

# Cours " Informatique Embarquée "

François Armand

## M2 SRI/Crypto

Exercice ! " # \$ % ! o & embre 2 ' % (   
A rendre avant le Samedi ' \* / % 2 / 2 ' % ( 2 + , # -   
. + séances de / 0 .

armand@informatique2.uni-paris.diderot.fr

## 3 linux \$ 4 EM 5 et 6 BusyBox

### 1 Consignes:

3a remise se fera en suivant les consignes habituelles

### 2 Introduction

Le but de ce TP est de créer un petit système basé sur un noyau Linux et utilisant BusyBox. La machine sur laquelle ce système tournera est une machine « fictive » qui sera fournie émulée par le logiciel **qemu**. "<http://www.qemu.org/>". 'emu sera présenté en cours lors du chapitre consacré aux machines virtuelles sous réserve de temps.

Pour l'essentiel il faut donc :

- récupérer les sources d'un noyau Linux les configurer et compiler le système
- « créer un disque » en fait un fichier qui pourra être utilisé par notre système cible
- on pourra utiliser un programme de type « hello world » comme programme `init` pour vérifier que tout fonctionne correctement.
- récupérer les sources de BusyBox les configurer et les installer sur le disque précédemment créé
- « jouer avec les commandes de BusyBox ».

La machine émulée par 'emu sera dans ce TP une machine de type P ) basée sur un processeur intel.

, référence# [http://pfichoux.free.fr/articles/lmfhs/0\\$busybox\\$bb1nutshell.pdf](http://pfichoux.free.fr/articles/lmfhs/0$busybox$bb1nutshell.pdf)

2 ans la suite il vous est possible d'effectuer le TP sur les bases validées "en repartant des archives déjà téléchargées sur les machines de l'56, & ou de partir en « hors pistes » en travaillant sur des versions différentes de celle testée. )ette possibilité de « hors pistes » est déconseillée aux personnes n'ayant jamais encore généré de noyau Linux. Par contre suivant le temps dont vous disposez vous pouvez suivre le parcours balisé puis recommencer en « hors pistes ».

.Il y a donc / parcours possibles :

- le parcours balisé "ce qui ne veut pas dire que l'arrivée est garantie
- le parcours hors pistes qui nécessite des droits super utilisateur sur la machine Linux et n'est donc pas réalisable sur les machines de l'56, .

## 3 Générer un noyau Linux

On va installer les sources du noyau Linux dans un répertoire. Sachez que le `$HOME` ne doit pas contenir de caractères spéciaux. La construction de Linux ne supporte pas ce genre de « fantaisies »

### 3.1 Parcours balisé

On utilisera la version 3.12.4 copiée localement

```
< mkdir TP5; cd TP5
```

```
< cp ~/Armand/Downloads/linux-3.12.4.tar.xz
```

### 3.2 Parcours hors pistes

Les sources du noyau Linux sont disponibles sur le site <http://www.kernel.org/>. Téléchargez la version de votre choix dans un répertoire de travail. Par exemple:

```
# mkdir TP5; cd TP5
```

```
# wget https://www.kernel.org/pub/linux/kernel/v3.x/linux-3.12.tar.xz
```

### 3.3 Génération du système

#### 3.3.1 Extraire les sources du noyau (balisé et hors pistes)

Pour générer un noyau Linux il vous faut maintenant extraire les sources Linux de l'archive. La commande `tar` si nécessaire se réfère au manuel de la commande `tar`. Attention l'archive en `.xz` nécessite l'option `-J`.

#### 3.3.2 Configurer le système (balisé et hors pistes)

```
# cd linux-3.12
```

```
# make help < vous donnera des indications sur ce que peut faire le make
```

```
# make defconfig < si vous voulez utiliser la configuration par défaut
```

```
# make menuconfig < si vous voulez configurer manuellement votre système
```

Il est conseillé de ne pas utiliser la configuration par défaut afin de minimiser les temps de génération du système Linux.

#### 3.3.3 Parcours balisé

Il y a un fichier de configuration pour la version 3.12.4 à l'emplacement suivant

`~/Armand/Downloads/config-3-12-fa` copiez-le dans votre répertoire linux dans le fichier `« .config »`. Il faudra quand même exécuter un `make menuconfig` par la suite cependant il vous est demandé pour vous familiariser avec les mécanismes de configuration du noyau d'effectuer quelques opérations décrites plus loin.

