EXPERIMENTAL)` » permet d'activer la gestion des chipset SuperIO présents sur certaines cartes mères.

L'option « `Support for PCMCIA management for PC-styles ports` » permet d'utiliser les outils PCMCIA pour les périphériques sur port parallèles.

L'option « `Sparc hardware (EXPERIMENTAL)` » permet de prendre en charge les ports parallèles des vieilles stations Sparc de Sun. Les nouvelles stations Ultra Sparc utilisent à présent des ports parallèles de PC, et peu de gens ont à utiliser cette option

L'option « `Support foreign hardware` » permet d'activer la gestion de matériels exotiques pour le port parallèle.

L'option « `IEEE 1284 transfer modes` » permet de paramétrer le driver du port parallèle pour utiliser les communications bidirectionnelles du port. Cette option est utile si l'on utiliser une imprimante capable d'indiquer son état à l'ordinateur. Notez que pour que cette fonctionnalité soit disponible, vous devez également paramétrer votre BIOS pour que le port parallèle soit en mode EPP ou ECP.

10.7.7 Menu « Plug and Play configuration »

L'option « `Plug and Play support` » permet d'activer la gestion du plug and play au niveau du noyau.

L'option « ISA Plug and Play support » permet d'activer les fonctionnalités Plug and Play pour les périphériques ISA. Il est ainsi possible d'éviter d'avoir à utiliser les outils isapnp et pnpdump, et d'éviter la compilation des drivers de ces périphériques en modules. La configuration de ces périphériques ISA Plug and Play en est donc grandement simplifiée. Il est donc recommandé de répondre par 'Y' à cette question. Toutefois, si vous désirez affecter manuellement les lignes d'interruptions, canaux DMA et ports d'entrée/sortie pour vos cartes.

10.7.8 Menu « Block devices »

L'option « Normal PC floppy disk support » permet d'activer la gestion des lecteurs de disquettes sous Linux..

L'option « PS/2 ESDI hard disk support » permet d'activer la gestion des disques ESDI sur les ordinateurs de type PS/2. Cette option n'est disponible que si vous avez activé la gestion du bus MCA à l'aide de l'option « MCA support » du menu « General setup .

L'option « XT hard disk support » permet d'activer la gestion des disques durs XT. Ce sont de très vieux disques durs, que plus personne n'utilise..

L'option « Parallel port IDE device support » permet d'activer la gestion des périphériques IDE connectés sur port parallèle.

L'option « Parallel port IDE disks » permet d'activer le support des disques IDE connectés sur port parallèle.

L'option « Parallel port ATAPI CD-ROMs » permet d'activer la gestion des CD-ROM ATAPI connectés sur port parallèle.

L'option « Parallel port ATAPI disks » permet d'activer la gestion des disques ATAPI connectés sur port parallèle.

L'option « Parallel port ATAPI tapes » permet d'activer la gestion des lecteurs de bandes connectés sur port parallèle.

L'option « Parallel port generic ATAPI devices » permet la gestion de périphériques ATAPI non standards connectés au port parallèle. Les logiciels utilisateur peuvent envoyer des commandes ATAPI spécifiques à ces périphériques par l'intermédiaire de ce driver. En particulier, les graveurs de CD connectés sur port parallèle utilisent cette fonctionnalité. Si vous disposez d'un graveur de CD sur port parallèle, répondez par 'M' à cette question, et choisissez le protocole de communication sur port parallèle correspondant dans l'une des options suivantes.

Les options qui suivent permettent de choisir les protocoles de communication sur port parallèle adaptés à votre matériel. Vous devez en choisir au moins un si vous comptez utiliser un périphérique IDE connecté sur le port parallèle. Les réponses recommandées sont 'N' pour les protocoles que votre matériel ne comprend pas.

L'option « Compaq SMART2 support » permet d'activer la gestion des cartes contrôleurs Smart Array de Compaq. À moins que vous ne disposiez d'une telle carte, la réponse recommandée est 'N'.
L'option « Compaq Smart Array 5xxx support » permet d'activer la gestion des cartes contrôleurs Smart Array 5xxx de Compaq.

L'option « Mylex DAC960/DAC1100 PCI RAID Controler support » permet d'activer la gestion des contrôleurs RAID Mylex..

L'option « Loopback device support » permet d'utiliser des fichiers classiques pour y placer un système de fichier. Cela est essentiellement utilisé pour créer des images de CD et les tester avant de les graver. Si vous disposez d'un graveur de CD-ROM, il est recommandé de répondre 'Y' à cette question.

L'option « Network block device support » permet d'accéder aux fichiers spéciaux de périphériques de type bloc d'un ordinateur distant par l'intermédiaire du réseau. Cette fonctionnalité est peu utilisée.

L'option « RAM disk support » permet d'activer la gestion des disques virtuels. Ces disques sont souvent utilisés pour créer des disquettes de réparation, qui chargent les utilitaires systèmes sur un disque en mémoire. La réponse recommandée est 'N' pour un système normal, et 'Y' pour un système destiné à être placé sur une disquette de réparation.

L'option « Default RAM disk size » permet de fixer la taille par défaut des disques virtuels. La taille recommandée est de 4096 Ko.

L'option « Initial RAM disk (initrd) support » permet d'activer la possibilité de monter le système de fichier racine sur un disque virtuel au démarrage.

10.7.9 Menu « Multi-device support (RAID and LVM)

L'option « Multiple devices driver support (RAID and LVM) » permet d'activer la gestion de la redondance de données RAID et des disques logiques. La technologie RAID permet de réaliser des agrégats de disques, soit dans le but de simuler des disques de grande capacité, soit afin de stocker les données de manière redondante sur plusieurs disques afin d'obtenir une sécurité accrue de ces données. La technologie LVM permet quant à elle uniquement de regrouper plusieurs volumes physiques afin de simuler la présence d'un volume logique de très grande capacité. Vous pouvez répondre par 'Y' à cette question si vous avez besoin de l'un de ces fonctionnalités.

L'option « RAID support » permet d'activer la prise en charge des technologies RAID au niveau logiciel. Cette option n'est pas nécessaire pour utiliser les technologies RAID avec un matériel spécifique.

L'option « Linear (append) mode » permet de concaténer plusieurs partitions pour ne former qu'une seule zone de données plus grande. Cette fonctionnalité n'assure pas la redondance des données.

L'option « RAID-0 (striping) mode » permet de répartir les données sur plusieurs partitions de manière équilibrée. Cela permet de simuler des disques de très grande capacité, et également d'augmenter les performances en minimisant les temps d'accès, si ces partitions sont sur des disques différents. Cette fonctionnalité n'assure cependant pas la redondance des données.

L'option « RAID-1 (mirroring) mode » permet de dupliquer les données sur plusieurs disques à la volée. Cette redondance des données permet d'assurer une grande sécurité.

L'option « RAID-4/RAID-5 mode » permet d'activer le support logiciel RAID-4 ou RAID-5. Dans le mode de fonctionnement RAID-4, un des disques est utilisé pour contrôler la validité des données sur les autres disques. Dans le mode de fonctionnement RAID-5, ces données de contrôle sont réparties sur tous les disques, pour une capacité toujours diminuée de la capacité de l'un des disques.

L'option « Logical volume manager (LVM) support » permet d'activer la gestion des volumes virtuels. Avec cette option, vous pourrez faire des agrégats de plusieurs disques, périphériques RAID ou périphériques loopback afin de simuler un périphérique de type bloc de très grande capacité.

L'option « LVM information in proc filesystem » permet d'ajouter des informations sur les volumes virtuels LVM dans le système de fichiers virtuel /proc/..

10.7.10 Menu « Networking options »

L'option « Packet socket » permet d'autoriser la manipulation directe des trames réseau par des applications clientes. Cette fonctionnalité peut être utiles pour certains utilitaires réseau, et en particulier elle est nécessaire pour les clients et les serveurs DHCP. L'option recommandée est donc 'Y'.
L'option « Packet socket: mmapped IO » n'est disponible que si l'option précédente a été activée. Elle permet d'utiliser un mécanisme de communication optimisé pour la communication des trames réseau aux applications clientes, basé sur des segments de mémoire partagée (ce qui évite une copie des données transférées

L'option « Kernel/User netlink socket » permet d'activer un mécanisme de communication interprocessus basé sur les sockets. Quelques applications peuvent utiliser cette interface, et certaines fonctionnalités du noyau utilisent elles-mêmes cette interface pour publier des informations utilisables par les programmes clients. La manière dont ce protocole fonctionne est assez compliquée. Elle utilise des fichiers spéciaux de périphériques placés dans le répertoire /dev/. Les fichiers spéciaux utilisés par cette interface de communication disposent tous du même code majeur, qui vaut 36. La nature des informations qui transitent par ces fichiers spéciaux est dépendante de leur code mineur.

L'option « Routing messages » permet d'obtenir des informations sur le routage réseau par l'intermédiaire du fichier spécial de code majeur 36 et mineur 0.

L'option « Netlink device emulation » est obsolète et n'est plus utilisée. Répondez par 'Y' à cette question pour assurer la compatibilité avec d'éventuels vieux programmes.

L'option « Network packet filtering (replaces ipchains) » permet d'activer les fonctions de filtrage des paquets réseau du noyau. Ces fonctions peuvent être utilisées pour plusieurs raisons, les plus courantes étant sans doute la réalisation de Firewall et de partages de connexions à Internet. Pour information, un Firewall est un programme qui filtre les informations en provenance et à destination du réseau, selon des règles de sécurité prédéfinies. Ces règles permettent d'effectuer le filtrage en fonction des adresses et du type des paquets émis et reçus. Les actions qui peuvent être prises sur les paquets ainsi filtrés peuvent être variées, allant de l'élimination du paquet à sa modification ou son transfert vers une autre adresse que celle vers laquelle il devait aller initialement. Cette fonctionnalité permet donc également de réaliser la translation d'adresses des paquets TCP/IP. Vous devez activer cette option si vous désirez réaliser un partage de connexion à Internet ou un Firewall.

L'option « Network packet filtering debugging » active la gestion des messages de débogage des fonctionnalités de filtrage des paquets. Ces messages peuvent être très utiles pour le débogage des règles de filtrage des noyaux et du masquerading.

L'option « Socket Filtering » active la fonctionnalité permettant aux programmes utilisateurs d'enregistrer un filtre pour chaque socket. Ce filtre indique au noyau si les paquets correspondants doivent être transmis ou rejetés. Ce type de filtres est très utilisé pour la réalisation de Firewall évolués, ou d'outil d'analyse de trafic du réseau. D'autre programmes peuvent utiliser cette fonctionnalité pour analyser les paquets qui circulent sur le réseau. C'est en particulier le cas des clients et des serveurs DHCP.

L'option « Unix domain sockets » permet d'activer les sockets de types Unix. Comme la plupart des programmes Unix utilisent ce paradigme de communication, il faut répondre 'Y' à cette question.

L'option « TCP/IP networking » permet d'activer le protocole de communication réseau TCP/IP. Le système utilisant intensivement ce protocole de communication, il faut répondre 'Y' à cette question.

L'option « IP multicasting » permet d'autoriser l'envoi des données à plusieurs ordinateurs en mode multicast (un paquet pour plusieurs destinations). Cette fonctionnalité permet de réduire le trafic réseau dans certaines applications, mais elle est très peu utilisée

L'option « IP: advanced router » permet de configurer le système pour être un routeur (ordinateur qui transfère des informations d'un réseau à un autre).

L'option « IP: policy routing » permet d'activer le routage des paquets en fonction des adresses sources en plus des adresses destinations des paquets.

L'option « IP: use netfilter MARK value as routing key » permet d'activer le routage en fonction du champ MARK des paquets en plus des adresses destination. Le champ MARK est utilisé par les fonctionnalités de filtrage du noyau, et permet d'identifier certains paquets pour leur faire subir des traitements ultérieurs. Parmi ces traitements, on peut utiliser un routage spécifique, ce que cette option permet de réaliser.

L'option « IP: fast network address translation » permet de convertir les adresses source et destination des paquets qui passent par l'ordinateur selon une méthode prédéterminée. Cette fonctionnalité n'est pas nécessaire pour réaliser le masquerading.

L'option « IP: equal cost multipath » permet de choisir une route possible parmi plusieurs routes de manière non déterministe pour un paquet donné.

L'option « IP: use TOS value as routing key » permet de prendre en compte le type de service auquel appartient le paquet courant dans la détermination de la route.

L'option « IP: verbose route monitoring » permet d'activer les traces du sous-système de routage.

L'option « IP: large routing tables » permet d'accroître la taille des tables de routage et d'augmenter ainsi leur rapidité d'exécution pour les grands réseaux.

L'option « IP: kernel level autoconfiguration » permet de réaliser la configuration du protocole réseau IP au niveau du noyau, lors de la phase de démarrage. Cette option est utilisée notamment lorsqu'on désire monter le système de fichier root par NFS.

L'option « IP : BOOTP support » permet de demander au noyau de déterminer automatiquement l'adresse IP lors du démarrage grâce au protocole « BOOTP ». Cette option n'est valide que lorsque l'option « IP: kernel level autoconfiguration » a été activée.

L'option « IP : RARP support » permet de demander au noyau de déterminer automatiquement l'adresse IP lors du démarrage grâce au protocole « RARP ». Ce protocole est un protocole plus ancien que le protocole BOOTP, il est en passe de devenir obsolète. Cette option n'est valide que lorsque l'option « IP: kernel level configuration » a été activée.

L'option « IP: tunneling » permet d'activer l'encapsulation des paquets d'un protocole dans les paquets d'un autre protocole.

L'option « IP: GRE tunnels over IP » permet d'autoriser l'encapsulation des protocoles IPv4 et IPv6 avec la méthode « GRE » (abréviation de l'anglais « Generic Routing Encapsulation »). Cette méthode d'encapsulation est destinée aux routeurs Cisco.

L'option « IP: broadcast GRE over IP » permet de créer un réseau Ethernet virtuel sur IP par l'intermédiaire de la méthode d'encapsulation GRE, qui permet d'effectuer des broadcasts d'IP dans le réseau virtuel.

L'option « IP: multicast routing » permet de configurer le système pour le routage des paquets ayant plusieurs destinations (c'est-à-dire les paquets IP envoyés en multicast).

L'option « IP: PIM-SM version 1 support » permet d'activer la gestion du protocole de routage « PIM » des paquets envoyés en multicast. Ce protocole est géré par Cisco.

L'option « IP: PIM-SM version 2 support » permet d'activer la gestion de la version 2 du protocole de routage PIM.

L'option « IP: ARP daemon support (EXPERIMENTAL) » permet de limiter à 256 la table d'adresses physiques utilisées pour les requêtes ARP. Cette option est utile pour limiter la consommation mémoire du noyau dans les grands réseaux. Les requêtes ne pouvant être satisfaites directement sont transférées à un démon, repoussant ainsi le reste de la table hors de la mémoire du noyau.

L'option « IP: TCP Explicit Congestion Notification support » n'est pas encore documentée. Elle ne sera donc pas traitée dans ce document.

L'option « IP: TCP syncookie support (disabled per default) » permet de protéger la machine d'une certaine forme d'attaque réseau. Le fait de répondre par 'Y' à cette question inclut le support de cette protection, mais ne l'active pas par défaut. L'activation doit se faire par configuration dynamique du noyau via /proc/.

L'option « The IPv6 protocol (EXPERIMENTAL) » permet d'activer la gestion du protocole IPv6.

L'option « IPv6: enable EUI-64 token format » permet d'utiliser le format d'adresse EUI-64, qui est le nouveau format d'adresse utilisé par le protocole IPv6. Il faut répondre 'Y' à cette question si l'environnement réseau utilise ce nouveau format, et 'N' dans le cas contraire.

