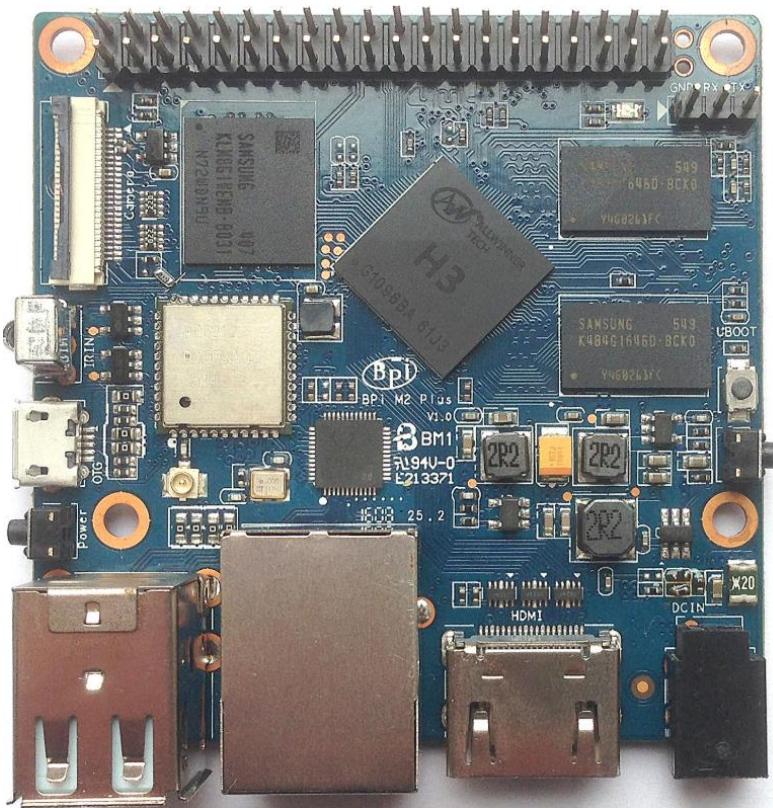


PARTIE 1 : PRISE EN MAIN ET ADMINISTRATION SYSTEME

IUT de Nice département GEII

Formation Linux Armbian

Bananapi M2+



Jean-Louis Salvat
17/03/2019

Table des matières

1	Introduction.....	4
2	Installation de l'OS sur la SD.....	6
2.1	Les versions disponibles d'OS	6
2.2	C'est quoi armbian ?	6
2.3	Installation sur la SD.....	7
3	Première connexion sur la cible	9
3.1	Introduction.....	9
3.2	C'est parti : utilisation du port COM USB OTG.....	9
3.3	Une autre solution : le port COM série de Debug.....	13
3.4	Dernière solution : connexion SSH vers la cible.....	15
3.5	Résumé des 3 manips.....	17
4	Configuration de la cible.....	18
4.1	Introduction.....	18
4.2	Première configuration	18
4.3	Changer le nom de la machine.....	21
4.3.1	Solution 1 : utilisation de l'outil de configuration du NetworkManager	21
4.3.2	Solution 2 : changer le fichier de configuration hostname et hosts dans /etc.....	23
4.3.3	Solution 3 : le script à tout faire armbian-config	24
4.4	Conclusion	24
5	Première connexion en wifi.....	25
5.1	Introduction.....	25
5.2	Utilisation de la commande nmtui (NetworkManager).....	26
5.3	Configuration du wifi sans nmtui	28
5.4	En cas de problème :	30
5.5	Conclusion	31
6	Comment copier des fichiers du pc host vers la cible ?	32
6.1	Introduction.....	32
6.2	Avec une clé USB	32
6.3	Winscp	34
6.4	Samba	35
6.4.1	Partage d'un répertoire de votre cible.....	37

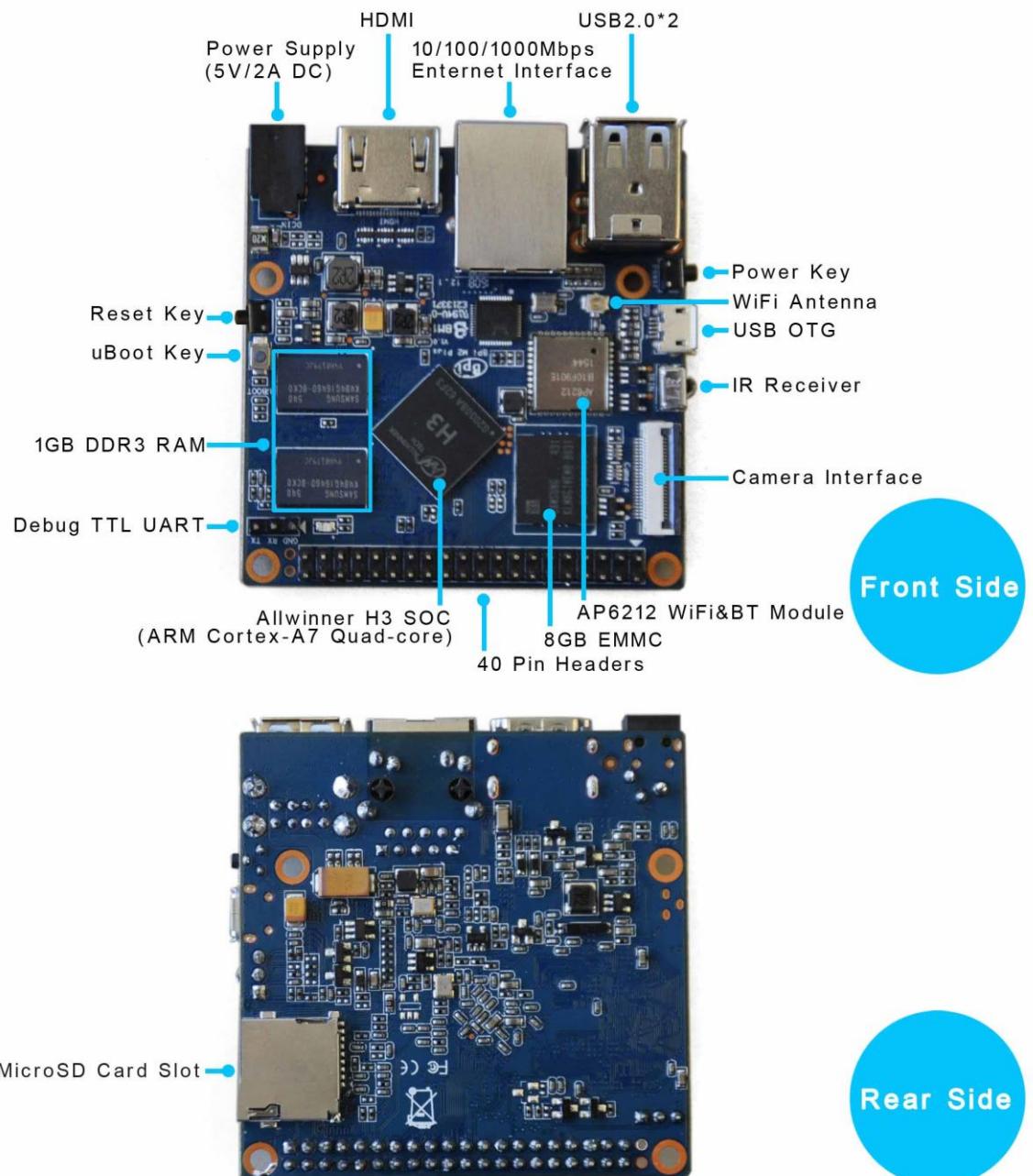
6.4.2	Ajouter les dossiers de partage.....	37
7	Avoir une interface graphique ?.....	39
7.1	Introduction.....	39
7.2	Deux solutions de test.....	39
7.3	Premiers tests.....	40
7.4	Manip1 : installation de codeblocks	43
7.5	Manip 2 : Installation de cheese	45
7.6	Manip 3 , changement de l'environnement graphique	45
7.7	Manip 4 : serveur graphique et clients graphiques	47
8	Exercices Linux utilisateurs et administration système.....	54
8.1	Introduction : le site https://linuxjourney.com/ /	54
8.2	Partie Getting started et Command Linux	54
8.3	Partie Text Fu	56
8.4	Partie user management.....	58
8.5	Permission	59
8.6	Processes.....	60
8.7	Packages	61
8.8	Device	62
8.9	File System.....	64
8.10	Récapitulatif et compléments	66
8.10.1	Résumé de commandes utilisateurs et systèmes	66
8.10.2	La notion de droits	69
8.10.3	Utilisation du clavier.....	69
8.10.4	Les redirection (> et >>) et les tunnels (ou pipes en anglais).....	70
8.10.5	Les répertoires.....	71
8.11	Manip 1 : gestion de la fréquence processeur	72
8.12	Manip 2 : utilisation de l'UART3	74
8.13	Manip 3 : le bootload u-boot	77
8.14	Pour aller plus loin	80
8.15	Kernel.....	83
8.16	Init.....	84
8.17	Manip 4 : Process Utilisation	85
8.18	Logging.....	87

8.19	Manip 5 : Utilisation d'une webcam	89
9	Manip bluetooth et HC05.....	94
9.1	Introduction.....	94
9.2	Le bluetooth	94
9.3	Premier test sous windows ou sous Android.....	95
9.4	Connexion avec le bananapi	97
9.5	En cas de problème	100
9.6	Pour aller plus loin.....	102
10	Création d'un point d'accès HotSpot.....	103
10.1	Solution 1 : configuration du hostspot à la main	104
10.2	Solution 2 : Hotspot utilisant un script.....	108
11	Compléments	109
11.1	Les systèmes de fichier.....	109
11.1.1	Introduction.....	109
11.1.2	Système de fichier FAT12 (FAT16 et FAT32)	109
11.1.3	Le système de fichier ext2 (ext3, ext4)	113
11.2	Compilation du noyau : les principes	117
11.3	Compilation du noyau	118
11.3.1	Introduction.....	118
11.3.2	Mais pourquoi compiler un noyau ?	119
11.3.3	Installation de la chaîne de développement croisée pour la cible :	119
11.3.4	Télécharger les sources du noyau	121
11.3.5	Configuration et compilation du noyau et des modules	121
11.3.6	Copie sur la cible	123
11.4	Compilation d'un module sur la cible.....	125
11.5	Uboot	132
11.6	uInitrd (ramdisk)	132
11.7	Boot du système	132

1 Introduction

Cette formation a pour but la prise en main d'un système embarqué sous OS Linux. Nous allons dans un premier temps faire connaissance avec l'OS Linux et les commandes de bases. La plupart des manipulations se feront sur la cible banana pi M2+ :

- ✓ Quad-core 1.2GHz Cortex-A7 H3
- ✓ 1GB DDR3
- ✓ 8GB eMMC onboard
- ✓ WiFi and BlueTooth onboard
- ✓ 10/100/1000Mbps Ethernet Port



Pour plus de détail sur la carte banana pi M2+ : <https://bananapi.gitbooks.io/bpi-m2-/content/en/>

Il existe de nombreuses cibles embarquées à OS Linux : citons la plus connue **Raspberry** mais aussi des cibles plus récentes comme **Orange Pi**, **Odroid**, **Beagle board**, **banana pi**. Toutes ces cartes sont alimentées en **5V** et consomment en moyenne entre **300 et 500mA** avec des pointes pouvant aller à **2A** lorsque la charge du processeur est importante. Le prix de ces cartes évolue entre **10 euros et 50 euros**.

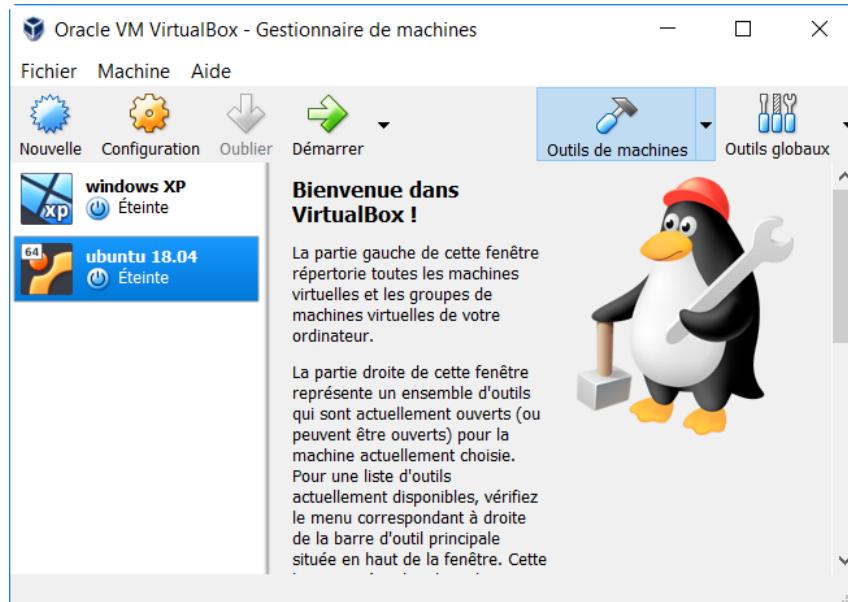
Les applications de ces cartes sont nombreuses :

- ✓ Domotique
- ✓ IOT
- ✓ Routeur
- ✓ TV Box
- ✓ Drones

L'intérêt de ces cartes par rapport à des systèmes embarqués sans OS est la connectivité qu'elles proposent (**bluetooth**, **Wifi**, **Ethernet**, **USB**, **SD**, **IR**, **Son**, **Video**) mais aussi leur bus externe (**GPIO**) permettant l'ajout de cartes (Hat ou chapeaux) d'extension permettant l'ajout d'un nombre important de fonctionnalités. On pourra aller sur le site <https://www.adafruit.com/category/286>.

Voici pour les avantages, l'inconvénient par rapport à un système embarqué de type arduino ou mbed est la consommation importante (impossible d'utiliser des piles pour l'alimentation) et la prise en main de la carte qui prend plus de temps puisqu'il est nécessaire de faire connaissance avec l'OS Linux. C'est ce que nous allons faire ici.

La plupart des manipulations que nous allons faire peuvent être faite sur une machine virtuelle Ubuntu. Pour installer une VM on pourra aller sur <https://fr.wikihow.com/installer-Ubuntu-sur-VirtualBox>.



Dans ce document, le host (c'est-à-dire le PC de travail) est sous Windows 10. Nous ne traiterons pas l'utilisation d'un host sous Linux, même si cela est en général plus simple.

2 Installation de l'OS sur la SD

Cette partie a déjà été réalisée, les bananapi sont fournis avec une carte SD de classe 10 de taille 16Go de type Sandisk ou Kingston.

2.1 Les versions disponibles d'OS

Il existe 2 versions de Linux installables sur le bananapi :

- ✓ Ubuntu (en version LTS c'est-à-dire Long Term Support)
- ✓ Debian

Ubuntu est une distribution basée sur les outils de la distribution historique Debian. Ces 2 distributions sont assez proches. La différence essentielle est la réactivité de Ubuntu qui met à jour ses distributions tous les 6 mois environ, alors que Debian fait des mises à jour tous les 2 ans en moyenne. Si l'on utilise les versions LTS d'ubuntu alors on retrouve le même temps de mise à jour des versions, avec l'avantage d'une maintenance plus longue (5 ans).

Versions d'Ubuntu ayant le label LTS			
Version	Nom de code de développement	Date de sortie	Date de fin de vie (EOL)
Ubuntu 18.04 LTS	<i>The Bionic Beaver</i>	26 avril 2018	Avril 2023 (rapporter un bug)
Ubuntu 16.04 LTS	<i>The Xenial Xerus</i>	21 avril 2016	Avril 2021 (rapporter un bug)
Ubuntu 14.04 LTS	<i>The Trusty Tahr</i>	17 avril 2014	Avril 2019 (rapporter un bug)

Voici les versions de Debian :

- [Debian 9 \(« Stretch »\)](#) — actuelle version stable ;
- [Debian 8 \(« Jessie »\)](#) — actuelle version oldstable ;
- [Debian 7 \(« Wheezy »\)](#) — version stable obsolète ;

2.2 C'est quoi armbian ?

Armbian est la version « *Lightweight Debian or Ubuntu based distribution specialized for ARM developing boards* ». Cette distribution est utilisée pour toutes les cibles Orange Pi et Banana Pi. Notons que pour les cartes Raspberry Pi une distribution spéciale Raspbian a été développée. Toutes ces distributions utilisent les outils Debian pour l'embarqué (<https://github.com/armbian/build>).

2.3 Installation sur la SD

L'image que nous avons choisi d'installer est la version de Debian Stretch .

- ✓ Télécharger l'image Stretch : <https://www.armbian.com/banana-pi-m2-plus/>

Armbian Stretch

mainline kernel 4.19.y

Server or light desktop usage scenarios.

Stable

SUPPORTED



Torrent download



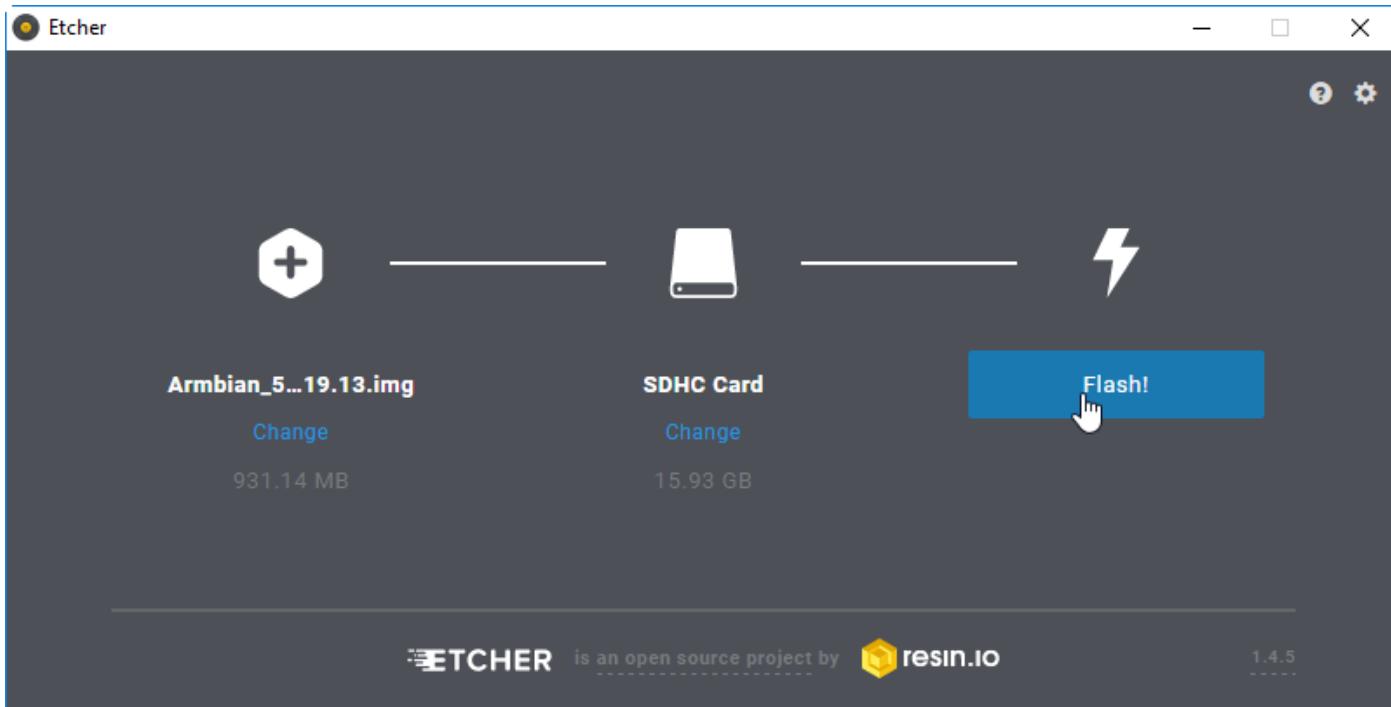
Direct download

L'outil permettant de copier l'image de la distribution est etcher

- ✓ Télécharger <https://www.balena.io/etcher/>



- ✓ Dézipper l'image Debian dans Téléchargement
- ✓ Puis lancer etcher
- ✓ Placer l'image dézippé de la distribution Stretch et insérer la SD Card



- ✓ Une fois l'image (qui fait un peu moins de 1Go) copiée, placer la SD sur la cible.

3 Première connexion sur la cible

3.1 Introduction

Il existe 5 solutions de connexion sur la cible :

- ✓ Utilisation d'un cable USB-A vers micro-USB (c'est la solution la plus simple)
- ✓ Utilisation d'un pont USB/TTL (FTDI) sur le connecteur DEBUG de la carte
- ✓ Utilisation d'un cable ethernet et connexion sur le réseau local
- ✓ Connexion en Wifi sur un point d'accès
- ✓ Connexion en bluetooth

Nous allons tester les 3 premières solutions et voir les avantages et inconvénients de chacunes.

Remarque : les 2 dernières solutions proposées (connexion en Wifi et bluetooth) nécessitent une configuration préalable, il ne reste donc que 3 solutions de connexion à la cible pour notre première connexion.

3.2 C'est parti : utilisation du port COM USB OTG

Pour cette manipulation vous aurez besoin d'un cable USB micro / USB A à connecteur sur le connecteur USB micro OTG de votre cible.



- ✓ Insérer la SDcard sur votre bananapi M2+. Cette SD card contient votre OS Linux ARMBIAN (distribution Debian Stretch) précédemment copié.

Votre cible possède 2 connecteurs USB 2.0 hosts (c'est-à-dire maître) et un connecteur USB micro OTG (qui peut être maître ou esclave) configuré en port com virtuel. C'est ce connecteur que nous allons utiliser, le PC va alors voir notre cible au travers d'une communication série virtuelle.

Nous avons aussi besoin d'une connexion ethernet pour faire les mises à jour de notre cible lors de la première connexion. Pour cette première manip vous n'aurez pas besoin d'avoir une connexion ethernet, ce cable peut donc ne pas être branché.



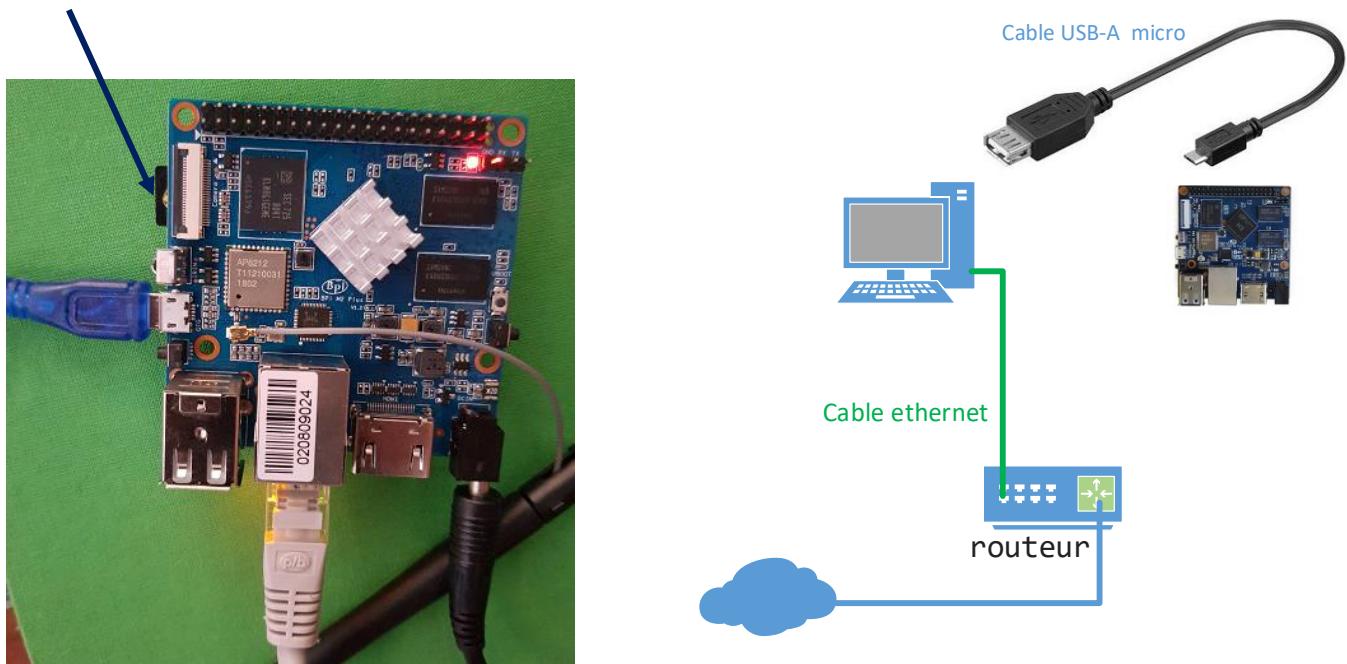


Figure 1 : connexion de la cible (cable ethernet + cable micro USB) et alimentation

✓ Connecter le cable USB sur le connecteur micro USB de la cible

✓ Brancher votre cible

Après une vingtaine de secondes, votre PC va installer le driver série de communication avec votre cible et vous devriez voir apparaître un port comm virtuel.

Remarque : vous aurez besoin d'un terminal pour vous connecter sur votre cible, il en existe un grand nombre, nous utiliserons pour notre part Tera Term ou putty qui sont gratuits. Il est préférable d'utiliser putty qui fournit un rendu visuel plus clair que Tera Term.

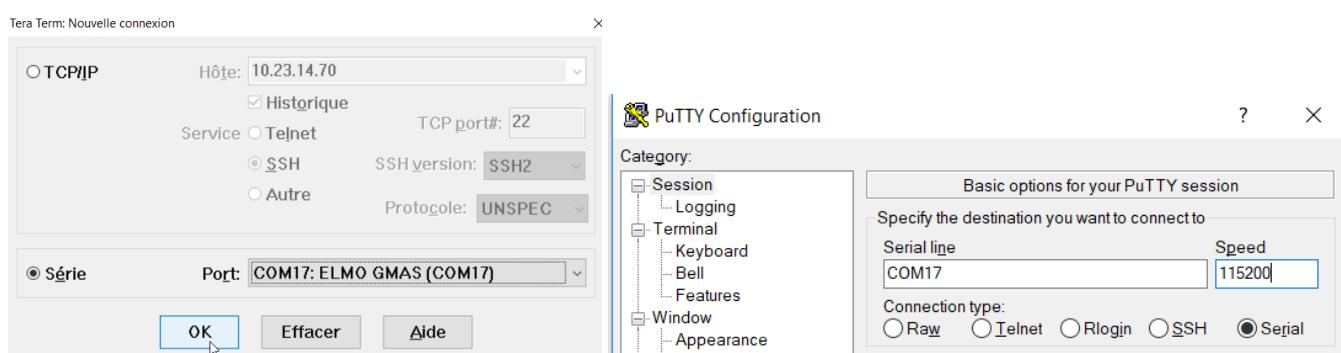


Figure 2 : connexion à notre cible via la connexion USB-micro (vue Tera Term et putty)

Vous aurez besoin de savoir si un port comm a été créé. Pour cela il suffit d'ouvrir le Gestionnaire de périphérique et de vérifier que le port com a été créé.

- ✓ Lancer la touche **Windows +X** pour accéder au **gestionnaire des périphériques** et vérifier le nom du comm créé (ic COM17)

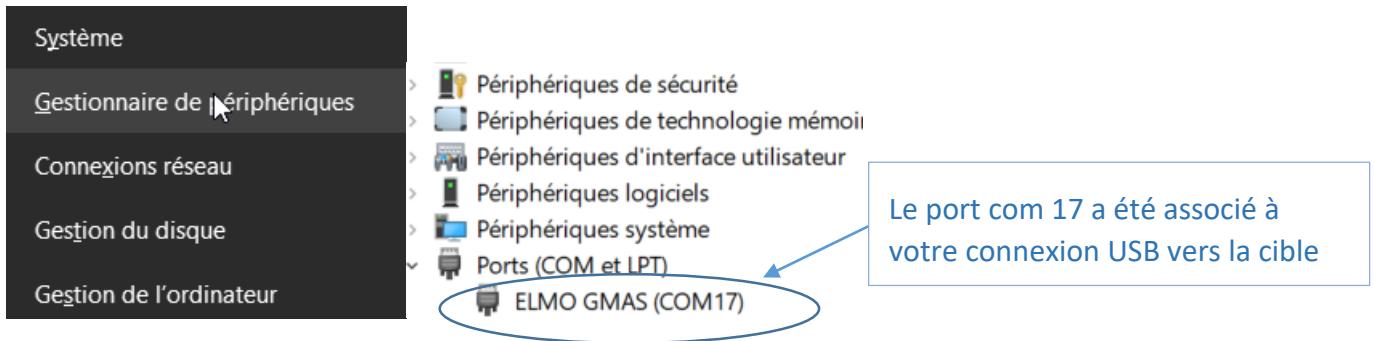
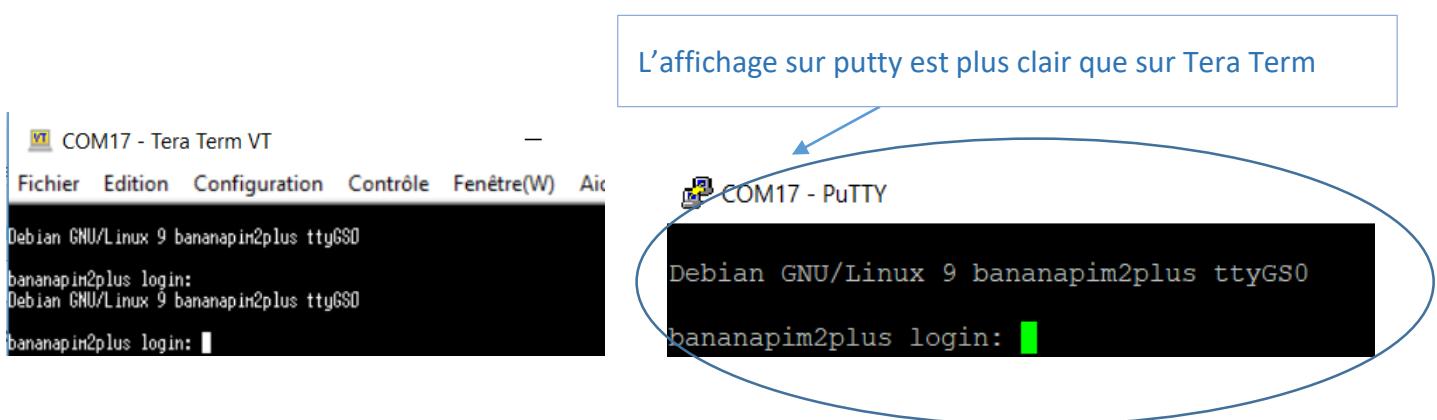


Figure 3 : vérification de la création du port COM après l'alimentation de la cible

La vitesse de communication comme le port com de DEBUG peut être configuré de 9600 bps à 115200bps, il sera préférable de choisir 115200bps pour avoir une communication plus rapide.

- ✓ Taper sur retour chariot pour faire apparaître la demande de login.



A la première utilisation le login est **root** et le mot de passe **1234** (voir la page 17 pour plus d'information). Si cette cible a déjà été configuré le login est **geii** et le mot de passe **geii** (**utilisateur sudoers configuré lors de la première connexion**).

En cas de problème : Si aucun port comm n'a été créé :

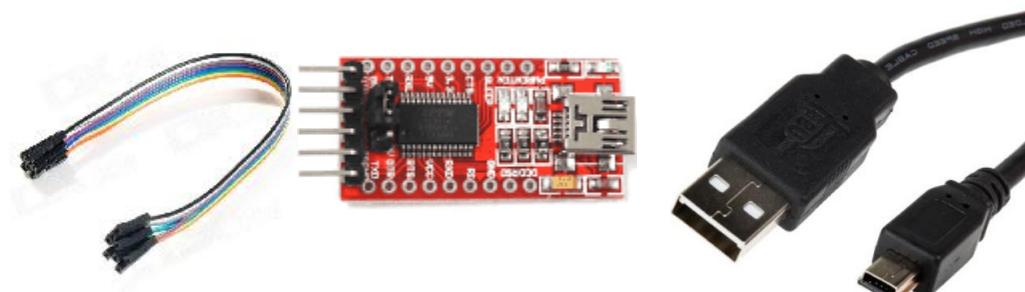
- ✓ Votre cable USB micro est de mauvaise qualité, tester avec un autre cable
- ✓ Votre SD card a été mal copiée ou n'est pas reconnue par uboot (le bootloader de la cible).
- ✓ Votre alimentation ne supporte pas les courants de pointes qui peuvent aller jusqu'à 2A, c'est cette dernière possibilité qui est la plus probable.

Dans tous les cas, il va falloir utiliser la solution port COM de DEBUG décrite juste après. C'est la seule façon de visualiser les messages du bootloader et du noyau linux (phase de démarrage) et donc de voir où est le problème.

3.3 Une autre solution : le port COM série de Debug

A la différence avec le port USB OTG du bananapi, le port Com de DEBUG existe pour toutes les cibles sous Linux (Raspberry , Orange PI, ...). Lorsque la cible n'a pas de connexion Ethernet ou Wifi c'est la seule possibilité de communication avec la cible qu'il reste. Cette communication bas-niveau permet de visualiser toutes les informations envoyées par le bootloader et Linux dès le démarrage de la cible. C'est ce que nous allons voir ici.

Il vous faudra pour cette manipulation un jeu de cable Dupont Femelle Femelle, un convertisseur usb/TTL FTDI et un cable mini-usb pour connecteur le FTDI à votre PC.



Cabler les broches TX /RX GND vers le FTDI, n'oubliez pas d'inverser le TX du FTDI vers le RX du bananapi

CON8 Pin Name	Default Function	GPIO
CON8 PO3	UART0-TXD	PH20
CON8 PO2	UART0-RXD	PH21
CON8 PO1	GND	

Figure 4 : tableau des numéros des 3 broches du COM DEBUG sur la cible

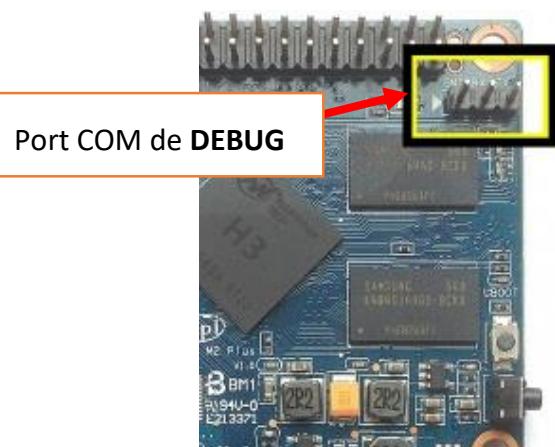




Figure 5 : Gestionnaire de périphérique, le cable USB micro et USB vers FTDI ont été branchés

Vous pouvez ensuite vous connecter avec putty ou teraterm (un port COM virtuel a du être crée dès que vous avez connecté le FTDI à votre PC, ici le COM3).

✓ *Rebooter votre cible afin de voir les messages au démarrage de la cible. Vitesse = 115200*

```

COM3 - PuTTY
U-Boot 2018.11-armbian (Feb 08 2019 - 09:09:41 +0100) Allwinner Technology

CPU: Allwinner H3 (SUN8I 1680)
Model: Banana Pi BPI-M2-Plus
DRAM: 1 GiB
MMC: SUNXI SD/MMC: 0, SUNXI SD/MMC: 1
Loading Environment from EXT4... ** File not found /boot/boot.env **

** Unable to read "/boot/boot.env" from mmc0:1 **
In: serial
Out: serial
Err: serial
Allwinner mUSB OTG (Peripheral)
Net: phy interface7
eth0: ethernet@1c30000
Warning: usb_ether using MAC address from ROM
, eth1: usb_ether
230454 bytes read in 25 ms (8.8 MiB/s)
starting USB...
USB0: USB EHCI 1.00
USB1: USB OHCI 1.0
USB2: USB EHCI 1.00

```

Figure 6 : extrait du boot de la cible après un reset

A la fin du process de lancement de votre OS, vous obtenez le même écran vous demandant de vous connecter sur la cible. La différence avec les autres types de connexion est donc la possibilité de voir les messages venant du bootloader (programme qui a la charge de lancer Linux) et le lancement de Linux. Par défaut le lancement du noyau et de votre distribution se fait en mode « quiet » c'est-à-dire sans message sur la console. Ce choix permet de démarrer votre cible plus rapidement, l'affichage des messages sur votre console à 115200bps ralenti le temps de boot.

Avant d'aller plus loin, nous allons voir la dernière solution de connexion à la cible : au travers une connexion SSH via le câble réseau.

3.4 Dernière solution : connexion SSH vers la cible

Pour cette dernière solution de connexion à la cible, vous aurez besoin de putty et d'un câble ethernet permettant de vous connecter à la cible.

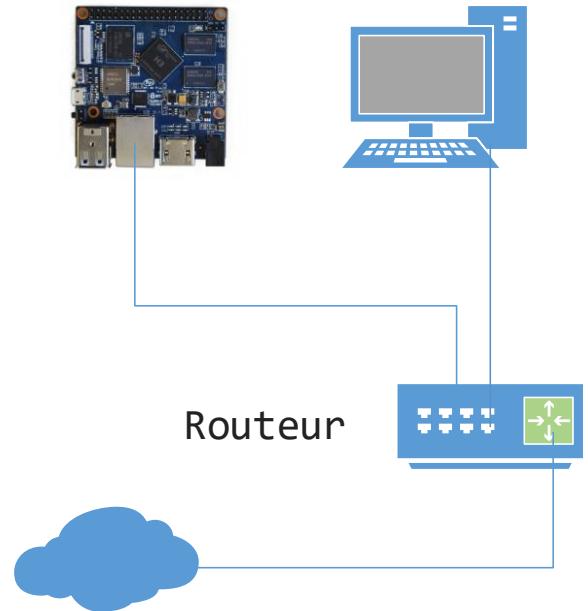


Figure 7 : connection ethernet sur la cible

Avant de commencer, vous aurez besoin de connaître l'adresse IP de votre cible. Pour cela vous avez plusieurs solutions :

- ✓ Utiliser un scanneur d'adresse IP sur votre réseau local (Advanced IP Scanner par exemple)
- ✓ Se connecter sur le routeur afin de connaître l'adresse IP de votre cible
- ✓ Se connecter sur la cible au travers du port com virtuel (Cf les chapitres précédents) et entrer la commande **sudo ifconfig** dans la console de commande.

Nous ne verrons ici que la première solution : l'utilisation d'un outil de scan d'adresse IP. En effet, la deuxième solution nécessite de connaître les logins et mot de passe de votre routeur (ce qui est normalement le cas si vous utilisez votre box à domicile) mais ce qui n'est pas possible au sein de l'IUT. Quant à la troisième solution, nous verrons cela au chapitre suivant, nous devons en effet nous connecter sur la cible avant de pouvoir lire l'adresse IP de notre cible.

- | | |
|--|---|
| ✓ Lancer Avdiance Ip Scanner pour voir si votre carte est vu sur le réseau |  Advanced IP Scanner |
|--|---|



Application de bureau

Dans notre exemple, l'adresse IP de notre cible est 10.23.14.70. Laisser 10s avant de faire un scan, le temps que l'OS de votre cible fasse un DHCP et reçoive une adresse IP.

Nous allons maintenant nous connecter sur la cible

- | |
|--|
| ✓ Lancer Putty et connectez-vous à votre cible |
|--|

The screenshot shows the Advanced IP Scanner interface. A search result for IP 10.23.14.70 is highlighted with a blue oval. The result includes the IP address, manufacturer (Dell), and MAC address (02:81:6D:49:CE:D6). Below the table, a note says "✓ Lancer putty pour vous connecter sur la cible".

