Linux From scratch: Construction d'une chaîne de compilation

Table of Contents

Le serveur de dev héberge un environnement permettant la consstruction d'un système d'exploitation Linux à partir du code source.

Une chaîne de compilation (en anglais : « toolchain ») désigne l'ensemble des paquets utilisés dans le processus de compilation d'un programme, pour un processeur donné.

Utilisation de la toolchain CLFS

Le site Linux From scratch fournit un livre décrivant l'installation étape par étape d'un système d'exploitation Linux à partir du code source.

Le lab LFS : Lab 1 - Construire une chaîne d'outils de compilation croisée a permis de construire Une chaîne de compilation sur le serveur de dev dans `/var/lfs/clfs/toolchain`.

Utilisation des outils de Centos

Les distributions fournissent les paquets nécessaires pour installer une toolchain depuis les dépôts officiels

Préparer l'environnement de build

Créer l'arbre de build

```
export CLFS="/var/lfs/<nom-de-l'arbre>"
mkdir -pv ${CLFS}/{rootfs,_src,_pkg}
```

Charger la directory roots avec un initramfs préconfiguré

```
cd ${CLFS}/rootfs
zcat /var/lfs/x86-busybox/rootfs.gz | cpio -idmv --no-absolute-filenames
```

A propos de ldconfig

Lorsqu'un logiciel requiert une liaison symbolique des bibliothèques il faut mettre à jour le cache.

La commande `ldconfig` est utilisée pour créer les liens nécessaires et mettre en cache les bibliothèques partagées les plus récentes trouvées dans les répertoires indiqués sur la ligne de commande, dans le fichier `/etc/ld.so.conf`, et dans les répertoires sûrs (`/lib` et `/usr/lib`). Le cache est utilisé par le chargeur/éditeur de liens `ld.so` ou `ld-linux.so`.

```
cp /etc/ld.so.conf* ${CLFS}/rootfs/etc -R
cp /sbin/ldconfig ${CLFS}/rootfs/sbin
```

Télécharger la source

Télécharger et décompresser l'archive and les `_src`

```
cd ${CLFS}/_src
wget https://curl.haxx.se/download/curl-7.66.0.tar.xz
tar -xvf curl-7.66.0.tar.xz
```

Construire le paquet

Dans le dossier décompressé, configurer et faire le build (le paquet doit être installé dans un répertoire distinct de l'arbre de build par exemple dans le dossier `_pkg`).

```
cd curl-7.66.0
./configure --prefix=/usr                           \
            --disable-static                        \
            --enable-threaded-resolver              \
            --with-ca-path=/etc/ssl/certs
```

```
make
make DESTDIR=$PWD/_pkg install
```

Retirer les chaînes de déboggage (pour gagner de l'espace) des fichiers créés dans les sous répertoires bin,lib et lib64.

```
strip -v _pkg/usr/lib/*
strip -v _pkg/usr/bin/*
```

Installer le package

Copier le contenu du dossier d'installation dans le dossier `rootfs`

```
cp -av _pkg/* ${CLFS}/rootfs
```