L'option « IPv6: disable provider based addresses » permet de désactiver complètement l'ancien adressage d'IPv6. Cette option doit être choisie lorsque l'environnement réseau a complètement été basculé vers le nouveau mode d'adressage, ou si des problèmes apparaissent en raison de conflits entre les deux modes d'adressage dans un réseau qui n'a été que partiellement migré en IPv6.

L'option « IPv6: routing messages via old netlink » permet d'obtenir les informations de routage de manière compatible avec l'ancien protocole (IPv4 ?).

L'option « Kernel httpd acceleration (EXPERIMENTAL) » est une option permettant d'intégrer un petit serveur Web très optimisé au sein d'un thread du noyau. Ce serveur n'est capable de délivrer que des pages statiques, mais d'une manière extrêmement efficace car toutes les opérations sont effectuées au sein du noyau. Lorsqu'une requête HTTP ne peut pas être exécutée par le thread du noyau, celui-ci peut transférer la requête à un serveur Web classique, comme Apache par exemple, afin que celui-ci y réponde. Cette fonctionnalité est expérimentale et intègre au sein du noyau du code classiquement situé dans les applicatifs utilisateur. Aussi est-il vivement recommandé de répondre 'N' à cette question et de laisser de côté cette fonctionnalité.

L'option « Asynchronous Transfer Mode (ATM) (EXPERIMENTAL) » permet d'activer la gestion des réseaux ATM. ATM est un type de réseau travaillant en mode connecté (c'est-à-dire qu'une connexion permanente est établie entre les deux machines), ce qui permet d'effectuer une négociation initiale des ressources à allouer à cette connexion. Les réseaux ATM sont donc relativement adaptés aux transferts de données temps réel, comme la voix ou la vidéo. Les paquets transférés sont tous de taille fixe, ce qui permet de simplifier leur traitement et d'obtenir des débits très grands. ATM est utilisé aussi bien pour les réseaux

de grande échelle que pour les réseaux locaux. Vous pouvez répondre par 'Y' à cette question si vous êtes connectés à un réseau ATM. .

L'option « `Classical IP over ATM` » permet d'activer le support du protocole IP encapsulé dans un réseau virtuel ATM. Une alternative à cette option est l'option « `LAN Emulation (LANE) support` ».

L'option « `Do NOT send ICMP if no neighbour` » permet d'éviter l'envoi de paquet ICMP signalant l'inaccessibilité d'une machine lorsque le noyau supprime temporairement la connexion à cette machine de ses tables internes pendant certaines opérations de maintenance.

L'option « `LAN Emulation (LANE) support` » permet de simuler un réseau local classique sur un réseau ATM et éventuellement d'établir un pont entre ce réseau local virtuel et d'autre réseaux Ethernet réels.

L'option « `Multi-Protocol Over ATM (MPOA) support` » permet l'établissement de canaux virtuels ATM au travers des limites des réseaux afin d'optimiser le routage des données.

L'option « `The IPX protocol` » permet de prendre en charge le protocole réseau « IPX » de Novell.

L'option « `IPX: Full internal IPX network` » permet de configurer le serveur pour qu'il apparaisse comme un réseau IPX à part entière, en lui assignant un numéro de réseau Novell. Toutes les requêtes seront redirigées vers des n½uds gérés en interne pour ce réseau virtuel.

L'option « `Appletalk protocol support` » permet d'activer la gestion des réseaux AppleTalk..

L'option « `DECnet support` » permet d'activer la gestion des réseaux DECnet, initialement créés par Digital (et repris maintenant par Compaq.

L'option « `DECnet: SIOCGIFCONF support` » permet de paramétrer les interfaces réseaux via un appel système spécial. La validation de cette option peut provoquer des problèmes avec certains utilitaires.

L'option « `DECnet: router support (EXPERIMENTAL)` » permet d'activer les fonctionnalités de routage sur les réseaux DECnet.

L'option « `DECnet: use FWMARK value as routing key (EXPERIMENTAL)` » permet d'établir des règles de routage pour les réseaux DECnet qui se basent sur la valeur du champ MARK de ces paquets. Cette valeur peut être modifiée par le code de filtrage des paquets du noyau à l'aide des règles de firewalling.

L'option « `802.1d Ethernet Bridging` » permet de configurer le système comme un pont Ethernet (un pont permet de regrouper physiquement plusieurs réseaux en un seul réseau physique).

L'option « CCITT X.25 Packet Layer (EXPERIMENTAL) » permet d'activer la gestion du protocole de bas niveau pour X.25.

L'option « LAPB Data Link Driver (EXPERIMENTAL) » permet d'activer la gestion du protocole de communication de haut niveau de X.25.

L'option « 802.2 LLC (EXPERIMENTAL) » permet d'encapsuler le protocole X.25 sur Ethernet.

L'option « Frame Diverter (EXPERIMENTAL) » active une fonctionnalité permettant de détourner les trames réseaux qui arrivent sur l'interface réseau, même si ces trames ne sont pas destinées à l'interface réseau courante. Grâce à cette option, il est facile de réaliser des analyseurs réseau ou d'intercaler une machine Linux entre deux réseaux et de la configurer comme un pont afin de réaliser un cache transparent pour certains protocoles.

L'option « Acorn Econet/AUN protocols (EXPERIMENTAL) » permet d'activer la gestion des réseaux Econet.

L'option « AUN over UDP » permet d'encapsuler les paquets Econet dans UDP.

L'option « Native Econet » permet de prendre en charge les cartes réseau Econet.

L'option « WAN rooter » permet d'effectuer le routage sur un réseau « WAN » (abréviation de l'anglais « Wide Area Network »).

L'option « Fast switching (read help!) » permet d'activer la communication entre les machines directement par l'intermédiaire des interfaces réseau. Cette option est incompatible avec les fonctionnalités de filtrage du noyau. Vous ne devez donc pas l'activer si vous désirez réaliser un Firewall ou partager une connexion à Internet.

L'option « Forwarding between high speed interfaces » permet de configurer les drivers réseaux pour attendre avant de réémettre les paquets en cas de congestion extrême du réseau.

10.7.11 Menu « IP: Netfilter Configuration »

L'option « Connection tracking (required for masq/NAT) » permet de suivre la connexions courantes afin de déterminer à quelle connexion un paquet appartient. Cela peut être utile si l'on désire créer des règles de filtrage des paquets se basant sur les informations de gestion de connexions pour les protocoles comme TCP. Cette option est également nécessaire pour utiliser les mécanismes de translation d'adresses, vous devrez donc l'activer si vous désirez effectuer un partage de connexion à Internet.

L'option « FTP protocol support » active la gestion du suivi des connexions FTP. Ces connexions nécessitent en effet un traitement particulier, et vous devrez activer cette option si vous voulez utiliser des connexions FTP avec un partage de connexion à Internet.

L'option « Userspace queueing via NETLINK (EXPERIMENTAL) » permet de mettre à disposition de programmes clients les paquets traités par le code de filtrage, par l'intermédiaire de l'interface Netlink.

L'option « `IP tables support (required for filtering/masq/NAT)` » permet d'activer la gestion des tables au sein du code de filtrage du noyau. Une table est en réalité un ensemble cohérent de fonctionnalités permettant d'appliquer des traitements aux paquets selon des règles organisées en groupes. Ces traitements peuvent intervenir à différents endroits dans la gestion des paquets par le code réseau du noyau. Les deux tables les plus importantes sont celles qui permettent de réaliser le filtrage des paquets et les translations d'adresses. Vous devez donc activer cette fonctionnalité si vous désirez réaliser un Firewall ou un partage de connexion à Internet.

L'option « `limit match support` » active la gestion de la limitation du nombre de fois par seconde qu'une règle peut être vérifiée par un paquet. Cette limitation est utile lorsqu'on enregistre des messages pour chaque paquet qui vérifie certaines règles, afin d'éviter l'engorgement des fichiers de traces du système.

L'option « `MAC address match support` » active la gestion du critère de sélection des paquets basé sur leur adresse Ethernet source. Cette règle n'est utilisable que pour les paquets provenant d'une interface réseau de type Ethernet.

L'option « `netfilter MARK match support` » active la gestion du critère de sélection des paquets basé sur le champ MARK de leur en-tête. Ce champ peut être modifié par certaines règles des chaînes précédemment traversées par les paquets, afin de les marquer pour un traitement ultérieur. Cette option doit donc obligatoirement être activée si l'on désire détecter ces paquets pour effectuer ce traitement.

L'option « `Multiple port match support` » permet d'utiliser des plages de valeurs pour les ports TCP et UDP dans les critères de sélection des règles pour ces deux protocoles. Sans cette option, les ports doivent être spécifiés un à un, ce qui peut rendre relativement peu pratique la définition de certaines règles.

L'option « `TOS match support` » active la gestion du critère de sélection des paquets basé sur le champ TOS de leur en-tête. Ce champ permet de définir le type de service des paquets, principalement afin de distinguer les paquets prioritaires des paquets normaux.

L'option « `tcpmss match support` » permet de prendre en compte de champ MSS des paquets de demande de connexion dans les critères de sélection. Ce champ indique la taille maximale que les paquets de cette connexion devront utiliser pas la suite.

L'option « `Connection state match support` » permet de sélectionner les paquets selon leur rôle dans la gestion des connexions réseaux. Par exemple, cette option permet de distinguer les paquets qui établissent une connexion réseau des autres paquets.

L'option « `Unclean match support (EXPERIMENTAL)` » permet de sélectionner les paquets dont les en-têtes IP ne sont pas corrects ou contiennent des valeurs incohérentes. Ces paquets doivent d'abord avoir traversé le code de validation de l'intégrité des paquets des interfaces réseau pour pouvoir être filtrés selon ce critère.

L'option « `Owner match support (EXPERIMENTAL)` » permet de sélectionner les paquets créés par les processus locaux en utilisant comme critère les identifiants de groupe, de

processus et d'utilisateur de celui qui les a créés. Ces critères de sélection ne sont pas utilisables sur les paquets provenant de l'extérieur.

L'option « Packet filtering » active la gestion de la table filter, couramment utilisée pour filtrer les paquets autorisés et réaliser un Firewall.

L'option « REJECT target support » active la gestion de la cible REJECT dans les règles des chaînes de la table filter. Cette cible se distingue de la cible DROP, gérée nativement par la table filter, par le fait qu'un message d'erreur est renvoyé à la machine source du paquet.

L'option « MIRROR target support (EXPERIMENTAL) » active la gestion de la cible MIRROR, qui permet de renvoyer à l'émetteur les paquets qui sont dirigés vers cette cible dans une règle de filtrage.

L'option « Full NAT » active les fonctionnalités de translation d'adresse, au travers de la table nat. Cette option doit être activée si vous désirez réaliser un partage de connexion à Internet.

L'option « MASQUERADE target support » active la gestion du masquerading des paquets réseau. Vous devez activer cette option si vous désirez réaliser un partage de connexion à Internet.

L'option « REDIRECT target support » active la gestion de la translation d'adresses destination pour rediriger les paquets vers la machine locale. Cette option est très utilisée pour réaliser des proxies qui devront fonctionner de manière transparente pour les clients.

L'option « Packet mangling » active la gestion de la table mangle. Cette table a pour but de donner les moyens d'effectuer diverses modifications des en-têtes des paquets, afin de les marquer pour un traitement ultérieur spécifique.

L'option « TOS target support » permet d'utiliser la cible TOS dans les règles des chaînes de la table mangle. Cette cible autorise la modification du champ « Type Of Service » des paquets, principalement dans le but de modifier leur priorité.

L'option « MARK target support » active la gestion de la cible MARK, qui permet de marquer les paquets avec un traceur afin de pouvoir les identifier ultérieurement. Par exemple, il est possible de modifier le routage de certains paquets, selon qu'ils sont marqués ou non. On peut ainsi réaliser des liaisons prioritaires pour certaines catégories de paquets.

L'option « LOG target support » active la gestion de la cible LOG. Cette cible permet d'enregistrer des messages de débogage dans les fichiers de trace du système. On veillera à activer également la gestion des limites sur les règles de filtrages afin d'éviter d'engorger les fichiers de traces si l'on désire utiliser cette option.

L'option « TCPMSS target support » active la gestion de la cible TCPMSS, qui permet de modifier le champ MSS des paquets TCP. Cette cible est utilisée généralement pour réduire cette taille à la valeur fixée par le champ MTU des interfaces réseau, afin de prévenir la perte de paquets quand cette taille n'est pas correctement définie. Cela peut se

produire avec certains fournisseurs d'accès à Internet, dont les passerelles sont mal configurées.

L'option « `ipchains (2.2-style) support` » n'est disponible que si l'on utilise les modules du noyau pour Netfilter. Elle permet d'utiliser les outils de configuration des Firewall et des translations d'adresses ipchains, qui étaient utilisés avec les noyaux 2.2. de Linux. Il s'agit donc d'une option de compatibilité. On préférera dorénavant la commande iptables

L'option « `ipfwadm (2.0-style) support` » n'est disponible que si l'on utilise les modules du noyau pour Netfilter. Elle permet d'utiliser les outils de configuration des Firewall et des translations d'adresses ipfwadm, qui étaient utilisés avec les noyaux 2.0. de Linux. Il s'agit donc d'une option de compatibilité. On préférera dorénavant la commande iptables

10.7.12 Menu « IPv6: Netfilter Configuration »

L'option « `IP6 tables support (required for filtering/masq/NAT)` » permet d'activer la gestion de Netfilter pour le protocole IPv6.

L'option « `limit match support` » permet d'activer la gestion des limites sur le nombre de fois par seconde qu'une règle peut être vérifiée. Cette option permet par exemple d'éviter d'engorger les fichiers de traces du système lorsqu'un grand nombre de règles sont vérifiées.

L'option « `netfilter MARK match support` » permet d'activer la gestion du critère de sélection des paquets basé sur le champ MARK de leur en-tête. Ce champ peut être modifié par les règles de la chaîne `mangle`, comme pour le protocole IPv4.

L'option « `Packet filtering` » permet d'activer la gestion de la table `filter`, afin de réaliser par exemple un Firewall.

L'option « `Packet mangling` » permet d'activer la gestion de la table `mangle`, qui autorise la modification de certains champs des en-têtes des paquets IP.

L'option « `MARK target support` » permet de prendre en charge la gestion de la cible MARK, afin de marquer les paquets vérifiant certaines règles, par exemple pour effectuer un traitement ultérieur.

10.7.13 Menu « QoS and/or fair queueing »

L'option « `QoS and/or fair queuing (EXPERIMENTAL)` » permet d'activer les options de configuration des algorithmes qui fixent les priorités sur les paquets à envoyer sur le réseau.

Les options qui suivent permettent de choisir les algorithmes à utiliser pour la détermination des priorités d'émission. Parmi ces algorithmes, on retrouve les algorithmes basés sur la notion de qualité de service, qui peuvent être activés grâce à l'option « `QoS support` ». Les options suivantes permettent de paramétrer les notions attachées à la qualité de service.

10.7.14 Menu « Telephony Support »

L'option « `Linux telephony support` » permet d'activer la gestion des cartes téléphoniques. Ces cartes permettent de transférer les communications téléphoniques sur des supports numériques, par exemple en encapsulant les informations sonores sur un réseau TCP/IP. Cette option n'a rien à voir avec un modem, qui fait exactement l'inverse (transfert des données informatiques sur un réseau téléphonique analogique.

L'option « `QuickNet Internet LineJack/PhoneJack support` » active la prise en charge des cartes téléphoniques Quicknet.