Advanced IP Scanner

Fichier Opérations Paramètres Afficher Aide

Analysé IP C

10.23.14.1-10.23.14.254 Rechercher

Liste des résultats Favoris

Statut	Nom	IP	Fabricant	Adresse MAC
Ordinateur	10.23.14.70	10.23.14.70	Dell	02:81:6D:49:CE:D6
Ordinateur	PC-SALVAT.iutnice.unice.fr	10.23.14.71		E4:B9:7A:26:DF:45

✓ Lancer putty pour vous connecter sur la cible

PuTTY Configuration

Category: Session

Host Name (or IP address): 10.23.14.70 Port: 22

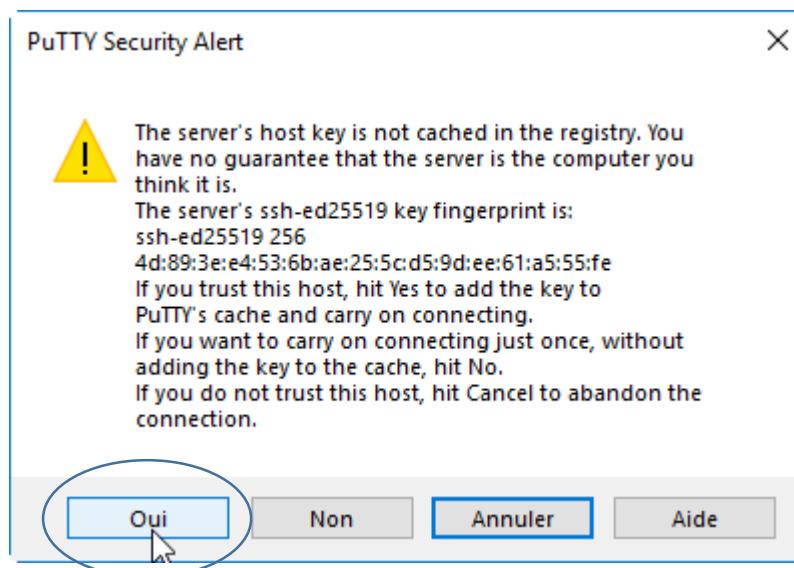
Connection type: SSH

Saved Sessions: Default Settings nodeus box

Close window on exit: Only on clean exit

About Help Open Cancel

La connexion avec la cible se fait en ssh (utilisation de clé de cryptage), à la première connexion, un échange de clé est effectué entre le PC et la cible



Vous obtenez alors la même console de connexion que celles vu précédemment.

Remarque : La connexion à la cible peut se faire au travers d'un cable ethernet mais aussi au travers du Wifi dans le cadre d'une utilisation à domicile avec une box ADSL ou fibre. En effet ces boxs connectent au sein de votre réseau local privé tous les appareils qu'ils soient Wifi ou sur cable Ethernet. Nous verrons ensuite comment se passer du cable ethernet, puisque votre bananapi possède un module Wifi intégré et donc est capable de se connecter au Wifi.

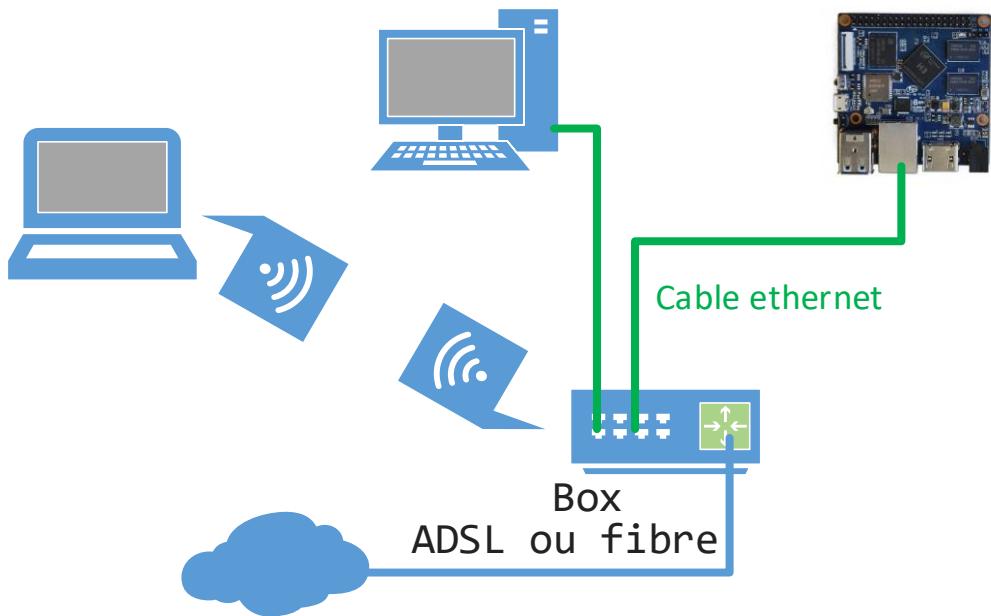


Figure 8 : connexion en SSH à la cible avec un PC portable connecté en Wifi sur le réseau local.

3.5 Résumé des 3 manips

Nous avons dans cette première partie montré comment se connecter sur la cible de 3 façons. Si vous avez suivi jusqu'au bout ces manips vous devriez avoir 3 fenêtre Putty ouvertes :

- ✓ La fenêtre Putty COM17 associée au cable USB micro
- ✓ La fenêtre Putty COM3 associé au cable USB + pont USB/TX (FTDI) vers le port de DEBUG
- ✓ La fenêtre Putty SSH de connexion à la cible via Ethernet.

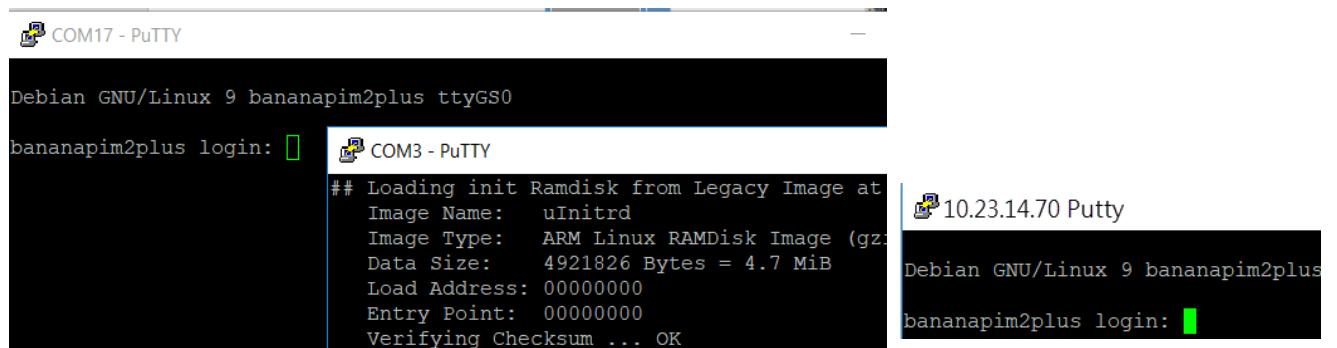


Figure 9 : copie d'écran des 3 fenêtres Putty de connexion à la cible

4 Configuration de la cible

4.1 Introduction

Quelle que soit la fenêtre de com que vous allez utiliser, nous allons maintenant nous connecter à la cible (login **root** et mot de passe par défaut **1234**) puis créer un utilisateur (il est fortement déconseillé de se connecter en root (super utilisateur) sur la cible. Nous lancerons ensuite une commande de mise à jour des paquets (package) logiciels de votre distribution ; en effet il existe toujours une différence entre l'image de votre distribution Debian copiée sur votre SD qui sera toujours plus ancienne que la distribution mis à jour continuellement sur le serveur DEBIAN. Pour finir, nous donnerons un nom à la cible afin de la différencier des autres bananapi ou appareils sur le réseau.

4.2 Première configuration

A la première connexion, voici le login et le mot de passe :

Login : **root**

Mot de passe : **1234**

```
login as: root
root@192.168.1.10's password:
You are required to change your password immediately (root enforced)

Welcome to ARMBIAN 5.69 stable Ubuntu 18.04.1 LTS 4.19.13-sunxi
System load:  0.08  0.23  0.12   Up time:      4 min
Memory usage: 8 % of 1000MB    IP:          192.168.1.10
CPU temp:    42°C
Usage of /:   5% of 15G

New to Armbian? Check the documentation first: https://docs.armbian.com
Changing password for root.
(current) UNIX password:
```

A la première installation, il vous est demandé de changer le mot de passe **root** qui est trop simple à trouver

Attention, il vous est demandé le mot de passe actuel avant de changer le mot de passe root. Le mot de passe doit faire au moins 8 caractères. Choisir **apprentis** comme mot de passe.

```
(current) UNIX password : 1234
Enter new UNIX password : apprentis
Retype new UNIX password:apprentis
```

Vous allez ensuite créer un utilisateur (question suivante), attention à mettre pour la cible l'utilisateur **geii** et mot de passe **geii**. C'est avec ce login et mot de passe que vous travaillerez dorénavant.

```
Creating a new user account. Press <Ctrl-C> to abort
Please provide a username (eg. your forename):geii
Adding user `geii' ...
Adding new group `geii' (1001) ...
Adding new user `geii' (1001) with group `geii' ...
Creating home directory `/home/geii' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:geii
Retype new UNIX password:geii
```

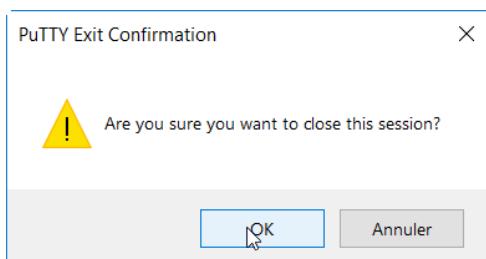
Pour la suite tapez sur Entrée pour toutes les autres questions. Vous devriez avoir l'écran suivant à la fin du process.

```
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for geii
Enter the new value, or press ENTER for the default
      Full Name []:
      Room Number []:
      Work Phone []:
      Home Phone []:
      Other []:

Is the information correct? [Y/n]

Dear geii, your account geii has been created and is sudo enabled.
Please use this account for your daily work from now on.
```

En général pour des raisons de sécurité on ne se connecte jamais en root. Votre OS est donc prés à l'emploi. Fermer la connexion ssh en root



- ✓ Relancer putty et connectez-vous avec le login geii : geii@10.23.14.70 (si l'adresse ip de votre cible est 10.23.14.70)

```
geii@bananapim2plus: ~
login as: geii
geii@192.168.1.11's password:

Welcome to ARMBIAN 5.69 stable Ubuntu 18.04.1 LTS 4.19.13-sunxi
System load: 0.00 0.00 0.00 Up time: 2:30 hours
Memory usage: 8 % of 1000MB IP: 192.168.1.11
CPU temp: 36°C
Usage of /: 5% of 15G

geii@bananapim2plus:~$
```

Nous allons maintenant **mettre à jour le système**. Cette opération se fait en 2 parties. D'abord la récupération des différences de version entre les paquets installés et les paquets à jour de la distribution.

```
sudo apt update
```

```
geii@bananapim2plus:~$ sudo get-apt update
sudo: get-apt: command not found
geii@bananapim2plus:~$ sudo apt-get update
Hit:1 http://ports.ubuntu.com bionic InRelease
Get:2 http://ports.ubuntu.com bionic-security InRelease [88.7 kB]
Get:4 http://ports.ubuntu.com bionic-updates InRelease [88.7 kB]
Get:5 http://ports.ubuntu.com bionic-backports InRelease [74.6 kB]
Get:3 https://apt.armbian.com bionic InRelease [18.9 kB]
Get:6 http://ports.ubuntu.com bionic-security/main armhf Packages [167 kB]
Get:7 http://ports.ubuntu.com bionic-security/universe armhf Packages [112 kB]
Get:8 http://ports.ubuntu.com bionic-security/multiverse armhf Packages [1,4 kB]
```

Puis téléchargement et installation des nouvelles versions des paquets, cela devrait prendre 3 à 5 minutes.

```
sudo apt upgrade
```

```
geii@bananapim2plus:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  apt apt-transport-https apt-utils armbian-config armbian-firmware bsutils
  dirmngr fdisk gnupg gnupg-l10n gnupg-utils gnupg2 gpg gpg-agent
  gpg-wks-client gpg-wks-server gpgconf gpgsm gpgv hostapd libapt-inst2.0
  libapt-pkg5.0 libasound2 libasound2-data libblkid1 libcaca0 libfdisk1
  libgssapi-krb5-2 libk5crypto3 libkrb5-3 libkrb5support0 libmount1
  libmysqlclient20 libnss-myhostname libnss-systemd libpam-systemd
  libpolkit-agent-1-0 libpolkit-backend-1-0 libpolkit-gobject-1-0
  libsmartcols1 libsystemd libudev1 libuuid1
```

Remarque : un paquet est un logiciel, ensemble de fichiers ou dll (library) avec un numéro de mise à jour. Sous Linux, chaque paquet est maintenu et mis à jour et une distribution est un ensemble de paquet cohérent entre eux.

4.3 Changer le nom de la machine...

Il existe plusieurs solutions pour changer le nom de votre cible (ou hostname), ici **bananapiM2plus**, afin que vous puissiez la différencier avec celle du votre voisin.

Nous allons voir ici 3 solutions :

- ✓ Utilisation de NetworkManager qui est le service à tout faire de la gestion du réseau
- ✓ Modification des fichiers de configuration du nom de la machine dans le répertoire /etc
- ✓ Utilisation de l'outil de configuration fourni par ARMBIAN : armbian-config

Si vous devez n'utiliser qu'une seule méthode, alors il faudra utiliser la deuxième méthode qui consiste à modifier les deux fichiers de configuration de nom **/etc/hostname** et **/etc/hosts**. Les 2 autres méthodes (pseudo-graphique) vont aussi modifier le fichier /etc/hostname, et oublier le fichier /etc/hosts ce qui aura tendance à ralentir la cible.

Les buts de ces manipulations sont aussi de commencer à découvrir quelques concepts et outils classiques de la gestion de notre cible, outils que nous reverrons par la suite.

4.3.1 Solution 1 : utilisation de l'outil de configuration du NetworkManager

Le **network manager** est le service (ou daemon) qui gère vos connexions sous Debian. Vous pouvez vérifier que ce service est bien lancé avec la commande

```
sudo service --status-all
```

```
[ + ]  kmod
[ + ]  loadcpufreq
[ + ]  motion
[ + ]  network-manager
[ + ]  networking
[ + ]  procps
[ + ]  resolvconf
```

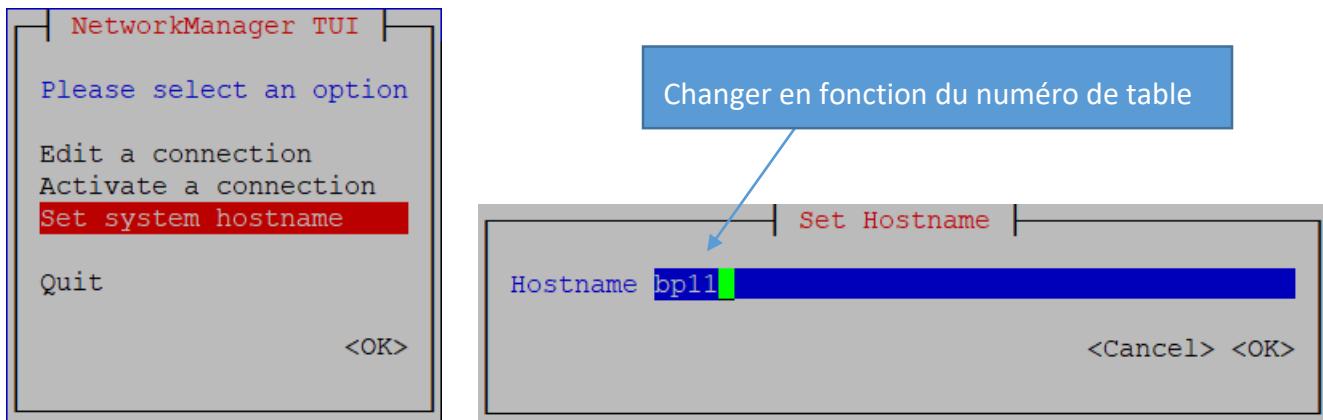
Le service réseau est bien lancé

Pourquoi mettons-nous sudo devant la plupart des commandes que nous avons vu jusqu'ici ?

sudo devant une commande permet de lancer cette commande en mode **root** (c'est-à-dire en mode super-utilisateur). En effet la plupart des commandes d'administration de l'OS Linux ne doivent pas être disponibles aux utilisateurs du système pour des raisons de sécurité. Seuls les utilisateurs « **sudoers** » ont la capacité avec la commande **sudo** de devenir **root** le temps d'une commande.

La configuration de ce service se fait par un outil pseudo-graphique (nous sommes en mode console) qui s'appelle **n(etwork)m(anager)t(ext)u(ser)i(interface)**.

```
sudo nmtui
```



Remarque : l'outil **nmtui** est surtout utilisé pour configurer simplement votre connexion ethernet ou wifi. Pour le wifi, il vous permet de vous connecter à un point d'accès et son ssid et de mettre la clé réseau. Une fois connecté, votre cible se connectera automatiquement lorsqu'elle sera à porté de ce point d'accès. C'est ce que nous verrons au chapitre suivant.

Une fois le **hostname** modifié, il ne manque plus qu'à relancer le système pour que le nom de votre cible soit visible dans Advance Ip Scanner.

```
sudo reboot
```

Autre solution : Au lieu de rebooter la cible, ce qui prend au moins 30s, on aurait tout simplement pu relancer le service (plus rapide)

```
sudo systemctl restart systemd-logind.service
```

✓ Après un certain temps (le temps du reboot) relancer advance ipscanner

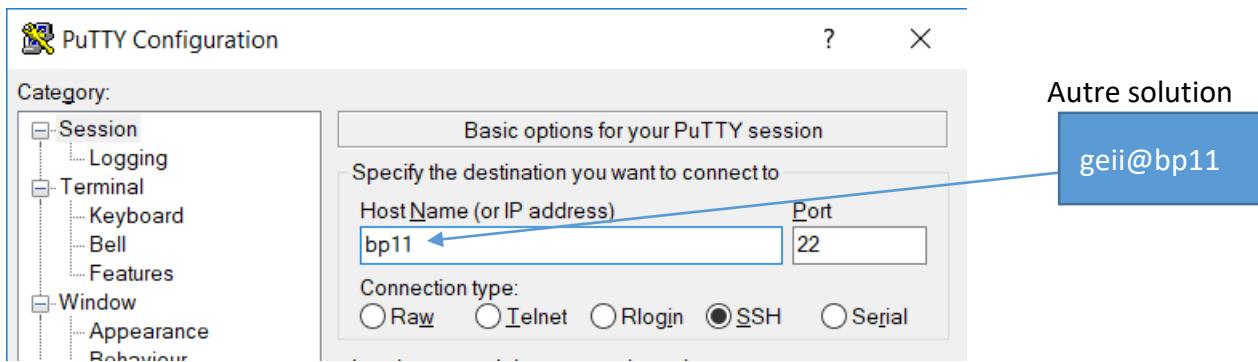
Vous devriez voir le nom de votre machine qui a été modifié

bp11.home

Vous pouvez donc vous connecter sur votre cible directement avec son nom

✓ Relancer putty

Cette fois, vous pouvez ne plus mettre l'adresse ip de votre cible mais le nom de votre cible



4.3.2 Solution 2 : changer le fichier de configuration hostname et hosts dans /etc

On aurait pu taper la commande **sudo nano /etc/hostname** et modifier le nom de votre cible dans le fichier. Vous pouvez vérifier que le fichier hostname a été modifié par **nmtui** (si vous avez suivi la première manip).

```
sudo nano /etc/hostname
```

```
bp11
```

Remarque : nano est un éditeur permettant de modifier puis sauver (CTR+O) le fichier. Pour quitter (CTR+X). Si le fichier a été modifié et non sauvé (vous n'avez pas tapé sur CTR+O) il vous sera demandé lors de votre commande CTR+X de sauver ou non (Y ou N) les changements.

Comme peut-on voir que la commande **nmtui** a déjà modifié le fichier **hostname** ?

Si on lance la commande ls avec les paramètres t (pour time) et l (pour long) du répertoire /etc on va afficher tous les fichiers du répertoire /etc classé en fonction de leur date de modification (on peut voir que le fichier hostname a été modifié le mardi 8 Mars à 16 :13 et 16 :08 alors que tous les autres fichiers l'ont été bien avant. Date actuelle de la manipulation.

```
ls -lt /etc
```

drwxr-xr-x 2 root root 4096 Mar 8 16:14	ssh
drwxr-xr-x 2 root root 4096 Mar 8 16:14	init.d
drwxr-xr-x 2 root root 4096 Mar 8 16:14	pam.d
drwxr-xr-x 2 root root 4096 Mar 8 16:14	default
drwxr-xr-x 2 root root 4096 Mar 8 16:14	init
drwxr-xr-x 5 root root 4096 Mar 8 16:13	systemd
drwxr-xr-x 4 root root 4096 Mar 8 16:13	udev
drwxr-xr-x 2 root root 4096 Mar 8 16:13	modules-load.d
drwxr-xr-x 2 root root 4096 Mar 8 16:13	sysctl.d
-rw-r--r-- 1 root root 184 Mar 8 16:13	hosts
-rw-r--r-- 1 root root 24146 Mar 8 16:12	ld.so.cache
drwxr-xr-x 2 root root 4096 Mar 8 16:12	ld.so.conf.d
drwxr-xr-x 2 root root 4096 Mar 8 16:12	update-motd.d
-rw-r--r-- 1 root root 108 Mar 8 16:08	resolv.conf.WX4UYZ
-rw-r--r-- 1 root root 5 Mar 8 16:08	hostname
-rw-r----- 1 root shadow 618 Feb 9 20:17	gshadow
-rw-r--r-- 1 root root 733 Feb 9 20:17	group

Remarque : tous les fichiers dans le répertoire **/etc** (le c comme configuration) sont des fichiers de configuration. Il faudra souvent modifier un fichier dans ce répertoire pour configurer un service ou un programme déjà installé. Nous verrons le détail des informations affichées par la commande `ls -l` plus tard.

Pour finir, il est préférable d'ajouter aux 2 premières lignes du fichier **/etc/hosts**, le nom de votre machine (bp11). **hosts** est le fichier interne permettant d'associer adresse IP avec nom de machine. Il est complémentaire à `hostname`. **Si vous n'ajoutez pas le nom bp11 dans le fichier hosts, le fonctionnement de votre cible risque d'être ralenti.**

```
sudo nano /etc/hosts
127.0.0.1 localhost bp11
::1   localhost bp11 ip6-localhost ip6-loopback
```

A rajouter aux 2 premières lignes du fichier hosts

Nous avons vu 2 solutions de modification du nom de votre cible. Jamais 2 sans 3, nous vous proposons une troisième solution simple et surtout permettant de modifier les 2 fichiers **hostname** et **hosts**.

4.3.3 Solution 3 : le script à tout faire armbian-config

Armbian-config est un script fourni par la distribution embarquée **ARMBIAN**. Elle est assez utile puisqu'elle permet de simplifier pas mal de tâches d'administration ou de configuration du système, et notamment de modifier le **hostname** de la machine.

Nous aurions pu directement commencer par ce script, mais il était plus intéressant de voir comment modifier ce nom de machine directement en ligne de commande, vous ne trouvez pas ?

Faites la manipulation, cela permettra d'être sur que les fichiers 2 fichiers `hostnames` et `hosts` ont été modifiés correctement.

```
sudo armbian-config
```

Configuration utility, Armbian 5.73 stable, 192.168.1.13

4.4 Conclusion

Vous avez pu voir différentes solutions permettant de changer le nom de votre cible. Cette partie est importante si l'on possède plusieurs cibles et que l'on veut les différencier sur le réseau.

5 Première connexion en wifi

5.1 Introduction

Notre cible possède un composant intégré AMPAK AP6212 intégrant le Wifi et le bluetooth.

http://linux-sunxi.org/Table_of_Allwinner_based_boards

Maker	Model	SoC	Mainline	PMU/Vreg	Board size (mm)	RAM (MHz)	Storage	eMMC/SPI Flash	Ethernet speed (chip)	Wifi	USB	Audio	Mic	IR	GPIO	CSI	DVI	Video out
Sinlinx	Sinlinx SinA31s	A31s	WiP	AXP221s	170x108	1GB (432)	SD	4/16 GB	100 (RTL8201CP)	-	5xUSB2, 1xOTG	3.5mm	3.5mm	-	3 x 20pin	-	+	LCD
Sinlinx	Sinlinx SinA33	A33	WiP	AXP223	99x82	1GB (552)	μSD	4 GB	-	-	1xUSB2, 1xOTG	3.5mm	3.5mm	-	3 x 20pin	-	+	LCD
Sinlinx	Sinlinx SinA33 Plus	A33	WiP	AXP223	145x108	1GB (552)	SD	4 GB	100 (?)	-	3xUSB2, 1xOTG	3.5mm	3.5mm	-	3 x 20pin	-	+	LCD,VGA
Sinovoip	Banana Pi M1+	A20	WiP	AXP209	92x60	1GB (432)	μSD, SATA	-	1000 (RTL8211E)	b/g/n (AP6181)	2xUSB2, 1xOTG	3.5mm, HDMI	+	RX	40pin	+	+	HDMI, DSI, 3.5mm
Sinovoip	Banana Pi M2	A31s	WiP	AXP221s	92x60	1GB (432)	μSD	-	1000 (RTL8211E)	b/g/n (AP6181)	4xUSB2*, 1xOTG	3.5mm, HDMI	+	RX	40pin	+	+	HDMI, DSI
Sinovoip	Banana Pi M2 Magic	A33	WiP	?	51x51	512MB	μSD	8G (/16/32/64G)	-	b/g/n (AP6212)	1xUSB2, 1xOTG	-	+	-	40pin	+	+	DSI
Sinovoip	Banana Pi M2+	H3	WiP	?	65x65	1GB (432)	μSD	8GB	1000 (RTL8211E)	b/g/n (AP6181)	2xUSB2, 1xOTG	3.5mm, HDMI	-	RX	40pin	+	-	HDMI

Le bluetooth ne fonctionne pas (pas de pilote Linux), par contre le wifi intégré est fonctionnel. C'est ce que nous allons voir ici en utilisant la puce intégrée wifi de notre bananapi M2+ pour nous connecter à un point d'accès.

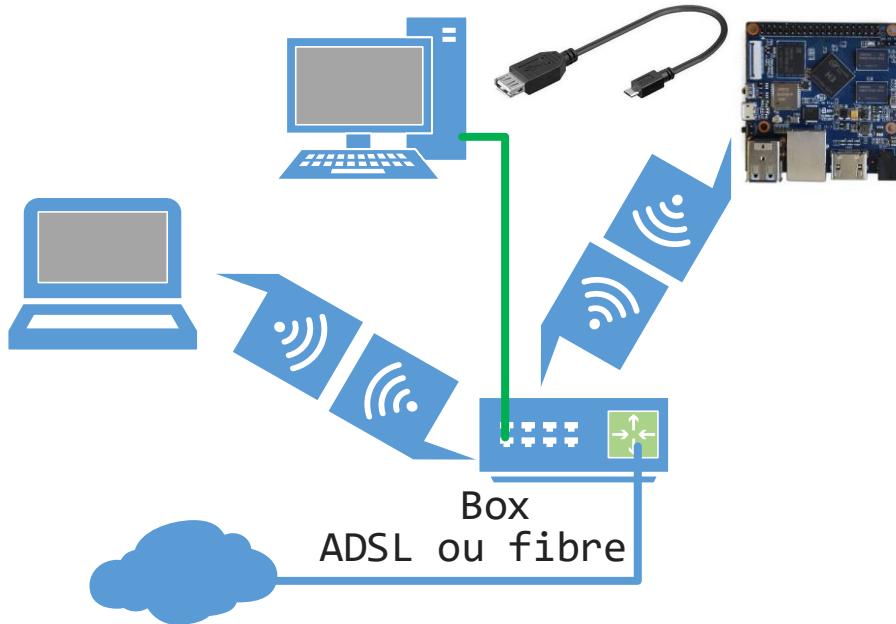


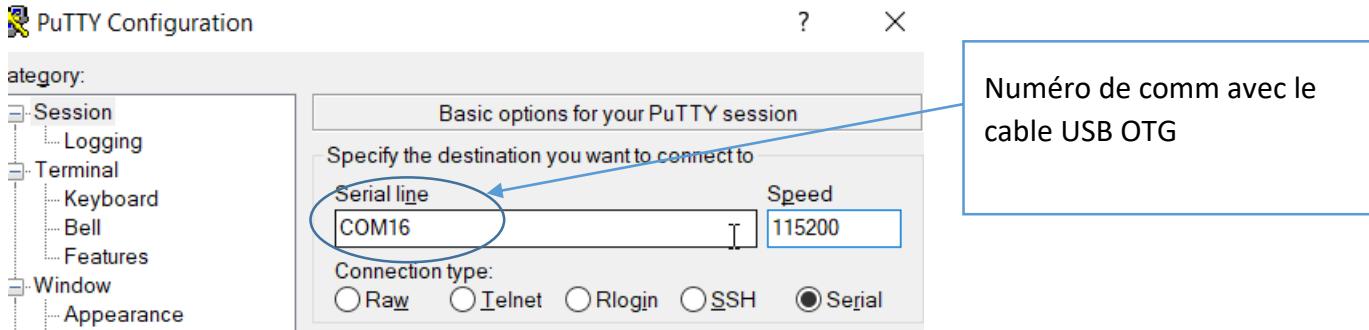
Figure 10 : connexion de notre cible à un point d'accès

Nous ne verrons pas ici comment connecter la cible à un réseau Wifi de type Hotspot Unice, solution difficile à mettre en œuvre, mais plutôt la solution classique d'une connexion wifi à un point d'accès wifi WPA ou WPA2 (SSID et passphrase) classique pour une connexion chez soi sur sa box.

5.2 Utilisation de la commande nmtui (NetworkManager)

Nous supposons dans cette manip que vous êtes connecté via le cable USB OTG à la cible .

Nous rappelons que pour se connecter en OTG sous putty, il suffit de sélectionner Serial et le numéro de com (que l'on peut trouver dans le gestionnaire de périphériques).



Un point d'accès a été créé en salle 2G :

SSID : salle_2G

Mot de passe : geii2017

Il existe une commande très simple permettant de se connecter via une interface graphique en mode texte à votre wifi, c'est la commande nmtui.

```
sudo nmtui
```

Accéder au sous menu **activate** wifi pour choisir le point d'accès sur lequel vous voulez vous connecter. Entrer le mot de passe.

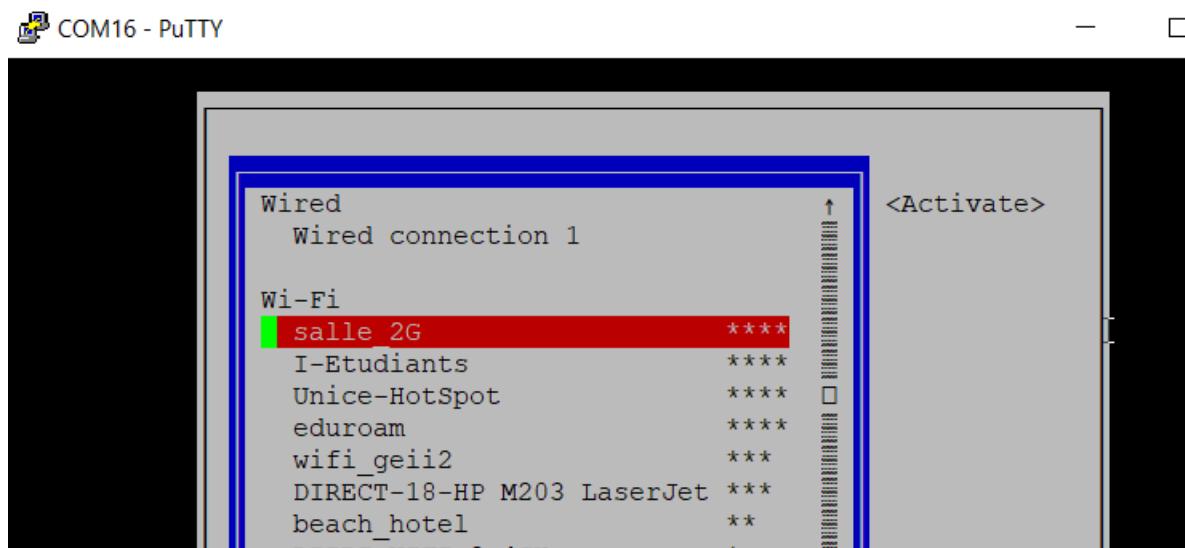


Figure 11 : connexion au SSID salle_2G via la commande nmtui.

Votre connexion Wifi est maintenant active. Vous pouvez vérifier avec la commande sudo ifconfig que le wifi est ok.

```
sudo ifconfig
```

```
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
      ether 02:81:07:4f:88:09 txqueuelen 1000 (Ethernet)
      RX packets 0 bytes 0 (0.0 B)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 0 bytes 0 (0.0 B)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
      device interrupt 38

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.1.17 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::8442:17c:c02d:c4ad prefixlen 64 scopeid 0x20<link>
        inet6 2a01:cb1d:3e2:9b00:b085:1daf:de97:bb6 prefixlen 64 scopeid 0x0<global>
          ether 10:d0:7a:79:c8:05 txqueuelen 1000 (Ethernet)
          RX packets 2308 bytes 1237411 (1.1 MiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 1516 bytes 205429 (200.6 KiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Comme on peut le voir, le résultat de la commande **ifconfig**, nous montre que l'interface **wlan0** (l'interface Wifi) est active et possède une adresse IP alors que l'interface eth0 n'est pas configuré. Dans cet exemple, le cable ethernet n'est pas branché, seul le cable USB micro a été connecté. On pourra tester que la connexion Wifi est ok. L'interface lo est la boucle locale en 127.0.0.1

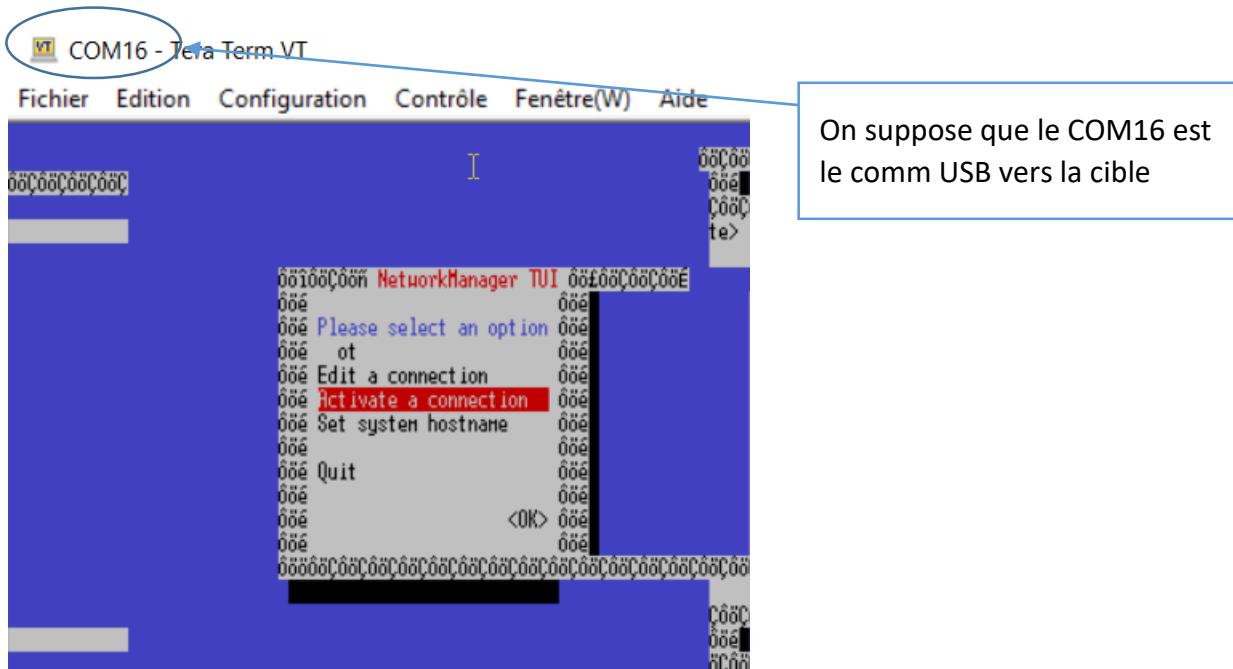
```
ping www.google.fr
```

```
PING www.google.fr(par10s34-in-x03.1e100.net (2a00:1450:4007:817::2003)) 56 data bytes
64 bytes from par10s34-in-x03.1e100.net (2a00:1450:4007:817::2003): icmp_seq=1 ttl=55 time=23.1 ms
64 bytes from par10s34-in-x03.1e100.net (2a00:1450:4007:817::2003): icmp_seq=2 ttl=55 time=20.8 ms
64 bytes from par10s34-in-x03.1e100.net (2a00:1450:4007:817::2003): icmp_seq=3 ttl=55 time=21.0 ms
64 bytes from par10s34-in-x03.1e100.net (2a00:1450:4007:817::2003): icmp_seq=4 ttl=55 time=20.8 ms
```

Appuyez sur CTR+C pour arrêter la commande

Après la première connection au réseau Wifi de votre point d'accés (téléphone ou borne wifi), au prochain démarrage, votre cible se connectera automatiquement à votre point d'accés.

Remarque : sur **teraterm** la visualisation des données séries n'est pas optimale.



La visualisation des données sur TeraTerm ne permet pas de visualiser les noms des points d'accès. C'est pour cette raison que nous avons utilisé putty ; voyons comment se connecter au Wifi sans utiliser **nmtui**, mais en utilisant l'interface **nmcli** (**n**etwork **m**anager **c**ommand **l**ine **i**nterface) qui est la commande de base de gestion du réseau. Cette partie n'est pas fondamentale mais continue à nous familiariser avec le monde Linux et sa multitude de commandes.

5.3 Configuration du wifi sans nmtui

Linux propose toujours plusieurs solutions pour arriver au même résultat. Nous allons voir ici comment configurer le wifi sans utiliser la commande nmtui. Nous allons utiliser l'outil nmcli qui est l'outil de gestion du réseau en ligne de commande alors que nmtui enrobe ces commandes sous forme graphique.

Dans un premier temps on pourra lancer une visu des différents AP (Access Point)

```
sudo nmcli device wifi
```

IN-USE	SSID	MODE	CHAN	RATE	SIGNAL	BARS
	eduroam	Infra	11	195 Mbit/s	92	****
	Unice-HotSpot	Infra	11	195 Mbit/s	90	****
	I-Etudiants	Infra	11	195 Mbit/s	89	****
	salle_2G	Infra	6	54 Mbit/s	87	****
	wifi_geii2	Infra	6	130 Mbit/s	67	***
	Unice-HotSpot	Infra	1	195 Mbit/s	62	***
	DIRECT-18-HP M203 LaserJet	Infra	6	130 Mbit/s	62	***

Ensuite faites appel à l'aide linux

```
sudo nmcli device wifi --help
```

```
geii@bp12:~ $ sudo nmcli device wifi --help
Usage: nmcli device wifi { ARGUMENTS | help }

Perform operation on Wi-Fi devices.

ARGUMENTS := [list [iface <iface>] [bssid <BSSID>]]

List available Wi-Fi access points. The 'iface' and 'bssid' options can be
used to list APs for a particular interface, or with a specific BSSID.

ARGUMENTS := connect <(B)SSID> [password <password>] [wep-key-type key|phrase] [iface <iface>]
               [bssid <BSSID>] [name <name>] [private yes|no] [hidden yes|no]
```

L'aide vous donne la façon d'utiliser cette commande : la commande nmcli est un commande proposant un grand nombre d'option ; **wifi-connect** permet de choisir le point d'accès, et **password** de donner la clé wifi.

