

RAPPORT DE TRAVAUX PRATIQUES n°3

LO52 – A2017

Étudiants : Pierre ROMET – Guillaume PROST

Sommaire

Table des matières

Mise en place de l'environnement de travail.....	2
Configuration par défaut du noyau.....	2
Configuration par défaut.....	2
Chargement et première compilation.....	3
Configuration avancée du noyau.....	4
Modification de la configuration.....	4
Compilation avec configuration personnalisée.....	8
Annexe.....	9

Le présent rapport a pour but de restituer le travail qui a été réalisé lors du TP n°3 de LO52 durant le semestre d'automne 2017 à L'UTBM, et qui portait sur la configuration et la compilation de kernel destiné à une carte **Hikey Linaro**.

Mise en place de l'environnement de travail (Partie 3)

La première étape de ce TP était de récupérer les deux dossiers contenant toutes les sources nécessaires aux autres étapes du TP. Pour cela, nous avons utilisé les deux méthodes qui étaient à notre disposition, à savoir la récupération d'une VM sous Ubuntu et la récupération du contenu de deux repository Git, dont les liens étaient fournis dans le sujet du TP.

Les deux méthodes ont été utilisées, car l'une des personnes du groupe avait un **OS Windows** sur sa machine de travail, tandis que l'autre personne avait un **OS Linux**. Le premier a donc dû récupérer la VM Linux contenant les sources du TP et le deuxième a pu cloner les repository indiqués pour récupérer les sources.

Configuration par défaut du noyau (Partie 4)

Configuration par défaut

Ensuite, il nous a été demandé de trouver le fichier de configuration par défaut du noyau de la carte Hikey. Nous avons dans un premier temps pensé, à tort, que celle-ci se trouvait dans **hikey-linaro/android/configs/android-base.cfg** car le noyau était destiné à terme à un appareil Android. Cependant, après avoir discuté avec le professeur, nous avons finalement trouvé le bon fichier, qui était **hikey-linaro/arch/arm64/configs/hikey_defconfig**.

Cette configuration va être utilisée pour faire une compilation croisée du noyau, car compte tenu de la puissance de calcul de la carte Hikey Linaro, il est plus pratique de réaliser la compilation du noyau sur une machine plus puissante, mais qui ne possède pas le même environnement que la carte sur laquelle sera déployée le noyau compilé.

Avant d'effectuer une compilation de noyau, il faut au préalable valider différentes étapes :

- S'assurer de disposer de la toolchain pré-compilée. Dans le cas de ce TP, cette toolchain pré-compilée se trouvait dans le repository Git **aarch64-linux-android-4.9**

- Configurer les variables **CROSS_COMPILE** et **ARCH** selon une méthode au choix parmi celles-ci : les inscrire dans le makefile, les passer en paramètre de la commande **make** lors de la compilation ou bien les inscrire en tant que variable d'environnement.

C'est cette dernière méthode que nous avons choisi d'utiliser, car le changement des valeurs des variables d'environnement peut être fait très facilement grâce à un script shell. La figure ci-contre représente le contenu de notre script shell :

```
ARCH=arm64
CROSS_COMPILE=~ /TP3/aarch64-linux-android-4.9/bin/aarch64-linux-android-

export ARCH
export CROSS_COMPILE
```

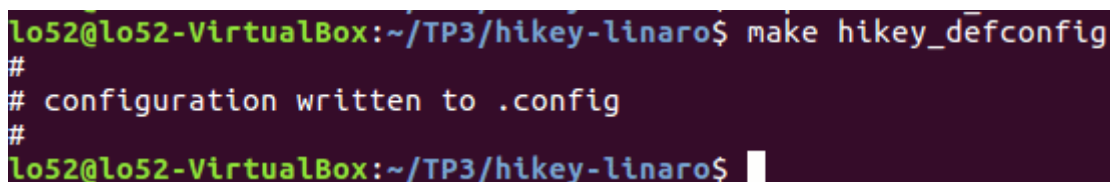
Illustration 1: Contenu du script shell permettant la modification des variables d'environnement

Les deux premières lignes permettent de définir des variables d'instance, et les deux instructions **export** permettent de les transformer en variables d'environnement. L'exécution d'un tel script se fait via un terminal de commande en exécutant l'instruction suivante : **./<nom_fichier>.sh**

Nous avons cependant rencontré une petite difficulté durant cette étape, puisqu'il nous a fallu attribuer la permission d'exécuter ce fichier à l'utilisateur courant (permissions 760).