### 3.3.4 Parcours hors pistes

Si vous avez utilisé une version différente de Linux, il faut alors régler sa configuration soi-même. Pour gagner du temps, éliminez un maximum de fonctionnalités dont nous n'aurons pas besoin : les file systèmes autre que `ext2`, tout ce qui est réseau, virtualisation, cryptographie... Les drivers inutiles. Éliminez le support des périphériques et matériels inutiles (support du son, ...)

### 3.3.5 Configurations finales

Assurez-vous en sorte que le noyau généré soit « le vCtre » Modifiez l'option qui permet de

définir le suffixe de la version du noyau. 2 ans le fichier confi! mis 4 votre disposition le suffixe de la version :@; / est défini comme étant « @fa ». , etrouve7 et modifie7 cette option pour personnalise7 votre version. + n pourra retrouver cette chaîne une fois notre système lancé en invoquant la commande uname @a.

Le mécanisme de confi!uration du noyau permet#

- d'exclure certaines fonctionnalités
- d'inclure certaines fonctionnalités dans le noyau dans le noyau
- de les définir comme des modules qui doivent alors \*tre char!és dynamiquement.

6= La confi!uration définie par le fichier confi!:@; /@fa ne définit aucun type de fichiers ext/fs. D'odifie7 la confi!uration pour qu'il soit inclus dans le noyau automatiquement. Ft pas sous forme de module.

>emande? de )aide si né essaie @

Gi votre confi!uration n'est pas correcte ce n'est pas très !rave. 8otre système ne démarrera probablement pas mais vous devrie7 obtenir des messages d'erreur qui avec un peu d'aide et d'entra!nement vous aideront 4 obtenir une confi!uration correcte de votre système.

Iléments de compte@rendu# (2 pt)

- .ndique7 comment vous ave7 procéder pour effectuer les confi!urations ci@dessus "points B et B&. 2onne7 le « chemin » de sélection dans le menu de confi!uration.
- + J se trouve la documentation du noyau LinuxK

Enfin\$ au as oA\$ &ous pourre?\$ en désespoir de ause\$ obtenir un fi 8ier de onfi7uration fon tionne) en me )e demandant!

### 3.3. ( Co #pile)

< ma=e

Gur la machine mono@processeur que 3(ai utilisée le temps d'exécution de cette commande a été de // minutes. .l est possible qu(en salle de TP ce soit un peu plus lon!.

Profite7 en #

- Pour chercher quelles sont les différences ma!eures entre la version :.; / de Linux et la version précédente....
- +u passe7 du temps sur le TP LMN ... *Ah oui, il arrive que l'on ait à travailler sur deux choses en même temps, par exemple, un bug urgent pour dépanner un client et un développement urgent pour la nouvelle version de logiciel qui doit sortir à la fin de la semaine !*

Iléments de compte@rendu#

- ) ombien de temps a pris votre compilation "donne7 des précisions sur l'environnement de !énérati&K
- + J se trouve le fichier !énéré quelle taille fait@iK
- ' uel est le format de ce fichierK
- Gi on avait !énéré le noyau Linux pour notre machine de développement que faudrait-il faire ensuite pour « installer » ce noyau et tenter de redémarrer notre machine avec notre nouveau noyauK "2onne7 les quelques commandes nécessaires&.
- ' uelles différences ave7@vous trouvé entre la :.; / et la version précédente K ' uelles sources ave7@vous utilisées K

Au as oA\$ i) est possible d'obtenir une arbores en e +.%2.fa pré ompi)ées p)us )ourde ;

installer\$ mais le temps de compilation devrait s'en trouver considérablement réduit!

## % **QEMU**

'FD 5 est un émulateur de machines. 'FD 5 peut par exemple émuler

- un P) xON sur un P) xON " : /bits et NO bits
- un P) xON sur une machine 4 base de PP)
- une machine basée sur un PP) sur une machine xON
- et beaucoup plus encore.

. Il est recommandé de comprendre un minimum ce que qemu peut vous permettre de faire #

```
# type qemu
# ls ...../qemu*
# man qemu
# qemu --help
```

'FD 5 ne se contente pas seulement d'émuler le processeur mais il fournit aussi une émulation de certains périphériques associés à une machine# mémoire physique disques lignes séries interfaces réseau écran graphique périphériques audio ... La configuration utilisable varie suivant les processeurs émulés. Dans notre cas nous nous en servons pour avoir une deuxième machine xON 4 notre disposition.

