LINUX: Construire un système de fichiers racine

Table of Contents

La création du système de fichiers racine implique la sélection des fichiers nécessaires au fonctionnement du système. Cette section décrit comment construire un système de fichiers racine compressé. Une option moins courante consiste à créer un système de fichiers non compressé sur une disquette directement montée en tant que racine.

Vue d'ensemble

Un système de fichiers racine doit contenir tout ce qui est nécessaire pour prendre en charge un système Linux complet. Pour pouvoir le faire, le disque doit inclure la configuration minimale requise pour un système Linux:

- La structure de base du système de fichiers,
- Ensemble minimal de répertoires: /dev, /proc, /bin, /etc, /lib, /usr, /tmp,
- Ensemble de base d'utilitaires: sh, ls, cp, mv, etc.,
- Ensemble minimal de fichiers de configuration: rc, inittab, fstab, etc.,
- Périphériques:/dev/hd , */dev/tty*, /dev/fd0, etc.
- Bibliothèque d'exécution pour fournir les fonctions de base utilisées par les utilitaires.

Bien sûr, n'importe quel système ne devient utile que lorsqu'on peut y exécuter quelque chose, et une disquette racine ne le devient généralement que lorsqu'on peut faire quelque chose comme:

- Vérifier un système de fichiers sur un autre lecteur. Par exemple, pour vérifier un système de fichiers racine sur un disque dur, il faut pouvoir démarrer Linux à partir d'un autre lecteur, comme avec un système de disquette racine. Ensuite, on doit pouvoir exécuter fsck sur le lecteur racine d'origine tant qu'il n'est pas monté.
- Restaurer tout ou partie d'un lecteur racine d'origine à partir d'une sauvegarde à l'aide d'utilitaires d'archivage et de compression tels que cpio, tar, gzip et ftape.

Cette section décrit comment construire un système de fichiers compressé, ainsi appelé, car il est compressé sur disque et, une fois démarré, non compressé sur un disque mémoire. Avec un système de fichiers compressé, on peut insérer de nombreux fichiers (environ six mégaoctets) sur une disquette 1440K standard. Comme le système de fichiers est beaucoup plus volumineux qu'une disquette, il ne peut pas être construit sur la disquette. Il faut le construire ailleurs, le compresser, puis le copier sur la disquette.

Création du système de fichiers

Pour créer un tel système de fichiers racine, on a besoin d'un périphérique suffisamment grand pour contenir tous les fichiers avant la compression.

Préparer le périphérique

Il faut un périphérique capable de contenir environ quatre mégaoctets. Il y a plusieurs choix:

- Utiliser un disque virtuel (DEVICE =/dev/ram0). Dans ce cas, la mémoire est utilisée pour simuler un lecteur de disque. Le disque mémoire doit être suffisamment grand pour contenir un système de fichiers de la taille appropriée. Si on utilise LILO, rechercher dans le fichier de configuration (/etc/lilo.conf) une ligne telle que RAMDISK = nnn, qui détermine la quantité maximale de RAM pouvant être allouée à un disque mémoire. La valeur par défaut est 4096K, ce qui devrait suffire. On ne doit pas essayer d'utiliser un tel disque mémoire sur une machine disposant de moins de 8 Mo de RAM. Vérifier de disposer d'un périphérique tel que /dev/ram0, /dev/ram ou /dev/ramdisk. Sinon, créer /dev/ram0 avec mknod (nombre majeur 1, mineur 0).
- Utiliser une partition de disque dur inutilisée suffisamment grande (plusieurs mégaoctets).
- Utiliser un périphérique loop, qui permet de traiter un fichier de disque comme un périphérique. À l'aide d'un périphérique loop, on peut créer un fichier de trois mégaoctets sur un disque dur et y créer le système de fichiers. Taper `man losetup` pour obtenir des instructions sur l'utilisation de périphériques loop. on peut le récupérer `losetup` dans les versions compatibles de mount et umount du paquet util-linux dans le répertoire ftp://ftp.win.tue.nl/pub/linux/utils/util-linux/.

