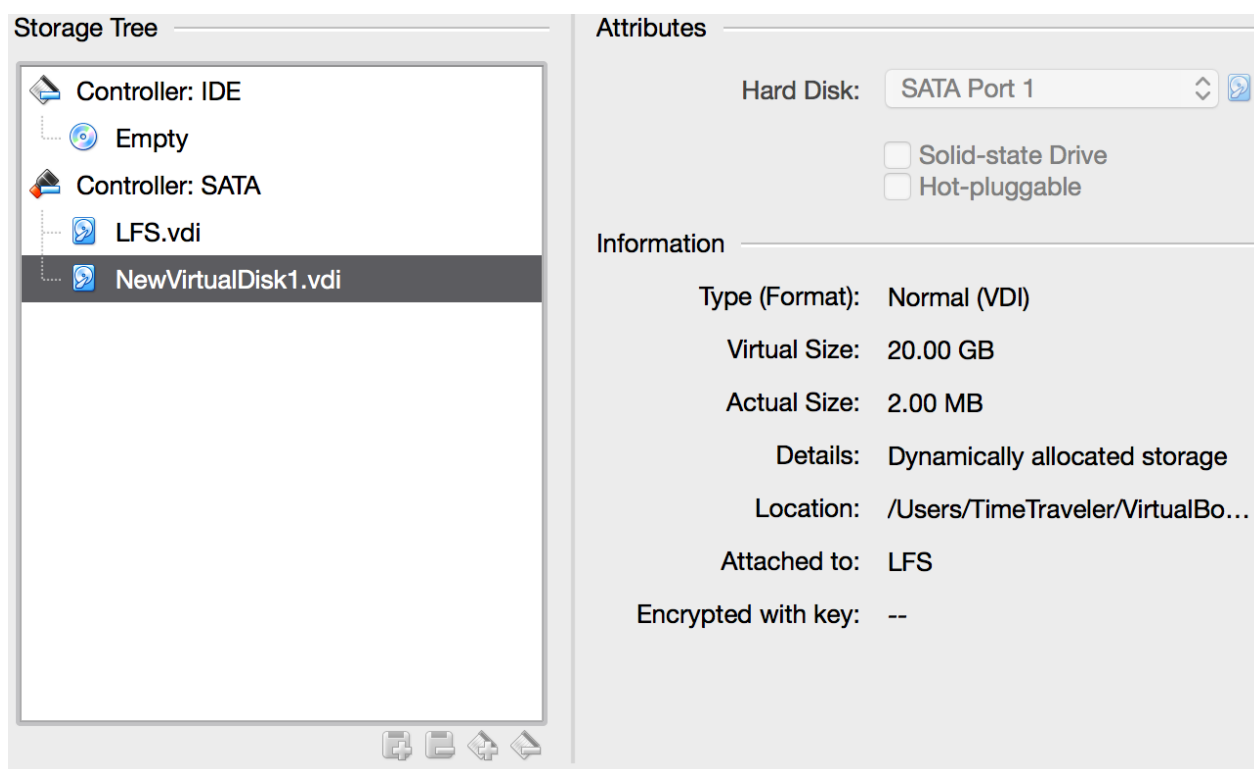


Procédure LFS Automatisé

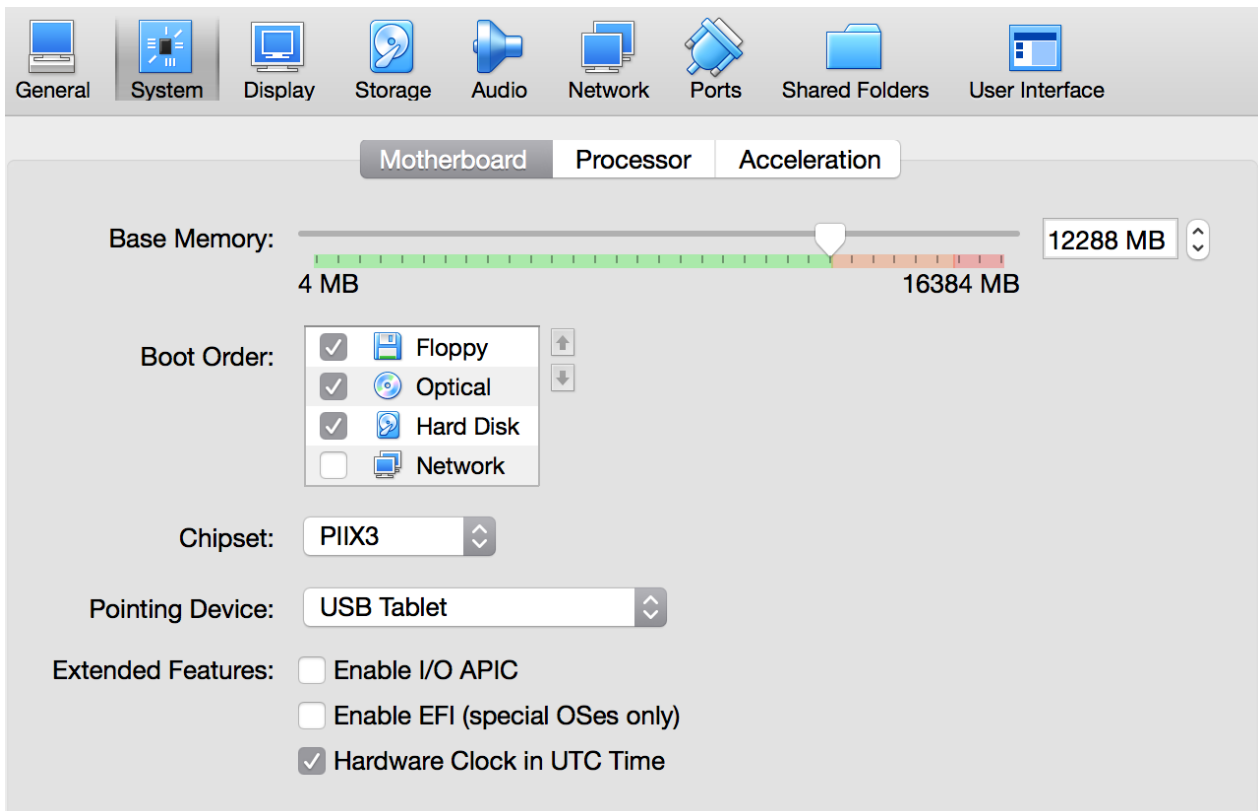
Auteur: Romain Claret

Machine Hôte

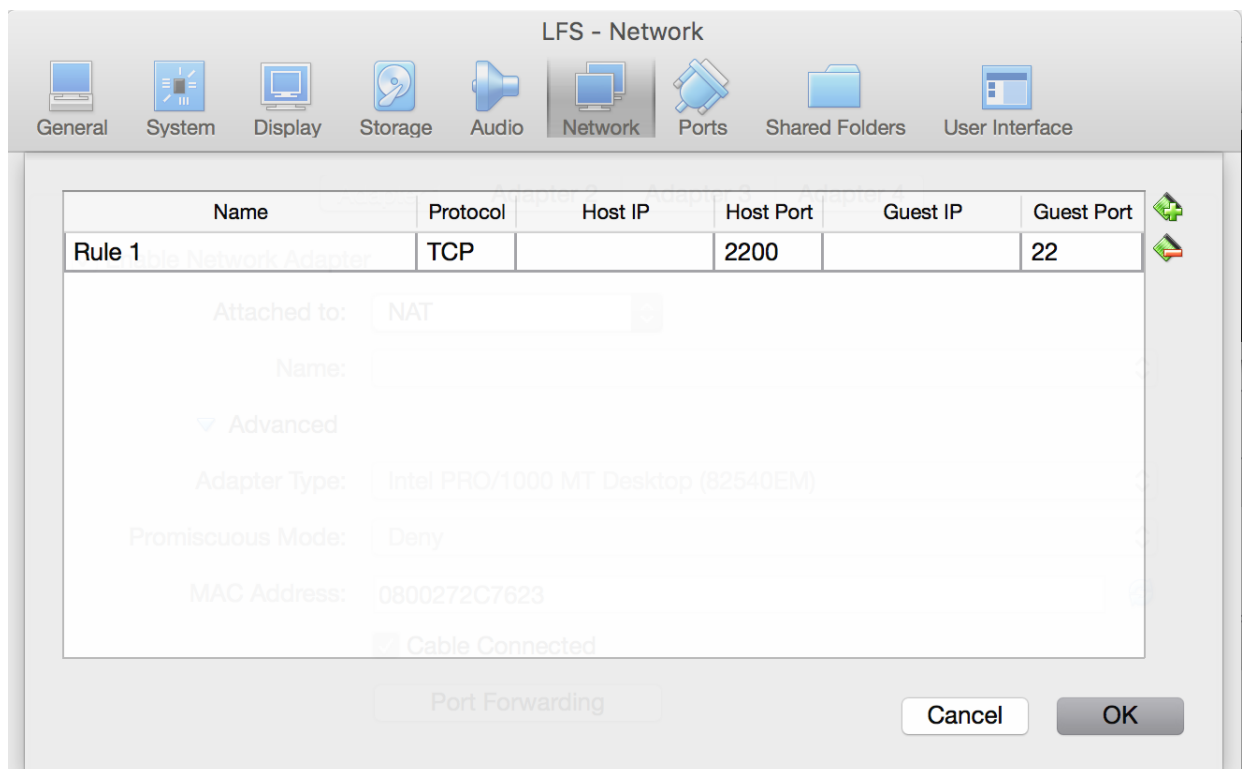
- Pour commencer nous avons avoir besoin d'une machine virtuelle dans virtualbox tournant avec **Debian 7.8 32-bit et 40 GB d'espace** (la ram et les coeurs peuvent être modifié on-the-fly plus tard). Le nom de l'image d'installation est: *debian-7.8.0-i386-xfce-CD-1.iso* trouvable ici au moment de la rédaction de ce rapport: <http://debian.nctu.edu.tw/debian-cd/7.8.0/i386/iso-cd/>
- Ajouter un Disk Dur supplémentaire avec **20GB** qui contiendra LFS. Concernant l'espace mémoire de ce disque, il est précisé dans la documentation qu'il faut au moins 4 GB pour la partition LFS elle-même, et il est recommandé 10 GB. Dans notre cas il a été pris le double.



