# Créer son premier RIM Linux !

12 août 2019

# Table des matières



RIM Linux, pour *Run In Memory* Linux, est un système GNU/Linux fonctionnant entièrement en mémoire vive.

N'importe quel système GNU/Linux moderne utilise la technologie du RIM. Comprendre cette technologie est donc un premier pas vers la réalisation de votre système.

Dans ce cas, pourquoi créer son propre système GNU/Linux ? Les raisons peuvent être multiples :

— Avoir un système parfaitement adapté à ses besoins.
— Créer sa propre distribution (oui oui, c'est possible).
— Tout simplement comprendre comment ça fonctionne.
— Et bien d'autres.

Ce tutoriel n'a pas pour objectif de vous apprendre tout cela, mais seulement de vous permettre de comprendre le fonctionnement et la création d'un RIM Linux. C'est donc un premier pas dans la création de distribution, rien de plus (mais rien de moins non plus ).

En effet, les technologies utilisées dans ce tutoriel, bien que généralement cachées par la distribution, sont indispensable tant pour la création d'un LiveCD (ou LiveUSB) que pour celle d'une distribution "classique". Donc normalement, quelque soit votre projet de création de système GNU/Linux (ou quelque_chose/Linux, coucou les systèmes Android), vous ne perdrez pas de temps en suivant ce tutoriel.

Pour ceux qui rêvent déjà : créer une distribution GNU/Linux à peu près utilisable est extrèmement chronophage, et parfois très pénible. Cela demande une grande tenacité. Le problème n'est pas de créer un système qui soit sympas, mais de le maintenir dans le temps, avec les dernières versions des programmes utilisés, et des développeurs qui ont des cycles de sorties radicalement différents.

Pour pouvoir suivre et comprendre ce tuto, il vous faut :

— Savoir utiliser sans paniquer la ligne de commande.
— Connaître les principes de base de l'arborescence UNIX (dont GNU/Linux fait partie).
— Avoir au moins une fois compilé un logiciel (juste pour voir ce que ça fait).

Nous allons donc parler de quelques points :

# 1. Nomenclature

### 1.0.1. Une question de vocabulaire

Contenu masqué n°1

Pour ma part, les dénominations étant multiples, j'utilise ce vocabulaire :

— `GNU/Linux` pour désigner précisément un système utilisant ces 2 briques.
— `Linux` pour désigner le noyau uniquement.
— `GNU` pour les programmes issus du projet GNU.

— `linux` ou `Tux` ou `tux` pour désigner un système UNIX utilisant le noyau Linux, que son système soit ou non fondé sur GNU.
— `système` pour parler de tout ce qui n'est pas le noyau.

# 2. La petite histoire

## 2.1. Petit constat

Le LiveCD Linux est une chose qui ne date pas d'hier. Le premier est Knoppix.

**2.1.0.1. Knoppix** Knoppix est une distribution fondée sur Debian et créée en 2000, la première à utiliser la technique du LiveCD. Elle est donc rapidement devenue populaire puisque cette spécificité technique est accompagnée de script de détection du matériel ainsi que d'outils de réparation du système, ce qui a permis d'avoir un SystemRescueCd avant l'heure. Comme toute distribution ancienne et relativement populaire, elle a donné naissance à un certain nombre de distributions dérivées .

---

Le LiveCD a pendant une période été une possibilité technique spécifique aux systèmes GNU/Linux. Pourtant, il n'existait pas aux premiers âges de ce système.

? Mais alors, comment faisait-on avant ?

Et bien comme nous allons le faire dans ce tuto ! Tout à la main ! Eh oui, GNU/Linux a été et reste un système "manuel".

Le LiveCD répond au départ à des impératifs techniques : on ne peut pas écrire sur un CD-R, et on n'a peut-être pas envie d'utiliser un CD-RW pour quelques utilisations seulement. Il faut donc pouvoir utiliser un système qui, de façon tout à fait naturelle, a besoin d'écrire dans son arborescence (ne serait-ce que pour les *logs*) alors que son support sera de toute façon en lecture seule. La solution la plus évidente est alors de faire entièrement tourner ce système grâce à une ressource présente sur chaque ordinateur : la RAM.

Mais avec le développement du LiveCD sont apparus d'autres usages :

— Distributions orientées sécurité ne laissant aucune trace sur l'ordinateur.
— Distributions axées sur la performance (la RAM permettant des performances spectaculaires par rapport à un disque dur), par exemple pour les ordinateurs anciens.
— Prolongement de la durée de vie des clés USB en limitant les écritures.

Néanmoins, pour les 2 derniers points, avoir un système "isolé" dans sa RAM n'est pas très pratique :

— Pas de sauvegarde de la configuration.
— Pas de sauvegarde des logiciels installés.
— En conséquence : temps de boot parfois long, car générique.

C'est alors que viennent les LiveUSB. Les LiveUSB sont globalement en RAM, mais une persistance (parfois partielle, parfois manuelle) est possible par l'utilisation de la clé sur un dossier particulier, éventuellement associé à des mécanismes de sauvegarde/restauration.

*i* Par la suite, nous allons d'abord créer un pur LiveCD, ce sera notre premier RIM-Linux, puis nous allons le transformer pour aboutir à un LiveUSB hybride.

# 3. Le noyau simplifié

## 3.1. Principe

Un court paragraphe pour vous présenter ce qu'est un noyau.

? Un noyau, c'est quoi ?

Le noyau, c'est *tout*. Le noyau d'un système d'exploitation (que nous nommerons par la suite OS), c'est tout ce que l'on ne voit pas, et qui pourtant est très utile. Petit exemple :

Chaque carte graphique possède son propre "langage" pour communiquer avec le système.

Vous imaginez si les développeurs devaient faire des applications qui étaient déclinées pour chaque modèle ? Et attention, pas seulement pour les modèles actuels, mais **pour ceux à venir** . Vous voyez bien que ce n'est pas gérable. C'est pour cela que l'on intègre au noyau des bouts de codes, appelés *driver*, qui permettent de communiquer au modèle en question. Le noyau propose un "langage" générique à l'application, qui n'a pas à s'occuper du matériel mais seulement de comment utiliser le matériel.

? Mais sur ma console, je ne vois pas où est l'OS ! Dans le jeu ?

Eh bien non, en fait, si on prend une console très basique type GameBoy, il n'y en a pas. En effet, le matériel est toujours le même, donc il n'y a pas besoin d'un OS !

? Donc un noyau, c'est un sac de drivers ?

Non. Le noyau, c'est *aussi* un sac de drivers. Mais ce n'est pas tout. C'est lui aussi qui permet de faire du multitâche, de gérer la mémoire, ...

Le noyau fait vraiment beaucoup de choses. Toujours pour simplifier la vie des développeurs et des utilisateurs.

## 3.2. Types de noyaux

La première chose qui vient à l'esprit, c'est un gros programme avec toutes les fonctionnalités dedans. C'est ce que l'on appelle un noyau *monolithique*. C'est simple, efficace, mais parfois dur à gérer. En effet, si vous faites un LiveCD générique, il faut bien supporter (théoriquement) tous les matériels existants, pour que votre LiveCD fonctionne sur tout les PCs existant.

Imaginons le PC d'un utilisateur. Celui-ci n'aura pas tout les matériels existant, donc il y aura des supports dans le noyau qui ne seront pas utilisé. On aura donc un gachi de place, puisque le noyau est plus gros que ce qu'il aurait pu être, et de temps parce qu'il a fallu charger tout ce code inutile en mémoire.

Il faudrait donc que les supports matériels qui ne sont pas très utilisés soient dans l'arborescence du système, et chargé en mémoire à la demande. C'est le concept du *micro-noyau*, où beaucoup de supports matériels sont "délocalisés" dans le système. L'inconvénient, c'est que l'on ne charge pas forcément tout les supports requis en même temps, du coup on risque de faire plein de petits accès disque au lieu d'un gros, ce qui risque de plomber le temps de boot, mais cette fois à la détection du matériel, pas lors du chargement du noyau en RAM (de plus, de par le fonctionnement interne d'un support "délocalisé", il est très généralement plus lent que son homologue intégré directement au noyau). On n'y est toujours pas .

Il y a donc 2 solutions intermédiaires, le noyau *monolithique modulaire*, qui est un noyau globalement monolithique, mais avec certains supports mis dans des *modules*, chargés à la demande, et le noyau hybride, qui est un micro-noyau avec des supports "relocalisés".

#### 3.2.0.1. En résumé :

- Noyau monolithique : un gros programme.
- Micro-noyau : un petit programme avec plein de petits modules.
- Noyau monolithique modulaire : un programme de taille intermédiaire avec quelques (petits ou gros) modules. Tend vers le noyau monolithique.
- Noyau hybride : même chose que le noyau monolithique modulaire, mais tend vers le micro-noyau.

Il existe aussi les noyaux mégalithiques et les exo-noyaux, mais ce sont plus des noyaux expérimentaux et aucun OS robuste et relativement connu n'utilise ces noyaux-là.

Si vous êtes intéressé par le sujet des noyaux, ou plus généralement des OS, ce tuto devrait vous plaire.

## 3.3. Linux

Le noyau Linux (car rappelons-le, Linux n'est qu'un noyau) est, c'est un de ses gros avantages, très configurable. Linux était originellement monolithique pur mais, depuis un certains nombre (élevé) d'années, est devenu modulaire. Toutefois, le support des modules est désactivable, ce qui fait que vous pouvez encore faire un noyau purement monolithique.

? Mais puisque ça prend plus de temps, quel est l'intérêt ?

Un noyau monolithique *générique* est moins bien qu'un monolithique modulaire. Si le noyau est optimisé, c'est-à-dire qu'il contient uniquement les supports dont on a besoin, il chargera en une seule passe tout les supports requis, sans avoir de code inutile de chargé. Il est donc plus efficace que le même noyau avec des modules.

### 3.4. Les choix de ce tuto

Nous allons utiliser un noyau monolithique pur, et ce pour quelques raisons :

— Pour apprendre au début, un noyau monolithique est plus simple à mettre en place (on n'a pas à gérer les modules qui "se baladent").
— Les parties noyau et système seront complètement isolées l'une de l'autre. Vous pourrez faire différentes versions de noyau et système, et les mélanger sans crainte.
— Cela vous poussera à optimiser votre noyau, et donc à entrer plus en détail dans la configuration du noyau.

## 4. Le boot simplifié

Après ce petit préliminaire de blabla, on entre dans la partie technique.

Mais avant de faire un système complet, il faut déjà comprendre comment il boote.

### 4.1. La chaîne du boot

C'est la suivante :

— BIOS (ou UEFI), qui fait quelques vérifications matérielles de base (par exemple il contrôle la présence de RAM).
— Le bootloader, chargé par le BIOS/(U)EFI, qui va charger en mémoire le noyau, lui passer des paramètres, puis l'exécuter.
— Le noyau, qui fait lui aussi des vérifications matérielles, puis qui réagit en fonction des paramètres passés par le bootloader.
— L'init, qui est le premier programme utilisateur (comprendre "pas du noyau") lancé, et ce par le noyau. L'init fait plusieurs choses, nous verrons quoi précisément par la suite, qui aboutissent à un OS prêt à l'emploi.

## 4.2. Le noyau lors du boot

Son paramètre principal, c'est l'endroit où est stocké le système (typiquement la partition montée sur `/`), qui va lui permettre de lancer l'init.

Si on est dans un noyau monolithique, pas de problème, il monte sa partition système, lance l'init et tout le monde est content.

Si on est dans un noyau monolithique modulaire, c'est plus problématique, car il se peut très bien que le système soit en ext4 ou en xfs, alors que le support de ces systèmes de fichiers (appelés aussi FS) est sous forme de module.

? Mais alors, il est où le problème ? Ça sera détecté non ?

Ah oui, parfaitement détecté. Mais, ils sont où les modules ? Dans la partition système !

Vous voyez le problème. C'est **exactement** comme dans les anciennes voitures avec les loquets, quand on fermait la porte de la voiture, loquet fermé, alors que la clé est à l'intérieur. Sauf que dans ces cas-là on pouvait briser la vitre. Il n'y a pas de vitre dans un FS. Dommage hein ?

La solution, c'est de passer par un système intermédiaire. Ce système intermédiaire, chargé en RAM par le noyau, contient tous les modules du noyau. Ce dernier peut donc monter sa partition système en toute tranquillité.

*i* Ce système intermédiaire, jusqu'aux noyaux 2.4 (inclus), c'était un initrd obligatoirement.

### 4.2.1. L'initrd

Contenu masqué n°2

### 4.2.2. L'initramfs

Contenu masqué n°3

### 4.3. Les choix de ce tuto

Nous allons utiliser un initramfs, car :

— C'est plus en accord avec l'objectif final de faire un LiveUSB.
— C'est plus simple à manipuler (du moins pour quelqu'un débutant dans le domaine), car le fonctionnement est celui d'une archive.
— On peut faire ce que fait un initrd avec un initramfs, ce choix ne sera donc pas limité uniquement à la création de linux live, mais pourra aussi servir pour une distribution avec un fonctionnement classique.

## 5. L'environnement de travail

Nous allons cannibaliser les bibliothèques de notre distribution, il faut donc que vous soyez sous tux. On va compiler et recompiler, créer plein de fichiers, alors pas question de mettre tout ça n'importe comment ! Créez dans votre dossier personnel un dossier `tux`, puis dedans un autre, par exemple `RIM` (ce sera la racine de notre projet), puis dedans :

— `build`
— `rootbase`
— `kernel`

Vous mettrez votre (ou vos) archive(s) de noyau dans le dossier kernel, que vous décompresserez à partir de celui-ci (vous aurez donc dans `kernel` un dossier par noyau), tout le reste dans `build`. Le dossier `rootbase` sera l'original de votre système alors pas question de le polluer avec des fichiers de compilation.