Chargement et première compilation

Une fois l'environnement de compilation validé, nous sommes passé au chargement de cette configuration par défaut. Pour se faire, nous nous sommes placé avec un terminal dans le dossier **hikey-linaro** et nous avons effectué la commande **make arch/arm64/configs/hickey_defconfig**.

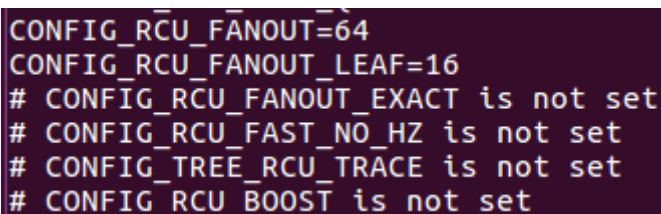


```
lo52@lo52-VirtualBox:~/TP3/hikey-linaro$ make hikey_defconfig
#
# configuration written to .config
#
lo52@lo52-VirtualBox:~/TP3/hikey-linaro$
```

Illustration 2: résultat de la commande "make hikey_defconfig"

Le chargement de la configuration a généré un fichier sans nom **.config** qui contient des instructions de configuration de la même forme que le fichier **hikey_defconfig.cfg**, mais en nombre beaucoup plus important, puisque le fichier de configuration initial faisait environ 420 lignes, tandis que le fichier **.config** fait plus de 4300 lignes.

Nous pensons qu'il contient toutes les instructions des fichiers de configuration présents dans le dossier **hikey-linaro** ainsi que ses sous-dossiers (à l'exception de ceux des architectures autres que arm64). Nous pouvons également noter que de nombreuses instructions de ce fichier **.config** sont de la forme **# <config_name> is not set**, c'est à dire commentées et suivies de **is not set** à la place d'une valeur.



```
CONFIG_RCU_FANOUT=64
CONFIG_RCU_FANOUT_LEAF=16
# CONFIG_RCU_FANOUT_EXACT is not set
# CONFIG_RCU_FAST_NO_HZ is not set
# CONFIG_TREE_RCU_TRACE is not set
# CONFIG_RCU_BOOST is not set
```

Nous avons ensuite lancé la configuration du noyau avec la commande **make**. Cette

Illustration 3: exemple de paramètres commentés dans le fichier ".config"

configuration a duré un certain temps, temps pendant lequel nous avons laissé tourner une machine dans son coin. Durant ce laps de temps, nous avons créé sur le repository Git le dossier **origin** contenant les fichiers de configuration demandés, et nous avons commencé la suite du TP.

Configuration avancée du noyau (Partie 5)

Modification de la configuration

La partie 5 de ce TP nous demandait de modifier certains paramètres de la configuration du noyau grâce à l'outil **menuconfig**, accessible via la commande **make menuconfig**. L'outil se présente comme une liste de chaînes de caractères représentant une structure en arbre et nous permettant de nous y déplacer. **Menuconfig** propose aussi une recherche de chaîne de caractère parmi les noms des propriétés configurables, permettant de trouver relativement facilement les éléments à modifier.