- Conseil: Penser à faire des snapshot souvent!
- Nous allons également donner un peu de puissance à notre machine virtuelle. Ici nous mettrons à disposition 12GB (12GB * 1024 = 12'288 MB) de ram:



- La machine est configurée et contrôlée par SSH à l'aide du port forwarding:



```
The authenticity of host '[127.0.0.1]:2200 ([127.0.0.1]:2200)' can't be established.  
ECDSA key fingerprint is SHA256:LP8xhtI6qRuAbZMN9pdXt7thehDE4Kbbm1sCUy+Y4t8.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added '[127.0.0.1]:2200' (ECDSA) to the list of known hosts  
.  
[root@127.0.0.1's password:  
Linux debian 3.2.0-4-486 #1 Debian 3.2.73-2+deb7u2 i686  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Fri Jan 8 13:14:59 2016
```

Particularités:

- Création d'un LFS 32-bit sur un disque dur secondaire (bootable)
- Le processus de création LFS a été automatisé à l'aide de scripts Shell.
- Un répertoire github contenant les scripts a été créé. Celui-ci est disponible publiquement à l'adresse: <https://github.com/Rocla/lfs-7.8>
 - De plus pour un côté académique, les commits sont aussi présents pour mettre en avant le journal de travail, avec les problèmes et solutions rencontrés.
- La structure des fichiers est comme suit:
 - Chapitre.Section-Utilisateur_Info
 - Chapitre: du livre
 - Section: du chapitre
 - Utilisateur: root, l'utilisateur local (ici Debian), ou change root (chroot)
 - Info: partie optionnelle du fichier, contiennent des informations supplémentaires

Étape -1 : Installation de la machine (si besoin)

- Installer Debian 7.2 32-bit dans virtualbox avec les propriétés suivantes:
 - hostname: debian
 - root password: lfs
 - full name username: lfs
 - username: lfs
 - password: lfs