```
sudo nmcli device wifi connect salle_2G password geii2017
```

```
geii@bp12:~ $ sudo nmcli device wifi connect salle_2G password geii2017
Device 'wlan0' successfully activated with '9d358bc0-a878-44da-a3c4-1f9e156fd825'.
```

Vérifier que c'est ok, ici l'adresse ip de notre carte Wifi est 10.23.3.146. Donc la carte semble configurée.

```
sudo ifconfig
```

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 10.23.3.146 netmask 255.255.255.0 broadcast 10.23.3.255
      inet6 fe80::7313:2184:418b:63c4 prefixlen 64 scopeid 0x20<link>
        ether 10:d0:7a:79:c8:05 txqueuelen 1000 (Ethernet)
```

On pourra pinguer google pour vérifier que c'est ok

```
ping www.google.fr
```

```
geii@bp12:~ $ ping www.google.fr
PING www.google.fr (172.217.18.35) 56(84) bytes of data.
64 bytes from ham02s12-in-f3.1e100.net (172.217.18.35): icmp_seq=1 ttl=52 time=8.43 ms
64 bytes from ham02s12-in-f3.1e100.net (172.217.18.35): icmp_seq=2 ttl=52 time=10.3 ms
```

5.4 En cas de problème :

On pourra vérifier que la connexion wifi est dans le bon mode (managed)

```
sudo iw wlan0 info
```

```
geiii@BP4:~$ sudo iw wlan0 info
Interface wlan0
    ifindex 4
    wdev 0x1
    addr 10:d0:7a:79:c7:b5
    type managed
    wiphy 0
    channel 1 (2412 MHz), width: 20 MHz, center1: 2412 MHz
    txpower 31.00 dBm
```

Et vérifier les messages du noyau, et vérifier que wlan0 (link is ready), pour vérifier que le driver est bien lancé (ce qui est le cas dans le dernier message).

```
dmesg | grep wlan0
```

```
geiii@bp12:~ $ dmesg | grep wlan0
[ 14.189020] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[ 14.228267] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[ 14.617197] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[ 901.863239] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[ 920.534378] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[ 1161.034623] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
```

Remarque : la commande grep wlan permet de filtrer tous les messages dmesg issus du noyau et des différents services lancés lors du démarrage de votre distribution. Essayez de lancer la commande dmesg pour voir tous ces messages. Nous verrons tout cela en détail dans la partie Exercices Linux.

On pourra vérifier si la carte wifi est bien reconnue (l'option -a permet de voir les cartes même si celle-ci ne sont pas activées).

```
sudo ifconfig -a
```

On pourra essayer de relancer le wifi

```
sudo ifconfig wlan0 up
```

ou bien en utilisant la commande plus haut niveau qui relance tous les services réseaux.

```
service network-manager restart
```

Vérifier aussi que la communication Wifi n'est pas bloquée (de façon logicielle)

```
rfkill list
```

```
geii@bp11:~$ rfkill list
0: phy0: Wireless LAN
    Soft blocked: no
    Hard blocked: no
1: hci0: Bluetooth
    Soft blocked: no
    Hard blocked: no
```

Pour activer l'interface WiFi

```
sudo rfkill unlock wifi
```

Pour désactiver l'interface WiFi

```
sudo rfkill lock wifi
```

5.5 Conclusion

Que se soit avec l'outil **nmtui** ou **nmcli**, la connexion au réseau wifi est assez simple. Une fois cette connexion configurée, celle-ci sera automatiquement active dès que la cible sera à proximité du point d'accès. Vous n'aurez donc pas à lancer cette connexion après avoir redémarrer votre cible.

Remarque : nous ne verrons pas ici comment utiliser la communication bluetooth pour se connecter à la cible, puisque la partie bluetooth de l' AP6212 n'est pas supporté par le noyau.

Nous en avons donc fini avec les différentes solutions de connection à la cible, nous allons maintenant nous intéresser au transfert de fichier de la cible vers le host (le PC).

6 Comment copier des fichiers du pc host vers la cible ?

6.1 Introduction

Le transfert de fichiers du host vers la cible est assez commun :

- ✓ Il peut être utilisé pour copier un fichier de données (datalogger) enregistré sur la cible, pour le traiter sur le host
- ✓ Lors du développement de programmes, il est souvent plus intéressant d'utiliser le PC pour développer et ainsi bénéficier de la puissance de calcul du PC par rapport à la cible puis ensuite de transférer le fichier source pour les compiler sur la cible, ou même en cas de développement croisé, de compiler le programme sur le PC et de transférer l'exécutable sur la cible.

Le PC qu'il soit sous Linux, Windows ou Mac reste l'outil de référence pour le développement des programmes, la compilation du noyau ou de librairies pour la cible. Il est donc primordial de savoir transférer les fichiers de la cible vers le host et vice-versa. Nous allons voir ici 3 solutions d'échanges de données entre la cible et le host :

- ✓ Utilisation d'un clé USB
- ✓ Utilisation de la commande scp (Winscp)
- ✓ Utilisation de Samba (service réseau Linux pour communiquer avec le service netbios de Windows)

Nous ne verrons pas ici les solutions proposées par linux et le système de fichier **nfs** qui permet comme samba (pour windows) de partager des fichiers entre 2 postes Linux.

6.2 Avec une clé USB

Cette question est classique. Supposons que nous voulions transférer un fichier vers la cible. La première solution est d'utiliser une clé usb, de copier le fichier ou les fichiers sur la clé, puis insérer la clé.

Une fois la clé insérée sur la cible, il faut vérifier que celle-ci a bien été détectée. Pour cela plusieurs solutions. La première consiste à regarder les partitions des différents disques visibles par le kernel.

```
more /proc/partitions
```

```
geii@bp12:~ $ more /proc/partitions
major minor #blocks name

      1       0      4096 ram0
      1       1      4096 ram1
      1       2      4096 ram2
      1       3      4096 ram3
    179       0  7676928 mmcblk0
    179       1  7519280 mmcblk0p1
    179       8  7634944 mmcblk1
    254       0     51200 zram0
    254       1   128016 zram1
    254       2   128016 zram2
    254       3   128016 zram3
    254       4   128016 zram4
      8       0  3994624 sda
      8       1  3994592 sda1
```



Nous voyons ici les partitions en ram (rami et zrami), les 2 mémoires flash (la SD = mmcblk0 et la flash interne mmcblk1) et pour finir le disque sda et sa première partition sda1 de taille 4Go. C'est notre clé USB

Une autre solution est de lancer la commande **dmesg** ou **sudo tail /var/log/kern.log**, pour visualiser les messages du noyau. Lors de l'insertion de la clé USB vous devriez voir apparaître les messages ci-dessous.

```
dmesg
```

```
[ 147.196546] sd 0:0:0:0: Attached scsi generic sg0 type 0
[ 147.197503] sd 0:0:0:0: [sda] 7989248 512-byte logical blocks: (4.09 GB/3.81
GiB)
[ 147.198411] sd 0:0:0:0: [sda] Write Protect is off
[ 147.198421] sd 0:0:0:0: [sda] Mode Sense: 03 00 00 00
[ 147.199142] sd 0:0:0:0: [sda] No Caching mode page found
[ 147.199150] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 147.204412] sda: sda1
[ 147.207757] sd 0:0:0:0: [sda] Attached SCSI removable disk
[ 262.998066] usb 6-1: USB disconnect, device number 2
```

Il ne reste plus qu'à monter la partition sur un répertoire vide. 2 répertoires sont destinés à ces montages : /media ou /mnt à la racine

```
sudo mount /dev/sda1 /media
```

On pourra alors aller dans le répertoire media et copier les fichiers ou répertoires dans un répertoire du compte geii.

```
sudo cp -r /media/rep_a_copier ~/nouveau_rep
```

Remarques : on suppose que la clé USB a déjà été formatée sous windows ou sous Linux et qu'elle contient des fichiers à copier. On pourra vérifier le type de la clé en lançant la commande **mount**

```
mount
```

```
/dev/sda1 on /media type vfat (rw,relatime,fmask=0022,dmask=0022,codepage=437,ic
charset=iso8859-1,shortname=mixed,errors=remount-ro)
```

Ici on peut voir que la partition sda1 a été montée sur le répertoire /media et est formatée en FAT32 (vfat). Linux est aussi capable de lire ou écrire sur du NTFS.

Une fois les fichiers copiés sur votre compte, vous pouvez aussi copier des fichiers de votre compte sur la clé. Attention cependant à démonter la clé usb avant de la retirer, si vous avez copier des fichiers sur la clé ; en effet comme sous Windows, les fichiers copiés ne sont pas obligatoirement copiés directement sur la clé mais en RAM (sur le cache) et ne seront copiés et synchronisés qu'une fois l'ordre de démontage de la clé envoyé.

```
sudo umount /media
```

```
geii@bp12:/media $ sudo umount /media
umount: /media: target is busy.
```

Impossible de démonter la clé USB si vous êtes dans le répertoire à démonter

Lors du démontage de la clé (commande umount), il faut que vous quittiez le répertoire /media ; comme sous windows si vous aurez le message target busy, c'est que vous êtes toujours sur le répertoire monté. Donc utilisez la commande cd (pour revenir vers votre /home/geii) puis démonter la clé.

```
cd  
sudo umount /media
```

Remarque : nous avons vu ici un certain nombre de notions nouvelles pour un débutant. La notion de partitions connues sous Windows (tout support physique de mémoire de masse : disque dur SSD ou non, SD, clé USB,... peut être partitionnée en une ou plusieurs partitions, chaque partition pouvant être vue comme un lecteur logique donc comme un disque dur virtuel). Les commandes **mount** et **umount** permettant de monter ou démonter une partition sur un répertoire. Nous avons aussi vu une notion plus complexe à comprendre et que nous reverrons par la suite. C'est la notion de fichier associé à du matériel.

/dev/sda1 : partition 1 de la clé USB

/dev/mmcblk0p1 : partition 1 de la SD card mmcblk0

/dev/mmcblk1 : deuxième mémoire flash emmc (8Go se trouvant sur votre cible)

/dev/mmcblk1boot0 : première partition de la deuxième mémoire flash emmc (8Go se trouvant sur votre cible)

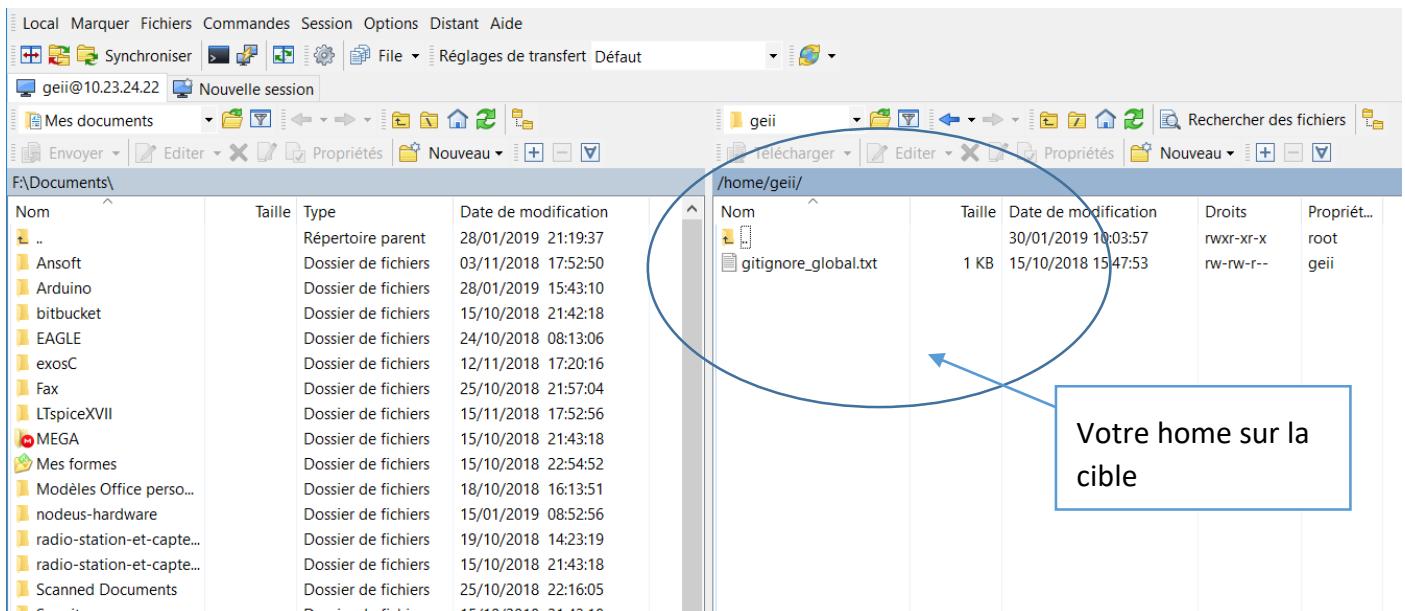
/dev/ram0 : zone de mémoire ram zone 0

Tous les fichiers se trouvant dans le répertoire /dev sont associés à du matériel au travers d'un driver. Ces fichiers sont l'interface entre le matériel et l'utilisateur.

6.3 Winscp

Winscp (<https://winscp.net/>) est un logiciel permettant de transférer des fichiers sur une cible distante à condition que cette cible ait un serveur ssh, ce qui est le cas de notre cible (c'est ce serveur qui nous permettait de nous connecter au travers du programme putty et du câble ethernet).

✓ Lancez Winscp et connectez vous à la cible



- ✓ Tester le glisser déposer de la gauche vers la droite
- ✓ Ouvrez une console putty et connectez vous sur la cible, visualiser le fichier

```
ls
```

Nous allons voir maintenant une autre solution, moins simple que les deux précédents, mais intéressante à mettre à œuvre dans un environnement windows et qui va nous permettre de monter un répertoire de notre cible sur un lecteur logique réseau sous Windows

6.4 Samba

Le protocole **SMB**, pour « Server Message Block » est le protocole réseau utilisé par Windows pour visualiser les machines ou les imprimantes se trouvant sur le réseau local. C'est le protocole utilisé lors des partages réseaux et pour le transfert de données (fichiers ou autres) entre les machines.



Samba est le logiciel (ou paquet /package en anglais) que nous allons installer sur notre cible afin de pouvoir rendre notre bananapi visible sous Windows et fournir une zone de partage permettant l'échange de fichiers entre notre PC ou tout autre PC se trouvant sur le réseau local et notre cible.

Dans cette solution, nous allons donc installer le paquet samba et le configurer afin de permettre le partage de fichiers.

Nous allons donc installer samba (l'option `-y` permet de répondre yes aux questions qui pourraient être posées lors de l'installation).

```
sudo apt-get -y install samba
```

Avant de modifier la configuration, commençons par faire une copie de sécurité du fichier `smb.conf`

```
sudo mv /etc/samba/smb.conf /etc/samba/smb.conf.bak
```

Créer un nouveau fichier de configuration

```
sudo nano /etc/samba/smb.conf
```

smb.conf

```
[global]
workgroup = WORKGROUP
server string = Samba Server %v
netbios name = bp1
security = user
map to guest = bad user
dns proxy = no
```

Chez vous avec une box, en général le groupe de travail s'appelle WORKGROUP.

Le nom de votre cible vue par windows. Mettez le même nom de votre hostname (bp suivi du numéro de table, ici table 1)

Remarque : si vous ne connaissez pas le nom du groupe de travail, ouvrez l'invite de commande sous Windows (ou powershell) et exécutez la commande suivante



Invite de commandes

Application de bureau

```
net config workstation
```

Version du logiciel

Windows 10 Pro

Domaine de station

WORKGROUP

Domaine de connexion

PC-SALVAT

A l'IUT, le nom du groupe de travail doit être iut-geii.unice.fr

Enregistrez le fichier de configuration – CTRL + X puis Y – puis relancez le service samba

```
sudo systemctl restart smbd.service
```

6.4.1 Partage d'un répertoire de votre cible

Il est possible de permettre à n'importe quel utilisateur d'accéder au dossier partagé. Nous n'allons par contre autoriser que les utilisateurs déclarés sur le système. Samba est capable d'utiliser les utilisateurs déjà présents sur le système. Il faut toutefois créer un mot de passe dédié à samba. Prenons par exemple l'utilisateur geii, exécutez la commande suivante pour définir le mot de passe Samba.

```
sudo smbpasswd -a geii
```

Remarque : l'utilisateur geii est maintenant autorisé à accéder aux répertoires partagés samba.

6.4.2 Ajouter les dossiers de partage

Vous pouvez partager des dossiers existants. Par sécurité, il est préférable de contrôler l'accès aux dossiers partagés. Pour cela, nous allons créer un nouveau dossier **share** (partage) dans lequel ou pourra ajouter d'autres répertoires. Chaque répertoire disposant de ces droits d'accès. La localisation du dossier n'est pas importante. Ici, nous allons créer le dossier partagé directement sur le bureau.

```
mkdir -p /home/geii/Desktop/Share
```

Ajoutons ces dossiers au fichier de configuration

```
sudo nano /etc/samba/smb.conf
```

Puis coller ce bloc de paramètres à la fin du fichier de configuration **smb.conf**.

```
[Public]
comment = partage All Users
path = /home/geii/Desktop/Share
valid users = @geii
force group = geii
create mask = 0660
directory mask = 0771
writable = yes
```

Le fichier **smb.conf** est donc le suivant :

smb.conf

```
[global]
workgroup = WORKGROUP
server string = Samba Server %v
netbios name = bp1
security = user
map to guest = bad user
dns proxy = no
```

[Public]

```
comment = partage All Users
path = /home/geii/Desktop/Share
valid users = @geii
force group = geii
create mask = 0660
directory mask = 0771
writable = yes
```

Maintenant que tout est configuré, il ne reste plus qu'à prendre en compte les nouveaux paramètres. Pour cela, sauvegardez le fichier (CTRL + X puis O) et exécutez la commande suivante

```
sudo systemctl restart smbd.service
```

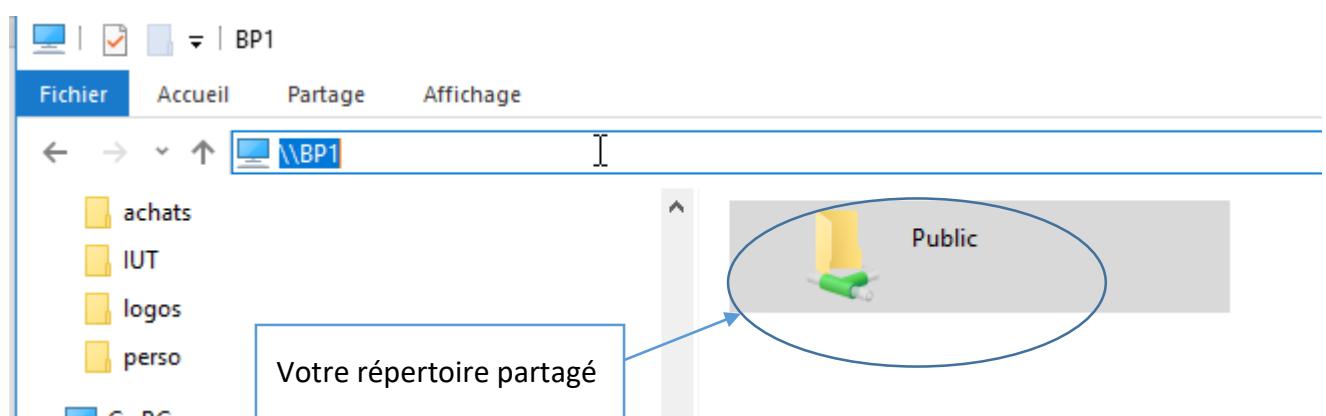
Sous windows, vous pouvez vérifier que votre cible peut être pinguer par son nom bp1.

- ✓ *Ouvrez une console sous windows et tapez **ping bp1** (le nom netbios de votre cible)*

Meilleur résultat

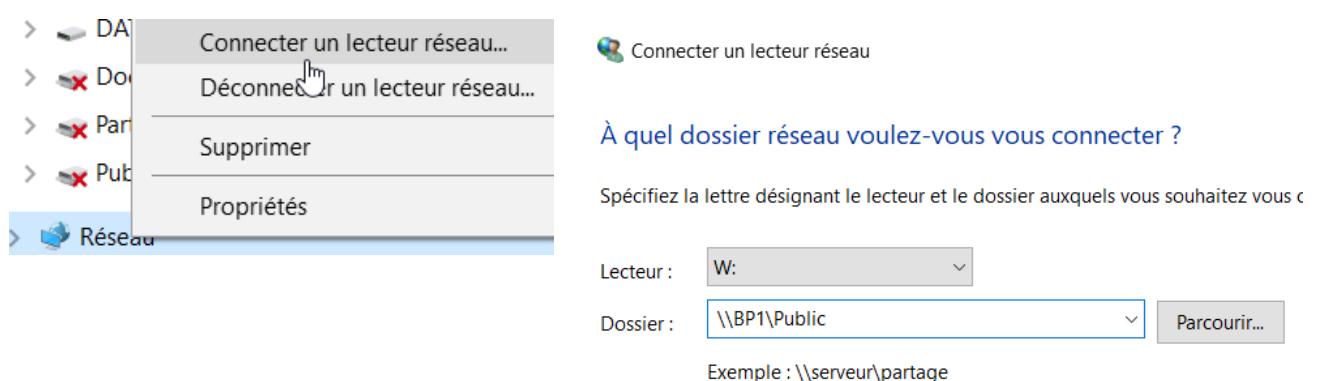


Remarque : si votre cible ne répond pas, vérifier que le nom de votre hostname est le même que celui mis dans le fichier **smb.conf**.



- ✓ *Vérifier que vous pouvez ajouter des fichiers et qu'ils sont visible dans le répertoire.*

Remarque : Vous pouvez même créer un lecteur réseau (clic droit sur réseau). Ici le disque virtuel W sera visible comme les autres disques et disponibles pour copier/coller des fichiers ou répertoires sur la cible.



Sources : <https://projetsdiy.fr/partage-fichiers-orange-pi-armbian-raspberry-raspbian/>

<https://linuxconfig.org/how-to-configure-samba-server-share-on-debian-9-stretch-linux>

7 Avoir une interface graphique ?

7.1 Introduction

Pour un utilisateur de Windows ou Mac, travailler sans interface graphique, comme nous l'avons fait jusqu'à présent, apparaît comme une solution rétrograde utilisée au début de l'informatique. Alors pourquoi revenir à l'âge de bronze de l'informatique alors que nous avons des outils graphiques puissants et simples d'utilisation ?

Dans l'embarqué les solutions nécessitant un écran sont plus coûteuses, on priviliera donc le plus souvent de ne pas mettre d'écran HDMI et si c'est nécessaire d'ajouter un affichage LCD de type caractères ou de taille très limité et d'utiliser l'écran d'une tablette ou d'un PC pour la visualisation des données. Prenons l'exemple des solutions Google Home ou Alexa mais aussi les box domotiques à monter soi-même <https://projetsdiy.fr/comment-fabriquer-box-domotique-diy-raspberry-pi3/> ou à acheter (Somfy,...), toutes ces box intègrent un cœur ARM avec de la RAM (en général au moins 512Mo DDR3) et de la mémoire flash (au minimum 16Go). Dans ce marché de l'embarqué, Linux (et Android) truste la plus grande part de marché.



Linux est donc une solution très utilisée dans l'embarqué et c'est une solution qui dans la plupart des cas, n'utilise pas d'interface graphique. L'interface graphique aura tendance à demander plus de puissance de calcul pour garder un fonctionnement fluide, on n'installera donc l'interface graphique seulement si celle-ci est nécessaire.

Dans ce chapitre nous allons installer une interface graphique Linux sur notre bananapi afin de tester son fonctionnement. Nous montrerons aussi par différentes manip, les principes qui régissent l'interface graphique sous Linux.

7.2 Deux solutions de test

Avant de commencer l'installation sur notre cible de l'interface graphique, voyons ensemble les solutions possibles permettant le test et la visualisation de notre interface. Il existe 2 solutions :

- ✓ Travailler directement à l'aide d'un câble HDMI, d'une souris et d'un clavier et utiliser le bananapi comme un PC.
- ✓ Utiliser notre PC, son clavier et sa souris comme écran déporté et faire l'économie d'un écran, d'un câble HDMI, d'un clavier et d'une souris.



Figure 12 : 2 solutions possible pour la gestion d'une interface graphique

L'installation de l'interface graphique peut se faire au travers de la commande armbian-config mais cela prendrait trop de temps (au moins 10 mn). Nous allons voir ici comment installer le minimum utile pour obtenir une interface graphique que nous visualiserons au travers de l'application VNC Viewer sur notre PC. Cette solution permet de faire l'économie d'un écran et d'un cable HDMI.

7.3 Premiers tests

Dans ce test, nous allons installer les paquets nécessaires à l'installation de notre interface graphique minimale :

- ✓ Le serveur graphique **Xorg** et ses composants
 - ✓ Un environnement graphique léger **xfce4** ou **lxde**. Ces environnements sont composés d'un grand nombre de programmes et notamment le gestionnaire de fenêtre (**windows manager**), les applications de gestion des icônes, des menus, le gestionnaire de fichiers,...
- Nous verrons le serveur graphique **Xorg** ne gère que l'écran, la souris et le clavier.

Lors de ces premiers tests, nous allons utiliser **xfce4** qui est un environnement graphique léger et tester la visualisation déportée via **VNC Viewer**.

Nous n'installerons ici que le minimum nécessaire (pas de navigateur, pas de client mail).

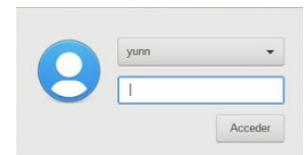
Lancer les commandes d'installation des différents paquets nécessaires :

```
sudo apt-get install xfwm4 xfce4-panel xfce4-settings xfce4-session xfce4-terminal xfdesktop4 xfce4-taskmanager tango-icon-theme
sudo apt-get install xfonts-base thunar mousepad
```

L'installation va prendre quelques minutes. Nous installons ici le package **xfwm4** (gestionnaire de fenêtre), les différents programmes associés à **xfce4** qui est notre environnement graphique, les **fonts** (permet l'affichage des caractères en mode graphique), **thunar** (l'explorateur de fichier) et **mouspad** (un

éditeur de fichier). Nous n'avons installé ici que le strict minimum permettant d'avoir un environnement graphique. Comme vous pouvez le voir.

Remarque : Nous n'avons pas besoin d'installer lightdm (pour light Display Manager) lorsque l'on utilise **tightvncserver** puisque le login se fait au travers du mot de passe créé lors de la première connexion au serveur **VNC**. Par contre si vous désirez visualiser votre interface graphique sur un écran HDMI, il faudra installer lightdm.



```
sudo apt-get install lightdm lightdm-gtk-greeter
```

Installer maintenant le serveur vncserver permettant une connexion à la cible via le PC.

```
sudo apt-get install tightvncserver
```

Votre interface graphique et votre serveur graphique sont installées. Vous n'avez plus qu'à rebooter pour prendre en compte l'installation de vos services graphiques.

```
sudo reboot
```

Après le reboot, il ne reste plus qu'à lancer notre serveur VNC sur l'écran 1 (l'écran 0 est déjà utilisé pour la sortie HDMI). Lors du lancement du serveur, il vous sera demandé un mot de passe qui sera utilisé par le client VNC. Un mot de passe trop court est interdit. Mettez **12345678** par exemple.

```
vncserver :1
```

```
You will require a password to access your desktops.
```

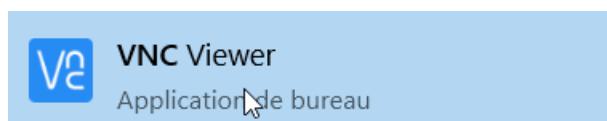
```
Password: 12345678
Verify: 12345678
Would you like to enter a view-only password (y/n)? n
```

Vérifier si le serveur est lancé

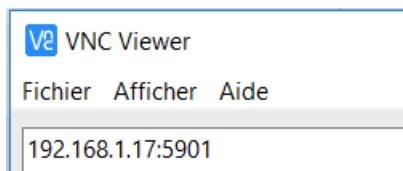
```
ps -ef | grep Xtightvnc
```

```
geii@bp12:~ $ ps -ef | grep Xtightvnc
geii      1585      1  6 08:43 ttys000  00:00:00 Xtightvnc :1 -desktop X -auth /home/geii/.Xauthority -geometry 1024x768 -depth 24 -rfbwait 120000 -rfbauth /home/geii/.vnc/passwd -rfbport 5901 -fp /usr/share/fonts/X11/misc/,/usr/share/fonts/X11/Type1/,/usr/share/fonts/X11/75dpi/,/usr/share/fonts/X11/100dpi/ -co /etc/X11
```

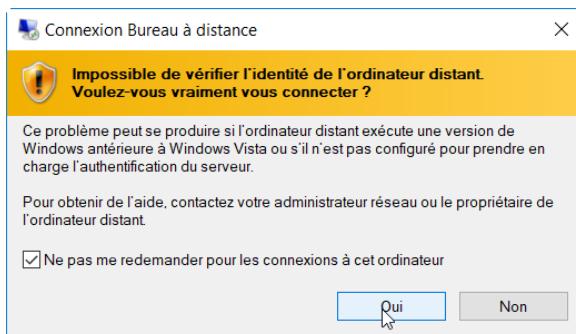
Sur votre PC, lancer le client vncViewer (il faudra peut-être l'installer au préalable)



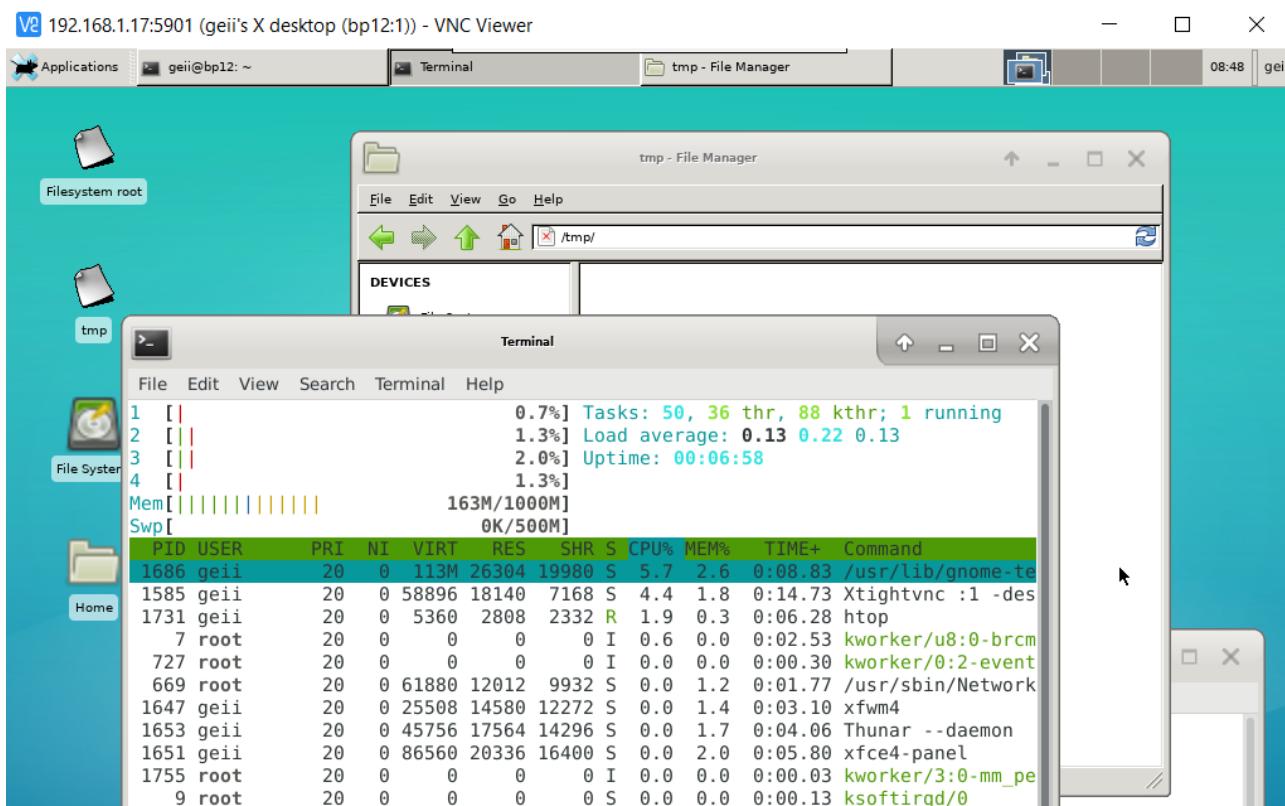
On suppose que votre cible est à l'adresse 192.168.1.17, on pourra aussi mettre le hostname bp1 à la place de l'adresse IP. La connexion se fait par défaut sur le port 5901.



Par défaut, la connexion n'est pas cryptée. Ce n'est pas très gênant lorsqu'on se connecte à un poste sur son propre réseau mais attention si vous accédez à distance à votre bureau car tout va transiter en claire sur internet...



Vous devriez visualiser l'écran déportée de votre cible



Remarque : on pourra arrêter le serveur vnc par la commande :

```
vncserver -kill :1
```

7.4 Manip1 : installation de codeblocks

Nous allons maintenant montrer qu'il est possible d'installer n'importe quel paquet graphique (ici **codeblock**) mais cela pourrait être un client graphique mail (par exemple sylpheed) ou de gestion d'image (Mirage) ou même un navigateur internet(dillo). L'installation de codeblock qui est un IDE de programmation C et C++ nécessite l'installation au préalable d'une chaîne de développement (compilateur, linker, debugger,...), ce qui est déjà le cas sur notre cible. Le paquet qui intègre tous les outils C et C++ se nomme **build-essential**, et il doit être déjà installé sur votre cible.

Vérifier que la chaîne de développement est déjà installé

```
sudo apt list --installed | grep build-essential
```

```
geii@bp12:~/vnc $ sudo apt list --installed | grep build-essential
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
build-essential/bionic,now 12.4ubuntu1 armhf [installed]
```

Si ce n'est pas le cas, installer le paquet build-essential dans lequel se trouve gcc, gdb,g++,...

```
sudo apt install build-essential
```

Vérifier quels sont les paquets disponibles ayant un lien avec codeblock

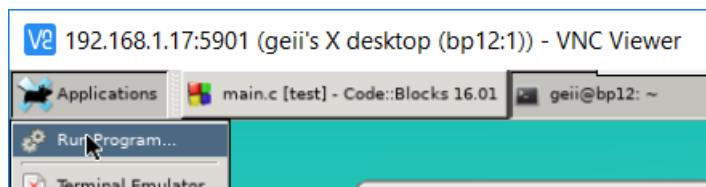
```
sudo apt-cache search codeblock
```

```
geii@bp12:~/vnc $ sudo apt-cache search codeblock
codeblocks - Code::Blocks integrated development environment (IDE)
codeblocks-common - common files for Code::Blocks IDE
codeblocks-contrib - contrib plugins for Code::Blocks IDE
codeblocks-dbg - Code::Blocks debugging libraries
codeblocks-dev - Code::Blocks development files (SDK)
libcodeblocks0 - Code::Blocks shared library
```

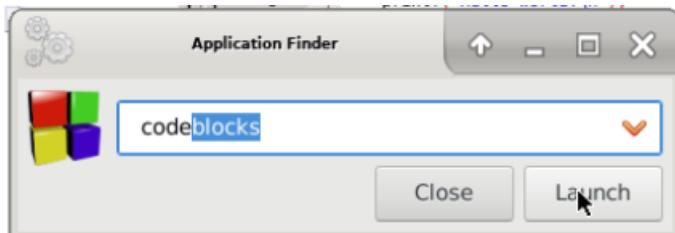
Puis installer codeblock

```
sudo apt install codeblocks
```

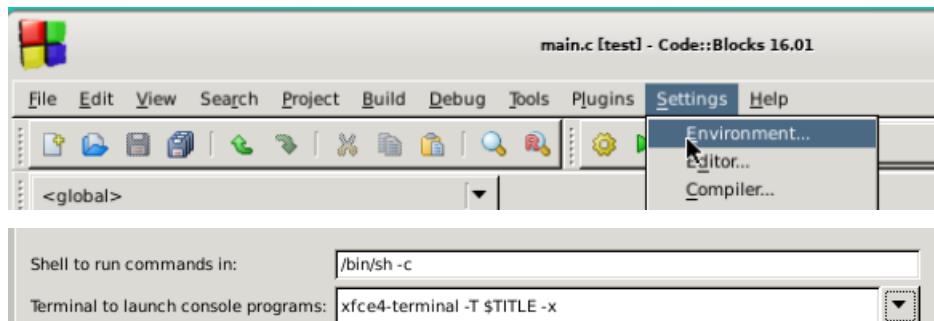
L'installation de codeblocks peut être faite dans une console putty ou bien dans une console de votre interface graphique. Une fois codeblocks installé, vous pouvez le lancer avec le lanceur d'application (Run Program)



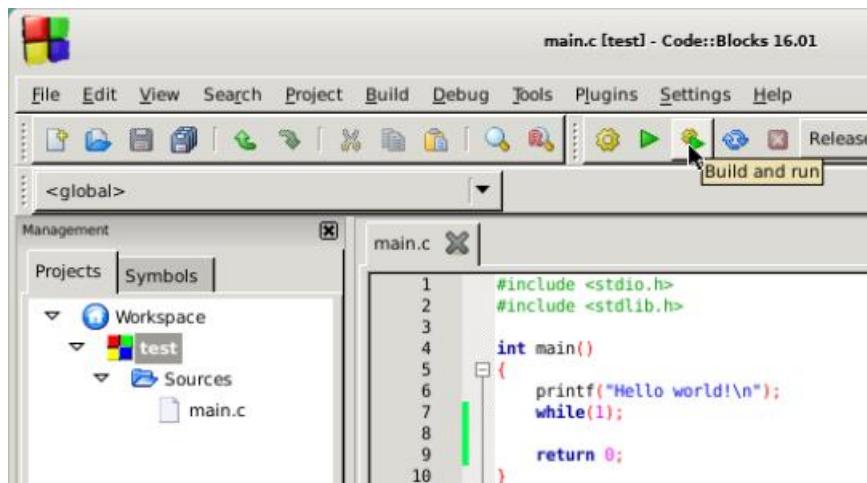
Puis chercher codeblocks



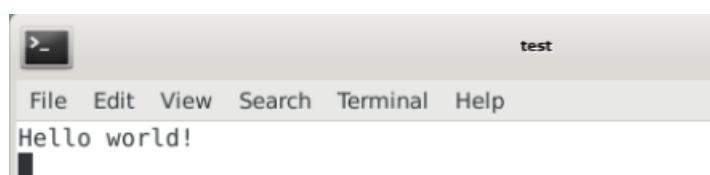
Une fois codeblocks lancé, vous pourrez configurer l'environnement pour activer la bonne console lors du lancement du programme (Settings Environnement, choisir xfce4-terminal qui est la console par défaut sous xfce4)



Une fois cette manip faites, allez dans File, New Project, Console application. Puis compiler et lancer le programme. Nous allons tester que notre IDE codeblocks



Vous devriez voir apparaître la console gnome-terminal avec le résultat du programme.



Notre premier programme C utilisant codeblock comme IDE est fonctionnel.

7.5 Manip 2 : Installation de cheese

Nous allons le voir, l'installation d'une webcam USB est assez simple. Cette manip permet de montrer la facilité d'installation d'un paquet (ici cheese) logiciel de visualisation de flux video.

- ✓ *Installer cheese qui est une application de visualisation de flux video. Puis lancer cheese.*
- Insérer une webcam. Vérifier que le flux video de la webcam est bien visualisée sur cheese.*



Toute la suite des manips sur l'interface graphique permet de montrer ce que veulent dire les notions de serveur graphique (Xwindows), d'environnement graphique en comparant 2 environnements (lxde et xfce4) et ce qu'est un client graphique. Ces manips ne sont pas fondamentales pour la suite, vous pouvez donc les passer si votre but est de rentrer dans le vif du sujet, à savoir la prise en main de votre cible et la connaissance des commandes de bases Linux.