Si on ne dispose pas de périphérique loop (/dev/loop0, /dev/loop1, etc.) sur le système hôte, il faut en créer un avec `mknod /dev/loop0 b 7 0`. Une fois installé les fichiers binaires spéciaux mount et umount, créer un fichier temporaire sur un disque dur ayant une capacité suffisante (par exemple, /tmp/fsfile):

```
dd if=/dev/zero of=/tmp/fsfile bs=1k count=nnn
```

Utiliser le nom de fichier à la place de DEVICE ci-dessous. Dans la commande de montage, il faudra

inclure l'option `-o loop` pour indiquer à mount d'utiliser un périphérique loop.

Après avoir choisi l'une de ces options, préparer DEVICE avec:

```
dd if=/dev/zero of=DEVICE bs=1k count=4096
```

Cette commande met à zéro le périphérique.

La remise à zéro du périphérique est essentielle car le système de fichiers sera compressé ultérieurement. Par conséquent, toutes les parties non utilisées doivent être remplies de zéros pour obtenir une compression maximale. Il faut y penser chaque fois que l'on déplace ou supprime des fichiers sur le système de fichiers. Le système de fichiers désaffectera correctement les blocs, mais il ne les supprimera plus. Si on effectue beaucoup de suppressions et de copies, le système de fichiers compressé risque de devenir beaucoup plus volumineux que nécessaire.

Créer le système de fichiers

Le noyau Linux reconnaît deux types de système de fichiers pour que les disques racine soient automatiquement copiés sur le disque mémoire. Ce sont minix et ext2, ext2 étant préféré. Lorsqu'on utilise ext2, il peut être utile d'utiliser l'option `-N` pour spécifier plus d'inodes que ceux par défaut; `-N 2000` est suggéré pour ne pas manquer d'inodes. On peut également enregistrer sur les inodes en supprimant de nombreux fichiers/dev inutiles. `mke2fs` créera par défaut 360 inodes sur une disquette de 1,44 Mo. 120 inodes suffisent amplement dans un système de sauvetage actuel La disquette racine, mais si on inclue tous les périphériques dans /dev, on risque de dépasser facilement 360. L'utilisation d'un système de fichiers racine compressé permet un système de fichiers plus volumineux, et donc plus d'inodes par défaut, mais il faudra peut-être réduire le nombre d'inodes.

Ainsi, la commande pour créer le système de fichier ressemblera à:

```
mke2fs -m 0 -N 2000 DEVICE
```

(Si on utilise un périphérique loop, le fichier disque utilisé doit être fourni à la place de ce périphérique.)

La commande mke2fs détecte automatiquement l'espace disponible et se configure en conséquence. Le paramètre `-m 0` l'empêche de réserver de l'espace pour le compte root et fournit donc davantage d'espace utilisable sur le disque.

Monter le périphérique:

```
mount -t ext2 DEVICE/mnt
```

(Il faut créer un point de montage /mnt s'il n'existe pas déjà.) Dans les sections restantes, tous les

noms de répertoire de destination sont supposés être relatifs à /mnt.

Remplir le système de fichiers

Voici un ensemble minimal de répertoires pour le système de fichiers racine [1]:

- **/dev** - Fichiers de périphérique nécessaires à l'exécution des E/S
- **/proc** - Stub de répertoire requis par le système de fichiers proc
- **/etc** - Fichiers de configuration système
- **/sbin** - Binaires critiques du système
- **/bin** - Fichiers binaires essentiels considérés comme faisant partie du système
- **/lib** - Bibliothèques partagées pour fournir un support au moment de l'exécution
- **/mnt** - Un point de montage pour la maintenance sur d'autres disques
- **/usr** - Utilitaires et applications supplémentaires

Trois de ces répertoires seront vides sur le système de fichiers racine, il suffit donc de les créer avec mkdir. Le répertoire **/proc** est fondamentalement un stub sous lequel le système de fichiers proc est placé. Les répertoires **/mnt** et **/usr** ne sont que des points de montage à utiliser après le démarrage du système boot **/root**. Par conséquent, il suffit de créer ces répertoires.

Les quatre répertoires restants sont décrits dans les sections suivantes.

/dev

Un répertoire /dev contenant un fichier spécial pour tous les périphériques à utiliser par le système est obligatoire pour tout système Linux. Le répertoire lui-même est un répertoire normal et peut être créé avec mkdir de la manière habituelle. Les fichiers spéciaux de périphériques doivent cependant être créés de manière spéciale à l'aide de la commande mknod.