10.7.15 Menu « ATA/IDE/MFM/RLL support »

L'option « `ATA/IDE/MFM/RLL support` » permet d'activer la gestion des disques durs et autres périphériques IDE. La réponse recommandée est '`Y`'. Les options spécifiques à chaque type de contrôleur sont accessibles dans le sous-menu « IDE, ATA and ATAPI Block devices ».

10.7.16 Menu « IDE, ATA and ATAPI Block devices »

L'option « `Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support` » permet d'activer la gestion des disques IDE. À moins que votre ordinateur ne soit complètement SCSI, répondez par '`Y`' ou '`M`' à cette question. Si vous répondez par '`M`', il faut que Linux soit installé sur un disque SCSI.

L'option « `Use old disk-only driver on primary interface` » permet d'utiliser un vieux driver pour les disques IDE connectés à la première interface IDE et qui poseraient quelques problèmes avec le driver général. Seule la première interface sera concernée par cette option, les autres interfaces utiliseront le nouveau driver.

L'option « `Include IDE/ATA-2 DISK support` » permet d'utiliser les disques durs IDE et ATAPI. Si l'ordinateur possède un disque IDE, il est fortement recommandé de répondre '`Y`' à cette question. On ne doit répondre '`N`' que si l'ordinateur est complètement SCSI. De plus, on prendra garde au fait que l'on ne peut pas mettre les fonctionnalités nécessaires au démarrage de l'ordinateur dans des modules. Cela signifie que si le disque de démarrage est un disque IDE, il ne faut pas mettre cette fonctionnalité dans un module.

L'option « `Use multi-mode by default` » permet d'activer la gestion du mode de transfert multiple par défaut. Cette option n'est nécessaire que pour quelques cas particuliers.

Il suit un certain nombre d'options permettant d'activer des fonctionnalités spécifiques aux principales grandes marques de disque dur. Ces options ne sont pas encore disponibles, et ne sont pas encore documentées. Elles ne seront donc pas décrites ici.

L'option « `PCMCIA IDE support` » permet d'activer la gestion des périphériques IDE connectés par un port PCMCIA sur les portables. Cette option n'est accessible que si vous avez activé la gestion des cartes PCMCIA avec l'option « `CardBus support` » du menu « PCMCIA/CardBus support ».

L'option « `Include IDE/ATAPI CDROM support` » permet d'utiliser les CD-ROM IDE et ATAPI. La plupart des ordinateurs possèdent un lecteur de CD-ROM ATAPI actuellement, il est donc recommandé d'activer cette fonctionnalité. On essaiera d'utiliser

cette fonctionnalité sous forme de module de préférence, parce que cette fonctionnalité n'est pas nécessaire en permanence. Cependant, si vous disposez d'un graveur de CD-ROM, il est recommandé de ne pas activer cette fonctionnalité. L'accès aux périphériques IDE ATAPI se fera alors par l'intermédiaire de l'émulateur de périphériques SCSI. Si vous disposez d'un lecteur de CD-ROM IDE et que ne voulez pas utiliser l'émulation SCSI, vous pouvez répondre 'Y' à cette question..

L'option « `Include IDE/ATAPI TAPE support` » permet d'inclure la gestion des périphériques de sauvegarde à bande IDE et ATAPI (streamers). Peu de gens disposent de tels périphériques, et ceux qui en ont un ne l'utilisent que pour les sauvegardes.

L'option « `Include IDE/ATAPI FLOPPY support` » permet d'inclure la gestion des lecteurs de disques amovibles IDE et ATAPI. C'est en particulier le cas pour les lecteurs LS120 et ZIP. Comme les lecteurs de disques amovibles IDE sont assez rares.

L'option « `SCSI emulation support` » permet d'émuler la présence d'un périphérique SCSI par un périphérique IDE en convertissant les requêtes SCSI en requêtes ATAPI. Cette fonctionnalité est utile lorsqu'on veut utiliser certains logiciels bas niveau qui ne peuvent travailler qu'avec des périphériques SCSI. C'est en particulier le cas des logiciel de gravage de CD. On notera que cette fonctionnalité ne sera pas activable si le pilote IDE est actif. Il est donc nécessaire d'utiliser celui-ci sous la forme de module, ou de ne pas l'utiliser du tout.

L'option « `CMD640 chipset bugfix/support` » permet de contourner une bogue des chipsets IDE CMD640. Vous ne devez activer cette option que si vous disposez d'un tel chipset.

L'option « `CMD640 enhanced support` » permet l'autodétection des paramètre idéaux pour les chipset IDE CMD640. Cette option n'est disponible que si vous avez activé le support des chipsets CMD640 dans la question précédente

L'option « ISA-PNP EIDE support » permet de prendre en charge la gestion des cartes ISA Plug and Play prenant en charge des disques EIDE additionnels. Cette option peut être nécessaire si ces cartes doivent être initialisées avant de chercher à utiliser les disques qui y sont connectées.

L'option « `RZ1000 chipset bugfix/support` » permet de contourner une bogue des chipsets RZ1000. Vous ne devez activer cette option que si vous disposez d'un tel chipset.

L'option « `Generic PCI IDE chipset support` » permet d'activer la gestion des périphériques IDE sur bus PCI. Vous devez répondre par 'Y' à cette question si vous disposez d'une carte mère PCI et de contrôleurs IDE.

L'option « `Sharing PCI IDE interrupts support` » permet d'activer le partage des lignes d'interruptions utilisées par les contrôleurs IDE avec les cartes présentes sur le bus PCI. Cette fonctionnalité nécessite un support matériel particulier de la part du contrôleur IDE, support qui n'est présent que sur certaines cartes mères. En général, l'activation de cette option ne perturbe pas le fonctionnement du système sur les ordinateurs incapables d'effectuer le partage des lignes d'interruptions des contrôleurs IDE.

L'option « Generic PCI bus-master DMA support » permet d'activer la gestion des disques Ultra DMA. La réponse recommandée est 'Y' si vous disposez d'une carte mère PCI gérant l'Ultra DMA et de disques IDE.

L'option « Boot off-board chipsets first support » permet d'inverser la numérotation des disques IDE connectés aux contrôleurs de la carte mère et des disques IDE connectés aux contrôleurs additionnels que l'on peut avoir sur une carte fille

L'option « Use PCI DMA by default when available » permet d'activer la gestion de l'Ultra DMA au démarrage. La réponse recommandée est 'Y', sauf si l'on veut désactiver le support de l'Ultra DMA.

L'option « ATA Work(s) In Progress (EXPERIMENTAL) » donne l'accès à des fonctionnalités extrêmement expérimentales concernant les disques IDE.

Il suit un certain nombre d'options qui permettent d'activer un support étendu pour différents chipsets. Il est recommandé de répondre par 'N' à ces questions, sauf à celles correspondant aux chipsets effectivement présents sur votre carte mère.

L'option « IGNORE word93 Validation BITS » permet d'éviter une inconsistance dans les spécifications du protocole matériel ATAPI. Ces spécifications n'ont pas été claires à un endroit, et il existe maintenant des différences mineures entre les différents chipsets présents sur le marché. Cette option permet de désactiver cette fonctionnalité ambiguë.

10.7.17 Menu « SCSI support »

L'option « SCSI support » permet d'activer la prise en charge des périphériques SCSI. Il faut répondre par 'Y' ou 'M' à cette question si vous disposez de tels périphériques. De plus, si le noyau se trouve sur un disque SCSI ou s'il a besoin de composants se trouvant sur le disque SCSI pour s'amorcer, il ne faut pas mettre en module cette fonctionnalité. Ceux qui n'ont pas de périphériques SCSI et ne comptent pas en utiliser peuvent répondre 'N' à cette question. En revanche, ceux qui disposent d'un périphérique ATAPI pour lequel ils n'ont pas de drivers, et pour lequel ils utilisent la couche d'émulation SCSI, doivent activer cette fonctionnalité. C'est en particulier le cas si vous avez un graveur de CD. Dans ce cas, il est recommandé de répondre 'Y' à cette question. Même dans les autres cas, il est conseillé de répondre par 'M' à cette question, afin de se réserver la possibilité de connecter ultérieurement un lecteur ZIP connecté sur port parallèle, qui nécessitera alors le support SCSI.

L'option « SCSI disk support » permet d'activer la gestion des disques durs SCSI. Il ne faut pas répondre par 'M' si votre système a besoin d'un disque SCSI pour démarrer. Cette option permet également de gérer les lecteurs ZIP connectés sur port parallèle.

L'option « Maximum number of SCSI disks that can be loaded as modules » permet de fixer le nombre maximal de disques SCSI qui pourront être utilisés si les drivers SCSI du noyau sont chargés en tant que modules. Il est recommandé de laisser la valeur par défaut dans ce champ.

L'option « SCSI tape support » permet d'activer la gestion des périphériques à bande SCSI. Notez que les lecteurs de bande OnStream SC-x0 ne sont pas pris en charge par ce driver générique, mais peuvent être utilisés malgré tout en activant l'option suivante « SCSI

`OnStream SC-x0 tape support` ». Le driver alors utilisé nécessitera la présence des fichiers spéciaux de périphérique `/dev/osstX`, qui utilisent le numéro de code majeur 206.

L'option « `SCSI CD-ROM support` » permet d'activer la gestion des CD-ROM SCSI. Vous devez activer cette fonctionnalité également si vous utilisez l'émulation SCSI pour un graveur de CD ATAPI. Si vous êtes dans l'un de deux ces cas, il est recommandé de répondre '`Y`' à cette question.

L'option « `Enable vendor-specific extensions (for SCSI CDROM)` » permet d'activer la gestion de CD-ROM SCSI disposant de commandes SCSI spécifiques. C'est notamment le cas pour les CD multisessions NEC/TOSHIBA et les graveurs de CD HP.

L'option « `Maximum number of CDROM devices that can be loaded as modules` » permet de fixer le nombre maximal de lecteurs de CDROM SCSI qui pourront être utilisés si les drivers du noyau sont chargés en tant que modules. Il est recommandé de laisser la valeur par défaut dans ce champ.

L'option « `SCSI generic support` » permet d'activer la gestion des périphériques SCSI non standards. Pour ces périphériques, il faut utiliser un programme capable d'envoyer les commandes SCSI appropriées à votre matériel. Il faut activer cette fonctionnalité pour la plupart des périphériques SCSI qui ne sont ni des disques, ni des lecteurs de CD-ROM, ni des lecteurs de bandes. C'est en particulier le cas si vous utilisez un graveur de CD-ROM. Dans ce cas, la réponse recommandée est '`Y`'.

L'option « `Enable extra checks in new queueing code` » permet d'activer des contrôles supplémenaires dans les drivers SCSI. Si une erreur est détectée, le système s'arrêtera en catastrophe immédiatement, limitant ainsi les risques de pertes de données ultérieurs. Si vous n'activez pas cette option, le système continuera à fonctionner normalement, mais les dégâts risquent d'être beaucoup plus importants à terme si une erreur se produit.

L'option « `Probe all LUNs on each SCSI device` » permet d'effectuer la détection de tous les numéros logiques d'unités SCSI de chaque périphérique. Comme la plupart des périphériques SCSI ne disposent que d'un seul numéro d'unité logique.

L'option « `Verbose SCSI error reporting (kernel size +=12K)` » permet d'utiliser un jeu de messages d'erreurs alternatif pour le SCSI. Ces messages sont plus lisibles, mais prennent plus de place dans le noyau.

L'option « `SCSI logging facility` » permet d'activer les traces du sous-système SCSI.

10.7.18 Menu « SCSI low-level drivers »

Ce jeu d'options permet de sélectionner le driver bas niveau SCSI adapté à votre matériel. Il faut connaître la marque et le modèle de votre adaptateur SCSI, ainsi que ses paramètres pour répondre à ces questions. Si vous utilisez l'émulation SCSI pour les périphériques ATAPI, il n'est pas nécessaire de choisir un driver bas niveau pour ceux-ci.

10.7.19 Menu « PCMCIA SCSI adapter support »

L'option « `PCMCIA SCSI adapter support` » permet la prise en charge les périphériques SCSI au format PCMCIA. Les options qui suivent correspondent aux drivers des différents

types de matériels SCSI au format PCMCIA que Linux est capable de gérer. Vous devez donc choisir les drivers correspondants à votre matériel.

10.7.20Menu « IEEE 1394 (FireWire) support »

L'option « `IEEE 1394 (FireWire) support (EXPERIMENTAL)` » permet d'activer la gestion des cartes FireWire. Ces cartes fournissent l'accès à un bus de données extrêmement rapide, que l'on utilise généralement pour connecter des périphériques exigeant une bande passante très élevée, comme les caméras vidéo par exemple.

L'option « `Texas Instruments PCILynx support` » permet d'activer la gestion des drivers pour les cartes FireWire PCILynx de Texas Instruments.

L'option « `Use PCILynx local RAM` » permet d'utiliser la mémoire vive embarquée dans certaines cartes PCILynx de Texas Instruments afin d'améliorer les performances. Ce type de carte est très rare, et ce driver ne fonctionne pas du tout avec les cartes qui ne disposent pas de mémoire vive embarquée.

L'option « `Support for non-IEEE1394 local ports` » permet d'accéder aux ressources matérielles des cartes PCILynx par l'intermédiaire de fichiers spéciaux de périphériques dédiés, au lieu de passer par l'interface FireWire standard.

L'option « `OHCI (Open Host Controller Interface) support` » active la prise en charge des contrôleurs IEEE 1394 respectant les spécifications OHCI (il s'agit d'un standard de communication pour les contrôleurs). Ce driver n'a été testé qu'avec un contrôleur de Texas Instruments, mais ce contrôleur est l'un des plus utilisés du marché.

L'option « `Video 1394 support` » n'est pas documentée et ne sera pas décrite dans ce document.

L'option « `Raw IEEE 1394 I/O support` » permet aux programmes de communiquer directement avec le matériel IEEE 1394. C'est en général le mode de fonctionnement désiré.

L'option « `Excessive debugging support` » est réservée pour les développeurs de drivers. Elle permet de stocker sur disque toutes les informations transitant sur le bus FireWire, ce qui sature généralement les disques durs extrêmement rapidement

10.7.21Menu « I2O support »

L'option « `I2O support` » permet d'activer la gestion des cartes d'entrée/sortie I2O. Ces cartes prennent en charge la gestion des entrées/sorties de manière uniforme, pour tous les périphériques matériels, et permettent donc d'écrire un driver uniforme au niveau du système d'exploitation pour toute une classe de périphériques. Ce driver uniforme (« OSM », abréviation de « Operating System Module ») communique avec des drivers spécifiques à chaque type de matériel, qui ne font aucune hypothèse sur le système d'exploitation utilisé. Ainsi, il est possible d'écrire des drivers communs à tous les systèmes d'exploitations. Si vous répondez par '`Y`', vous devrez choisir le type de carte I2O installée et la liste des drivers I2O OSM que vous voulez utiliser.

10.7.22Menu « Network device support »

L'option « Network device support » permet d'activer la gestion des diverses interfaces réseau que Linux peut prendre en charge. On notera que les choix faits pour les protocoles de communication dans le menu « Networking options » sont nécessaires (la plupart des composants des systèmes Unix utilisent les fonctions réseau du système pour communiquer) mais non suffisants pour accéder au monde extérieur. Si l'on n'active pas cette option, seuls les processus de la machine locale seront accessibles (par l'intermédiaire de l'interface réseau « loopback »). Pour accéder aux autres ordinateurs, que ce soit par une carte réseau, un câble parallèle ou série, ou par modem via un fournisseur d'accès à Internet, il faut en plus répondre 'Y' à cette question.

L'option « Dummy net driver support » permet d'activer la fonctionnalité de réseau virtuel. Cette fonctionnalité fournit une interface virtuelle sur un réseau ne contenant qu'une seule machine. Cette interface peut être utilisée pour faire croire à des programmes réseaux que la machine est effectivement connectée à un réseau.