#### 5.0.1. Petit rappel sur la compilation

Compiler un logiciel c'est, dans l'ordre :

1. Télécharger les sources du dit logiciel
2. Décompresser les sources
3. **Lire le fichier README et/ou INSTALL**
4. Très généralement faire les étapes suivantes :
   — ./configure
   — make
   — make install

**5.0.1.1. Remarque sur la notation** Pour ceux qui ne seraient pas habitués à celle-ci, le fait de mettre un "`#`" en début de la ligne de commande dénote que la commande a besoin des droits root, donc que vous fassiez les commandes en tant que root ou avec `sudo` devant.

# 6. Premier RIM-Linux

#### 6.0.0.1. Ce que nous allons faire

1. Configurer et compiler un noyau, notre premier !
2. Configurer et compiler busybox.

## 6.1. Le noyau Linux

### 6.1.1. Les versions du noyau

*Majeur.mineur.publication-modification*

Exemples :

— 2.6.32
— 3.15.5-2

Contenu masqué n°4

### 6.1.2. Compilation

Allez on attaque la ligne de commande ! Allez sur le site du noyau linux , et prenez un noyau récent et stable en `longterm`. À l'heure où j'écris ceci, la version 3.14.12 est disponible, donc c'est sur celle-ci que je vais travailler. Mais ne vous en faites pas, la configuration change très peu d'une version à l'autre, surtout que nous n'allons pas utiliser les toutes dernières avancées du noyau. À la limite, même un noyau 2.6 pourrait faire l'affaire, mais on va éviter de tenter le diable.

Décompressez-le dans le dossier `kernel`, par exemple via la commande :

```
1 tar -xJf chemin_vers_le_noyau kernel/
```

Nous sommes ici situés à la racine du projet.

Le `J` indique que nous utilisons la compression xz, le `x` que nous voulons extraire (et non rajouter des fichiers dans l'archive), le `f` que l'archive est donnée par un fichier (et pas par l'entrée standard, soit par défaut le clavier et vos petits doigts).

Ensuite, pour ceux qui ne l'ont pas déjà fait, vous utilisez la ligne de commande.

> Mais c'est moche la ligne de commande !

Toi, tu sors ! Plus sérieusement, vous n'allez pas avoir le choix donc ce n'est pas le peine de râler.

Avant toute chose il faut aller dans le dossier du noyau, pour cela c'est bien entendu la commande `cd` :

```
cd kernel/linux-[VERSION]
```

Où [VERSION] est la version du noyau que vous utilisez.

La configuration du noyau se fait avec un configurateur appelé par *make*. Concrètement vous allez taper `make [configurateur]`, où [configurateur] peut être :

- `config` : là, c'est pour les durs, les rugueux du noyau. Une interminable série de questions. Je n'ai jamais réussi à faire les choses proprement avec celui-là.
- `menuconfig` : configurateur textuel, à peu près le même look que l'interface d'un BIOS. Donc pas forcément d'un esthétisme ébouriffant, mais efficace.
- `nconfig` : pareil que le précédent, mais plus joli. Non disponible sur un noyau 2.6.x .
- `xconfig` : configurateur graphique fonctionnant sous Qt.
- `gconfig` : configurateur graphique fonctionnant sous Gtk.

> Mais tu nous as menti ! On peut le configurer graphiquement !

Certes, sauf que vous êtes obligé de le lancer en console. Et puis de toute façon vous allez faire un tux en console, alors pourquoi ça vous embête ?

Pour les exemples, j'utilise un `make nconfig`qui donne en premier lieu ceci (sur un système 64 bits, un 32 aura des entrées légèrement différentes) :

> Contenu masqué n°5

Là, normalement, je devrais être sympa et vous donner un fichier de configuration déjà fait pour aller plus vite, ou lister rapidement les choses à cocher ou non. Et bien vous savez quoi ? Je ne vais pas le faire !

Il faut vraiment que vous fouilliez un peu dans la configuration du noyau, c'est indispensable. C'est comme vouloir faire du pain avec de la farine toute faite (farine + levure + arômes) plutôt que de le faire soi-même.

Je vais juste vous dire quelques points :

- Dans General Setup, **ne pas** cocher l'option `Embedded system`. Cela évitera de faire de grosses, grosses bêtises (vous ne pourrez faire que des petites ).
- Toujours dans General Setup, **laisser coché** le support de l'initrd/l'initramfs.

— **Décocher** `Enable loadable module support` à la racine du menu.
— Eviter de décocher les debugs, c'est votre premier, soyez sympas avec vous-même.
— N'oubliez pas de sauvegarder votre configuration.

Pour avoir quelque chose de sain pour votre architecture, vous pouvez, si vous le voulez, faire un `make defconfig` avant.

Après cela, vous tapez `make bzImage`. Cela va prendre un peu de temps. D'après mes expériences et des stats que j'ai glanées :

— 2 minutes sur un core i7 (-j8).
— ~45 minutes sur un pentium de base.
— Pas loin de 12h sur une rapsberry pi.

Si vous avez des stats à me proposer, transmettez-les par MP/sur le forum/mail, je ferais une mise à jour ici.

Pour accélérer la compilation sur des machines multi-coeurs, utilisez `make -jX` plutôt que `make`, avec X le nombre de coeurs que votre processeur possède.

## 6.2. Busybox

On télécharge ici , en prenant la version la plus récente (qui est ... tout en bas ). On décompresse l'archive dans `build`, et ensuite c'est un peu près la même chose qu'avec le noyau :

1. Configuration (via `make config` ou `make menuconfig`, les résultats étant identiques à ceux du noyau), en n'oubliant pas d'être dans le dossier racine de busybox (celui qui vient d'être sorti de l'archive).
2. Compilation (`make`).
3. Installation (`make install`).

Un menuconfig donne une jolie interface textuelle :

Contenu masqué n°6

La version de busybox que j'utilise est la `1.22.1`, et comme là c'est moins grave, je vous mets une configuration possible :

Contenu masqué n°7

Ce fichier sera enregistré dans la racine du dossier de busybox, et avec le nom `.config`.

Une fois le `make install` lancé (éventuellement avec l'option -j), nous avons un dossier `_install` situé dans la racine du dossier de busybox qui contient tout le système créé par busybox.

? Mais il y a des liens symboliques partout ! Pourquoi tu dis que c'est installé ?

En fait, tous nos programmes résident dans `_install/bin/busybox`. Ce programme va avoir un comportement différent selon le nom par lequel on l'appelle : c'est un *multi-call binairy*. Et les liens symboliques permettent de changer le nom par lequel on l'appelle. En conséquence, pour l'utilisateur il n'y a aucune différence (il tape toujours cp par exemple), mais par rapport aux coreutils de GNU, le fonctionnement interne est bien plus proche d'une philosophie "embarquée", puisqu'il n'y a qu'un seul programme (= bien plus léger, car une seule liste de dépendance, un seul entête ELF, ...).

## 6.3. Intégration à notre tux

Nous allons donc copier strictement tout ça vers rootbase. En admettant que le dossier `_install` se trouve dans `build/busybox-[version]` (soit le comportement par défaut : à la racine du dossier de busybox), la commande sera :

```
1 cp -a ./_install/* ../../rootbase/
```

! Dans cet exemple nous sommes situés à la racine du dossier de busybox.

Maintenant vient la cannibalisation de la distribution : on va pomper toutes les librairies. Sauf qu'on ne va copier que ce dont on a besoin. Pour ça nous avons une arme : `ldd`. Ce programme permet de connaître toutes les librairies (non statiquement) utilisées par un programme. Nous allons donc d'abord mettre les librairies utilisées par busybox avec ldd :

```
1 ldd rootbase/bin/busybox
```

! Dans cet exemple, nous sommes à la racine du projet.

Avec la configuration que je vous ai donnée, le retour de ldd est le suivant :

```
1 linux-gate.so.1 (0xb77a9000)
2 libm.so.6 => /usr/lib/libm.so.6 (0xb7739000)
```

```
libc.so.6 => /usr/lib/libc.so.6 (0xb7577000)
/lib/ld-linux.so.2 (0xb77aa000)
```

On va donc copier *strictement dans le même chemin* les librairies. C'est-à-dire :

- `libm` dans `rootbase/usr/lib`
- `libc` dans `rootbase/usr/lib`
- `ld-linux` dans `rootbase/lib`

On va donc, dans `rootbase`, créer les dossiers suivants :

- `lib`
- `usr/lib`

Si vous avez la flemme de tout recopier proprement, vous pouvez faire un lien symbolique de usr/lib vers lib, via

```
ln -s ../lib lib
```

en étant dans `rootbase/usr`, en lieu et place du dossier, comme ça vous mettez tout dans rootbase/lib et tout le monde est content (sauf les puristes, mais les puristes m'ont déjà tué et torturé 3 fois, alors au point où on en est... ).

Ensuite, dans `rootbase`, on crée les dossiers suivants :

- `etc`
- `dev`
- `opt`
- `tmp`
- `mnt`
- `root`
- `run`
- `var`

Après, on va optimiser un peu pour gagner de la place, avec `strip`, qui est un programme permettant d'enlever les symboles inutiles, notamment ceux permettant le déboguage. Ce n'est pas la peine de faire ça sur busybox, il est déjà strippé, on va faire cela sur les bibliothèques :

```
strip lib/*.so
strip usr/lib/*.so
```

Et pour finir, on va donner au noyau ce qu'il veut : son init. Positionnez-vous dans `rootbase`, puis créez le lien symbolique vers busybox :

```
ln -s bin/busybox init
```

Et voilà ! Plus qu'à tester tout ça.

# 7. Faire booter votre RIM-Linux

Nous allons voir comment packager notre tux pour qu'il puisse booter. D'abord, on va faire les étapes communes puis ensuite celles spécifiques à chacun d'entre vous.

## 7.1. La préparation du noyau et du système

> Cette étape n'est pas spécifique au premier RIM-Linux, elle s'applique aussi au 2e et à ceux d'après.

Nous n'allons pas faire un iso ! Et non ! Tout simplement parce que notre RIM est destiné à être mis sur une clé, donc on va se simplifier la vie.

### 7.1.1. Le noyau

Allez dans le dossier de création des binaires, pour beaucoup d'entre vous ce sera `arch/x86/boot`, et prenez le fichier `bzImage`. Mettez-le dans le dossier racine du projet, celui qui contient `root base`, `build`, `kernel`, pour ne pas le perdre.

### 7.1.2. Le système

Là, il va falloir faire une archive cpio, compressée de préférence. On va faire ça au moyen de ce petit script (que l'on met à la racine du projet également) :

Contenu masqué n°8

Le principe est le suivant : on crée une archive cpio (`cpio -o`) avec le format qui va bien (`-Hnewc`) compressée en gzip (`gzip`) à fond (`-9`, donc très compressé). `cpio` prend une liste de fichier que l'on génère grâce à `find`.

> Ce script sera appelé **en se positionnant à la racine du projet**.

Il créera à la racine du projet le fichier `RIM.cpio.gz`.

## 7.2. Identification de l'architecture

Pour ceux qui ne savent pas s'ils sont en BIOS ou UEFI, cette commande vous apportera la réponse :

```
# blkid -s PTTYPE -o value /dev/[périphérique_clé_usb]
```

L'option `-s` sert de filtre (on ne prend que les champs correspondant au type de la partition), et l'option `-o` spécifie ce que l'on veut tirer (la valeur de PTTYPE).

`[périphérique_clé_usb]` représente un fichier de périphérique (**pas une partition**, un périphérique entier). Si vous n'avez que votre disque dur, ce devrait être `sda` (je me répète, mais **pas** sda1 ou sda2, **sda**). Cette commande va vous renvoyer le type de la table des partitions du périphérique. Comme je suppose que vous avez tous un disque bien formaté comme il faut au départ, puisque vous pouvez l'utiliser, si la commande vous renvoie `dos`, c'est que vous êtes en BIOS, `gpt` en UEFI.

## 7.3. Préparation de la clé USB

Pendant cette étape de partitionnement, il faut **ABSOLUMENT** que votre clé ne soit pas montée.

Faisons une petite clé maison. Prenez une clé, et lancez un logiciel de partitionnement (je recommande *très fortement* gparted, mais un (c)fdisk peut faire l'affaire pour les spécialistes). Suivant votre architecture, vous allez refaire une table des partitions :

— msdos pour un BIOS.
— GPT pour un (U)EFI.

**ATTENTION !!** Réécrire la table des partitions est une opération *très courte, irréversible et destructrice* : vous allez perdre toutes vos partitions, ainsi que vos données. Toutes, sans exception.

Si vous faites une table gpt, les "grands pros" devront utiliser (c)gdisk, et non (c)fdisk.

Ensuite, quelle que soit votre architecture, vous allez créer une seule partition en FAT32 ou FAT16, et vous lui mettrez le drapeau (*flag* en anglais) `boot` (`bootable` pour certains logiciels). Ensuite vous écrivez vos partitions et quittez le logiciel.

On monte ensuite la clé, et on y crée le dossier `boot`, dans lequel on place notre noyau (le bzImage), et notre initramfs (RIM.cpio.gz).

## 7.4. Configuration spécifique au BIOS

**Les utilisateurs d'un (U)EFI peuvent sauter cette section.**

Contenu masqué n°9

## 7.5. Configuration spécifique à l'UEFI

**Les utilisateurs de BIOS peuvent sauter cette section.**

Contenu masqué n°10

## 7.6. Et pour finir ...

On démonte la clé, on redémarre (en ayant bien configuré son BIOS/(U)EFI pour que la clé passe avant le disque dur dans l'ordre de boot), et ça marche !

# 8. Second RIM-Linux

### 8.0.1. Réflexion

Nous avons donc créé un premier RIM-Linux ! Oui, mais...

- Il n'a aucune configuration (en terme d'init).
- Il ne supporte pas le hotplug.
- Il ne supporte pas la persistance.
- Il n'a aucune application.