7.6 Manip 3 , changement de l'environnement graphique

Nous allons maintenant installer un deuxième environnement graphique : **lxde**.

Installer lxde

```
sudo apt install lxde
```

Pour l'installation, le gestionnaire de fenêtre (même si lxde a été installée) est toujours xfce4.

Nous avons 2 solutions pour changer le gestionnaire

- ✓ Utilisation de la commande **update-alternatives --config x-session-manager**
- ✓ Modification du fichier de démarrage de l'interface graphique **xstartup**

Commençons par la première solution : taper la commande et choisir lxde.

```
sudo update-alternatives --config x-session-manager
```

Selection	Path	Priority	Status
*	/usr/bin/startxfce4	50	auto mode
1	/usr/bin/lsession	49	manual mode
2	/usr/bin/openbox-session	40	manual mode
3	/usr/bin/startlxde	50	manual mode
4	/usr/bin/startxfce4	50	manual mode
5	/usr/bin/xfce4-session	40	manual mode

Press <enter> to keep the current choice[*], or type selection number: 3

Choisir startlxde (choix numéro 3) pour changer d'environnement graphique.

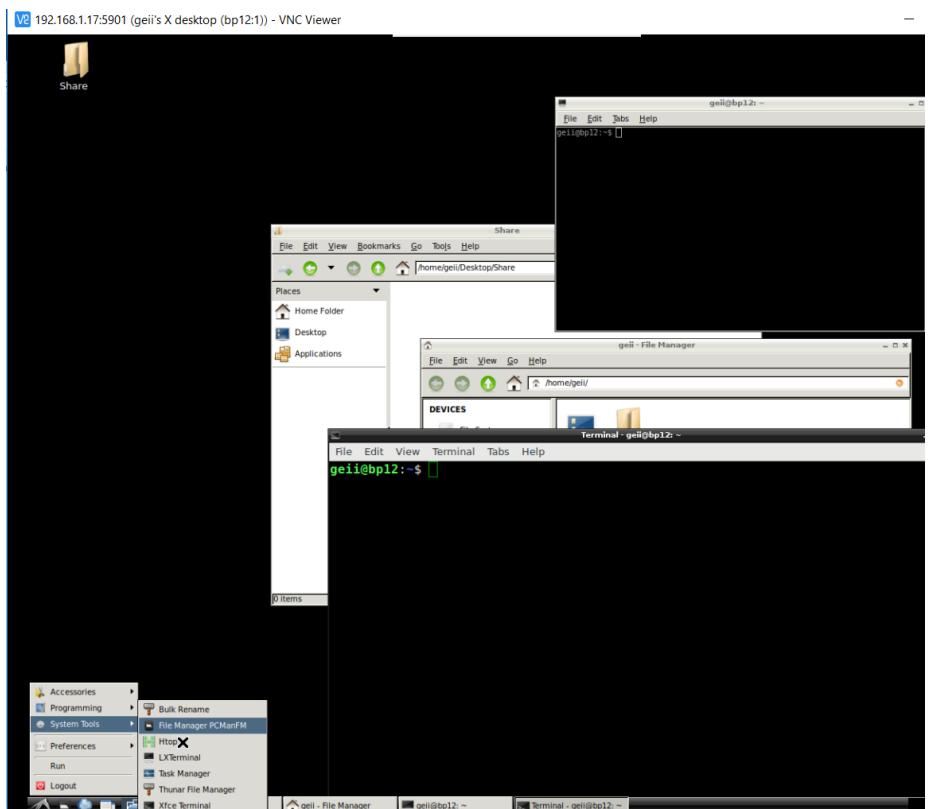
Arrêter le serveur graphique

```
Vncserver -kill :1
```

Puis relancer le serveur graphique pour prendre en compte la modification. Lors du lancement du serveur, vous pouvez choisir la taille de l'affichage, pour pourrez essayer entre les 2 solutions ci-dessous :

```
vncserver -geometry 1280x1024 :1
vncserver -geometry 1440x900 :1
vncserver -geometry 1920x1080 :1
```

Vérifier que votre environnement a changé (c'est maintenant lxde qui est lancé).



Remarque : avec lxde, les applications PCManFM (explorateur de fichier), LXTerminal (console) et Leafpad (Editeur de texte) ont été installées. Mais nous avons aussi les applications installées pour l'environnement xfce4 puisque nous pouvons aussi utiliser Xfce Terminal, Thunar File Manager ou MousePad (éditeur), applications qui font partie de l'environnement graphique xfce4. Par contre, les menus sont différents ainsi que l'environnement graphique. Ouvrez quelques applications afin de vérifier le bon fonctionnement de votre nouvelle interface graphique.

Remarque : nous aurions aussi pu modifier le fichier xstartup appelé lors du lancement du serveur vnc dans le répertoire **/home/geii/.vnc**

```
nano /home/geii/.vnc/startup
```

```
geii@bp12:~/.vnc $ nano xstartup
```

Et remplacer la dernière ligne du fichier **/etc/X11/Xsession** par **lxsession -e LXDE -s Lubuntu**

Remarque : lorsque vous lancez votre serveur de bureau à distance avec la commande **vncserver**, celui-ci attend une connexion d'un client vnc (ici **vncviewer**). Une fois la connexion établie, **vncserver** lance le serveur graphique **Xorg** et exécute les commandes se trouvant dans le fichier **/home/geii/.vnc /startup**. Ces commandes vont alors lancer votre environnement graphique.

7.7 Manip 4 : serveur graphique et clients graphiques

Dans cet exercice, nous allons tester différentes solutions de lancement d'applications graphiques et voir le résultat sur VNCViewer. Le but est ici de comprendre les notions de serveur graphique X11, de gestionnaire de fenêtre, de client graphique et d'environnement graphique.

Ce premier exercice consiste à enlever le lancement de l'environnement graphique au démarrage. Le fichier xstartup ressemblera à ca. Ou même vous pouvez tout commenter. Ainsi aucune application ne sera lancée à l'exception de Xorg (ou X11) le serveur graphique.

```
geii@bp12:~/.vnc $ nano xstartup
GNU nano 2.9.3                               xstartup

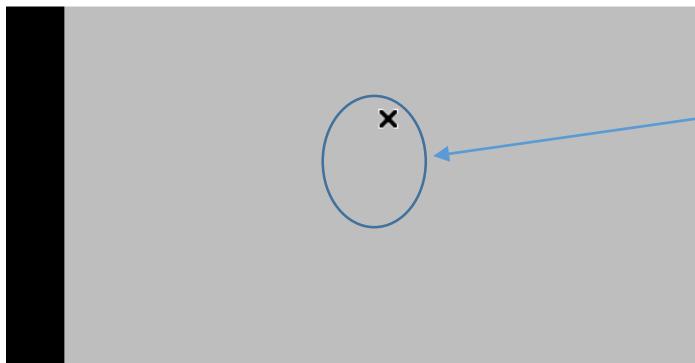
#!/bin/sh

xrdb $HOME/.Xresources
xsetroot -solid grey
#x-terminal-emulator -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
#x-window-manager &
# Fix to make GNOME work
export XKL_XMODMAP_DISABLE=1
#lxterminal &
#/usr/bin/lxsession -s LXDE &
```

Arrêter le serveur vnc et le relancer pour prendre en compte la modification de xstartup.

```
vncserver -kill  :1
vncserver  :1
```

VNC 192.168.1.17:5901 (geii's X desktop (bp12:1)) - VNC Viewer

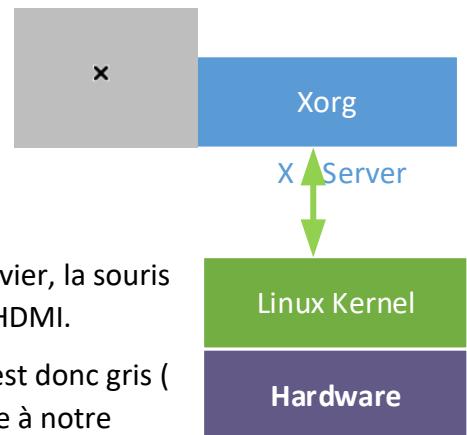


C'est le serveur Xorg qui gère la souris. Par défaut l'icone de la souris est une croix.

Le schéma actuel de notre test peut être symbolisé par le schéma ci-contre. Sauf que dans notre manip, le flux vidéo n'est pas envoyé vers la sortie HDMI par le kernel (noyau) mais au serveur VNC qui l'envoie via le réseau au client VNC Viewer.

Comme on peut le voir c'est le noyau et Xorg qui gèrent le clavier, la souris et l'écran virtuel dans notre cas ou réel si l'on a un écran sur le port HDMI.

Par contre, le serveur Xorg ne gère rien de plus. Notre écran est donc gris (à cause de la commande xsetroot –solid grey) avec une croix associée à notre souris.



Continuons le test, et ajoutons au lancement les programme Codeblocks et Thunar, sans lancer le gestionnaire de fenêtre xfwm4 ou l'environnement graphique Ixde (qui englobe le gestionnaire de fenêtre et les autres programmes de gestions des icônes,...).

Modifier votre fichier xstartup pour lancer seulement 2 programmes graphiques codeblocks et Thunar qui est un client graphique classique. Votre fichier xstartup est donc constitué de 2 lignes :

```
codeblocks &
Thunar &
```

```
nano /home/geii/.vnc/startup
```

```

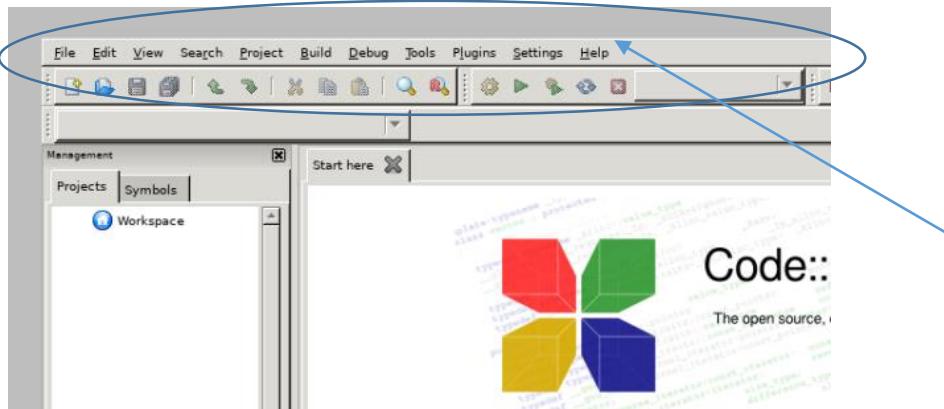
GNU nano 2.7.4
File:

#!/bin/sh

#xrdb $HOME/.Xresources
#xsetroot -solid grey
#x-terminal-emulator -geometry 80x24+10+10
#x-window-manager &
# Fix to make GNOME work
#export XKL_XMODMAP_DISABLE=1
#/etc/X11/Xsession
codeblocks &
Thunar &
```

Arrêter vnc serveur et le relancer.

```
vncserver -kill :1  
vncserver:1
```



Impossible de déplacer les 2 fenêtres clientes qui ont été lancées. Il manque le gestionnaire de fenêtre.

Dans la console, tuez codeblocks

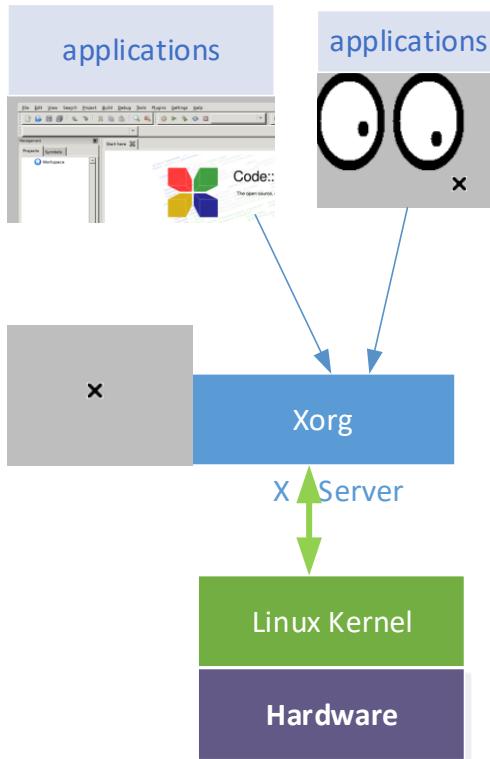
```
sudo killall codeblocks
```

Vous devriez voir apparaître Thunar (l'explorateur de fichier), Thunar était caché par codeblocks.

Remarque : Les 2 applications ont bien été lancée mais Xorg ne permet pas de gérer les fenêtres clientes. Xorg est un serveur graphique permettant de gérer l'affichage bas niveau, gère la souris et le clavier mais ne fait pas la gestion des applications clientes.

Remarque : la commande codeblocks & permet de lancer codeblocks en tache de fond, le programme codeblocks est lancé et on peut récupérer l'utilisation de la console.

On peut résumer la manip que l'on vient de faire à ce schéma. Les 2 applications clientes sont visibles car gérées par Xorg. Par contre, impossible de les déplacer. Nous allons ajouter un gestionnaire de fenêtres afin de vérifier l'importance d'un gestionnaire de fenêtre comme xfwm4 ou openbox.



Modifier votre fichier xstartup pour lancer les 2 programmes graphiques codeblocks et Thunar et aussi le gestionnaire de fenêtre xfwm4. Le fichier xsartup est donc constitué de 3 lignes :

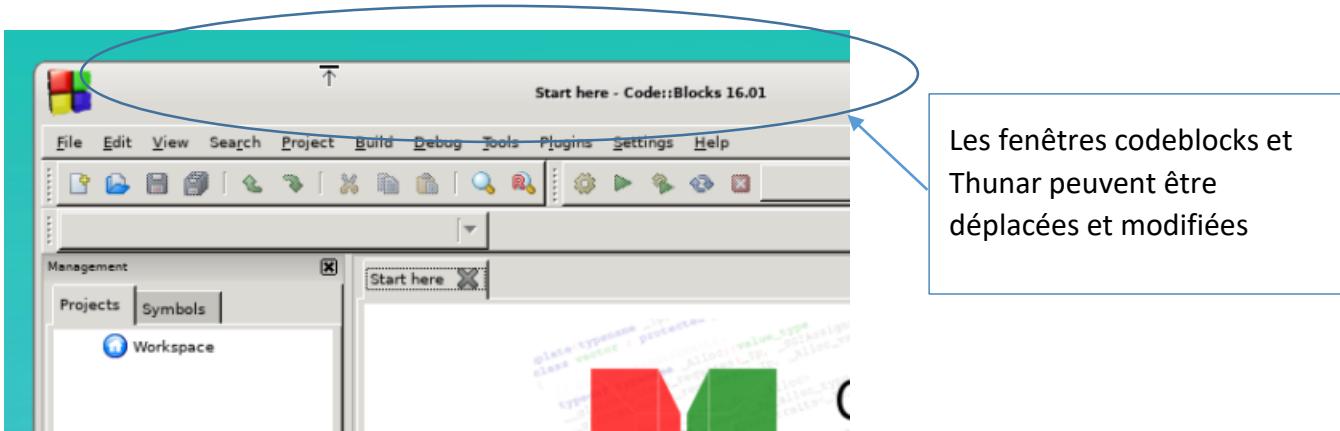
```

codeblocks &
Thunar &
xfwm4 &
  
```

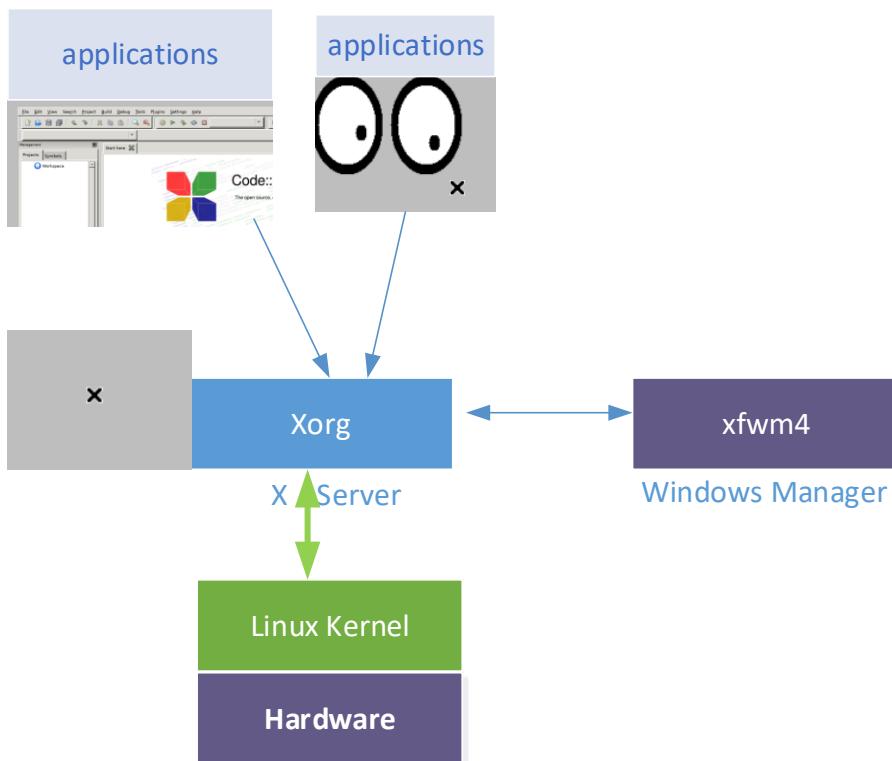
Arrêter vnc serveur et le relancer.

```

vncserver -kill :1
vncserver :1
  
```



Vérifier que l'on peut maintenant déplacer et redimensionner la fenêtre codeblocks et ce grâce au gestionnaire de fenêtre. La figure ci-dessous permet de comprendre ce que nous avons utilisé dans ce test et les interactions entre les différents éléments.



Exercice : modifier le fichier xstartup et tester le gestionnaire de fenêtre openbox utilisé par l'environnement graphique Ixde (remplacer la ligne xfwm4 & par openbox &). Vérifier que le gestionnaire openbox fonctionne aussi bien de xfwm4.

Pour finir nous allons ajouter en fin de fichier lancement de l'environnement graphique Xsession qui comporte par défaut le lancement du gestionnaire de fenêtre xfwm4 (nous n'avons donc pas besoin de la lancer avant de lancer Xsession). Les applications lancés avant seront automatiquement ouvertes au lancement de notre environnement.

Avant de terminer cette manip, nous allons installer quelques applications graphiques simples et historiques (clients bas niveaux Xorg).

```
sudo apt install x11-apps
```

Remarque : Nous avons installé xeyes qui est inclus dans le paquet x11-apps et qui suit la position de la souris. Nous allons lancé ce client à la place de Thunar (nous pourrions lancé aussi Thunar dans le fichier xstartup).

Modifier le fichier xstartup comme ci-dessous

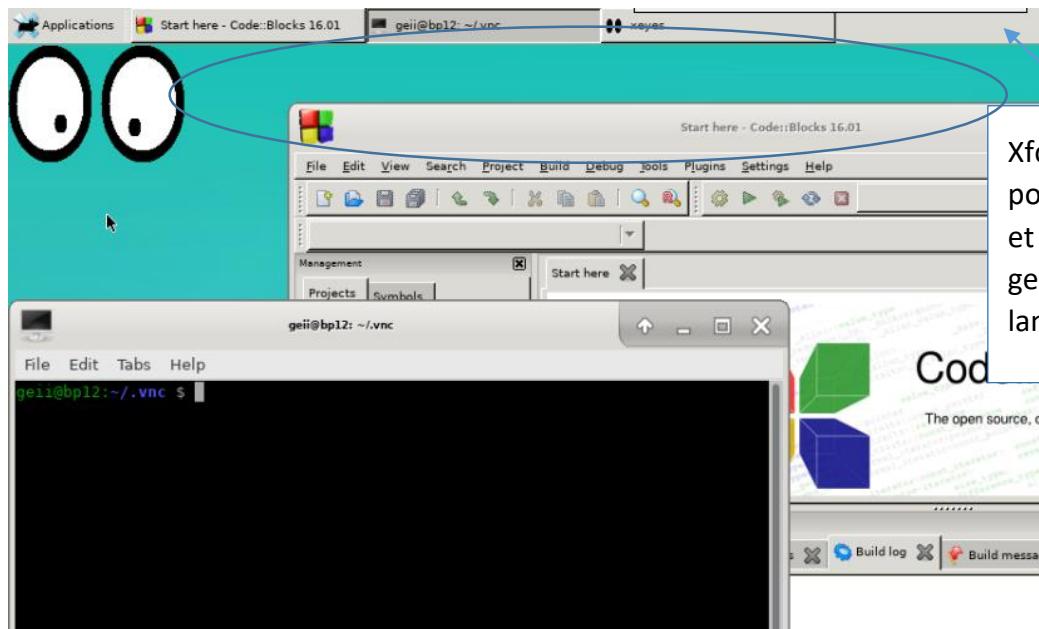
```
nano /home/geii/.vnc/xstartup
```

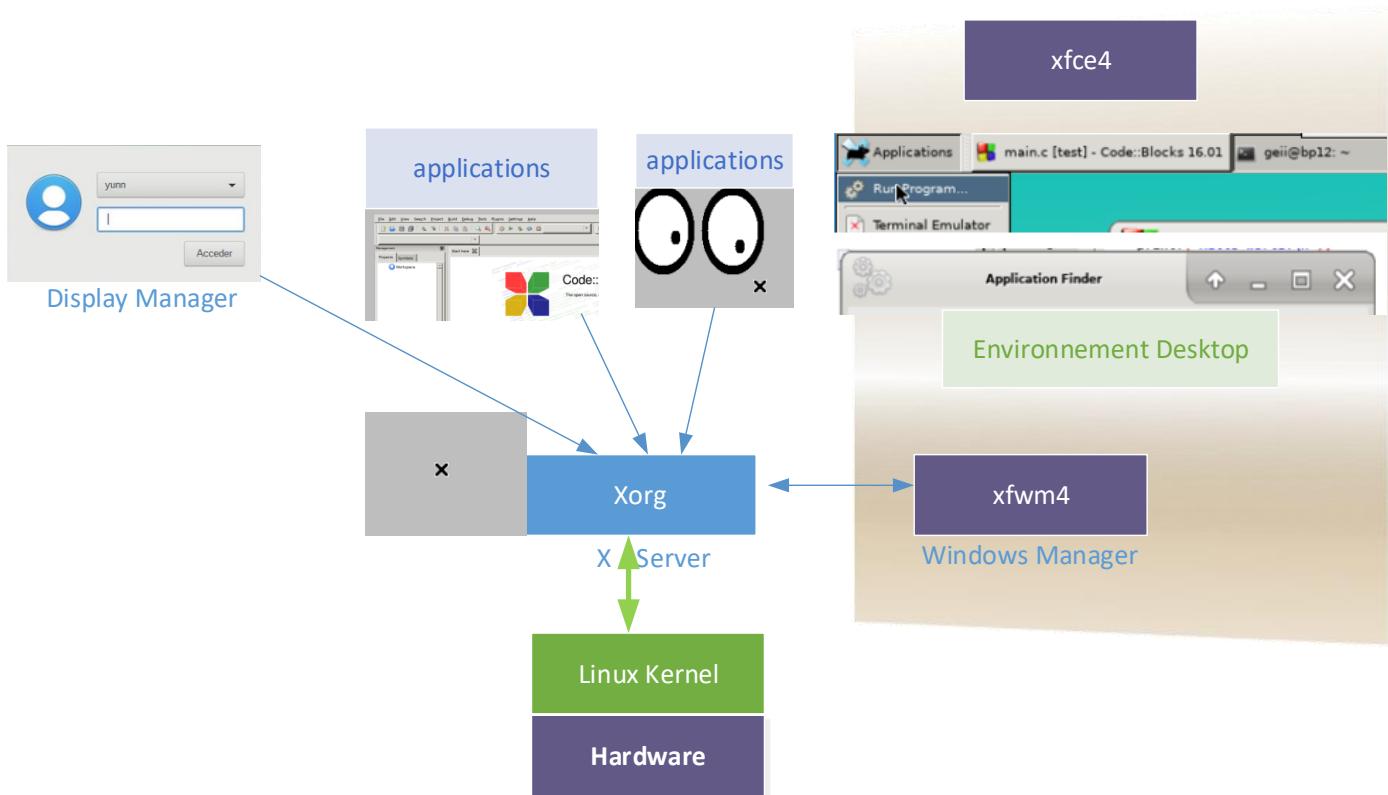
```
#!/bin/sh

xrdb $HOME/.Xresources
xsetroot -solid grey
#x-terminal-emulator -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
#x-window-manager &
# Fix to make GNOME work
export XKL_XMODMAP_DISABLE=1
lxterminal &
codeblocks &
xeyes &
/etc/X11/Xsession &
```

Arrêter vnc serveur et le relancer.

```
vncserver -kill :1
vncserver :1
```





Voici donc le schéma de principe global. Le fait de lancer Xsession en fin du fichier xstartup permet de lancer xfce4 (ou lxde), environnement graphique qui permet ensuite de configurer le bureau, lancer les applications avec le principe des menus ou ouvrir des fichiers avec l'explorateur.

8 Exercices Linux utilisateurs et administration système

8.1 Introduction : le site <https://linuxjourney.com/>

Nous allons maintenant prendre en main notre cible et notamment comprendre les différentes commandes Linux, le système de fichier, le noyau et toutes les notions importantes permettant la prise en main de notre cible. Le cours et les exercices Grasshopper sont sur <https://linuxjourney.com/>. Après chaque chapitre, répondre aux questions donnés ci-dessous

8.2 Partie Getting started et Command Linux

Questions	Réponses
C'est quoi UNIX ?	
Quant a été développé Linux ?	
Qui a développé Linux ?	
Aller sur https://upload.wikimedia.org/wikipedia/commons/a/af/Android-System-Architecture.svg et sur https://en.wikipedia.org/wiki/Linux_kernel_interfaces#/media/File:Linux_API.svg	
C'est quoi le kernel Linux (ou noyau) qui est utilisé par les distributions Linux et Android ?	
Donner 2 exemples de distributions Linux	
C'est quoi un paquet ou package ?	
Créer un répertoire test dans votre home et placez-vous dans ce répertoire.	
Créer le fichier vide test_bash_hello	
Ecrire quelque chose dans le fichier Hello	
Afficher le contenu du fichier Hello	
Que renvoie la commande whoami ?	
Que veut dire la commande pwd ?	
Que fait la commande cd / ?	
Et la commande cd ?	
Et la commande cd .. ?	
Donner 2 fichiers cachés dans votre home ?	
Que fait la commande ls -R et ls -t ?	

Questions	Réponses
Donner la commande permettant d'afficher les attributs de tous les fichiers et répertoire de votre home.	
Donner la commande permettant de créer un fichier vide	
Que renvoie la commande <code>file /bin/ls</code> ?	
Que renvoie la commande <code>file ~/.bashrc</code> ?	
Que renvoie <code>cat .config .bash_logout</code> ?	
Dans le bash, à quoi servent les flèches haut et bas ?	
Que fait la commande <code>history</code> ?	
Donner la commande permettant de copier le répertoire <code>test</code> (et son contenu) dans un autre répertoire <code>test_copie</code> dans votre home	
Créer le fichier <code>mon_test_vide</code> dans le répertoire <code>test</code> et le déplacer dans le répertoire <code>test_copie</code> .	
Effacer le répertoire <code>test_copie</code>	
Que fait la commande <code>find /bin /usr/bin -name ls</code>	
Que fait la commande <code>find / -type d -name test</code>	
Et la commande <code>sudo find / -type d -name test</code>	
Donner 2 solutions pour trouver de l'information sur la commande <code>cp</code>	

8.3 Partie Text Fu

Questions	Réponses
<p>On tape à la suite les commandes</p> <pre>echo test > rm ls more rm rm rm</pre> <p>Expliquer</p>	
Donner la différence entre > et >> dans le bash	
Donner 2 solution pour créer un fichier vide	
Que veut dire la commande	
<code>cat < peanuts.txt > banana.txt</code>	
En utilisant nano, ecrire bonjour dans le fichier peanuts.txt, copier le contenu de peanuts.txt dans banana.txt avec la commande cp	
Faire la même chose en utilisant >	
Que fait la commande	
<code>ls /fake/directory 2> error.log</code>	
Quel est le résultat de la commande	
<code>ls /fake/directory 2> /dev/null</code>	
Que fait la commande	
<code>ls -la /etc less</code>	
Et la commande	
<code>more -la /etc less</code>	
Quelle est la commande permettant de connaître vos variables d'environnement ?	

Questions	Réponses
ajouter le paquet zsh qui est un autre shell <code>sudo apt install zsh</code>	
Quel est votre shell par défaut ?	
Lancer la commande <code>chsh -s \$(which zsh)</code>	
Arrêter putty et le relancer puis taper 2 (installation par défaut du nouveau shell) Expliquer (vérifier votre nouveau shell par défaut)	
Que fait la commande <code>head -n 15 /var/log/syslog</code>	
Que fait la commande <code>tail /var/log/syslog</code>	
A quoi sert la commande <code>sort</code> ?	
A quoi sert la commande <code>wc</code> ?	
Que fait la commande <code>env grep SHELL</code>	
Que fait la commande <code>grep alias ~/.bashrc</code>	
Comment connaître le nombre de ligne du fichier <code>/etc/passwd</code>	
Que renvoie la commande <code>pwd</code> ?	
Que renvoie <code>echo \$PWD</code>	
Quand on lance une commande (par exemple <code>ls</code>) quelle est la variable d'environnement utilisée pour chercher la commande dans les répertoires par défaut ?	

8.4 Partie user management

Questions	Réponses
L'utilisateur geii peut-il lire le contenu de /etc/shadow ? Expliquer	
Quel utilisateur peut visualiser ce fichier ?	
L'utilisateur geii peut-il lire contenu de /etc/passwd? Expliquer	
Lancer la commande grep ^sudo /etc/group	
Qui fait parti du groupe sudo ?	
Si on efface l'utilisateur du groupe sudo, cet utilisateur pourra-t-il lancer la commande sudo ?	
Que fait la commande sudo su ?	
Quel est le shell du root ?	
Taper CTR+D ou logout pour sortir de la session root et se reconnecter en root. Quelle est votre variable SHELL d'environnement ?	
Lancer la commande more /etc/passwd	
Quel est le shell du root et de geii ?	
Créer un utilisateur bob	
Changer son mot de passe	
Quels sont les fichiers dans /etc mis à jour ?	
Utiliser la commande grep pour visualiser les informations relatives à bob dans les 3 fichiers	
Se déconnecter et se reconnecter en utilisant le login de bob	
Quel est le shell de bob ?	
Se reconnecter en utilisant le compte geii et supprimer le compte de bob	
L'utilisateur geii peut-il modifier le fichier /etc/group ? Expliquer.	

8.5 Permission

Questions	Réponses
Créer un fichier test.txt . Quel sont vos droits sur ce fichier ?	
L'utilisateur bob peut-il visualiser votre fichier ?	
Vérifier en créant l'utilisateur bob et en vous connectant sur une deuxième console putty avec l'utilisateur bob	
Modifier les droits du fichier test.txt pour que seul l'utilisateur geii puisse visualiser et modifier ce fichier.	
Existe-t-il une différence entre les 2 commandes : chmod 600 test.txt et chmod go-rw test.txt ?	
Aller dans le répertoire home. Combien il y a de fichier ? C'est fichiers sont-ils des répertoires ? Justifier.	
Changer le propriétaire et le groupe du fichier test.txt pour qu'il appartienne à bob :bob	
L'utilisateur geii peut-il maintenant lire le contenu du fichier ? tester.	
Et en utilisant la commande sudo ? tester.	
Lors de la création d'un fichier celui-ci est de type rw-r--r-- (Un fichier n'est jamais exécutable lors de sa création). Quelle est la commande qui permet de changer les permissions par défaut ?	
Le fichier /usr/bin/passwd a des droits spéciaux. Expliquer.	
Le répertoire /tmp a des droits particuliers. Si geii créer un fichier. Celui-ci pourra-t-il être effacé par bob ?	
Enlever le sticky bit du répertoire tmp . En tant que geii.	
Avec le login de bob effacer le fichier. Expliquer.	
Ajouter le sticky bit sur le répertoire /tmp . Tester.	

8.6 Processe

Questions	Réponses
En utilisant la commande htop donner 3 services actuellement lancés .	
Expliquer le résultat de la commande <code>ps aux grep ssh</code>	
Que veut dire <code>kill -15 1234</code> ?	
Que fait la commande <code>killall systemd</code> ? expliquer.	
Le répertoire /proc est un répertoire système. Visualiser le process 1 : <code>more /proc/1/status</code> . Quel est ce process ?	
Visualiser les informations dans /proc/cpuinfo et /proc/meminfo	
Combien il y a de support de masse ? <code>more /proc/partitions</code>	
Ecrire dans le fichier test.sh : <code>while true ;do echo bonjour sleep 5 done</code>	
Rendre le fichier exécutable : <code>chmod+x test.sh</code>	
Lancer le fichier test.sh : <code>./test.sh</code>	
Arrêter le programme (CTR+C) et le relancer en arrière plan (&)	
Le replacer au premier plan.	
En utilisant la commande ps l, donner les différents process (père et fils) associé au lancement de test.sh	
Passer le programme en arrière plan	
Tuer le process avec le signal 15	

8.7 Packages

Questions	Réponses
Compresser votre répertoire en zip <code>tar czvf /home/geii/geii.tar.gz /home/geii</code>	
déplacer le fichier geii.tar.gz dans le répertoire /tmp	
Dézipper le répertoire dans /tmp	
Lister les paquets installés sur votre cible	
Visualiser le détail du paquet ssh <code>dpkg -L ssh. Existe-t-il un fichier de config ?</code>	
Quelle est la différence entre dpkg et apt ?	
Que fait la commande <code>dpkg -l grep ssh</code>	
Pour visualiser les paquets installables ayant un lien avec mp3 : <code>apt-cache search mp3</code>	
Donner un nom de serveur mp3	
Pour savoir pour n'importe quel fichier de quel dépôt il dépend (ici ntp.conf) <code>dpkg -S /etc/ntp.conf</code>	
Visualiser les fichiers du paquet ntp	
Pour avoir les informations sur les dépendances du paquet systemd <code>apt-cache show ntp</code>	
Comment installer le compilateur C et tous ses outils sur la cible. ce compilateur est-il installé ?	
Suivre l'installation de pidgin https://www.howtogeek.com/105413/how-to-compile-and-install-from-source-on-ubuntu/	
Que fait la commande <code>sudo apt autoremove</code> ?	
Que fait la commande <code>sudo apt purge</code> ?	

8.8 Device

Questions	Réponses
Lancer la commande <code>ps aux grep tty</code> Donner le nom des 4 consoles. Expliquer sachant qu'il y a 4 consoles (une sur la sortie HDMI, une sur la sortie UART TTL, une sur le connecteur micro-usb et une si vous êtes connectés via ssh)	
Quels sont les drivers (Major Number) associés aux différentes consoles. (aller dans le répertoire /dev)	
Ces consoles sont-elles de type caractère ou block ? Expliquer.	
Lancer la commande <code>ls /dev/mm*</code> Ces devices correspondent aux 2 supports flash EMMC (sd card et emmc du bananapi). Quel est le numéro du driver qui gère ces 2 supports ?	
Ce driver est-t-il de type caractère ou block ? Expliquer.	
Lancer la command <code>df -h grep mm</code> (La commande df (disk free) permet d'afficher à l'écran la taille de l'espace disque occupée et la taille de l'espace disque libre). En déduire quel numéro de mmc correspond à la SD ?	
Combien il y a de partition sur le emmc interne ?	
Existe-t-il un périphérique de type disque dur (<code>sd*</code> ou <code>hd*</code>) ?	
Insérer une clé usb sur un des port USB A de la cible. Existe-t-il un périphérique de type disque dur (<code>sd*</code> ou <code>hd*</code>) ?	
Lancer la commande <code>tail /var/log/syslog</code> . Expliquer	
Lancer la commande <code>ls /sys/block</code> expliquer	

Quelques explication : Loop disk permet à une zone de ram d'être vu comme un disque dur (utiliser pour monter des disques iso par exemple) ; ram disk est une zone en mémoire qui est utilisé pour monter un système de fichier (par exemple /var/log) qui sera effacer après un boot (zone en RAM). Zram disk , est la même chose que ram disk mais compressé (pour gagner de la place).

Questions	Réponses
Lancer la commande <code>lsusb</code>	
Quel est le nom de la clé usb qui a été reconnu ?	
Lancer la commande <code>lspci</code> . Expliquer (notre cible est une cible arm et non x86...)	
Lancer la commande <code>lsscsi</code> . Installer le package si nécessaire.	
En utilisant <code>lsscsi -h</code> trouver l'option qui permet de connaitre la taille de la clé USB. Quelle est cette taille ?	
Lancer la commande <code>udevadm info --query=all --name=/dev/mmcblk1</code>	

Aller sur <https://doc.ubuntu-fr.org/dd> et répondre aux questions :

Questions	Réponses
Donner la commande permettant de sauvegarder tout le contenu de la sd mmcblk0 vers le emcc interne mmcblk1	
Visualiser le contenu d'une image iso .	
Effacer toutes les données de la clé USB	
Que fait la commande <code>dd if=/dev/sda2 (root) of=/home/user/root.img bs=4096 conv=notrunc,noerror</code>	

Exécuter les commandes suivantes et visualiser la led rouge sur la cible. Expliquer. Comment rallumer la led rouge se trouvant sur la cible ? Comment lire la température du processeur ?

```
cd /sys/class/leds/bananapi-m2-plus\:red\:pwr
sudo su
echo none > trigger
echo 0 > brightness
echo 1 > brightness
more /sys/class/thermal/thermal_zone0/temp
```

Comment arrêter ou activer un des 4 processeurs de notre carte ?

Arrêt du CPU3

```
echo 0 > /sys/devices/system/cpu/cpu3/online
```

Vérifier dans /proc/cpuinfo (le cpu3 n'apparaît plus)

8.9 File System

Questions	Réponses
Rappeler la commande pour ajouter l'utilisateur bob. Où se trouve son répertoire ?	
Allez dans le répertoire /boot quelle est la taille du noyau zImage ? (ls -l /boot)	
Où se trouvent les fichiers de configuration des services.	
Dans quel répertoire on peut trouver les logs des services et du noyau ?	
Les répertoires /proc et /sys sont des répertoires particuliers. A quoi servent ces répertoires.	
Qu'a-t-on dans les répertoires /bin et /usr/bin ?	
Qu'a-t-on dans les répertoires /sbin et /usr/sbin ?	
Quelle est la différence entre ces répertoires et les fichiers qu'ils hébergent	
Qu'a-t-on dans /lib et /usr/lib ?	
Lancer la commande df -hT . C'est quoi ext4fs ? Expliquer la notion de journalisation.	
Lancer la commande gparted -l. Expliquer	<pre>Model: SD 00000 (sd/mmc) Disk /dev/mmcblk0: 7861MB Sector size (logical/physical): 512B/512B Partition Table: msdos Disk Flags: Number Start End Size Type File system Flags 1 4194kB 7704MB 7700MB primary ext4</pre>
Expliquer la notion de partition.	

