Construire un mini-ubuntu basé sur debootstrap et busybox

Table of Contents

L'utilisation de l'outil debootstrap + busybox pour créer un système aussi petit présente les avantages suivants:

- Le petit ubuntu généré par debootstrap facilite l'installation d'outils supplémentaires à l'aide d'apt
- Peut copier directement le module de pilote sur une petite image
- Les scripts personnalisés sont très simples et faciles

L'ensemble du petit système représente environ 100 M sans installer de logiciel supplémentaire ni de modules de noyau, et peut être ajouté à busybox et réduit à 40-50 M (y compris la bibliothèque de base complète). Après avoir installé python3 (full python3), on peut le réduire à environ 110 M.

Initramfs est un système de fichiers complet d'un système Linux (supprime le noyau, supprime les logiciels inutilisés), puis le développe en mémoire et exécute le programme (/init/sbin/init).

Création du système de base

Debootstrap

Utiliser debootstrap pour générer un système minibase ubuntu contenant l'ensemble du système de fichiers de base et l'outil apt (sans noyau):

```
sudo debootstrap --variant=minbase trusty mini \
```

```
    http://mirrors.aliyun.com/ubuntu/
```

On peut utiliser n'importe quel autre miroir.

utiliser chroot en liaison avec dpkg pour afficher les logiciels installés:

```
sudo chroot mini dpkg -l
```

ou

```
sudo chroot mini/bin/bash
```

Afin de permettre son démarrage en tant qu'initramfs du noyau, il faut placer un fichier **init** dans le répertoire racine, écrire le processus de démarrage dans le fichier **init** (montage du système de fichiers, exécuter /sbin/init, etc.). Ce qui suit est copié à partir d'une partie de `/usr/share/initramfs-tools/init`, car elle ne monte pas la partition racine sur le disque dur, ce qui réduit considérablement le code.

```
cat << EOF | sudo tee mini/init
#!/bin/sh

[ -d /dev ] || mkdir -m 0755 /dev
[ -d /root ] || mkdir -m 0700 /root
[ -d /sys ] || mkdir /sys
[ -d /proc ] || mkdir /proc
[ -d /tmp ] || mkdir /tmp
mkdir -p /var/lock
mount -t sysfs -o nodev,noexec,nosuid sysfs /sys
mount -t proc -o nodev,noexec,nosuid proc /proc
# Certaines choses ne fonctionnent pas correctement sans /etc/mtab.
ln -sf /proc/mounts /etc/mtab

grep -q '\<quiet\>' /proc/cmdline || echo "Loading, please wait..."

# Cela ne devient /dev que sur le système de fichiers réel si les scripts de
udev
# sont utilisés; ce qu'ils seront, mais il est utile de préciser
if ! mount -t devtmpfs -o mode=0755 udev /dev; then
        echo "W: devtmpfs not available, falling back to tmpfs for /dev"
        mount -t tmpfs -o mode=0755 udev /dev
        [ -e /dev/console ] || mknod -m 0600 /dev/console c 5 1
        [ -e /dev/null ] || mknod /dev/null c 1 3
fi
mkdir /dev/pts
mount -t devpts -o noexec,nosuid,gid=5,mode=0620 devpts /dev/pts || true
```

```
mount -t tmpfs -o "noexec,nosuid,size=10%,mode=0755" tmpfs /run
mkdir /run/initramfs
# lien symbolique de compatibilité pour les emplacements pre-oneiric
ln -s /run/initramfs /dev/.initramfs

# Set modprobe env
export MODPROBE_OPTIONS="-qb"

# mdadm a besoin que le nom d'hôte soit défini, ceci avant que les règles
udev soient appelées!
if [ -f "/etc/hostname" ]; then
        /bin/hostname -b -F /etc/hostname 2>&1 1>/dev/null
fi

exec /sbin/init
EOF

Sudo chmod + x mini/init
```

Récupération du Noyau

Lorsque GRUB charge le noyau et **initramfs**, le noyau étend **initramfs** en mémoire, puis exécute **init** dans son répertoire racine, qui est le script ci-dessus. Le script ci-dessus effectuera le travail de montage, préparera la structure de répertoires, puis exécutera **/sbin/init** pour convertir le processus d'initialisation d'ubuntu (system-v init, upstart, systemd, créer automatiquement des fichiers de périphérique avec udev, etc.).