L'option « Bonding driver support » permet de réunir plusieurs connexions Ethernet entre deux machines pour simuler une connexion unique, dont la bande passante est la somme des bandes passantes des connexions ainsi regroupées.

L'option « EQL (serial line load balancing) support » permet d'activer la gestion de connexions multiples sur plusieurs connexions série (par exemple par l'intermédiaire de deux modems). Cette fonctionnalité nécessite également le support de cette fonctionnalité du côté de la machine distante.

L'option « Universal TUN/TAP device driver support » permet d'activer la gestion d'interfaces réseau virtuelles tunX et tapX (où 'X' est le numéro de chaque interface), dont les données peuvent être lues et écrites directement par des applications normales, par l'intermédiaire de fichiers spéciaux de périphériques /dev/tunX et /dev/tapX. Les interfaces de type tun se comportent exactement comme des interfaces réseau point à point (elles ne permettent donc de communiquer qu'avec une seule autre machine), alors que les interfaces de type tap simulent le fonctionnement d'une carte Ethernet classique. Cette fonctionnalité permet donc aux programmes classiques d'accéder aux paquets routés vers ces interfaces, et de recevoir et d'envoyer des données brutes sur le réseau. Les interfaces tunX et tapX sont créées dynamiquement, lorsque les programmes qui désirent les utiliser s'enregistrent au niveau du système en ouvrant le fichier spécial de périphérique /dev/net/tun, de type caractère et de codes majeur et mineur égaux respectivement à 10 et 200.

L'option « Ethertap network tap (OBSOLETE) » n'est activée que si l'option « Kernel/User netlink socket » a été activée. Elle permet de gérer un fichier spécial de type block et de numéro majeur 36 et de numéro mineur 16, qui fournit les données brutes d'une interface Ethernet. Ce fichier peut être utilisé par un programme pour envoyer et recevoir des trames Ethernet directement sur ce pseudo périphérique au plus bas niveau. Cette fonctionnalité est devenue obsolète depuis les noyaux 2.4, car ceux-ci proposent le mécanisme plus général de l'option « Universal TUN/TAP device driver support ».

L'option « General Instruments Surfboard 1000 » permet d'activer la gestion des cartes Sufboard 1000, qui sont généralement utilisées pour connecter un modem câble à

l'ordinateur. Ces modems permettent de recevoir des informations en provenance d'Internet à haut débit, mais ne permettent pas d'envoyer d'informations à destination d'Internet. Le canal montant doit donc toujours être réalisé par une liaison téléphonique classique

L'option « `FDDI driver support` » permet d'activer la gestion des cartes Ethernet FDDI. Les options qui suivent permettent de sélectionner les drivers pour les cartes FDDI gérées par Linux.

L'option « `HIPPI driver support (EXPERIMENTAL)` » permet la gestion des cartes HIPPI. Les options qui suivent permettent de sélectionner les drivers pour les cartes de ce type.

L'option « `PLIP (parallel port) support` » permet d'activer les connexions par port parallèle. Ce type de connexion peut être utilisé pour transférer des fichiers par un câble parallèle entre deux ordinateurs.

L'option « `PPP (point-to-point protocol) support` » permet d'activer la gestion des connexions PPP. Ces connexions sont utilisées pour accéder à Internet via un modem, ou pour établir une connexion de manière plus générale via un câble série. Il est conseillé d'activer le support de ce type de connexion, à moins que vous soyez sûr de ne jamais vous connecter à Internet.

L'option « `PPP multinlink support (EXPERIMENTAL)` » permet d'utiliser plusieurs lignes PPP pour augmenter la bande passante en les agrégeant. Par exemple, si vous disposez de plusieurs lignes téléphoniques, vous pouvez vous connecter plusieurs fois à votre fournisseur d'accès et accroître ainsi votre bande passante. Bien entendu, pour que cette technique fonctionne, il faut que le fournisseur d'accès l'autorise.

L'option « `PPP filtering` » permet d'autoriser le filtrage des paquets traversant les interfaces PPP existantes dans le système. Cette fonctionnalité peut être utile pour réaliser un firewall ou pour effectuer des actions spécifiques lorsque certains paquets sont reçus.

L'option « `PPP support for async serial ports` » permet de prendre en charge les communications PPP sur les câbles série classiques. C'est en général l'option qui convient lorsqu'on veut se connecter à Internet à l'aide d'un modem classique. Notez que cette option n'est pas utilisable pour les connexions à l'aide d'un modem RNIS.

L'option « `PPP support for sync tty ports` » permet de prendre en charge les communications PPP avec les adaptateurs synchones, tels que les cartes SyncLink.

L'option « `PPP Deflate compression` » permet de prendre en charge l'algorithme de compression Deflate (le même algorithme que celui utilisé par le programme **gzip**) pour comprimer les données transmises sur les connexions PPP.

L'option « `PPP BSD-Compress compression` » permet de prendre en charge l'algorithme de compression LZW pour comprimer les données transmises sur les connexions PPP. Cet algorithme est soumis à un brevet logiciel, mais ces brevets sont illégaux en Europe et il est tout à fait légal de l'utiliser.

L'option « PPP over Ethernet (EXPERIMENTAL) » permet de réaliser des connexions point à point sur un réseau de type Ethernet.

L'option « SLIP (serial line) support » permet de connecter deux ordinateurs par une ligne série ou par un modem. Les options suivantes permettent de fixer les paramètres des connexions SLIP. Le protocole SLIP est en train de tomber en désuétude et est remplacé par PPP.

L'option « CSLIP compressed headers » permet d'activer la compression des en-têtes IP pour le protocole SLIP.

L'option « Keepalive and linefill » permet d'activer la surveillance de la connexion SLIP. Cette option est utilisée sur les connexions SLIP passant par des lignes analogiques de mauvaise qualité.

L'option « Six bit SLIP encapsulation » permet de faire en sorte que le protocole SLIP n'envoie que des données codées sur 6 bits. Cela permet d'améliorer la qualité des transmissions sur les réseaux peu fiables, ou qui ne peuvent pas transférer plus 7 bits de données.

L'option « Fibre Channel driver support » permet d'activer la gestion des adaptateurs Fiber Channel. L'option qui suit donne la possibilité de sélectionner le driver pour les cartes Interphase à base de chipset Tachyon

L'option « Red Creek Hardware VPN (EXPERIMENTAL) » permet d'activer la gestion des cartes permettant de créer des réseaux privés virtuels.

L'option « Traffic Shaper (EXPERIMENTAL) » permet d'activer la possibilité de contrôler le débit maximal de données à travers une interface réseau.

10.7.23Menu « ARCnet devices »

L'option « ARCnet support » permet d'activer la gestion des cartes de type ARCnet. Les options qui suivent correspondent aux différents drivers pour les différentes cartes de ce type. Si vous avez une carte de ce type, répondez par 'M' ou 'Y' à cette question et choisissez le driver correspondant à votre matériel dans la suite du menu.

10.7.24Menu « AppleTalk devices »

L'option « Appletalk interfaces support » active la gestion des interfaces réseau Appletalk. Ce sont des interfaces réseau permettant de communiquer sur les réseaux Apple d'ancienne génération.

Les options « Apple/Farallon LocalTalk PC support » et « COPS LocalTalk PC support » permettent de gérer les adaptateurs LocalTalk pour PC. Les options qui suivent permettent de choisir les drivers appropriés à votre matériel.

L'option « AppleTalk-IP driver support » permet de se connecter à un réseau utilisant le protocole IP via un adaptateur réseau AppleTalk. Cette fonctionnalité est expérimentale.

L'option « IP to Appletalk-IP Encapsulation support » permet d'encapsuler les paquets IP dans des paquets AppleTalk. Cette fonctionnalité est utilisée par les machines Linux qui n'ont accès qu'à un réseau Appletalk.

L'option « Appletalk-IP to IP Decapsulation support » permet d'extraire les paquets IP encapsulés dans des paquets IP. Cette fonctionnalité est utilisée par les machines Linux qui font office de passerelles vers Internet pour les machines d'un réseau Appletalk.

10.7.25Menu « Ethernet (10 or 100Mbit) »

L'option « Ethernet (10 or 100Mbit) » permet d'activer la gestion des cartes de type Ethernet. Les options qui suivent correspondent aux différents drivers pour les différentes cartes de ce type. Si vous avez une carte de ce type, répondez par 'M' ou 'Y' à cette question et choisissez le driver correspondant à votre matériel dans la suite du menu. La réponse recommandée est 'N'. La plupart des cartes réseau sont compatibles NE2000. Il existe deux types de drivers pour ces cartes, selon qu'elles sont ISA ou non. Le driver ISA peut être choisi en activant les options « Other ISA cards » et « NE2000/NE1000 support ». Le driver pour les cartes PCI peut être sélectionné en activant les options « EISA, VLB, PCI and on board controllers » et « PCI NE2000 support ».

10.7.26Menu « Ethernet (1000 Mbit) »

Les options disponibles dans ce menu permettent de choisir les drivers à utiliser pour les cartes Ethernet Gigabit. Choisissez l'option qui correspond à votre matériel si vous en disposez d'une.

10.7.27Menu « Wireless LAN (non-hamradio) »

L'option « Wireless LAN (non-hamradio) » permet d'activer la gestion des connexions sans fil (mais qui ne sont pas gérées par les fonctionnalités radio-amateurs). Les options qui suivent permettent de choisir les drivers disponibles.

L'option « STRIP (Metricom starmode radio IP) » permet d'activer la gestion des adaptateurs réseau radio Metricom.

L'option « AT&T WaveLAN & DEC RoamAbout DS support » permet d'activer la gestion des adaptateurs réseau radio WaveLAN.
L'option « Aironet Arlan 655 & IC2200 DS support » permet d'activer la gestion des adaptateurs réseau radio Aironet 655 et IC2200 DS.

L'option « Aironet 4500/4800 series adapters » permet d'activer la gestion des adaptateurs réseau radio Aironet de la série 4500/4800. Les options qui suivent correspondent aux différents types de cartes de cette série. L'option « Aironet 4500/4800 PROC interface » donne la possibilité de configurer ces cartes par l'intermédiaire du système de fichiers virtuels /proc/.

10.7.28Menu « Token ring devices »

L'option « Token Ring driver support » permet d'activer la gestion des réseaux Token Ring. Les options qui suivent permettent de choisir les drivers pour ces adaptateurs.

10.7.29Menu « Wan interfaces »

L'option « `Comtrol Hostess SV-11 support` » permet d'activer la gestion des cartes Comtrol.

L'option « `COSA/SRP sync serial boards support` » permet d'activer la gestion des cartes COSA et SRP.

L'option « `MultiGate (COMX) synchronous serial boards support` » permet d'activer la gestion des drivers des cartes MultiGate. Les options qui suivent permettent de paramétrer les fonctionnalités supportées par le driver.

L'option « `Etinc PCISYNC serial board support (EXPERIMENTAL)` » n'est pas encore documentée et ne sera pas décrite plus en détail ici.

L'option « `LanMedia Corp. SSI/V.35, T1/E1, HSSI, T3 boards` » permet d'activer la gestion des drivers des cartes LanMedia.

L'option « `Sealevel Systems 4021 support` » permet d'activer la gestion des drivers des cartes Sealevel ACB 56.

L'option « `SyncLink HDLC/SYNCPPP support` » n'est pas documentée à l'heure actuelle.

L'option « `Generic HDLC driver` et ses sous-options ne sont pas encore documentées et ne seront donc pas décrites plus en détail ici.

L'option « `Frame relay DLCI support` » permet d'activer la gestion du protocole Frame Relay. Les options qui suivent permettent de configurer le driver.

L'option « `WAN router drivers` » permet d'activer les options de menu suivantes, qui donnent la possibilité de faire la configuration des adaptateurs WAN.

L'option « `Sangoma WANPIPE(tm) multiprotocol cards` » permet d'activer la gestion des cartes WAN Sangoma. Les options qui suivent permettent de configurer les paramètres pour ce driver.

L'option « `LAPB over Ethernet driver` » permet d'activer un driver pour une carte X.25 virtuelle.

L'option « `X.25 async driver` » permet d'activer la gestion du protocole X.25 sur une ligne série normale ou sur un modem.

L'option « `SBNI12-xx support` » permet d'activer la gestion des cartes SBNI12-xx.

10.7.30Menu « PCMCIA network device support »

L'option « `PCMCIA network device support` » permet d'activer la gestion des cartes réseau au format PCMCIA. Les options qui suivent correspondent aux divers drivers pris en charge par Linux.

10.7.31Menu « ATM drivers »

L'option « ATM over TCP » permet d'encapsuler ATM sur un réseau TCP/IP. Ce type d'encapsulation n'est utile qu'à des fins de test et de développement.

L'option « Efficient Networks ENI155P » active la gestion des cartes ATM ENI155P d'Efficient Networks et Power155 de SMC. Les options qui suivent permettent de configurer le driver.

L'option « Fujitsu FireStream (FS50/FS155) » active la gestion des cartes ATM FireStream de Fujitsu

L'option « ZeitNet ZN1221/ZN1225 » active la gestion des cartes ATM ZN1221 et ZN1225 de ZeitNet. Les options qui suivent permettent de configurer le driver.

L'option « IDT 77201 (NICStAR) (ForeRunnerLE) » active la gestion des cartes ATM basées sur le chipset NICStAR. Les options qui suivent permettent de configurer le driver.

L'option « Madge Ambassador (Collage PCI 155 Server) » active la gestion des cartes ATM basées sur le chipset ATMizer. L'option qui suit permet d'activer les informations de débogage du driver.

L'option « Madge Horizon [Ultra] (Collage PCI 25 and Collage PCI 155 Client) » active la gestion des cartes ATM basées sur le chipset Horizon. L'option qui suit permet d'activer les informations de débogage du driver.

L'option « Interphase ATM PCI x575/x525/x531 » active la gestion des cartes ATM ChipSAR d'Interphase. L'option qui suit permet d'activer les informations de débogage du driver.

L'option « FORE Systems 200E-series » active la gestion des cartes ATM FORE Systems 200E. Les options qui suivent permettent de configurer le driver.

10.7.32Menu « Amateur Radio support »

L'option « Amateur Radio support » permet d'activer la gestion des communications Radio de Linux. Les options qui suivent permettent de préciser les options du protocole réseau utilisé pour ces communications.

10.7.33Menu « AX.25 network device drivers »

Ce menu permet de choisir les drivers bas niveau à utiliser pour le support des communications Radio. Il faut choisir le driver correspondant à votre matériel.

10.7.34Menu « IrDA subsystem support »

L'option « IrDA subsystem support » permet d'activer la gestion des périphériques infrarouges.

Il faut choisir le protocole de communication que vous désirez dans les options qui suivent. Les dernière options permettent de choisir les options de ces protocoles.

10.7.35Menu « Infrared-port device drivers »

Ces options permettent de choisir les drivers bas niveau pour les périphériques infrarouges. Vous devez choisir le driver correspondant à votre matériel.

10.7.36Menu « ISDN subsystem »

L'option « `ISDN support` » permet d'activer la gestion des interfaces « ISDN » (« Numéris » en France). Si vous disposez d'une telle interface.

L'option « `Support synchronous PPP` » permet d'utiliser une version de PPP qui ne gère pas les synchronisations dans la communication entre les deux ordinateurs. Cette fonctionnalité n'est en effet plus nécessaire avec ISDN, puisque c'est un protocole numérique.

L'option « `Use VJ-compression with synchronous PPP` » permet d'utiliser l'algorithme de Van Jacobson pour la compression des en-têtes IP dans le protocole PPP synchrone.

