



**Compte rendu**

**Belhadj Walid**  
**uapv2101413**

## 1 Raspberry pi - le matériel

[https://en.wikipedia.org/wiki/Raspberry\\_Pi](https://en.wikipedia.org/wiki/Raspberry_Pi)

- Quel est la référence du processeur qui équipe votre carte Raspberry Pi 2 B?

La référence du processeur de la carte Raspberry PI 2 B est : **ARM Cortex-A7**

- Quel est son type d'architecture ? Combien de cœurs possède-t-il ?

Type d'architecture : 32 bits, et quad-core (4 cœurs).

- Quel est la quantité de RAM disponible ?

Ram de 1 Go.

- La mémoire est-elle gérée par une MMU ?

Oui.

- Quel est le ratio entre la mémoire CPU/GPU par défaut ?

GPU 250 Mhz, CPU : 900 MHz, ratio :  $900/250 = 3.6$

- Quels sont les bus de communication disponibles ?

Les bus disponibles sont ; I2C, PISCAN-2 CAN-bus, ( controller area network), SPI RPI, GPIO

- Quelle version de VFP (Vector Floating Point) support-il ?

Version 4

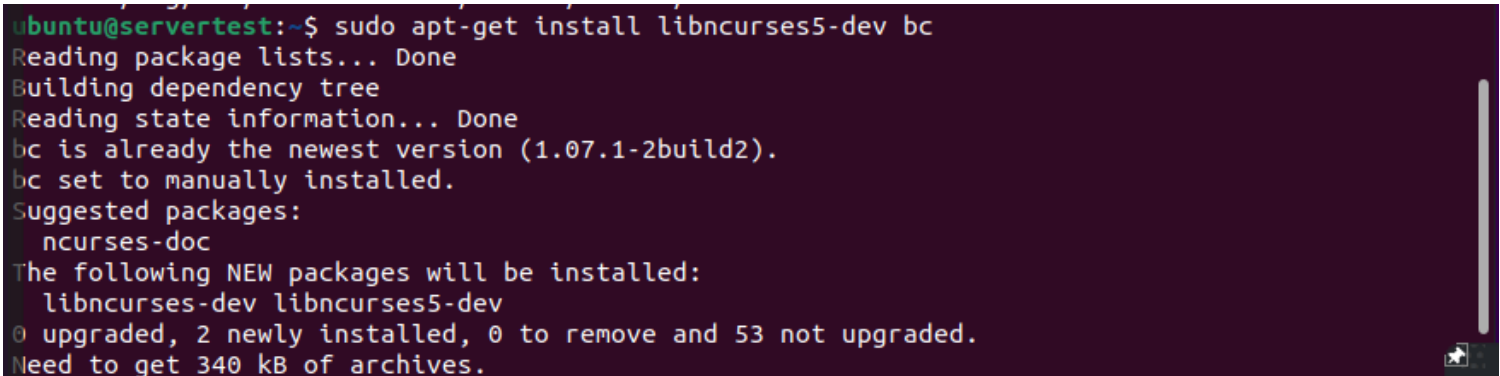
## 2 Buildroot pour Raspberry Pi 2 et QEMU

### 6.1 Installation de l'environnement Buildroot

**Buildroot** est un outil en ligne de commande permettant de générer sa propre image de GNU/Linux optimisée pour l'embarqué.

Commencez par installer les dépendances nécessaires à l'utilisation de Buildroot :

```
sudo apt-get install libncurses5-dev bc
```



```
ubuntu@servertest:~$ sudo apt-get install libncurses5-dev bc
Reading package lists... Done
Building dependency tree
Reading state information... Done
bc is already the newest version (1.07.1-2build2).
bc set to manually installed.
Suggested packages:
  ncurses-doc
The following NEW packages will be installed:
  libncurses-dev libncurses5-dev
0 upgraded, 2 newly installed, 0 to remove and 53 not upgraded.
Need to get 340 kB of archives.
```

Clonez maintenant les sources de l'outil **Buildroot** dans votre dossier de personnel :

```
cd ~/Documents
git clone https://github.com/buildroot/buildroot.git
```

```
Processing triggers for man-db (2.9.3-2) ...
Help ubuntu@servertest:~$ cd ~/Documents
ubuntu@servertest:~/Documents$ git clone https://github.com/buildroot/buildroot.git
Cloning into 'buildroot'...
remote: Enumerating objects: 415124, done.
remote: Counting objects: 100% (2411/2411), done.
remote: Compressing objects: 100% (1023/1023), done.
Receiving objects: 14% (58118/415124), 18.59 MiB | 1.83 MiB/s
```

## 6.2 Construction d'un système de base

Vous pouvez maintenant générer votre première image minimale de GNU/Linux à partir de **Buildroot**.

Comme nous ne disposons pas d'une carte Raspberry, nous allons utiliser QEMU pour l'émuler

### 6.2.1 Installation de QEMU

```
sudo apt-get install qemu-system-arm
```

```
ubuntu@servertest:~/Documents$ sudo apt-get install qemu-system-arm
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ibverbs-providers ipxe-qemu ipxe-qemu-256k-compatible-efi-roms libcacard0 libdaxctl1 libfdt1
  libibverbs1 libiscsi7 libndctl6 libpmem1 librados2 librbd1 librdmacm1 libsllp0
  libspice-server1 liburing1 libusbredirparser1 libvirglrenderer1 qemu-block-extra
  qemu-efi-aarch64 qemu-efi-arm qemu-system-common qemu-system-data qemu-system-gui
  qemu-utils sharutils
```

- Assurez vous que la version de qemudont vous disposez est bien supérieure à 2.4

```
Qemu-system-arm -version
```

```
ubuntu@servertest:~/Documents$ qemu-system-arm -version
QEMU emulator version 5.0.0 (Debian 1:5.0-5ubuntu9.6)
Copyright (c) 2003-2020 Fabrice Bellard and the QEMU Project developers
ubuntu@servertest:~/Documents$
```

- Listez l'ensemble des plateformes (machines) supportées par QEMU :

```
ubuntu@server:~/Documents$ qemu-system-arm -machine help
Supported machines are:
akita             Sharp SL-C1000 (Akita) PDA (PXA270)
ast2500-evb       Aspeed AST2500 EVB (ARM1176)
ast2600-evb       Aspeed AST2600 EVB (Cortex A7)
borzoi            Sharp SL-C3100 (Borzoi) PDA (PXA270)
canon-a1100       Canon PowerShot A1100 IS
cheetah           Palm Tungsten|E aka. Cheetah PDA (OMAP310)
collie            Sharp SL-5500 (Collie) PDA (SA-1110)
connex            Gumstix Connex (PXA255)
cubieboard        cubietech cubieboard (Cortex-A8)
emcraft-sf2       SmartFusion2 SOM kit from Emcraft (M2S010)
highbank          Calxeda Highbank (ECX-1000)
lmx25-pdk          ARM i.MX25 PDK board (ARM926)
integratorcp      ARM Integrator/CP (ARM926EJ-S)
kzm               ARM KZM Emulation Baseboard (ARM1136)
lm3s6965evb       Stellaris LM3S6965EVb
lm3s811evb        Stellaris LM3S811EVb
mainstone         Mainstone II (PXA27x)
mclmx6ul-evk      Freescale i.MX6UL Evaluation Kit (Cortex A7)
mclmx7d-sabre     Freescale i.MX7 DUAL SABRE (Cortex A7)
microbit          BBC micro:bit
```

- Assurez vous de la prise charge de la plateforme Raspberry pi 2

```
qemu-system-arm -machine help
```

```
palmetto-bmc    OpenPOWER Palmetto BMC (ARM926EJ-S)
raspi2          Raspberry Pi 2B
realview-eb     ARM RealView Emulation Baseboard (ARM926EJ-S)
```

- Listez les processeurs supportés par l'émulateur pour cette plateforme.

```
qemu-system-arm -machine raspi2 -cpu help
```

- Assurez vous que le processeurs qui équipe le Raspberry pi 2B est bien supporté par l'émulateur

```
cortex-a15
cortex-a7
cortex-a8
```

### 6.2.2 Génération de la première image

- Prenez connaissance du fichier **README**, quelles sont les actions à mener pour construire un système ?