### 4.1 Si on générerait un programme pour processeur arm !

Écrire un programme aussi compliqué que « Hello 'emu »

. Il faut maintenant le compiler pour processeur B , D .) Oup de chance Pierre Letouzey nous a gentiment installé un compilateur croisé ! rQce 4 [emdebian.org/](http://emdebian.org/) &. Le répertoire `/usr/bin` doit normalement être connu dans votre variable shell RPBTP.

+n peut donc compiler #

```
# arm-linux-gnueabi-gcc -static hello.c -o hello_arm
```

Puis vérifier la tête de notre programme 9

```
# file hello_arm
```

, este 4 exécuter ce programme... . Il suffit de trouver une machine avec un processeur B , D##. Votre téléphone est probablement la machine qu'il vous faut... Plus simple utilisons 'emu #

```
# qemu-arm hello_arm
```

Saouh 9 Lon K Ben moi si 9

Iléments de compte@rendu#

- Pourquoi `-static` 4 la compilation K

4.2 Oui, mais nous on a généré un noyau Linux pour x8 et notre machine est une 4 bits et on ne peut pas changer le noyau booté dessus...

+n peut dans un premier temps vérifier si notre noyau Linux généré précédemment semble fonctionner raisonnablement#

```
# qemu-system-i386 -nographic -append "console=ttyS0" -kernel
TP6/linux.3-12/ ... à vous de trouver la suite !
```

Si vous avez généré un noyau pour machine NO bits #

```
# qemu-system-x86_64 -nographic -append "console=ttyS0" -kernel  
TP6/linux.3-12/ ... trouvez la suite !
```

Le système devrait démarrer mais comme il n'a pas de système de fichiers à disposition il ne pourra pas terminer son initialisation de manière utile. Il y a assez peu de chances pour que la commande ci-dessus vous soit réellement utile. Pour terminer qemu trouvez son pid et tuez-le depuis un autre terminal.

*Hors-Pistes : Utilisez votre version de système.*

*Hors-Pistes 2 : Récupérez une version récente de QEMU, générez-la et utilisez cette version!*

## (2) Fabriquer un disque 4EM5

Pour que notre système puisse fonctionner il faut lui fournir « un disque » dans notre cas une émulation de disque contenant un « root file system ». On pourrait aussi utiliser un « ramdisk ».

### 3.1 Parcours balisé

Installez l'archive `~armand/Downloads/genimage.tgz`

```
# mkdir -p ~/TP5/mondisque  
# cd ~/TP5/mondisque  
# tar -zxvf ~armand/Downloads/genimage.tgz
```

Créez un répertoire de nom `root` dans le répertoire `mondisque` que vous venez de créer. Le répertoire `root` est la racine de l'arborescence qui servira de système de fichiers racine à votre noyau Linux.

```
# mkdir -p ~/TP5/mondisque/root
```

Vous êtes maintenant prêts à créer un disque avec un système de fichiers ext/fs qui pourra être utilisé par votre noyau Linux au sein de l'emu. L'arborescence de fichiers sur ce disque sera une copie de ce l'arborescence que vous créez sous « `root` ». Il y a donc à y mettre des fichiers.

### 4.3.2 Parcours Hors Pistes

*Il y a d'autres moyens de parvenir à créer un disque pour Linux sous Qemu. Cela suppose cependant d'avoir les droits de super utilisateur sur la machine de développement, ce qui n'est pas possible sur les machines de la salle de TP. Si vous avez une machine Linux personnelle vous pouvez tenter cette séquence d'opérations décrite ci-dessous :*

#### 4.3.2.1 Créer le disque

Créer un disque pour une machine Qemu :

```
# qemu-img create -f raw mondisque.img 20M
```

```
# losetup -f --show mondisque.img #crée une entrée dans /dev
```

```
# fdisk /dev/loop1 # on reprend le nom obtenu par losetup
```

- *fdisk permet de formater un disque*
- *Utilisez le menu pour créer une partition occupant tout l'espace du disque*
- *n, p, 1, RC, RC (où RC signifie « retour chariot/ entrée / enter..)*
- *N'oubliez pas de sauvegarder la déclaration de la partition (w) avant de quitter (q)*

- On peut relancer fdisk pour vérifier que la partition a bien été créée (fdisk -l)
- Notez la taille de la partition ainsi créée.