L'option « `Support generic MP (RFC 1717)` » permet de regrouper plusieurs connexions ISDN avec le protocole PPP synchrone, et ce afin d'accroître la bande passante.

L'option « `Support BSD compression (module only)` » active la gestion de la compression de donnée avec l'algorithme BSD. Cette fonctionnalité n'est disponible que sous la forme de module pour des raisons de licences logicielles.

L'option « `Support audio via ISDN` » permet d'activer la gestion des commandes vocales dans l'émulateur de modem du sous-système ISDN.

L'option « `Support AT-Fax Class 1 and 2 commands` » permet d'activer la gestion des commandes vocales des FAX.

L'option « `X.25 PLP on top of ISDN` » permet d'activer l'encapsulation du protocole X.25 dans le protocole ISDN.

10.7.37Menu « ISDN feature submodules »

L'option « `isdnloop support` » permet d'activer la gestion d'une interface ISDN virtuelle, afin de tester la configuration sans effectuer d'appel réel.

L'option « `Support isdn diversion services` » permet d'activer la gestion de services ISDN supplémentaires

10.7.38Menu « Passive ISDN cards »

L'option « `HiSax SiemensChipSet driver support` » permet d'activer les options de configuration pour les cartes à base de chipsets Siemens. Ce sont les cartes les plus courantes. Les options qui suivent permettent de sélectionner le type de matériel correspondant.

10.7.39Menu « Active ISDN cards »

L'option « `ICN 2B and 4B support` » permet d'activer le support pour des cartes ISDN fabriquées par la société ICN.

L'option « PCBIT-D support » permet d'activer la gestion des cartes ISDN fabriquées par la société Octal.

L'option « Spellcaster support » permet d'activer la gestion des cartes ISDN Spellcaster.

L'option « IBM Active 2000 support » permet d'activer la gestion des cartes ISDN IBM Active 2000.

L'option « Eicon active card support » permet d'activer la gestion des cartes Eicon. Les options qui suivent permettent de choisir le driver approprié au matériel dont vous disposez.

L'option « CAPI2.0 support » permet d'activer la gestion de l'interface de programmation CAPI (« Common ISDN Application Programming Interface », permettant aux programmes clients d'utiliser de manière uniforme toutes les cartes ISDN.

L'option « Verbose reason code reporting (kernel size +=7K) » permet d'indiquer le degré de traces utilisé par le driver des cartes AVM.

L'option « CAPI2.0 Middleware support (EXPERIMENTAL) » n'est pas encore documentée et ne sera pas décrite dans ce document.

L'option « CAPI2.0 /dev/capi support » n'est pas documentée et ne sera pas décrite ici.

L'option « CAPI2.0 filesystem support » n'est pas encore documentée et ne sera pas décrite dans ce document.

L'option « CAPI2.0 capidrv interface support » n'est pas documentée et ne sera pas décrite ici.

Les options qui suivent permettent de prendre en charge les cartes AVM.

L'option « Hypercope HYSDN cards (Champ, Ergo, Metro) support (module) » permet de prendre en charge les cartes ISDN actives d'Hypercope.

L'option « HYSDN CAPI2.0 support » permet de gérer l'interface de programmation CAPI 2.0 pour les cartes Hypercope..

10.7.40Menu « Old CD-ROM drivers (not SCSI, not IDE) »

L'option « Support non-SCSI/IDE/ATAPI CDROM drives » permet d'activer la gestion des lecteurs de CD-ROM non SCSI et non IDE. Ce sont essentiellement des vieux lecteurs de CD-ROM, parfois connectés sur une carte son par une interface propriétaire. Les options qui suivent permettent de choisir le driver qui correspond au matériel dont vous disposez.

10.7.41Menu « Input Core Support »

L'option « Input core support » permet d'activer un nouveau mécanisme de gestion des périphériques de saisie et d'entrée interactifs. Ce mécanisme permettra à terme de gérer tous les périphériques d'entrée de manière uniforme, et donc de simplifier la gestion des

données provenant de l'utilisateur. Les périphériques déjà capables d'utiliser ce nouveau mécanisme comprennent les joysticks et certains périphériques USB. Vous devez activer cette fonctionnalité si vous désirez connecter un joystick, un clavier ou une souris USB sur votre ordinateur.

L'option « `Keyboard support` » permet de prendre en charge les claviers USB en redirigeant les données provenant de ces claviers vers le traitement classique du clavier de Linux. Si vous disposez d'un tel clavier, il n'y a normalement rien à faire pour l'utiliser, puisque le BIOS assure la compatibilité ascendante des claviers USB connectés sur les ports USB de la carte mère. Cependant, cette technique ne fonctionne pas pour les claviers connectés sur des cartes USB filles, et vous pouvez avoir besoin d'activer la gestion de ces claviers ici. La réponse recommandée est '`N`', sauf si vous désirez connecter un clavier USB sur votre ordinateur et qu'il ne fonctionne pas nativement. Notez que cette option est incompatible avec l'option « `USB HIDBP Keyboard (basic) support` » du menu « `USB support` », qui permet de gérer les claviers USB d'une manière plus légère afin de réduire la taille du noyau dans les systèmes embarqués.

De la même manière, l'option « `Mouse support` » permet de prendre en charge les souris USB en simulant une souris PS/2 classique, via le fichier spécial de périphérique `/dev/input/mice`, de codes majeur et mineur 13 et 63. Ce driver permet également d'accéder aux souris USB via les fichiers spéciaux de périphériques `/dev/input/mouseN`, où `N` est le numéro de la souris, et dont les numéros de codes majeur et mineurs sont respectivement 13 et 32 à 63. Ce driver est également incompatible avec le driver simplifié que l'on peut activer avec l'option « `USB HIDBP Mouse (basic) support` » du menu « `USB support` ».

Les options « `Horizontal screen resolution` » et « `Vertical screen resolution` » permettent de définir la résolution horizontalement et verticalement de l'écran, afin de permettre l'utilisation d'une tablette de digitalisation USB comme une souris. Ces deux données permettent de faire la conversion entre les coordonnées de la tablette et les coordonnées de l'écran.

L'option « `Joystick support` » active la prise en charge des joysticks USB et permet aux applications d'y accéder au travers des fichiers spéciaux de périphériques `/dev/input/jsN`, où `N` est le numéro du joystick, et dont les numéros de codes majeur et mineurs sont respectivement 13 et 0 à 31. Notez que cette option n'est pas nécessaire pour la prise en charge des joystick non-USB, bien que l'activation de l'option « `Input core support` » restent obligatoire.

L'option « `Event interface support` » active la gestion des fichiers spéciaux `/dev/input/eventN`, où `N` est le numéro du périphérique, qui permettent de lire les événements provenant des périphériques d'entrée USB de manière générique. Ces fichiers peuvent être utilisés par plusieurs périphériques distincts, et permettront à terme de connecter plusieurs périphériques d'entrée sur la même machine, tout en permettant une gestion uniforme de la part des applications. Actuellement, très peu de programmes utilisent cette fonctionnalité, et il est encore rare de voir des machines à plusieurs claviers et plusieurs souris. À terme cependant, cette fonctionnalité sera certainement utilisée pour permettre l'utilisation d'une même machine par plusieurs personnes (il suffit pour cela de développer un driver pour les serveurs X de XFree86 capable de lire les données provenant de ces fichiers spéciaux de périphériques).

10.7.42Menu « Character devices »

L'option « Virtual terminal » permet d'activer la gestion des terminaux virtuels, accessibles avec les combinaisons de touche ALT+Fx (ou CTRL+ALT+Fx sous XWindow). Il faut au moins un terminal virtuel pour pouvoir utiliser le clavier et l'écran de la machine.

L'option « Support for console on virtual terminal » permet de placer la console système (les flux standards du noyau, et le login en mode mono utilisateur) sur un terminal virtuel. Si vous répondez 'N' à cette question, la console ne sera pas accessible avec le clavier et l'écran.

L'option « Standard/generic (8250/16550 and compatible UARTs) serial support » permet d'activer la gestion des ports série classiques. Vous pouvez vous en passer si vous n'avez ni modem ni souris série, et si vous ne désirez pas utiliser le port série du tout.

L'option « Support for console on serial port » permet d'activer la redirection de la console système sur un port série. En général, la console est redirigée vers un terminal virtuel.

L'option « Extended dumb serial driver options » permet d'activer les options de configurations étendues pour les ports série classiques.

L'option « Support more than 4 serial ports » permet d'activer la gestion de ports série additionnels

.

L'option « Support for sharing serial interrupts » permet d'activer la gestion des interruptions de plusieurs ports série d'une même carte par un seul canal d'interruption. Il faut que la carte série gère le partage des interruptions pour pouvoir utiliser cette option..

L'option « Autodetect IRQ on standard ports (unsafe) » permet de demander au noyau de tenter une détection automatique de la ligne d'interruption utilisée par les ports série. Cette fonctionnalité n'est pas très sûre.

L'option « Support special multiport boards » permet d'activer certaines cartes série, qui peuvent indiquer quel port série est en attente de traitement de la part du driver.

L'option « Support the Bell Technologies HUB6 card » permet d'activer la gestion des cartes HUB6.

L'option « Non-standard serial port support » permet d'activer les options correspondantes à des ports série non standards. Les options qui suivent permettent de configurer les drivers correspondants.

L'option « Unix98 PTY support » permet d'activer les pseudo terminaux compatibles Unix 98. Les pseudo terminaux sont des composants logiciels permettant d'émuler des terminaux. Il y a deux composants par pseudo terminal : la partie maître est le programme qui cherche à accéder au pseudo terminal, et la partie esclave est le programme qui simule le terminal physique. Les composants maîtres peuvent ouvrir le pseudo terminal par l'intermédiaire des fichiers spéciaux /dev/ptyxx. En revanche, il existe deux protocoles de nommage des fichiers spéciaux utilisés par les composants esclaves. L'ancienne méthode

utilisait les fichiers spéciaux `/dev/ttyxx`, mais elle a été remplacée par un autre protocole, dans lequel le composant esclave doit d'abord ouvrir le fichier spécial `/dev/ptmx`, qui lui donne le numéro `n` de pseudo terminal qu'il doit simuler, et il ouvre ensuite le fichier `/dev/pts/n`. Ce dernier fichier spécial est créé à la volée, grâce à un système de fichier virtuel. Pour que ce mécanisme fonctionne, il faut que le système de fichiers `/dev/pts/` ait été également choisi parmi les systèmes de fichiers, et qu'il soit monté (voir la configuration du fichier `/etc/fstab`). Si vous répondez 'Y', l'option suivante vous permet de choisir le nombre de pseudo terminaux que vous désirez que le système gère. La valeur par défaut est 256.

L'option « `Parallel printer support` » permet d'activer la gestion des imprimantes connectées sur port parallèle. Le fait d'activer cette option n'empêche pas d'utiliser le port parallèle pour d'autres périphériques. Il est recommandé d'activer cette option sous forme de module parce qu'ainsi, les fonctionnalités d'impressions ne seront chargées que lorsque c'est nécessaire.

L'option « `Support for console on line printer` » permet de rediriger les messages destinés à la console sur une imprimante connectée au port parallèle. Ces messages sont alors imprimés au fil de l'eau.

L'option « `Support for user-space parallel port device drivers` » active la gestion du fichier spécial de périphérique `/dev/parport`, au travers duquel les programmes peuvent accéder de manière uniforme au port parallèle. Cette fonctionnalité n'est pas nécessaire pour utiliser une imprimante ou des périphériques ATAPI connectés sur port parallèle.

L'option « `QIC-02 tape support` » permet d'activer la gestion des lecteurs de cassettes non SCSI du type QIC. L'option suivante permet d'indiquer si la configuration dynamique du lecteur doit être utilisée.

L'option « `Intel i8x0 Random Number Generator support` » active la prise en charge des générateurs de nombres aléatoires présents sur les cartes mères à base de chipsets i8x0 d'Intel.

L'option « `/dev/nvram support` » permet d'activer l'accès à la mémoire non volatile de l'horloge temps réel. Cette mémoire peut être accédée par l'intermédiaire du fichier spécial de périphérique `/dev/nvram`, de code majeur 10 et de code mineur 144.

L'option « `Enhanced Real Time Clock Support` » permet d'activer l'accès au compteurs de l'horloge temps réel par l'intermédiaire du fichier spécial de périphérique `/dev/rtc`, de code majeur 10 et de code mineur 135. Cette option doit être activée si vous disposez d'une machine multiprocesseur.

L'option « `Double Talk PC internal speech card support` » permet d'activer la gestion du synthétiseur de voix Double Talk.

L'option « `Siemens R3964 line discipline` » active la gestion des synchronisations entre les périphériques utilisant le protocole de communication R3964 de Siemens

L'option « `Applicom intelligent fieldbus card support` » active la gestion des cartes Applicom « intelligent fieldbus ».

L'option « /dev/agpgart support (AGP support) » active la gestion des transferts de données accélérés par le bus AGP pour les drivers de cartes graphiques. Ces transferts peuvent être pilotés au travers du fichier spécial de périphérique /dev/agpgart, de code majeur 10 et de code mineur 175. Cette fonctionnalité n'est pas nécessaire pour le bon fonctionnement des cartes graphiques AGP, mais elle permet d'accroître les performances des drivers 3D. Elle est également indispensable pour permettre l'accès au registre MTRR du processeur permettant de contrôler le bus AGP. La réponse recommandée est 'Y' si vous disposez d'une carte graphique AGP, et 'N' sinon. Les options qui suivent permettent de sélectionner le type de chipset utilisé par la carte mère, afin de déterminer la manière dont le port AGP fonctionne. Si vous avez activé cette fonctionnalité, vous devez choisir l'option correspondant à votre chipset également.

L'option « Direct Rendering Manager (XFree86 DRI support) » permet d'activer le support de l'architecture DRI permettant aux pilotes graphiques d'accéder directement aux ressources des cartes graphiques. Cette fonctionnalité est à la base du fonctionnement de l'architecture 3D de Linux. Les options suivantes permettent de sélectionner le type de carte graphique installé sur le système. La réponse recommandée est 'N', sauf si vous disposez d'une de ces cartes graphiques. Dans ce cas, vous devriez certainement utiliser XFree86 4.0.3, afin de bénéficier des accélérations 3D de ces cartes graphiques. Notez que certaines cartes graphiques 3D, bien que parfaitement supportées sous Linux, n'utilisent pas les fonctionnalités DRI du noyau. C'est en particulier le cas pour toutes les cartes graphiques à base de puce NVidia, pour lesquelles il faut utiliser le module fourni avec le driver de NVidia.

10.7.43Menu « I2C support »

L'option « I2C support » permet d'activer la gestion du protocole de communication I2C. Ce protocole de communication est utilisé par beaucoup de micro-contrôleurs, et peut être nécessaire pour accéder à certaines fonctionnalités. En particulier, les cartes d'acquisition TV basées sur les puces électroniques Bt848 nécessitent cette fonctionnalité pour être utilisable. La réponse recommandée est 'N', sauf si vous disposez d'une telle carte.

L'option « I2C bit-banging interfaces » active la gestion des adaptateurs « bit-banging ». Cette option est nécessaire pour faire fonctionner les cartes d'acquisition TV basées sur les puces électroniques Bt848. La réponse recommandée est 'N', sauf si vous disposez d'une telle carte. Les options suivantes activent la gestion des périphériques utilisant cette interface. Il n'est pas nécessaire de les activer pour faire fonctionner les cartes d'acquisition TV à base de Bt848.

L'option « I2C PCF 8584 interfaces » active la gestion des adaptateurs PCF. Les suivantes permettent d'activer les périphériques utilisant cette interface.