Déjà, il faudra être en super utilisateur pour construire un système et suivre ces instructions suivantes :

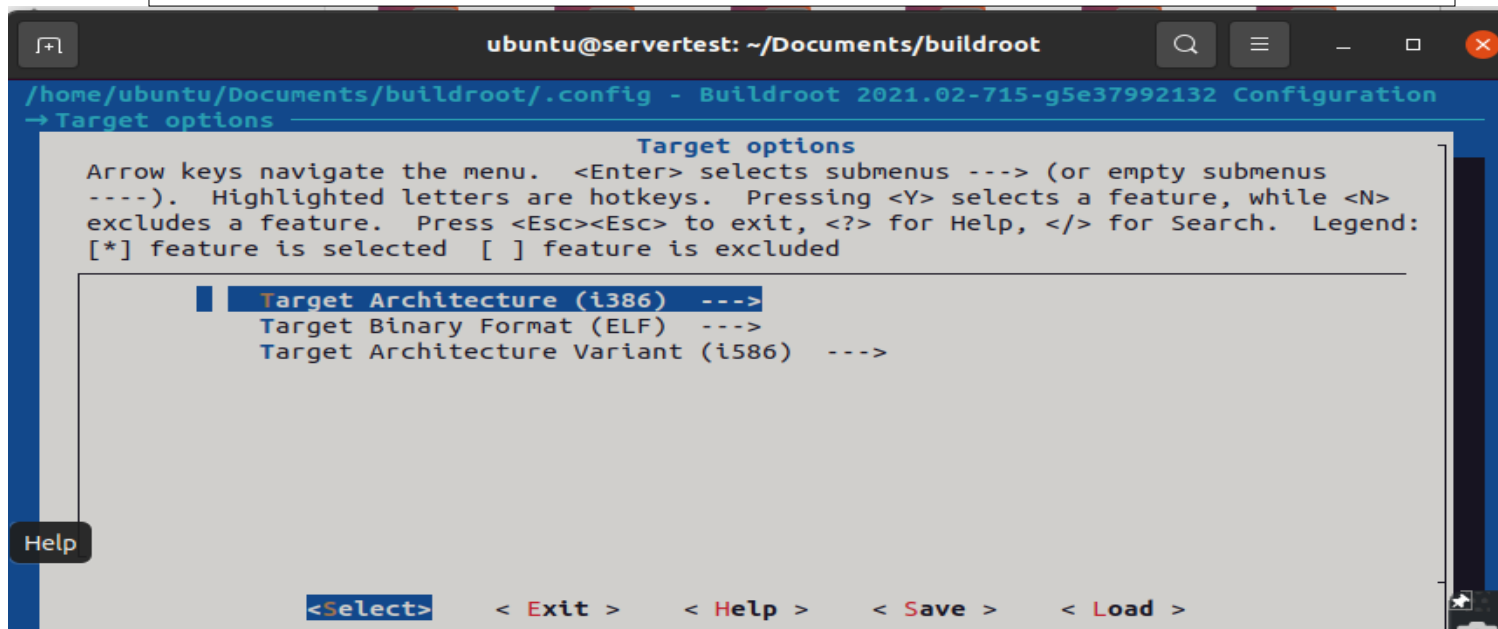
- 1) On lance la commande '**make menuconfig**'
- 2) On sélectionne l'architecture cible et les packages qu'on souhaite compiler.
- 3) On lance la commande '**make**'
- 4) On patiente un peu la fin de la construction .
- 5) On cherche le noyau, le bootloader, le filesystem racine, etc. dans output/images

D'une manière plus large on peut définir les étapes de construction d'un système comme suit :

Installer des bibliothèques de développement

1. Préparer le dossier cible
2. Créer un dossier de travail et définir une chaîne d'outils
3. Créer et configurer une racine système
4. Télécharger Qt
5. Configurer Qt pour la compilation croisée
6. Compiler, installer et déployer Qt
7. Configurer Qt Creator pour la compilation croisée Raspberry Pi
  - Consultez le menu de configuration de buildroot par défaut, notamment **Target options**.

```
~/Documents/buildroot$ make menuconfig
```



- Buildroot contient des fichiers de pré-configuration pour un grand nombre de plateformes. Prenez connaissance du fichier de définition de la configuration pour une cible Raspberry Pi 2. Les fichiers de configuration standard des différentes cibles supportées par Buildroot se trouvent dans le dossier **configs**. matériel.
- Appliquez le fichier de configuration standard pour le Raspberry Pi 2 :

```
~/Documents/buildroot$ make raspberrypi2_defconfig
```

```
ubuntu@servertest:~/Documents/buildroot$ make raspberrypi2_defconfig
mkdir -p /home/ubuntu/Documents/buildroot/output/build/buildroot-config/lxdialog
PKG_CONFIG_PATH="" make CC="/usr/bin/gcc" HOSTCC="/usr/bin/gcc" \
  obj=/home/ubuntu/Documents/buildroot/output/build/buildroot-config -C support/kconfig -
  f Makefile.br conf
/usr/bin/gcc -D_DEFAULT_SOURCE -D_XOPEN_SOURCE=600 -DCURSES_LOC="ncurses.h" -DNCURSES_WID
ECHAR=1 -DLOCALE -I/home/ubuntu/Documents/buildroot/output/build/buildroot-config -DCONFIG
elp "\" /home/ubuntu/Documents/buildroot/output/build/buildroot-config/conf.o /home/ubuntu
/Documents/buildroot/output/build/buildroot-config/zconf.tab.o -o /home/ubuntu/Documents/b
uildroot/output/build/buildroot-config/conf
#
# configuration written to /home/ubuntu/Documents/buildroot/.config
```

- Observez la création du fichier **.config**.
- Consultez le fichier **readme.txt** du dossier **board/raspberrypi2**.
- Les différentes étapes à mener pour compiler :

La toute première étape est de créer une configuration.

- 1) On configure **Buildroot** : pour notre cas raspberry pi model 2 B « **make raspberrypi2\_defconfig** »
- 2) on construit le file system racine ( rootfs) « **make** ».

La configuration make ( make, make menuconfig ..etc ) permet de:

- onfigurer, construire et installer la chaîne d'outils de compilation croisée, ou simplement importer une chaîne d'outils externe;
- configurer, construire et installer les packages cibles sélectionnés;
- construire une image du noyau, si elle est sélectionnée;
- créer une image du chargeur de démarrage, si elle est sélectionnée;
- créer un système de fichiers racine dans des formats sélectionnés

- 3) Il reste maintenant de déployer le système sur une carte micro-SD

BuildRoot est un projet en développement rapide et de nouvelles fonctionnalités sont souvent ajoutées.