- Use entire disk and all files in one partition
- Software selection (espace pour sélectionner ou désélectionner):

```
[ ] Debian desktop environment
[ ] Web server
[ ] Print server
[ ] SQL database
[ ] DNS Server
[ ] File server
[ ] Mail server
[*] SSH server
[ ] Laptop
[*] Standard system utilities
```

- Installer le grub
- Penser à faire une snapshot une fois l'installation terminée.
- Mettre à jour l'os
 - Se connecter en root avec le password: lfs
 - `apt-get update`
 - `apt-get upgrade`

Étape 0 : Initialisation (Chapitre 0)

- Se connecter en root avec le password: lfs
- Installer les packages suivants: git
 - `apt-get install git build-essential`
- S'il y a une demande pour "Debian GNU/Linux 7.8.0 _Wheezy_ - Official i386 xfce-CD Binary-1 20150110-13:31" lors d'une installation, commenter cette ligne dans `sources.list`
 - `vi /etc/apt/sources.list`

```
# deb cdrom:[Debian GNU/Linux 7.8.0 _Wheezy_ - Official i386 xfce-CD Binary-1 20
150110-13:31]/ wheezy main

# deb cdrom:[Debian GNU/Linux 7.8.0 _Wheezy_ - Official i386 xfce-CD Binary-1 20
150110-13:31]/ wheezy main
```

- Cloner le répertoire git contenant les scripts d'installation à la racine du root:
 - `cd ~`
 - `git clone https://github.com/Rocla/lfs-7.8.git`

- `cd lfs-7.8`
- Exécuter l'étape 0. Cette étape vérifie la structure globale de la machine hôte:
 - `chmod +x 0.0-root_initial.sh`
 - `./0.0-root_initial.sh`
- Dans le cas d'une erreur fatale 3:
 - `ln -svf bash /bin/sh`
 - `./0.0-root_initial.sh`
- Installer les packages manquants. Dans le cas de notre distribution:
 - `apt-get install bison gawk g++`
 - `./0.0-root_initial.sh`
- Installer les libraires manquantes. Dans le cas de notre distribution:
 - Seulement les libraires: **libgmp.la**, **libmpfr.la** et **libmpc.la** ne sont pas trouvées. Il n'y a pas de problème selon la documentation LFS 7.8 si les trois manquent en même temps ou si aucun n'est manquant. Cependant il y a un problème s'il y en a un, ou deux, qui manquent.
- Penser à faire une snapshot une fois cette étape terminée.
- Notons qu'il est possible de totalement automatiser les étapes de 1 à 3. Cependant, il est recommandé d'utiliser ces étapes si elles ont déjà été exécutées une à une.
 - `./2.to.4-root_do-all-preparations.sh`

Étape 1 : Partitionnement (Chapitre 2)

- Cette étape crée une nouvelle partition qui contiendra notre LFS. Deux choix s'offrent à nous à présent: exécuter tous les scripts de l'étape 1 ou les exécuter un à un.
 - Dans le cas de l'exécution de tous les scripts:
 - `./2.all-root_make-new-partitions.sh`
 - Dans le deuxième cas, l'exécution un à un des scripts:
 - `./2.3-root_create-files-system-on-partitions.sh`
 - Une vérification est effectuée pour savoir si l'étape 1 doit être lancé depuis zéro. Par exemple si une mauvaise manipulation a été faite et l'on doit recommencer depuis le début alors qu'on était dans des étapes plus avancées.

- Partitionnement du Root et Swap du disque dur secondaire.
- `./2.4-root_set-lfs-variable.sh`
- Exportation de la variable `$LFS`. À noter que cette étape ne sert pas dans notre cas, car nous utilisons des scripts pour l'exécution de nos commandes et les variables sont stockés dans **script-root_commun-variables.sh**.
- `./2.5-root_mount-new-partitions.sh`
- Dans la documentation, il existe une procédure pour avoir de multiples partitionnements du `/usr`. Cependant, ça ne sera pas fait ici. KISS

Étape 2 : Packages and Patches (Chapitre 3)

- Lors de cette étape nous téléchargerons tous les packages et patches nécessaire au LFS.
- `./3.all-root_packages-patches.sh`
- Les téléchargements sont déposés dans un dossier d'archivage pour éviter de télécharger de multiple fois les mêmes fichiers (backup). Lors d'une nouvelle exécution du script, les archives seront utilisées au lieu de télécharger.
- Une fois les téléchargements terminés, les archives sont vérifiées avec le checksum officiel et sont copiées dans un répertoire de travail.

