



- ENSEIRB - 4JLG -



4èmes Journées Linux de Grenoble
Linux Embarqué et Temps Réel
27 - 28 Novembre 2003



**ATELIER 3 : LINUX EMBARQUE.
CONSTRUIRE SON LECTEUR
MULTIMEDIA.
CONSTRUIRE SON ROUTEUR WIFI

CORRIGE**

Patrice KADIONIK
kadionik@enseirb.fr
www.enseirb.fr/~kadionik



1. TP 1 : CONSTRUCTION D'UN LINUX EMBARQUE SUR UNE CLE USB

1. Se placer dans le répertoire de travail 4JLGpk/tp1.

```
host% cd ~/4JLGpk/tp1
```

2. Analyser le système de fichiers root (répertoire root_fs/) qui sera utilisé pour construire la distribution Linux embarqué. Quel est le package mis en œuvre pour les commandes UNIX de base ? Quel est son intérêt ?

```
host% ls -lR root_fs|more
```

utilisation du package busybox. Faible empreinte mémoire.

3. On désire booter depuis un clé USB le PC light, c'est à dire la cible.

Se placer dans le répertoire ~/4JLGpk/tp1/linux-2.4.21. Configurer un noyau Linux 2.4.21 qui prend en charge les disques durs USB. On se servira du fichier config_usb comme fichier de configuration du noyau. On observera le patch appliqué au fichier ~/4JLGpk/tp1/linux-2.4.21/init/do_mount.c (fonction mount_root()).

Quel est son intérêt ?

```
host% cd ~/4JLGpk/tp1/linux-2.4.21
```

```
host% cp config_usb .config
```

Le patch du fichier do_mount.c permet d'attendre la reconnaissance de la clé USB par le noyau Linux avant de montage de la partition root hébergée dans la clé.

```
host% make clean
```

```
host% make dep
```

```
host% make bzImage
```

4. Générer les modules Linux et les installer sur l'hôte.

```
host% make modules
```

```
host% make modules_install
```

5. Recopier le noyau bzImage ainsi généré dans le root_fs précédent.

```
host% cp ~/4JLGpk/tp1/linux-2.4.21/arch/i386/boot/bzImage  
~/4JLGpk/tp1/root_fs/boot
```

6. Recopier les modules précédents générés dans le root_fs précédent.

```
host% cp -r /lib/modules/2.4.21  
~/4JLGpk/tp1/root_fs/lib/modules
```

7. Brancher votre clé USB sur le PC de développement, c'est à l'hôte. Créer une partition Linux. Formater la partition Linux pour un système de fichiers Linux ext3 avec 2000 inodes. La clé USB est vue par Linux comme un périphérique SCSI (/dev/sda, partition 1 /dev/sda1).

```
host% fdisk /dev/sda
```



```

Commande (m pour aide) : p
Commande (m pour aide) : n
Action de commande
    e    Etendue
    p    Partition primaire (1-4)
p
Nombre de partitions (1-4): 1
Premier cylindre (1-1024, 1 par défaut) :
Utilisation de la valeur par défaut 1
Dernier cylindre ou +size ou +sizeM ou +sizeK (1-1024, 1024
par défaut) :
Utilisation de la valeur par défaut 1024

```

```

Commande (m pour aide) : p

```

```

Disk /dev/sda: 66 MB, 66060288 bytes
3 heads, 42 sectors/track, 1024 cylinders
Units = cylinders of 126 * 512 = 64512 bytes

```

Périphérique	Amorce	Début	Fin	Blocs	Id
/dev/sda1	1	1024	64491	83	Linux

```

Commande (m pour aide) : w
La table de partition a été modifiée !
Appel de ioctl() pour relire la table de partition.
Synchronisation des disques.