BuildRoot a une interface de configuration similaire à l'interface de configuration du noyau Linux. Pour lancer l'interface de configuration

```
~/Documents/buildroot$ make
```

```
ubuntu@servertest:~/Documents/buildroot$ make qemu_arm_versatile_defconfig
#
# configuration written to /home/ubuntu/Documents/buildroot/.config
#
ubuntu@servertest:~/Documents/buildroot$
```



- Lancez la compilation du système (vous devez être connecté à internet) :

```
~/Documents/buildroot$ make
```

```
#
ubuntu@servertest:~/Documents/buildroot$ make
/usr/bin/make -j1 O=/home/ubuntu/Documents/buildroot/output HOSTCC="/usr/bin/gcc" HOSTCXX="/usr/bin/g++" synccfg
>>> linux-headers 5.10.7 Downloading
wget --passive-ftp -nd -t 3 -O '/home/ubuntu/Documents/buildroot/output/build/.linux-5.10.7'
"8193; 24377; 40901; 57343"
Allocating group tables: done
Writing inode tables: done
Copying files into the device: done
Writing superblocks and filesystem accounting information: done

ln -snf /home/ubuntu/Documents/buildroot/output/host/arm-buildroot-linux-uclibcgnueabi/sysroot /home/ubuntu/Documents/buildroot/output/staging
>>> Executing post-image script board/qemu/post-image.sh
ubuntu@server:~/Documents/buildroot$
```

- À l'issue de la génération, vous obtenez 2 fichiers dans le dossier output/images/. Le noyau correspond au fichier **zImage** et le système racine au fichier **rootfs.ext2**

```
~/Documents/buildroot$ cd output/images/
~/Documents/buildroot$ ls -l
```

```
ubuntu@server:~/Documents/buildroot$ cd output/images/
ubuntu@server:~/Documents/buildroot/output/images$ ls -l
total 7412
-rw-r--r-- 1 ubuntu ubuntu 62914560 Apr 21 20:41 rootfs.ext2
-rwxr-xr-x 1 ubuntu ubuntu 454 Apr 21 20:41 start-qemu.sh
-rwxr-xr-x 1 ubuntu ubuntu 8880 Apr 21 20:41 versatile-pb.dtb
-rw-r--r-- 1 ubuntu ubuntu 2871896 Apr 21 20:41 zImage
ubuntu@server:~/Documents/buildroot/output/images$
```

- Quelle est la taille du noyau et du système de fichier (en Mo).  
**62.91456 Mo pour rootfs.ext2 et 2.871896 Mo pour zImage**
- Démarrez le système dans **QEMU** à l'aide du script généré par **Buildroot** :

```
~/Documents/buildroot/output/images$ ./start-qemu.sh
```

- Testez votre système et commentez les éléments suivant:
  - Estimez la durée de boot : **15 seconds**
  - Bannière d'accueil, mot de passe root
  - Allure du prompt, disposition du clavier, # (AZERTY )
  - Connexion réseau, connexion à distance par ssh

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:34:56
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:3456/64  Scope:Link
          inet6 addr: fec0::5054:ff:fe12:3456/64  Scope:Site
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1344 (1.3 KiB)  TX bytes:1360 (1.3 KiB)
```

```
# ping google.fr
PING google.fr (216.58.204.131): 56 data bytes
64 bytes from 216.58.204.131: seq=0 ttl=255 time=83.600 ms
64 bytes from 216.58.204.131: seq=1 ttl=255 time=10.740 ms
```

- Identifiez la version du noyau linux installé.

```
# uname -a
Linux buildroot 5.10.7 #1 Wed Apr 21 20:32:17 EDT 2021 armv5tejl GNU/Linux
#
```

- Consultez le script /etc/init.d/rcS

```
ubuntu@server: ~/Documents/buildroot/out...  x  ubuntu@server: ~/Documents/buildroot/out...  x
#!/bin/sh

# Start all init scripts in /etc/init.d
# executing them in numerical order.
#
for i in /etc/init.d/S??* ;do

    # Ignore dangling symlinks (if any).
    [ ! -f "$i" ] && continue

    in
        # Source shell script for speed.
        (
            trap - INT QUIT TSTP
            set start
            . $i
        )
    ;;
*)
    # No sh extension, so fork subprocess.
    $i start
- /etc/init.d/rcS 1/27 3%
```

### Processus de démarrage du système :

Lorsqu'un ordinateur x86 est démarré, le processeur regarde la fin de la mémoire système pour le BIOS (Basic Input / Output System) et l'exécute. Le programme BIOS est écrit dans une mémoire permanente en lecture seule et est toujours disponible pour utilisation. Le BIOS fournit l'interface de niveau le plus bas aux périphériques et contrôle la première étape du processus de démarrage.

Le BIOS teste le système, recherche et vérifie les périphériques, puis recherche un lecteur à utiliser pour démarrer le système. Habituellement, il vérifie le lecteur de disquette (ou le lecteur de CD-ROM sur de nombreux systèmes plus récents) pour un support de démarrage, le cas échéant, puis il regarde le disque dur. L'ordre des lecteurs utilisés pour le démarrage est généralement contrôlé par un paramètre BIOS particulier du système. Une fois Linux installé sur le disque dur d'un système, le BIOS recherche un Master Boot Record (MBR) à partir du premier secteur du premier disque dur, charge son contenu en mémoire, puis lui transmet le contrôle.



Ce MBR contient des instructions sur la façon de charger le chargeur de démarrage GRUB (ou LILO), en utilisant un système d'exploitation présélectionné. Le MBR charge ensuite le chargeur de démarrage, qui prend en charge le processus (si le chargeur de démarrage est installé dans le MBR). Dans la configuration par défaut de Red Hat Linux, GRUB utilise les paramètres du MBR pour afficher les options de démarrage dans un menu. Une fois que GRUB a reçu les instructions correctes pour que le système d'exploitation démarre, que ce soit à partir de sa ligne de commande ou du fichier de configuration, il trouve le fichier de démarrage nécessaire et confie le contrôle de la machine à ce système d'exploitation.

## 6.3 Amélioration du système

### 6.3.1 Prompt du shell global

Le système minimal présente un shell minimaliste dans lequel il n'est pas aisé de savoir où l'on se trouve dans le système de fichier et où l'utilisateur connecté n'est pas identifié clairement. On sait juste s'il s'agit ou non de root (\$ ou #).

Pour configurer le prompt du shell afin qu'il indique clairement l'utilisateur connecté et le dossier courant, il faut modifier le fichier **/etc/profile**. Le prompt du shell d'un terminal y est représenté par la variable **PS** ou le numéro du terminal. Ici, nous n'avons qu'un seul terminal configuré : **PS1**

Pour obtenir un prompt de type **user@hostname : directory**

```
if [ "$PS1" ]; then
    export PS1="\u@\h:\w "
    if [ "`id -u`" -eq 0 ]; then
        export PS1=${PS1}"# "
    else
        export PS1=${PS1}"$ "
    fi
fi
```

- Modifiez le fichier **/etc/profile** comme ci-dessus.

```
ubuntu@server: ~/Documents/buildroot/out... x ubuntu@server: ~/Documents/buildroot/out... x
export PATH="/bin:/sbin:/usr/bin:/usr/sbin"

if [ "$PS1" ]; then
    export PS1="\u@\h:\w "
    if [ "`id -u`" -eq 0 ]; then
        export PS1=${PS1}"#"
    else
        export PS1=${PS1}"$ "
    fi
fi

export EDITOR="/bin/vi"
```

- Redémarrer la cible et observez l'allure du prompt quand vous changez de dossier.