Il existe cependant un raccourci: copiez les fichiers de périphériques à partir du répertoire de votre disque dur /dev existant. La seule condition requise est de copier les fichiers spéciaux du périphérique à l'aide de l'option -R. Cela copiera le répertoire sans tenter de copier le contenu des fichiers. Utiliser une lettre majuscule R. Par exemple:

```
cp -dpR /dev/fd[01]* /mnt/dev
cp -dpR /dev/tty[0-6] /mnt/dev
```

en supposant que la disquette est montée sur /mnt. Les commutateurs dp permettent de s'assurer que les liens symboliques sont copiés en tant que liens, plutôt que d'utiliser le fichier cible, et que les attributs de fichier d'origine sont préservés, préservant ainsi les informations de propriété.

On peut également utiliser ls -l pour afficher les numéros de périphérique majeur et mineur des périphériques souhaités puis les créer sur la disquette à l'aide de mknod.

Quels que soient les fichiers de périphériques créés, vérifier que tous les périphériques spéciaux dont on aura besoin ont été placés sur la disquette de secours. Par exemple, ftape utilise des unités de bande. Il faut donc toutes les copier pour pouvoir accéder à votre lecteur de disquette à partir du disque d’amorçage.

Un inode est requis pour chaque fichier spécial de périphérique et les inodes peuvent parfois être une ressource rare, en particulier sur les systèmes de fichiers sur disquette. Il est préférable de sélectionner les fichiers de périphérique à inclure. Par exemple, si on n'a pas de disques SCSI, on peut ignorer `/dev/sd*` en toute sécurité; Si on n'a pas l'intention d'utiliser les ports série, vous pouvez ignorer `/dev/ttyS*`.

Si, lors de la construction du système de fichiers racine, on a l'erreur `Il ne reste plus d'espace sur le périphérique` mais une commande df indique que l'espace est toujours disponible, on est probablement à court d'inodes. Un `df -i` affichera l'utilisation de l'inode.

Inclure les fichiers suivants de ce répertoire: console, kmem, mem, null, ram0 et tty1.

/etc

Le répertoire /etc contient les fichiers de configuration. Ce qu'il devrait contenir dépend des programmes que vous avez l'intention de lancer. Sur la plupart des systèmes, ceux-ci peuvent être divisés en trois groupes:

- Requis en tout temps, par exemple rc, fstab, passwd.
- Peut être nécessaire, mais personne n'est trop sûr.
- Junk qui s'est glissé dans.

Les fichiers qui ne sont pas essentiels peuvent généralement être identifiés avec la commande:

```
ls -ltru
```

Cette liste répertorie les fichiers en ordre de date de dernier accès inversé. Ainsi, si aucun fichier n’est utilisé, on peut les omettre d’une disquette racine, le nombre de fichiers de configuration peut ainsi être réduit à trois jeux de fichiers:

- Ceux que l'on doit configurer pour un système de démarrage/racine:
  - **rc.d/** - scripts de changement de niveau de démarrage et d'exécution du système
  - **fstab** - liste des systèmes de fichiers à monter
  - **inittab** - paramètres du processus init, le premier processus démarré au démarrage.
  - **gettydefs** - paramètres du processus init, le premier processus démarré au démarrage.
- Ceux que je devrais ranger pour un système de démarrage/racine:
  - **passwd** - Liste critique de utilisateurs, répertoires personnels, etc. **Important**: passwd doit contenir au moins root. Si on veut que d'autres utilisateurs se connectent, il faut que leurs répertoires personnels et leurs shells existent.
  - **group** - groupes d'utilisateurs.
  - **shadow** - mots de passe des utilisateurs. Si la sécurité est importante, **passwd** et **shadow** doivent être élagués pour éviter de copier les mots de passe des utilisateurs hors du système et pour éviter les connexions non souhaitées lors du démarrage à partir

d'une disquette.
    - **termcap** - la base de données des capacités du terminal, est généralement de plusieurs centaines de kilo-octets. La version de la disquette de démarrage/racine doit être réduite pour ne contenir que le (s) terminal(s) utilisé(s), ce qui correspond généralement à l'entrée linux ou linux-console.
- Le reste. Ils travaillent en ce moment, alors je les laisse tranquilles.