Questions	Réponses
<p>Essayer de créer 2 partitions sur la mmc interne La première partition fera 10Mo et la deuxième le reste. Puis formater la première partition en ext4 Puis monter cette partition sur /mnt Et copier un fichier sur cette partition</p>	
<p>Rebooter. La partition est-elle montée par défaut ?</p>	
<p>Modifier le fichier fstab afin d'ajouter le montage automatique.</p>	
<p>Tester et rebooter</p>	
<p>Avez-vous une partition de swap sur votre cible ? A quoi sert le swap ?</p>	
<p>Que fait la commande du –h et df -h</p>	
<p>A quoi sert la commande fsck ?</p>	
<p>C'est quoi un inode ?</p>	
<p>C'est quoi un lien symbolique et a quoi ca sert ?</p>	

8.10 Récapitulatif et compléments

8.10.1 Résumé de commandes utilisateurs et systèmes

commandes	explications	exemples
nano -c	Editeur avec affichage du numéro de ligne	<code>nano -c test.txt</code>
pwd	Path directory (Where we are)	<code>pwd</code>
whereis	Where is executable	<code>whereis armbian-config</code>
ls	- a shows all (including hidden files) - l displays long format - R gives a recursive listing - r gives a reverse listing - t sorts last modified - S sorts by file size - h gives human readable file sizes	<code>ls -al</code> <code>ls -lt</code>
cd	Change directory <code>cd</code> <code>cd /</code> pour aller à la racine <code>cd ..</code> pour remonter d'un cran	<code>cd ../../test</code>
mkdir	Make directory	<code>mkdir rep</code>
rm	Delete a file - r recursive delete (use for directories) - d remove empty directories	<code>rm -r mon_rep/</code>
cp	- r recursive copy	<code>cp test.txt test1.txt</code>
mv	Move file (déplace un fichier ou un répertoire)	<code>mv test.txt test1.txt</code> <code>mv rep/ rep1/</code>
touch	Créer un fichier vide	<code>touch test.txt</code>
cal	Renvoie le calendrier	<code>cal</code>
date	Renvoie la date courante	<code>date</code>
alias	Renvoie les alias par défaut. Il est possible de modifier les alias en modifiant le fichier .bashrc	<code>alias</code>
env	Renvoie les variables d'environnement	
echo	Affiche bonjour Affiche le contenu de la variable d'environnement Créer un fichier mon_file et met « ajout » devant Ajoute au fichier mon_file la chaîne de caractères « ajout »	<code>echo "bonjour"</code> <code>echo \$PATH</code> <code>echo "ajout" > mon_file</code> <code>echo "ajout" >> mon_file</code>

dpkg - reconfigure	Reconfigure un paquet déjà installé	dpkg - reconfigure tzdata (pour reconfigurer la time zone)
dpkg	-l Liste tous les paquets installés -L liste tous les fichiers du paquet	dpkg -L wget
apt-cache	search xxx : Cherche tous les paquets ayant le nom xxx	apt-cache search wget
apt-get ou apt	install pour installer un paquet remove pour désinstaller un paquet tout en gardant les fichiers de configuration purge pour désinstaller complètement (avec les fichiers de configuration) update : mise à jour du cache upgrade : mise à jour des paquets	apt-get install bluez apt purge bluez
ln	-s créer un lien symbolique	ln -s test test-associated
chmod	Change les droits des fichiers User Group other 421 421 421 rwx rwx rwx	chmod ug+w test2.txt chmod 664 test2.txt chmod u=rw, g=rw, o=r test2.txt
chown	Change file owner. Change owner and group at the same time. Recursively change ownership of /tmp/test .	chown geii a.txt chown geii:users a.txt chown -R geii /tmp/test
adduser	Ajout d'un utilisateur	useradd bob
usermod	Modifie les propriétés du compte geii (ici ajoute le groupe dialout au compte geii)	sudo usermod -a -G dialout geii
groups	Affiche les groupes auquel appartient l'utilisateur courant	
umask	Change le masque par défaut appliqué lors de la création d'un fichier ou d'un répertoire umask 022 ; touch test -> test aura les droits 755	umask 022 umask u=rwx, g=rx, o=rx
lsblk	Connaitre les disques montés (plus simple à lire que la commande mount)	lsblk
lsusb	Connaitre les différents équipements usb reconnus	lsusb
cat more less	3 commandes pour visualiser un fichier	cat /more/proc/partitions
du	Disk usage -h (human) -s (total)	du -hs /etc
df	Liste tous les systèmes de fichiers mais en plus simple que mount	df -h
mount	Monter un système de fichier	mount /dev/sda1 /media/stick
find	Recherche de fichiers dans les répertoires et sous répertoires Toutes les erreurs ne sont pas affichées Cherche iostream dans le répertoire courant Cherche tous les fichiers en majuscule ou minuscule qui commence par iostream Recherche les fichiers qui ont été modifiés il y a moins de 1 jour dans /home	cd / find -name iostream* 2>/dev/null sudo find -name iostream sudo find . -iname iostream* sudo find /home -mtime -1

grep	Recherche récursive dans /home de test dans tous les fichiers Recherche tous les termes cpu ou CPU dans le flux de sortie de la commande dmesg	<code>grep -rn /home -e "test"</code> <code>dmesg grep -i cpu</code>
sort	Classement par ordre croissant	<code>sort < animals.txt > sorted.txt</code>
dmesg	Affiche les informations venant du noyau	<code>dmesg head - 5</code> <code>dmesg tail - 5</code>
head	Affiche les 10 premières ligne du fichier test.txt Affiche les 50 premières ligne du fichier	<code>head test.txt</code> <code>sudo head -50 /var/log/syslog</code>
tail	Affiche les 10 dernières lignes en temps réel Affiche les 50 dernières lignes du fichier	<code>sudo tail -f /var/log/syslog</code> <code>sudo tail -50 /var/log/syslog</code>
tar	Tar et compresse en gzip le répertoire dir Tar et compresse en bzip2 le répertoire dir Untar un fichier tar compressé (bzip ou gzip)	<code>tar cvfz name.tar.gz dir</code> <code>tar cvfj name.tar.dir</code> <code>tar xvf name.tar.gz .</code>
ps	Process status : permet de connaitre les process actuellement actif	<code>ps -ef ou ps -aux</code>
htop	Outil semi-graphique de visualisation des processus (permet aussi de chercher et tuer des process)	
ifconfig	Affiche la configuration réseau	<code>sudo ifconfig wlan0 grep "inet"</code>

8.10.2 La notion de droits

Lorsque l'on utilise la commande ls -l, chaque fichier apparaît avec un certain nombre de paramètres correspondant à 3 types d'utilisateurs

- ✓ User : c'est le propriétaire du fichier, 3 bits permettent de définir les droits du propriétaire r(read), w(write) et x(exécutable)
- ✓ Group : le propriétaire du fichier fait parti d'un ou plusieurs groupes. Si un autre utilisateur appartient à l'un des groupes du propriétaire, il aura les droits associés en rwx du group
- ✓ Others : ce sont tous les autres utilisateurs non propriétaire et ne faisant pas partie du groupe. Par exemple un autre utilisateur aura les droits rwx associés à Others.

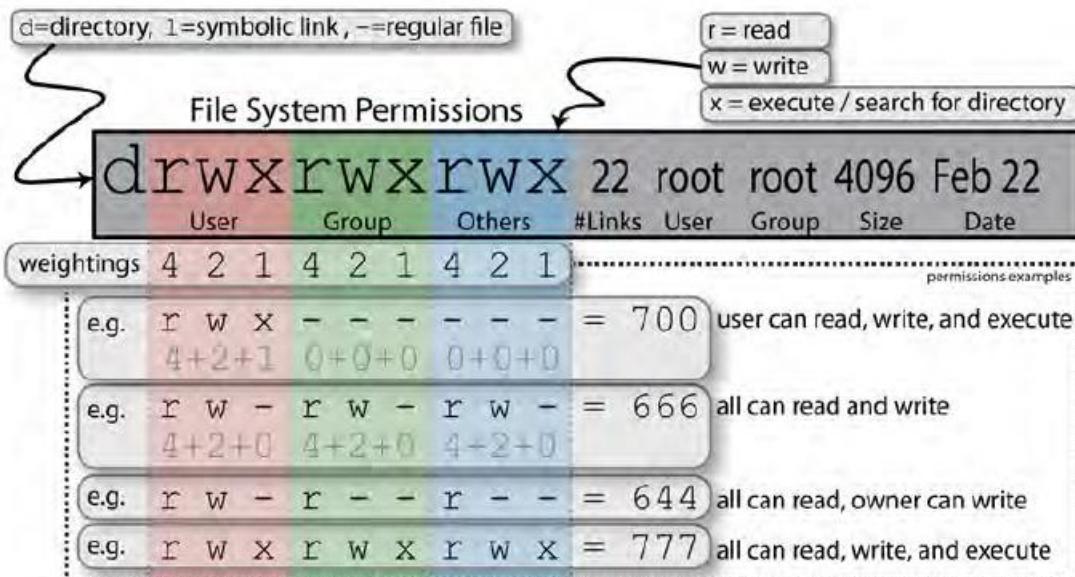


Figure 13 : les fichiers ou répertoires explication pour la commande ls -al

8.10.3 Utilisation du clavier

La couche Tabulation permet de compléter automatiquement le nom du fichier, le nom du répertoire ou même le nom de la commande exécutable. Par exemple, pour accéder au répertoire / tmp, vous pouvez taper cd /t, puis appuyer sur Tab, qui complétera automatiquement la commande à cd /tmp/.

Un certain nombre de combinaisons sont utiles lors que l'on travaille avec le bash :

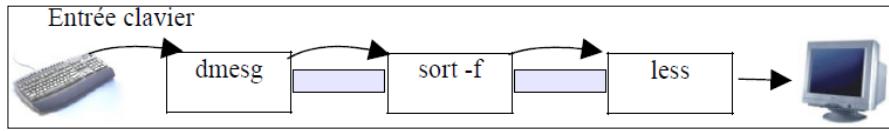
- ✓ Ctrl + A Vous ramène au début de la ligne que vous tapez
- ✓ Ctrl + E vous amène à la fin de la ligne que vous tapez
- ✓ Ctrl + U Efface au début de la ligne. Ctrl + E puis Ctrl + U efface la ligne
- ✓ Ctrl + L Efface l'écran
- ✓ Ctrl + C Tue le processus en cours d'exécution
- ✓ Ctrl + Z Place le processus en cours en arrière-plan. Tapez bg puis laissez-le en marche en arrière-plan, puis fg le ramènera au premier plan

8.10.4 Les redirection (> et >>) et les tunnels (ou pipes en anglais)

Les redirections sont souvent utilisées pour envoyer le flux standard vers un fichier.

- ⊕ Une commande linux par défaut sort les résultats sur la console. Il existe 2 sortes de sorties : les sorties normales (1) et erreurs (2). les 2 sorties s'affichent sur la console par défaut
- ⊕ Pour rediriger le résultat d'une commande (sorties 1 et 2) vers un fichier commande > fichier.
- ⊕ Pour rediriger les messages d'erreurs d'une commande vers un fichier commande 2> fichier.
- ⊕ Pour rediriger tous les messages (standard et erreur) vers un fichier commande &> fichier
- ⊕ Pour rediriger les messages vers « rien » (donc ne pas les afficher) on dispose du fichier /dev/null qui absorbe toute information sans la stocker. commande 2> /dev/null pour ne pas afficher les messages d'erreurs.

Les tunnels permettent de sérialiser plusieurs commandes. Chaque commande reçoit comme entrée la sortie de la commande précédente.



✓ Expliquer les commandes ci-dessous :

```

ls 1> ~/test
ls > ~/test
ls >>~/test
ls | sort -r | tail
ls | head -2 > ~/test
ls | tail -2 >> ~/test
cat ~/test | sort -r &> ~/test_r
cat /proc/cpuinfo /proc/meminfo > ~/infosys.txt
    
```

8.10.5 Les répertoires

Votre cible Linux est organisée en dossier à partir de la racine. Ces dossiers ont chacun une fonction qui est résumée ci-dessous :

Répertoires	explications
/bin et /usr/bin	Dossiers où se trouvent les différentes commandes utilisateur
/boot	le noyau et les fichiers de configuration et initrd (RAM File)
/dev	les fichiers spéciaux (associés aux drivers) permettant l'accès au matériel
/etc	les fichiers de configuration du système et certains scripts
/home	la base des répertoires utilisateurs
/lost+found	le stockage des fichiers (ou noeuds) retrouvés par fsck
/lib	les librairies système et les modules
/media	Dossier pour le montage de sitck USB ou SD
/mnt	Dossier pour le montage de systèmes de fichiers temporaires
/proc	un système de fichiers virtuels permettant l'accès aux variables du noyau (vide au démarrage de la machine)
/opt	Pour l'installation de logiciels « third party »
/root	le répertoire de base du super utilisateur
/sbin et /usr/sbin	les fichiers exécutables pour l'administration du système
/srv	Ou l'on retrouve les données associées au serveurs ftp web rsync,...
/sys	Ou se trouve le matériel associé au système de fichier sysfs (vide au démarrage)
/tmp	les fichiers temporaires
/usr	Répertoire le plus gros où se trouvent les programmes, les librairies et les fichiers accessibles pour l'utilisateur. Contient les application programs pour tous les utilisateurs et de nombreux répertoires importants pour votre système : /usr/include (C/C++ header files), /usr/lib (C/C++ library files), /usr/src (Linux kernel source), /usr/bin (user executables), /usr/local (similar to /usr but for local users), and /usr/share (shared files and media between users)
/var	les données variables liées à la machine (spool, traces)

Répondre aux questions suivantes :

- ✓ *Dans quel répertoire se trouve les sources du noyau ? Ce répertoire existe sur votre cible ?*
- ✓ *Dans quel répertoire se trouve les headers du noyau ? Ce répertoire existe sur votre cible ?*
- ✓ *Donner une commande se trouvant dans le répertoire /usr/bin*
- ✓ *Afficher le PATH de l'utilisateur geii. Cet utilisateur peut-il lancer les commandes se trouvant dans /usr/sbin ? expliquer*
- ✓ *Idem pour root (le super utilisateur)*

8.11 Manip 1 : gestion de la fréquence processeur

Pour cette manip, nous allons visualiser les fichiers se trouvant dans le répertoire `/sys/` et utiliser la commande `cpufreq-set`

- ✓ Copier le programme suivant dans le fichier `check_temp.sh`

```
# cat check_temp.sh
#!/bin/bash
sleept=5
while true; do
    cat /sys/devices/virtual/thermal/thermal_zone0/temp
    sleep $sleept
done
```

- ✓ Rendre le programme exécutable puis le lancer en tache de fond
- ✓ Tuer ce programme en utilisant htop (F3)
- ✓ Modifier ce programme pour que l'affichage de la température des processeurs se fasse toutes les 2s et relancer le programme en tache de fond
- ✓ Tuer ce programme en utilisant la commande kill
- ✓ Relancer le programme en tache de fond
- ✓ Quelle est la température du processeur ? dans quel fichier se trouve cette information ?

Nous allons maintenant visualiser la fréquence actuelle des 4 CPU

```
more /sys/devices/system/cpu/cpufreq/policy0/*
more /sys/devices/system/cpu/cpufreq/policy0/cpuinfo_cur_freq
```

Ouvrir une nouvelle console et lancer un programme de stress qui va pousser la charge des 4 CPU à 100%

```
stress -c 4
```

- ✓ Visualiser la température et la charge des CPU grâce à htop.

The screenshot shows three terminal windows side-by-side:

- Terminal 1 (Left):** Shows a list of processes and their IDs (e.g., 41987, 41866, 50457, 50699, 51304, 52030, 52756, 53361).
- Terminal 2 (Middle):** Shows the output of `htop`. It displays four CPU cores at 100% usage, system load average (3.97), uptime (06:23:17), and CPU frequencies (1.20 GHz). It also shows memory usage (91.3M/1000M) and swap status (OK/500M). A process table at the bottom shows a single task (PID 14519, user geii) using 100% CPU.
- Terminal 3 (Right):** Shows the command `stress -c 4` being run in the background, with the message "stress: info: [14515] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd".

On peut modifier la fréquence actuelle des cpu et la diminuer. Pour connaitre les fréquences disponibles il suffit d'afficher le fichier `scaling_available_frequencies`.

Changer la fréquence actuelle du processeur à 480000

```
sudo cpufreq-set -f 480000
```

- ✓ Expliquer ce qu'il se passe. La température est-elle plus haute ou plus basse ?
 - ✓ Aurait-on pu modifier directement le fichier `cpuinfo_cur_freq` sans utiliser la commande `cpufreq-set` ?
 - ✓ Tuer le process `stress` et visualiser la température des processeurs. Conclure

```
geii@bp11: ~
40293
40535
40898
40535
40414
40898
41140
41261
41261
41019
41503
41745
41503
42108
41140

000
every 5 seconds

geii@bp11: ~
1 [|||||100.0%] Tasks: 40, 8 thr; 4 running
2 [|||||100.0%] Load average: 3.84 1.86 0.73
3 [|||||100.0%] Uptime: 06:21:38
4 [|||||100.0%] CpuFreq1: 480 MHz
Mem[||||| 91.3M/1000M] CpuFreq2: 480 MHz
Swp[ 0K/500M] CpuFreq3: 480 MHz
Cpu Temp: 41 C CpuFreq4: 480 MHz

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
14517 geii 20 0 1848 64 0 R 99.4 0.0 1:34.60 stress -z

@bp11:/sys/class/thermal/thermal_zone0
bp11:/sys/class/thermal/thermal_zone0$ 
bp11:/sys/class/thermal/thermal_zone0$ 
▼ bp11:/sys/class/thermal/thermal_zone0$ 
geii@bp11:/sys/class/thermal/thermal_zone0$ 
geii@bp11:/sys/class/thermal/thermal_zone0$ 
geii@bp11:/sys/class/thermal/thermal_zone0$ 
geii@bp11:/sys/class/thermal/thermal_zone0$ stress -c 4
stress: info: [14515] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
```

- ✓ Quel est l'intérêt de diminuer la fréquence des processeurs ?

- ✓ Lancer la commande `cpufreq-info` et donner la fréquence minimale qui peut être allouée aux CPU

cpufreq-info

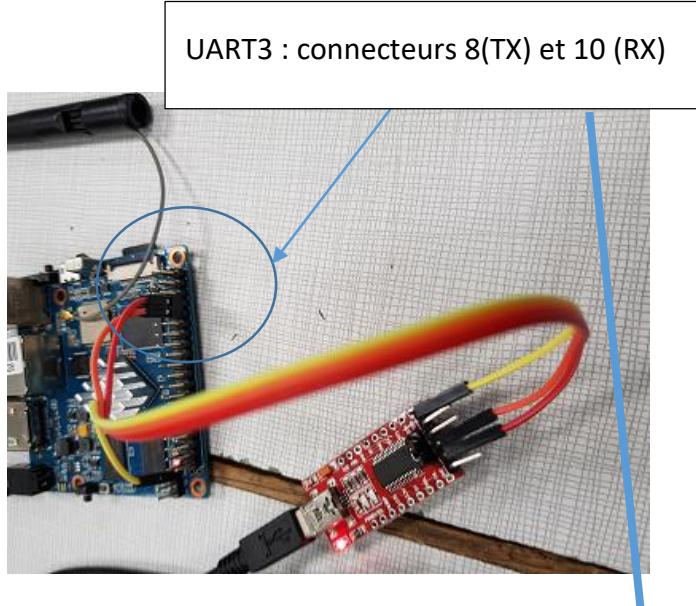
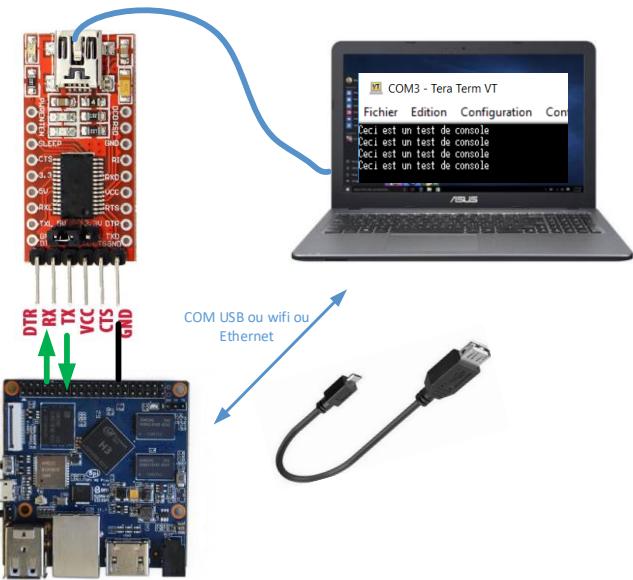
- ✓ D'après vous, comment agit cette commande, pour lire toutes les informations CPU ?

8.12 Manip 2 : utilisation de l'UART3

Cette manip va nous permettre de tester l'UART3 qui se trouve sur le connecteur 40 broches. A la différence avec l'UART de DEBUG celle-ci n'est pas utilisé comme console de DEBUG et donc est libre pour être utilisé pour communiquer avec un micro-controleur ou un autre circuit à UART.

On suppose que vous êtes déjà connecté sur la cible au travers du cable USB micro ou bien au travers une connexion SSH en Ethernet ou Wifi.

- ✓ Brancher votre FTDI sur le port UART3, comme montré sur le schéma. Nous allons maintenant utiliser le port UART3.



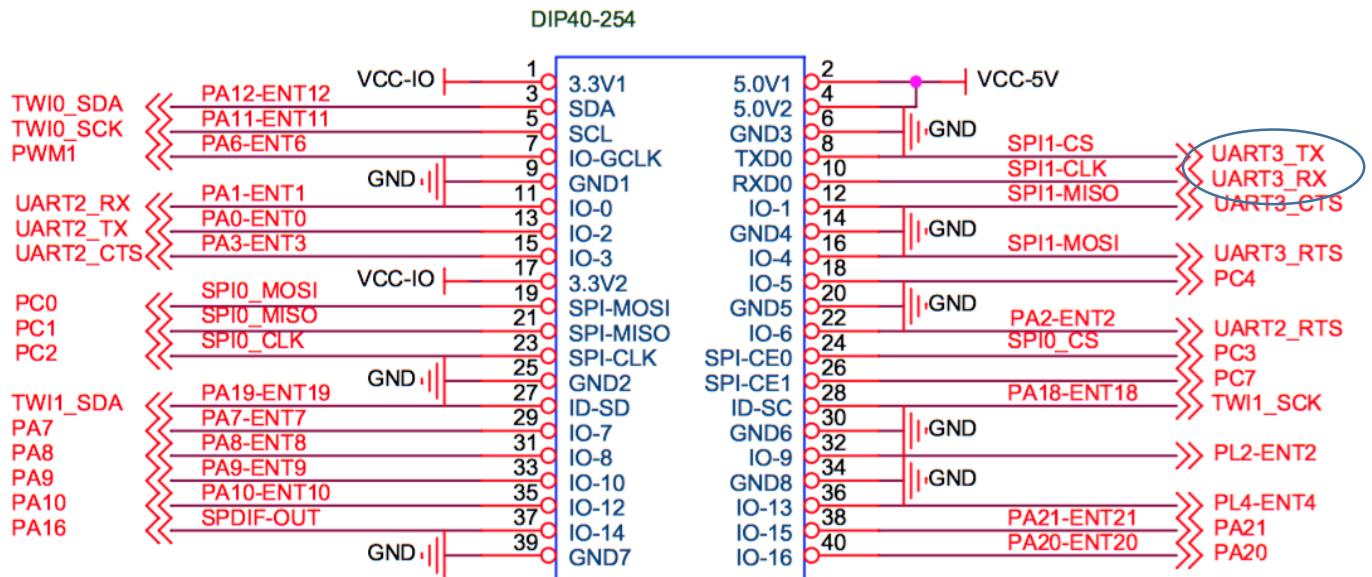
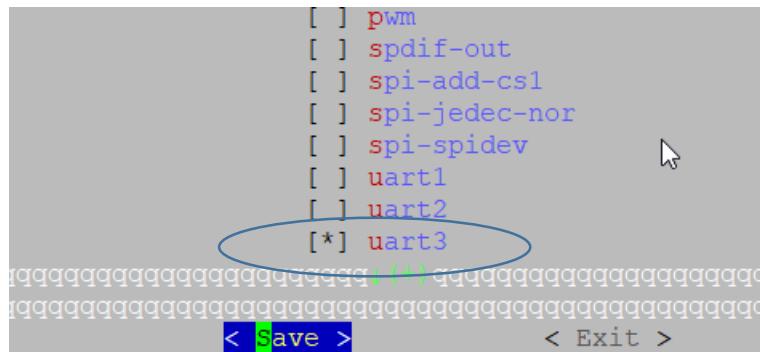


Figure 14 : connecteur GPIO du bananapi M2+

Nous allons tout d'abord activer l'UART3, par défaut tous les drivers de gestion du matériel (I2C, SPI, PWM, UART) sont désactivés, il faut donc le driver de gestion de l'UART3

- ✓ Lancer la commande armbian-config , System, Hardware et activer uart3



Puis reboot pour mettre à jour la prise en charge de l'UART3 par le noyau.

Pour vérifier que le ttyS3 a bien été activé :

```
dmesg | grep tty
```

ou

```
sudo grep tty /var/log/kern.log
```

Pour accéder au port il faut que le fichier associé au driver soit présent

- ✓ Vérifier que le fichier associé existe dans /dev

```
geii@bp11:~$ ll /dev/ttys3
crw-rw---- 1 root dialout 4, 67 Feb 6 17:03 /dev/ttys3
```

ttyS3 appartient au root et au groupe dialout. L'utilisateur s'il veut pouvoir utiliser l'UART3 sans passer par la commande sudo, doit avoir les droits du groupe dialout.

Ajouter le groupe dialout à votre utilisateur

```
sudo usermod -a -G dialout geii
```

Vérifier que le groupe dialout est bien ajouté

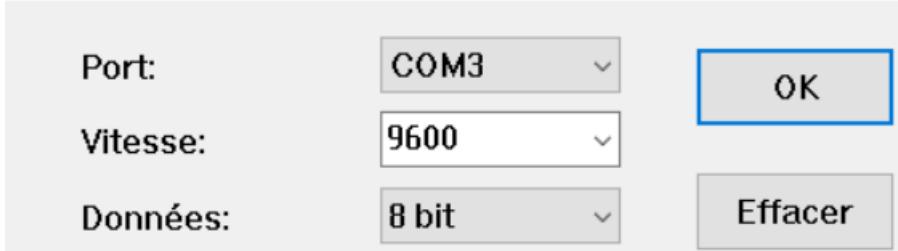
```
sudo groups geii
geii@bp11:~$ sudo groups geii
geii : geii dialout sudo audio video plugdev systemd-journal input netdev ssh
```

Votre UART3 est active et prête à l'emploi, nous allons pouvoir vérifier son bon fonctionnement.

Nous allons tester au travers de teraterm sur le PC, le bon fonctionnement du transfert de caractère entre le PC et notre cible.

- ✓ *Ouvrir térorterm sur votre PC. Vérifier que la vitesse de communication est de 9600 bps.
(Configuration, port serie)*

Tera Term: Config. port série

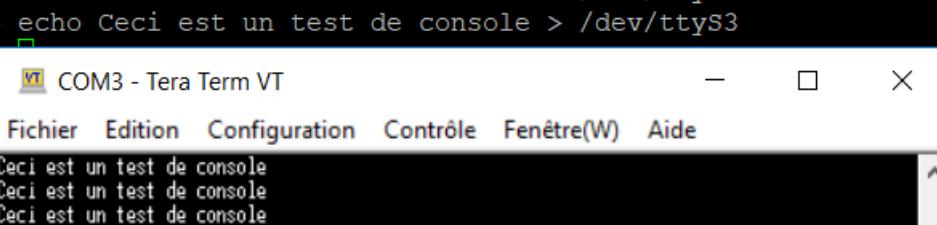


Votre teraterm est prêt à recevoir les données de votre cible.

Sur la cible utilisez la commande **echo test > /dev/ttys3** afin d'envoyer test vers le périphérique ttys3. Vous devriez voir apparaître les caractères envoyés sur teraterm du PC. Si ce n'est pas le cas, vérifier votre connexion entre Tx/Rx du PC et de la cible.

```
echo test > /dev/ttys3
```

```
geii@bp11:~$ echo Ceci est un test de console > /dev/ttys3
geii@bp11:~$ echo Ceci est un test de console > /dev/ttys3
geii@bp11:~$ echo Ceci est un test de console > /dev/ttys3
geii@bp11:~$ echo Ceci est un test de console > /dev/ttys3
geii@bp11:~$ echo Ceci est un test de console > /dev/ttys3
geii@bp11:~$ echo Ceci est un test de console > /dev/ttys3
geii@bp11:~$ echo Ceci est un test de console > /dev/ttys3
geii@bp11:~$
```



The screenshot shows a terminal window titled "COM3 - Tera Term VT". The window has a standard title bar with minimize, maximize, and close buttons. Below the title bar is a menu bar with options: Fichier, Edition, Configuration, Contrôle, Fenêtre(W), and Aide. The main area of the window displays four lines of text: "Ceci est un test de console", repeated four times. The window is set against a dark background.

Et dans l'autre sens ? Nous avons testé que l'envoi de caractère de la cible vers le PC fonctionnait. Vérifions que la communication fonctionne dans les 2 sens.

Nous allons utiliser screen qui est la même chose que teraterm mais pour linux. Screen est installé par défaut, si ce n'est pas le cas utiliser la commande sudo apt install screen pour installer screen.

Lancer la console sur le port `ttyS3` (vitesse par défaut 9600).

screen /dev/ttyS3

Tous les caractères tapés sur la cible apparaissent dans teraterm sur le PC et tous les caractères tapés sur le PC apparaissent dans screen. Le test d'envoie et de réception de caractères est validé.



eeeeeeee

eeeeeeee

eeeeeeee

eeeeeeee

eeeeeeee

aaaaaaa

La comm fonctionne !

8.13 Manip 3 : le bootload u-boot

Lire la partie Boot system de <https://linuxjourney.com/>.

Dans cette manip nous allons entrer sur le bootloader après le reboot afin de comprendre ce qu'est un bootloader et quelle est sa fonction. Nous finirons par un schéma expliquant les principes généraux du démarrage d'un système linux.

Le bootloader pour les processeurs ARM est u-boot, pour les processeurs Intel c'est grub. Nous allons donc nous intéresser à u-boot et à quelques unes de ses caractéristiques.

Le noyau est lancé par le bootloader u-boot. Mais quelle est la fonction d'un bootloader ?

Le bootloader permet

- ✓ D'initialiser le hardware et notamment la mémoire
- ✓ De paramétrier le lancement du noyau
- ✓ De lancer le noyau.

Vous allez connecter le port série de DEBUG (enlever la connexion sur UART3 de la manip 2 et placer votre connexion sur le connecteur 3 points série du debug console. Ouvrir teraterm ou putty puis rebooter votre cible.

Au moment du démarrage, le bootloader vérifie que l'utilisateur n'a pas tapé sur la barre d'espace et lance le noyau. Si l'utilisateur a tapé sur la barre d'espace avant le lancement du noyau, uboot arrête le lancement automatique et l'utilisateur peut alors interagir avec le bootloader. C'est ce que nous allons faire dans cette manip.



```
VT COM3 - Tera Term VT
Fichier Edition Configuration Contrôle Fenêtre(W)
root@bp11:/home/geii# ls
create_ap public
root@bp11:/home/geii#
```

Au moment du redémarrage tapez plusieurs fois sur la barre espace afin d'arrêter le boot par défaut et entrer dans la console de commande du bootloader uboot.

- ✓ *Taper sur la barre espace (une ou plusieurs fois) sur teraterm au lancement de la cible afin de rentrer dans l'interface de commande du bootloader.*

Vous êtes sur l'interface de commande du bootloader.

- ✓ *Lancer la commande bdinfo pour connaître les adresses mémoires. On peut voir que la mémoire RAM est à l'adresse 0x40000000*

```
=> bd
arch_number = 0x00001029
boot_params = 0x40000100
DRAM bank = 0x00000000
-> start = 0x40000000
-> size = 0x40000000
baudrate = 115200 bps
TLB addr = 0x7FFF0000
relocaddr = 0x70F7E000
reloc off = 0x33F7E000
irq_sp = 0x79F59C20
sp start = 0x79F59C10
Early malloc usage: 190 / 400
fdt_blob = 79f59c38
=> load mmc 0:1 0x40000000 /boot/zImage
```

U-boot est capable de monter le système de fichier sur la mmc 0 partition 1 (c'est là où se trouve notre Linux sur la SD card)

Uboot comprend la commande ls (il est capable de lire le système de fichier).

- ✓ *Lancer la commande ls mmc 0:1 /boot. Cette commande bas niveau demande à uboot de lire sur la mmc 0 partition 1 le répertoire /boot.*

```
=> ls mmc 0:1 /boot
<DIR> 4096 .
<DIR> 4096 ..
<SYM> 17 dtb
<SYM> 21 uInitrd
<SYM> 21 zImage
160835 config-4.19.17-sunxi
<DIR> 4096 overlay-user
230454 boot.bmp
0 .next
1536 armbian_first_run.txt.template
8145003 initrd.img-4.19.17-sunxi
201 armbianEnv.txt
8145067 uInitrd-4.19.17-sunxi
4882 boot-desktop.png
3726 boot.cmd
3798 boot.scr
<SYM> 17 dtb.old
7313680 vmlinuz-4.19.17-sunxi
3232207 System.map-4.19.17-sunxi
<DIR> 12288 dtb-4.19.17-sunxi
=> load
load - load binary file from a filesystem

Usage:
load <interface> [<dev[:part]> [<addr> [<filename> [bytes [pos]]]]]
- Load binary file 'filename' from partition 'part' on device
  type 'interface' instance 'dev' to address 'addr' in memory.
  'bytes' gives the size to load in bytes.
  If 'bytes' is 0 or omitted, the file is read until the end.
  'pos' gives the file byte position to start reading from.
  If 'pos' is 0 or omitted, the file is read from the start.
=> load mmc 0:1 0 /boot/zImage
7313680 bytes read in 489 ms (14.3 MiB/s)
```

Nous allons maintenant charger le noyau en zone RAM .à l'adresse 0x42000000.

```
load mmc 0:1 42000000 /boot/zImage
```

```
=> load mmc 0:1 42000000 /boot/zImage
7313680 bytes read in 362 ms (19.3 MiB/s)
=> iminfo 42000000
```

Nous venons avec cette commande de lire le noyau zImage qui se trouve sur la SD Card dans le répertoire /boot et de le recopier en RAM à l'adresse 0x420000000. Lors du lancement du noyau au démarrage de notre système, c'est la première opération qui est faite par le bootloader.

La deuxième opération faite par le bootloader lors du lancement est le passage de paramètre aux noyau chargé en RAM et son lancement. C'est ce que nous allons faire.

Puis lancer le noyau qui est à l'adresse 0x420000000 avec son fichier RamDisk et son FDT tree

```
bootz 42000000 43300000 43000000
```

```
=> bootz 42000000 43300000 43000000
## Loading init Ramdisk from Legacy Image at 43300000 ...
  Image Name: uInitrd
  Image Type: ARM Linux RAMDisk Image (gzip compressed)
  Data Size: 8145003 Bytes = 7.8 MiB
  Load Address: 00000000
  Entry Point: 00000000
  Verifying Checksum ... OK
## Flattened Device Tree blob at 43000000
  Booting using the fdt blob at 0x43000000
  Loading Ramdisk to 4983b000, end 49fff86b ... OK
    reserving fdt memory region: addr=43000000 size=6d000
  Loading Device Tree to 497cb000, end 4983afff ... OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
[    0.00000] Booting Linux on physical CPU 0x0
[    0.00000] Linux version 4.19.17-sunxi (root@armbian.com) (gcc version 7.2.1 20171
[    0.00000] CPU: ARMv7 Processor [410fc075] revision 5 (ARMv7), cr=50c5387d
[    0.00000] CPU: div instructions available; patching division code
[    0.00000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cach
[    0.00000] OF: fdt: Machine model: Banana Pi BPI-M2-Plus
[    0.00000] Memory policy: Data cache writealloc
[    0.00000] cma: Reserved 128 MiB at 0x75c00000
[    0.00000] psci: probing for conduit Method from DT.
[    0.00000]
```

Comme on peut le voir, le noyau est lancé, il se dézippe puis se lance. Vient alors le problème de son paramétrage qui est dans /boot/boot.scr. Nous n'irons pas plus loin. L'exercice était de voir le rôle du bootloader et de voir son interface de commande. Nous n'aurons pas besoin de rentrer dans ces détails dans le futur. Par contre la modification du fichier boot.scr ou armbianEnv.txt permet de paramétriser le noyau (chargement des drivers i2C,...) mais aussi de changer le debug sur ttyS1 et garder ttyS0 libre. Le bootloader a été installé au tout début de la SD... par la commande :

```
dd if=u-boot-sunxi-with-spl.bin of=/dev/mmc bs=1024 seek=8
```

8.14 Pour aller plus loin ...

Voyons ensemble comment démarre notre cible.

1. Au démarrage le processeur charge son Program Counter à la valeur 0xFFFF_0000 (adresse de la ROM interne) et exécute les instructions du boot loader (1) se trouvant dans le processeur.
2. Ce bootloader va tester si la SD card est bootable et si c'est le cas , il charge le bootloader(2) u-boot en RAM puis l'exécute (une SD est bootable si elle possède une séquence « magique » à un endroit déterminé)
3. Lors de l'exécution de u-boot celui-ci va chercher son fichier de configuration dans /boot/boot.scr sur la SD pour récupérer les variables d'environnement (zone RAM de chargement du noyau, les paramètres à donner aux noyau, ...)
4. Ensuite uboot charge le noyau en RAM puis le lance en lui appliquant les paramètres de boot lue dans boot.scr (3)
5. Le noyau se dézippe puis se lance (initialisation du matériel et lancement des drivers)

6. A la fin le noyau monte le système de fichier initrd (ramdisk) dans lequel se trouve un système de fichier minimal permettant d'insérer les drivers qui n'ont pas été compilé dans le noyau afin de pouvoir monter le système de fichier se trouvant sur la SD et de lancer le premier programme /sbin/init qui lance la distribution debian et les nombreux services (Systemd, ntpd,...).

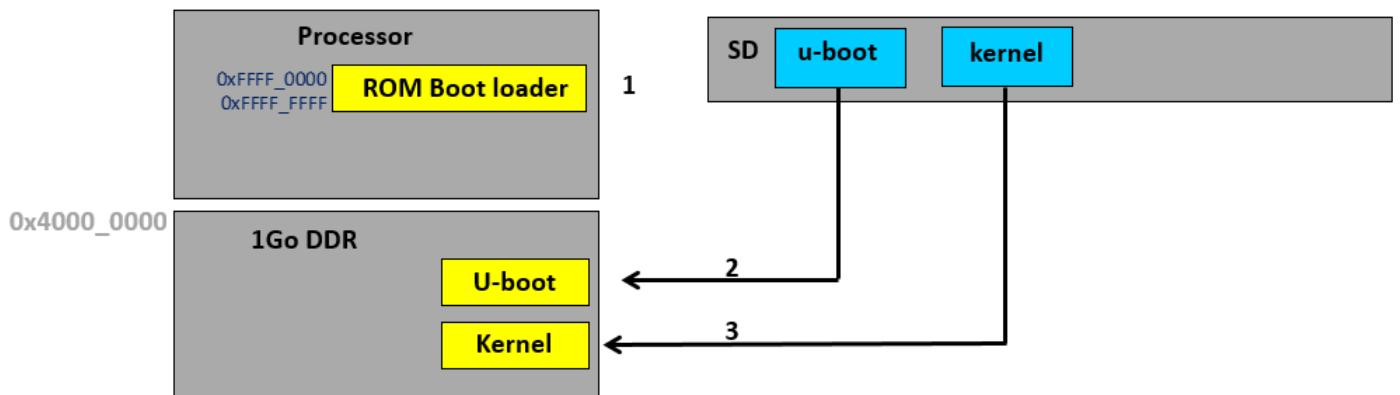


Figure 15 : processus de démarrage de la cible bananapi M2+

Regardons ensemble comment est organisé la SD Card ou la mmc interne. Tout d'abord, la table de partitions dans laquelle se trouve les informations sur la ou les partitions de la flash, leur taille, leur type... (commande fdisk ou gparted) se trouve au début de la flash. Puis vient à l'adresse 8ko le bootloader, c'est à cette adresse que le bootloader interne se trouvant dans le SOC allwinner ira pointer à condition qu'il ait détecté que la SD est bootable. Le ROM Bootloader charge alors U-boot en RAM et le lance. U-boot prend alors le relais et vient charger le fichier /boot/boot.scr dans lequel se trouve les paramétrages du noyau. Il vient ensuite charger le noyau zImage en RAM puis lance le noyau qui prend le relais. Une fois le noyau lancé celui-ci lance la commande init qui est le premier process qui charge tous les autres. En utilisant la commande ps -ef on pourra voir que le process qui a le numéro 1 est le process init.