Étape 3 : Final Preparations (Chapitre 4)

- Lors de cette étape, nous allons créer l'utilisateur **lfs** et finaliser les préparations.
- Dans le cas de l'exécution de tous les scripts:
 - `./4.all-root_final-preparations.sh`
 - Attention à bien lire les instructions lorsque vous (humain) êtes requis.
- Dans le deuxième cas, l'exécution un à un des scripts:
 - `./4.2-root_create-lfs-tools-directory.sh`
 - Nous créons le dossier qui contiendra les outils pour l'utilisateur `lfs`
 - `./4.3-root_adding-lfs-user.sh`
 - Création du groupe pour le futur utilisateur `lfs` qui sera `lfs`
 - Création de l'utilisateur `lfs` et l'attribuons au groupe `lfs`
 - Donner les droits sur les outils à l'utilisateur `lfs`

- Copier les dossiers du répertoire git qui concerne l'utilisateur *lfs* dans son home

```
- su - lfs  
- cd setup-scripts  
- ./4.4-lfs_setting-up-environment.sh
```

- Création des variables d'environnement pour l'utilisateur *lfs*

- `.bash_profile`
- `.bashrc`

```
- source ~/.bash_profile
```

Étape 4 : Constructing a Temporary System (Chapitre 5)

- Durant cette étape nous allons mettre à disposition de notre utilisateur LFS les outils pour la construction notre OS.

- Petite note avant de commencer, nous utiliserons **tarball** pour la gestion des archives téléchargées.
- Dans le cas d'une restauration, c'est-à-dire si les étapes de ce chapitre ont déjà été effectuées une fois:

```
- ./5.0-root_restore-tools.sh
```

- Dans le cas de l'exécution de tous les scripts:

```
- ./5.all-lfs_construct-tools.sh
```

- Attention à bien lire les instructions lorsque vous (humain) êtes requis.
- S'il y a des erreurs, pensez à regarder le détail des étapes ci-dessous.

- Dans le dernier cas, l'exécution un à un des scripts:

```
- ./5.3-lfs_check-tools.sh
```

- Vérification de l'environnement et des liens symboliques
- Notons que la vérification de l'environnement est effectuée automatiquement pour chacun des scripts ci-dessous, dans le cas où ils ne seraient pas exécutés dans l'ordre des instructions officielles. Comme je le dis souvent "Rules are meant to be broken!"

```
- ./5.4-lfs_binutils-2.25.1-pass-1.sh
```

```
- ./5.5-lfs_gcc-5.2.0-pass-1.sh
```

- L'erreur: "no include path in which to search for stdc-predef.h" semble ne pas affecter l'installation du LFS: <https://wiki.debian.org/toolchain/BootstrapIssues>
- ./5.6-lfs_linux-4.2-api-headers.sh
- L'erreur: "*no include path in which to search for stdc-predef.h*" ne semble pas affecter l'installation du LFS: <https://wiki.debian.org/toolchain/BootstrapIssues>
- ./5.7-lfs_glibc-2.22.sh
- Les 2 erreurs: "*echo* " ne semble pas affecter l'installation du LFS
- ./5.8-lfs_libstdcpp-5.2.0.sh
- ./5.9-lfs_binutils-2.25.1-pass-2.sh
- ./5.10-lfs_gcc-5.2.0-pass-2.sh
- ./5.11-lfs_tcl-core-8.6.4.sh
- ./5.12-lfs_expect-5.45.sh
- L'erreur: "*rm: error while loading shared libraries: T?@: invalid mode for dlopen(): Invalid argument*" ne semble pas affecter l'installation du LFS.
- ./5.13-lfs_dejagnu-1.5.3.sh
- ./5.14-lfs_check-0.10.0.sh
- Les 3 erreurs: "*fprintf(stderr, \"%s:%d: Error in call to fwrite, wrote %ld instead of %d:\", __FILE__, __LINE__, written, to_write);*" ne semble pas affecter l'installation du LFS.
- ./5.15-lfs_ncurses-6.0.sh
- Les 4 erreurs: "*[/tools/lib/...;*" ne semble pas affecter l'installation du LFS.
- ./5.16-lfs_bash-4.3.30.sh
- Les 4 erreurs: "*the text of a system error...*" ne semble pas affecter l'installation du LFS.
- ./5.17-lfs_bzip2-1.0.6.sh
- ./5.18-lfs_coreutils-8.24.sh
- ./5.19-lfs_diffutils-3.3.sh
- ./5.20-lfs_file-5.24.sh
- ./5.21-lfs_findutils-4.4.2.sh
- ./5.22-lfs_gawk-4.1.3.sh