Nous allons donc remédier à ces points :

## 8.1. La configuration de base

## 8.2. Le clavier

**8.2.0.1. Méthode classique** `kbd` est très certainement présent sur votre distribution. En vous mettant à la racine du projet, recopiez ces commandes :

```
cp /bin/loadkeys rootbase/bin
strip rootbase/bin/loadkeys
mkdir -p rootbase/usr/share/kbd/keymaps/i386/azerty
cp -a /usr/share/kbd/keymaps/i386/include
    rootbase/usr/share/kbd/keymaps/i386
cp /usr/share/kbd/keymaps/i386/azerty/fr-latin1.map.gz
    rootbase/usr/share/kbd/keymaps/i386/azerty/
```

Vous chargerez le clavier en mettant `/bin/loadkeys /usr/share/kbd/keymaps/i386/azerty/fr-latin1.map.gz` dans la section clavier des scripts d'init.

**8.2.0.2. Méthode busybox** Sur votre distribution, avec votre clavier bien configuré, faites un `dumpkmap > fr.kmap` (c'est un outil busybox, donc vous pouvez utiliser celui de `_install`). Vous le chargerez par la commande `loadkmap < chemin/vers/fr.kmap` dans la section clavier des scripts d'init. Une bonne solution est d'enregistrer le fichier produit dans `rootbase/etc/fr.kmap`.

### 8.2.1. Les FS virtuels

On va monter :

- **procfs** dans **/proc**, parce que c'est bien pratique.
- **sysfs** dans **/sys**, requis par le hotplug [m/u]dev.
- **devpts** dans **/dev/pts**, pour *el ratón* (la souris quoi).
- **usbfs** dans **/dev/bus/usb**, pour l'USB.
- **shm** dans **/dev/shm**, pour la mémoire partagée.

Pour se simplifier les choses (et surtout alléger - donc accélérer - les scripts init), on va faire dans les règles de l'art : par le fichier `etc/fstab` :

```
    proc        /proc               proc        defaults        0       0
    sysfs        /sys               sysfs        defaults        0       0
    devpts        /dev/pts        devpts        defaults        0       0
shm        /dev/shm        tmpfs        defaults        0       0
    usbfs        /dev/bus/usb        usbfs        defaults        0       0
```

Les champs étant tous séparés par une tabulation.

### 8.2.2. Les utilisateurs

Créez le fichier `rootbase/etc/passwd` avec le contenu suivant :

```
root:x:0:0:I am the master:/root:/bin/sh
```

Le fichier `rootbase/etc/group` :

```
root::0:root
```

Le fichier `rootbase/etc/shadow` :

```
root::9804:0:::::
```

### 8.2.3. Les premiers script init

Fichier `rootbase/etc/inittab` :

Contenu masqué n°11

Fichier `rootbase/etc/profile` :

Contenu masqué n°12

## 8.3. Le hotplug

Vous avez le choix entre la méthode busybox et la méthode classique.

#### 8.3.0.1. Méthode classique : udev

Contenu masqué n°13

#### 8.3.0.2. Méthode busybox : mdev

Contenu masqué n°14

## 8.4. La persistance

Je tiens à saluer ici l'équipe de Slitaz , dont j'ai entièrement repris le principe de persistance. Il tient en 2 points :

— La méthode la plus simple pour sauvegarder les données est de refaire l'initramfs.
— Le répertoire qui va être le plus long (le plus inutile ?) à archiver/compresser est `/home`.

Par conséquent, on va utiliser un script de sauvegarde, qu'il faudra lancer à la main, et on va monter la clé dans /home !

#### 8.4.0.1. Le script de sauvegarde

Créez le fichier `rootbase/usr/bin/save_liveusb.sh` avec ceci :

Contenu masqué n°15

#### 8.4.0.2. Le montage de /home

On va monter la partition de la clé sur /home, ce qui permettra, en plus de réduire le temps de création du nouvel initramfs, de permettre aux utilisateurs de bénéficier de tout l'espace disponible sur la clé ainsi que de sauvegarder (via notre script) une image modifiée.

Ce montage va se faire le plus simplement du monde par un `mount` dans les scripts d'init. Sauf que pour cela, il nous faut 2 informations :

1. La partition à monter.
2. Si on souhaite effectivement monter cette partition.

En effet, pour le point 2, on peut souhaiter utiliser la distribution uniquement en LiveCD (ce qui permet de retirer la clé après le boot).

? Comment passer ces informations ?

Au boot ! En effet, le bootloader passe des paramètres au noyau, qui va les mettre dans des variables d'environnements, qui sont donc exploitables par nos scripts.

**En pratique** Vous allez rajouter dans le champ `option` de votre bootloader deux paramètres, `home=[partition]`, où [partition] est égal à l'UUID.

**Exemple :** supposons que l'UUID de ma partition est `truc-muche`. Je mettrais donc `home=truc-muche`.

Ensuite, vous rajoutez le paramètre `ifmount`, qui prendra la valeur `yes` pour monter la partition, et `no` sinon.

**Exemple pour une entrée type LiveUSB :** dans `option`, on rajoute `home=truc-muche ifmount=yes`. Et voilà.

## 8.5. Finalisation

## 8.6. Fin des scripts d'init

Fichier `rootbase/etc/init.d/rcS` :

Contenu masqué n°16

Fichier `/etc/init.d/helper.rcs` :

Contenu masqué n°17

Fichier `/etc/init.d/rc.shutdown` :

Contenu masqué n°18

## 8.7. Les applications

## 8.8. Le réseau

**8.8.0.1. Connexion câblée** On a déjà les net-tools (ifconfig, route) par busybox, il ne vous reste donc qu'à faire un script pour spécifier l'adresse IP, le DNS et la route par défaut.

*En pratique :* Créez le fichier `opt/wired_net.sh` (vous pouvez le faire à partir de la distribution bootée si vous le souhaitez) avec :

```
1 ifconfig [interface] [adresse_ip]
2 route add -net default gw [adresse_ip_de_la_box_ou_du_rooteur]
```

Puis ensuite, on édite le fichier `etc/resolv.conf` et on y met :

```
nameserver [adresse_ip_du_serveur_dns]
```

? Oui mais moi je veux faire du DHCP !

Vous voulez faire du DHCP ? Et bien vous allez dans la configuration de busybox, vous cochez le client DHCP, et vous le faites tourner en root, et il se débrouille tout seul ! Rien à faire !

## 9. Pour aller plus loin

On peut envisager ces quelques points :

- Optimisation de la place : utilisation d'un noyau et d'un initramfs en LZMA, suppression de certaines fonctionnalités non essentielles.
- Accélération de la distribution : utilisation d'un noyau et d'un initramfs en LZO, utilisation de plusieurs points de montages (donc de nombreuses partitions sur la clé), utilisation des asmutils quand ceux-ci sont plus rapide que busybox.
- Ajout de logiciels.
- Intégration de tous les programmes permettant de développer sur la distribution.
- Création d'un installeur, d'une méthode de "vampirisation" d'un LiveUSB déjà existant.
- Création de scripts d'automatisation de configuration, notamment pour le réseau.
- Faire en sorte que la distribution soit plus adaptée avec un portable : wifi (via `wpa_supplicant`), économie d'énergie, par exemple avec l'arrêt complet des disques durs (voir du côté de `hdparm`).

Bref, vous avez du boulot.

---

Eh bien voilà ! Vous avez un système fonctionnel, avec support du hotplug et de la persistance. Vous pouvez maintenant ajouter ce que vous voulez à votre système, pour en faire, pourquoi pas, la prochaine distribution Linux qui déchire.

Vous avez toutes les armes pour conquérir le monde !

N'hésitez pas à utiliser le forum, notamment si problèmes techniques, ou tout simplement pour faire connaître votre distro.

Si ce tuto vous a plu (ou non d'ailleurs), que vous avez des remarques à faire, je suis preneur ! Entre les MP, les commentaires ici et le topic de présentation sur le forum , vous avez de la place pour vous exprimer.

Sinon, vous pouvez m'envoyer un mail à l'adresse suivante : dosmpm at gmail dot com (mais la réponse sera plus lente à venir)

*Sources et remerciements :*

— Un vieux numéro de *GNU/Linux Magazine France* (le n°80, de février 2006) pour le principe de base, le premier RIM-Linux, ainsi qu'une partie du second (udev, un bout des scripts d'init).
— La distribution Slitaz, pour le principe de la persistance.
— La documentation busybox, pour mdev.
— Le wiki archlinux pour l'installation détaillée de syslinux.
— Le site de rodsmith sur tux et l'UEFI, ainsi que le wiki d'archlinux, pour le boot en UEFI.

*Remerciements seuls :*

— La distro *core linux* (anciennement micro-core linux), une branche de Tiny Core Linux, pour m'avoir donné l'envie de faire mieux.
— La distribution Slackware, pour m'avoir permis de faire quelques tests de recompilation de noyau sans me cracher à la figure (je la recommande d'ailleurs si d'aventure vous souhaitez faire la même chose).
— La distribution Archlinux, pour m'avoir forcé à connaître wpa_supplicant et gummiboot. <3
— Ubuntu, pour m'avoir fait passer à d'autres distros pour plusieurs raisons.
— Le projet busybox, sans qui rien n'aurait été possible dans ce tuto.
— Le projet du noyau Linux, pour nous donner une stabilité et une performance très satisfaisante.
— Le projet GNU, pour avoir initié cette grande aventure qu'est le logiciel libre.
— ZdS pour permettre de partager tout cela.
— Et enfin Arius, qui me supporte, et c'est déjà beaucoup.

# Contenu masqué

## Contenu masqué n°1

Pour les puristes, sachez que nous n'allons pas faire un GNU/Linux ici. En effet, la partie système (coreutils notamment) sera gérée par busybox, qui n'est pas dans le projet GNU. C'est donc un BusyBox/Linux.

> Mais c'est complètement absurde !

Sachez que certains peuvent débattre pendant des pages et pages (sur un forum d'un matériel utilisant GNU/Linux, 53 pages. Oui oui, 53 pages là-dessus) sur la dénomination "correcte" de ce système. Retourner au texte.

## Contenu masqué n°2

Initrd, pour Initial Ramdisk, est, comme son nom l'indique, une sorte de partition en RAM. Dans sa version *a minima*, il contient l'interpréteur shell, le chargeur de module, les modules, et un fichier qui fera office d'init, `linuxrc`, qui est un simple fichier shell. Sa fonction est de charger les modules requis puis de repasser la main au noyau.

Cet initrd a une taille fixe. Par conséquent, on ne peut pas l'agrandir si on a besoin de place, ni le réduire s'il est trop grand.

Pour en faire un LiveCD, il suffit de dire au noyau que la partition système est la RAM, et c'est fini. On restera indéfiniment dans cet initrd. Plutôt simple non ?

Oui, mais un peu rigide. C'est pourquoi depuis les noyaux 2.6 il existe une alternative. Retourner au texte.

## Contenu masqué n°3

L'initramfs a un fonctionnement très différent.

Fondamentalement, ce n'est pas une partition, c'est une archive. Juste un fichier. Du coup, sa taille est indéterminée, et peut varier à notre guise (pas au-delà de la capacité de la RAM installée bien sûr).

Mais, c'est plus compliqué de rester en RAM. En effet, le noyau, lui, veut toujours monter une partition système.

La solution, c'est de ne pas rendre la main, de faire en sorte que notre init dans l'initramfs (oui c'est bien init, pas linuxrc) reste maître. Et après on est tranquille. Retourner au texte.

## Contenu masqué n°4

Actuellement, le majeur est à 3. Un mineur impair dénote un noyau de développement, un mineur pair, un noyau stable. Ce qui ne veut pas dire qu'un noyau stable est forcément plus vieux qu'un noyau de développement : ce sont 2 branches qui évoluent en parallèle, et quand une fonctionnalité pratique et stable est dans la branche développement, elle est progressivement incluse dans la branche stable.

La version de modification dénote un travail en aval de votre distribution (distro) sur le noyau, c'est donc un noyau qui sera légèrement différent des autres portant la même combinaison M.m.pub . Retourner au texte.

## Contenu masqué n°5

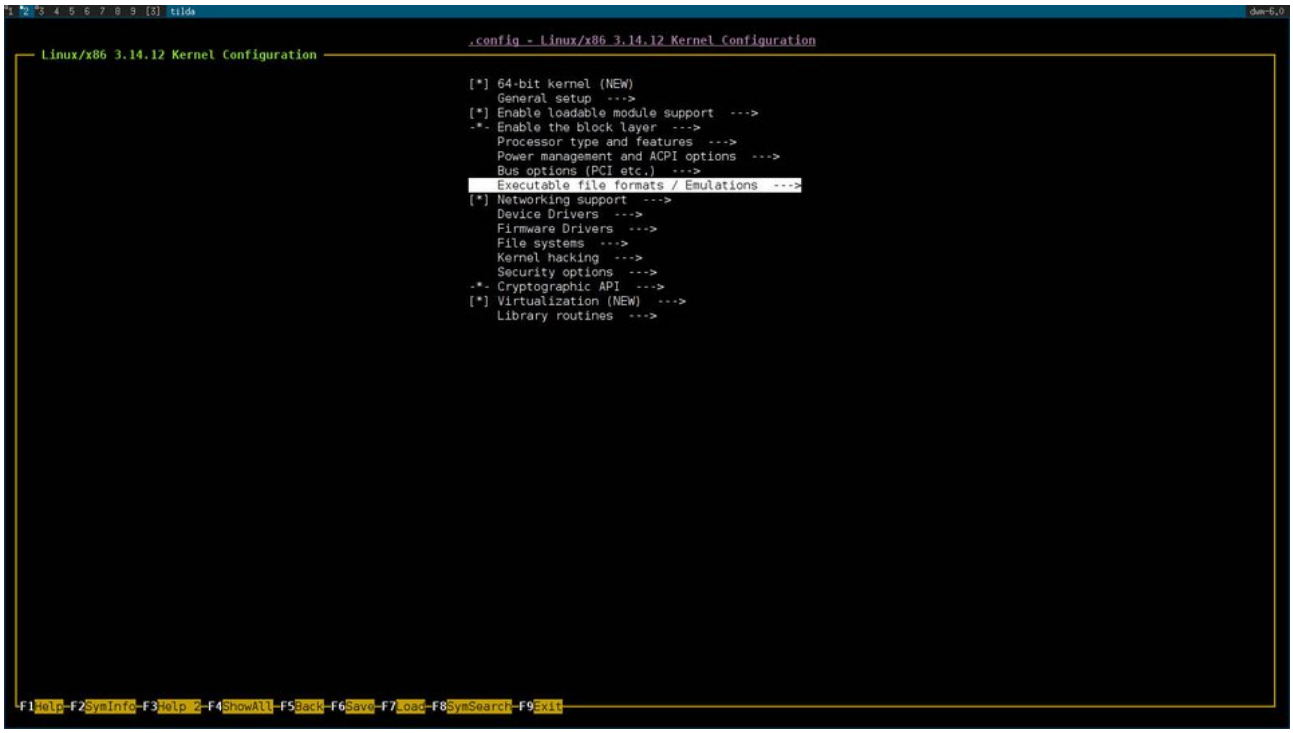


FIGURE 9. – Résultat de la commande : un beau petit menu !