Modifier la configuration de lancement d'ubuntu pour une connexion automatique :

```
for i in $(find mini/etc/init -type f -name "tty*"); do
    sed -e "s|/sbin/getty -8|/sbin/getty --autologin root -8|" -i $i
done
```

Le module du noyau et le pilote matériel ne sont pas inclus dans la minibase. Il faut cpier le module à partir du noyau ou de la machine. Par exemple, télécharger une image linux via apt-get et extraire le module du noyau.

```
sudo apt-get download linux-image-$(uname -r)
sudo rm -rf linux-image-$(uname -r)
sudo dpkg -x $(find . -type f -name "linux-image-$(uname -r)*.deb" | head -n
1) linux-image-$(uname -r)
sudo cp -af linux-image-$(uname -r)/lib mini/
```

Puis générer les dépendances de modules :

```
sudo chroot mini depmod
```

Nettoyer :

```
sudo chroot mini apt-get clean
```

Réduction du système

Le petit système généré ci-dessus est fondamentalement disponible, mais pour continuer à réduire la taille nécessaire pour une exécution sur une machine avec moins de mémoire, on peut encore continuer à couper. L'étape la plus importante consiste à utiliser busybox pour gérer la plus grande partie du travail. Busybox comprend même le chargement et l'échange à chaud pilotés par l'appareil

Installation de Busybox

```
sudo chroot mini apt-get -y --no-install-recommends install \
    busybox-static
```

Définir les variables **apps** et **extra_apps** pour contenir les commandes prises en charge par busybox, et définir les fonctions pour remplacer les commandes d'origine par busybox :

```
applets=$(mini/bin/busybox --list)
apps=
for i in $applets; do
    apps="$apps $(sudo chroot mini which $i)"
done
extra_apps=
for i in $applets; do
    if ! sudo chroot mini which $i > /dev/null; then
        extra_apps="$extra_apps /bin/$i "
    fi
done

function fix_missing() {
    for i in $apps $extra_apps; do
        if ! test -f mini/$i; then
            sudo ln -sf /bin/busybox mini/$i
        fi
    done
}
```

Nettoyage

Nettoyer les paquets pouvant être supprimés directement:

```
sudo chroot mini apt-get -y --force-yes purge adduser busybox-initramfs \
    cpio ifupdown initramfs-tools initscripts initramfs-tools-bin \
    iproute2 locales mountall makedev plymouth procps upstart libprocps3 \
    libcgmanager0 libusb-1.0-0 usbutils libdbus-1-3 libnih-dbus1 libdrm2 \
    libjson-c2 libjson0 kmod libkmod2 module-init-tools file libmagic1 \
    libnih1 libplymouth2 pciutils libudev1 udev
fix_missing
```

Forcer la suppression des paquets indésirables:

```
sudo chroot mini /bin/sh -c 'echo "Yes, do as I say!" |\
    apt-get -y --force-yes purge diffutils findutils hostname'
fix_missing

sudo chroot mini /bin/sh -c 'echo "Yes, do as I say!" |\
    apt-get -y --force-yes purge -y --force-yes login libmount1 mount\
        grep gzip sed mount'
fix_missing

sudo chroot mini /bin/sh -c 'echo "Yes, do as I say!" |\
    apt-get -y --force-yes purge -y --force-yes sysvinit-utils lsb-base\
        e2fsprogs e2fslibs bsdutils libblkid1 libuuid1 passwd tzdata
insserv'
fix_missing

sudo chroot mini /bin/sh -c 'echo "Yes, do as I say!" |\
    apt-get -y --force-yes purge ncurses-base ncurses-bin'
sudo chroot mini /bin/sh -c 'echo "Yes, do as I say!" |\
    apt-get purge coreutils'
fix_missing
```

On peut continuer à nettoyer en supprimant des bibliothèques inutiles.

Nettoyer le cache d'apt ::

```
sudo chroot mini apt-get clean
sudo rm -rf mini/usr/share/locale/*
sudo rm -rf mini/usr/share/man/*
sudo rm -rf mini/usr/share/doc/*
sudo rm -rf mini/var/log/*
sudo rm -rf mini/var/lib/apt/lists/*
sudo rm -rf mini/var/cache/*
sudo rm -rf mini/etc/rc*
```

Configuration de l'init

Comme udev, system-v init, upstart, etc. ont été supprimés au cours du processus de nettoyage, un nouveau script init prenant en charge busybox est requis. Sous Ubuntu, l'udev original lui-même prend en charge le chargement initial du pilote et du périphérique, mais **mdev** n'est appellé que lorsqu'il est hotplug il faut inspecter le répertoire `/sys/devices` pour charger le module de pilote:

```
cat > mini/init << EOF
#!/bin/sh
mount -t proc -o nodev,noexec,nosuid proc /proc
mount -t sysfs -o nodev,noexec,nosuid sysfs /sys
ln -sf /proc/mounts /etc/mtab
mount -t tmpfs -o size=64k,mode=0755 tmpfs /dev
[ -e /dev/console ] || mknod -m 0600 /dev/console c 5 1
[ -e /dev/null ] || mknod /dev/null c 1 3
mkdir -p /dev/pts
mount -t devpts -o noexec,nosuid,gid=5,mode=0620 devpts /dev/pts
mount -t tmpfs -o "noexec,nosuid,size=10%,mode=0755" tmpfs /run
echo "Loading modules..."
# hotplug with mdev
echo /bin/mdev > /proc/sys/kernel/hotplug
mdev -s
# coldplug: load devices supporting modules
find /sys/devices -name modalias | xargs -r cat | xargs -r modprobe -qa
# extra modules
[ -f /etc/modules ] && cat /etc/modules | grep -v "^[[:blank:]]#" |\
    xargs -r modprobe -qa
exec /sbin/init
EOF

chmod +x mini/init
```