- ./5.23-lfs_gettext-0.19.5.1.sh
- ./5.24-lfs_grep-2.21.sh
- ./5.25-lfs_gzip-1.6.sh
- ./5.26-lfs_m4-1.4.17.sh
- ./5.27-lfs_make-4.1.sh
- ./5.28-lfs_patch-2.7.5.sh
- ./5.29-lfs_perl-5.22.0.sh
- ./5.30-lfs_sed-4.2.2.sh
- Les 4 erreurs: *“*** [utf8...” ne semble pas affecter l’installation du LFS.*
- ./5.31-lfs_tar-1.28.sh
- ./5.32-lfs_texinfo-6.0.sh
- ./5.33-lfs_util-linux-2.27.sh
- ./5.34-lfs_xz-5.2.1.sh
- ./5.35-lfs_stripping.sh

Étape 5 : Installing Basic System Software (Chapitre 6)

- Nous allons maintenant installer les programmes de pour notre LFS.

- Dans le cas de l’exécution de tous les scripts:

- ./6.all-part-1-root_installing-basic-system.sh
- ./6.all-part-2-chroot_installing-basic-system.sh

- Attention à bien lire les instructions lorsque vous (humain) êtes requis.
- S’il y a des erreurs, pensez à regarder le détail des étapes ci-dessous.

- Dans le dernier cas, l’exécution un à un des scripts:

- ./6.2-root_preparing-virtual-kernel.sh
- ./6.4-root_chroot-environment.sh

- Votre utilisateur va s’appeler: “I have no name!#” c’est normal, c’est due au fait que, etc/passwd n’est pas encore créé.

- ./6.5-chroot_creating-directories.sh

- ./6.6-chroot_essentials.sh

- Vous êtes maintenant “bash-4.3#”, c’est normal, nous sommes passés dans un nouveau terminal.

- ./6.7-chroot_api-headers.sh

- ./6.8-chroot_man-pages.sh

- ./6.9-chroot_glibc.sh

- Des erreurs sont détectées ici, mais n’affectent pas LFS.

- ./6.10-chroot_toolchain.sh

- Soyez attentif durant le sanity check. Vérifiez bien que les résultats soit les même que proposé durant la vérification selon votre architecture (32-bits/64-bits).

- ./6.11-chroot_zlib.sh

- ./6.12-chroot_file.sh

- ./6.13-chroot_binutils.sh

- ./6.14-chroot_gmp.sh

- Vérifiez bien que la valeur 188 soit présente quand la question est posée.

- ./6.15-chroot_mpfr.sh

- ./6.16-chroot_mpc.sh

- ./6.17-chroot_gcc.sh

- Cette partie est très très longue, ~4h sur ma machine.
- Soyez attentif durant le sanity check. Vérifiez bien que les résultats soient les mêmes que proposé durant la vérification selon votre architecture (32-bits/64-bits).
- Des erreurs sont détectées ici, mais n’affectent pas LFS.