Retourner au texte.

## Contenu masqué n°6

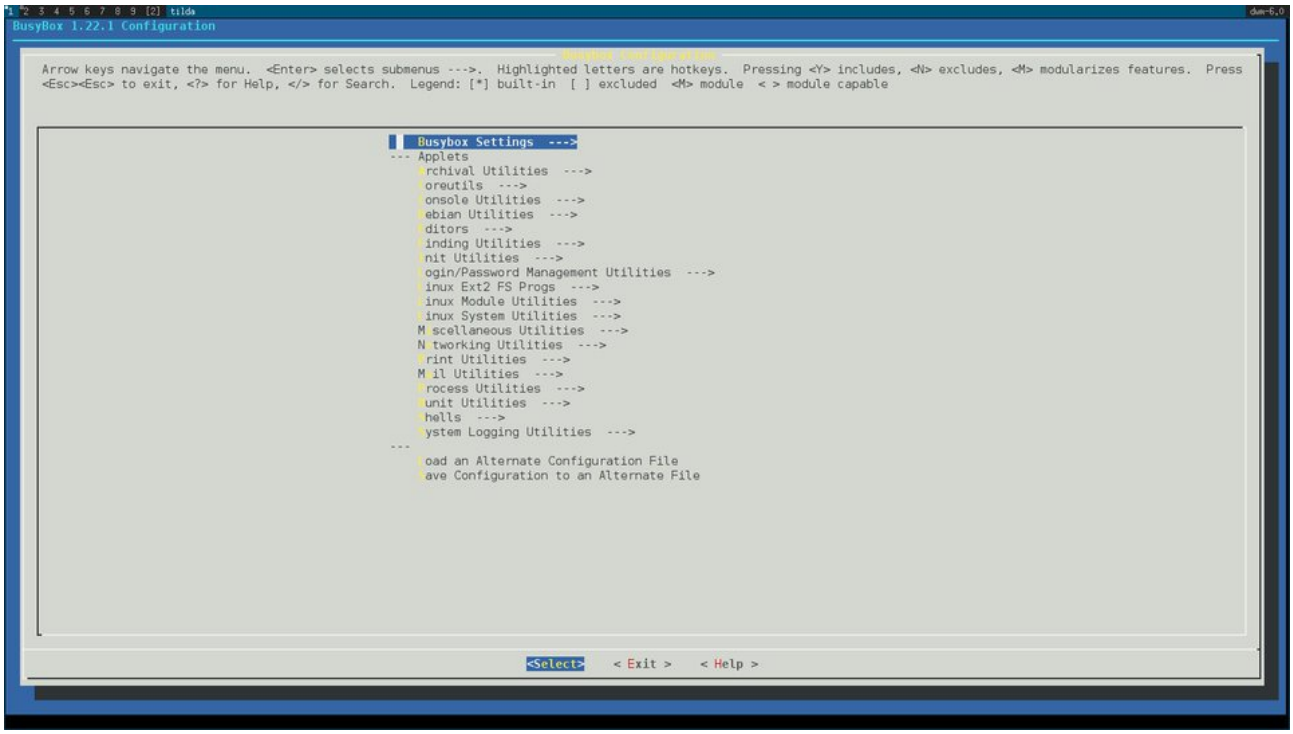

FIGURE 9. – Le configurateur de busybox

Retourner au texte.

## Contenu masqué n°7

```
#
# Automatically generated make config: don't edit
# Busybox version: 1.21.1
# Mon Aug 11 13:36:34 2014
#
CONFIG_HAVE_DOT_CONFIG=y

#
# Busybox Settings
#

#
# General Configuration
#
CONFIG_DESKTOP=y
# CONFIG_EXTRA_COMPAT is not set
# CONFIG_INCLUDE_SUSv2 is not set
# CONFIG_USE_PORTABLE_CODE is not set
CONFIG_PLATFORM_LINUX=y
CONFIG_FEATURE_BUFFERS_USE_MALLOC=y
# CONFIG_FEATURE_BUFFERS_GO_ON_STACK is not set
# CONFIG_FEATURE_BUFFERS_GO_IN_BSS is not set
CONFIG_SHOW_USAGE=y
CONFIG_FEATURE_VERBOSE_USAGE=y
CONFIG_FEATURE_COMPRESS_USAGE=y
CONFIG_FEATURE_INSTALLER=y
# CONFIG_INSTALL_NO_USR is not set
CONFIG_LOCALE_SUPPORT=y
CONFIG_UNICODE_SUPPORT=y
# CONFIG_UNICODE_USING_LOCALE is not set
CONFIG_FEATURE_CHECK_UNICODE_IN_ENV=y
CONFIG_SUBST_WCHAR=63
CONFIG_LAST_SUPPORTED_WCHAR=767
# CONFIG_UNICODE_COMBINING_WCHARS is not set
CONFIG_UNICODE_WIDE_WCHARS=y
# CONFIG_UNICODE_BIDI_SUPPORT is not set
# CONFIG_UNICODE_NEUTRAL_TABLE is not set
# CONFIG_UNICODE_PRESERVE_BROKEN is not set
CONFIG_LONG_OPTS=y
CONFIG_FEATURE_DEVPTS=y
# CONFIG_FEATURE_CLEAN_UP is not set
CONFIG_FEATURE_UTMP=y
```

*Contenu masqué*

```
CONFIG_FEATURE_WTMP=y
CONFIG_FEATURE_PIDFILE=y
CONFIG_PID_FILE_PATH="/var/run"
CONFIG_FEATURE_SUID=y
CONFIG_FEATURE_SUID_CONFIG=y
CONFIG_FEATURE_SUID_CONFIG_QUIET=y
# CONFIG_SELINUX is not set
# CONFIG_FEATURE_PREFER_APPLETS is not set
CONFIG_BUSYBOX_EXEC_PATH="/proc/self/exe"
CONFIG_FEATURE_SYSLOG=y
# CONFIG_FEATURE_HAVE_RPC is not set

#
# Build Options
#
# CONFIG_STATIC is not set
# CONFIG_PIE is not set
# CONFIG_NOMMU is not set
# CONFIG_BUILD_LIBBUSYBOX is not set
# CONFIG_FEATURE_INDIVIDUAL is not set
# CONFIG_FEATURE_SHARED_BUSYBOX is not set
CONFIG_LFS=y
CONFIG_CROSS_COMPILER_PREFIX=""
CONFIG_SYSROOT=""
CONFIG_EXTRA_CFLAGS="-Os"
CONFIG_EXTRA_LDFLAGS=""
CONFIG_EXTRA_LDLIBS=""

#
# Debugging Options
#
# CONFIG_DEBUG is not set
# CONFIG_DEBUG_PESSIMIZE is not set
# CONFIG_WERROR is not set
CONFIG_NO_DEBUG_LIB=y
# CONFIG_DMALLOC is not set
# CONFIG_EFENCE is not set

#
# Installation Options ("make install" behavior)
#
CONFIG_INSTALL_APPLET_SYMLINKS=y
# CONFIG_INSTALL_APPLET_HARDLINKS is not set
# CONFIG_INSTALL_APPLET_SCRIPT_WRAPPERS is not set
# CONFIG_INSTALL_APPLET_DONT is not set
# CONFIG_INSTALL_SH_APPLET_SYMLINK is not set
# CONFIG_INSTALL_SH_APPLET_HARDLINK is not set
# CONFIG_INSTALL_SH_APPLET_SCRIPT_WRAPPER is not set
CONFIG_PREFIX="./_install"

```

*Contenu masqué*

```
#
# Busybox Library Tuning
#
# CONFIG_FEATURE_SYSTEMD is not set
CONFIG_FEATURE_RTMINMAX=y
CONFIG_PASSWORD_MINLEN=6
CONFIG_MD5_SMALL=1
CONFIG_SHA3_SMALL=0
CONFIG_FEATURE_FAST_TOP=y
# CONFIG_FEATURE_ETC_NETWORKS is not set
CONFIG_FEATURE_USE_TERMIOS=y
CONFIG_FEATURE_EDITING=y
CONFIG_FEATURE_EDITING_MAX_LEN=1024
# CONFIG_FEATURE_EDITING_VI is not set
CONFIG_FEATURE_EDITING_HISTORY=255
# CONFIG_FEATURE_EDITING_SAVEHISTORY is not set
# CONFIG_FEATURE_EDITING_SAVE_ON_EXIT is not set
# CONFIG_FEATURE_REVERSE_SEARCH is not set
CONFIG_FEATURE_TAB_COMPLETION=y
CONFIG_FEATURE_USERNAME_COMPLETION=y
CONFIG_FEATURE_EDITING_FANCY_PROMPT=y
# CONFIG_FEATURE_EDITING_ASK_TERMINAL is not set
CONFIG_FEATURE_NON_POSIX_CP=y
CONFIG_FEATURE_VERBOSE_CP_MESSAGE=y
CONFIG_FEATURE_COPYBUF_KB=4
# CONFIG_FEATURE_SKIP_ROOTFS is not set
# CONFIG_MONOTONIC_SYSCALL is not set
CONFIG_IOCTL_HEX2STR_ERROR=y
# CONFIG_FEATURE_HWIB is not set

#
# Applets
#

#
# Archival Utilities
#
CONFIG_FEATURE_SEAMLESS_XZ=y
CONFIG_FEATURE_SEAMLESS_LZMA=y
CONFIG_FEATURE_SEAMLESS_BZ2=y
CONFIG_FEATURE_SEAMLESS_GZ=y
# CONFIG_FEATURE_SEAMLESS_Z is not set
# CONFIG_AR is not set
# CONFIG_FEATURE_AR_LONG_FILENAMES is not set
# CONFIG_FEATURE_AR_CREATE is not set
CONFIG_BUNZIP2=y
CONFIG_BZIP2=y
CONFIG_CPIO=y
CONFIG_FEATURE_CPIO_O=y
CONFIG_FEATURE_CPIO_P=y
```

*Contenu masqué*

```
# CONFIG_DPKG is not set
# CONFIG_DPKG_DEB is not set
# CONFIG_FEATURE_DPKG_DEB_EXTRACT_ONLY is not set
CONFIG_GUNZIP=y
CONFIG_GZIP=y
CONFIG_FEATURE_GZIP_LONG_OPTIONS=y
CONFIG_GZIP_FAST=1
CONFIG_LZOP=y
CONFIG_LZOP_COMPR_HIGH=y
# CONFIG_RPM2CPIO is not set
# CONFIG_RPM is not set
CONFIG_TAR=y
CONFIG_FEATURE_TAR_CREATE=y
CONFIG_FEATURE_TAR_AUTODETECT=y
CONFIG_FEATURE_TAR_FROM=y
CONFIG_FEATURE_TAR_OLDGNU_COMPATIBILITY=y
# CONFIG_FEATURE_TAR_OLDSUN_COMPATIBILITY is not set
CONFIG_FEATURE_TAR_GNU_EXTENSIONS=y
CONFIG_FEATURE_TAR_LONG_OPTIONS=y
CONFIG_FEATURE_TAR_TO_COMMAND=y
CONFIG_FEATURE_TAR_UNAME_GNAME=y
CONFIG_FEATURE_TAR_NOPRESERVE_TIME=y
# CONFIG_FEATURE_TAR_SELINUX is not set
# CONFIG_UNCOMPRESS is not set
CONFIG_UNLZMA=y
CONFIG_FEATURE_LZMA_FAST=y
CONFIG_LZMA=y
CONFIG_UNXZ=y
CONFIG_XZ=y
CONFIG_UNZIP=y

#
# Coreutils
#
CONFIG_BASENAME=y
CONFIG_CAT=y
CONFIG_DATE=y
CONFIG_FEATURE_DATE_ISOFMT=y
# CONFIG_FEATURE_DATE_NANO is not set
CONFIG_FEATURE_DATE_COMPAT=y
# CONFIG_HOSTID is not set
# CONFIG_ID is not set
# CONFIG_GROUPS is not set
CONFIG_TEST=y
CONFIG_FEATURE_TEST_64=y
CONFIG_TOUCH=y
CONFIG_FEATURE_TOUCH_SUSV3=y
CONFIG_TR=y
CONFIG_FEATURE_TR_CLASSES=y
CONFIG_FEATURE_TR_EQUIV=y
```