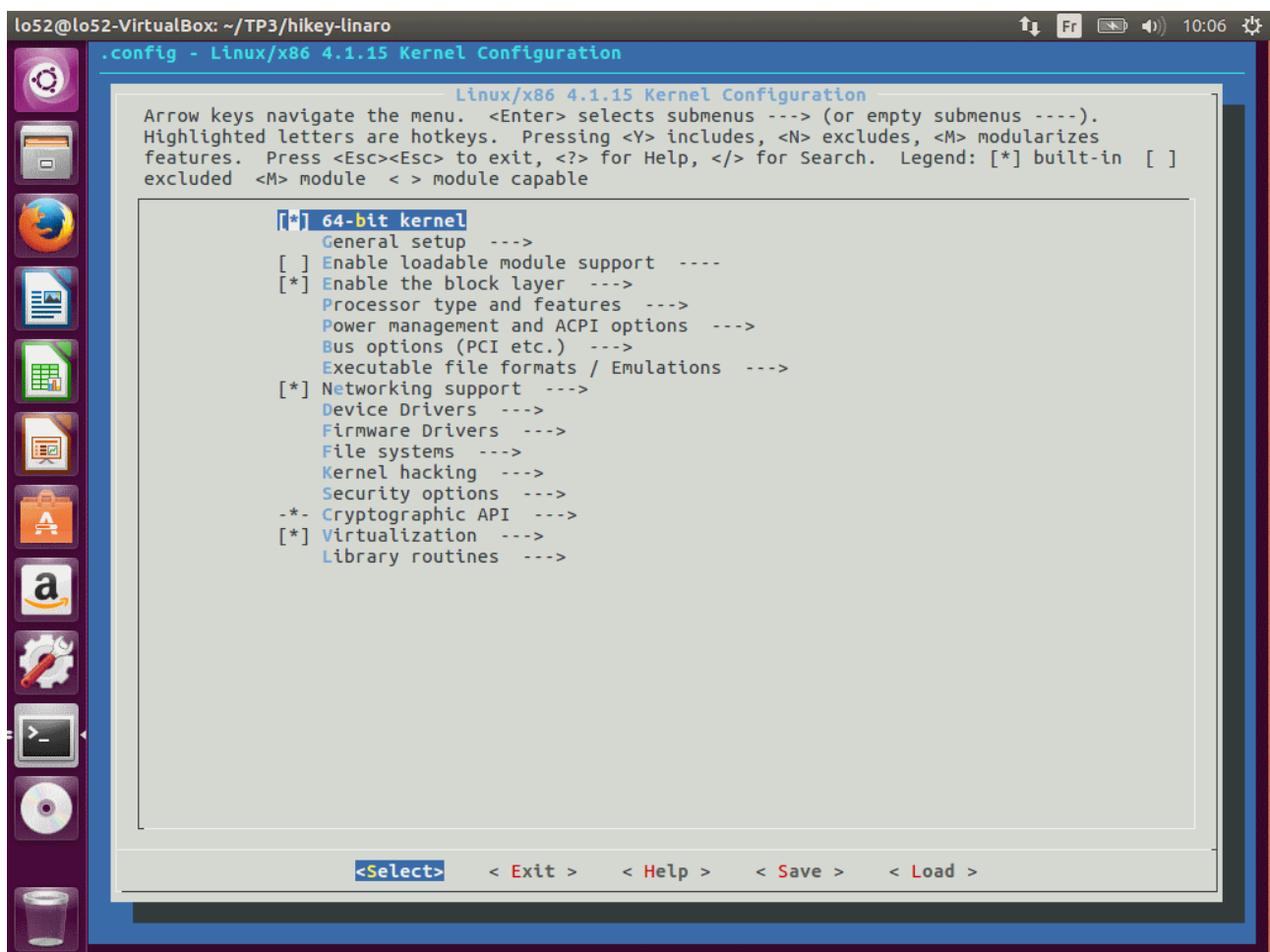


Illustration 4: Interface principale de l'outil menuconfig

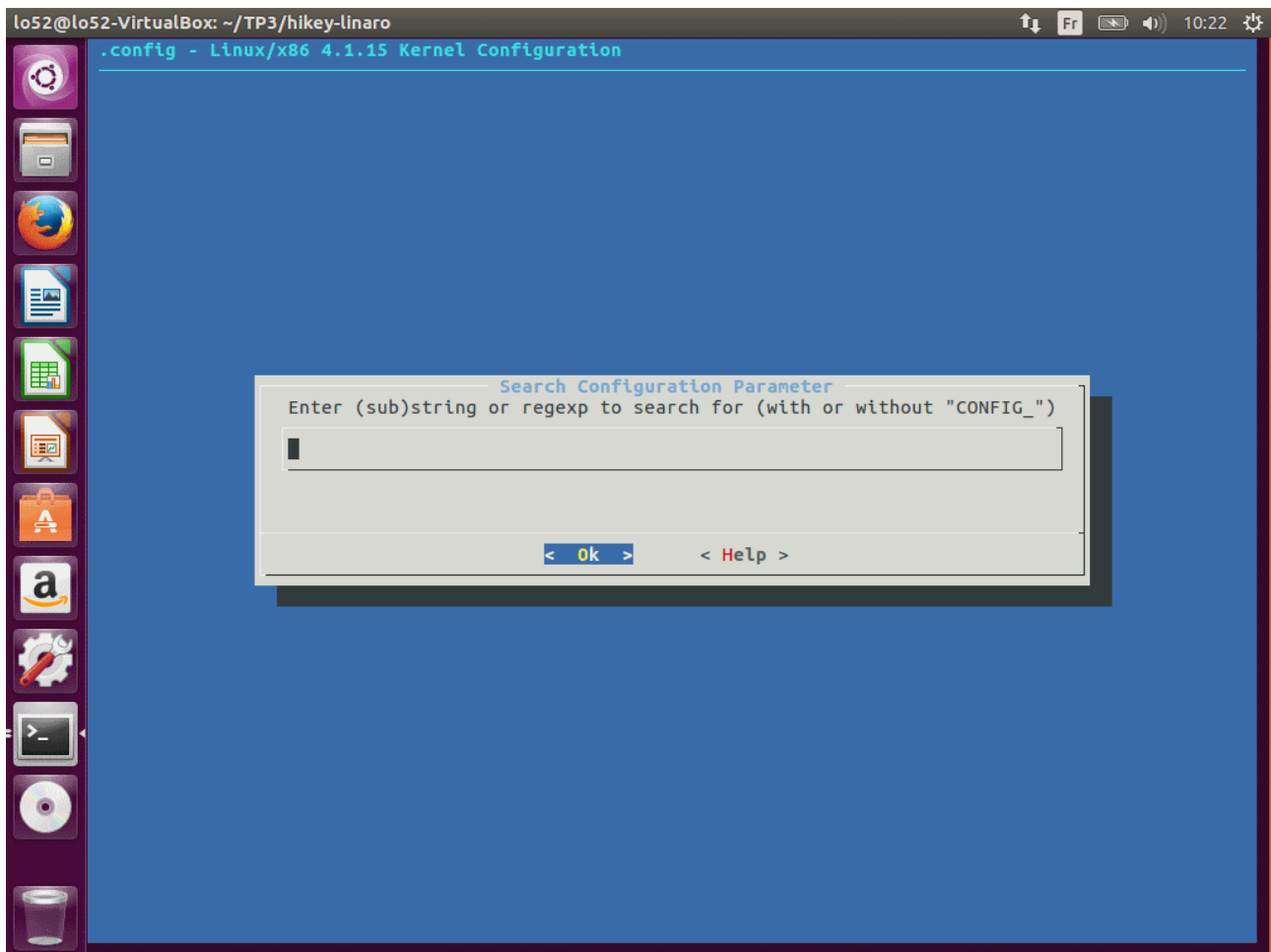


Illustration 5: Interface de recherche de paramètres de l'outil menuconfig

Le tableau suivant répertorie les différents paramètres que nous avons modifiés, avec leur emplacement dans **menuconfig** et la valeur que nous leur avons donné.