Créer des liens sympoblyque pour utiliser l'init de busybox

```
ln -sf /bin/busybox mini/bin/sh
ln -sf /bin/busybox mini/sbin/init
```

L'init de busybox nécessite un fichier inittab pour décrire le processus d'initialisation:

```
cat > mini/etc/inittab << EOF
::sysinit:/etc/init.d/rcS
::ctrlaltdel:/sbin/reboot
::shutdown:/sbin/swapoff -a
::shutdown:/bin/umount -a -r
::restart:/sbin/init
tty1::respawn:/bin/sh
tty2::askfirst:/bin/sh
```

```
tty3::askfirst:/bin/sh
tty4::askfirst:/bin/sh
EOF

touch mini/etc/init.d/rcS
chmod +x mini/etc/init.d/rcS
```

Encore une fois, recharger le module du noyau:

```
sudo apt-get download linux-image-$(uname -r)
sudo rm -rf linux-image-$(uname -r)
sudo dpkg -x $(find . -type f -name "linux-image-$(uname -r)*.deb" | head -n
1) linux-image-$(uname -r)
sudo cp -af linux-image-$(uname -r)/lib mini/
sudo chroot mini depmod
```

A la fin de cette étape, la taille du répertoire est d'environ 80 M (y compris le module du noyau), mais vous peut faire mieux.

Réduction en profondeur

Utiliser `du -hs * mini` pour voir tout le répertoire. On peut voir que l'espace le plus occupé est le répertoire des modules du noyau, qui représente près de la moitié de l'espace.

Ce qui suit est l'empreinte du module du noyau (mini/lib/modules/4.4.0-31-generic) ::

```
2.6M    arch
1.4M    crypto
11M     drivers
8.1M    fs
1.8M    lib
12M     net
832K    sound
508K    ubuntu
20K     virt
```

Net inclut **ceph**, **netfilter**, **openvswitch**, **atm**, **x2**, **appletalk**, etc., peuvent être supprimés.

Importanat: revérifier les dépendances après la suppression:

```
Chroot mini depmod
```

Le répertoire `mini/usr/lib` est également très volumineux: certaines bibliothèques peuvent être supprimées, par exemple, il existe de nombreuses bibliothèques de code dans `mini/usr/lib/x86_64-linux-gnu/gconv`. **Libdb-5.3.so** est également très volumineux, peut être supprimé sans utiliser man, etc.

À ce stade, on ne peut pas utiliser apt-get pour supprimer des éléments, utiliser au maximum dpkg, et le forcer à le nettoyer. Il n'y a pas de problèmes techniques à ce stade, et avec suffisamment de soin et de patience, on peut obtenir un système suffisamment petit.

Il existe d'autres moyens de personnaliser linux, tels que LFS http://linuxfromscratch.org/, buildroot https://buildroot.org/ et ainsi de suite.

Pour sauvegarder ce petit Linux sur le disque dur, c'est aussi très simple. Copier l'intégralité de la mini-copie sur une partition séparée, charger la partition où se trouve le répertoire racine, puis basculer vers `/sbin/init` dans le script init via `switch_root`:

```
# Une fois le module chargé ...
Monter/dev/sda3/mnt
Mkdir -p/mnt/mnt
Exec switch_root/mnt/sbin/init
```

Een raison de la commutation de la partition racine, des répertoires tels que `/proc` `/sys` `/dev` `/tmp` nécessitent un traitement supplémentaire: `mount -o move to mnt` ou définir des points de montage dans le fichier `mini/fstab` .

Etapes finales

Empaqueter et copier dans le répertoire de démarrage

```
(cd mini; sudo find . | sudo cpio -o -H newc | sudo gzip -9 > ../initramfs-
mini.gz)
sudo cp -f initramfs-mini.gz /boot
```

Redémarrer, appuyer sur c dans le répertoire grub pour entrer cmdline:

```
linux /vmlinux-<xxxxx>
initrd /initramfs-mini.gz
boot
```

Démarrage

Redémarrer l'ordinateur, appuyer sur c dans l'écran d'ammorce grub pour entrer dans cmdline et utiliser la commande suivante pour démarrer (compléter avec tab):

```
Linux/vmlinux- <xxxxx>
Initrd /initramfs-mini.gz
```

Après l'amorçage, le noyau extraira initramfs-mini.gz et entrera rapidement à l'invite de commande ubuntu. Ce Linux de base peut exécuter des opérations courantes, à l'exception d'outils tels que ping (qui peuvent être installés via apt-get), qui sont exécutés directement en mémoire, ce qui est identique à l'habitude (tous les fichiers étant chargés en mémoire,). La vitesse de la commande devrait être un peu plus rapide).