*Contenu masqué*

```
CONFIG_BASE64=y
CONFIG_WHO=y
CONFIG_USERS=y
CONFIG_CAL=y
CONFIG_CATV=y
CONFIG_CHGRP=y
CONFIG_CHMOD=y
CONFIG_CHOWN=y
CONFIG_FEATURE_CHOWN_LONG_OPTIONS=y
CONFIG_CHROOT=y
CONFIG_CKSUM=y
# CONFIG_COMM is not set
CONFIG_CP=y
CONFIG_FEATURE_CP_LONG_OPTIONS=y
CONFIG_CUT=y
CONFIG_DD=y
CONFIG_FEATURE_DD_SIGNAL_HANDLING=y
CONFIG_FEATURE_DD_THIRD_STATUS_LINE=y
CONFIG_FEATURE_DD_IBS_OBS=y
CONFIG_DF=y
CONFIG_FEATURE_DF_FANCY=y
CONFIG_DIRNAME=y
CONFIG_DOS2UNIX=y
CONFIG_UNIX2DOS=y
CONFIG_DU=y
CONFIG_FEATURE_DU_DEFAULT_BLOCKSIZE_1K=y
CONFIG_ECHO=y
CONFIG_FEATURE_FANCY_ECHO=y
CONFIG_ENV=y
CONFIG_FEATURE_ENV_LONG_OPTIONS=y
CONFIG_EXPAND=y
CONFIG_FEATURE_EXPAND_LONG_OPTIONS=y
CONFIG_EXPR=y
CONFIG_EXPR_MATH_SUPPORT_64=y
CONFIG_FALSE=y
CONFIG_FOLD=y
CONFIG_FSYNC=y
CONFIG_HEAD=y
CONFIG_FEATURE_FANCY_HEAD=y
# CONFIG_INSTALL is not set
# CONFIG_FEATURE_INSTALL_LONG_OPTIONS is not set
CONFIG_LN=y
CONFIG_LOGNAME=y
CONFIG_LS=y
CONFIG_FEATURE_LS_FILETYPES=y
CONFIG_FEATURE_LS_FOLLOWLINKS=y
CONFIG_FEATURE_LS_RECURSIVE=y
CONFIG_FEATURE_LS_SORTFILES=y
CONFIG_FEATURE_LS_TIMESTAMPS=y
CONFIG_FEATURE_LS_USERNAME=y
```

*Contenu masqué*

```
CONFIG_FEATURE_LS_COLOR=y
CONFIG_FEATURE_LS_COLOR_IS_DEFAULT=y
CONFIG_MD5SUM=y
CONFIG_MKDIR=y
CONFIG_FEATURE_MKDIR_LONG_OPTIONS=y
CONFIG_MKFIFO=y
CONFIG_MKNOD=y
CONFIG_MV=y
CONFIG_FEATURE_MV_LONG_OPTIONS=y
CONFIG_NICE=y
CONFIG_NOHUP=y
CONFIG_OD=y
CONFIG_PRINTENV=y
CONFIG_PRINTF=y
CONFIG_PWD=y
CONFIG_READLINK=y
CONFIG_FEATURE_READLINK_FOLLOW=y
# CONFIG_REALPATH is not set
CONFIG_RM=y
CONFIG_RMDIR=y
CONFIG_FEATURE_RMDIR_LONG_OPTIONS=y
CONFIG_SEQ=y
CONFIG_SHA1SUM=y
CONFIG_SHA256SUM=y
CONFIG_SHA512SUM=y
CONFIG_SHA3SUM=y
CONFIG_SLEEP=y
CONFIG_FEATURE_FANCY_SLEEP=y
CONFIG_FEATURE_FLOAT_SLEEP=y
CONFIG_SORT=y
CONFIG_FEATURE_SORT_BIG=y
CONFIG_SPLIT=y
CONFIG_FEATURE_SPLIT_FANCY=y
CONFIG_STAT=y
CONFIG_FEATURE_STAT_FORMAT=y
CONFIG_STTY=y
CONFIG_SUM=y
CONFIG_SYNC=y
CONFIG_TAC=y
CONFIG_TAIL=y
CONFIG_FEATURE_FANCY_TAIL=y
CONFIG_TEE=y
CONFIG_FEATURE_TEE_USE_BLOCK_IO=y
CONFIG_TRUE=y
CONFIG_TTY=y
CONFIG_UNAME=y
CONFIG_UNEXPAND=y
CONFIG_FEATURE_UNEXPAND_LONG_OPTIONS=y
CONFIG_UNIQ=y
CONFIG_USLEEP=y
```

*Contenu masqué*

```
# CONFIG_UUDECODE is not set
# CONFIG_UUENCODE is not set
CONFIG_WC=y
CONFIG_FEATURE_WC_LARGE=y
CONFIG_WHOAMI=y
CONFIG_YES=y

#
# Common options for cp and mv
#
CONFIG_FEATURE_PRESERVE_HARDLINKS=y

#
# Common options for ls, more and telnet
#
CONFIG_FEATURE_AUTOWIDTH=y

#
# Common options for df, du, ls
#
CONFIG_FEATURE_HUMAN_READABLE=y

#
# Common options for md5sum, sha1sum, sha256sum, sha512sum, sha3sum
#
CONFIG_FEATURE_MD5_SHA1_SUM_CHECK=y

#
# Console Utilities
#
# CONFIG_CHVT is not set
CONFIG_FGCONSOLE=y
CONFIG_CLEAR=y
# CONFIG_DEALLOCVT is not set
CONFIG_DUMPKMAP=y
CONFIG_KBD_MODE=y
CONFIG_LOADFONT=y
CONFIG_LOADKMAP=y
CONFIG_OPENVT=y
CONFIG_RESET=y
CONFIG_RESIZE=y
CONFIG_FEATURE_RESIZE_PRINT=y
CONFIG_SETCONSOLE=y
CONFIG_FEATURE_SETCONSOLE_LONG_OPTIONS=y
CONFIG_SETFONT=y
CONFIG_FEATURE_SETFONT_TEXTUAL_MAP=y
CONFIG_DEFAULT_SETFONT_DIR=""
CONFIG_SETKEYCODES=y
CONFIG_SETLOGCONS=y
CONFIG_SHOWKEY=y
```

```
343
344 #
345 # Common options for loadfont and setfont
346 #
347 CONFIG_FEATURE_LOADFONT_PSF2=y
348 CONFIG_FEATURE_LOADFONT_RAW=y
349
350 #
351 # Debian Utilities
352 #
353 CONFIG_MKTEMP=y
354 CONFIG_PIPE_PROGRESS=y
355 # CONFIG_RUN_PARTS is not set
356 # CONFIG_FEATURE_RUN_PARTS_LONG_OPTIONS is not set
357 # CONFIG_FEATURE_RUN_PARTS_FANCY is not set
358 # CONFIG_START_STOP_DAEMON is not set
359 # CONFIG_FEATURE_START_STOP_DAEMON_FANCY is not set
360 # CONFIG_FEATURE_START_STOP_DAEMON_LONG_OPTIONS is not set
361 CONFIG_WHICH=y
362
363 #
364 # Editors
365 #
366 CONFIG_PATCH=y
367 CONFIG_VI=y
368 CONFIG_FEATURE_VI_MAX_LEN=4096
369 # CONFIG_FEATURE_VI_8BIT is not set
370 CONFIG_FEATURE_VI_COLON=y
371 CONFIG_FEATURE_VI_YANKMARK=y
372 CONFIG_FEATURE_VI_SEARCH=y
373 CONFIG_FEATURE_VI_REGEX_SEARCH=y
374 CONFIG_FEATURE_VI_USE_SIGNALS=y
375 CONFIG_FEATURE_VI_DOT_CMD=y
376 CONFIG_FEATURE_VI_READONLY=y
377 CONFIG_FEATURE_VI_SETOPTS=y
378 CONFIG_FEATURE_VI_SET=y
379 CONFIG_FEATURE_VI_WIN_RESIZE=y
380 CONFIG_FEATURE_VI_ASK_TERMINAL=y
381 CONFIG_AWK=y
382 CONFIG_FEATURE_AWK_LIBM=y
383 CONFIG_CMP=y
384 CONFIG_DIFF=y
385 CONFIG_FEATURE_DIFF_LONG_OPTIONS=y
386 CONFIG_FEATURE_DIFF_DIR=y
387 CONFIG_ED=y
388 CONFIG_SED=y
389 CONFIG_FEATURE_ALLOW_EXEC=y
390
391 #
392 # Finding Utilities
```

*Contenu masqué*

```
#
CONFIG_FIND=y
CONFIG_FEATURE_FIND_PRINT0=y
CONFIG_FEATURE_FIND_MTIME=y
CONFIG_FEATURE_FIND_MMIN=y
CONFIG_FEATURE_FIND_PERM=y
CONFIG_FEATURE_FIND_TYPE=y
CONFIG_FEATURE_FIND_XDEV=y
CONFIG_FEATURE_FIND_MAXDEPTH=y
CONFIG_FEATURE_FIND_NEWER=y
CONFIG_FEATURE_FIND_INUM=y
CONFIG_FEATURE_FIND_EXEC=y
CONFIG_FEATURE_FIND_USER=y
CONFIG_FEATURE_FIND_GROUP=y
CONFIG_FEATURE_FIND_NOT=y
CONFIG_FEATURE_FIND_DEPTH=y
CONFIG_FEATURE_FIND_PAREN=y
CONFIG_FEATURE_FIND_SIZE=y
CONFIG_FEATURE_FIND_PRUNE=y
CONFIG_FEATURE_FIND_DELETE=y
CONFIG_FEATURE_FIND_PATH=y
CONFIG_FEATURE_FIND_REGEX=y
# CONFIG_FEATURE_FIND_CONTEXT is not set
CONFIG_FEATURE_FIND_LINKS=y
CONFIG_GREP=y
CONFIG_FEATURE_GREP_EGREP_ALIAS=y
CONFIG_FEATURE_GREP_FGREP_ALIAS=y
CONFIG_FEATURE_GREP_CONTEXT=y
CONFIG_XARGS=y
CONFIG_FEATURE_XARGS_SUPPORT_CONFIRMATION=y
CONFIG_FEATURE_XARGS_SUPPORT_QUOTES=y
CONFIG_FEATURE_XARGS_SUPPORT_TERMOPT=y
CONFIG_FEATURE_XARGS_SUPPORT_ZERO_TERM=y

#
# Init Utilities
#
# CONFIG_BOOTCHARTD is not set
# CONFIG_FEATURE_BOOTCHARTD_BLOATED_HEADER is not set
# CONFIG_FEATURE_BOOTCHARTD_CONFIG_FILE is not set
CONFIG_HALT=y
# CONFIG_FEATURE_CALL_TELINIT is not set
CONFIG_TELINIT_PATH=""
CONFIG_INIT=y
CONFIG_FEATURE_USE_INITTAB=y
# CONFIG_FEATURE_KILL_REMOVED is not set
CONFIG_FEATURE_KILL_DELAY=0
CONFIG_FEATURE_INIT_SCTTY=y
CONFIG_FEATURE_INIT_SYSLOG=y
CONFIG_FEATURE_EXTRA_QUIET=y
```

*Contenu masqué*

```
# CONFIG_FEATURE_INIT_COREDUMPS is not set
# CONFIG_FEATURE_INITRD is not set
CONFIG_INIT_TERMINAL_TYPE="linux"
CONFIG_MESG=y
CONFIG_FEATURE_MESG_ENABLE_ONLY_GROUP=y

#
# Login/Password Management Utilities
#
# CONFIG_ADD_SHELL is not set
# CONFIG_REMOVE_SHELL is not set
CONFIG_FEATURE_SHADOWPASSWDS=y
CONFIG_USE_BB_PWD_GRP=y
CONFIG_USE_BB_SHADOW=y
CONFIG_USE_BB_CRYPT=y
CONFIG_USE_BB_CRYPT_SHA=y
CONFIG_ADDUSER=y
CONFIG_FEATURE_ADDUSER_LONG_OPTIONS=y
# CONFIG_FEATURE_CHECK_NAMES is not set
CONFIG_FIRST_SYSTEM_ID=1000
CONFIG_LAST_SYSTEM_ID=9999
CONFIG_ADDGROUP=y
CONFIG_FEATURE_ADDGROUP_LONG_OPTIONS=y
CONFIG_FEATURE_ADDUSER_TO_GROUP=y
CONFIG_DELUSER=y
CONFIG_DELGROUP=y
CONFIG_FEATURE_DEL_USER_FROM_GROUP=y
CONFIG_GETTY=y
CONFIG_LOGIN=y
# CONFIG_LOGIN_SESSION_AS_CHILD is not set
# CONFIG_PAM is not set
CONFIG_LOGIN_SCRIPTS=y
# CONFIG_FEATURE_NOLOGIN is not set
# CONFIG_FEATURE_SECURETTY is not set
CONFIG_PASSWD=y
CONFIG_FEATURE_PASSWD_WEAK_CHECK=y
CONFIG_CRYPTPW=y
CONFIG_CHPASSWD=y
CONFIG_FEATURE_DEFAULT_PASSWD_ALGO="md5"
CONFIG_SU=y
CONFIG_FEATURE_SU_SYSLOG=y
CONFIG_FEATURE_SU_CHECKS_SHELLS=y
CONFIG_SULOGIN=y
# CONFIG_VLOCK is not set

#
# Linux Ext2 FS Progs
#
CONFIG_CHATTR=y
CONFIG_FSCK=y
```