Pour le reste, copier simplement tous les fichiers texte du répertoire /etc, ainsi que tous les exécutables du répertoire /etc. Il suffira probablement de ne copier que ces fichiers. La seule méthode sûre est de commencer avec inittab et de déterminer ce qui est requis.

La plupart des systèmes utilisent maintenant un répertoire /etc/rc.d/ contenant des scripts de shell pour différents niveaux d'exécution. Le minimum est un seul script rc, mais il peut être plus simple de copier simplement inittab et le répertoire /etc/rc.d du système existant et d'élaguer les scripts shell dans le répertoire rc.d afin de supprimer les traitements non pertinents sur une disquette.

Il faudra Configurer les fichiers

le fichier rc

doit contenir:

```
#!/bin/sh
/bin/mount -av
/bin/hostname Kangaroo
```

Important: Il doit être exécutable

Le fichier fstab

doit contenir au moins:

```
/dev/ram0       /               ext2    defaults
/dev/fd0        /               ext2    defaults
/proc           /proc           proc    defaults
```

On peut y copier des entrées du fstab existant, mais il est préférable de ne pas monter automatiquement aucune des partitions de disque dur; utilise le mot clé noauto avec ceux-ci. Le disque dur peut être endommagé ou mort lorsque le disque de démarrage est utilisé.

Le fichier inittab

doit être modifié pour que sa ligne sysinit exécute rc ou le script de démarrage de base utilisé. De plus, pour s'assurer que les utilisateurs de ports série ne peuvent pas se connecter, mettre en commentaire toutes les entrées de getty qui incluent un périphérique ttys ou ttyS à la fin de la ligne.

Laisser les ports tty pour pouvoir se connecter à la console.

Un fichier inittab minimal ressemble à ceci:

```
id:2:initdefault:
si::sysinit:/etc/rc
1:2345:respawn:/sbin/getty 9600 tty1
2:23:respawn:/sbin/getty 9600 tty2
```

Le fichier inittab définit ce que le système exécutera dans divers états, notamment le démarrage, le passage en mode multi-utilisateur, etc. Vérifier attentivement les noms de fichiers mentionnés dans inittab; si init ne trouve pas le programme mentionné, le disque de démarrage se bloque et on risque même de ne pas recevoir de message d'erreur.

Certains programmes ne peuvent pas être déplacés ailleurs car d'autres ont codé en dur leurs emplacements. Par exemple, /etc/shutdown est codé en dur dans /etc/reboot. Si on déplace le redémarrage vers /bin/reboot, puis émet une commande d'arrêt, le processus échouera car il ne trouvera pas le fichier de redémarrage.

/bin et/sbin

Le répertoire /bin est un endroit pratique pour les utilitaires supplémentaires dont vous avez besoin pour effectuer des opérations de base, tels que ls, mv, cat et dd. Voir l'Annexe C pour un exemple de liste de fichiers contenus dans les répertoires /bin et /sbin. Il n'inclut aucun des utilitaires requis pour restaurer à partir d'une sauvegarde, tels que cpio, tar et gzip. C'est parce que je les place sur une disquette d'utilitaires distincte pour économiser de l'espace sur la disquette de démarrage/racine. Une fois la disquette d'amorçage/racine amorcée, elle est copiée sur le disque mémoire, laissant ainsi le lecteur de disquette libre pour monter une autre disquette, la disquette utilitaire. Je monte généralement ceci en tant que /usr.

La création d'une disquette d'utilitaire est décrite ci-dessous à la section 9.2. Il est probablement souhaitable de conserver une copie de la même version des utilitaires de sauvegarde utilisés pour écrire les sauvegardes afin de ne pas perdre de temps à essayer d'installer des versions qui ne peuvent pas lire vos bandes de sauvegarde.

Il faut inclure les programmes suivants: init, getty ou équivalent, login, mount, un shell capable d'exécuter vos scripts rc, un lien entre sh et le shell.

Les bibliothèques

Cas général

Dans /lib, on place les bibliothèques et les chargeurs partagés nécessaires. Si les bibliothèques nécessaires ne se trouvent pas dans votre répertoire/lib, le système ne pourra pas démarrer.