Nom du paramètre	Emplacement	Nouvelle valeur
NFC subsystem support	Networking support	Y
NFC Hardware simulator driver	Networking support -> NFC subsystem support -> Near Field Communication (NFC) Devices	Y
Intel PRO/Wireless 2200BG and 2915ABG Network Connection	Device Driver -> network device support-> Wireless LAN	Y
Maximum number of GPUs	Device Driver -> Graphic support -> VGA Arbitration	32
Bootup logo	Device Drivers -> Graphics Support	Y
Black and white linux logo	Device Drivers -> Graphics Support -> Bootup logo	Y
MTP gadget	Device Drivers -> USB support -> USB Gadget Support	N

USB LED driver support	Device Drivers -> USB support	Y
TI SOC drivers support	Device Drivers -> SOC (System On Chip) specific Drivers	Y

Nous avons effectué d'abord une recherche de tous les emplacements en nous répartissant les paramètres à rechercher (quand la compilation du kernel fut terminée), puis nous avons effectué les modifications ci-dessus. En plus de ces dernières, nous avons également modifié la valeur de la variable d'environnement **ARCH** de **arm64** à **ARMv8 Versatile**, comme demandé dans le sujet.

Le sujet demandait également de trouver la localisation de l'option relative aux wakelock ainsi que son nom. Cette option se trouve sous **Power Management and ACPI options** et se nomme **user space wakeup sources interface**.

Compilation avec configuration personnalisée

Une fois toutes ces modifications faites, nous avons lancé une compilation. Le fichier **.config** après la modification était plus long que celui utilisé lors de la première compilation du noyau (issu du fichier de configuration par défaut) : en effet, on note que le fichier **.config** a ce moment du TP fait 33 lignes de plus qu'au début, et que plusieurs des lignes correspondant aux paramètres que nous avons modifié avec **menuconfig** sont dé-commentés.

Cependant, nous n'observons pas de différences majeures entre les images compilées du noyau.

Une fois la compilation faite, nous avons défini notre configuration personnalisée comme configuration par défaut, nous l'avons renommée et nous l'avons déplacé dans le dossier contenant les configurations pour l'architecture **arm64** en exécutant les deux commandes suivantes :

```
~/TP3/hikey-linaro$ make savedefconfig
~/TP3/hikey-linaro$ mv defconfig arch/arm64/configs/hikeylo52_defconf
```

Illustration 6: Commandes shell permettant la sauvegarde d'une configuration par défaut

Cela nous permet par la suite de faire un différentiel entre ce nouveau fichier et le fichier de configuration utilisé lors de la première compilation du noyau. Le résultat de ce différentiel se trouve ci-dessous, dans la partie **Annexe** de ce rapport.

Enfin, nous avons créé le dossier **final** contenant le fichier **hikeylo52_defconf** et le **.config** associé.

Annexe

```
diff hikeylo52_defcong hikey_defconfig
0a1
> CONFIG_LOCALVERSION_AUTO=y
181a183
> CONFIG_CFG80211_WEXT=y
190,196d191
< CONFIG_NFC=y
< CONFIG_NFC_DIGITAL=y
< CONFIG_NFC_NCI=y
< CONFIG_NFC_NCI_SPI=y
< CONFIG_NFC_HCI=y
< CONFIG_NFC_SHDLC=y
< CONFIG_NFC_SIM=y
243d237
< CONFIG_IPW2200=y
285d278
< CONFIG_VGA_ARB_MAX_GPUS=32
298,300d290
< CONFIG_LOGO=y
< # CONFIG_LOGO_LINUX_VGA16 is not set
< # CONFIG_LOGO_LINUX_CLUT224 is not set
321d310
< CONFIG_USB_LED=y
329a319,320
> CONFIG_USB_CONFIGFS_F_MTP=y
> CONFIG_USB_CONFIGFS_F_PTP=y
366d356
< CONFIG_SOC_TI=y
424a415
> CONFIG_CRYPTO_MICHAEL_MIC=y
432a424
> CONFIG_CRC_CCITT=y
```

Illustration 7: Résultat du différentiel entre les fichiers de configuration "hikeylo52_defcong" et "hikey_defconfig"