- ./6.18-chroot_bzip2.sh

- ./6.19-chroot_pkg-config.sh

- ./6.20-chroot_ncurses.sh

- ./6.21-chroot_attr.sh

- ./6.22-chroot_acl.sh

- ./6.23-chroot_libcap.sh
- ./6.24-chroot_sed.sh
- ./6.25-chroot_shadow.sh
- ./6.26-chroot_psmisc.sh
- ./6.27-chroot_procps-ng.sh
- ./6.28-chroot_e2fsprogs.sh
 - Des erreurs sont détectées ici, mais n'affectent pas LFS.
- ./6.29-chroot_coreutils.sh
- ./6.30-chroot_iana-etc.sh
- ./6.31-chroot_m4.sh
 - Des erreurs sont détectées ici, mais n'affectent pas LFS.
- ./6.32-chroot_flex.sh
- ./6.33-chroot_bison.sh
 - Des erreurs sont détectées ici, mais n'affectent pas LFS.
- ./6.34-chroot_grep.sh
 - Des erreurs sont détectées ici, mais n'affectent pas LFS.
- ./6.35-chroot_readline.sh
- ./6.36-chroot_bash.sh
 - Lisez bien les instructions.
 - Des erreurs sont détectées ici, mais n'affectent pas LFS.
- ./6.37-chroot_bc.sh
- ./6.38-chroot_libtool.sh
 - Des erreurs sont détectées ici, mais n'affectent pas LFS.
- ./6.39-chroot_gdbm.sh
- ./6.40-chroot_expats.sh
 - Des erreurs sont détectées ici, mais n'affectent pas LFS.
- ./6.41-chroot_inetutils.sh
- ./6.42-chroot_perl.sh

- ./6.43-chroot_xml-parser.sh
- ./6.44-chroot_autoconf.sh
 - Des erreurs sont détectées ici, mais n'affectent pas LFS.
- ./6.45-chroot_automake.sh
 - Des erreurs sont détectées ici, mais n'affectent pas LFS.
- ./6.46-chroot_diffutils.sh
 - Des erreurs sont détectées ici, mais n'affectent pas LFS.
- ./6.47-chroot_gawk.sh
- ./6.48-chroot_findutils.sh
- ./6.49-chroot_gettext.sh
- ./6.50-chroot_intltool.sh
- ./6.51-chroot_gperf.sh
- ./6.52-chroot_groff.sh
- ./6.53-chroot_xz.sh
- ./6.54-chroot_grub.sh
- ./6.55-chroot_less.sh
- ./6.56-chroot_gzip.sh
- ./6.57-chroot_iproute2.sh
 - Des erreurs sont détectées ici, mais n'affectent pas LFS.
- ./6.58-chroot_kbd.sh
 - Des erreurs sont détectées ici, mais n'affectent pas LFS.
- ./6.59-chroot_kmod.sh
- ./6.60-chroot_libpipeline.sh
- ./6.61-chroot_make.sh
- ./6.62-chroot_patch.sh
- ./6.63-chroot_sysklogd.sh
- ./6.64-chroot_sysvinit.sh
- ./6.65-chroot_tar.sh

- Des erreurs sont détectées ici, mais n'affectent pas LFS.

```
- ./6.66-chroot_texinfo.sh
- ./6.67-chroot_eudev.sh
- ./6.68-chroot_util-linux.sh
```

- Des erreurs sont détectées ici, mais n'affectent pas LFS.

```
- ./6.69-chroot_man-db.sh
- ./6.70-chroot_vim.sh
```

- Les erreurs ici sont impressionnantes avec les couleurs, mais rien de bien méchant.

```
- ./6.72-chroot_stripping.sh
- ./6.73-chroot_cleaning-up.sh
```

- Suivez bien les instructions !
- Oui, il faut bien taper **exit** 3 fois de suite.
- La dernière commande est:
- ./6.73-chroot_cleaning-up.sh

Étape 6 : Installing Basic System Software (Chapitre 7)

- Ici nous finalisons l'installation de notre LFS.