Éléments de compte-rendu:

- Dans la séquence ci-dessus, qu'est-ce qui nécessite d'avoir les droits super-utilisateur?

#### 4.3.2.2 Peupler le disque créé

En fait, ce qui nous intéresse n'est pas le disque « physique » émulé dans le fichier mondisque.img, mais la partition, donc un sous ensemble du disque « physique ». c'est cette partition qu'il va nous falloir initialiser pour pouvoir la peupler avec BusyBox ultérieurement.

```
# losetup -d /dev/loop1 #défaire l'association entre loop1 et mondisque.img
# losetup -f --show -o 32256 mondisque.img
#crée une entrée dans /dev pour accéder à la partition (et non plus au disque)
# mke2fs -j /dev/loop1 xxxxx
# xxxx est la taille de la partition obtenue ci-dessus
# mkdir ~TP5/mnt
# mount /dev/loop1 ~TP5/mnt
```

Vous avez un système de fichiers vide dans votre arborescence! Il vous reste à le peupler!

Éléments de compte-rendu:

- Dans la séquence de commandes ci-dessus, il manque des commandes pour produire un équivalent de ce que fait genimage. Lesquelles?
- Dans la séquence ci-dessus, qu'est-ce qui nécessite d'avoir les droits super-utilisateur? (Bis repetita)

#### & \*on+our le #onde,

+n peut avant de jouer avec BusyBox créer un programme « init » extrêmement original et vérifier si le système fonctionne correctement.

- )rée7 en ) un programme de type qui répète ce message périodiquement chaque seconde pendant ;T secondes avant de se terminer.  

```
printf (Hello World!\n");
```
- Bttention la configuration préparée "confi!@: @; /@fa& vise une machine : / bits... .Il faut donc générer un binaire pour machine : / bits...  

```
# gcc -m32 hello.c -o hello_32
# file hello_32 ; ./hello_32
```
- Compile7 en utilisant une édition de liens statique.
  - )omment PourquoiK
- )opie7 votre programme dans root\$sbin\$init  

```
# mkdir -p ~/TP5/mondisque/root/sbin
# cp hello ~/TP5/mondisque/root/sbin/init
```
- )rée7 un répertoire \$dev. Les « devices » sont créés à partir d'un fichier dev.txt

```
# mkdir -p ~/TP5/mondisque/root/dev
```

- Puis lancez la commande `!enima!e` pour obtenir un disque par exemple baptisé `test.img`
- # `cd ~/TP5/mondisque`
- # `./genimage`

Vous trouverez dans ce répertoire un fichier `dev.txt` utile pour définir d'autres device pour votre système si nécessaire.

Vous avez maintenant un disque prêt à l'emploi. Félicitations !

#### Éléments de compte-rendu

- Listez l'arborescence obtenue "format @l"
- Quelle commande avez-vous utilisée pour générer votre programme « init » ? Pourquoi ?
- Donnez le résultat de la commande `file` sur votre programme `init` \$ hello.
- Quelle est la taille sur disque de cette commande `init` \$hello ? Par quelle "s" commande"s" avez-vous trouvé cette information ?
- Quelles sont les tailles de segments de code et de données sur disque et en mémoire ? Par quelle"s" commande"s" avez-vous trouvé ces informations ?
- À quelle adresse en mémoire virtuelle le segment de code sera-t-il placé ? Et le segment de données ? Et la pile ? Par quelle"s" commande"s" avez-vous trouvé ces informations ?
- Pourquoi faut-il utiliser une édition de liens statique ?
- Que faudrait-il faire pour utiliser une édition de liens dynamique ?

### #.1 \$ne fois le & "is"ue ( %abri"ué

- , lancez `'emu`
- # `qemu-system-x86 -nographic -kernel TPT/linux.3.12/arch/x86/boot/bzImage -hda tst.img -append 'root=/dev/hda1 console=ttyS0'`
- Gi « Pello Sorld » apparaît c'est tout bon. Sinon « .Il y a quelque chose cloche là dedans (y retourne immédiatement » "Boris 8ian".

#### Éléments de compte-rendu

- Quelles erreurs éventuelles avez-vous rencontrées ? Pourquoi ? Comment avez-vous résolu ces problèmes ?
- À quoi sert l'argument « `@append` » de `'emu` ? Que peut-on passer comme valeur à cet argument ? + J'ai trouvé cette information ?
- Que se passe-t-il quand votre programme « `init` » se termine après avoir affiché « Pello Sorld » ? Pourquoi ?