L'option « I2C device interface » active la gestion des fichiers spéciaux de périphériques /dev/i2c-*.

10.7.44Menu « Mice »

L'option « Bus Mouse Support » active la prise en charge des souris bus. Ces souris sont des souris connectées sur des bus spécifiques, comme par exemple sur une carte

graphique. Cette option permet d'inclure le driver générique de ce type de souris, les drivers spécifiques peuvent être sélectionnés avec les options suivantes.

L'option « `ATIXL busmouse support` » permet d'activer la gestion des souris bus connectées sur certaines cartes graphiques ATI.

L'option « `Logitech busmouse support` » permet de gérer les souris bus Logitech. Attention, les souris Logitech connectées sur port PS/2 ne sont pas des souris bus et ne sont pas gérées par ce driver.

L'option « `Microsoft busmouse support` » permet de gérer les souris bus Microsoft. Attention, les souris Microsoft connectées sur port PS/2 ne sont pas des souris bus et ne sont pas gérées par ce driver.

L'option « `Mouse Support (not serial and bus mice)` » permet d'activer la gestion des souris qui ne sont connectées ni au port série, ni sur un bus souris. Si vous disposez d'une souris série ou d'une souris bus, vous devez répondre '`N`' à cette question. Sinon, la réponse recommandée est '`Y`'.

L'option « `PS/2 mouse (aka "auxiliary device") support` » permet de gérer toutes les souris connectées sur un port PS/2, quelles que soit leur marque. La plupart des souris vendues actuellement son de ce type.

L'option « `C&T 82C710 mouse port support (as on TI Travelmate)` » permet de gérer les souris pour Travelmate. Ces souris sont des souris PS/2 particulières à ces ordinateurs. Il est recommandé d'essayer le driver générique PS/2 avant de choisir celui-ci, aussi faut-il répondre '`N`' à cette question en général.

L'option « `PC110 digitizer pad support` » permet d'activer la gestion de l'émulation souris pour les PC110 digitizer.

10.7.45Menu « Joystick support »

L'option « `Jostick support` » permet d'activer les options de gestion des joysticks. Les options suivantes correspondent aux drivers des différents types de joysticks supportés.

10.7.46Menu « Watchdog cards »

L'option « `Watchdog Timer Support` » permet d'activer la détection des blocages systèmes grâce à un chien de garde. Le chien de garde exige un accès en écriture sur le fichier spécial de périphérique `/dev/watchdog`, de code majeur 10 et de code mineur 130, au moins une fois par minute. Si cette condition n'est pas vérifiée, la machine est redémarrée automatiquement. La gestion de cette fonctionnalité peut être réalisée matériellement grâce à une carte spéciale, ou logiciellement.

L'option « `Disable watchdog shutdown on close` » permet de maintenir l'activité du chien de garde même si le processus qui le réveille régulièrement ferme le fichier `/dev/watchdog`. Ce n'est pas le cas si la réponse '`N`' est donnée, dans ce cas, le chien de garde ne fonctionnera plus en cas d'arrêt du processus de surveillance.

L'option « `Software Watchdog` » permet d'activer la gestion du chien de garde logiciel.

L'option « WDT Watchdog timer » permet d'activer la gestion des cartes WDT. Ces cartes ne pouvant pas être configurées automatiquement, vous devrez spécifier manuellement le port et la ligne d'interruption à l'aide du paramètre du noyau wdt lors du démarrage du système. Si vous répondez 'Y' ou 'M', vous pourrez spécifier les fonctionnalités de cette carte à l'aide des options « WDT501 features » et « Fan Tachometer ».

L'option « WDT PCI Watchdog timer » permet d'activer la gestion des cartes WDT PCI.

Les options « WDT501 features » et « Fan Tachometer » permettent de spécifier les paramètres du système surveillés par les cartes WDT : température et vitesse du ventilateur.

L'option « Berkshire Products PC Watchdog » permet d'activer la gestion des cartes Berkshire.

L'option « Acquire SBC Watchdog Timer » permet d'activer la gestion des chiens de garde des PSC-6x86 Single Board Computer d'Acquire Inc.

L'option « SBC-60XX Watchdog Timer » permet d'activer la gestion des chiens de garde des ordinateurs disposant d'une carte mère SBC 6010.

L'option « Mixcom Watchdog » permet d'activer la gestion des chiens de garde Mixcom.

L'option « Intel i810 TCO timer / Watchdog » permet d'activer la gestion des chiens de garde matériels intégrés dans les chipsets i810 et i815 d'Intel. Ces chiens de garde peuvent surveiller l'activité du système régulièrement et contrôler sa température.

10.7.47Menu « Ftape, the floppy tape device driver »

L'option « Ftape (QIC-80/Travan) support » permet d'activer les options de gestion des lecteurs de bande connectés sur le contrôleur de disquettes. Si vous avez un tel périphérique, répondez 'Y' ou 'M'. Sinon, répondez 'N'.

L'option « Zftape, the VFS interface » permet d'inclure la gestion des systèmes de fichiers virtuels dans le driver de ftape. Il faut impérativement activer cette option, faute de quoi le driver sera inutilisable. Répondez par 'Y' à cette question si vous avez activé la fonctionnalité ftape.

L'option « Default block size » permet de spécifier la taille par défaut des blocs utilisés par les programmes d'archivage. La valeur 10240 correspond à la taille des blocs utilisée par GNU tar, c'est la valeur recommandée.

L'option « Number of ftape buffers (EXPERIMENTAL) » est expérimentale et il est très déconseillé de la modifier. La valeur par défaut est 3.

L'option « Enable procfs status report (+2kb) » permet de générer un répertoire ftape/ dans le système de fichiers virtuel /proc/. Ce répertoire contient des informations concernant l'état courant du driver ftape.

L'option « Debugging output » permet de fixer la quantité de traces que le driver ftape génère. La valeur recommandée est « Normal ».

L'option « Floppy tape controllers » permet de sélectionner le type de contrôleur de disquettes. La valeur recommandée est « Standard ». Si vous choisissez un autre type de contrôleur, vous devrez spécifier les paramètres matériels pour ce contrôleur dans les trois options qui suivent.

L'option « Default FIFO threshold (EXPERIMENTAL) » est expérimentale et ne doit pas être modifiée. La valeur recommandée est 8.

L'option « Maximal data rate to use (EXPERIMENTAL) » permet de réduire la vitesse maximale de transfert utilisé par le driver ftape. Cette option est expérimentale et ne doit pas être modifiée. La valeur par défaut est 2000.

10.7.48Menu « PCMCIA character device support »

Les options de ce menu permettent d'activer la gestion des cartes PCMCIA de type série, comme par exemple les modems, les ports série et les cartes réseau intégrant un modem.

10.7.49Menu « Multimedia devices »

L'option « Video For Linux » permet d'activer les options de gestion de la vidéo et de radio sous Linux. Il faut activer cette option pour accéder aux deux menus suivants, qui permettent de choisir les drivers adaptés au type de matériel vidéo installé et les options de configuration pour certains de ces drivers.

10.7.50Menu « Video For Linux »

L'option « V4L information in proc filesystem » permet d'inclure des informations complémentaire sur l'état de l'interface Vidéo pour Linux dans le système de fichiers /proc/..

L'option « I2C on parallel port » permet d'utiliser le port parallèle comme un interface I2C pour les contrôleurs vidéo qui reconnaissent ce protocole

Les options qui suivent permettent de choisir les drivers pour les différents types de cartes d'acquisition vidéo. Vous devez activer la fonctionnalité correspondant au matériel dont vous disposez. En général, il est recommandé d'utiliser les drivers sous forme de modules, car certains de ces drivers ne sont pas capables de faire la distinction entre les différents modèles de cartes qui utilisent la même électronique, et il faut leur communiquer ces informations sous la forme de paramètres lors du chargement des modules. C'est en particulier le cas pour les cartes vidéo basées sur la puce électronique Bt848 et les puces qui en sont dérivées.

10.7.51Menu « Radio Adapters »

Les options proposées par ce menu vous permettront d'activer les drivers pour les différentes cartes radio supportées par Linux. Vous devez activer le driver correspondant à votre matériel, et éventuellement spécifier les paramètres matériels de cette carte.

10.7.52Menu « File systems »

L'option « `Quota support` » permet d'activer la gestion des quotas de disque utilisé par utilisateur. Cette fonctionnalité n'est disponible que pour les systèmes de fichiers EXT2.

L'option « `Kernel automounter support` » permet d'effectuer le montage des disques NFS automatiquement (c'est-à-dire à la demande) au niveau du noyau.

L'option « `Kernel automounter version 4 support (also supports v3)` » permet d'activer la nouvelle version du montage automatique des disques NFS.

Les options qui suivent permettent de prendre en compte les systèmes de fichiers de différents systèmes d'exploitation. Parmi ces systèmes, certains sont encore en cours de développement et ne fonctionnent pas très bien. Cela signifie qu'il est très déconseillé d'utiliser ces drivers pour écrire des données sur ces systèmes de fichiers : de très grosses pertes de données peuvent s'ensuivre. En particulier, il ne faut surtout pas utiliser le système de fichier NTFS en écriture : les développeurs de cette partie du noyau sont, encore actuellement, certains qu'il détruira le système de fichiers. En revanche, les systèmes de fichiers qui sont complètement gérés peuvent être utilisés sans risques. Cependant, ils peuvent ne pas être assez puissants pour stocker les informations de fichiers nécessaires à tout système Unix. Dans ce cas, le système de fichiers ne peut être utilisé que pour stocker des données d'applications, pas le système lui-même. C'est en particulier le cas de la FAT. Afin de corriger les limitations de ce système de fichiers, une extension nommée UMSDOS a été développée. Cette extension stocke dans chaque répertoire un fichier caché décrivant les droits Unix, et rend ainsi la FAT exploitable sous Linux. Cette fonctionnalité est très intéressante si l'on veut installer Linux sur un disque déjà formaté en FAT. Il faut prendre garde cependant au fait que les performances seront alors déplorables (les accès disques sont plus lents dans un facteur 2 à 3). Le problème de performances se pose d'ailleurs pour tous les systèmes de fichiers qui ont été intégré à Linux mais qui ne sont pas natifs. Ainsi, même sans utiliser UMSDOS, la FAT32 est deux fois plus lente que les systèmes de fichiers natifs. Le système de fichiers le plus utilisé actuellement sous Linux est EXT2, qui offre à la fois la sécurité, les fonctionnalités et les performances. Il est fortement recommandé de l'utiliser dès que l'on peut se passer des autres systèmes de fichiers.

Il est impératif de compiler le système de fichiers de la partition root dans le noyau. Si cela n'est pas fait, le noyau ne pourra pas monter la partition root et se terminera en affichant le message « kernel panic ». Il ne faut pas compiler ce système de fichiers en tant que module, pour les mêmes raisons. La réponse recommandée pour l'option « `Second extended fs support` » est donc 'Y' et rien d'autre, et on installera la partition root sur un tel système de fichier.

Certains systèmes de fichiers nécessitent le support de systèmes de fichiers de base. En particulier, les systèmes de fichiers VFAT (qui gère les FAT32), MSDOS fs (qui gère les partitions DOS 12 et 16 bits) et UMSDOS exigent tous trois le support du système de fichiers DOS FAT en général. Pour utiliser ces systèmes de fichiers, on devra donc activer l'option « `DOS FAT fs support` ». De même, pour lire les CD-ROM au format Juliette (l'extension Microsoft au format ISO 9660 pour supporter les noms longs sous Windows), il faudra activer l'option « `ISO 9660 CDROM file system support` ».De toutes façons, il est fortement recommandé de gérer ce système de fichiers si l'on veut utiliser des CD-ROMs.

Quatre systèmes de fichiers sont virtuels. Ils ne correspondent à aucun support physique, et leur arborescence est créée uniquement en mémoire, à la volée, par le noyau. Il s'agit du système de fichiers `/proc/` (option « `/proc file system support` »), qui fournit des informations dynamiquement sur l'état du système, du système de fichiers `/dev/` (option « `/dev file system support (EXPERIMENTAL)` »), qui permet de gérer les fichiers spéciaux de périphériques à la volée, du système de fichiers `/dev/pts/` (option « `/dev/pts file system for Unix98 PTYs` »), qui permet de créer des fichiers spéciaux de périphériques à la demande pour les pseudo terminaux et du système de fichiers `/dev/shm/` (option « `Virtual memory file system support (former shm fs)` »), qui permet de gérer les segments de mémoire partagée POSIX. Il est fortement recommandé d'activer la gestion des systèmes de fichiers `/proc/`, `/dev/pts/` et `/dev/shm/`, car ils sont utilisés par beaucoup de programmes. Le système de fichiers `/dev/` pourra être utilisé sur les systèmes embarqués ou les disquettes de démarrage, pour lesquels la taille prise par le répertoire `/dev/` peut être gênante. Dans ce cas, on aura intérêt à réaliser le montage automatique de ce système de fichiers à l'aide de l'option « `Automatically mount at boot` ».

10.7.53Menu « Network File Systems »

L'option « `Coda file system support (advanced network fs)` » permet d'activer le support du système de fichiers réseau Coda, qui donne accès aux périphériques par le réseau comme s'ils étaient branchés sur la machine locale. Cette option active les fonctionnalités clientes au niveau du noyau, mais il faut également des outils complémentaires pour implémenter les fonctions serveurs.

L'option « `NFS file system support` » permet d'activer le support NFS classique en tant que machine cliente. NFS est moins puissant que Coda, mais est cependant le plus répandu. Il lui est donc encore préférable pour l'instant.

L'option « `Provide NFSv3 client support` » permet d'activer le support de la version 3 du protocole NFS en tant que machine cliente.

L'option « `Root file system on NFS` » permet d'indiquer au noyau que le système de fichiers root est placé sur un serveur NFS. Dans ce cas, le noyau se connectera automatiquement à ce serveur pour monter le système de fichiers root. Pour réaliser ce type de montage, il faut avoir répondu '`Y`' à la question « `IP: kernel level autoconfiguration` », afin que les couches réseaux soient configurées au niveau du noyau lors du démarrage du système.

L'option « `NFS server support` » permet d'activer le support NFS en tant que machine serveur.

L'option « `Provide NFSv3 server support` » permet d'activer le support de la version 3 du protocole NFS en tant que serveur.

L'option « `SMB file system support (to mount WfW shares, etc.)` » permet d'accéder aux répertoires partagés par les machines fonctionnant sous Windows. Cette fonctionnalité n'est possible que si les postes Windows utilisent TCP/IP comme protocole de réseau de base, et non NetBIOS.

Les « Use a default NLS » et « Default Remote NLS Option » permettent de spécifier une page de code par défaut pour lire les noms de fichiers et de répertoires partagés par le serveur.

L'option « NCP file system support (to mount NetWare volumes) » permet d'accéder aux volumes NetWare. Cette fonctionnalité nécessite la gestion d'IPX au niveau de la machine Linux. On notera qu'il est inutile d'activer cette fonctionnalité si l'on désire faire en sorte que la machine Linux soit serveur de fichier Novell. Les options qui suivent permettent de paramétrer les accès aux volumes NetWare (sécurité, lock de fichiers, droits d'accès...).

10.7.54Menu « Partition Types »

Quelques systèmes utilisent un format différent pour les tables de partitions que le format utilisé par le BIOS des PC. Pour lire les données stockées sur les disques partitionnés par ces systèmes sur des machines d'architecture différente, vous devez activer une gestion spéciale des tables de partitions. Les options de ce menu permettent d'activer la gestion des tables de partitions de ces systèmes.