Faire un ldd sur le binaire créé

```
ldd _pkg/usr/bin/curl
    linux-vdso.so.1 =>  (0x00007ffe82fb9000)
    libcurl.so.4 => /lib64/libcurl.so.4 (0x00007fea6a9f7000)
    libssl.so.10 => /lib64/libssl.so.10 (0x00007fea6a785000)
    libcrypto.so.10 => /lib64/libcrypto.so.10 (0x00007fea6a322000)
    libz.so.1 => /lib64/libz.so.1 (0x00007fea6a10c000)
    libpthread.so.0 => /lib64/libpthread.so.0 (0x00007fea69ef0000)
    libc.so.6 => /lib64/libc.so.6 (0x00007fea69b22000)
    libidn.so.11 => /lib64/libidn.so.11 (0x00007fea698ef000)
    libssh2.so.1 => /lib64/libssh2.so.1 (0x00007fea696c5000)
    libssl3.so => /lib64/libssl3.so (0x00007fea69472000)
    libsmime3.so => /lib64/libsmime3.so (0x00007fea6924b000)
    libnss3.so => /lib64/libnss3.so (0x00007fea68f1e000)
    libnssutil3.so => /lib64/libnssutil3.so (0x00007fea68ced000)
    libplds4.so => /lib64/libplds4.so (0x00007fea68ae9000)
    libplc4.so => /lib64/libplc4.so (0x00007fea688e4000)
    libnspr4.so => /lib64/libnspr4.so (0x00007fea686a5000)
    libdl.so.2 => /lib64/libdl.so.2 (0x00007fea684a1000)
    libgssapi_krb5.so.2 => /lib64/libgssapi_krb5.so.2 (0x00007fea68254000)
    libkrb5.so.3 => /lib64/libkrb5.so.3 (0x00007fea67f6a000)
    libk5crypto.so.3 => /lib64/libk5crypto.so.3 (0x00007fea67d37000)
    libcom_err.so.2 => /lib64/libcom_err.so.2 (0x00007fea67b33000)
    liblber-2.4.so.2 => /lib64/liblber-2.4.so.2 (0x00007fea67923000)
    libldap-2.4.so.2 => /lib64/libldap-2.4.so.2 (0x00007fea676d0000)
    /lib64/ld-linux-x86-64.so.2 (0x00007fea6ac73000)
    librt.so.1 => /lib64/librt.so.1 (0x00007fea674c7000)
    libkrb5support.so.0 => /lib64/libkrb5support.so.0 (0x00007fea672b7000)
    libkeyutils.so.1 => /lib64/libkeyutils.so.1 (0x00007fea670b3000)
    libresolv.so.2 => /lib64/libresolv.so.2 (0x00007fea66e99000)
    libsasl2.so.3 => /lib64/libsasl2.so.3 (0x00007fea66c7c000)
```

```
    libselinux.so.1 => /lib64/libselinux.so.1 (0x00007fea66a54000)
    libcrypt.so.1 => /lib64/libcrypt.so.1 (0x00007fea6681d000)
    libpcre.so.1 => /lib64/libpcre.so.1 (0x00007fea665bb000)
    libfreebl3.so => /lib64/libfreebl3.so (0x00007fea663b7000)
```

Linux-vdso.so.1 est une bibliothèque virtuelle qui est automatiquement mappée dans l'espace d'adressage d'un processus par le noyau. Elle n'existe pas dans le système de fichiers.

Copier les librairies si nécessaire

```
cp -anLf
/lib64/{libcurl.so.4,libssl.so.10,libcrypto.so.10,libz.so.1,libpthread.so.0,
libc.so.6,libidn.so.11,libssh2.so.1,libssl3.so,libsmime3.so,libnss3.so,libns
sutil3.so,libplds4.so,libplc4.so,libnspr4.so,libdl.so.2,libgssapi_krb5.so.2,
libkrb5.so.3,libk5crypto.so.3,libcom_err.so.2,liblber-2.4.so.2,libldap-2.4.s
o.2,ld-linux-
x86-64.so.2,librt.so.1,libkrb5support.so.0,libkeyutils.so.1,libresolv.so.2,l
ibsasl2.so.3,libselinux.so.1,libcrypt.so.1,libpcre.so.1,libfreebl3.so}
${CLFS}/rootfs/lib64/
```

Utilisation d'un jail chroot

Description de la prison

Afin de ne pas toucher au système, on peut exécuter les étapes de l'installation dans une prison chroot. Le système de fichiers racine étant `/lfs/{nom_du_jail}`. Lorsque cet environnement est en place, on peut facilement ajouter ou supprimer des bibliothèques pour vérifier les dépendances ou la compatibilité. Ces bibliothèques ne doivent pas nécessairement correspondre à celles de /lib.

Création de la la structure du système de fichiers racine

La première chose à faire est de créer la structure du système de fichiers racine dont on a besoin pour un système minimal sous le répertoire spécifique, dans cet exemple: /lfs/busybox.

```
mkdir -pv /lfs/busybox
cd /lfs/busybox/
mkdir -pv dev/pts proc etc lib usr/lib var/run var/log
ln -s lib/ lib64
```

Ajout de /dev/pts et /dev

Outre la structure de répertoires, on a également besoin de périphériques et de liens vers les répertoires `/dev/pts` et `/proc` d'origine:

```
mknod dev/urandom c 1 9
chmod 0666 dev/urandom
mknod dev/ptmx c 5 2
chmod 0666 dev/ptmx
mknod dev/tty c 5 0
chmod 0666 dev/tty
mount -o bind /dev/pts dev/pts/
mount -o bind /proc proc/
```

Ajout de fichiers pour se connecter en chroot

```
cp /etc/localtime etc/
cp /etc/nsswitch.conf etc/
cp /etc/resolv.conf etc/
cp /etc/host.conf etc/
cp /etc/hosts etc/
cp /etc/shells etc/
touch var/log/lastlog
touch var/run/utmp
touch var/log/wtmp
```

Démarrer la prison chroot

Démarrer la prison.

```
chroot /lfs/busybox/
```