```
Welcome to Buildroot
buildroot login: root
root@buildroot:~ #vi /etc/profile
root@buildroot:~ #
root@buildroot:~ #
```

- Expliquez le test **if [ "`id -u`" -eq 0 ]; then ...** du script.

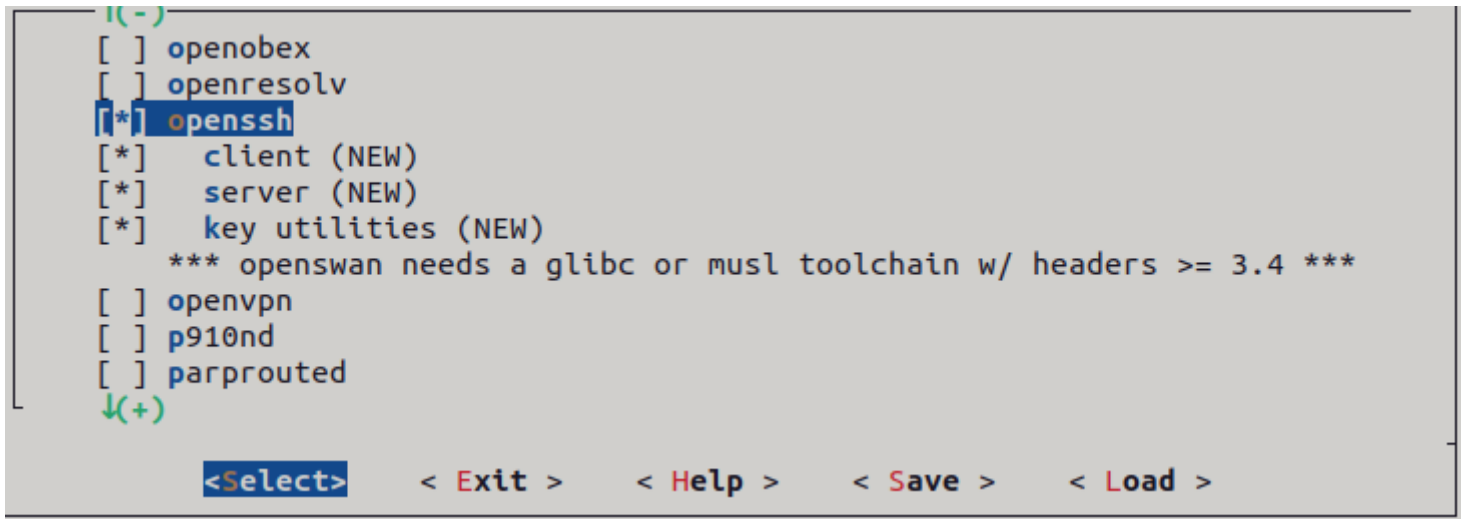
Cette ligne de code est une exécution de commande « **id -u** » test de l'ID user s'il est à 0 (init) ça veut dire utilisateur principal ou **root** donc on exécute l'instruction **export PS1=\${PS1}"#" "** qui nous permet d'avoir le prompt **# ( root )**

**Sinon** on exécute l'instruction **export PS1=\${PS1}"\$ " "** un utilisateur normal.

### 6.3.2 Accès à distance

La prise en main à distance d'un système est en général assuré au moyen d'un serveur ssh. Vous pourrez utiliser au choix le package dropbear ou openSSH. Nous utiliserons ici **openSSH**:

**make menuconfig** → Target packages → Networking applications  
→ openssh



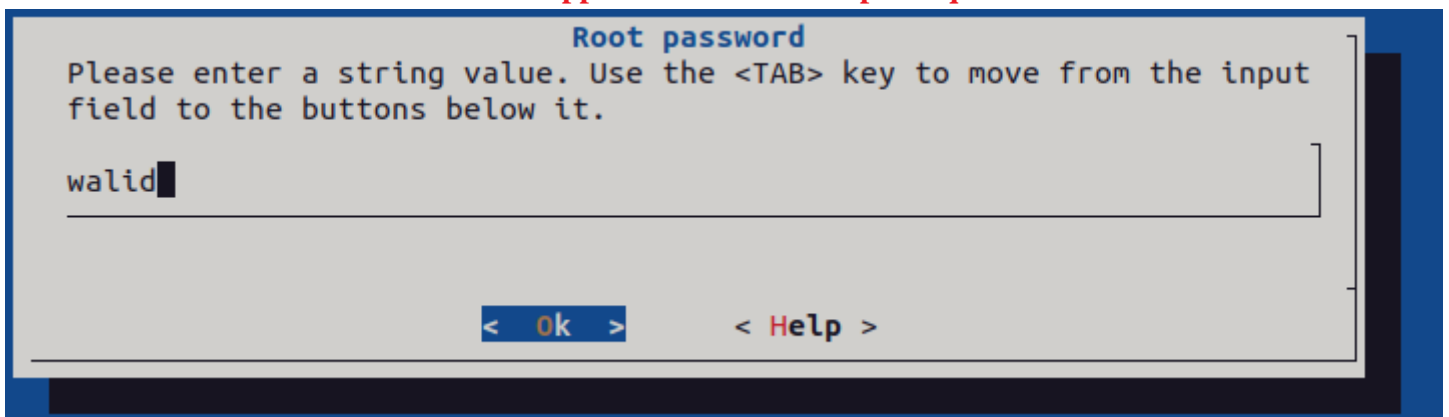
```
(-)  
[ ] openobex  
[ ] openresolv  
[*] openssh  
[*]   client (NEW)  
[*]   server (NEW)  
[*]   key utilities (NEW)  
*** openswan needs a glibc or musl toolchain w/ headers >= 3.4 ***  
[ ] openvpn  
[ ] p910nd  
[ ] parprouted  
↓(+)  
  
<Select>  < Exit >  < Help >  < Save >  < Load >
```

Dans les deux cas, il faut au préalable sécuriser l'accès local au système en créant un mot de passe pour root et un au moins un utilisateur local.

- Sécurisez l'accès au système en attribuant un mot de passe à root:

**make menuconfig -> System configuration -> Root password**

**Assurez vous de bien noter dans votre rapport de TP le mot de passe que vous avez attribué à root**



```
Root password  
Please enter a string value. Use the <TAB> key to move from the input  
field to the buttons below it.  
  
valid  
  
< ok >  < Help >
```

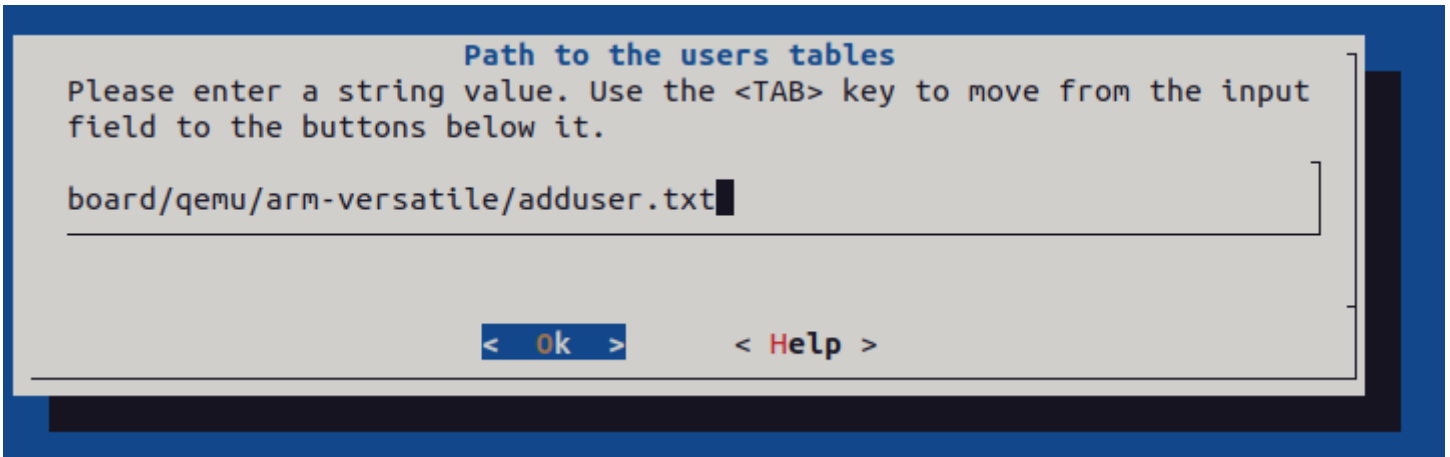
La création de comptes d'utilisateurs locaux se fait au moyen d'un fichier texte qui répond à la syntaxe de Makeusers définie dans le **chapitre 24** de la documentation de Buildroot :

<https://buildroot.org/downloads/manual/manual.html>