```

```

host% mkfs.ext3 -N 2000 /dev/sda1

```

8. Monter la clé USB sur le répertoire /key. Y transférer le système de fichiers root_fs. Installer sur la clé le bootloader lilo en utilisant comme fichier de configuration le fichier /key/etc/lilo.conf.

```

host% cd ~/4JLGpk/tp1/
host% mount -t ext3 /dev/sda1 /key
host% cp -r root_fs/* /key
host% lilo -v -r /key

```

9. Démonter la clé. Tester le résultat produit en bootant la cible PC Light depuis la clé USB

```

host% umount /key

```

Le PC light est éteint. Brancher la clé USB puis rebooter la cible.

10. Quelle est la taille de la distribution Linux embarqué ainsi construite ?

```

host% df

```



2. TP 2 : CONSTRUCTION D'UN LECTEUR MULTIMEDIA SOUS LINUX EMBARQUE

Le but de ce TP est de construire un lecteur multimédia multiformat sur une cible sous Linux embarqué. La cible doit être capable de lire un DVD ou un fichier MP3 par exemple...

1. Se placer dans le répertoire de travail ~/4JLGpk/tp2.

```
host% cd ~/4JLGpk/tp2
```

2. On va utiliser le projet libre MPlayer qui est un lecteur multimédia multiformat : MP3, MPEG2, DivX, XVID, MPEG4 et multisupport : DVD, VCD, fichier...

Aller dans le répertoire MPlayer-0.90 et lancer le script mp_configure_pk de configuration puis compiler le projet Mplayer.

```
host% cd ~/4JLGpk/tp2/Mplayer-0.90
```

```
host% ./mp_configure_pk
```

```
host% make
```

3. Recopier le root_fs précédemment créé durant le TP 1 sous ~/4JLGpk/tp2 et y installer l'exécutable mplayer sous ~/4JLGpk/tp2/root_fs/bin.

```
host% cp -r ~/4JLGpk/tp1/root_fs ~/4JLGpk/tp2
```

```
host% cp ./mplayer ~/4JLGpk/tp2/root_fs/bin
```

4. Il convient aussi d'installer les bibliothèques dynamiques utilisées par mplayer dans le root_fs. Pour cela, on utilisera le script utilitaire mklibs.sh. Voir l'aide en ligne avec :

```
host% mklibs.sh -h
```

```
host% cd ~/4JLGpk/tp2/root_fs
```

```
host% mklibs.sh -v -d lib bin/*
```

5. Copier le répertoire ~/4JLGpk/tp2/mp3 contenant un fichier de test MP3 dans le root_fs.

```
host% cd ~/4JLGpk/tp2/
```

```
host% cp -r mp3 root_fs
```

6. Mettre à jour le système de fichiers root de la clé USB avec le nouveau root_fs et réinstaller le bootloader sur la clé.

```
host% cd ~/4JLGpk/tp2/
```

```
host% mount -t ext3 /dev/sda1 /key
```

```
host% \rm -rf /key/*
```

```
host% cp -r root_fs/* /key
```

```
host% lilo -v -r /key
```

7. Tester le résultat produit en bootant la cible depuis la clé USB.

On activera le DMA du lecteur de DVD avec le script godma. On testera la lecture d'un DVD présent dans le lecteur DVD de la cible avec le script gomplayer.

```
host% umount /key
```

Boot sur la clé USB.



```
target% godma  
target% gomplayer 1
```

8. Tester la lecture d'un fichier de test MP3.

```
target% mplayer /mp3/tst1.mp3
```

9. Modifier les scripts d'initialisation du système pour lancer automatiquement la lecture du DVD présent dans le lecteur de DVD. Rebooter la cible et tester le résultat final.

```
target% cd /etc/rc.d  
target% vi S01mplayer.sh  
Y ajouter les lignes en fin de fichier :  
/bin/godma  
/bin/gomplayer 1
```

PS : dans le root_fs utilisé, tous les codecs pour mplayer ainsi que ses fichiers de configuration ont été préalablement inclus afin de gagner du temps.