*Contenu masqué*

```
CONFIG_LSATTR=y
CONFIG_TUNE2FS=y

#
# Linux Module Utilities
#
# CONFIG_MODINFO is not set
# CONFIG_MODPROBE_SMALL is not set
# CONFIG_FEATURE_MODPROBE_SMALL_OPTIONS_ON_CMDLINE is not set
# CONFIG_FEATURE_MODPROBE_SMALL_CHECK_ALREADY_LOADED is not set
# CONFIG_INSMOD is not set
# CONFIG_RMMOD is not set
# CONFIG_LSMOD is not set
# CONFIG_FEATURE_LSMOD_PRETTY_2_6_OUTPUT is not set
# CONFIG_MODPROBE is not set
# CONFIG_FEATURE_MODPROBE_BLACKLIST is not set
# CONFIG_DEPMOD is not set

#
# Options common to multiple modutils
#
# CONFIG_FEATURE_2_4_MODULES is not set
# CONFIG_FEATURE_INSMOD_TRY_MMAP is not set
# CONFIG_FEATURE_INSMOD_VERSION_CHECKING is not set
# CONFIG_FEATURE_INSMOD_KSYMOOPS_SYMBOLS is not set
# CONFIG_FEATURE_INSMOD_LOADINKMEM is not set
# CONFIG_FEATURE_INSMOD_LOAD_MAP is not set
# CONFIG_FEATURE_INSMOD_LOAD_MAP_FULL is not set
# CONFIG_FEATURE_CHECK_TAINTED_MODULE is not set
# CONFIG_FEATURE_MODUTILS_ALIAS is not set
# CONFIG_FEATURE_MODUTILS_SYMBOLS is not set
CONFIG_DEFAULT_MODULES_DIR=""
CONFIG_DEFAULT_DEPMOD_FILE=""

#
# Linux System Utilities
#
CONFIG_BLOCKDEV=y
CONFIG_MDEV=y
CONFIG_FEATURE_MDEV_CONF=y
CONFIG_FEATURE_MDEV_RENAME=y
CONFIG_FEATURE_MDEV_RENAME_REGEXP=y
CONFIG_FEATURE_MDEV_EXEC=y
CONFIG_FEATURE_MDEV_LOAD_FIRMWARE=y
CONFIG_REV=y
CONFIG_ACPID=y
CONFIG_FEATURE_ACPID_COMPAT=y
CONFIG_BLKID=y
# CONFIG_FEATURE_BLKID_TYPE is not set
CONFIG_DMESG=y
```

*Contenu masqué*

```
# CONFIG_FEATURE_DMESG_PRETTY is not set
CONFIG_FBSET=y
CONFIG_FEATURE_FBSET_FANCY=y
CONFIG_FEATURE_FBSET_READMODE=y
CONFIG_FDFLUSH=y
CONFIG_FDFORMAT=y
CONFIG_FDISK=y
# CONFIG_FDISK_SUPPORT_LARGE_DISKS is not set
CONFIG_FEATURE_FDISK_WRITABLE=y
# CONFIG_FEATURE_AIX_LABEL is not set
# CONFIG_FEATURE_SGI_LABEL is not set
# CONFIG_FEATURE_SUN_LABEL is not set
# CONFIG_FEATURE_OSF_LABEL is not set
CONFIG_FEATURE_GPT_LABEL=y
CONFIG_FEATURE_FDISK_ADVANCED=y
CONFIG_FINDFS=y
CONFIG_FLOCK=y
# CONFIG_FREERAMDISK is not set
# CONFIG_FSCK_MINIX is not set
CONFIG_MKFS_EXT2=y
# CONFIG_MKFS_MINIX is not set
# CONFIG_FEATURE_MINIX2 is not set
# CONFIG_MKFS_REISER is not set
CONFIG_MKFS_VFAT=y
# CONFIG_GETOPT is not set
# CONFIG_FEATURE_GETOPT_LONG is not set
CONFIG_HEXDUMP=y
CONFIG_FEATURE_HEXDUMP_REVERSE=y
# CONFIG_HD is not set
CONFIG_HWCLOCK=y
CONFIG_FEATURE_HWCLOCK_LONG_OPTIONS=y
# CONFIG_FEATURE_HWCLOCK_ADJTIME_FHS is not set
CONFIG_IPCRM=y
CONFIG_IPCS=y
CONFIG_LOSETUP=y
CONFIG_LSPCI=y
CONFIG_LSUSB=y
CONFIG_MKSWAP=y
CONFIG_FEATURE_MKSWAP_UUID=y
CONFIG_MORE=y
CONFIG_MOUNT=y
CONFIG_FEATURE_MOUNT_FAKE=y
CONFIG_FEATURE_MOUNT_VERBOSE=y
# CONFIG_FEATURE_MOUNT_HELPERS is not set
CONFIG_FEATURE_MOUNT_LABEL=y
# CONFIG_FEATURE_MOUNT_NFS is not set
CONFIG_FEATURE_MOUNT_CIFS=y
CONFIG_FEATURE_MOUNT_FLAGS=y
CONFIG_FEATURE_MOUNT_FSTAB=y
# CONFIG_PIVOT_ROOT is not set
```

*Contenu masqué*

```
CONFIG_RDATE=y
CONFIG_RDEV=y
CONFIG_READPROFILE=y
CONFIG_RTCWAKE=y
CONFIG_SCRIPT=y
CONFIG_SCRIPTREPLAY=y
# CONFIG_SETARCH is not set
CONFIG_SWAPONOFF=y
CONFIG_FEATURE_SWAPON_PRI=y
# CONFIG_SWITCH_ROOT is not set
CONFIG_UMOUNT=y
CONFIG_FEATURE_UMOUNT_ALL=y

#
# Common options for mount/umount
#
CONFIG_FEATURE_MOUNT_LOOP=y
CONFIG_FEATURE_MOUNT_LOOP_CREATE=y
# CONFIG_FEATURE_MTAB_SUPPORT is not set
CONFIG_VOLUMEID=y

#
# Filesystem/Volume identification
#
CONFIG_FEATURE_VOLUMEID_EXT=y
CONFIG_FEATURE_VOLUMEID_BTRFS=y
# CONFIG_FEATURE_VOLUMEID_REISERFS is not set
CONFIG_FEATURE_VOLUMEID_FAT=y
CONFIG_FEATURE_VOLUMEID_EXFAT=y
# CONFIG_FEATURE_VOLUMEID_HFS is not set
# CONFIG_FEATURE_VOLUMEID_JFS is not set
CONFIG_FEATURE_VOLUMEID_XFS=y
# CONFIG_FEATURE_VOLUMEID_NILFS is not set
CONFIG_FEATURE_VOLUMEID_NTFS=y
CONFIG_FEATURE_VOLUMEID_ISO9660=y
CONFIG_FEATURE_VOLUMEID_UDF=y
# CONFIG_FEATURE_VOLUMEID_LUKS is not set
CONFIG_FEATURE_VOLUMEID_LINUXSWAP=y
# CONFIG_FEATURE_VOLUMEID_CRAMFS is not set
# CONFIG_FEATURE_VOLUMEID_ROMFS is not set
# CONFIG_FEATURE_VOLUMEID_SQUASHFS is not set
# CONFIG_FEATURE_VOLUMEID_SYSV is not set
# CONFIG_FEATURE_VOLUMEID_OCFS2 is not set
# CONFIG_FEATURE_VOLUMEID_LINUXRAID is not set

#
# Miscellaneous Utilities
#
CONFIG_CONSPY=y
CONFIG_LESS=y
```

```
CONFIG_FEATURE_LESS_MAXLINES=9999999
CONFIG_FEATURE_LESS_BRACKETS=y
CONFIG_FEATURE_LESS_FLAGS=y
CONFIG_FEATURE_LESS_MARKS=y
CONFIG_FEATURE_LESS_REGEXP=y
CONFIG_FEATURE_LESS_WINCH=y
CONFIG_FEATURE_LESS_ASK_TERMINAL=y
CONFIG_FEATURE_LESS_DASHCMD=y
CONFIG_FEATURE_LESS_LINENUMS=y
# CONFIG_NANDWRITE is not set
# CONFIG_NANDDUMP is not set
CONFIG_SETSERIAL=y
# CONFIG_UBIATTACH is not set
# CONFIG_UBIDETACH is not set
# CONFIG_UBIMKVOL is not set
# CONFIG_UBIRMVOL is not set
# CONFIG_UBIRSVOL is not set
# CONFIG_UBIUPDATEVOL is not set
CONFIG_ADJTIMEX=y
CONFIG_BBCONFIG=y
CONFIG_FEATURE_COMPRESS_BBCONFIG=y
CONFIG_BEEP=y
CONFIG_FEATURE_BEEP_FREQ=4000
CONFIG_FEATURE_BEEP_LENGTH_MS=30
CONFIG_CHAT=y
CONFIG_FEATURE_CHAT_NOFAIL=y
# CONFIG_FEATURE_CHAT_TTY_HIFI is not set
CONFIG_FEATURE_CHAT_IMPLICIT_CR=y
CONFIG_FEATURE_CHAT_SWALLOW_OPTS=y
CONFIG_FEATURE_CHAT_SEND_ESCAPES=y
CONFIG_FEATURE_CHAT_VAR_ABORT_LEN=y
CONFIG_FEATURE_CHAT_CLR_ABORT=y
CONFIG_CHRT=y
CONFIG_CROND=y
CONFIG_FEATURE_CROND_D=y
CONFIG_FEATURE_CROND_CALL_SENDMAIL=y
CONFIG_FEATURE_CROND_DIR="/var/spool/cron"
# CONFIG_CRONTAB is not set
CONFIG_DC=y
CONFIG_FEATURE_DC_LIBM=y
# CONFIG_DEVFSD is not set
# CONFIG_DEVFSD_MODLOAD is not set
# CONFIG_DEVFSD_FG_NP is not set
# CONFIG_DEVFSD_VERBOSE is not set
# CONFIG_FEATURE_DEVFS is not set
CONFIG_DEVMEM=y
CONFIG_EJECT=y
CONFIG_FEATURE_EJECT_SCSI=y
CONFIG_FBSPLASH=y
# CONFIG_FLASHCP is not set
```

*Contenu masqué*

```
# CONFIG_FLASH_LOCK is not set
# CONFIG_FLASH_UNLOCK is not set
# CONFIG_FLASH_ERASEALL is not set
CONFIG_IONICE=y
# CONFIG_INOTIFYD is not set
CONFIG_LAST=y
CONFIG_FEATURE_LAST_SMALL=y
# CONFIG_FEATURE_LAST_FANCY is not set
CONFIG_HDPARM=y
CONFIG_FEATURE_HDPARM_GET_IDENTITY=y
# CONFIG_FEATURE_HDPARM_HDIO_SCAN_HWIF is not set
# CONFIG_FEATURE_HDPARM_HDIO_UNREGISTER_HWIF is not set
# CONFIG_FEATURE_HDPARM_HDIO_DRIVE_RESET is not set
# CONFIG_FEATURE_HDPARM_HDIO_TRISTATE_HWIF is not set
CONFIG_FEATURE_HDPARM_HDIO_GETSET_DMA=y
CONFIG_MAKEDEVS=y
# CONFIG_FEATURE_MAKEDEVS_LEAF is not set
CONFIG_FEATURE_MAKEDEVS_TABLE=y
CONFIG_MAN=y
# CONFIG_MICROCOM is not set
CONFIG_MOUNTPOINT=y
# CONFIG_MT is not set
# CONFIG_RAIDAUTORUN is not set
CONFIG_READAHEAD=y
CONFIG_RFKILL=y
CONFIG_RUNLEVEL=y
CONFIG_RX=y
CONFIG_SETSID=y
CONFIG_STRINGS=y
# CONFIG_TASKSET is not set
# CONFIG_FEATURE_TASKSET_FANCY is not set
CONFIG_TIME=y
CONFIG_TIMEOUT=y
CONFIG_TTYSIZE=y
CONFIG_VOLNAME=y
CONFIG_WALL=y
# CONFIG_WATCHDOG is not set

#
# Networking Utilities
#
CONFIG_NAMEIF=y
CONFIG_FEATURE_NAMEIF_EXTENDED=y
CONFIG_NBDCLIENT=y
CONFIG_NC=y
CONFIG_NC_SERVER=y
CONFIG_NC_EXTRA=y
# CONFIG_NC_110_COMPAT is not set
CONFIG_PING=y
CONFIG_PING6=y
```

*Contenu masqué*

```
CONFIG_FEATURE_FANCY_PING=y
CONFIG_WHOIS=y
CONFIG_FEATURE_IPV6=y
# CONFIG_FEATURE_UNIX_LOCAL is not set
CONFIG_FEATURE_PREFER_IPV4_ADDRESS=y
# CONFIG_VERBOSE_RESOLUTION_ERRORS is not set
CONFIG_ARP=y
CONFIG_ARPING=y
CONFIG_BRCTL=y
CONFIG_FEATURE_BRCTL_FANCY=y
CONFIG_FEATURE_BRCTL_SHOW=y
# CONFIG_DNSD is not set
CONFIG_ETHER_WAKE=y
CONFIG_FAKEIDENTD=y
# CONFIG_FTPD is not set
# CONFIG_FEATURE_FTP_WRITE is not set
# CONFIG_FEATURE_FTPD_ACCEPT_BROKEN_LIST is not set
CONFIG_FTPGET=y
CONFIG_FTPPUT=y
CONFIG_FEATURE_FTPGETPUT_LONG_OPTIONS=y
CONFIG_HOSTNAME=y
# CONFIG_HTTPD is not set
# CONFIG_FEATURE_HTTPD_RANGES is not set
# CONFIG_FEATURE_HTTPD_USE_SENDFILE is not set
# CONFIG_FEATURE_HTTPD_SETUID is not set
# CONFIG_FEATURE_HTTPD_BASIC_AUTH is not set
# CONFIG_FEATURE_HTTPD_AUTH_MD5 is not set
# CONFIG_FEATURE_HTTPD_CGI is not set
# CONFIG_FEATURE_HTTPD_CONFIG_WITH_SCRIPT_INTERPR is not set
# CONFIG_FEATURE_HTTPD_SET_REMOTE_PORT_TO_ENV is not set
# CONFIG_FEATURE_HTTPD_ENCODE_URL_STR is not set
# CONFIG_FEATURE_HTTPD_ERROR_PAGES is not set
# CONFIG_FEATURE_HTTPD_PROXY is not set
# CONFIG_FEATURE_HTTPD_GZIP is not set
CONFIG_IFCONFIG=y
CONFIG_FEATURE_IFCONFIG_STATUS=y
CONFIG_FEATURE_IFCONFIG_SLIP=y
CONFIG_FEATURE_IFCONFIG_MEMSTART_IOADDR_IRQ=y
CONFIG_FEATURE_IFCONFIG_HW=y
CONFIG_FEATURE_IFCONFIG_BROADCAST_PLUS=y
CONFIG_IFENSLAVE=y
CONFIG_IFPLUGD=y
CONFIG_IFUPDOWN=y
CONFIG_IFUPDOWN_IFSTATE_PATH="/var/run/ifstate"
CONFIG_FEATURE_IFUPDOWN_IP=y
CONFIG_FEATURE_IFUPDOWN_IP_BUILTIN=y
# CONFIG_FEATURE_IFUPDOWN_IFCONFIG_BUILTIN is not set
CONFIG_FEATURE_IFUPDOWN_IPV4=y
CONFIG_FEATURE_IFUPDOWN_IPV6=y
CONFIG_FEATURE_IFUPDOWN_MAPPING=y
```