10.7.55Menu « Native Language Support »

Les options fournies ici permettent de choisir les pages de codes à utiliser pour les jeux de caractères dans les systèmes de fichiers. Ces pages de codes doivent être choisies en fonction de la langue d'installation du système. Les pages de codes recommandées pour un système français sont les suivantes :

- « Codepage 437 (United States, Canada) », indispensable en raison de la majorité des programmes provenant des États Unis ;
- « Codepage 850 (Europe) », pour les caractères accentués européens ;
- « NLS ISO 8859-1 (Latin 1; Western European Languages) » pour les caractères accentués européens sur les CDROMs ;
- « NLS ISO 8859-15 (Latin 9; Western European Languages with Euro) » pour les caractères accentués européens sur les CDROMs, y compris quelques caractères additionnels non gérés par la page de code ISO 8859-1.

L'option « Default NLS Option » permet de choisir la page de code par défaut à utiliser parmi celles qui ont été choisies. La réponse recommandée est « iso8859-1 ».

10.7.56Menu « Console drivers »

L'option « VGA text console » permet d'utiliser les modes texte VGA pour l'affichage en mode texte.

L'option « Video mode selection support » permet de choisir le mode texte à utiliser au démarrage de Linux. Ce mode peut être indiqué grâce à un paramètre passé au noyau lors du démarrage.

L'option « MDA text console (dual-headed) (EXPERIMENTAL) » permet d'activer le support de l'affichage multiple réalisable à l'aide d'une carte MDA et d'une carte VGA. Cette option ne doit être choisie que si la carte MDA est la carte d'affichage principale.

10.7.57Menu « Frame-buffer support »

L'option « `Support for frame buffer devices (EXPERIMENTAL)` » permet d'activer le support des cartes graphiques par l'intermédiaire d'une interface unifiée et d'un buffer vidéo nommé « frame buffer », accessible via les fichiers spéciaux `/dev/fb*`. Cette interface permet aux programmes d'accéder aux fonctionnalités des cartes graphiques de manière portable. Le support de cette fonctionnalité n'est en général pas nécessaire pour les PC, puisque beaucoup de serveurs X ont été développés pour cette plate-forme, et que dans le pire des cas, un serveur X basé sur le standard VESA 2.0 est disponible. Cependant, pour les autres plates-formes, ou si vous disposez d'une carte graphique exotique, il peut être utile d'activer cette fonctionnalité et d'utiliser un serveur X basé sur cette interface. Les options qui suivent permettent d'inclure les drivers pour différentes cartes graphiques. Vous trouverez de plus amples informations sur ces options dans la huitième partie de ce document..

L'option « `Virtual Frame Buffer support (ONLY FOR TESTING!)` » permet d'activer la gestion d'une carte graphique virtuelle en mémoire seulement. Ce driver consomme énormément de ressources et ne doit être utilisé que pour tester des applications basées sur cette interface..

L'option « `Advanced low level driver options` » permet d'activer les options permettant de fixer certains paramètres bas niveau. Ces paramètres peuvent être fixés grâce aux options suivantes. En particulier, les formats de pixels gérés, ainsi que les polices de caractères à utiliser peuvent être précisés.

10.7.58Menu « Sound »

L'option « `Sound card support` » permet d'activer la gestion des cartes son. Si vous ne disposez pas de carte son, choisissez la réponse '`N`', sinon, activez cette fonctionnalité.

Les options qui suivent permettent de choisir les drivers pour les cartes son non standards. Ces drivers gèrent les cartes son correctement, mais ne fournissent pas exactement la même interface de programmation que le standard actuel sous Linux qu'est « OSS » (abréviation de l'anglais « Open Sound System »). Certaines fonctionnalités ne seront donc pas accessibles de la manière standard avec ces cartes.

Vient ensuite l'option « `OSS sound modules` », qui permet d'activer la sélection des drivers compatibles avec l'interface OSS. Les options qui suivent celle-ci permettent de choisir et de configurer les drivers pour les cartes son compatibles. Il est recommandé d'utiliser les drivers OSS si possible. Notez qu'une certain nombre de drivers ont également été développés dans le cadre de l'architecture ALSA (abréviation de l'anglais « Advanced Linux Sound Architecture »). Si vous ne trouvez pas de driver approprié à votre matériel ici, vous pouvez essayer d'installer ces drivers.

Notez enfin que le driver de son pour les cartes d'acquisition TV basées sur la puce Bt848 se situe parmi les drivers OSS (option « `TV card (bt848) mixer support` »). Vous devez donc l'intégrer sous forme de module si vous disposez d'une telle carte.

10.7.59Menu « USB support »

L'option « `Support for USB` » permet de prendre en charge les périphériques USB sous Linux.

L'option « USB verbose debug message » permet de demander aux drivers USB d'envoyer des informations de débogage dans les fichiers de traces du système.

L'option « Preliminary USB device filesystem » active la gestion des informations sur les ports USB dans le système de fichiers virtuels /proc/. Ces informations comprennent en particulier la liste des périphériques connectés sur le bus USB.

L'option « Enforce USB bandwidth allocation (EXPERIMENTAL) » active la gestion de la bande passante du bus USB entre les différents périphériques qui y sont connectés. Cette option est expérimentale.

Les trois options suivantes (« UHCI (Intel PIIX4, VIA, ...) support », « UHCI Alternate Driver (JE) support » et « OHCI (Compaq, iMacs, OPTi, SiS, ALi, ...) support » permettent d'inclure les drivers pour les différents types de contrôleurs USB. On recense deux différents types de contrôleurs USB : les contrôleurs UHCI (abréviation de l'anglais « Universal Host Controller Interface ») et les contrôleurs OHCI (abréviation de « Open Host Controller Interface »). Les contrôleurs UHCI sont fabriqués par Intel et VIA essentiellement, alors que les contrôleurs OHCI sont présents sur les chipsets non Intel (Compaq, SiS, Aladdin). Vous devez choisir le driver approprié au chipset de votre carte mère. Pour les contrôleurs UHCI, vous avez le choix entre deux drivers.

L'option « USB Audio support » permet de prendre en charge les périphériques audio connectés sur le port USB, comme des hauts-parleurs par exemple.

L'option « USB Bluetooth support (EXPERIMENTAL) » active la gestion des périphériques USB Bluetooth (capables d'effectuer des transmissions sans fil).

L'option « USB Mass Storage support » active la gestion des périphériques USB de masse (lecteurs de CD, etc.).

L'option « USB Mass Storage verbose debug » permet de demander au driver des périphériques USB de masse de générer des messages de débogage dans les fichiers de trace du système.

L'option « Freecom USB/ATAPI Bridge support » permet de gérer les passerelles entre le bus USB et les périphériques ATAPI connectés sur ce bus.

L'option « USB Modem (CDC ACM) support » permet de prendre en charge les modems analogiques et Numéris utilisant l'interface CDC ACM (abréviation de l'anglais « Communication Device Class Abstract Control Model ») connectés sur le port USB.

L'option « USB Printer support » active la gestion des imprimantes connectées sur le port USB.

L'option « USB Human Interface Devices (full HID) support » permet d'activer la prise en charge des périphériques d'entrée et de saisie tels que les claviers, souris et tablettes de digitalisation. Cette option n'est activable que si l'on a activé également la fonctionnalité « Input core support » du menu « Input core support ». Ce driver gère complètement les périphériques d'entrée USB, et est incompatible avec les drivers simplifiés pour le

clavier et la souris, que l'on peut activer avec les options « USB HIDBP Keyboard (basic) support » et « USB HIDBP Mouse (basic) support ».

L'option « USB HIDBP Keyboard (basic) support » permet d'activer un driver simplifié pour les claviers USB. Ce driver peut être utilisé pour alléger le noyau dans les systèmes embarqués, mais ne gère pas toutes les touches des claviers étendus. Ce driver est incompatible avec le driver « USB Human Interface Device (full HID) support ».

L'option « USB HIDBP Mouse (basic) support » permet d'activer un driver simplifié pour les souris USB. Ce driver peut être utilisé pour alléger le noyau dans les systèmes embarqués. Il est incompatible avec le driver « USB Human Interface Device (full HID) support ».

L'option « Wacom Intuos/Graphire tablet support » active la prise en charge des tablettes graphiques Wacom USB. Cette option nécessite d'avoir également activé l'option « Mouse support » ou l'option « Event interface support » du menu « Input core support ».

L'option « USB Kodak DC-2xx Camera support » active la prise en charge des caméras USB Kodak DC-2xx et de quelques appareils compatibles.

L'option « USB Mustek MDC800 Digital Camera support (EXPERIMENTAL) » permet de prendre en charge les caméras USB Mustek MDC800 et de quelques appareils compatibles.

L'option « USB Scanner support » active la prise en charge des scanners USB.

L'option « Microtek X6USB scanner support (EXPERIMENTAL) » permet de prendre en charge les scanners Microtek X6USB. Ces scanners apparaîtront dans le système comme des périphériques SCSI génériques. Notez qu'il faut une version modifiée de SANE pour utiliser ce driver.

L'option « USB IBM (Xirlink) C-it Camera support » permet de connecter une caméra USB Xirlink d'IBM à votre ordinateur. Cette caméra sera utilisable via l'interface Video4Linux, que l'on devra donc également activer.

L'option « USB OV511 Camera support » permet de connecter une caméra USB OV511 à votre ordinateur. Cette caméra sera utilisable via l'interface Video4Linux, que l'on devra donc également activer.

L'option « D-Link USB FM radio support (EXPERIMENTAL) » permet de connecter une radio FM USB D-Link. Cette radio sera utilisable via l'interface Video4Linux, que l'on devra donc également activer.

L'option « DABUSB driver » permet la prise en charge d'un récepteur USB DAB (abréviation de l'anglais « Digital Audio Broadcasting »).

L'option « PLUSB Prolific USB-Network driver (EXPERIMENTAL) » permet de prendre en charge les ponts USB-USB PL-2032 de la société Prolific. Ces ponts permettent de relier en réseau deux ordinateurs par l'intérmédiaire de leurs ports USB.

L'option « USB ADMtek Pegasus-based ethernet device support (EXPERIMENTAL) » permet de prendre en charge les cartes Ethernet USB.

L'option « NetChip 1080-based USB Host-to-Host Link (EXPERIMENTAL) » permet de prendre en charge les ponts USB-USB NetChip 1080.

L'option « USS720 parport driver » permet d'activer les convertisseurs USB / Port parallèle permettant de connecter les périphériques utilisant un port parallèle sur le port USB de l'ordinateur.

L'option « USB Diamond Rio500 support (EXPERIMENTAL) » permet de prendre en charge les lecteurs MP3 USB Rio500.

10.7.60Menu « USB Serial Converter support »

L'option « USB Serial Converter support » permet de prendre en charge les périphériques USB utilisant une interface série classique. Les options qui suivent activent les drivers spécifiques aux différents matériels supportés par Linux.

10.7.61Menu « Kernel hacking »

L'option « Magic SysRq key » permet d'activer la gestion des séquences de touches utilisant la touche SysRq (ou Syst) en cas de plantage du noyau. Cette option permet de déboguer le noyau lorsqu'on développe une nouvelle fonctionnalité.

10.8 Licence GPL

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

10.8.1 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in

effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

10.8.2 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".
Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.
You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)
These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the

whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.
Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.
In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)
The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.
If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.
If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.
It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.
This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.
Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.
10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

10.9 Table des matières détaillée

TABLE DES MATIÈRES **3**

TABLE DES ILLUSTRATIONS ET DES TABLEAUX **5**

PRÉFACE **6**

A QUI S'ADRESSE CE RAPPORT ? 6
VUE D'ENSEMBLE 6
CONVENTIONS UTILISEES 7
REMERCIEMENTS 7

1 INTRODUCTION **8**

1.1 APPLICATIONS CONCRETES 8

2 APPROCHE ET CAHIER DES CHARGES **10**

2.1 CAHIER DES CHARGES 10
2.2 APPROCHE 10

3 L'OS LINUX **11**

3.1 INTRODUCTION 11
3.1.1 Les capacités de l'Os *11*
3.2 STRUCTURE DE L'OS 12
3.2.1 Architecture du noyau *13*
3.2.2 Système de fichiers *14*
3.3 LES PERMISSIONS 16
3.3.1 Généralités *16*
3.3.2 Propriétaires et groupes *16*
3.3.3 Représentation des droits d'accès *16*
3.4 LE DEMARRAGE 17
3.4.1 Procédure de boot *17*
3.4.2 Démarrage de l'OS *18*
3.5 LINUX EN TANT QU'OS TEMPS REEL 18

4 LINUX, OS EMBARQUE **20**

4.1 CHOISIR LINUX 20
4.1.1 Avantages *20*
4.1.2 Désavantages *21*
4.2 LE HARDWARE 22
4.2.1 Systèmes finis *22*
4.2.2 Processeurs *22*
4.2.3 Mémoire RAM et mémoire de masse *23*
4.2.4 Périphériques d'entrées/sortie *25*
4.2.5 Affichage *25*
4.2.6 Conclusion *26*
4.3 PORTABILITE DU CODE SUR DIFFERENTES ARCHITECTURES 26
4.3.1 La portabilité aujourd'hui *26*
4.3.2 Linux sur différentes architectures *27*
4.3.3 Conclusion *28*

5 LINUX OPTIMISE POUR X86 29

5.1 GENERALITES 29
5.2 L'OPTIMISATION ? 29
5.3 SYSTEME DE FICHIERS 30
5.3.1 Généralités 30
5.3.2 Ext2 31
5.3.3 ReiserFs 31
5.3.4 Initrd 31
5.3.5 JFFS 32
5.3.6 Conclusion 33
5.4 SUPPRESSION DU SWAPPING 33
5.4.1 Généralités 33
5.4.2 Suppression du swapping 34
5.4.3 Comment supprimer le swapping ? 34
5.4.4 Conclusion 34
5.5 OPTIMISATION DE LA PROCEDURE DE DEMARRAGE 35
5.5.1 Généralités 35
5.5.2 Chargeurs Linux 35
5.5.3 Démarrage du noyau 36
5.5.4 Fichiers d'initialisation 36
5.5.5 Conclusion 37
5.6 OPTIMISATION DES LIBRAIRIES C 37
5.6.1 Généralités 37
5.6.2 Statique vs Dynamique 38
5.6.3 Eliminer les librairies inutiles 38
5.6.4 Librairies alternatives 38
5.6.5 Conclusion 40
5.7 SHELL ET COMMANDES MINIMALES 40
5.7.1 Généralités 40
5.7.2 Le Shell 41
5.7.3 Optimisation des commandes standards 41
5.7.4 Alternatives aux commandes standards 41
5.7.5 Conclusion 43
5.8 OPTIMISATION DU NOYAU 43
5.8.1 Généralités 43
5.8.2 Choix de la version du noyau 43
5.8.3 Compilation du noyau 44
5.8.4 zImage ou bzImage 45
5.8.5 Modification du code source 46
5.8.6 Taille du noyau décompressé et de ses structures de données 46
5.8.7 Drivers chargeables ou modules 46
5.8.8 Conclusion 47
5.9 SIMPLIFICATION DU NOMBRE DE FICHIERS 47
5.9.1 Généralité 47
5.9.2 Facilité de l'administration 47
5.9.3 Economie en mémoire de masse 48
5.9.4 Conclusion 49
5.10 COMPILATION 49
5.10.1 Généralité 49
5.10.2 Options de compilations 49