Voir le projet OM-CUBE (Open MultiMedia Machine) :

<http://www.enseirb.fr/~kadionik/om-cube/om-cube.html>

3. TP 3 : CONSTRUCTION D'UN ROUTEUR WIFI SOUS LINUX EMBARQUE

Le but de ce TP est de construire un routeur Wifi sur une cible sous Linux embarqué. La cible ne possède pas de slot PCMCIA pour supporter une carte Wifi PCMCIA. Les extensions ou clés Wifi sur USB sont très rarement supportées par Linux pour l'instant. On a donc choisi une plateforme moins compacte mettant en œuvre la cible PC Light et un routeur/point d'accès (AP) Wifi/ADSL connectés entre eux par une liaison Ethernet. Les fonctions de routage sont alors déportées sur le routeur/AP Wifi/ADSL au lieu d'être sur la cible PC Light.

Il existe néanmoins une version de PC Light intégrant une extension Wifi. Il faudra alors intégrer les fonctions de routeur dans la cible PC Light ainsi que les fonctionnalités de pare feu. Ce cas de figure ne sera pas traité ici...

1. Se placer dans le répertoire de travail ~/4JLGpk/tp3.

```
host% cd ~/4JLGpk/tp3
```

2. Le répertoire tools contient un certain nombre d'utilitaires rangés dans les sous-répertoire bin et sbin pour configurer et espionner le port Ethernet de la cible. Recopier respectivement ces utilitaires sous ~/4JLGpk/tp3/root_fs/bin et ~/4JLGpk/tp3/root_fs/sbin.

```
host% cp ~/4JLGpk/tp3/tools/sbin/* ~/4JLGpk/tp3/root_fs/sbin  
host% cp ~/4JLGpk/tp3/tools/bin/* ~/4JLGpk/tp3/root_fs/bin
```



3. On va embarquer un serveur www léger httpd dans la cible. Se placer dans le répertoire de ~/4JLGpk/tp3/httpd et compiler httpd. Installer l'exécutable produit dans le root_fs sous ~/4JLGpk/tp3/root_fs/sbin.

```
host% cd ~/4JLGpk/tp3/httpd
host% make
host% cp httpd ~/4JLGpk/tp3/root_fs/sbin
```

4. Comme précédemment, il convient aussi d'installer les bibliothèques dynamiques utilisées par ces utilitaires dans le root_fs. Pour cela, on utilisera le script utilitaire mklibs.sh.

```
host% cd ~/4JLGpk/tp3/root_fs
host% mklibs.sh -v -d lib sbin/* bin/*
```

5. Le root_fs fourni contient un noyau Linux compilé ainsi que les modules Linux de pilotage du contrôleur Ethernet de la cible (8139too, contrôleur Ethernet Realtek). Mettre à jour le système de fichiers root de la clé USB avec le nouveau root_fs et réinstaller le bootloader sur la clé.

```
host% cd ~/4JLGpk/tp3/
host% mount -t ext3 /dev/sda1 /key
host% \rm -rf /key/*
host% cp -r root_fs/* /key
host% lilo -v -r /key
```

6. Tester le résultat produit en bootant la cible depuis la clé USB.

```
host% umount /key
Boot sur la clé USB.
```

7. Sur la cible, charger les modules du contrôleur Ethernet 8139too puis configurer les interfaces réseau lo et eth0. On lancera le client DHCP dhclient pour récupérer une adresse IP pour l'interface eth0 attribuée par le point d'accès/routeur Wifi/ADSL

```
target% ifconfig lo
target% modprobe 8139too
target% dhclient eth0
```

8. Tester la connectivité IP avec l'AP/routeur Wifi avec la commande ping sachant que l'adresse IP du routeur/AP Wifi est 192.168.0.1.

```
target% ping 192.168.0.1
```

9. Lancer le serveur www httpd en tâche de fond puis lancer l'espion tcpdump sur l'interface Ethernet eth0.

```
target% httpd &
target% tcpdump -vv -i eth0
```

9. Depuis un PC possédant une carte Wifi convenablement configurée pour accéder au routeur/AP Wifi (SSID=pk, pas de WEP, pas de contrôle sur l'adresse MAC, client DHCP), se connecter au serveur www embarqué de la cible via Wifi à l'aide d'un navigateur web. Observer les traces sur la console de la cible...

That's all folks !