### #.2 \$n peu "e "ynamisme!

Lorsqu'on va maintenant tenter de recommencer cette expérience mais en utilisant une édition de liens dynamique pour notre programme `init`.

- Procédez à une édition de liens dynamique de votre programme `hello$init`.
- Il vous faut déterminer les bibliothèques dynamiques \$ partagées dont il a besoin et les copier à l'endroit approprié dans le répertoire `root` créé plus haut.
- Les commandes `ldd` et `cp` devraient suffire à votre bonheur. "« » (est quand le bonheur ? » @) ali@ La bibliothèque ... !ate... n'est pas une vraie bibliothèque inutile de chercher à la

recopier.

- , écrivez votre disque et relancez le FD 5. Persévérez jusqu'à obtenir un fonctionnement correct ou épuisement du disque.

### Iléments de compte rendu

- Quelle commande avez-vous utilisée pour générer votre programme « init » ?
- Donnez le résultat de la commande file sur votre programme init \$ hello.
- Quelle est la taille sur disque de cette commande init \$hello ? Par quelle commande avez-vous trouvé cette information ?
- Quelles sont les tailles de segments de code et de données sur disque et en mémoire ? Par quelle commande avez-vous trouvé ces informations ?
- Donnez le résultat de la commande ldd.
- Indiquez quelles bibliothèques vous avez copiées dans votre arborescence root.

### ( \*usy \*ox

« Pello %orld » est un peu limité comme fonctionnalités pour un système même si ça pourrait faire un système embarqué très utile sur un appareil dont le seul but serait d'afficher « Pello %orld » sur un écran quand on le met sous tension. C'est une sorte de « coucou » suisse électronique. On va donc enrichir le système avec BusyBox.

### 1.1 Première étape

- Récupérez les sources de BusyBox par exemple

```
# mkdir ~/TP5/BB; cd ~/TP5/BB;
# wget http://busybox.net/downloads/busybox-1.16.0.tar.bz2
```

 ou une autre version.
- Configurez BusyBox de manière à ce que le résultat soit généré avec une bibliothèque dynamique.
- Configurez BusyBox de la manière suivante :
  - `# make menuconfig`
  - Pensez à générer pour un processeur : / bits "options de build" du compilateur
  - Archivez ce que `tar, gzip, unzip`
  - Éliminez les utilitaires Debian
  - Supprimez `ed` et `patch` des éditeurs fournis par BusyBox
  - Supprimez le support de l'argument `-maxdepth` de la commande `find`
  - Éliminez la commande `msg`
  - Supprimez le support des commandes « `minix` » de manipulation de systèmes de fichiers.
  - Supprimez la commande `beep` et le support de `splashscreens`.
  - Éliminez le support des démons réseau de P8N des utilitaires d'impression et de mail.
- Compilez

```
# make
```
- Pensez aux bibliothèques dynamiques

- Installez votre BusyBox dans votre système de fichiers qui sert à fabriquer votre disque Linux
 

```
# make CONFIG_PREFIX=~TP6/mondisque/root install
# cd ~TP6/mondisque; ./genimage
```
- Relancez votre machine sous 'FD 5'.
- Notez l'attention clavier qwerty par défaut...

### Il éments de compte rendu# (2 pt)

- Brevévous rencontré des problèmes? Comment les avez-vous résolus?
- Quelle séquence de commandes avez-vous essayé sur votre machine 'FD 5'?
- Quelle taille fait votre fichier binaire exécutable BusyBox? Quelle est la taille de sa section de code? La taille de sa section de données sur disque en mémoire?
- Pourriez-vous configurer busybox de manière à ce que cette commande ait une taille plus réduite que le \$bin\$bash de votre machine de développement?

## 2. Un peu plus compliqué

- Réécrire le répertoire suivant dans votre arborescence root#
  - root\$etc\$init.d
  - root\$proc
- Réécrire le fichier#
  - root\$etc\$init.d\$rcG
  - avec le contenu suivant#
 

```
#!/bin/sh
loadkmap < /etc/french.kmap
mount -t proc /proc
mount -o remount,rw /
mount -a
```
- Sur un terminal de votre machine de développement exécutez la commande suivante#
 

```
...../genimage/root/bin/dumpkmap >
.../genimage/root/etc/french.kmap
```
- Rendez le script rcG exécutable
 

```
# chmod a+rx
```
- Reconstruisez votre disque et relancez votre système.