```
ubuntu@server:~/Documents/buildroot/board/qemu/arm-versatile$ cat adduser.txt
admin -1 admin -1 =admin /home/admin /bin/sh -admin user
ubuntu@server:~/Documents/buildroot/board/qemu/arm-versatile$
```

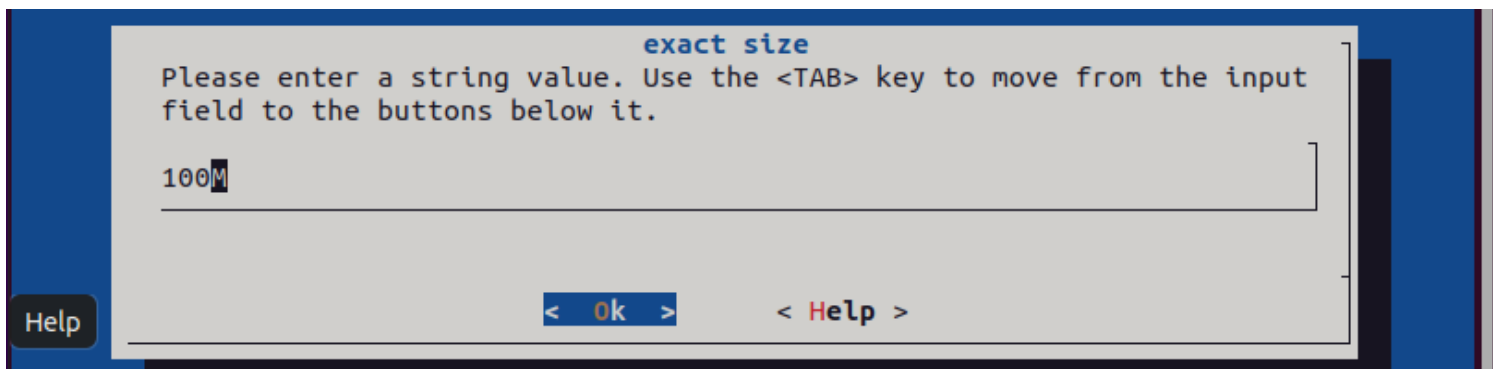
Indiquez à Buildroot la précédente sence de ce fichier pour créer notre utilisateur conformément au paragraphe **9.6. Adding custom user accounts** de la documentation de Buildroot :

**make menuconfig** → **System configuration** → **Path to the users tables**



- Modifiez la taille exact du système de fichier à 100Mo

**make menuconfig** → **Filesystem Image** → **exact size**



- Relancez le processus de construction

```
Allocating group tables: done
Writing inode tables: done
Copying files into the device: done
Writing superblocks and filesystem accounting information: done

>>> Executing post-image script board/qemu/post-image.sh
ubuntu@server:~/Documents/buildroot$
ubuntu@server:~/Documents/buildroot/output/images$ ls -l
total 15560
-rw-r--r-- 1 ubuntu ubuntu 104857600 Apr 23 12:30 rootfs.ext2
-rwxr-xr-x 1 ubuntu ubuntu    454 Apr 23 12:30 start-qemu.sh
-rwxr-xr-x 1 ubuntu ubuntu   8880 Apr 21 20:41 versatile-pb.dtb
-rw-r--r-- 1 ubuntu ubuntu  2871896 Apr 21 20:41 zImage
ubuntu@server:~/Documents/buildroot/output/images$
```

Redémarrez la cible : Il est nécessaire ici de modifier le script de démarrage de la cible pour pouvoir accéder au service ssh depuis votre hôte en utilisant une redirection de port. Copiez le script start-qemu.sh et renommez le start-qemu-net.sh

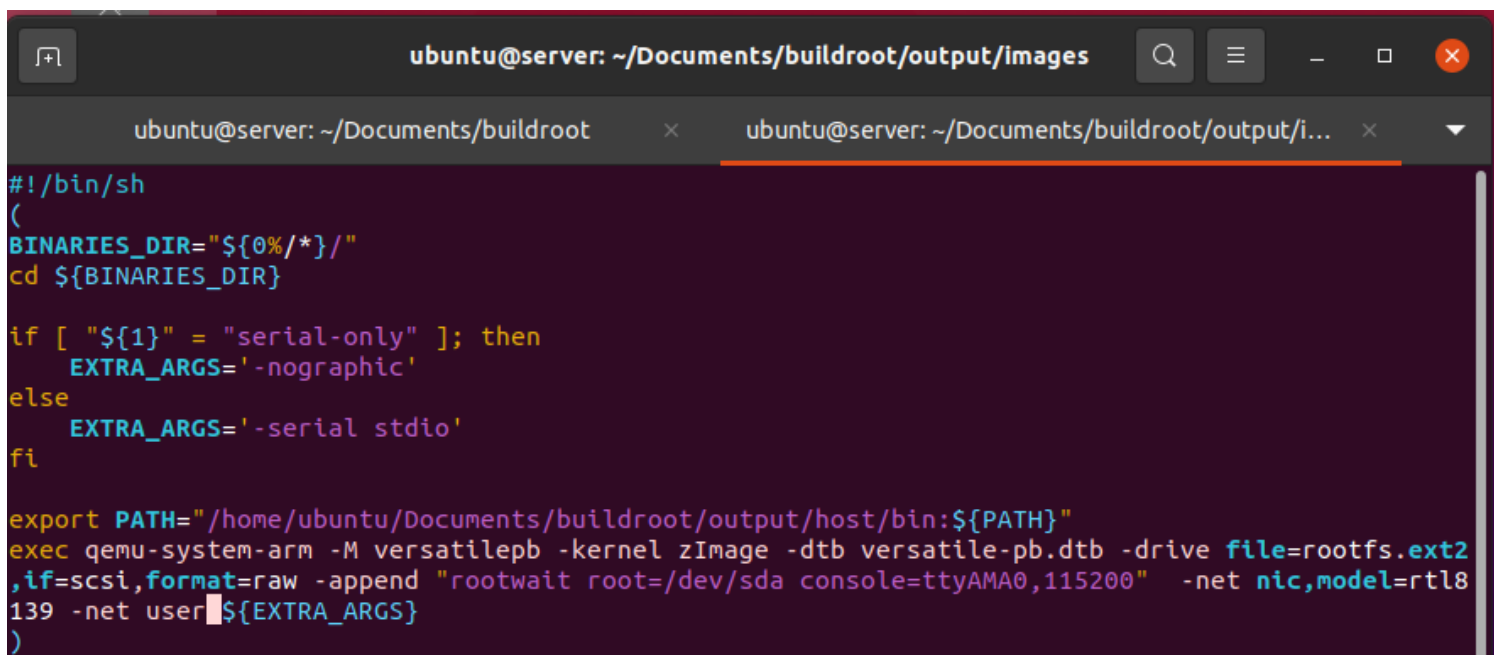
```
~/Documents/buildroot/output/images$ cp start-qemu.sh start-qemu-net.sh
```

- Modifiez le script start-qemu-net.sh comme ci-dessous :

```
#!/bin/sh
IMAGE_DIR="${0%/*}/"

if [ "${1}" = "serial-only" ]; then
    EXTRA_ARGS='-nographic'
else
    EXTRA_ARGS='-serial stdio'
fi

export PATH="/home/marco-virt/Documents/buildroot/output/host/bin:${PATH}"
exec qemu-system-arm \
    -M versatilepb -kernel ${IMAGE_DIR}/zImage \
    -dtb ${IMAGE_DIR}/versatile-pb.dtb \
    -drive file=${IMAGE_DIR}/rootfs.ext2,if=scsi,format=raw \
    -append "rootwait root=/dev/sda console=ttyAMA0,115200" \
    -net nic,model=rtl8139 -net user,hostfwd=tcp::5555-:22 \
    ${EXTRA_ARGS}
```



The screenshot shows a terminal window with a dark background and light-colored text. The window title is 'ubuntu@server: ~/Documents/buildroot/output/images'. The terminal content shows the modified script, which is identical to the one in the code block above. The script sets the IMAGE\_DIR, handles command-line arguments for serial-only or stdio output, sets the PATH, and then executes qemu-system-arm with various options for kernel, dtb, drive, and network configuration.



