Projet RIM Linux

2016-10-14

Abstract

Projet Architecture et Système.

Création du RIM-Linux

Younesse Kaddar

En ligne : http://younesse.net/Architecture-systeme/RIM-Linux

1. Le fichier de configuration du noyau utilisé est le fichier `config_kernel`
2. L'image iso est `output.iso`

Le noyau Linux

```
mkdir -p build kernel RIM-Linux/rootbase
```

```
cd kernel
```

```
wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.9.tar.xz
tar -xJf linux-4.9.tar.xz
rm linux-4.9.tar.xz
cd linux-4.9
```

```
make nconfig
```

- Décocher `Enable loadable module support`
- Dans General Setup :
  - ne laisser que le support de l'initrd/l'initramfs

```
make -j11 bzImage
```

BusyBox

```
cd ../build
wget http://www.busybox.net/downloads/busybox-1.26.0.tar.bz2
bzip2 -d busybox-1.26.0.tar.bz2
tar xvf busybox-1.26.0.tar
rm busybox-1.26.0.tar
cd busybox-1.26.0
make menuconfig
make -j11
make -j11 install

cp -a ./_install/* ../../RIM-Linux/rootbase

cd ../../RIM-Linux/


ldd rootbase/bin/busybox
```

qui renvoie :

```
    linux-vdso.so.1 =>  (0x00007ffc42a9b000)
    libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007ffa52a51000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007ffa5268c000)
    /lib64/ld-linux-x86-64.so.2 (0x00007ffa52d57000)
```

Puis :

```
mkdir -p rootbase/lib/x86_64-linux-gnu/ rootbase/lib64

cp /lib/x86_64-linux-gnu/{libm.so.6,libc.so.6} rootbase/lib/x86_64-linux-gnu/

cp /lib64/ld-linux-x86-64.so.2 rootbase/lib64/


strip -v rootbase/lib/x86_64-linux-gnu/* rootbase/lib64/*

cd rootbase

rm linuxrc

ln -s bin/busybox init
```

On va récupérer la trame mise à notre disposition :

```
cp ~fhh/share/tp/projet/rim.linux.template.tbz2 ./

bzip2 -d rim.linux.template.tbz2
```

```
tar xvf rim.linux.template.tar
```

```
rm rim.linux.template.tar
```

Puis :

- On change le nom de l'utilisateur dans
  - `etc/passwd`
- Dans `etc/init.d/rcS` :
  - on décommente la ligne `busybox loadkmap < /etc/fr.kmap` pour activer le clavier français (on ajoutera un fichier `etc/fr.kmap` est bien présent)
  - on change le nom d'hôte : `/bin/hostname RIM-Linux`

Archive cpio de l'initramfs et isolinux

```
cd ..
```

```
mkdir -p rootcd/{boot,isolinux}
```

```
cp ../kernel/linux-4.9/arch/x86_64/boot/bzImage rootcd/boot/vmlinuz
```

```
find rootbase/* -print | cpio -o -Hnewc > root
```

```
cat root | gzip -9 > rootcd/boot/root.gz
```

```
rm root
```

On récupère isolinux

```
cd ../build
```

```
wget https://www.kernel.org/pub/linux/utils/boot/syslinux/syslinux-6.03.tar.xz
```

```
tar xJf syslinux-6.03.tar.xz
```

```
rm syslinux-6.03.tar.xz
```

```
cd ..
```

```
cp build/syslinux-6.03/bios/core/isolinux.bin RIM-Linux/rootcd/isolinux
```

```
cp build/syslinux-6.03/bios/com32/elflink/ldlinux/ldlinux.c32 RIM-Linux/rootcd/isolinux
```

Puis, on crée les fichiers

- `RIM-Linux/rootcd/isolinux/isolinux.cfg` :

```
display boot.txt

default 1

label 1
    kernel /boot/vmlinuz
    append initrd=/boot/root.gz
```

- `RIM-Linux/rootcd/isolinux/boot.txt` : le message de bienvenue

```
cd RIM-Linux

mkisofs -o output.iso -b isolinux/isolinux.bin -c isolinux/boot.cat -no-emul-boot -boot-load
```

Shell personnalisé

On se place à la racine du répertoire `RIM-Linux`.

Dans `rootbase/etc/inittab`, on remplace `/bin/sh` par `/bin/shell`, en ayant pris soin de placer notre shell personnalisé `shell` dans `rootbase/bin/shell`.

Il reste à ajouter les bibliothèques qu'utilise `shell` :

```
ldd shell
```

renvoie

```
linux-vdso.so.1 =>  (0x00007ffe3ba9a000)
libreadline.so.6 => /lib/x86_64-linux-gnu/libreadline.so.6 (0x00007fb5dd018000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fb5dcc53000)
libtinfo.so.5 => /lib/x86_64-linux-gnu/libtinfo.so.5 (0x00007fb5dca2a000)
/lib64/ld-linux-x86-64.so.2 (0x00007fb5dd25e000)
```

```
cp /lib/x86_64-linux-gnu/libreadline.so.6 rootbase/lib/x86_64-linux-gnu/

cp /lib/x86_64-linux-gnu/libc.so.6 rootbase/lib/x86_64-linux-gnu/

cp /lib/x86_64-linux-gnu/libtinfo.so.5 rootbase/lib/x86_64-linux-gnu/
```

(`cp /lib64/ld-linux-x86-64.so.2 rootbase/lib64/` : déjà fait précédemment).