Presque tous les programmes nécessitent au moins la bibliothèque libc, libc.so.N, où N est le numéro de version actuel. Vérifier le contenu du répertoire /lib. Le fichier libc.so.N est généralement un lien symbolique vers un nom de fichier avec un numéro de version complet:

```
% ls -l /lib/libc*
-rwxr-xr-x   1 root     root      4016683 Apr 16 18:48 libc-2.1.1.so*
lrwxrwxrwx   1 root     root           13 Apr 10 12:25 libc.so.6 ->
libc-2.1.1.so*
```

Dans ce cas, on aura besoin de libc-2.1.1.so. Pour trouver d'autres bibliothèques, il faut vérifier les dépendances de tous les fichiers binaires que l'on veut souhaite inclure avec ldd. Par exemple:

```
% ldd /sbin/mke2fs
libext2fs.so.2 => /lib/libext2fs.so.2 (0x40014000)
libcom_err.so.2 => /lib/libcom_err.so.2 (0x40026000)
libuuid.so.1 => /lib/libuuid.so.1 (0x40028000)
libc.so.6 => /lib/libc.so.6 (0x4002c000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Chaque fichier à droite est requis. Le fichier peut être un lien symbolique.

Certaines bibliothèques sont assez grandes et ne tiendront pas facilement sur le système de fichiers racine. Par exemple, le fichier libc.so indiqué ci-dessus fait environ 4 mo Il faudra probablement supprimer les bibliothèques lorsqu'on les copie sur le système de fichiers racine.

Dans /lib, il faut également inclure un chargeur pour les bibliothèques. Le chargeur sera soit ld.so (pour les bibliothèques A.OUT, qui ne sont plus courantes), soit ld-linux.so (pour les bibliothèques ELF). Les versions les plus récentes de ldd indiquent exactement quel chargeur est nécessaire, comme dans l'exemple ci-dessus, mais les versions plus anciennes risquent de ne pas l'être.Pour être sûr de ce dont on a besoin, exécuter la commande file sur la bibliothèque. Par exemple:

```
% file /lib/libc.so.4.7.2 /lib/libc.so.5.4.33 /lib/libc-2.1.1.so
/lib/libc.so.4.7.2: Linux/i386 demand-paged executable (QMAGIC), stripped
/lib/libc.so.5.4.33: ELF 32-bit LSB shared object, Intel 80386, version 1,
stripped
/lib/libc-2.1.1.so: ELF 32-bit LSB shared object, Intel 80386, version 1,
not stripped
```

QMAGIC indique que 4.7.2 concerne les bibliothèques A.OUT et ELF indique que 5.4.33 et 2.1.1

correspondent à ELF.

Copier le ou les chargeurs spécifiques dont on a besoin dans le système de fichiers racine. Les bibliothèques et les chargeurs doivent être soigneusement comparés aux fichiers binaires inclus. Si le noyau ne peut pas charger une bibliothèque nécessaire, il peut se bloquer sans message d'erreur.

Bibliothèques pour PAM et NSS

Le système peut nécessiter des bibliothèques chargées dynamiquement, non visibles par ldd. Si on ne le fournit pas, on aura peut-être des difficultés à se connecter ou à utiliser le disque de démarrage.

PAM (modules d'authentification enfichables)

Si le système utilise PAM (Pluggable Authentication Modules), il faut le configurer sur le disque de démarrage. En résumé, PAM est une méthode modulaire sophistiquée d'authentification des utilisateurs et de contrôle de leur accès aux services. Un moyen simple de déterminer si le système utilise PAM est d'exécuter ldd sur l'exécutable de connexion; si la sortie inclut libpam.so, il faut embarqué PAM.

Heureusement, la sécurité ne pose généralement aucun problème pour les disques d'amorçage, car toute personne ayant un accès physique à une machine peut généralement faire ce qu'elle veut. Par conséquent, on peut désactiver efficacement PAM en créant un simple fichier /etc/pam.conf dans le système de fichiers racine, qui ressemble à ceci:

```
OTHER   auth       optional     /lib/security/pam_permit.so
OTHER   account    optional     /lib/security/pam_permit.so
OTHER   password   optional     /lib/security/pam_permit.so
OTHER   session    optional     /lib/security/pam_permit.so
```

Copier également le fichier /lib/security/pam_permit.so dans le système de fichiers racine. Cette bibliothèque n'est qu'environ 8K, elle impose donc un surcoût minimal.

Cette configuration permet à quiconque d'accéder aux fichiers et aux services de l'ordinateur. Pour des raisons de sécurité on veut protéger le disque de démarrage, il faudra copier tout ou partie de la configuration PAM du disque dur dans le système de fichiers racine et toutes les bibliothèques nécessaires dans/lib/security sur le système de fichiers racine.

Il faut également inclure /lib/libpam.so sur votre disque de démarrage.

NSS (Name Service Switch)

Si on utilise glibc (autrement dit libc6), il faut prendre des dispositions pour les services de noms ou on ne pourra pas se connecter. Le fichier `/etc/nsswitch.conf` contrôle les recherches dans la base de données pour différents services. Si on ne prévoit pas d'accéder aux services du réseau (par exemple, les recherches DNS ou NIS), il suffit de préparer un fichier nsswitch.conf simple, ressemblant à ceci:

```
passwd:     files
shadow:     files
group:      files
hosts:      files
services:   files
networks:   files
protocols:  files
rpc:        files
ethers:     files
netmasks:   files
bootparams: files
automount:  files
aliases:    files
netgroup:   files
publickey:  files
```

Cela spécifie que chaque service doit être fourni uniquement par des fichiers locaux. Il faut également inclure `/lib/libnss_files.so.X`, où X est 1 pour glibc 2.0 et 2 pour glibc 2.1. Cette bibliothèque sera chargée dynamiquement pour gérer les recherches de fichiers.

Si on envisage d'accéder au réseau à partir du disque d'amorçage, on peutcréer un n plus élaboré fichier sswitch.conf. CIl faut inclure un fichier /lib/libnss_service.so.1 pour chaque service spécifié.

Modules

Si on a un noyau modulaire, il faut déterminer quels modules on veutcharger à partir de votre disque de démarrage après le démarrage. Par exemple on peut inclure des modules ftape et zftape si on utilise des bandes de sauvegarde, des modules pour les périphériques SCSI si on en a et éventuellement des modules pour le support PPP ou SLIP si on veut accéder au réseau en cas d'urgence.

Ces modules peuvent être placés dans `/lib/modules`. Il faut également inclure insmod, rmmod et lsmod. Selon que l'on veut charger des modules automatiquement ou non, on peut également inclure modprobe, depmod et swapout. Pour utiliser kerneld, l'inclure dans /etc/conf.modules.

Cependant, le principal avantage de l'utilisation de modules est que l'on peut déplacer des modules non critiques sur un disque utilitaire et les charger en cas de besoin, en utilisant moins d'espace sur le disque racine. Si on doit faire face à de nombreux périphériques différents, cette approche est préférable à la construction d'un énorme noyau avec de nombreux pilotes intégrés.

Pour amorcer un système de fichiers ext2 compressé, on doit disposer de la prise en charge de ramdisk et ext2. Ils ne peuvent pas être fournis sous forme de modules.

Derniers détails

Certains programmes système, tels que login, se plaignent si le fichier `/var/run/utmp` et le répertoire `/var/log` n'existent pas. Alors:

```
mkdir -p /mnt/var/{log,run}
touch /mnt/var/run/utmp
```

Enfin, après avoir configuré toutes les bibliothèques dont vous avez besoin, exécutez ldconfig pour refondre `/etc/ld.so.cache` sur le système de fichiers racine. Le cache indique au chargeur où trouver les bibliothèques. On peut le faire avec:

```
ldconfig -r /mnt
```

Envelopper

Lorsqu'on a fini de construire le système de fichiers racine, le démonter, le copier dans un fichier et le compresser:

```
umount /mnt
dd if=DEVICE bs=1k | gzip -v9 > rootfs.gz
```

Lorsque cela sera terminé, on aura un fichier `rootfs.gz`. Ceci est le système de fichiers racine compressé. Vérifier sa taille pour s'assurer qu'il tienne sur une disquette; sinon, il faut revenir en arrière et supprimer certains fichiers.