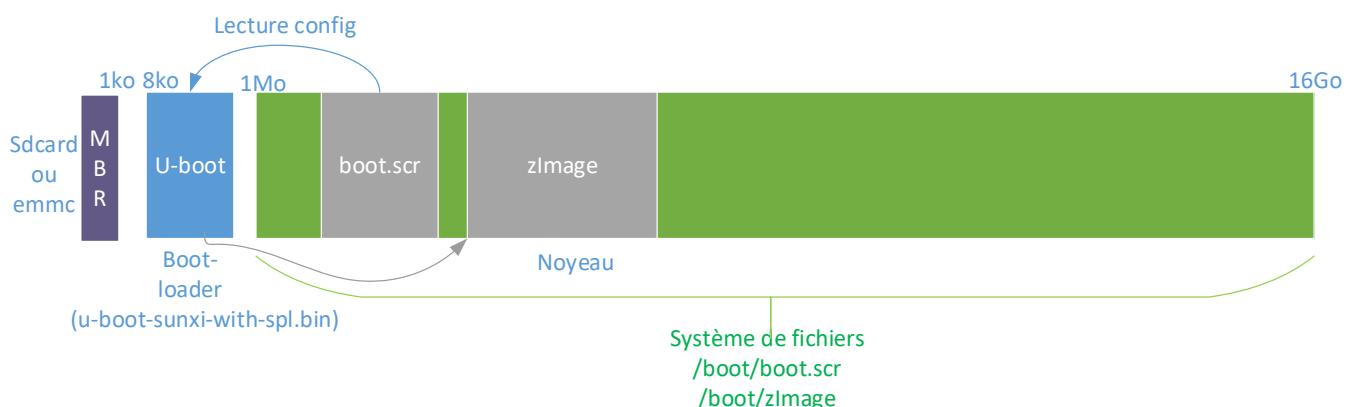


Figure 16 : les programmes utilisés lors du boot : visualisation de la FLASH

Zone mémoire du processeur H3 (4 Go) : Le SOC (System On Chip) H3 est constitué de 4 coeurs A7 cadencé au maximum à 1,3GHz, de RAM L1, RAM L2, d'une mémoire ROM (à l'adresse 0xFFFF0000), d'une RAM utilisateur, d'un GPU Mali et d'un grand nombre de registres. Associé à ce SOC le bananapi M2+ possède une RAM DDR3 1Go et une mémoire Flash EMMC 8Go.

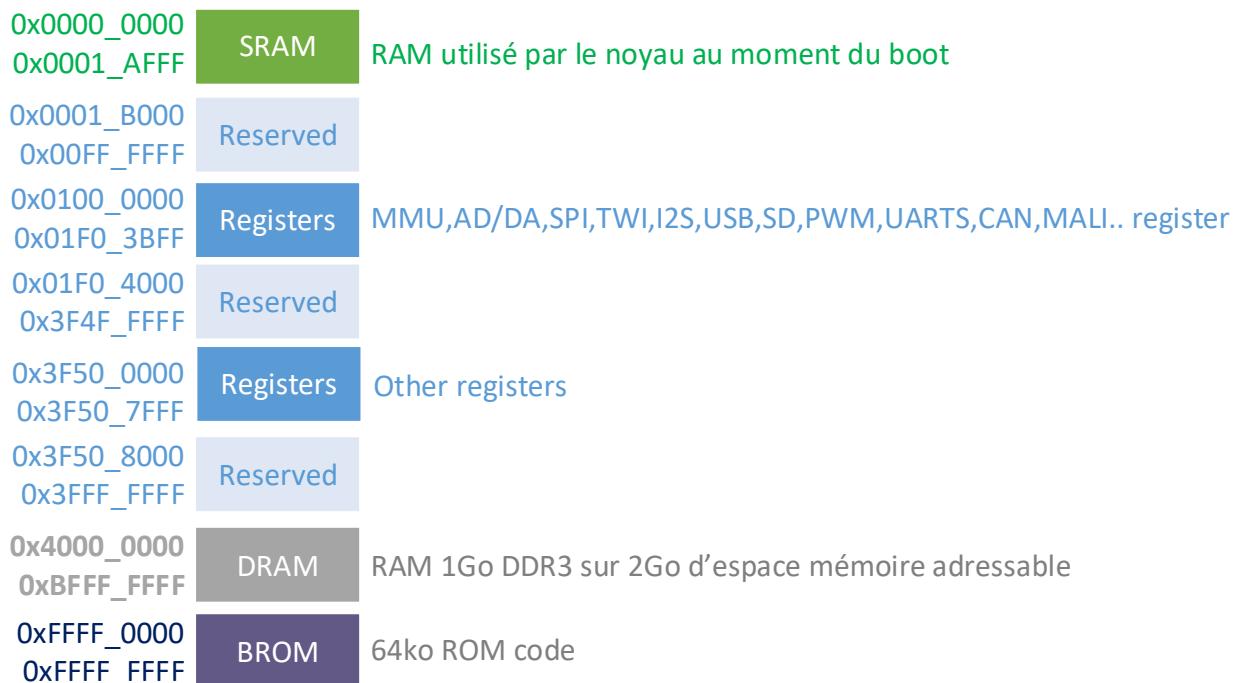


Figure 17 : adressage mémoire du M2+

Le plan mémoire du bananapi est résumé sur la figure 17. La RAM se trouve à l'adresse 0x4000000.

https://linux-sunxi.org/images/4/4b/Allwinner_H3_Datasheet_V1.2.pdf p84

Pour plus d'information :

http://linux-sunxi.org/Bootable_SD_card

<https://linux-sunxi.org/BROM>

http://linux-sunxi.org/Memory_map

8.15 Kernel

Questions	Réponses
Quelle est la différence entre le kernel et les commandes utilisateurs dans /bin /usr et /sbin ?	
Où se trouvent les dlls ou bibliothèques de library ? On pourra lancer strace ls (installer le package strace si nécessaire)	
Que se passe-t-il si l'on déplace une librairie utilisée par ls ? Faites la manip.	
Comment connaître la version actuelle du noyau sur la cible ?	
Aller sur https://www.kernel.org/ , quelle est la version stable actuellement.	
Il existe différents Kernel, celui pour PC 64 bits (amd64) et d'autres pour arm. Qui gère la partie du kernel pour le BPI M2+ et les oranges pi ? http://linux-sunxi.org/Mainlining_Effort	
Peut-on avoir 2 noyau sur la cible ? Comment lancer l'un ou l'autre ?	
On pourra voir les sources du noyau actuel pour les processeurs Allwinner https://github.com/linux-sunxi/linux-sunxi	
Est-il possible de compiler les sources du noyau et l'installer sur notre cible ?	
Peut-on connaître avec quelles options le noyau sur la cible a été compilé ?	
Quels sont les drivers chargés actuellement ? Donnez en 2. Le module bluetooth est-il chargé ? On pourra utiliser la commande grep pour filtrer	
Insérer le module bluetooth	
Vérifier que le module a bien été lancé. Et qu'il est bien présent (lsmod) tail /var/log/kern.log	

8.16 Init

Questions	Réponses
Lancer la commande <code>systemd-analyze blame</code> pour connaitre le temps de lancement des différents services. Quel est le service qui prend le plus de temps ?	
Lancer la commande <code>service --status-all</code> . Certains services sont inactifs (-) et d'autres actifs (+). Vérifier avec <code>htop</code> que les services en - ne sont pas en cours d'exécution	
<i>Lancer la commande <code>systemctl status systemd-timesyncd</code> c'est la synchronisation avec serveur ntp</i>	
Nous avons aussi un client ntp <code>sudo systemctl status ntp</code> qui est arrêté	
Lancer la commande date	
Vous pouvez modifier la zone tout d'abord, vérifier quelle zone nous appartenons <code>timedatectl list-timezones grep Paris</code>	
Puis changer la zone de votre cible : <code>sudo timedatectl set-timezone Europe/Paris</code>	
Relancer la commande date.	
Enlever le paquet ntp qui n'a aucun role dans la distribution puisque remplacé par timesyncd <code>sudo apt purge ntp</code>	

8.17 Manip 4 : Process Utilisation

Lire plus particulièrement la partie <https://linuxjourney.com/lesson/systemd-overview>

Afin de comprendre la notion de process, nous allons créer un programme test.sh qui est le suivant :

/home/geii/test.sh

```
#!/bin/bash
while true
do
echo bonjour
sleep 5
done
```

Rendre le programme exécutable et le tester

```
chmod +x test.sh
./test.sh
```

Sur notre distribution, c'est Systemd qui gère le lancement des différents process.

- ✓ *Créer le fichier systemd-test.service dans le répertoire /etc/systemd/system/*

/etc/systemd/system/systemd-test.service

```
[Unit]
Description=test Client Daemon
After=network-online.target
[Service]
Type=simple
User=geii
Group=geii
UMask=007
ExecStart=/home/geii/test.sh
Restart=on-failure
[Install]
WantedBy=multi-user.target
```

Relancer systemctl

```
sudo systemctl deamon-reload
```

puis vérifier l'état du service

```
sudo systemctl status systemd-test
```

puis lancer le service

```
sudo systemctl start systemd-test
```

puis vérifier l'état du service

Lancer la commande lsof | grep test.sh

```
lsof | grep test.sh
```

- ✓ *Quels sont les fichiers utilisés par votre script ?*

```
sudo systemctl status systemd-test
```

utiliser la commande journalctl afin d'avoir accès aux information issu de votre service

```
sudo journalctl -f -u systemd-test
```

Activer le service pour qu'il démarra au lancement

```
sudo systemctl enable systemd-test
```

- ✓ *Rebooter et vérifier*
- ✓ *Arrêter le service*
- ✓ *Le désactiver pour le prochain démarrage*

Questions	Réponses
Donner 3 types de système de lancement des différents process lors de l'init de Linux.	
Quel est le principal système de lancement de fichier utilisé par votre distribution ?	
Debian utilisait avant SystemV, ce système de lancement existe-t-il encore ?	
Où se trouvent les fichiers de configuration de SystemV ?	
Où se trouvent les fichiers de configuration de systemD ?	
Lancer la commande iostat. Expliquer	

Activer dans /etc/default/sysstat la gestion automatique de l'écriture dans /var/log des données processeurs, mémoires,... chaque jour.	
Lancer la commande sudo dpkg -L sysstat afin de connaitre tous les fichiers installé par ce paquet.	
Il y a des fichiers cron expliquer. On pourra aller dans le répertoire /etc/cron.daily/	
Ouvrir le fichier, lancer la commande /usr/lib/sysstat/sa2 -A (qui est dans le fichier).	
Aller dans /var/lib/sysstat et vérifier qu'un fichier a bien été créé	
Lancer la commande	
sudo sar -q	

8.18 Logging

Questions	Réponses
Ouvrez le fichier se trouvant dans /etc/rsyslog et dire dans quels fichiers vont les messages issus du noyau	
Que fait dmesg , ya til une différence avec kern.log	
Lancer la commande journalctl.	
Que trouve t on dans le fichier syslog ?	
Envoyer un info au fichier log et vérifier (commande logger)	
En utilisant la commande df -h, dire ou se trouve /var/log.	
Que se passe-t-il a chaque reboot ? Expliquer.	

Quelques remarques :

Pour connaitre les fichiers installés par le paquet hostapd :

```
dpkg -L hostapd
```

```
geii@bananapim2plus:~$ dpkg -L hostapd
/.
/etc
/etc/default
/etc/default/hostapd
/etc/hostapd
/etc/hostapd/ifupdown.sh
/etc/hostapd.conf
/etc/init.d
/etc/init.d/hostapd
/etc/network
/etc/network/if-post-down.d
/etc/network/if-pre-up.d
/usr
/usr/sbin
/usr/sbin/hostapd
/usr/sbin/hostapd_cli
```

Les fichiers de configuration sont dans /etc, on peut donc en déduire que le fichier hostapd.conf est le fichier de configuration du hostapd

Pour savoir si le service hostapd est lancé :

On pourra utiliser

```
htop
```

CPU Usage										Memory Usage		Swap Usage		Filesystem Usage		Network Usage		Disk Usage		Process Status				
PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command	Mem	Swap	File	Mount	Net	Link	Disk	IO	Process	Threads	Running		
1	[20	0	31488	13200	8136	S	0.0	1.3	0:00.00	/usr/bin/python3 /usr/share/unattended-upgrades/unattended-upg	0.0%	0.0%	Tasks: 44, 10 thr; 1 running	Load average: 0.02 0.02 0.03	Uptime: 06:26:48	CpuFreq1: 1.20 GHz	CpuFreq2: 1.20 GHz	CpuFreq3: 1.20 GHz	CpuFreq4: 1.20 GHz				
2	[20	0	31488	13200	8136	S	0.0	1.3	0:00.65	/usr/bin/python3 /usr/share/unattended-upgrades/unattended-upg	0.0%	0.0%											
3	[20	0	8396	3928	3432	S	0.0	0.4	0:00.35	/usr/sbin/sshd -D	1.3%	0.0%											
4	[20	0	38188	5632	4748	S	0.0	0.5	0:00.00	/usr/lib/policykit-1/polkitd --no-debug	109M/1000M	0K/500M											
Mem	[20	0	38188	5632	4748	S	0.0	0.5	0:00.15	/usr/lib/policykit-1/polkitd --no-debug	CpuFreq1: 1.20 GHz	CpuFreq2: 1.20 GHz	CpuFreq3: 1.20 GHz	CpuFreq4: 1.20 GHz									
Swp	[20	0	38188	5632	4748	S	0.0	0.5	0:00.30	/usr/lib/policykit-1/polkitd --no-debug													
Cpu Temp:	37 C	20	0	3512	1440	1332	S	0.0	0.1	0:00.01	/sbin/agetty -o -p -- \u00a9 --keep-baud 115200,38400,9600 ttysG0													
1255	root	20	0	3732	1268	1164	S	0.0	0.1	0:00.01	/sbin/agetty -o -p -- \u00a9 --keep-baud 115200,38400,9600 ttysG0													
1257	root	20	0	3512	1616	1508	S	0.0	0.2	0:00.01	/sbin/agetty -o -p -- \u00a9 --noclear ttys1 linux													
1258	root	20	0	9956	4700	3972	S	0.0	0.5	0:00.45	sshd: geii [priv]													
2370	root	20	0	9956	4684	3956	S	0.0	0.5	0:00.45	sshd: geii [priv]													
2372	geii	20	0	10096	5624	4460	S	0.0	0.5	0:00.34	/lib/systemd/systemd --user													
2373	geii	20	0	31024	2048	96	S	0.0	0.2	0:00.00	(sd-pam)													
2466	geii	20	0	10184	4160	3420	S	0.0	0.4	0:00.02	sshd: geii@notty													
2467	geii	20	0	1884	1324	1192	S	0.0	0.1	0:00.02	/usr/lib/openssh/sftp-server													
2469	root	20	0	9956	4700	3972	S	0.0	0.5	0:00.31	sshd: geii [priv]													
2554	geii	20	0	10184	4084	3344	S	0.0	0.4	0:00.00	sshd: geii@notty													
2555	geii	20	0	4496	2444	2196	S	0.0	0.2	0:00.02	-bash													
2561	root	20	0	9956	4696	3964	S	0.0	0.5	0:00.38	sshd: geii [priv]													
2646	geii	20	0	10184	4004	3264	S	0.0	0.4	0:00.49	sshd: geii@pts/0													
2647	geii	20	0	5212	3356	2420	S	0.0	0.3	0:01.03	-bash													
4991	dnsmasq	20	0	7160	1732	1420	S	0.0	0.2	0:00.23	/usr/sbin/dnsmasq -x /run/dnsmasq/dnsmasq.pid -u dnsmasq -r /													
5047	root	20	0	5424	1692	1360	S	0.0	0.2	0:00.05	/usr/sbin/hostapd -B -P /run/hostapd.pid /etc/hostapd.conf													

Ou avec la commande ps qui permet de lister tous les process en cours.

```
ps -ef
```

Autre solution :

`ps -ef | grep host`

```
geii@bananapim2plus:~$ ps -ef | grep host
root      5047      1  0 16:23 ?        00:00:00 /usr/sbin/hostapd -B -P /run/hostapd.pid /etc/hostapd.conf
geii      5976  5778  0 16:56 pts/1    00:00:00 grep --color=auto host
```

On peut aussi utiliser la commande service qui donne tous les services ou daemons en cours d'exécution

`sudo service --status-all`

8.19 Manip 5 : Utilisation d'une webcam

Le but de la manip ici est d'insérer une webcam sur le port USB et de rendre cette webcam IP, c'est-à-dire diffusé le flux video via ethernet (ou le wifi).

Cette manip sera divisée en 2 partie :

- ✓ La première va nous permettre de prendre en main la webcam
- ✓ Puis nous lancerons la prise de photos toutes les minutes à l'aide d'un service timer

Nous finirons par l'utilisation d'un outil de transfert de flux video sur IP.

- ✓ *Branchez votre webcam sur le port usb de votre cible*

lsusb pour vérif que la webcam est ok

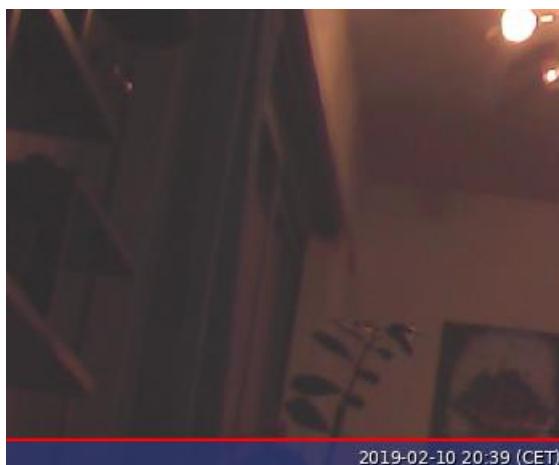
puis installer l'outil pour prendre des photos

`sudo apt-get install fswebcam`

puis prendre une photo à partir de la webcma (dans le fichier test.jpg).

`fswebcam test.jpg`

- ✓ *récupérer la photo via winscp et ouvrez la pour vérifier.*



Nous allons créer un script take_picture.sh qui crée un fichier jpg avec la date de la photo;

- ✓ *Puis le fichier take_picture.sh*

```
/home/geii/webcam/take_picture.sh
```

```
#!/bin/bash
DATE=$(date +"%Y-%m-%d_%H%M")
if [ ! -d /tmp/webcam ] ; then
    mkdir /tmp/webcam;
fi;
fswebcam --no-banner /tmp/webcam/$DATE.jpg
```

- ✓ *rendre ce fichier exécutable*
- ✓ *puis le tester*

```
geii@bp12:~/webcam$ ./take_picture.sh
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
Adjusting resolution from 384x288 to 352x288.
--- Capturing frame...
Captured frame in 0.00 seconds.
--- Processing captured image...
Disabling banner.
Writing JPEG image to '/tmp/webcam/2019-02-10_2214.jpg'.
```

Une fois le test du fichier de prise de photo validé, nous allons créer un service qui sera appelé toute les minutes et qui appellera notre script qui prendra la photo et placera dans le nom du fichier la date et l'heure de la prise de vue.

Nous allons créer le fichier /etc/systemd/system/pict_every_mn.timer et le fichier /etc/systemd/system/pict_every_mn.service. Ces 2 fichiers sont associés par leur nom. Le premier ayant l'extension timer va être déclenché selon un temps que nous choisirons, ce timer lancera alors le service du même nom.

Le premier (on créer un évènement toutes les minutes)

```
/etc/systemd/system/pict_every_mn.timer
```

```
[Unit]
Description=My timer every minute
[Timer]
OnUnitActiveSec=1min
Persistent=true
[Install]
WantedBy=timers.target
```

```
/etc/systemd/system/pict_every_mn.service
```

```
[Unit]
Description=take picture each minute
[Service]
Type=simple
```

```
ExecStart=/home/geii/webcam/take_picture.sh
user=geii
```

Une fois les 2 fichiers créé (il faudra utiliser sudo), vérifier que votre timer fait parti des timers pris en charge par systemctl.

```
sudo systemctl list-timers --all
```

```
geii@bp12:~/webcam$ sudo systemctl list-timers --all
NEXT          LEFT            LAST          PASSED        UNIT           ACTIVATES
Sun 2019-02-10 21:48:00 CET 23s left      Sun 2019-02-10 21:47:19 CET 17s ago pict.every_mn.timer
Mon 2019-02-11 00:00:00 CET 2h 12min left Mon 2019-02-04 07:34:21 CET 6 days ago fstrim.timer
Mon 2019-02-11 06:53:24 CET 9h left       Sun 2019-02-10 07:09:13 CET 14h ago apt-daily-upgrade.timer
Mon 2019-02-11 08:51:43 CET 11h left      Sun 2019-02-10 20:46:25 CET 1h 1min ago motd-news.timer
Mon 2019-02-11 09:36:08 CET 11h left      Sun 2019-02-10 21:18:34 CET 29min ago apt-daily.timer
Mon 2019-02-11 20:50:25 CET 23h left      Sun 2019-02-10 20:50:25 CET 57min ago systemd-tmpfiles-clean.timer
n/a             n/a            n/a           n/a           every_mn.timer
```

On pourra ensuite voir toutes les minutes les fichiers dans /tmp/webcam

```
geii@bp12:~/webcam$ ls /tmp/webcam
2019-02-10_2210.jpg 2019-02-10_2211.jpg 2019-02-10_2212.jpg 2019-02-10_2213.jpg 2019-02-10_2214.jpg
```

Arrêter et désactiver le service

```
geii@bp12:~/webcam$ sudo systemctl disable pict_every_mn.timer
Removed /etc/systemd/system/timers.target.wants/pict_every_mn.timer.
geii@bp12:~/webcam$ sudo systemctl stop pict_every_mn.timer
```

Et la vidéo ? Nous allons maintenant utiliser un autre programme qui gère la vidéo : motion.

Installer le paquet motion

```
sudo apt install motion
```

Nous allons configurer motion pour que celui-ci se lance comme service

```
sudo nano /etc/default/motion
```

Modifier l'option pour que le service se lance automatiquement

```
start_motion_daemon=no --> start_motion_daemon=yes
```

Press Ctrl+O to save and type Ctrl+X to exit.

Nous allons aussi modifier la configuration de motion pour qu'il puisse être utilisé comme serveur de streaming. Par défaut il n'écoute que sur le port localhost. Nous allons changer cela pour qu'il écoute toutes les adresses.

```
sudo nano /etc/motion/motion.conf
```

```
daemon on
stream_localhost off
webcontrol_localhost off
```

```
#####
#####
```

```
# Daemon
#####
# Start in daemon (background) mode and release terminal (default: off)
daemon on
#####
# Live Stream Server
#####
# Restrict stream connections to localhost only (default: on)
stream_localhost off
#####
# HTTP Based Control
#####
# Restrict control connections to localhost only (default: on)
webcontrol_localhost off
#####
```

Press Ctrl+O to save and type Ctrl+X to exit.

Puis lancer le service motion

```
sudo service motion start
```

```
sudo motion
```

source : <http://awesomeprojectsxyz.blogspot.com/2015/09/beginners-guide-how-to-setup-usb-webcam.html>

puis tester sur internet

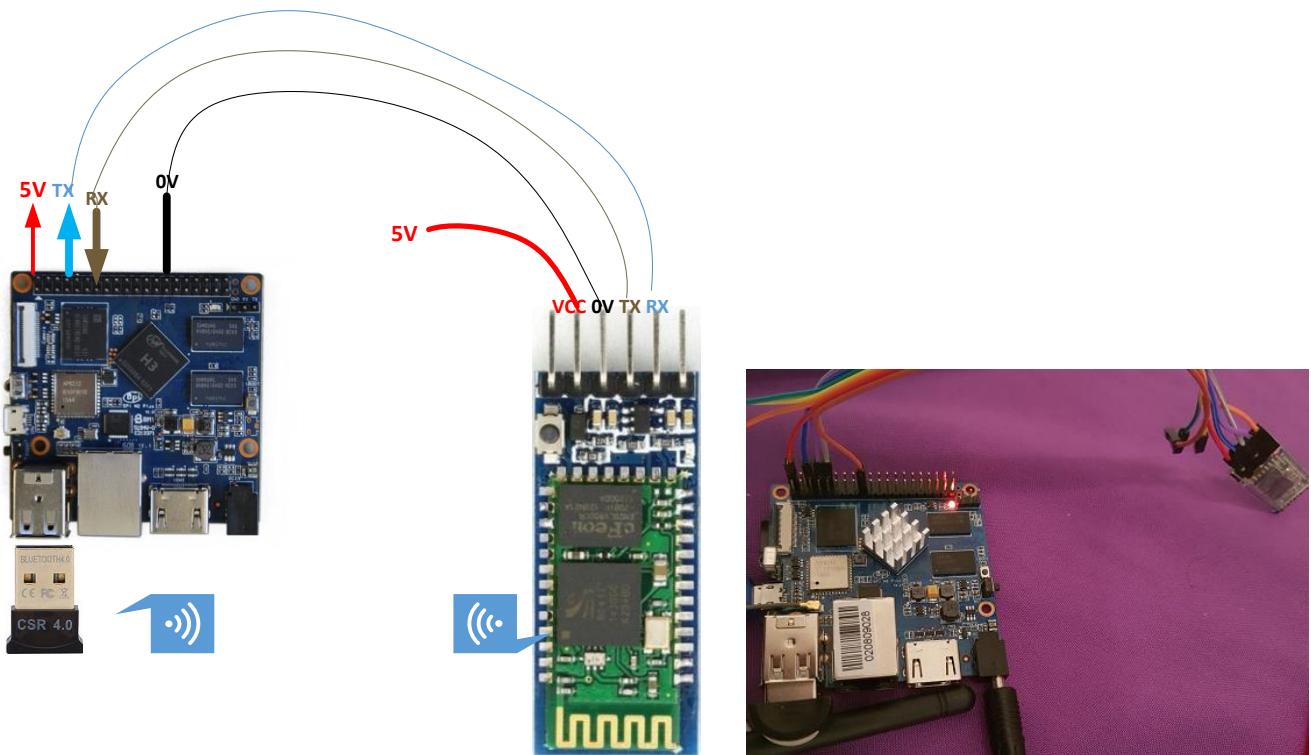


9 Manip bluetooth et HC05

9.1 Introduction

Nous allons tester la communication bluetooth entre le bananapi M2+ et le module HC05 (ou HC06). Le driver du module AP6212A qui gère le wifi et le bluetooth ne fonctionne pas avec le bluetooth. Nous allons donc ajouter un dongle bluetooth compatible Linux (il en existe un grand nombre (DSD tech, Plugable USB adapter, Panda Bluetooth, <https://wonderfulengineering.com/10-best-bluetooth-dongles-for-raspberry-pi-suitable-for-any-project/>).

Le principe de fonctionnement est simple : nous allons utiliser le com TTYS3 pour se connecter au module HC05 (ou HC06) qui est par défaut configuré en mode slave. Nous utiliserons le dongle USB en mode Master afin de scanner les modules bluetooth et nous connecterons au HC05 (ou HC06) en mode SPP. L'utilisation de 2 fenêtres putty permettront alors de communiquer en bluetooth entre le dongle et le module HC05 (ou HC06).



9.2 Le bluetooth

Pour plus d'information sur le bluetooth, on pourra aller voir sur le site de commentcamarche :

<https://www.commentcamarche.net/contents/107-fonctionnement-du-bluetooth>

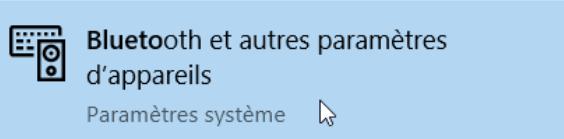
<http://www.linux-magazine.com/Issues/2017/197/Command-Line-bluetoothctl>

9.3 Premier test sous windows ou sous Android

On pourra tester le HC05 ou HC06 directement sous windows ou sous Android (blueterm ou serial bluetooth terminal par exemple) en connexion bluetooth.

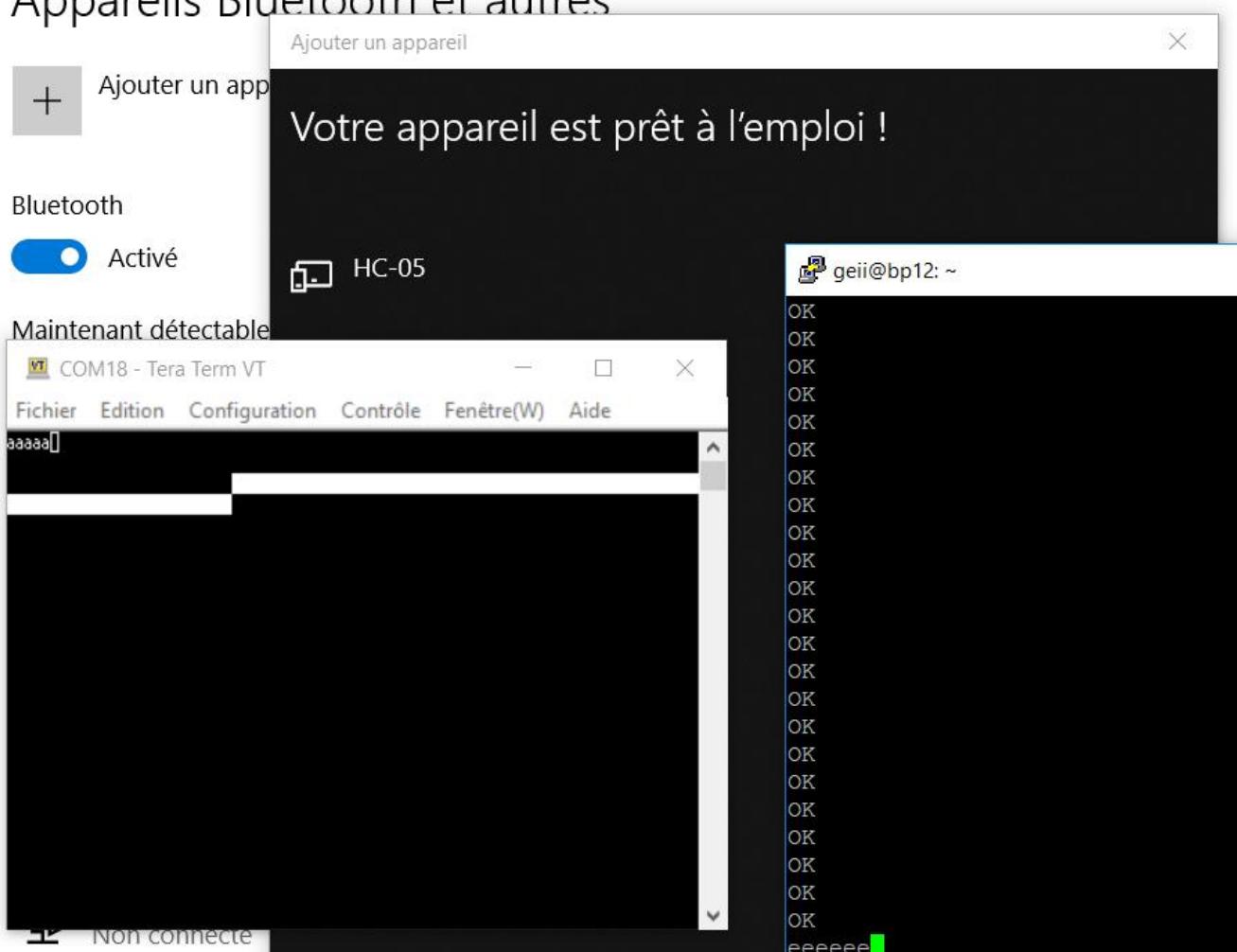
- ✓ *Sous Windows 10 , dans la zone de recherche tapez bluetooth*

Meilleur résultat



- ✓ *Puis cliquer sur ajouter un appareil (il faudra activer le bluetooth au préalable si ce n'est pas déjà fait)*
- ✓ *Vous devriez voir apparaître le module HC-05 ou HC-06, il suffit alors de se connecter (code 1234) puis d'ouvrir le COM bluetooth généré (ici le COM18) sous teraterm par exemple.*
- ✓ *Avec putty connectez-vous au bananapi (on suppose que le com ttyS3 a été activé). Vous pouvez lancer la commande sudo screen /dev/ttyS3 38400 pour le HC05 ou 9600 pour le HC06*

Appareils Bluetooth et autres



Remarque : une fois l'appairage effectué, vous êtes connecté au module bluetooth HC05 (ou HC06). Le module qui clignotait avant la connexion ne doit plus clignoter, la led rouge reste allumée. La connexion est établie, vous devriez voir apparaître un port comm sous windows.

Le HC06 a une vitesse de communication par défaut 9600 bauds alors que le HC05 a vitesse par défaut 38400 et surtout le HC05 peut être configuré en mode maître ou esclave alors que le HC06 n'est qu'en mode esclave.

Quelques commandes pour le HC06 :

AT, répond « OK ».

AT+VERSION, renvoie la version du HC06

AT+NAMEb11, le HC06 répond « OKSetname » et le module bluetooth prend le nom « bp11 ».

AT+BAUD8, le HC06 répond « OK115200 »,

code	Vitesse (bauds)
1	1 200
2	2 400
3	4 800

4	9 600
5	19 200
6	38 400
7	57 600
8	115 200

Quelques informations : <https://knowledge.parcours-performance.com/arduino-bluetooth-hc-05-hc-06/>

Pour le HC05, comme pour le HC06, il faudra vérifier la version du firmware afin de pouvoir faire une recherche google de la documentation de la version du firmware utilisée.

Quelques commandes pour le HC05 :

AT\r\n le HC05 renvoie OK

AT+VERSION?\r\n le HC05 renvoie la version du firmware

AT+ROLE=1\r\n ou bien **AT+ROLE=0\r\n**, permet de changer de mode (1=master/0=slave)

AT+ROLE ?\r\n, renvoie le mode par défaut

AT+UART=115200\r\n, modifie la vitesse par défaut.

9.4 Connexion avec le bananapi

- ✓ Déconnectez le module HC05 (ou HC06) de votre PC ou votre téléphone. Vérifier que la led clignote. Nous allons maintenant connecter le bananapi M2+ à notre HC05 (ou HC06)

Nous allons tout d'abord vérifier que le dongle bluetooth a été détectée.

Lancer la commande de visualisation des logs (option -f pour avoir les messages en temps réel).

```
journalctl -f
```

- ✓ Placer le dongle sur le bannapi

Vous devriez voir apparaître les messages issus du noyau.

```
Feb 25 18:51:27 bp11 kernel: Bluetooth: Core ver 2.22
Feb 25 18:51:27 bp11 kernel: NET: Registered protocol family 31
Feb 25 18:51:27 bp11 kernel: Bluetooth: HCI device and connection manager initialized
Feb 25 18:51:27 bp11 kernel: Bluetooth: HCI socket layer initialized
Feb 25 18:51:27 bp11 kernel: Bluetooth: L2CAP socket layer initialized
Feb 25 18:51:27 bp11 kernel: Bluetooth: SCO socket layer initialized
Feb 25 18:51:27 bp11 kernel: usbcore: registered new interface driver btusb
Feb 25 18:51:27 bp11 systemd[1]: Starting Load/Save RF Kill Switch Status...
Feb 25 18:51:27 bp11 systemd[1]: Reached target Bluetooth.
Feb 25 18:51:27 bp11 systemd[1]: Started Load/Save RF Kill Switch Status.
```

On aurait aussi pu utiliser la commande dmesg.

Vous pouvez aussi vérifier que le bluetooth n'est pas verrouillé.

```
rfkill list
```

```
geii@bp11:~$ rfkill list
0: phy0: Wireless LAN
    Soft blocked: no
    Hard blocked: no
1: hci0: Bluetooth
    Soft blocked: no
    Hard blocked: no
```

On suppose que le paquet bluez est installé

```
dpkg -s bluez
```

Si ce n'est pas le cas, installer le paquet bluez

```
sudo apt install bluez
```

Une fois le paquet installé, vérifier que le service bluetooth est lancé

```
service --status-all | grep bluetooth
```

```
geii@bp11:~$ service --status-all | grep bluetooth
[ + ]  ap6212-bluetooth
[ + ]  bluetooth
```

Vous pouvez alors scanner le bluetooth avec la commande bluetoothctl

```
bluetoothctl
scan on
```

```
[bluetooth]# scan on
Discovery started
[CHG] Controller 00:1F:81:00:08:30 Discovering: yes
[NEW] Device 3C:6A:A7:41:9D:E3 3C-6A-A7-41-9D-E3
[NEW] Device 98:D3:31:FB:68:D2 98-D3-31-FB-68-D2
[CHG] Device 98:D3:31:FB:68:D2 LegacyPairing: no
[CHG] Device 98:D3:31:FB:68:D2 Name: HC-05
[CHG] Device 98:D3:31:FB:68:D2 Alias: HC-05
[CHG] Device 3C:6A:A7:41:9D:E3 LegacyPairing: no
[CHG] Device 3C:6A:A7:41:9D:E3 Name: PC-SALVAT
[CHG] Device 3C:6A:A7:41:9D:E3 Alias: PC-SALVAT
[bluetooth]# devices
Device 3C:6A:A7:41:9D:E3 PC-SALVAT
Device 98:D3:31:FB:68:D2 HC-05
[CHG] Device 3C:6A:A7:41:9D:E3 LegacyPairing: yes
[CHG] Device 3C:6A:A7:41:9D:E3 RSSI: 28
[CHG] Device 98:D3:31:FB:68:D2 LegacyPairing: yes
[CHG] Device 98:D3:31:FB:68:D2 RSSI: 28
[bluetooth]# pair 98:D3:31:FB:68:D2
Attempting to pair with 98:D3:31:FB:68:D2
[CHG] Device 98:D3:31:FB:68:D2 Connected: yes
Request PIN code
[agent] Enter PIN code: 1234
```

Une fois le scan effectué, il est possible de stopper le scan (commande scan off).

Le scan ne montrer que les nouveaux devices bluetooth découverts. Pour visualiser tous les devices bluetooth (les nouveaux et ceux qui ont déjà été découvert) on pourra lancer la commande devices

```
devices
```

Vous pouvez aussi faire une demande d'information sur un id particulier

```
Info XX :XX :XX :XX :XX :XX
```

Si vous voyez le module HC05 (ou HC06), vous pouvez vous connecter en utilisant la commande pair

```
pair XX :XX :XX :XX :XX :XX
```

Cette manip ne sera faite qu'une seule fois, tapez 1234.

Vous êtes maintenant connecté au HC05.