```

1 #!/bin/sh
2 (
3 IMAGE_DIR="${0%/*}/"
4
5 if [ "${1}" = "serial-only" ];
6
7     then EXTRA_ARGS='-nographic'
8 else
9     EXTRA_ARGS='-serial stdio'
10 fi
11
12 export PATH="/home/ubuntu/Documents/buildroot/output/host/bin:${PATH}"
13 exec qemu-system-arm -M versatilepb -kernel ${IMAGE_DIR}/zImage -dtb ${IMAGE_DIR}/versatile-
    pb.dtb -drive file=${IMAGE_DIR}/rootfs.ext2,if=scsi,format=raw -append "rootwait root=/dev/sda
    console=ttyAMA0,115200" -net nic,model=rtl8139 -net user,hostfwd=tcp::5555-:22 ${EXTRA_ARGS}
14 )

```

- Observez les effets de la reconstruction sur les modifications réalisées antérieurement (le prompt ?).

La taille du noyau est devenue plus grande que la première fois.

Temps de démarrage est plus long a mis 50 seconds, le deuxième environs 30 seconds.

Assurez vous de la nécessité de fournir le bon mot de passe root pour accéder au système.

```

Welcome to Buildroot
buildroot login: admin
Password:
$
$ █

```

```

Welcome to Buildroot
buildroot login: admin
Password:
admin@buildroot:~ $ █

```

Accédez à la cible depuis votre PC hôte via une session ssh. Assurez-vous que root ne puisse pas y accéder directement.

```
ssh-admin@localhost -p 5555
```

### Compte root

```

Rhythmbox User Name: ubuntu
Password: ubuntu (sudo su -)
root@localhost's password:
Permission denied, please try again.
root@localhost's password: █

```

### Compte admin

```
ubuntu@server:~$ sudo ssh admin@localhost -p 5555
The authenticity of host '[localhost]:5555 ([127.0.0.1]:5555)' can't be established.
ECDSA key fingerprint is SHA256:rVl2pWmnXwJcm9ddcOWXwea/7vCSOEoMQdNBLM0nITY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '[localhost]:5555' (ECDSA) to the list of known hosts
.
admin@localhost's password:
$
```

Acceptez la clé de chiffrement. Il sera nécessaire de la supprimer lors des futures tentatives de connexion après chaque reconstruction du système.

**Pourquoi root ne devrait jamais pouvoir ouvrir une connexion distante directement.**

Root ne peut pas accéder à distance parce que root est l'utilisateur par défaut dans le système et son accès se fait directement dans le logging normal,

On a besoin de personnaliser le fichier **sshd\_config** sur la carte avec "PermitRootLogin yes" défini sur capable de ssh en tant qu'utilisateur root.

et les utilisateurs autorisés à accéder sont définis dans le fichier **adduser.txt** et si on autorise un accès à distance.

### 6.3.3 Persistance des modifications sur rootfs

Les modifications réalisées directement sur cible à partir de l'hôte fonctionnent très bien pour un système unique et final. Cependant, lors la prochaine reconstruction du système, ces changements auront disparu car ils ne sont pas présents dans la configuration de buildroot.

Il est important que le processus de construction reste entièrement reproductible, si l'on veut s'assurer que la prochaine version inclura les configurations personnalisées.

Pour ce faire, le plus simple est d'utiliser le mécanisme de recouvrement du rootfs de Buildroot (overlay mechanism).

Cette substitution de fichiers est spécifique au projet en cours. Il faut donc créer un répertoire personnalisé pour ce projet dans la source de buildroot pour la cible visée dans le dossier :

**board/<manufacturer>/<boardname>/fs-overlay/** Créez le dossier de recouvrement fs-

```
~/Documents/buildroot $ cd board/qemu/arm-versatile/
~/Documents/buildroot/board/qemu/arm-versatile $ mkdir fs-overlay
```

overlay et copiez-y le script précédents assurant la bonne configuration du prompt.

Fichier profile

```

ubuntu@server: ~/Documents/buildroot/...  ×  ubuntu@server: ~/Documents/buildroot/...  ×  ▾
Terminal ATH="/bin:/sbin:/usr/bin:/usr/sbin"

if [ "$PS1" ]; then
    export PS1="\u@\h:\w "
    if [ "`id -u`" -eq 0 ]; then
        export PS1=${PS1}"# "
    else
        export PS1=${PS1}"$ "
    fi
fi
export EDITOR='/bin/vi'
# Source configuration files from /etc/profile.d
for i in /etc/profile.d/*.sh ; do
    if [ -r "$i" ]; then
        . $i
    fi
done
unset i

```

Contenu du dossier fs-factory

```

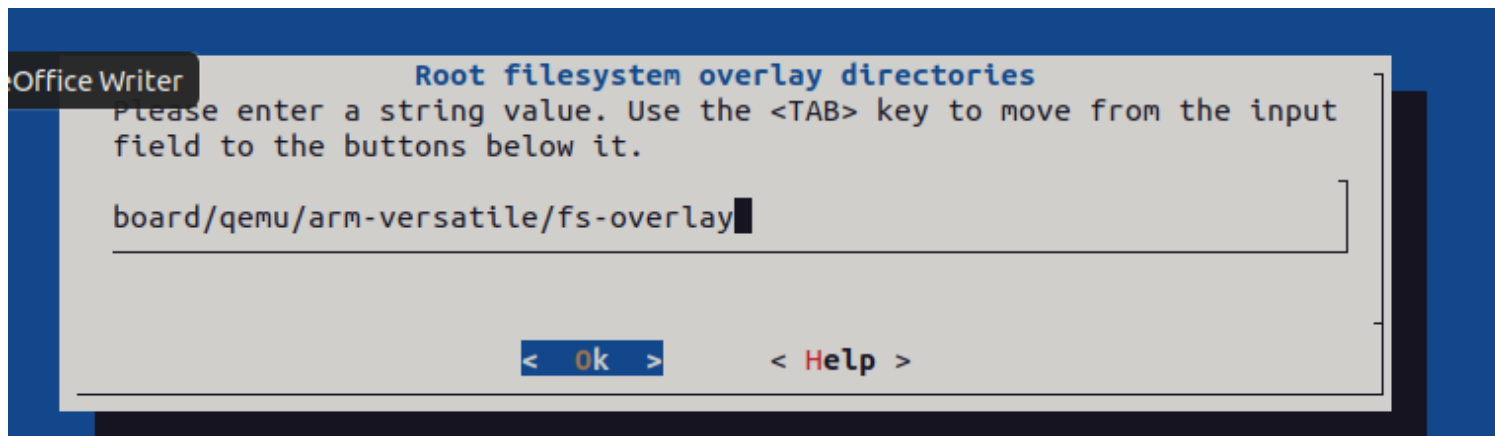
ubuntu@server:~/Documents/buildroot/board/qemu/arm-versatile$ tree
.
├── adduser.txt
├── fs-overlay
│   └── etc
│       └── profile
├── linux.config
├── linux-nommu.config
├── patches
│   └── linux
│       └── versatile-nommu.patch
└── readme.txt