*Contenu masqué*

```
# CONFIG_FEATURE_IFUPDOWN_EXTERNAL_DHCP is not set
# CONFIG_INETD is not set
# CONFIG_FEATURE_INETD_SUPPORT_BUILTIN_ECHO is not set
# CONFIG_FEATURE_INETD_SUPPORT_BUILTIN_DISCARD is not set
# CONFIG_FEATURE_INETD_SUPPORT_BUILTIN_TIME is not set
# CONFIG_FEATURE_INETD_SUPPORT_BUILTIN_DAYTIME is not set
# CONFIG_FEATURE_INETD_SUPPORT_BUILTIN_CHARGEN is not set
# CONFIG_FEATURE_INETD_RPC is not set
CONFIG_IP=y
CONFIG_FEATURE_IP_ADDRESS=y
CONFIG_FEATURE_IP_LINK=y
CONFIG_FEATURE_IP_ROUTE=y
CONFIG_FEATURE_IP_TUNNEL=y
CONFIG_FEATURE_IP_RULE=y
CONFIG_FEATURE_IP_SHORT_FORMS=y
# CONFIG_FEATURE_IP_RARE_PROTOCOLS is not set
CONFIG_IPADDR=y
CONFIG_IPLINK=y
CONFIG_IPROUTE=y
CONFIG_IPTUNNEL=y
CONFIG_IPRULE=y
CONFIG_IPCALC=y
CONFIG_FEATURE_IPCALC_FANCY=y
CONFIG_FEATURE_IPCALC_LONG_OPTIONS=y
CONFIG_NETSTAT=y
CONFIG_FEATURE_NETSTAT_WIDE=y
CONFIG_FEATURE_NETSTAT_PRG=y
CONFIG_NSLOOKUP=y
CONFIG_NTPD=y
CONFIG_FEATURE_NTPD_SERVER=y
CONFIG_PSCAN=y
CONFIG_ROUTE=y
CONFIG_SLATTACH=y
CONFIG_TCPSVD=y
CONFIG_TELNET=y
CONFIG_FEATURE_TELNET_TTYPE=y
CONFIG_FEATURE_TELNET_AUTOLOGIN=y
CONFIG_TELNETD=y
CONFIG_FEATURE_TELNETD_STANDALONE=y
CONFIG_FEATURE_TELNETD_INETD_WAIT=y
CONFIG_TFTP=y
# CONFIG_TFTPD is not set

#
# Common options for tftp/tftpd
#
CONFIG_FEATURE_TFTP_GET=y
CONFIG_FEATURE_TFTP_PUT=y
CONFIG_FEATURE_TFTP_BLOCKSIZE=y
CONFIG_FEATURE_TFTP_PROGRESS_BAR=y
```

*Contenu masqué*

```
# CONFIG_TFTP_DEBUG is not set
CONFIG_TRACEROUTE=y
CONFIG_TRACEROUTE6=y
CONFIG_FEATURE_TRACEROUTE_VERBOSE=y
# CONFIG_FEATURE_TRACEROUTE_SOURCE_ROUTE is not set
CONFIG_FEATURE_TRACEROUTE_USE_ICMP=y
CONFIG_TUNCTL=y
CONFIG_FEATURE_TUNCTL_UG=y
CONFIG_UDHCPC6=y
# CONFIG_UDHCPD is not set
# CONFIG_DHCPRELAY is not set
# CONFIG_DUMPLEASES is not set
# CONFIG_FEATURE_UDHCPD_WRITE_LEASES_EARLY is not set
# CONFIG_FEATURE_UDHCPD_BASE_IP_ON_MAC is not set
CONFIG_DHCPD_LEASES_FILE=""
CONFIG_UDHCPC=y
CONFIG_FEATURE_UDHCPC_ARPING=y
# CONFIG_FEATURE_UDHCP_PORT is not set
CONFIG_UDHCP_DEBUG=9
CONFIG_FEATURE_UDHCP_RFC3397=y
CONFIG_FEATURE_UDHCP_8021Q=y
CONFIG_UDHCPC_DEFAULT_SCRIPT="/usr/share/udhcpc/default.script"
CONFIG_UDHCPC_SLACK_FOR_BUGGY_SERVERS=80
CONFIG_IFUPDOWN_UDHCPC_CMD_OPTIONS="-R -n"
CONFIG_UDPSVD=y
CONFIG_VCONFIG=y
CONFIG_WGET=y
CONFIG_FEATURE_WGET_STATUSBAR=y
CONFIG_FEATURE_WGET_AUTHENTICATION=y
CONFIG_FEATURE_WGET_LONG_OPTIONS=y
CONFIG_FEATURE_WGET_TIMEOUT=y
CONFIG_ZCIP=y

#
# Print Utilities
#
# CONFIG_LPD is not set
# CONFIG_LPR is not set
# CONFIG_LPQ is not set

#
# Mail Utilities
#
CONFIG_MAKEMIME=y
CONFIG_FEATURE_MIME_CHARSET="us-ascii"
CONFIG_POPMAILDIR=y
CONFIG_FEATURE_POPMAILDIR_DELIVERY=y
CONFIG_REFORMIME=y
CONFIG_FEATURE_REFORMIME_COMPAT=y
CONFIG_SENDMAIL=y
```

*Contenu masqué*

```
893
894 #
895 # Process Utilities
896 #
897 CONFIG_IOSTAT=y
898 CONFIG_LSOF=y
899 CONFIG_MPSTAT=y
900 CONFIG_NMETER=y
901 CONFIG_PMAP=y
902 CONFIG_POWERTOP=y
903 CONFIG_PSTREE=y
904 CONFIG_PWDX=y
905 CONFIG_SMEMCAP=y
906 CONFIG_TOP=y
907 CONFIG_FEATURE_TOP_CPU_USAGE_PERCENTAGE=y
908 CONFIG_FEATURE_TOP_CPU_GLOBAL_PERCENTS=y
909 CONFIG_FEATURE_TOP_SMP_CPU=y
910 CONFIG_FEATURE_TOP_DECIMALS=y
911 CONFIG_FEATURE_TOP_SMP_PROCESS=y
912 CONFIG_FEATURE_TOPMEM=y
913 CONFIG_UPTIME=y
914 CONFIG_FEATURE_UPTIME_UTMP_SUPPORT=y
915 CONFIG_FREE=y
916 CONFIG_FUSER=y
917 CONFIG_KILL=y
918 CONFIG_KILLALL=y
919 CONFIG_KILLALL5=y
920 CONFIG_PGREP=y
921 CONFIG_PIDOF=y
922 CONFIG_FEATURE_PIDOF_SINGLE=y
923 CONFIG_FEATURE_PIDOF_OMIT=y
924 CONFIG_PKILL=y
925 CONFIG_PS=y
926 # CONFIG_FEATURE_PS_WIDE is not set
927 # CONFIG_FEATURE_PS_LONG is not set
928 CONFIG_FEATURE_PS_TIME=y
929 CONFIG_FEATURE_PS_ADDITIONAL_COLUMNS=y
930 # CONFIG_FEATURE_PS_UNUSUAL_SYSTEMS is not set
931 CONFIG_RENICE=y
932 CONFIG_BB_SYSCTL=y
933 CONFIG_FEATURE_SHOW_THREADS=y
934 CONFIG_WATCH=y
935
936 #
937 # Runit Utilities
938 #
939 # CONFIG_RUNSV is not set
940 # CONFIG_RUNSVDIR is not set
941 # CONFIG_FEATURE_RUNSVDIR_LOG is not set
942 # CONFIG_SV is not set
```

*Contenu masqué*

```
CONFIG_SV_DEFAULT_SERVICE_DIR=""
# CONFIG_SVLOGD is not set
CONFIG_CHPST=y
CONFIG_SETUIDGID=y
CONFIG_ENVUIDGID=y
CONFIG_ENVDIR=y
# CONFIG_SOFTLIMIT is not set
# CONFIG_CHCON is not set
# CONFIG_FEATURE_CHCON_LONG_OPTIONS is not set
# CONFIG_GETENFORCE is not set
# CONFIG_GETSEBOOL is not set
# CONFIG_LOAD_POLICY is not set
# CONFIG_MATCHPATHCON is not set
# CONFIG_RESTORECON is not set
# CONFIG_RUNCON is not set
# CONFIG_FEATURE_RUNCON_LONG_OPTIONS is not set
# CONFIG_SELINUXENABLED is not set
# CONFIG_SETENFORCE is not set
# CONFIG_SETFILES is not set
# CONFIG_FEATURE_SETFILES_CHECK_OPTION is not set
# CONFIG_SETSEBOOL is not set
# CONFIG_SESTATUS is not set

#
# Shells
#
CONFIG_ASH=y
CONFIG_ASH_BASH_COMPAT=y
# CONFIG_ASH_IDLE_TIMEOUT is not set
CONFIG_ASH_JOB_CONTROL=y
CONFIG_ASH_ALIAS=y
CONFIG_ASH_GETOPTS=y
CONFIG_ASH_BUILTIN_ECHO=y
CONFIG_ASH_BUILTIN_PRINTF=y
CONFIG_ASH_BUILTIN_TEST=y
CONFIG_ASH_CMDCMD=y
# CONFIG_ASH_MAIL is not set
CONFIG_ASH_OPTIMIZE_FOR_SIZE=y
CONFIG_ASH_RANDOM_SUPPORT=y
CONFIG_ASH_EXPAND_PRMT=y
# CONFIG_CTTYHACK is not set
# CONFIG_HUSH is not set
# CONFIG_HUSH_BASH_COMPAT is not set
# CONFIG_HUSH_BRACE_EXPANSION is not set
# CONFIG_HUSH_HELP is not set
# CONFIG_HUSH_INTERACTIVE is not set
# CONFIG_HUSH_SAVEHISTORY is not set
# CONFIG_HUSH_JOB is not set
# CONFIG_HUSH_TICK is not set
# CONFIG_HUSH_IF is not set
```

*Contenu masqué*

```
# CONFIG_HUSH_LOOPS is not set
# CONFIG_HUSH_CASE is not set
# CONFIG_HUSH_FUNCTIONS is not set
# CONFIG_HUSH_LOCAL is not set
# CONFIG_HUSH_RANDOM_SUPPORT is not set
# CONFIG_HUSH_EXPORT_N is not set
# CONFIG_HUSH_MODE_X is not set
# CONFIG_MSH is not set
CONFIG_FEATURE_SH_IS_ASH=y
# CONFIG_FEATURE_SH_IS_HUSH is not set
# CONFIG_FEATURE_SH_IS_NONE is not set
CONFIG_FEATURE_BASH_IS_ASH=y
# CONFIG_FEATURE_BASH_IS_HUSH is not set
# CONFIG_FEATURE_BASH_IS_NONE is not set
CONFIG_SH_MATH_SUPPORT=y
CONFIG_SH_MATH_SUPPORT_64=y
CONFIG_FEATURE_SH_EXTRA_QUIET=y
# CONFIG_FEATURE_SH_STANDALONE is not set
# CONFIG_FEATURE_SH_NOFORK is not set
CONFIG_FEATURE_SH_HISTFILESIZE=y

#
# System Logging Utilities
#
CONFIG_SYSLOGD=y
CONFIG_FEATURE_ROTATE_LOGFILE=y
CONFIG_FEATURE_REMOTE_LOG=y
CONFIG_FEATURE_SYSLOGD_DUP=y
CONFIG_FEATURE_SYSLOGD_CFG=y
CONFIG_FEATURE_SYSLOGD_READ_BUFFER_SIZE=256
CONFIG_FEATURE_IPC_SYSLOG=y
CONFIG_FEATURE_IPC_SYSLOG_BUFFER_SIZE=4
CONFIG_LOGREAD=y
CONFIG_FEATURE_LOGREAD_REDUCED_LOCKING=y
CONFIG_FEATURE_KMSG_SYSLOG=y
CONFIG_KLOGD=y

#
# klogd should not be used together with syslog to kernel printk
    buffer
#
CONFIG_FEATURE_KLOGD_KLOGCTL=y
# CONFIG_LOGGER is not set
```

Retourner au texte.

## Contenu masqué n°8

```
#!/bin/sh
ICI=$(pwd)
CHEMIN=./rootbase
echo "Le chemin de création est $CHEMIN."
echo "Création en cours ..."
cd $CHEMIN
find ./* -print | cpio -o -Hnewc > $ICI/RIM.cpio
cd $ICI
cat RIM.cpio | gzip -9 > RIM.cpio.gz
echo "Fait."
echo "Faites-vous plaisir avec votre système."
```

Retourner au texte.

## Contenu masqué n°9

Le bootloader utilisé va être syslinux. Attention aux confusions, car syslinux est une ”suite” de bootloader qui comporte :

— syslinux : pour booter à partir d'une FAT32.
— isolinux : pour booter à partir d'un CD.
— extlinux : pour booter à partir d'un ext[2/3/4] ou d'un btrfs.
— pxelinux : pour booter en utilisant la technologie PXE (boot à partir du réseau).