Sortez de l'interface de commande bluetoothctl

```
quit
```

Vous pouvez maintenant créer un comm associé au module HC05 –ou HC06)

```
sudo rfcomm connect /dev/rfcomm1 XX :XX :XX :XX :XX :XX
```

Votre comm existe (/dev/rfcomm1) vous pouvez maintenant lancer une communication bluetooth

```
sudo screen /dev/rfcomm1
```

Et un autre putty connecté au HC05

```
sudo screen /dev/ttys3
```

A la prochaine connexion, il suffira juste de vérifier que le contrôleur bluetooth existe et de monter un com

```
geii@bp12:~$ bluetoothctl
[NEW] Controller 00:1F:81:00:08:30 bp12 [default]
[NEW] Device 98:D3:31:FB:68:D2 HC-05
[NEW] Device 98:D3:32:70:5A:00 HC-06
[CHG] Controller 00:1F:81:00:08:30 Powered: yes
[CHG] Controller 00:1F:81:00:08:30 Alias: bp12
[CHG] Controller 00:1F:81:00:08:30 UUIDs: 00001801-0000-1000-8000-00805f9b34fb
[CHG] Controller 00:1F:81:00:08:30 UUIDs: 0000110e-0000-1000-8000-00805f9b34fb
[CHG] Controller 00:1F:81:00:08:30 UUIDs: 00001200-0000-1000-8000-00805f9b34fb
[CHG] Controller 00:1F:81:00:08:30 UUIDs: 00001800-0000-1000-8000-00805f9b34fb
[CHG] Controller 00:1F:81:00:08:30 UUIDs: 0000110c-0000-1000-8000-00805f9b34fb
[CHG] Controller 00:1F:81:00:08:30 Pairable: yes
[bluetooth]# quit
[DEL] Controller 00:1F:81:00:08:30 bp12 [default]
geii@bp12:~$ sudo rfcomm connect /dev/rfcomm1 98:D3:32:70:5A:00
[sudo] password for geii:
Connected /dev/rfcomm1 to 98:D3:32:70:5A:00 on channel 1
Press CTRL-C for hangup
```

9.5 En cas de problème

Si on a des problèmes avec bluetoothctl, lorsque vous lancer la commande bluetoothctl, vous ne voyez apparaitre aucun controller.

```
geii@bp12:~$ bluetoothctl
[bluetooth]# list
[bluetooth]# devices
No default controller available
[bluetooth]#
```

Utiliser la commande sudo . Si la commande sudo ne fonctionne pas, vérifiez avec la commande hciconfig –a pour vérifier que le contrôleur bluetooth existe

```
[bluetooth]# quit
geii@bp12:~$ hciconfig -a
hci0:  Type: Primary  Bus: USB
      BD Address: 00:1F:81:00:08:30  ACL MTU: 1021:4  SCO MTU: 180:1
      UP RUNNING  PS SCAN
      RX bytes:1232 acl:14 sco:0 events:56 errors:0
      TX bytes:877 acl:13 sco:0 commands:31 errors:0
      Features: 0xff 0x3e 0x09 0x76 0x80 0x01 0x00 0x80
      Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
      Link policy: RSWITCH HOLD SNIFF
      Link mode: SLAVE ACCEPT
      Name: 'bp12'
      Class: 0x000000
      Service Classes: Unspecified
      Device Class: Miscellaneous,
      HCI Version: 2.0 (0x3)  Revision: 0x44
      LMP Version: 2.0 (0x3)  Subversion: 0x3
      Manufacturer: Cambridge Silicon Radio (10)
```

Dans ce cas, le mieux est de débrancher le dongle USB et le rebrancher

On pourra aussi vérifier que le service bluetooth est activé

```
sudo rfkill unblock bluetooth
```

Et que le service bluetooth est activé et fonctionnel.

```
sudo systemctl status bluetooth.service
```

S'il n'est pas lancé, le lancer

```
sudo systemctl start bluetooth.service
```

On peut alors lancer le scan (une autre solution qu'avec bluetoothctl)

```
sudo hciconfig hci0 up
```

```
geii@bp11:/etc/systemd/system$ sudo hciconfig hci0 up
geii@bp11:/etc/systemd/system$ sudo hciconfig -a
hci0:  Type: Primary  Bus: USB
        BD Address: 00:1F:81:00:08:30  ACL MTU: 1021:4  SCO MTU: 180:1
        UP RUNNING
        RX bytes:1296 acl:0 sco:0 events:49 errors:0
        TX bytes:202 acl:0 sco:0 commands:48 errors:6
        Features: 0xff 0x3e 0x09 0x76 0x80 0x01 0x00 0x80
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy: RSWITCH HOLD SNIFF
        Link mode: SLAVE ACCEPT
        Name: 'Accel-OB2'
        Class: 0x000000
        Service Classes: Unspecified
        Device Class: Miscellaneous,
        HCI Version: 2.0 (0x3)  Revision: 0x44
        LMP Version: 2.0 (0x3)  Subversion: 0x3
        Manufacturer: Cambridge Silicon Radio (10)
```

On pourra aussi vérifier que le service SP existe pour notre HC05

```
geii@bp12:~$ sdptool search SP 98:D3:32:70:5A:00 HC-06
Inquiring ...
Searching for SP on 98:D3:32:70:5A:00 ...
Service Name: Dev B
Service RecHandle: 0x10004
Service Class ID List:
    "Serial Port" (0x1101)
Protocol Descriptor List:
    "L2CAP" (0x0100)
    "RFCOMM" (0x0003)
        Channel: 1
Language Base Attr List:
    code_ISO639: 0x656e
    encoding: 0x6a
    base_offset: 0x100
```

Et aussi lancer le scan. Puis se connecter.

```
geii@bp12:~$ sudo hcitool scan
Scanning ...
      3C:6A:A7:41:9D:E3      PC-SALVAT
      98:D3:31:FB:68:D2      HC-05
geii@bp12:~$ sudo rfcomm connect /dev/rfcomm1 98:D3:31:FB:68:D2
Connected /dev/rfcomm1 to 98:D3:31:FB:68:D2 on channel 1
Press CTRL-C for hangup
```

9.6 Pour aller plus loin...

<https://cdn-learn.adafruit.com/downloads/pdf/reverse-engineering-a-bluetooth-low-energy-light-bulb.pdf>

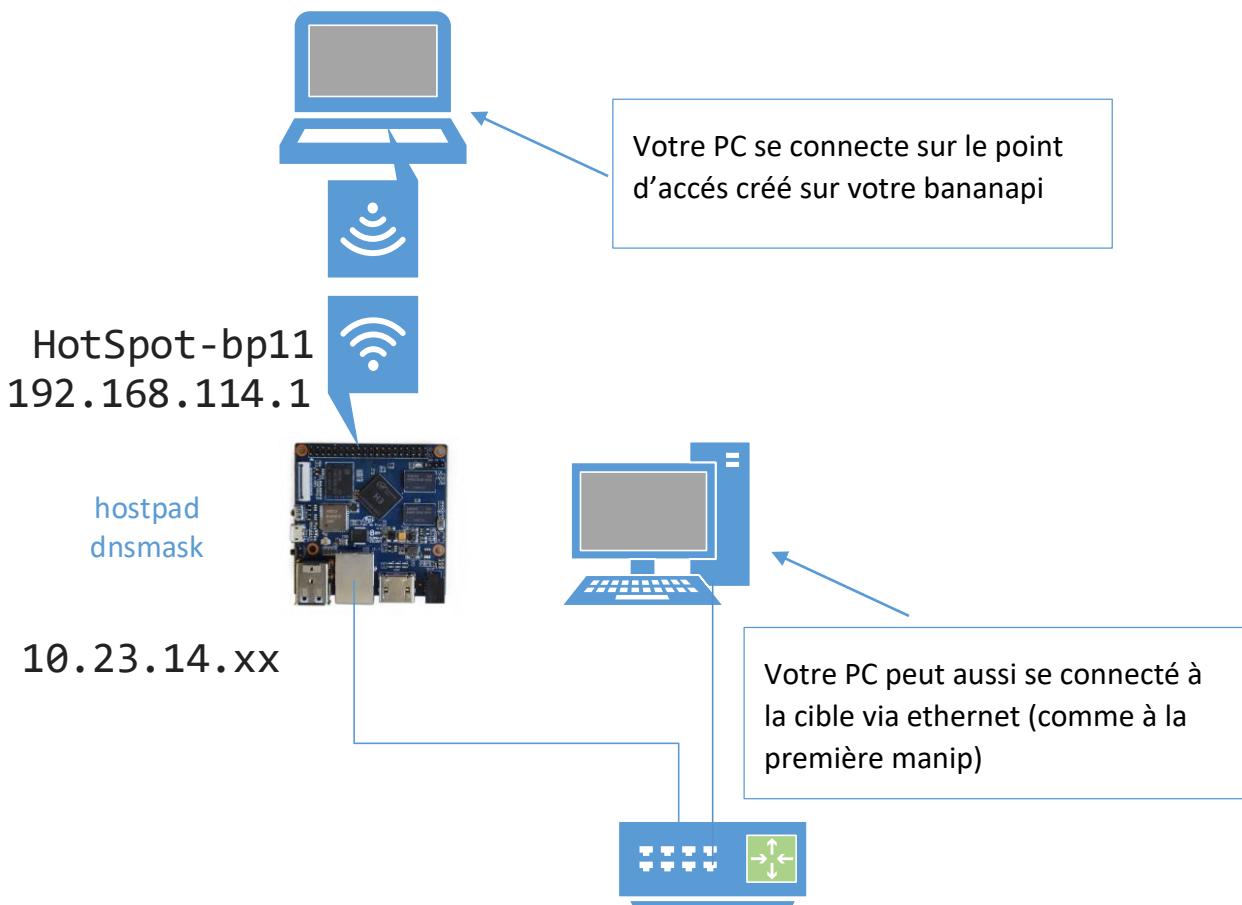
<https://www.linuxjournal.com/content/bluetooth-hacks>

<https://wiki.archlinux.fr/bluetooth>

Si on veut la connexion à chaque fois <http://owen-hsu.blogspot.com/2017/01/rpi-hc-05.html>

10 Crédit d'un point d'accès HotSpot

Nous allons voir ici une autre façon de se connecter à votre bananapi qui est par défaut configuré en station (client Wifi). Cette solution est plus compliquée à mettre en œuvre puisque nous allons passer notre cible en point d'accès Wifi. Dans cette configuration, il est nécessaire de connecter votre bananapi à Ethernet et d'installer le service dnsmasq (serveur DHCP + serveur DNS) service qui sera utilisé pour donner une adresse IP et gérer les noms pour les stations wifi qui se connecteront à votre cible et le service hostapd permettant de gérer le protocole wifi de votre point d'accès.



Nous allons voir 2 façons de configurer notre bananapi en un point d'accès :

- ✓ La première solution consiste à créer le point d'accès à la main (avec un ensemble de commandes que nous allons expliquer). Cette solution ne fonctionnera qu'une fois. Lors du reboot de votre cible, la configuration en point d'accès sera perdue.
- ✓ La deuxième solution, comme vous le verrez est beaucoup plus simple à mettre en œuvre, elle utilise un script dans lequel se trouvent l'ensemble des commandes utilisées dans la première solution. Le lancement du point d'accès est alors fortement simplifié puisqu'elle ne nécessite qu'une ligne de commande (le lancement du script).

10.1 Solution 1 : configuration du hostspot à la main

La création d'un point d'accès nécessite 2 services

- ✓ hostapd (service de gestion du point d'accès)
- ✓ dnsmasq (server DHCP et DNS)

Installer les 2 services. Hostapd est déjà installé sur votre distribution.

```
sudo apt-get install hostapd dnsmasq
```

Puis passer en mode root pour toute la suite, cela va éviter d'utiliser la commande sudo avant chaque commande

```
sudo su
```

Vérifier avec la commande **ifconfig wlan0** que votre SOC Wifi AP6212 est fonctionnel. Dans ce cas la commande ifconfig wlan0 devrait vous renvoyer quelque chose comme ça. On suppose que la carte wifi n'a pas déjà été configurée et apairée à un point d'accès. Si c'est le cas, cela ne changera pas la manip.

```
ifconfig wlan0
```

```
wlan0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
      ether 10:d0:7a:79:c8:05 txqueuelen 1000 (Ethernet)
      RX packets 130 bytes 10589 (10.5 KB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 49 bytes 6935 (6.9 KB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Nous allons tout d'abord arrêter le service **wpa_supplicant** qui est le service qui s'occupe de connecter la cible à un point d'accès.

```
systemctl stop wpa_supplicant
```

Puis nous allons passer la carte wifi en mode managed, c'est le mode utilisé pour passer en mode AP (access Point). On arrête la carte wifi, on passe en mode managé, puis on relance la carte wifi.

```
ifconfig wlan0 down
iwconfig wlan0 mode managed
ifconfig wlan0 up
```

Vérifier que la carte wifi est toujours valide et fonctionnelle. Le fait d'avoir changé le mode de la carte wifi n'a rien changé à la disponibilité du wifi pour notre système.

```
ifconfig wlan0
```

Vous allez maintenant dans le répertoire de root, créer le répertoire ap. Ce répertoire va accueillir les fichiers de configuration hostapd.conf et dnsmasq.conf de vos 2 services. Normalement ces fichiers

devraient être placé dans /etc mais nous allons au plus simple. Le but est ici de montrer de façon simple comment monter notre point d'accès à la main...

```
sudo mkdir /root/ap
cd /root/ap
```

Puis vous allez maintenant créer le fichier hostapd.conf, fichier de configuration de votre point d'accès.

```
nano hostapd.conf
```

hostapd.conf

```
interface=wlan0
driver=nl80211
ssid=bp11
hw_mode=g
channel=5
macaddr_acl=0
ignore_broadcast_ssid=0
auth_algs=1
wpa=2
wpa_key_mgmt=WPA-PSK
rsn_pairwise=TKIP
wpa_passphrase=12345678
```

Une fois le fichier hostapd.conf validé, lancer le serveur hostapd en tache de fond (pour avoir la console putty encore utilisable).

```
hostapd hostapd.conf &
```

```
root@bp11:~/ap# Configuration file: hostapd.conf
wlan0: Could not connect to kernel driver
Using interface wlan0 with hwaddr 10:d0:7a:c8:05 and ssid "bp11"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
```

La gestion du wifi est lancé, vous devriez voir apparaître bp11 dans la zone de recherche Wifi de votre PC ou téléphone portable. Par contre, vous ne pourrez pas encore obtenir une adresse IP puisque le serveur DHCP (dnsmasq n'est pas encore configuré et lancé).

Nous allons maintenant configurer le wifi de notre cible. Son adresse sera 192.168.114.1

```
ifconfig wlan0 up 192.168.114.1 netmask 255.255.255.0
```

```
wlan0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
      inet 192.168.114.1 netmask 255.255.255.0 broadcast 192.168.114.255
        ether 10:d0:7a:c8:05 txqueuelen 1000 (Ethernet)
          RX packets 130 bytes 10589 (10.5 KB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 49 bytes 6935 (6.9 KB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Nous allons maintenant ajouter une table de routage pour donner l'adresse de la gateway (gw)

```
route add -net 192.168.114.0 netmask 255.255.255.0 gw 192.168.114.1
```

```
sudo nano dnsmasq.conf
```

dnsmasq.conf

```
interface=wlan0
dhcp-range=192.168.114.100,192.168.114.150,255.255.255.0,12h
dhcp-option=3,192.168.114.1
dhcp-option=6,192.168.114.1
server=8.8.8.8
log-queries
log-dhcp
listen-address=127.0.0.1
```

Quelques explications pour comprendre ce fichier :

- ✓ **interface:** Access Point Interface, wlan0
- ✓ **dhcp-range:** IP range for nodes, de l'adresse 192.168.114.100 à 192.168.114.150
- ✓ **dhcp-option:3:** Gateway IP, ici 192.168.114.1
- ✓ **dhcp-option:6:** DNS, ici 192.168.114.1

Le fichier dnsmasq.conf, sauvé, vous pouvez lancer le serveur dhcp.

```
dnsmasq -C dnsmasq.conf -d &
```

```
root@bp11:~/ap# dnsmasq -C dnsmasq.conf -d
```

```
dnsmasq: failed to create listening socket for port 53: Address already in use
```

Comme on peut le voir, le lancement du service DHCP est empêché par un autre service qui utilise déjà ce port. Ce service est systemd-resolved, nous allons l'arrêter.

```
systemctl stop systemd-resolved
```

On pourra voir si le service n'est plus utilisé en utilisant la commande

```
netstat -anlp | grep -w LISTEN
```

Le port 53 est maintenant libre, le serveur DHCP dnsmasq devrait pouvoir être lancé.

```
dnsmasq -C dnsmasq.conf -d
```

```
root@bp11:~/ap# dnsmasq -C dnsmasq.conf -d
dnsmasq: started, version 2.79 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus i18n IDN DHCP DHCPv6 no-Lua
TFTP conntrack ipset auth DNSSEC loop-detect inotify
dnsmasq-dhcp: DHCP, IP range 192.168.114.100 -- 192.168.114.150, lease time 12h
dnsmasq: using nameserver 8.8.8.8#53
dnsmasq: reading /etc/resolv.conf
dnsmasq: using nameserver 8.8.8.8#53
dnsmasq: using nameserver 192.168.1.1#53
dnsmasq: using nameserver fe80::327c:b2ff:fed4:b169%eth0#53
dnsmasq: read /etc/hosts - 6 addresses
```

Nous n'avons pas lancé le serveur en tache de fond. C'est pas grave, nous pouvons exécuter une deuxième fois putty pour se connecter une deuxième fois et finir la configuration de notre HotSpot.

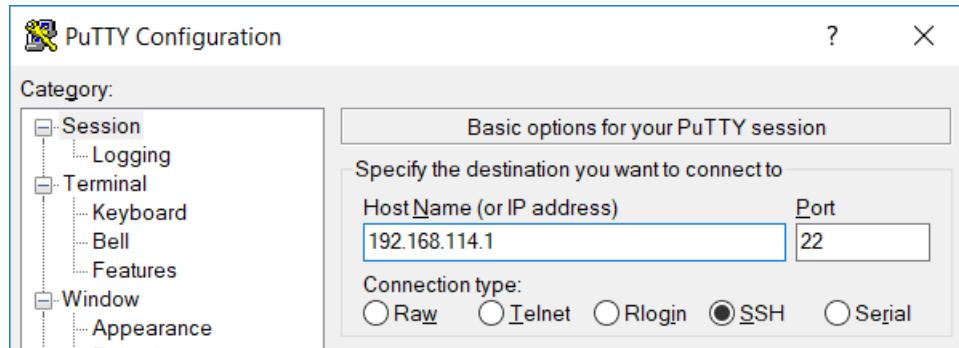
```
root@bp11:~/ap# dnsmasq -C dnsmasq.conf -d
dnsmasq: started, version 2.79 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus i18n IDN DHCP DHCPv6 no-Lua
TFTP conntrack ipset auth DNSSEC loop-detect inotify
dnsmasq-dhcp: DHCP, IP range 192.168.114.100 -- 192.168.114.150, lease time 12h
dnsmasq: using nameserver 8.8.8.8#53
dnsmasq: reading /etc/resolv.conf
dnsmasq: using nameserver 8.8.8.8#53
dnsmasq: using nameserver 192.168.1.1#53
dnsmasq: using nameserver fe80::327c:b2ff:fed4:b169%eth0#53
dnsmasq: read /etc/hosts - 6 addresses
```

192.168.1.13 - PuTTY
login as: geii
geii@192.168.1.13's password: [REDACTED]

Notre HotSpot est presque terminé, il manque un dernier élément permettant de transférer toutes les trames venant du Wifi vers ethernet et vice versa pour que notre PC une fois connectée à la cible, puisse accéder à Internet (qui est sur eth0) au travers du wifi (wlan0). C'est la commande iptables et l'activation au niveau du noyau du port forwarding qui va permettre cela.

```
iptables --table nat --append POSTROUTING --out-interface eth0 -j MASQUERADE
iptables --append FORWARD --in-interface wlan0 -j ACCEPT
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Voila, nous avons terminé la configuration de notre Hotspot. Evidemment ce fut un peu fastidieux et surtout les services ne seront pas relancés lors du prochain démarrage. Le but était ici de rentrer dans le détail de cette configuration pour commencer à intégrer quelques notions que nous verrons plus loin.



```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.114.1 netmask 255.255.255.0 broadcast 192.168.114.255
        ether 10:d0:7a:c8:05 txqueuelen 1000 (Ethernet)
          RX packets 1462 bytes 289082 (289.0 KB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 760 bytes 265808 (265.8 KB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

10.2 Solution 2 : Hotspot utilisant un script

Dans cette deuxième solution, le hotspot avec être créé par un script qui va lancer toutes les commandes que nous avons du faire à la main. Le principe d'un script est simple : mettre un ensemble de commande bash dans un fichier et exécuter ce fichier. C'est ce que nous allons tester.

Pour cette manip placer vous sur votre répertoire geii.

```
git clone https://github.com/oblique/create_ap
cd create_ap
sudo make install
```

```
geii@bp11:~$ ls
geii@bp11:~$ git clone https://github.com/oblique/create_ap
Cloning into 'create_ap'...
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1034 (delta 0), reused 0 (delta 0), pack-reused 1033
Receiving objects: 100% (1034/1034), 340.18 KiB | 1.09 MiB/s, done.
Resolving deltas: 100% (571/571), done.
```

Une fois les fichiers installés. Vous pouvez lancer le script de lancement de votre hotspot. Tout est expliqué dans le fichier README.md.

- ✓ *tester le point d'accès (ici sans mot de passe) OPEN, il faut compter au moins 30s avant que le point d'accès apparaisse.*

```
sudo create_ap wlan0 eth0 bp11
```

Une fois la commande effectuée, vous pourrez vous connecter au réseau bp11 (dans notre cas).

SSID :	bp11
Protocole :	802.11g
Type de sécurité :	WPA2 - Personnel
Bande passante réseau :	2,4 GHz
Canal réseau :	1
Adresse IPv4 :	10.0.0.165
Serveurs DNS IPv4 :	10.0.0.1

11 Compléments

11.1 Les systèmes de fichier

11.1.1 Introduction

Nous avons utilisé au cours de ce document les notions de système de fichier ext4, les notions de noyaux et de modules ou drivers. Nous allons dans cette partie tenter d'expliquer ces concepts de façon plus précise et nous donnerons des exemples permettant de mieux comprendre ces notions.

Commençons donc par la notion de système de fichier. Nous verrons ensemble 2 types de système de fichier, le système de fichier FAT16 (utilisé encore dans l'embarqué) et ext2 (même principe que ext3 et ext4) utilisé par Linux. Il en existe bien d'autre, citons FAT32, ntfs (utilisé par windows) iso9660 (pour les CD), mais aussi des systèmes de fichier moins connus comme reiserfs, unionfs, pour ne citer qu'eux.

Un système de fichier décrit la manière dont les données sont organisées sur un support de masse (disque dur, mmc, sd ou clé usb). Commençons par nous intéresser au système de fichier FAT12 utilisé à l'époque sur les disquettes et capable de lire jusqu'à 8Mo de données. Même si ce système de fichier n'est plus utilisé, il est simple à comprendre et permet d'introduire les principaux concepts des systèmes de fichier.

Quels que soient les supports utilisés au cours de l'histoire des systèmes de fichier (cassette, disquette, CD, DVD, disque dur, mmc, sd,), les données sont lues octets par octets. Le système de fichier va regrouper ces octets par blocs ou secteurs (pour les disques durs) de 512 ou 1024 ou 2048 ou 4096 octets, assigner un numéro à chacun de ces secteurs et gérer chaque bloc lors de la création ou l'effacement de fichiers ou répertoires.

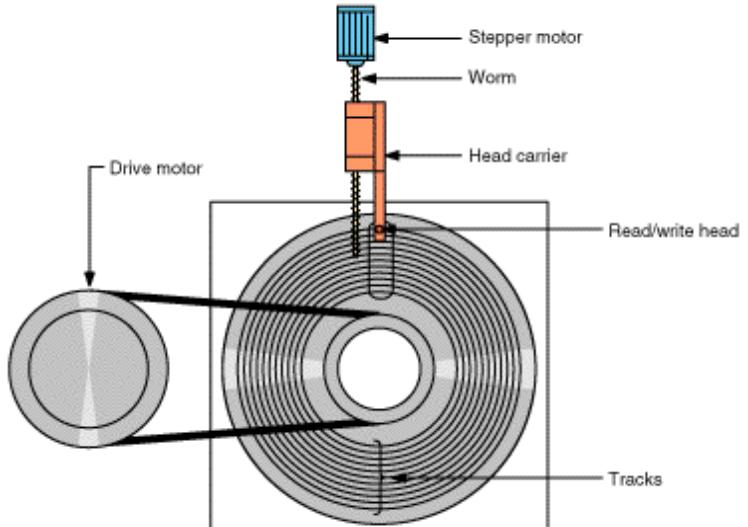
11.1.2 Système de fichier FAT12 (FAT16 et FAT32)

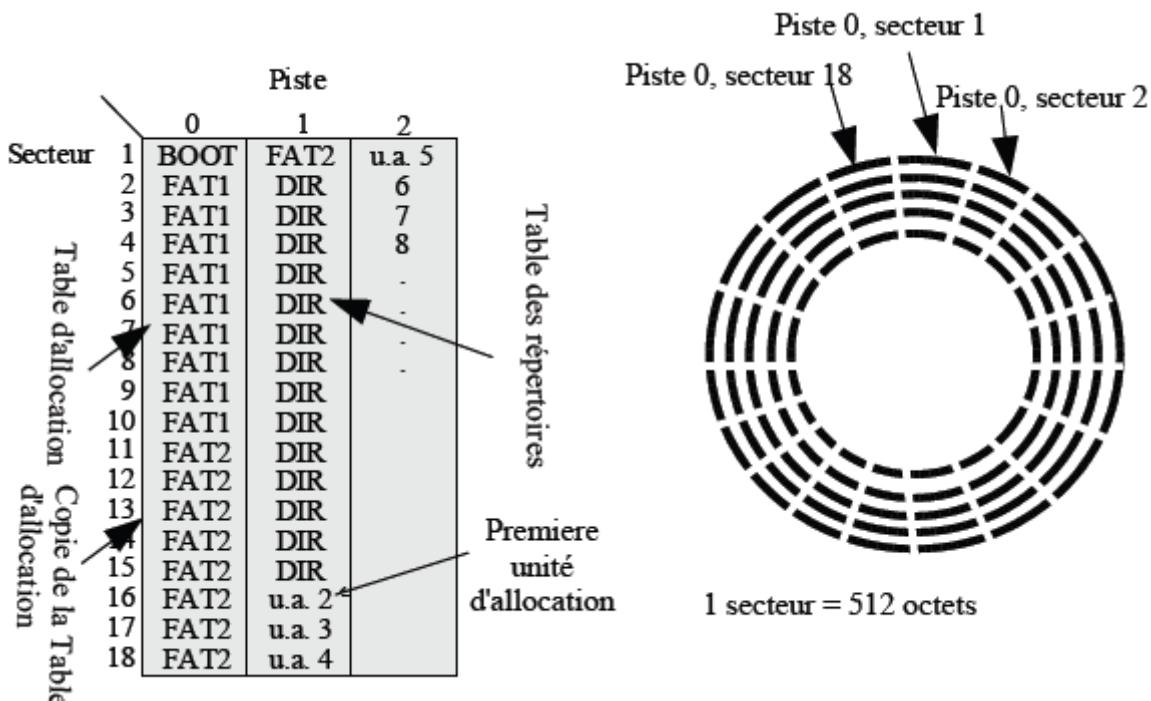
Passons au système de fichier FAT12 utilisées dans les disquettes par Microsoft dans les années 90.

Les pistes ou tracks sont découplés en secteurs. Il y a **18 secteurs par piste** sur une disquette. Le formatage bas-niveau consiste à découper le support magnétique en pistes et secteurs. Chaque secteur est numéroté et on lui rajoute des informations de synchronisation (préambule) et un CRC (suffixe pour vérifier le support). Pour un secteur d'information utile de **512 octets** il faut **571 octets** vrai.

A la différence avec les **disques durs** qui doivent être impérativement **partitionnés**, une disquette ne possède pas de MBR et donc ne possède pas de table de partitions.

Le formatage haut niveau va permettre d'écrire les informations sur le **boot sector** (premier secteur), la **table d'allocation** (**FAT1** sur 9 secteurs), la copie de la table d'allocation (noté **FAT2** sur la figure 22) et la table des répertoires (**root folder** sur 14 secteurs). Viennent ensuite les secteurs dans lesquels seront enregistrés les différents fichiers (commence à ua2, unité d'allocation).





Le boot sector :

Le premier secteur (piste 0 secteur 1) d'une disquette est le secteur de boot. Ce secteur de boot, s'il est correctement rempli permet de lancer le système d'exploitation.

Avec **windows** c'est la commande **format a: /s** qui permet de formater une disquette de boot. Si la disquette n'est pas bootable (la disquette n'a pas été formatée avec l'option /s) alors la première instruction **jmp 0x3E**, instruction de saut vers le programme de boot , n'existe pas. Le tableau ci-dessous donne les informations importantes se trouvant dans le premier secteur. Dans le secteur de boot à l'adresse **0x1FE** se trouve le magic number qui permet au système d'exploitation de savoir si la disquette est bootable.

secteur de boot d'une disquette

adresse	explication
0x000	jmp 0x3E
0x03	paramètres du système de fichiers (nombre de secteurs, taille d'un secteur, , taille de la fat, ...)
0x3E	programme de boot
0x1FE	magic number

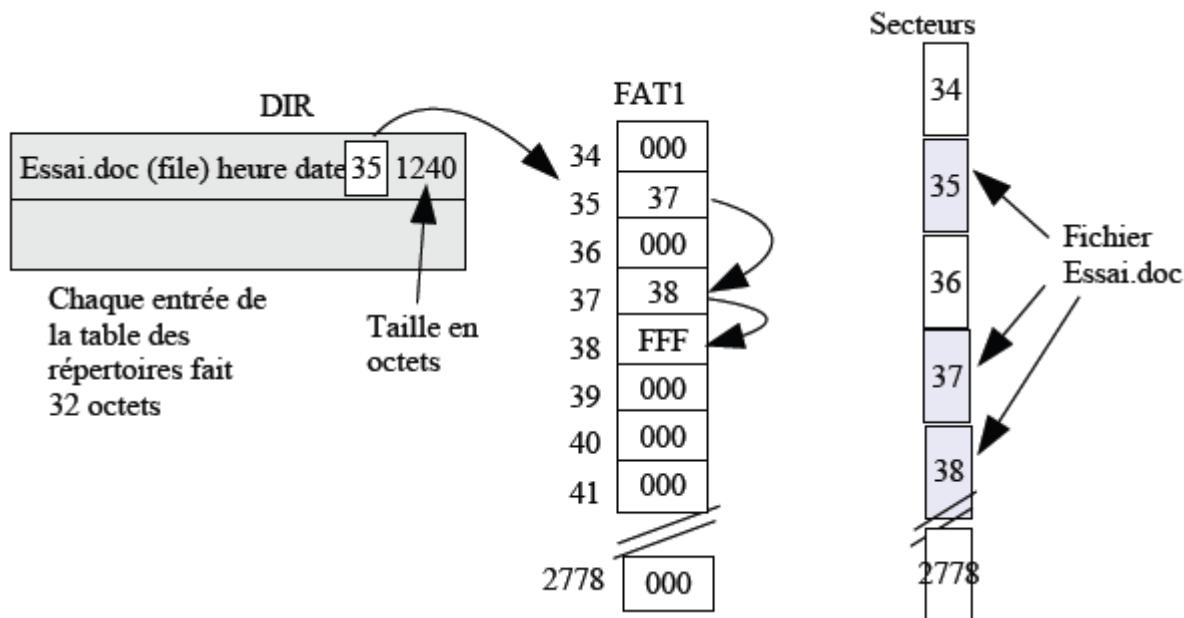
La FAT (File Allocation Table) ou table d'allocation :

Pour connaître quels sont les secteurs utilisés pour un fichier ou un répertoire, il faut un tableau ou **table d'allocation** dont chaque case pointe sur un secteur. Chaque case de la table d'allocation peut prendre la valeur 0 (dans le cas où le secteur correspondant est vide) , la valeur FFF (dans le cas où le secteur associé correspond à la fin d'un fichier) ou le numéro du secteur suivant . Les cases de la table d'allocation contiennent un pointeur vers le prochain secteur du fichier.

Prenons un exemple: le fichier **Essai.doc** est constitué de 3 secteurs, les secteurs 35, 37 et 38 . Dans ce cas, la table d'allocation aura à la case 35 la valeur 37 (le prochain secteur) , et ainsi de suite jusqu'à atteindre la fin du fichier (valeur FFF dans la case 38). Pour rendre cette structure accessible, il manque l'information permettant de connaître le premier secteur du fichier. Cette information est codée dans la **table des répertoires (DIR)** dans laquelle figure aussi le **nom du fichier**.

Pour les disquettes utilisant le système de fichier **FAT12**, la taille d'une **case de la FAT est de 12 bits** (1 octet et demi). De plus la FAT étant l'élément essentiel du système de fichier, il existe une copie de celle-ci . On aura donc l'information dupliqué notée **FAT2**.

Les valeurs de chaque case étant codées sur 12 bits, la valeur maximale d'une case est $2^{12} - 1 = 4095$. Etant donné que chaque case pointe vers un secteur de 512 octets, la taille max accessible est $4095 * 512$ octets, soit une taille de 2 Mo. Donc le système **FAT12** ne peut adresser que 2 Mo de mémoire (suffisant pour disquette de 1,44 Mo).



Pour connaître le nombre de cases nécessaire pour parcourir tous les secteurs de la disquette, il suffit de diviser $1,44 \text{ Mo} / 512 = 2812$ cases. En utilisant 9 secteurs pour la FAT, on peut coder 3072 secteurs soit plus que les 2812 secteurs contenus dans une disquette de 1,44 Mo.

Le root folder ou table des répertoires.

Pour connaître le nom d'un fichier (ou d'un répertoire), sa date de création, sa taille et sa place sur la disquette, on utilise une table des répertoires. Chaque entrée dans la table des répertoires fait 32 octets.

Détail des 32 octets de la table des répertoires

offset	longueur	Explication
0x00	8	Nom du fichier
0x08	3	extension
0x0B	1	attribué
0x0C	10	réservé
0x16	2	heure création
0x18	2	date création
0x1A	2	premier secteur
0x1C	4	taille

Sachant que le root folder fait 14 secteurs, le **nombre de fichiers et répertoires max** enregistrables sur une disquette est $14 * 512 / 32 = 224$.

Il existe d'autres systèmes de fichiers FAT utilisés pour les disques durs. Il existe le **FAT16** (la table d'allocation est codée sur 16 bits ce qui permet d'accéder au maximum à 65535 clusters, et la **FAT32** (codé sur 32 bits) utilisé par Windows 98. Le système de fichier **FAT16** et **FAT32** utilisent la notion de **cluster**.

Un **cluster** peut faire **2,4,8 secteurs** et constitue l'unité de base de la FAT. Ce qui fait que les cases de la FAT pointent vers des clusters et non plus sur des secteurs. En général, ces clusters font **4 ko** (8 secteurs). Avec windows xp ou windows 2000, il y a eu l'apparition d'un nouveau système de fichier : **ntfs**, plus puissant qui ajoute la notion de droits sur les fichiers (comme pour le système de fichiers ext2).

Ces systèmes de fichiers propres au monde Windows peuvent être lues et écrits par Linux (sauf pour ntfs) mais ne sont pas utilisés pour accueillir Linux. Celui-ci utilise le système de fichier **ext2** que nous allons étudier.

- ✓ Pour créer un système de fichier FAT12 ou FAT16 sur une disquette

```
mkfs.mdos /dev/fd0
```

- ✓ Pour créer un système de fichier FAT32 sur une partition hda1

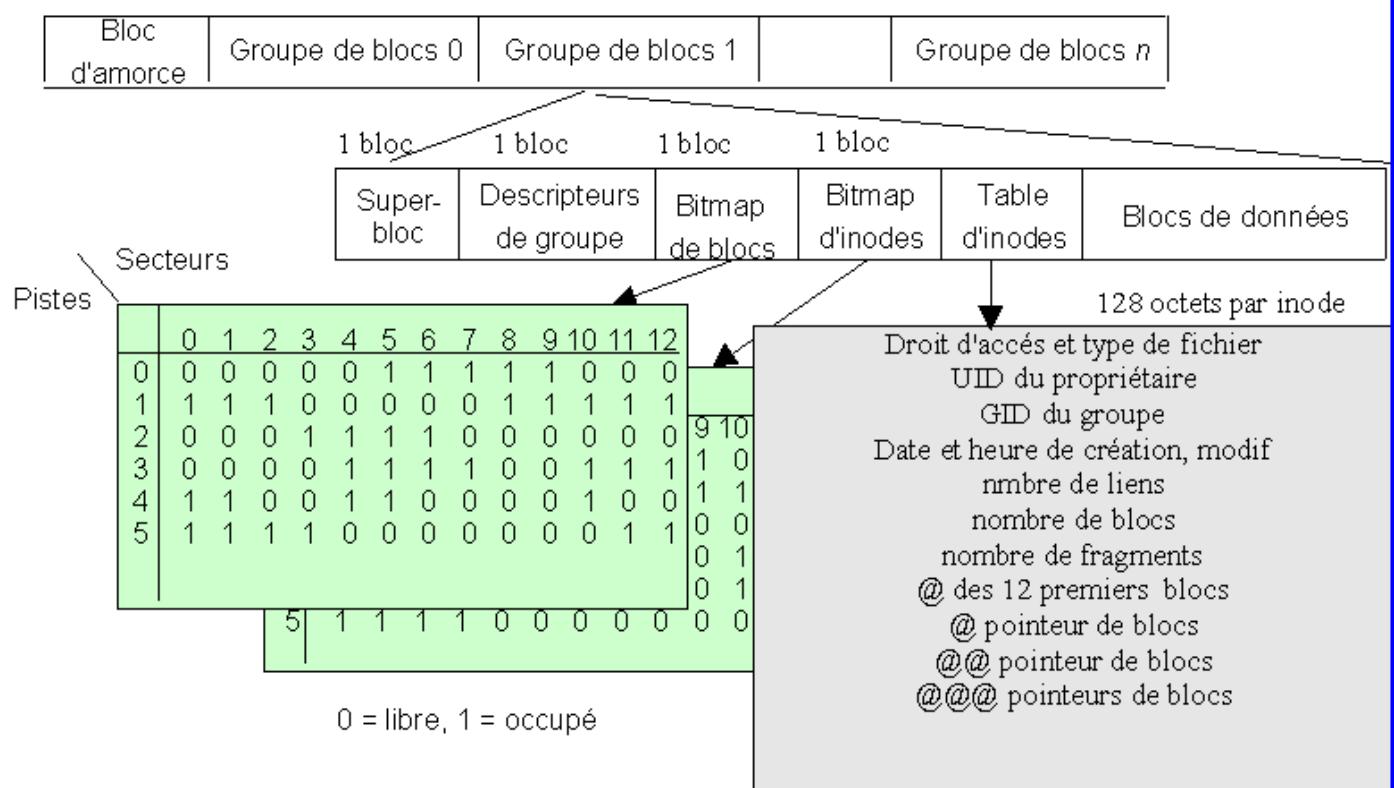
```
mkfs.vfat /dev/hda1
```

11.1.3 Le système de fichier ext2 (ext3, ext4)

Le système de fichier ext2 utilise 2 notions : la notion de **blocs de données** (correspond à la notion de **cluster** sous Windows) qui constitue l'unité de base du système de fichier (un bloc de données est un ensemble de secteurs) et la notion **d'inode**. Un **inode** regroupe toutes les informations nécessaires à la localisation des blocs constituant le répertoire ou le fichier. Il possède de plus de nombreuses informations sur les **droits d'accès, le numéro du propriétaire, du groupe, ...**.

Pour connaître les blocs de données libres, ext2 utilise la technique du **bitmap**: chaque bit de ce bloc bitmap pointe vers un bloc de données, avec l'information libre (valeur 0) ou occupé (valeur 1). Comme il faut aussi avoir une information sur les inodes utilisés et ceux libres, il existe un **bitmap d'inode**. Enfin pour connaître le nom d'un fichier, il faut une **table des répertoires** dans laquelle on associe un nom de fichier ou de répertoire à un inode. Cette table des répertoires se trouve sur le premier bloc de données.

Le lanceur du système d'exploitation se trouve dans le bloc d'amorce ou **boot sector**. Ext2 découpe la partition en **groupe de bloc**.



Chaque groupe de blocs contient une **copie du superbloc**.

Dans Linux, la taille d'un **inode** est de **128 octets**.

Un fichier est constitué d'un ensemble de blocs. chaque bloc regroupe **2, 4 ou 8 secteurs** (taille d'un bloc = **1,2 ou 4 ko**).