Puis, ne pas oublier :

```
strip -v lib/x86_64-linux-gnu/*
```

Clavier

Dans `rootbase/etc/init.d/rcS`, on veillera bien à décommenter la ligne `busybox loadkmap < /etc/fr.kmap`.

Puis :

- Si on a les droits administrateurs (ce qui n'est pas le cas sur les machines de l'ENS) : "‘ su

../build/busybox-1.26.0/_install/bin/busybox dumpkmap > rootbase/etc/fr.kmap

- Sinon :

cd rootbase/etc/

wget http://younesse.net/assets/ArchiSysteme/fr.kmap

cd ../..

Pour le message d'accueil indiquant comment passer le système en clavier français ou anglais

- `rootbase/etc/motd` : le message de bienvenue

```
    ```
    / __\/ \/ __/| / \   / \/ \ /|/ \ /\\ \//
    | \/|| || |\/|| | | | || |\ ||| | || \ /
    | /| || | || | |_/\| || | \||| _/| / \
    _/__/_/ \| ____/_/_/ \|____//__/\\
                        .--.
                       |o_o |
                       |:_/ |
                      //   \ \
                     (| |)
                    /'_   _/`\
                    ___)=(___/
    Use: 'loadkmap < /etc/us.kmap' (en_US),
            'loadkmap < /etc/uk.kmap' (en_UK)
        or 'loadkmap < /etc/fr.kmap' (fr_FR) to change the keyboard mapping

    To display this message again : 'cat /etc/motd'
    ```

    - Puis :
```

- dans `rootbase/etc/init.d/rcS`, ajouter la ligne :

    ```sh
    /etc/init.d/monscript
    ```
- créer `rootbase/etc/init.d/monscript` :

    ```sh
    cat /etc/motd
    ```

Réseau

mkdir -p rootbase/usr/share/udhcpc/

cp ../build/busybox-1.26.0/examples/udhcp/simple.script rootbase/usr/share/udhcpc/default.script

chmod +x rootbase/usr/share/udhcpc/default.script

Puis ajouter les lignes suivantes au fichier `rootbase/etc/init.d/rcS` :

ifconfig eth0 up udhcpc -i eth0 “‘

Enfin :

```
cd rootbase
echo "127.0.0.1      localhost" > etc/hosts
echo "localnet    127.0.0.1" > etc/networks
echo "RIM-Linux" > etc/hostname
echo "order hosts,bind" > etc/host.conf
echo "multi on" >> etc/host.conf
```

Configurations supplémentaires

Pour une configuration plus exhaustive, on peut créer les fichiers :

- `rootbase/etc/nsswitch.conf` :

  ```
  # /etc/nsswitch.conf: GNU Name Service Switch config.

  passwd:     files
  group:      files
  shadow:     files

  hosts:      files dns
  networks:   files
  ```

- rootbase/etc/securetty :

```
# /etc/securetty: List of terminals on which root is allowed to login.

console

# For people with serial port consoles
ttyS0

# Standard consoles
tty1
tty2
tty3
tty4
tty5
tty6
tty7
```

- rootbase/etc/shells :

```
# /etc/shells: valid login shells.
/bin/shell
/bin/sh
/bin/ash
/bin/hush
```

- rootbase/etc/issue :

```
RIM-Linux, Younesse Kaddar \r \l
```

- rootbase/etc/busybox.conf :

```
# /etc/busybox.conf: Busybox configuration.

[SUID]
# Allow command to be run by anyone.
su = ssx root.root
passwd = ssx root.root
loadkmap = ssx root.root
mount = ssx root.root
reboot = ssx root.root
halt = ssx root.root
```

  – on veillera aussi à protéger ce fichier :

```
chmod 600 rootbase/etc/busybox.conf
```

Utilitaires

Les scripts `cpio_creator`, `kernel_copy`, `rebuild_iso`, `send_home` créés à la racine de `RIM-Linux/` automatisent certaines tâches répétitives effectuées pendant les tests.

- `cpio_creator` :

```
#!/bin/sh
HERE=$(pwd)
ROOTBASE=./rootbase
echo "The path is : $ROOTBASE."
echo "Creating ..."
cd $ROOTBASE
find ./* -print | cpio -o -Hnewc > $HERE/root
cd $HERE
cat root | gzip -9 > root.gz
rm root
mv root.gz rootcd/boot
echo "Done."
```

- `kernel_copy` :

```
#!/bin/sh
cp ../kernel/linux-4.9/arch/x86_64/boot/bzImage rootcd/boot/vmlinuz
```

- `rebuild_iso [options]` : crée l'image iso du RIM Linux et la lit avec `qemu`
  - options :
    - `-kernel` : copie le noyau `bzImage` créé dans le répertoire `../kernel`
    - `-kvm` : lance `qemu` avec l'option `-enable-kvm`

```
#!/bin/sh
./cpio_creator
if [ "$1" == "-kernel" ] || [ "$2" == "-kernel" ] ; then
  ./kernel_copy
  echo "Kernel copied"
else
  echo "Kernel not copied"
fi
mkisofs -o output.iso -b isolinux/isolinux.bin -c isolinux/boot.cat -no-emul-boot -boot

if [ "$1" == "-kvm" ] || [ "$2" == "-kvm" ] ; then
  echo "Qemu : KVM enabled"
  qemu-system-x86_64 -m 4G -enable-kvm -cdrom output.iso
else
  echo "Qemu : KVM disabled"
```

```
  qemu-system-x86_64 -m 4G -cdrom output.iso
fi
```

- send_home :

```
#!/bin/sh
rm -rf ~/RIM-Linux
cp -R ../RIM-Linux/ ~/
```