4 directories, 6 files

```

- Indiquez la présence de ce dossier à **buildroot**.

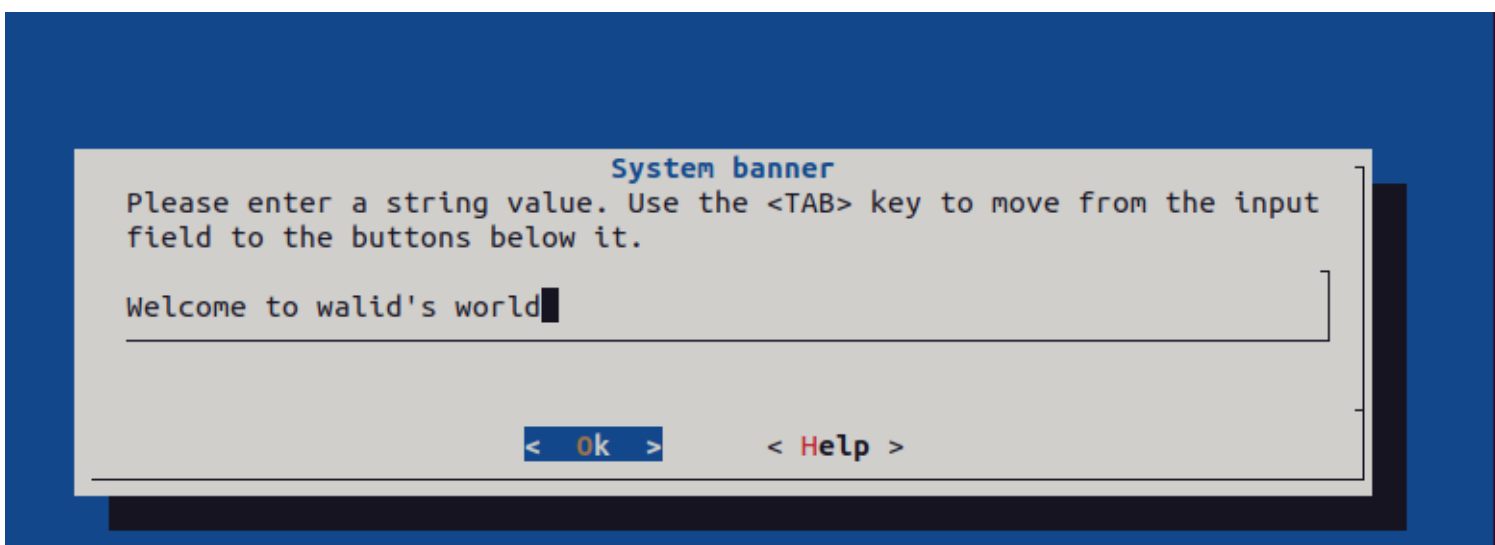
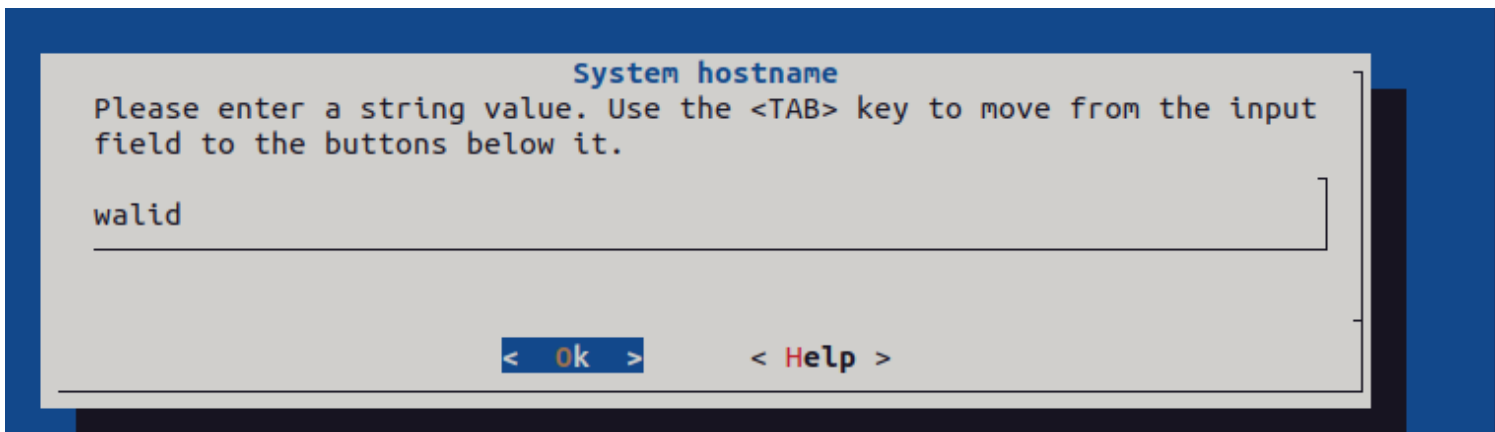
make menuconfig → System configuration → Root filesystem overlay directories



- Ajoutez au système un nom d'hôte et une bannière d'accueil.

make menuconfig → System configuration → System hostname

→ System banner



- Relancez le processus de construction.
- Redémarrez la cible et observez les modifications réalisées.

```
echo "PRETTY_NAME=\"Buildroot 2021.05-git\"" \
) > /home/ubuntu/Documents/buildroot/output/target/usr/lib/os-release
ln -sf ../usr/lib/os-release /home/ubuntu/Documents/buildroot/output/etc
>>> Sanitizing RPATH in target tree
PER_PACKAGE_DIR=/home/ubuntu/Documents/buildroot/output/per-package /home/ubuntu/Documents/buildroot/support/scripts/fix-rpath target
>>> Copying overlay board/qemu/arm-versatile/fs-overlay
touch /home/ubuntu/Documents/buildroot/output/target/usr
>>> Generating root filesystems common tables
```

```
Starting sshd: OK
```

```
Welcome to walid's world
walid login: admin
Password:
admin@walid:~ $
```

### 6.3.4 Serveur web

Plusieurs serveur web sont disponible dans buildroot : Apache, lighttpd, nginx. Nous choisisons ici d'installer nginx :

make menuconfig → Target packages → Networking applications  
→ nginx

```
[ ] mtr
*** nftables needs a toolchain w/ wchar, headers >= 3.12 ***
[*] nginx --->
[ ] ngircd
```

- Relancez le processus de construction.

```
make[1]: Leaving directory '/home/ubuntu/Documents/buildroot/output/build/nginx-1.18.0'
>>> nginx 1.18.0 Installing to target
PATH="/home/ubuntu/Documents/buildroot/output/host/bin:/home/ubuntu/Documents/buildroot/o
```

Modifiez le script de démarrage start-gemu-net.sh pour ajouter une redirection du port 80 vers le port 8080

```
-net nic,model=rtl8139 -net user,hostfwd=tcp::5555-:22,hostfwd=tcp::8080-:80 \
```

```
#!/bin/sh
(
IMAGE_DIR="${0%*/}/*"

if [ "${1}" = "serial-only" ];
then EXTRA_ARGS='-nographic'
else
EXTRA_ARGS='-serial stdio'
fi