- Dans le cas de l'exécution de tous les scripts:

```
- ./7.all-chroot_configuration_bootscripts.sh
```

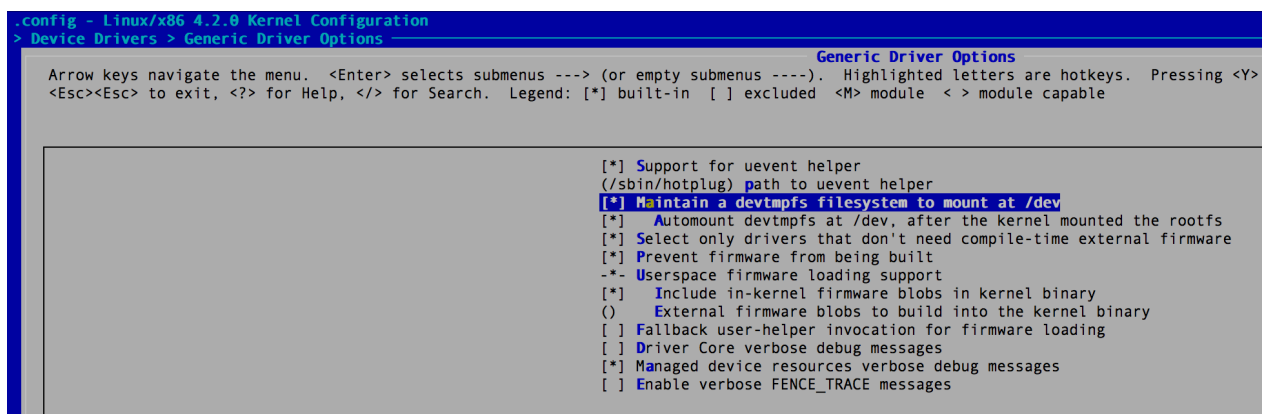
- Dans le dernier cas, l'exécution un à un des scripts (cela devrait être rapide et sans difficulté particulière):

```
- ./7.2-chroot_bootscripts.sh
- ./7.4-chroot_managing-devices.sh
- ./7.5-chroot_network.sh
- ./7.6-chroot_system-v.sh
- ./7.7-chroot_bash-shell.sh
- ./7.8-chroot_etc-inputrc.sh
```

```
- ./7.9-chroot_etc-shells.sh
```

Étape 7 : Installing Basic System Software (Chapitre 8)

- Nous rendons ici notre LFS bootable. Dans notre cas, nous configurons notre LFS pour booter depuis le Grub de l'hôte.
 - Dans le cas de l'exécution de tous les scripts:
 - ./8.all-chroot_make-bootable.sh
 - ./8.3-chroot_linux-42-part-2.sh
 - Dans le dernier cas, l'exécution un à un des scripts (cela devrait être rapide et sans difficulté particulière):
 - ./8.2-chroot_etc-fstab.sh
 - ./8.3-chroot_linux-42-part-1.sh
 - N'oubliez pas le **popd** à la fin des instructions !
 - Vous êtes libre ici de configurer votre LFS comme vous le souhaitez, mais en respectant une seule règle:



```
config - Linux/x86 4.2.0 Kernel Configuration
> Device Drivers > Generic Driver Options

Generic Driver Options
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y>
<Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

[*] Support for uevent helper
(/sbin/hotplug) path to uevent helper
[*] Maintain a devtmpfs filesystem to mount at /dev
[*] Automount devtmpfs at /dev, after the kernel mounted the rootfs
[*] Select only drivers that don't need compile-time external firmware
[*] Prevent firmware from being built
-* Userspace firmware loading support
[*] Include in-kernel firmware blobs in kernel binary
() External firmware blobs to build into the kernel binary
[ ] Fallback user-helper invocation for firmware loading
[ ] Driver Core verbose debug messages
[*] Managed device resources verbose debug messages
[ ] Enable verbose FENCE_TRACE messages
```

```
- ./8.3-chroot_linux-42-part-2.sh
```

Étape 8 : Installing Basic System Software (Chapitre 9)

- Nous y voilà enfin. Fiou... C'est l'heure de vérité ! Mettons à jour notre Grub et rebootons !
 - Il y a deux étapes séquentielles restantes (pour le côté jouissif de la chose 😊):

– ./9.1-chroot_the-end.sh

- Lisez bien les instructions !
- **update-grub** est important l'oubliez pas !

– ./9.3-chroot-reboot.sh

- La dernière pression de la touche *Enter* est enfin arrivée.
- Voilà c'est enfin derrière nous. *une larme coule le long de nos joues*