5.10.3 Conclusion 52
5.11 OPTIMISATION DES APPLICATIONS 52
5.11.1 Généralités 52
5.11.2 Compilation 52
5.11.3 Compression des exécutables 53
5.11.4 XIP :Execute in place 53
5.11.5 Ecriture des applications 54
5.11.6 Conclusion 55
5.12 AFFICHAGE 55
5.12.1 Généralités 55
5.12.2 Console VGA 55
5.12.3 Console série 55
5.12.4 Ecran LCD en mode texte 55
5.12.5 Graphique 56
5.12.6 Conclusion 58
5.13 APPLICATIONS MINIMALES POUR LINUX EMBARQUE 59
5.13.1 Généralités 59
5.13.2 E3 59
5.13.3 Boa Webserver 59
5.13.4 Thttpd 59
5.13.5 Ash 59
5.13.6 Pcmcia-cs 59
5.13.7 MAD : Mpeg Audio Decoder 60
5.13.8 LSH 60
5.13.9 Pppd 60
5.13.10 OpenSSL 60
5.13.11 BlueZ Bluetooth stack 60
5.13.12 uDHCP 60
5.13.13 Conclusion 61
5.14 CONCLUSION 61

6 PROCESSUS DE CONSTRUCTION D'UN SYSTEME LINUX EMBARQUE 62

6.1 INTRODUCTION 62
6.2 METHODE ET PLATE-FORME DE DEVELOPPEMENT 62
6.2.1 Plate-forme de développement 62
6.2.2 Méthode de développement 63
6.3 CONSTRUCTION DU SYSTEME 63
6.3.1 Installation du système de fichiers 64
6.3.2 Image Initrd ou fichier binaire contenant l'OS 67
6.3.3 Création des répertoires nécessaires 68
6.3.4 Modification du code source du noyau 71
6.3.5 Compilation du noyau 73
6.3.6 Installation des librairies 76
6.3.7 Installation des commandes minimales 78
6.3.8 Configuration des scripts d'initialisation minimaux 81
6.3.9 Paramètres de démarrage 85
6.3.10 Activation du mode console par terminal série 88
6.3.11 Configuration Ethernet 90
6.3.12 Installation du serveur http BOA 96
6.3.13 Compression des exécutables à l'aide de BZEXE 97

6.3.14 Affichage à l'aide d'un LCD à caractères 98
6.3.15 Affichage en mode texte VGA 100
6.3.16 Affichage graphique 100
6.4 CONCLUSION 104

7 TESTS DU SYSTEME OPTIMISE 105

7.1 MEMOIRE RAM MINIMALE 105
7.2 PLACE EN MEMOIRE DE MASSE 105
7.3 LE SYSTEME D'EXPLOITATION EN GENERALE 105
7.4 LA PROCEDURE DE DEMARRAGE 105
7.5 ARCHITECTURE X86 105
7.6 MODULE CHARGEABLES 106
7.7 LA CONNEXION PAR TERMINAL SERIE 106
7.8 L'AFFICHAGE 107
7.9 LE RESEAU ETHERNET 107

8 SYNTHESE 108

8.1 PROBLEMES RENCONTRES 108
8.1.1 Compilation à l'aide de la ucLibc 108
8.1.2 Compilation du noyau 108
8.1.3 Problème du noyau 2.4.0 108
8.1.4 Utilisation des modules chargeables 109
8.1.5 Géométrie du disque dur et Lilo 109
8.1.6 Notion de temps en programmation 110
8.1.7 Compilation de PicoGui 110
8.1.8 Incompatibilité avec la carte Pc-104 de la Hevs 110
8.2 EVOLUTION DU PROJET 110
8.2.1 Stockage en Flash 110
8.2.2 Les autres architectures 111
8.2.3 Linux temps réel 111
8.2.4 Solutions de mises à jour 111

9 CONCLUSION 112

10 ANNEXES 113

10.1 BIBLIOGRAPHIE ET LIENS UTILES 114
10.1.1 Généralités 114
10.2 COMPLÉMENTS SUR LINUX 117
10.2.1 L'arborescence FHS des répertoires 118
10.2.2 Systèmes de fichiers supportés par Linux 123
10.2.3 Fichiers périphériques 125
10.3 AFFICHEUR LCD 181
10.3.1 Le port parallèle 182
10.3.2 LCD HD44780 184
10.3.3 Programme pour LCD HD44780 185
10.4 CODES SOURCES ET SCRIPTS UTILISÉS 194
10.4.1 Script de création des fichiers périphériques 195
10.4.2 Fichier protocols 199
10.4.3 Fichier de configuration de BOA 200
10.4.4 Script BZEXE 206
10.4.5 Réduction des Librairies C 211

10.5 DESCRIPTION DES COMMANDES BUSYBOX 231
10.6 DESCRIPTION DES COMMANDES TINYLOGIN 270
10.7 OPTIONS DE COMPILATION DU NOYAU 272
10.7.1 Menu « Code maturity level options » 272
10.7.2 Menu « Loadable module support » 272
10.7.3 Menu « Processor type and features » 272
10.7.4 Menu « General setup » 273
10.7.5 Menu « Memory Technology Devices (MTD) » 276
10.7.6 Menu « Parallel port support » 276
10.7.7 Menu « Plug and Play configuration » 276
10.7.8 Menu « Block devices » 277
10.7.9 Menu « Multi-device support (RAID and LVM) 278
10.7.10 Menu « Networking options » 279
10.7.11 Menu « IP: Netfilter Configuration » 284
10.7.12 Menu « IPv6: Netfilter Configuration » 287
10.7.13 Menu « QoS and/or fair queueing » 287
10.7.14 Menu « Telephony Support » 288
10.7.15 Menu « ATA/IDE/MFM/RLL support » 288
10.7.16 Menu « IDE, ATA and ATAPI Block devices » 288
10.7.17 Menu « SCSI support » 290
10.7.18 Menu « SCSI low-level drivers » 291
10.7.19 Menu « PCMCIA SCSI adapter support » 291
10.7.20 Menu « IEEE 1394 (FireWire) support » 292
10.7.21 Menu « I2O support » 292
10.7.22 Menu « Network device support » 293
10.7.23 Menu « ARCnet devices » 295
10.7.24 Menu « AppleTalk devices » 295
10.7.25 Menu « Ethernet (10 or 100Mbit) » 296
10.7.26 Menu « Ethernet (1000 Mbit) » 296
10.7.27 Menu « Wireless LAN (non-hamradio) » 296
10.7.28 Menu « Token ring devices » 296
10.7.29 Menu « Wan interfaces » 297
10.7.30 Menu « PCMCIA network device support » 297
10.7.31 Menu « ATM drivers » 298
10.7.32 Menu « Amateur Radio support » 298
10.7.33 Menu « AX.25 network device drivers » 298
10.7.34 Menu « IrDA subsystem support » 298
10.7.35 Menu « Infrared-port device drivers » 299
10.7.36 Menu « ISDN subsystem » 299
10.7.37 Menu « ISDN feature submodules » 299
10.7.38 Menu « Passive ISDN cards » 299
10.7.39 Menu « Active ISDN cards » 299
10.7.40 Menu « Old CD-ROM drivers (not SCSI, not IDE) » 300
10.7.41 Menu « Input Core Support » 300
10.7.42 Menu « Character devices » 302
10.7.43 Menu « I2C support » 304
10.7.44 Menu « Mice » 304
10.7.45 Menu « Joystick support » 305
10.7.46 Menu « Watchdog cards » 305
10.7.47 Menu « Ftape, the floppy tape device driver » 306

10.7.48 Menu « PCMCIA character device support » *307*
10.7.49 Menu « Multimedia devices » *307*
10.7.50 Menu « Video For Linux » *307*
10.7.51 Menu « Radio Adapters » *307*
10.7.52 Menu « File systems » *308*
10.7.53 Menu « Network File Systems » *309*
10.7.54 Menu « Partition Types » *310*
10.7.55 Menu « Native Language Support » *310*
10.7.56 Menu « Console drivers » *310*
10.7.57 Menu « Frame-buffer support » *311*
10.7.58 Menu « Sound » *311*
10.7.59 Menu « USB support » *311*
10.7.60 Menu « USB Serial Converter support » *314*
10.7.61 Menu « Kernel hacking » *314*
10.8 LICENCE GPL 315
10.8.1 Preamble *315*
10.8.2 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION *316*
10.9 TABLE DES MATIÈRES DÉTAILLÉE 320
10.10 GLOSSAIRE 326
10.11 CD-ROM ET DISQUETTE 334

10.10 Glossaire

[ko]
Kilo octet

[Mo]
Méga octet

Amiga
Type d'ordinateur produit par Commodore

API
Application Programming Interface: appel système offert par le noyau

Architecture
Spécificité de construction de l'ordinateur: gestion mémoire, bus, etc.

ARM
Concepteur de corps de microprocesseurs

Atari
Firme productrice d'ordinateurs et de jeux video

biblothèque
Autre nom donné aux librairies

Bill Gates
Fondateur de Microsoft

BIOS
Basic Input Output System : première application lancée automatiquement sur Pc, elle sert à initialiser la machine

BSD UNIX
Type de Unix

bug
Expression du monde informatique signifiant: problème

C
Langage de programmation de haut niveau. Linux est écrit en langage C.

Cd-Rom
Compact Disc, read-only-memory

CGI
Langage évolué intégré aux serveurs http

chargeur
Programme servant à lancer Linux

compatible x86
Ordinateur équipé d'un processeur compatible 80x86

compilateur
Programme permettant à partir d'un fichier contenant du code de réaliser un exécutable

Debian
Entreprise commercialisant une distribution de Linux

debuging
Action de trouver les erreurs (bug) dans une application

DiskOnChip
Périphérique de stockage en mémoire Flash construit par M-System

DMA
Direct Memory Access : procédure d'accès direct à la mémoire sans passer par le CPU

drivers
Application chargée de dialoguer avec un périphérique

DVD
Digital Vesatile Disk : média de stockage principalement utilisé pour les films

Ecos
Système d'exploitation de Red Hat pour les machines embarquées

ELISE
Embedded LInux SEtup . Distribution créée par Gaspoz Frédéric

EOF
End Of File : marqueur de fin d'une chaîne de caractères

Ext2
Extended 2 : type de système de fichier Linux

FAT16/FAT32
Système de fichiers de Windows 95/98

FPU
Floating Point Unit ou coprocesseur arithmétique

FrameBuffer
Mode vidéo sous Linux permettant de méler texte et graphique

FTP
File Transfer Protocol

gcc
Programme libre de compilation d'application C

GNU
Ce dit d'un logiciel libre

GPL
General Public Licence : licence de diffusion d'un logiciel libre

Hevs
Haute Ecole Valaisanne

HTTP
HyperText Transfert Protocol

IDE
Intelligent Device Electronics: standard pour la réalisation de périphériques de stockage tels que les disques durs

IEEE
Institute of Electrical and Electronics Engineers

Initrd
Système de fichiers Linux se chargeant en mémoire RAM

inode
Structure de donnée d'un système de fichier, servant à localiser un fichier sur un support de stockage

Intel
Entreprise créant des microprocesseurs et des composants électroniques dont la famille des x86.

IP
Internet Protocol

ISO 9660
Format de système de fichier pour Cd-Rom

JFFS
Jounalising Flash File System

kernel
Autre nom donné au noyau de Linux

LCD
Liquid Crystal Display

librairies
Ensemble de routine d'un langage de programmation, mis à disposition des programmeurs.

Linus Thorvald
Créateur de Linux

Linux
Système d'exploitation dérivé du standard Unix. Linux Is Not UniX

Mandrake
Entreprise commercialisant une distribution de Linux

MBR
Master Boot Record: le premier secteur d'un disque dur, accédé en premier par le BIOS après le démarrage

mémoire de masse
Mémoire de stockage: disque dur, cd-rom,…

micro-noyau
Le noyau ne possède que quelques fonctions de base , le reste étant une couche supérieure.

Minix
Premier système d'exploitation, pour Pc, dérivé d'Unix

MMU
Management Memory Unit: hardware de gestion de la mémoire

module
Pilote de périphériques chargeable

monolithiques
Tous les drivers sont inclus dans le noyau

MP3
Format de compression audio populaire

MPEG
Format de compression audio-video.

MS-DOS
Disk Operating System. Système d'exploitation qui équipait en standard les Pc jusque vers le milieu des années 90.

multi-tâche
Ce dit d'un système d'exploitation capable de gérer plusieurs processus en même temps.

multi-utilisateur
Ce dit d'un système d'exploitation qui permet la connexion de plusieurs utilisateurs

NeXt
Firme fabricant des stations de travail professionnel (n'existe plus)

NFS
Network File Protocol

noyau
Cœur du système d'exploitation

NTFS
Système de fichiers de Windows NT

Open Source
La libre diffusion d'une application et de son code est nommée Open Source

ordonnanceur
Programme du noyau chargé de gérer les différents processus

Os
Operating System : système d'exploitation

patch
Programme servant à appliquer une modification à un autre programme

Pc industriel
Ordinateur au standard Pc compatible, disposant d'une taille réduite.

Pc-104
Standard définissant un Pc de petite taille

PCI
Peripheral Communications Interconnect :bus inclus dans les ordinateurs compatibles x86

pilote de périphériques
Application chargée de dialoguer avec un périphérique

portabilité
Adaptation d'une application à une architecture spécifique

POSIX
Portable Operating System Interface. Standard Unix pour faciliter la portabilité

PowerPC
Processeur Motorola et IBM

PPP
Point-to-Point Protocol

préemptif

Possibilité d'un système d'exploiation de stopper n'importe quelle application pour passer la main à une autre

primitive
Application simple

QNX
Entreprise réalisant des systèmes d'exploitation industriels

RAM
Random Access Memory. Aussi appelée mémoire vive, la RAM peut être accédée en lecture/écriture. Cependant si elle n'est pas alimentée, elle perd son contenu.

Red Hat
Entreprise commercialisant une distribution de Linux

ReiserFS
Système de fichiers Linux

ressources système
Cpu, mémoire, périphériques….

rétro-éclairage
Eclairage de l'écran LCD

RISC
Reduced Instruction Set Computing

ROM
Read Only Memory

scheduling
Programme du noyau chargé de gérer les différents processus

script
Fichier contenant du code interpréter par le Shell

SCSI
Small Computer System Interface

Shell
Interpreteur de commande qui peut être comparé à l'interface DOS.

SLIP
Serial Line Protocol

SMTP
Simple Mail Transfer Protocol

SNMP

Simple Network Management Protocol

socket
Point d'entrée réseau logiciel

SSH
Protocol de communication sécurisé

StrongARM
Type de processeur ARM pour systèmes embarqués

Suse
Entreprise commercialisant une distribution de Linux

swap
Méthode qui utilise la mémoire de masse pour remplacer la mémoire RAM

System V
Standard Unix définissant des normes telles que le
démarrage…

Système de fichiers
Type d'organisation des données sur un support de stockage

Système embarqué
Ordinateur de taille réduit en comparaison a un Pc de bureau. Il possède, généralement, des ressources limitées.

TCP
Transmission Protocol

Telnet
Application permettant l'accès à une machine distante à l'aide de commande.

Temps réel
Se dit d'un système devant répondre aux sollicitations de son environnement physique dans des délais précis

terminal
Clavier et écran qui se branche à une unité de travail.

Unix
Système d'exploitation professionnel, multi-utilisateurs et multi-tâches. Créé en 1974, il est entièrement écrit en C

uptime
Le nombre de jours pendant lesquels un système d'exploitation a fonctionné correctement

USB
Universal Serial Bus

VF
Ecran du type Vaccum Fluorecent ressemblant au LCD

VGA
Standard de sortie video

Wind River
Entreprise réalisant des systèmes d'exploitation industriels

Windows CE
Système d'exploitation de Microsoft pour les machines embarquée

Windows NT
Système d'exploitation professionnel de Microsoft

wrapper
Fichier indiquant au compilateur les chemins d'une librairie

X11, Xfree86, XWindows
Application principales servant à l'affichage graphique sous Linux

10.11 Cd-Rom et disquette