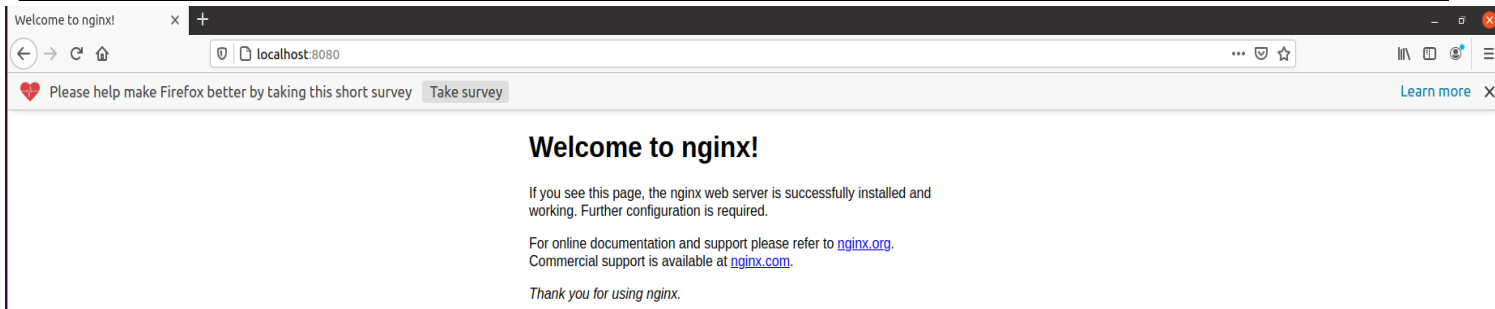
export PATH="/home/ubuntu/Documents/buildroot/output/host/bin:${PATH}"
exec qemu-system-arm -M versatilepb -kernel ${IMAGE_DIR}/zImage -dtb ${IMAGE_DIR}/versatile-pb.dtb -drive file=${IMAGE_DIR}/rootfs.ext2,if=scsi,format=raw -append "rootwait root=/dev/sda console=ttyAMA0,115200" -net nic,model=rtl8139 -net user,hostfwd=tcp::5555-:22,hostfwd=tcp::8080-:80 ${EXTRA_ARGS}
)
```

Redémarrez la cible accédez au service web depuis votre hôte en utilisant l'URL

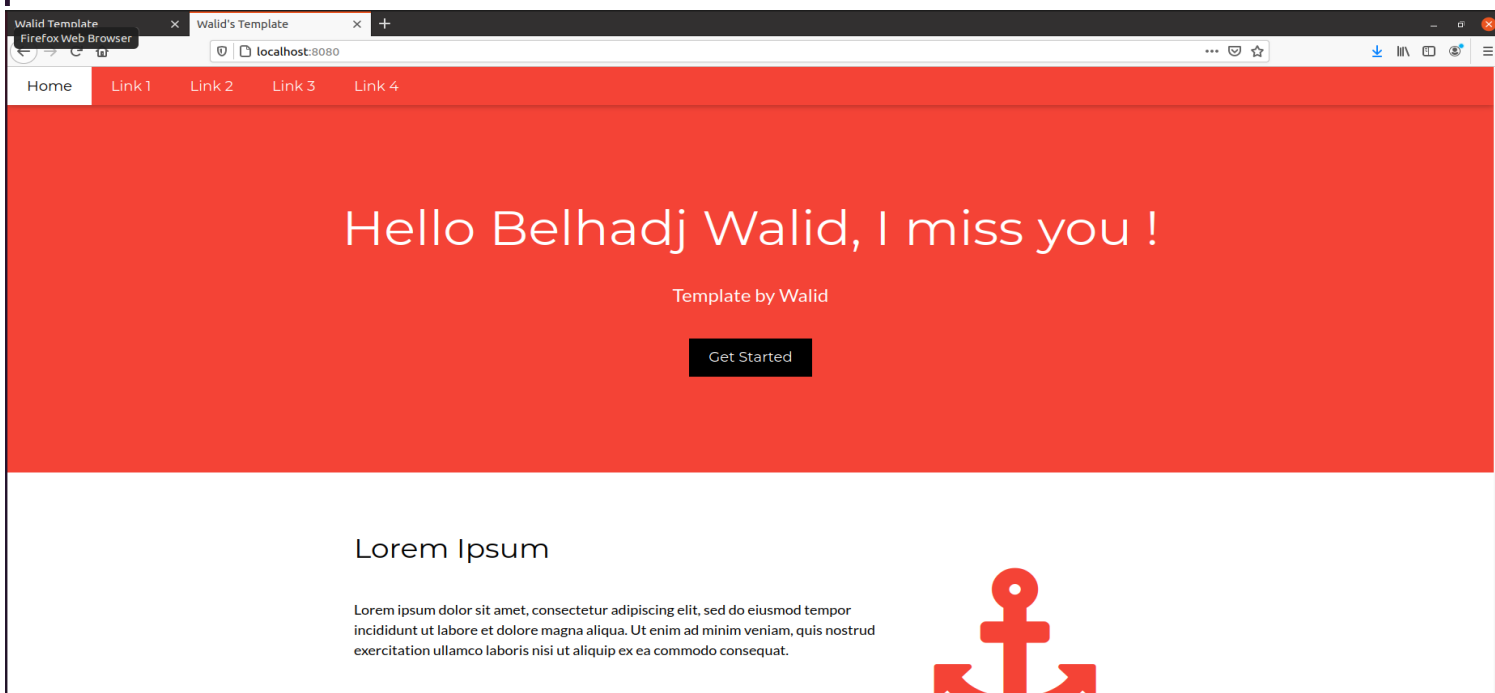
<http://localhost:8080>

```
Starting nginx...
random: crng init done
ssh-keygen: generating new host keys: RSA DSA ECDSA ED25519
Starting sshd: OK

Welcome to walid's world
walid login: admin
Password:
admin@walid:~ $
```



- Le dossier racine du serveur web se trouve dans /usr/html. Proposez une page index.html qui contient vos nom et prénom.



Modifiez l'arborescence du dossier fs-overlay pour que la prochaine reconstruction contienne votre page web



```
ubuntu@server:~/Documents/buildroot/board/qemu/arm-versatile$ tree
```

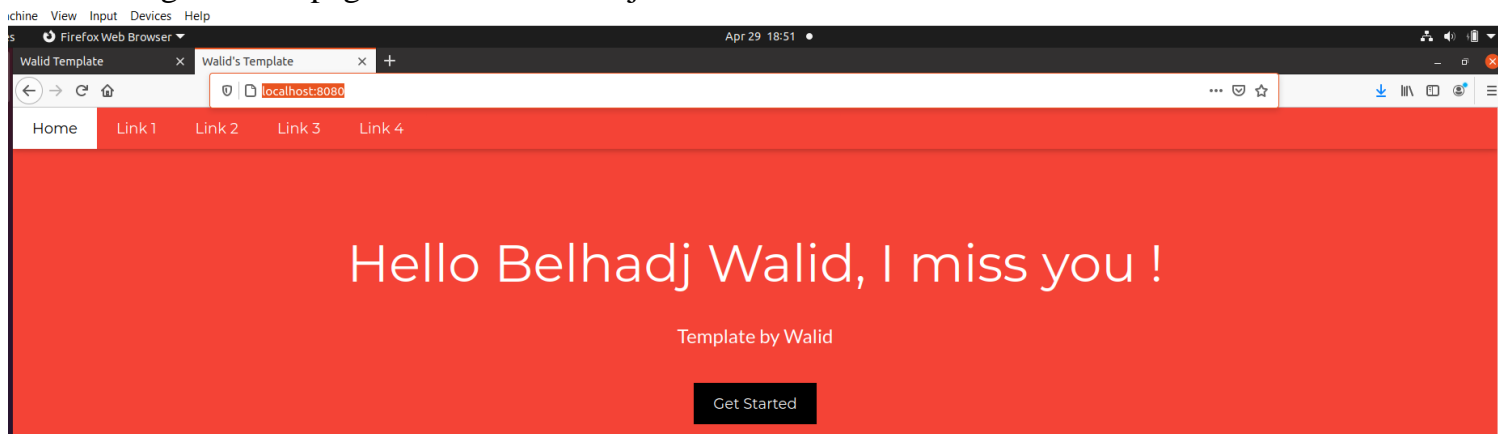
```
.
├── adduser.txt
├── fs-overlay
│   ├── etc
│   │   └── profile
│   ├── usr
│   │   └── html
│   │       └── index.html
├── linux.config
├── linux-nommu.config
├── patches
│   └── linux
│       └── versatile-nommu.patch
└── readme.txt
```

```
6 directories, 7 files
```

**make**

```
ln -sf ../usr/lib/os-release /home/ubuntu/Documents/buildroot/output/target/etc
>>> Sanitizing RPATH in target tree
PER_PACKAGE_DIR=/home/ubuntu/Documents/buildroot/output/per-package /home/ubuntu/Documents/buildroot/support/scripts/fix-rpath target
>>> Copying overlay board/qemu/arm-versatile/fs-overlay
touch /home/ubuntu/Documents/buildroot/output/target/usr
>>> Generating root filesystems common tables
rm -rf /home/ubuntu/Documents/buildroot/output/build/buildroot-fs
```

Sur le navigateur : la page a été bien mise à jour



### 3 Image finale