Nous allons utiliser, parmi syslinux, syslinux. Clair ? Nous aurions pu choisir extlinux, mais une clé en FAT32 a le gros avantage d'être compatible avec un nombre cataclysmique d'OS (dont Windows et OS X), ce qui est toujours appréciable.

Les raisons du choix de cette suite :

— Les différents bootloader de la suite obéissent à la même syntaxe en ce qui concerne la configuration.
— Les bootloader de cette suite sont utilisables dans nombre de situations, pas uniquement celle qui nous intéresse ici.
— Les bootloader de cette suite ont la réputation d'être légers (et robustes).
— Les bootloader de cette suite peuvent changer dynamiquement de configuration (je pense à lilo, qui n'a pas forcément cette souplesse).
— Il est relativement simple de créer un menu (même graphique) avec ces bootloader.

### 9.0.1. Installation de syslinux

On télécharge syslinux ici (encore une fois, le plus récent est tout en bas ), on décompresse dans `build`, **et on ne compile pas**.

? Mais ... Pourquoi ?

Parce que les binaires sont déjà inclus, et qu'on ne va pas repasser du temps sur un truc qui doit de toute façon être un binaire générique.

On va créer le dossier `boot/syslinux` sur la clé, et on va y mettre :

- `bios/com32/chain/chain.c32`
- `bios/com32/menu/menu.c32`
- `bios/com32/modules/poweroff.c32`
- `bios/com32/modules/reboot.c32`
- `bios/com32/modules/config.c32`
- `bios/com32/elflink/ldlinux/ldlinux.c32`
- `bios/com32/libutil/libutil.c32`

*Note :* les chemins sont donnés à partir de la racine de l'archive décompressée de syslinux. De plus, on ne recrée pas les dossiers en question, on met tous les fichiers dans `boot/syslinux`, sans sous-dossiers.

Ensuite, on fait la commande suivante :

```
# [chemin_vers_racine_syslinux]/bios/extlinux/extlinux --install
    [chemin_vers_clé_usb]/boot/syslinux
```

Cette commande va mettre le fichier `ldlinux.sys` dans le dossier `boot/syslinux`, et faire "pointer" le début de la partition en question sur ce fichier. Il faut donc absolument **NE PLUS TOUCHER À LDLINUX.SYS**. Tout mouvement sur ce fichier rendra caduc le boot.

#### 9.0.1.1. Ecriture de la MBR

```
# dd bs=440 count=1 conv=notrunc
    if=[chemin_vers_syslinux]/bios/mbr/mbr.bin
    of=/dev/[périphérique_clé_usb]
```

On fait une copie bit-à-bit (`dd`) de 1 (`count=1`) bloc de 440 octets (`bs=440`) sans troncature (`conv=notrunc`) de la source (`if=`) à la destination (`of=`).

**9.0.1.2. Configuration de syslinux** Dans `boot/syslinux`, créez un fichier syslinux.cfg. Ce sera la configuration de base. Pour ceux qui n'ont pas envie de passer des heures dans la doc de syslinux, voici une configuration possible :

```
UI menu.c32
PROMPT 0
```

```
3
4  MENU TITLE Boot Menu
5  TIMEOUT 50
6  DEFAULT rim
7
8  LABEL rim
9          MENU LABEL RIM-Linux
10         LINUX ../bzImage
11         APPEND root=none rw
12         INITRD ../RIM.cpio.gz
13
14 LABEL poweroff
15         MENU LABEL Power off
16         COM32 poweroff.c32
17
18 LABEL reboot
19         MENU LABEL Reboot
20         COM32 reboot.c32
```

Retourner au texte.

## Contenu masqué n°10

Il va nous falloir recréer l'arborescence de base de l'UEFI. On va donc créer 2 dossiers sur la clé :

- `EFI`
- `EFI/Boot`

### 9.0.2. Le grand choix

La suite syslinux peut être utilisée (comme pour le BIOS), pour l'UEFI, **mais** (et c'est le plus embêtant) il n'y a pas de support du chainload.

?

Gné ? Chainload ?

Le chainload c'est quand votre bootloader ne sait pas charger un OS particulier. Il va alors passer la main à un autre bootloader (en fait à un binaire quelconque) qui lui va faire le boulot. Tout ça pour vous dire que vous ne pourrez charger que des OS type UNIX avec syslinux UEFI.

Je vous propose donc une alternative : `gummiboot`. Ce programme est fait uniquement pour l'UEFI, mais ce n'est pas un bootloader, c'est un `bootmanager`, c'est-à-dire qu'il ne fait que du chainload.

?

Mais s'il ne peut pas booter linux, à quoi ça sert ?

En fait, il peut. En effet, les noyaux récents permettent d'inclure un mini-booloader, EFI stub, qui ne fait que charger son noyau. On peut donc avec gummiboot chainloader sur l'EFI stub, donc sur le fichier du noyau, et faire booter ce dernier. Cette méthode est notamment utilisée par archlinux pour ses installations UEFI.

Mais également, comme il peut chainload, il peut chainloader sur ... syslinux !

Vous avez donc le choix entre 3 configurations :

— Syslinux seul, si vous n'avez besoin de rien d'autre.
— Gummiboot seul, si vous avez besoin du chainload et que vous ne voulez pas vous prendre la tête à faire 2 installation et configuration successives.
— Gummiboot par-dessus syslinux, pour une installation robuste et complète.

Dans tous les cas, l'installation ne change pas, c'est uniquement la configuration de gummiboot qui change (si gummiboot il y a).

## 9.1. Installation

### 9.1.1. Syslinux

**Note :** je vais utiliser dans la suite le répertoire [syslinux]. Celui-ci correspond au `EFI/Boot` de la clé si vous voulez installer syslinux seul, ou `EFI/syslinux` si vous voulez faire un gummiboot par dessus syslinux. Comme pour un syslinux BIOS, on décompresse l'archive du projet, puis ensuite on copie quelques fichiers dans [syslinux] :

— `efi64/com32/elflink/ldlinux/ldlinux.e64`
— `efi64/com32/elflink/ldlinux/ldlinux.elf`
— `efi64/com32/menu/menu.c32`
— `efi64/com32/modules/poweroff.c32`
— `efi64/com32/modules/config.c32`
— `efi64/com32/modules/reboot.c32`
— `efi64/com32/chain/chain.c32`
— `efi64/com32/libutil/libutil.c32`
— `efi64/com32/lib/libcom32.c32`
— `efi64/com32/lib/libcom32.elf`

Et c'est tout !

### 9.1.2. Gummiboot

Je vous ai mis le binaire d'archlinux ici (en cas de problèmes pour le télécharger envoyez un mail ou un MP), il devrait marcher pour tout le monde. Vous le mettez dans `EFI/Boot`, avec comme nom `bootx64.efi`, et c'est fini.

*Contenu masqué*

## 9.2. Configuration

### 9.2.1. Syslinux

C'est la même que celle d'un syslinux BIOS, je vous renvoie donc à la partie précédente. Le seul changement est que le fameux fichier `syslinux.cfg` ne sera pas mis dans `boot/syslinux` mais dans [syslinux].

### 9.2.2. Gummiboot

#### 9.2.2.1. Commun

On crée un petit dossier `EFI/loader`, puis `EFI/loader/entries`.

#### 9.2.2.2. Gummiboot seul

Dans `loader/loader.conf` :

```
default rim
timeout 4
```

Dans `loader/entries/rim.conf` :

```
title RIM-Linux
linux      /boot/bzImage
initrd     /boot/RIM.cpio.gz
options    root=none rw
```

#### 9.2.2.3. Gummiboot + Syslinux

Dans `loader/loader.conf` :

```
default syslinux
timeout 4
```

Dans `loader/entries/syslinux.conf` :

```
title      Syslinux bootloader
efi        /EFI/syslinux/syslinux.efi
```

Retourner au texte.

## Contenu masqué n°11

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

# Shells
tty1::respawn:-/bin/ash
tty2::askfirst:-/bin/ash
tty3::respawn:/sbin/getty 38400 /dev/tty3 linux
tty4::respawn:/sbin/getty 38400 /dev/tty4 linux

# Restarting init ?
::restart:/sbin/init

# Before rebooting
::ctrlaltdel:/sbin/reboot
::shutdown:/etc/init.d/rc.shutdown
```

Retourner au texte.

## Contenu masqué n°12

```
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin
LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib
TERM=linux
HOME=/root
DISPLAY=:0
LANG=FR_fr
export LANG PATH LD_LIBRARY_PATH
umask 026
```

Retourner au texte.

## Contenu masqué n°13

Udev requiert qu'il y ait un `sysfs` dans `/sys`. Udev va servir à remplir `/dev` avec ce qui est présent sur le système. Il gère le coldplug et le hotplug. Depuis 2014 le code source d'udev a été fusionné avec celui de `systemd`, qui est un remplaçant d'un init de type UNIX tel que celui de busybox.

? Mais alors, c'est quoi le problème avec systemd ?

Systemd n'est pas spécialement adapté à notre usage. En effet, il n'est pas aussi simple de ne pas lui faire monter une partition sur `/` qu'avec celui de busybox, il n'est pas aussi léger, et *surtout* son intérêt principal est de paralléliser les tâches pour un boot plus rapide. Or, nous n'avons pas ce besoin, puisqu'il n'y a presque aucune tâche !

Personnellement, après avoir choisi dans le bootloader de booter ce tux, je boote sur un Acer Aspire One en 2s50, donc je pense que nous n'avons pas forcément un fort besoin de paralléliser les tâches. D'autant plus que systemd requiert d'utiliser dbus, ce qui fait que l'on va, avec ces 2 programmes, déjà alourdir nettement le système. Normalement, il est possible de compiler udev sans compiler tout systemd (les programmes sont séparés), mais je n'ai pas encore assez exploré cette possibilité, donc pour l'instant je n'en parle pas. Il est donc très fortement recommandé de passer par mdev. Retourner au texte.

## Contenu masqué n°14

Comme udev, il faut qu'il y ait un `sysfs` dans `/sys`, sauf qu'il n'utilise pas la même syntaxe, et surtout n'utilise qu'un seul fichier de configuration facultatif.

On le lance avec `mtab -s`. Retourner au texte.

## Contenu masqué n°15

```
1  #!/bin/sh
2
3  ROOTFS_IMG=RIM.cpio.gz
4
5  if mountpoint /home > /dev/null
6  then
7          echo "Saving current rootfs"
8          cp /home/boot/$ROOTFS_IMG /home/boot/$ROOTFS_IMG.bak
9          echo "Generating new rootfs ..."
10         find /* -print | grep -v boot | grep -v home | grep -v tmp
               | grep -v proc | grep -v sys | cpio -o -H newc | gzip
              -9 > /tmp/$ROOTFS_IMG
11         echo "Saving new rootfs"
12         mv /tmp/$ROOTFS_IMG /home/boot/$ROOTFS_IMG
13 else
14         echo "Usb-stick not mounted !"
15 fi
```

Il ressemble un peu au premier script pour générer l'archive cpio, mais avec quelques ajouts :

— Il faut stocker la nouvelle archive dans la clé, il faut donc qu'elle soit montée, ce qu'on vérifie avec `mountpoint /home`. On redirige la sortie de cette commande dans le vide cosmique parce qu'elle sort "c'est bien une partition", ce dont on n'a pas besoin puisque c'est un script, et donc on ne veut pas polluer la sortie à l'écran avec n'importe quoi.

— `grep` permet de ne prendre que les résultats comportant l'expression passée en paramètre. L'option `-v` permet d'inverser ce fonctionnement : grep filtre tout les résultats contenant l'expression en question en les enlevant.

Retourner au texte.

## Contenu masqué n°16

```
#!/bin/sh

# Recreate missing directories (ignored when re-creating initramfs)
/bin/mkdir /home
/bin/mkdir /boot
/bin/mkdir /tmp
/bin/mkdir /proc
/bin/mkdir /sys

# Mount pseudo file systems
mount -n -a

# Start asynchronous shell script
/etc/init.d/helper.rcs&

# === HOTPLUG ===

# Example with udev WARNING : may be outdated

#/sbin/udevd --daemon&
#/sbin/udevstart&
#echo "/sbin/udev" > /proc/sys/kernel/hotplug

# Example with mdev

#mdev -s
#echo "/sbin/mdev" > /proc/sys/kernel/hotplug

# ===============

# Syslog in circular buffer ( -C ) or not
/sbin/syslog -C

# Klogd
/sbin/klogd -c 2

# === KEYBOARD ===

# Example with busybox
#busybox loadkmap < /etc/fr.kmap
```

```
41
42 # ================
43
44 # Hostname
45 /bin/hostname rim-linux
46
47 # Loopback
48 /sbin/ifconfig lo up
49
50 # Clear screen
51 clear
```

Retourner au texte.

## Contenu masqué n°17

```
1  #!/bin/sh
2
3  # Checking if we need to mount usb-stick
4  if [ $ifmount = no ]
5  then
6  # Recreating user repositories
7          my_dir=$(cat /etc/passwd | cut -d : -f 1 | grep -v root)
8          for i in $my_dir
9          do
10                 mkdir /home/$i
11                 chown $i:$i /home/$i
12         done
13
14 else
15 # Waiting system full-boot to mount usb-stick
16         sleep 3
17         /bin/mount -U $home /home
18 fi
```

Retourner au texte.

## Contenu masqué n°18

```
1 #!/bin/sh
2 echo "Umounting all filesystems..."
3
4 if mountpoint /home > /dev/null
```

*Contenu masqué*

```
then
        /bin/umount -r /home
fi
/bin/umount -a -r
echo "Shutting off swap ..."
/sbin/swapoff -a
```

Retourner au texte.