Pour connaître le nombre de blocs utilisés par la table d' inodes , il suffit de connaître le nombre d'inodes par blocs. Pour 184 inodes, et pour une taille de blocs de 1 ko, un bloc peut contenir 8 inodes. Il faudra donc $184/8=23$ blocs pour avoir la place de coder 184 inodes. Ce qui permet de coder 184 fichiers ou répertoires au maximum dans un groupe de bloc.

Les **bitmaps** occupent chacun 1 bloc ou u.a. Ceci limite donc la taille des groupes à 8 192 blocs pour des blocs de 1 Ko. ou 32 768 blocs pour des blocs de 4 Ko. Les inodes sont répartis également parmi les groupes de blocs. Le nombre d'inodes par groupe ne peut non plus dépasser les nombres ci-dessus.

Le superbloc :

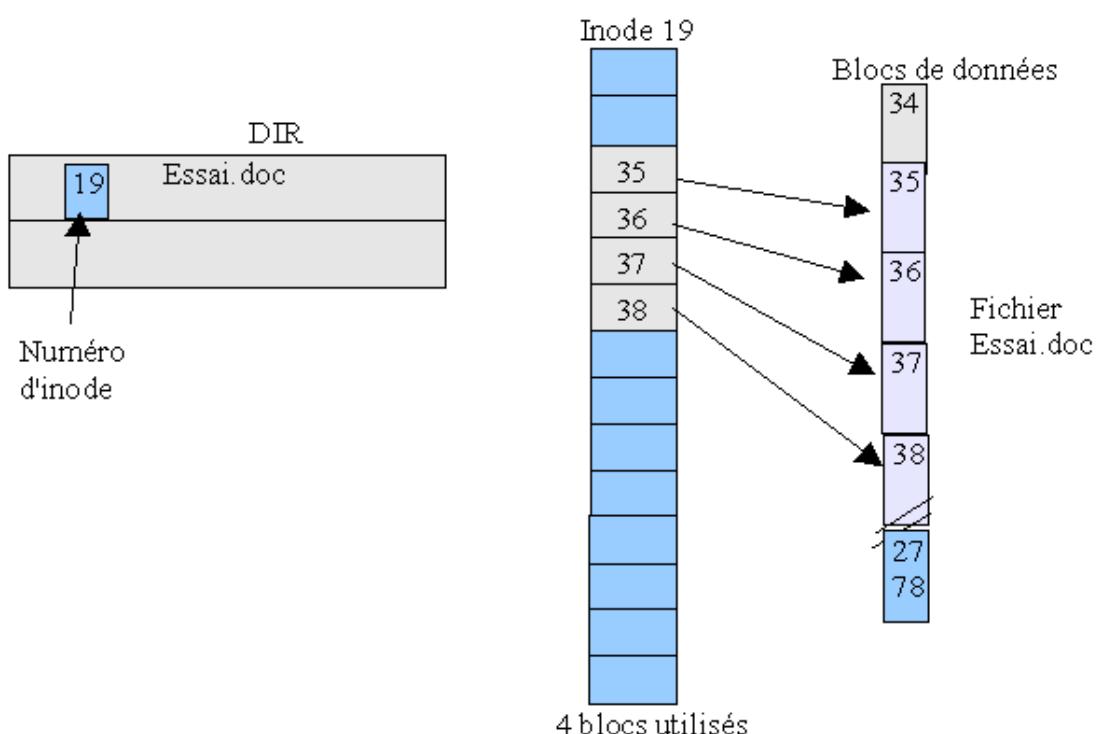
Le super bloc possède les informations clés sur le système de fichiers (nombre d'inodes, nombre de blocs, nombre de blocs réservés, nombre de blocs libres, nombre d'inodes libres, @ du premier bloc de données, taille du bloc, dates, ...).

Le descripteur de groupe (group descriptor) :

Le descripteur de groupe possède les adresses du block bitmap, de l' Inode bitmap, de l' inode table, ..., le nombre de répertoires.

Le root directory (ou table des répertoires)

La table des répertoires se trouve au premier bloc de la zone de données. Cette table permet d'associer un inode à un nom de fichier. Dans l'exemple ci-dessous, on peut voir que le fichier essai.doc est associé au numéro d'inode 19 dans la table des répertoires. L'inode 19 possède l'information sur les blocs de données associés qui sont les blocs 35,36,37 et 38.



- ✓ Pour formater la première partition d'un disque dur sda en ext4

```
mkfs.ext4 /dev/sda1
```

```
mke2fs 1.32 (09-Nov-2002)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
184 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
1 block group
8192 blocks per group, 8192 fragments per group
184 inodes per group

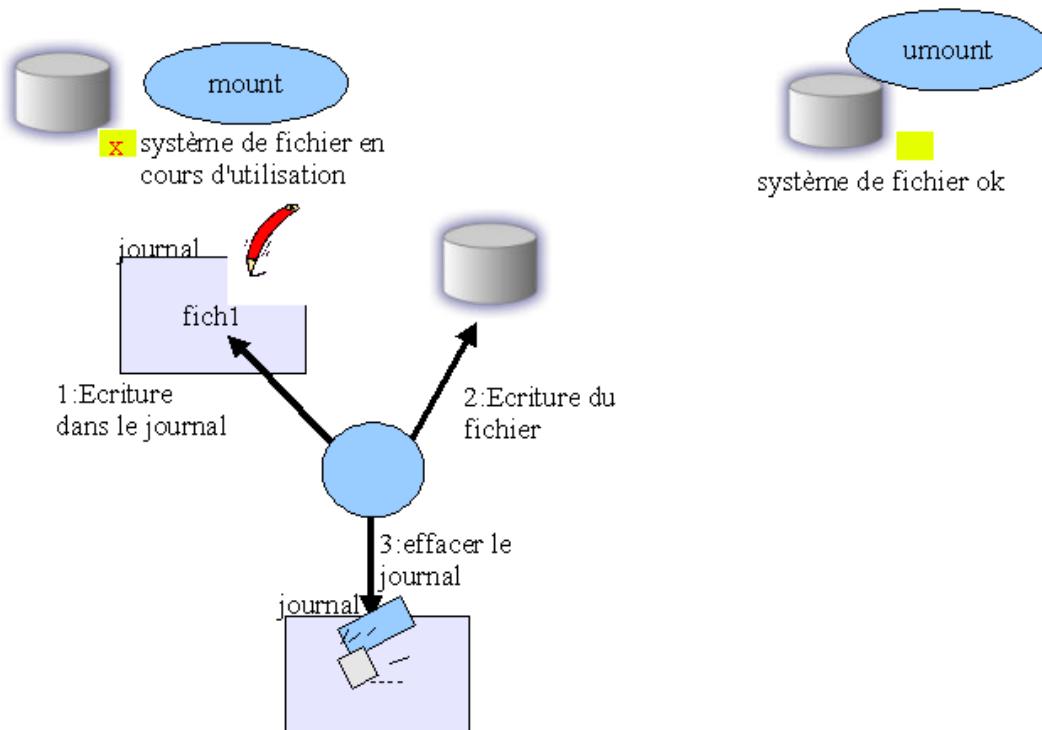
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

Journalisation :

A chaque montage d'une partition un bit est mis à jour et est enlevé par umount. En cas de coupure d'électricité (bit non enlevé), Linux lancera automatiquement e2fsk (file check) sur cette partition au prochain démarrage. Au moment de la coupure d'électricité le système était peut être en train d'écrire dans un fichier ou pire dans les méta blocs (superbloc, table inode,...), ce qui dans ce cas peut provoquer une perte de fichiers, voire plus grave une corruption du système de fichier.



La plupart des systèmes de fichiers actuels sont journalisés (ext3,ext4, xfs, reiserfs). La journalisation du système de fichier rend celui-ci beaucoup plus robuste puisque chaque fois que le système écrit dans la partition celui-ci fait une copie du fichier et du méta bloc associé (ou bien seulement du méta bloc) dans la zone de journal. A la fin , les blocs écrits sont effacés. En cas de crash, il ne reste dans le journal que les fichiers corrompus. e2fsck n'a plus qu'à recopier les fichiers du journal dans la partition.



- ✓ Pour créer un système de fichier ext3

```
mkfs.ext2 -j /dev/hda2
```

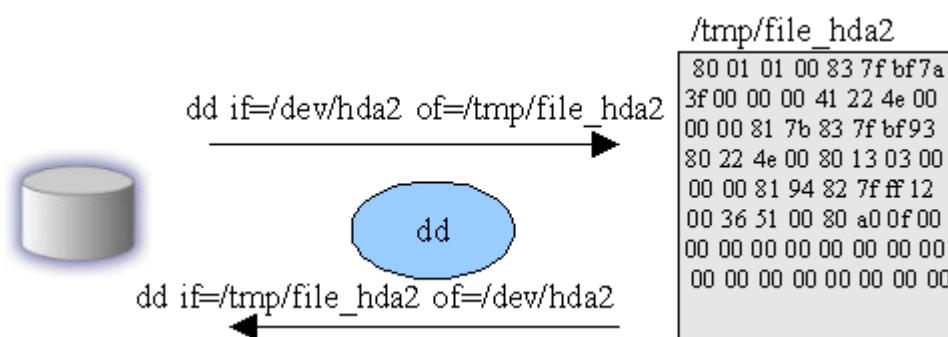
- ✓ Conversion d'un système de fichier ext2 en ext3

```
tune2fs -j /dev/hda2
```

- ✓ Forcer la vérification du système de fichier ext2 (le système de fichier ne doit pas être monté en rw)

```
e2fsck /dev/hda2
```

Copier un système de fichier commande dd



La commande dd permet de copier physiquement tout type de fichier (fichier spécial du répertoire /dev ou autres). La figure 30 nous montre un exemple dans lequel on vient recopier l'image du disque hda2 dans un fichier file_hda2. Ceci n'est possible que si le système de fichiers auquel appartient le fichier n'est pas sur le disque hda2.

Une fois le disque dur recopié, il est possible de faire un **gzip file_hda2** pour compresser l'image du disque.

- ✓ Pour monter l'image du disque dur sur un répertoire /mnt/hda2

```
mount -o loop /tmp/file_hda2 /mnt/hda2
```

11.2 Compilation du noyau : les principes

Le noyau (kernel) est le programme qui se trouve dans le répertoire /boot et qui gère le matériel (RAM, flash, I2C, SPI, RTC,...), le réseau, ordonne les tâches (programmes ou process), gère les systèmes de fichiers. C'est lui qui est chargé en RAM par le bootloader et qui ensuite va lancer le premier process init de votre distribution.

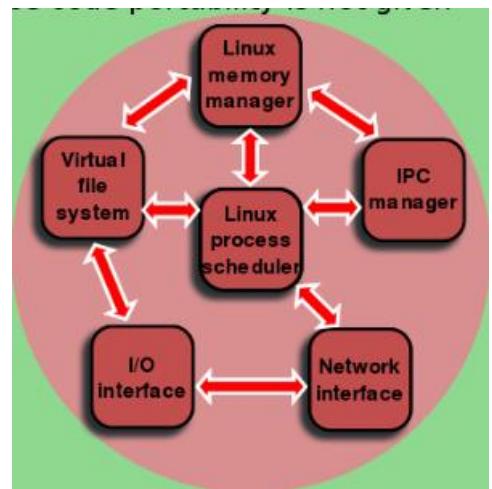
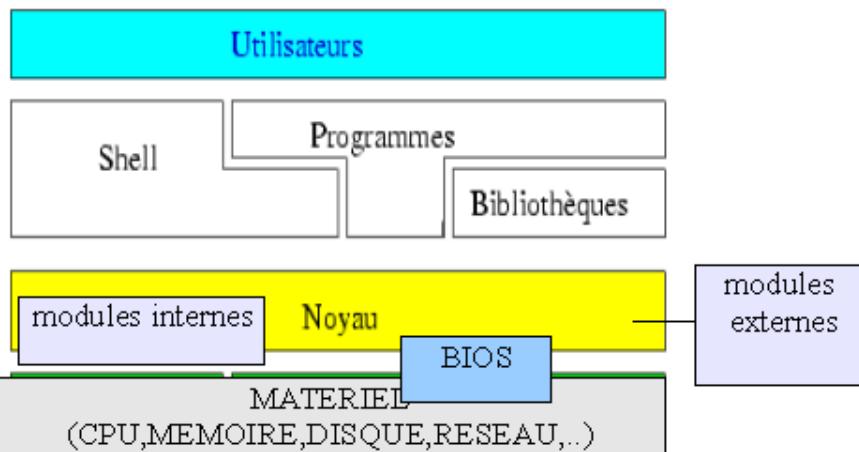


Figure 18 : les fonctions du noyau
(https://fr.wikipedia.org/wiki/Noyau_Linux#/media/File:Linux_kernel_interfaces.svg)



- **Le noyau :** se trouve dans le répertoire /boot et se nomme en général zImage ou bzImage ou vmlinuz-x.y.z (début 2019 : la version du noyau x.y-z = 4.19.20). Le noyau gère entre autres la pile TCP/IP, le scheduler (gestionnaire des tâches), la gestion de la mémoire (mmu), l'accès au matériel (disque dur, carte graphique, contrôleur d'interruption, bus PCI, USB,...). Sa taille est en général comprise entre 1 à 10 Mo. Le noyau est par défaut compressé et c'est le premier fichier qui est décompressé et chargé en RAM par le bootloader (grub pour architecture x86 ou uboot pour ARM).

- ✓ Pour connaître la version du noyau installée sur le système :

```
uname -r
```

- ✓ Pour connaître la taille du noyau installé :

```
du -h /boot/* | grep vmlinuz-$(uname -r)
```

- **Les modules ou drivers : fichiers *.ko :** Les distributions Linux (Ubuntu, Red Hat, Mandriva,...) fournissent un noyau minimal associé à un grand nombre de modules (drivers). Au démarrage de la machine, le noyau charge en mémoire les drivers utiles à la gestion des périphériques détectés.. Cette solution permet à une distribution de fonctionner sur n'importe quelle machine tout en gardant un noyau avec une taille mémoire en RAM minimale. Il est aussi possible d'insérer au noyau un driver de façon dynamique avec la commande *modprobe*. Pour savoir quels sont les

modules actuellement chargés, on pourra utiliser la commande `lsmod`. Le code des drivers se trouvent dans le répertoire `/lib/modules/` avec l'extension `.ko`.

- ✓ *Pour connaître la taille totale des modules sur le système*

```
du -hs /lib/modules/
```

- **Les bibliothèques dynamiques :fichiers `*.so`** : les bibliothèques (équivalentes aux fichiers `dll` sous windows) se trouvent principalement dans les répertoires `/lib` et `/usr/lib` (on pourra faire un `find / -name *.so`), ce sont des fichiers compilés en mode utilisateurs à la différence avec les drivers qui eux sont compilés en mode kernel (accès direct au matériel et à toute la mémoire).

- ✓ *Pour connaître les `dlls` et la taille mémoire de ces `dll` :*

```
du -ch /lib/arm-linux-gnueabihf/*.so
```

```
du -ch /usr/lib/arm-linux-gnueabihf/*.so
```

- ✓ *Les applications sont liées à ces `dll`. Pour connaître les `dlls` liées au bash (interpréteur)*

```
ldd /bin/bash
```

- **Les programmes** : les programmes se trouvent dans `/bin` `/usr/bin` `/sbin` et `/usr/sbin`.

- **Le shell ou interpréteur** : Il existe plusieurs shell sous linux. Le plus utilisé est `bash`. Dans l'embarqué on utilise plutot le shell minimal `ash` (empreinte mémoire faible).

- ✓ *Pour connaître le shell en fonction sur votre distribution. La commande `env` permet de donner les variables d'environnement du shell.*

```
env | grep SHELL
```

11.3 Compilation du noyau

11.3.1 Introduction

Le noyau Linux est la brique essentielle à tout système embarqué sous Linux, Android et Mac puisque ces 3 systèmes d'exploitation utilisent le noyau Linux pour la partie bas niveau. Le noyau est mis à jour continuellement et peut être téléchargé sur <https://www.kernel.org/doc/html/latest/>.

Les sources du noyau englobe toutes les architectures de microprocesseurs (X86,amd64,arm, arm64, mips,...) et tous les drivers. Ainsi lorsque l'on télécharge les sources du noyau pour le compiler, ces sources du noyau font plus de 1Go. C'est lors de la compilation du noyau que l'on choisit l'architecture (type de microcontrôleur ou microprocesseur, plateforme) mais aussi les drivers à insérer (soit dans le noyau soit comme module).

Pour les processeurs Allwinner qui sont basés sur l'architecture arm, une branche de développement linux <https://github.com/linux-sunxi/linux-sunxi/> est régulièrement intégré au noyau (kernel.org) par Linux Torvald qui est à la base du versionnement des sources du noyau.

Vous pouvez voir le résumé des développements faits sur cette branche :

http://linux-sunxi.org/Linux_mainlining_effort

11.3.2 Mais pourquoi compiler un noyau ?

Dans la plupart des cas, la compilation du noyau n'est pas nécessaire puisque celui-ci a été compilé et validé par le responsable de la distribution Linux utilisé. Par exemple pour notre cible **armbian**. Cette opération peut être nécessaire si l'on désire ajouter un module ou bien avoir la dernière version du noyau si celui-ci a mis à jour un module qui avait un problème dans la version courante utilisée.

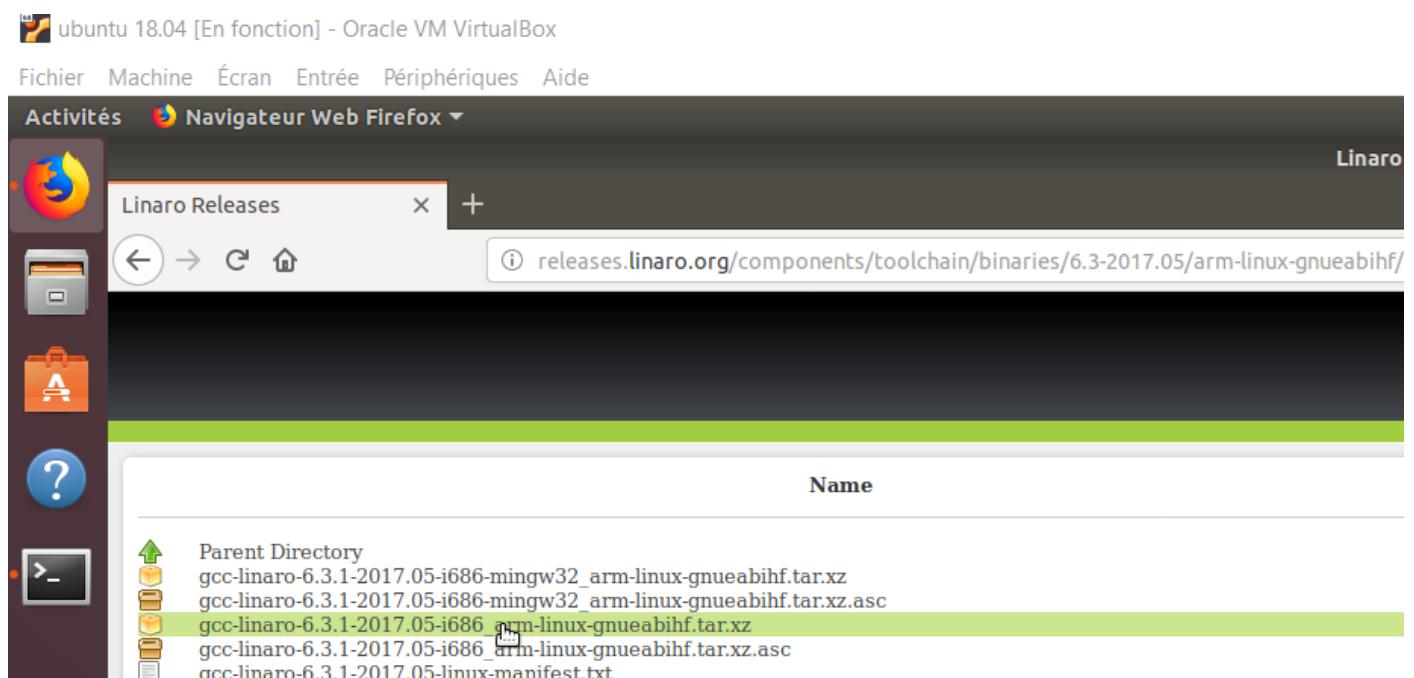
Les solutions de compilation du noyau : il existe 2 solutions pour compiler le noyau pour notre cible. La première solution consiste à télécharger les sources du noyau sur notre cible puis à compiler le noyau directement sur la cible. Cette première solution va prendre plusieurs heures, en effet même si notre cible est à 1.2GHz en 4 coeurs, la compilation du noyau est toujours plus lente sur une cible arm que sur un PC. Pour des PC récents il est possible que la compilation d'un noyau sous Linux prenne dix fois moins de temps que sur le PC que sur la cible. Nous allons donc choisir la deuxième solution : compilation du noyau sur notre PC sous VirtualBox avec une distribution Ubuntu 18.04. Cette solution sera toujours plus lente qu'un PC sous Linux évidemment, mais fait l'économie du double boot (Windows ou Linux).

Nous supposons donc que vous avez installé une machine virtuelle Ubuntu sous VirtualBox ou VmWare et que vous avez lancé cette machine virtuelle.

11.3.3 Installation de la chaîne de développement croisée pour la cible :

Il existe plusieurs solutions pour récupérer une chaîne de développement :

- ✓ L'utilisation des outils de la distribution armbian (outils utilisés pour construire la distribution) https://docs.armbian.com/Developer-Guide_Build-Preparation/
- ✓ Utiliser apt-get install gcc-arm-linux-gnueabihf
- ✓ Télécharger une chaîne de développement prête à l'emploi, c'est cette dernière solution que nous allons tester.



- | |
|--|
| <ul style="list-style-type: none"> ✓ Lancer votre distribution Ubuntu ✓ Télécharger gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf.tar.xz |
|--|

Site de Linaro : <http://releases.linaro.org/components/toolchain/binaries/6.3-2017.05/arm-linux-gnueabihf/>

- ✓ *Extraire le fichier et le copier dans le répertoire /opt*

```
sudo mv gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf /opt
```

- ✓ *Vérifier que votre répertoire /opt contient votre chaîne de développement croisé.*

salvat@salvat-VirtualBox: /opt

Fichier Édition Affichage Rechercher Terminal Aide

```
salvat@salvat-VirtualBox:/opt$ ls -l
total 12
drwxr-xr-x 9 salvat salvat 4096 févr. 26 16:24 gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf
```

Lors de la compilation du noyau nous aurons besoin d'installer des paquets suivant : flex bison et les sources de libncurses et libssl. Nous allons donc installer ces paquets

```
sudo apt install flex bison
sudo apt-get install libncurses5-dev libncursesw5-dev
sudo apt-get install libssl-dev
```

Pour terminer, nous avons besoin d'ajouter notre chaîne de développement au PATH afin de pouvoir appeler le compilateur croisé de n'importe quel répertoire.

```
export PATH=/opt/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf/bin:$PATH
```

On pourra vérifier que le compilateur croisé est accessible

```
arm-linux-gnueabihf-gcc -v
```

Remarque : lors du prochain démarrage la chaîne de compilation ne sera plus dans notre chemin. Il est possible de modifier ceci en ajoutant à la fin du fichier **.bashrc** de notre répertoire, l'ajout du chemin.

Ajouter le chemin en fin de votre fichier **.profile** qui est le fichier exécuté lors de votre connexion

```
echo 'PATH=/opt/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf/bin:$PATH' >> .bashrc
```

ouvrir sous Ubuntu une nouvelle fenêtre et vérifier que la ligne a été ajoutée à **.bashrc**

```
echo $PATH
```

```
salvat@salvat-VirtualBox:~$ echo $PATH
/opt/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf/bin:/home/salvat/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

11.3.4 Télécharger les sources du noyau

La dernière version des sources du noyau pour notre bananpi M2+ se trouve sur le site de linux-sunxi. Nous allons donc télécharger sur notre répertoire les sources du noyau.

```
cd
git clone --depth 1 -b sunxi-next --single-branch https://github.com/linux-sunxi/linux-sunxi
cd linux-sunxi
```

Une fois dans le répertoire, vous pouvez visualiser la taille des sources du noyau (ici 7,8Go)

```
salvat@salvat-VirtualBox:~/linux-sunxi$ du -hs .
1,8G .
```

Les sources du noyau que nous venons de télécharger ont une taille de 1.8Go. Dans ce répertoire se trouvent un certain nombre de répertoires dont le détail est donné ci-dessous.

- ✓ **arch** : contient le code source des architectures x86,ppc, alpha, m68k, arm, ...
- ✓ **Documentation** : contient des fichiers de docs sur les différents modules du noyau
- ✓ **drivers** : contient l'arborescence des drivers (pilotes) de périphérique
- ✓ **fs** : contient les sources de gestions des systèmes de fichiers supportés par Linux (ext2, fat, iso9600)
- ✓ **include** : contient les fichiers d'entête ***.h des différents fichiers C
- ✓ **init** : contient le fichier principal main.c du noyau Linux
- ✓ **ipc** : contient le code source de l'IPC System V du noyau (partage mémoire, sémaphore et messages)
- ✓ **kernel** : contient le source des fonctions majeures du noyau
- ✓ **mm** : contient les fonctions de gestion mémoire (mmu)
- ✓ **net** : contient le code source des différents protocoles réseau (tcp/ip et x25)
- ✓ **scripts** : contient le code source des outils de configuration du noyau

11.3.5 Configuration et compilation du noyau et des modules

Avant de lancer la compilation du noyau, il va falloir le paramétriser. En effet, compiler plus de 1Go de fichier source n'aurait pas de sens, sachant que nous voulons compiler le noyau pour la cible allwinner H3. Il existe 2 solutions :

1. Utiliser le fichier de configuration générique pour les processeurs allwinner se trouvant dans les sources du noyau (arch/arm/configs/sunxi_defconfig)
2. Utiliser le fichier de configuration utilisé lors de la compilation du noyau par l'équipe armbian. Ce fichier est disponible sur la cible.

Dans les 2 cas, il est nécessaire de copier le fichier de configuration choisi dans le fichier .config dans le répertoire linux-sunxi.

Solution 1 : il faudra lancer la commande **cp arch/arm/configs/sunxi_defconfig .config**

Solution 2 : le fichier de configuration utilisé se trouve dans /proc/config.gz . Il faut donc copier le fichier /proc/config.gz et le copier dans sunxi. C'est cette deuxième solution que nous allons choisir. En effet la solution 1 est une solution générique, qui ne prend pas en compte les spécificités du la carte bananapi M2+ alors que la solution 2 reprend les choix de compilation faites par l'équipe ARMBIAN pour recompiler le noyau avec la version 5.0 alors que notre noyau est actuellement compilé en 4.19.20

Sur votre machine virtuelle linux vous pouvez copier le fichier config.gz de votre cible vers votre répertoire courant avec la commande scp, puis dézipper ce fichier.

```
scp geii@bp11:/proc/config.gz config.gz
gunzip < config.gz > config
```

Pour finir, copier ce fichier dans .config

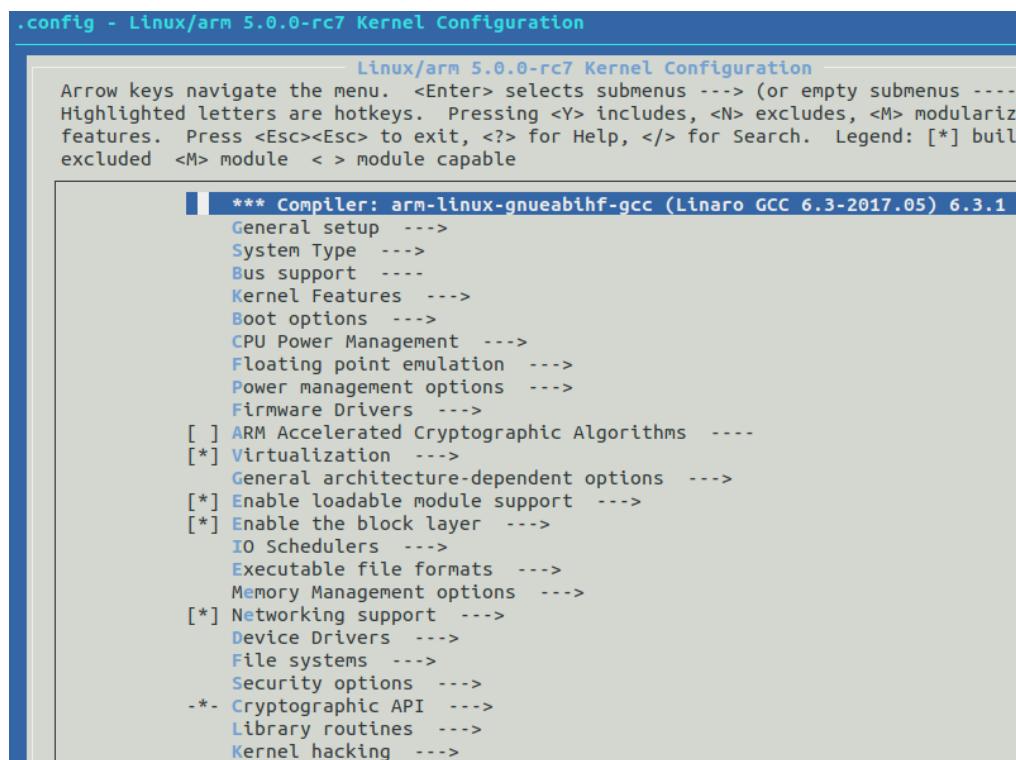
```
cp config .config
```

Notre fichier .config est maintenant défini, nous pouvons commencer le processus de compilation. Pour commencer, nous allons lancer la commande ci-dessous, qui va comparer le fichier de configuration .config utilisé pour compiler le noyau 4.19.20 et le comparer avec les choix proposés par le nouveau noyau 5.0. Lors de ce processus, des questions vont vous être posés, tapez ENTREE (choix par défaut) pour les nouvelles fonctionnalités de votre noyau.

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- oldconfig
```

Vous pouvez maintenant visualiser les différents modules qui vont être compilés. L'option menuconfig de votre Makefile vous permet de visualiser le fichier .config mais de manière plus simple.

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
```



Cette étape n'est pas fondamentale, c'est avec cet outil que vous pourriez ajouter ou enlever des modules, pour l'instant nous allons faire confiance au fichier de configuration téléchargé de notre cible et lancer la compilation du noyau

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j4 zImage
```

La compilation du noyau devrait prendre entre 5mn et 30mn, à la fin de cette compilation vous devriez voir le fichier zImage dans le répertoire arch/arm/boot/

dtbs: le build tree se trouve dans les sources du noyau dans arch/arm/boot/dts/ et est nécessaire lors du lancement du noyau par le bootloader. C'est lui qui paramètre le noyau.

Vous pouvez visualiser ces fichiers dans ~/linux-sunxi/arch/arm/boot/dts . Une fois compilé les fichiers utiles à notre noyau devront être copié dans le répertoire /boot (extension .dtb).

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- dtbs
```

Il ne reste plus qu'à compiler les modules: rappelons que les modules sont les drivers que nous avons choisi de ne pas intégrer dans le noyau et de compiler séparément. En général, le noyau est assez petit et les modules sont compilés séparément et chargé lors du démarrage. Cela permet dans une distribution d'avoir un nombre important de driver sur le disque dur et de n'avoir en RAM que les drivers nécessaires.

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j4 modules
```

Puis créer l'architecture permettant au noyau de pouvoir charger les modules

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- INSTALL_MOD_PATH=output modules_install
```

11.3.6 Copie sur la cible

Notre noyau et les modules sont prêt à être copiés sur la cible.

Copier le noyau :

```
cd arch/arm/boot/
scp zImage geii@bp11:/home/geii/
```

Puis copier le source tree

```
cd dts
scp sun8i-h3-bananapi-m2-plus.dtb geii@bp11:/home/geii/
```

Nous allons devoir maintenant copier les modules. Nous n'allons pas copier ces modules un par un, cela serait trop long, nous allons plutôt archiver le répertoire complet, copier sur la cible et le désarchiver. Les modules ont été compilé dans le répertoire output

```
cd ~/linux-sunxi/output
tar czvf lib.tar.gz lib/
```

Puis on l'envoie sur la cible

```
scp lib.tar.gz geii@bp11:/home/geii/
```

Sur la cible vous devriez avoir les fichiers lib.tar.gz et zImage ainsi que le fichier dtb

```
geii@bp11:~$ ls
bananaM2p config  lib.tar.gz  sun8i-h3-bananapi-m2-plus.dtb  zImage
```

Dézipper l'archive sur votre cible et copier la dans le répertoire /lib/modules ou se trouvent déjà les modules pour le noyau 4.19.20.

```
tar xzvf lib.tar.gz
cd lib/modules
sudo cp -R * /lib/modules/
```

Vous devriez avoir dans /lib/modules le répertoire associé à la nouvelle version du noyau (en plus de celle qui est déjà présente sur la cible).

```
geii@bp11:/lib/modules$ ls
4.19.20-sunxi  5.0.0-rc7-gbf7fc5741
```

Pour finir, copier le noyau et le dtb (toujours sur la cible)

```
cd
sudo cp zImage /boot/vmlinuz-5.0.0-rc7-gbf7fc5741
sudo cp sun8i-h3-bananapi-m2-plus.dtb /boot/
```

```
geii@bp11:/boot$ ls -l
total 30660
-rw-r--r-- 1 root root      201 Feb 26 19:17 armbianEnv.txt
-rw-r--r-- 1 root root    1536 Jan  9 16:08 armbian_first_run.txt.template
-rw-r--r-- 1 root root  230454 Jan  9 16:08 boot.bmp
-rw-r--r-- 1 root root    3726 Jan  9 16:05 boot.cmd
-rw-r--r-- 1 root root   4882 Jan  9 16:08 boot-desktop.png
-rw-rw-r-- 1 root root    3798 Jan  9 16:09 boot.scr
-rw-r--r-- 1 root root 160860 Feb  9 18:02 config-4.19.20-sunxi
lrwxrwxrwx 1 root root        17 Feb 15 06:59 dtb -> dtb-4.19.20-sunxi
drwxr-xr-x 3 root root  12288 Feb 15 06:58 dtb-4.19.20-sunxi
lrwxrwxrwx 1 root root        17 Jan  9 16:07 dtb.old -> dtb-4.19.13-sunxi
-rw-r--r-- 1 root root 8145340 Feb 15 07:02 initrd.img-4.19.20-sunxi
drwxrwxr-x 2 root root    4096 Jan  9 16:08 overlay-user
-rw-r--r-- 1 root root  20558 Feb 27 08:32 sun8i-h3-bananapi-m2-plus.dtb
-rw-r--r-- 1 root root 3235472 Feb  9 18:02 System.map-4.19.20-sunxi
lrwxrwxrwx 1 root root       21 Feb 15 07:02 uInitrd -> uInitrd-4.19.20-sunxi
-rw-r--r-- 1 root root 8145404 Feb 15 07:02 uInitrd-4.19.20-sunxi
-rwxr-xr-x 1 root root 7323200 Feb  9 18:02 vmlinuz-4.19.20-sunxi
-rwxr-xr-x 1 root root 4076776 Feb 27 08:28 vmlinuz-5.0.0-rc7-gbf7fc5741
lrwxrwxrwx 1 root root      21 Feb 15 07:01 zImage -> vmlinuz-4.19.20-sunxi
```

Il ne reste plus qu'à changer le lien zImage qui pointe vers le noyau 4.19.20 et de le faire pointer vers notre nouveau noyau 5.0

```
sudo rm zImage
sudo ln -s vmlinuz-5.0.0-rc7-gbf7fc5741 zImage
```

✓ Vous pouvez rebooter votre cible et vérifier avec uname-r la version du noyau utilisé.

Le test a été fait avec les fichiers dtb de la cible précédemment compilée. Vous pouvez modifier le fichier armbianEnv.txt pour prendre en compte le n

Modif /boot/armbianEnv.txt

fdtfile=sun8i-h3-bananapi-m2-plus.dtb

et on modif verbosity=7 pour avoir plus d'info sur le noyau

Ressources :

http://linux-sunxi.org/Manual_build_howto#boot.cmd

<https://blog.brichacek.net/how-to-compile-linux-kernel-for-orange-pi-zero/>

http://linux-sunxi.org/Sinovoip_Banana_Pi_M2%2B

<https://notsyncing.net/?p=blog&b=2016.orangepi-pc-custom-kernel>

<http://www.orangepi.org/Docs/Building.html>

11.4 Compilation d'un module sur la cible

Une autre solution consiste à garder notre noyau et à lui ajouter un module qui n'avait pas été compilé. A la différence avec la compilation du noyau qui peut prendre plusieurs heures sur la cible, la compilation d'un seul module (ou de quelques modules) peut être fait directement sur la cible. C'est ce que nous allons tester ici.

Télécharger les sources du noyau (ici c'est la version 4.20.14) sur la cible. Nous pourrions choisir une version plus récente.

```
wget https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.20.14.tar.gz
```

Puis on détarre

```
tar xzvf linux-4.20.14.tar.gz
```

On pourra vérifier que les sources du noyau sont bien présentes.

```
ls linux-4.20.14
```

```
geii@bp11:~/linux-4.20.14$ ls
arch      crypto      include   kernel      mm          scripts
block     Documentation init      lib          Module.symvers security
certs     drivers      ipc       LICENSES    net          sound
COPYING   firmware    Kbuild   MAINTAINERS README    tools
CREDITS   fs          Kconfig  Makefile   samples    uinput
```

Nous allons par exemple compiler le driver uinput qui n'a pas été compilé. Pour s'en convaincre, il suffit de faire une recherche du module uinput dans /lib/modules. Vous ne devriez rien avoir.

```
find /lib/modules -name "uinput*"
```

Copie de /proc/config.gz

```
cp /proc/config.gz ~/
gunzip .config
cp .config linux-4.20.14/
```

Puis on modifie de **.config** (m pour module et y pour intégré dans le noyau)

```
cd linux-4.20.14
nano .config
```

```
GNU nano 2.7.4                                     File: .config

# CONFIG_INPUT_CPCAP_PWRBUTTON is not set
# CONFIG_INPUT_ATI_REMOTE2 is not set
# CONFIG_INPUT_KEYSPAN_REMOTE is not set
# CONFIG_INPUT_KXTJ9 is not set
# CONFIG_INPUT_POWERMATE is not set
# CONFIG_INPUT_YEALINK is not set
# CONFIG_INPUT_CM109 is not set
# CONFIG_INPUT_REGULATOR_HAPTIC is not set
CONFIG_INPUT_AXP20X_PKE=y
CONFIG_INPUT_UINPUT=m
# CONFIG_INPUT_PCF8574 is not set
# CONFIG_INPUT_PWM_BEEPER is not set
```

Il ne reste plus qu'à modifier le **Makefile**. En effet la version du noyau utilisé pour la compilation du module 4.20.14 n'est pas celle du noyau actuellement sur la cible 4.20.19. Il est donc important de modifier la version du Makefile, ce numéro de version sera ensuite intégré dans le module compilé et rejeté par le noyau lors de son chargement si celui-ci n'a pas la même version.

Ouvrir le Makefile dans le répertoire linux-sunxi

```
VERSION = 4
PATCHLEVEL = 20
SUBLEVEL = 14
EXTRAVERSION =
NAME = ""
```

Puis modifier ce **Makefile** pour avoir la même version que celle de la cible. Attention à mettre le même nom pour la partie EXTRAVERSION.

L'entête du **Makefile** est alors le suivant :

```
VERSION = 4
PATCHLEVEL = 19
SUBLEVEL = 20
EXTRAVERSION = -sunxi
NAME = "People's Front"
```

Voici le résultat de la commande :

```
nano Makefile
GNU nano 2.7.4                               File: Makefile

# SPDX-License-Identifier: GPL-2.0
VERSION = 4
PATCHLEVEL = 19
SUBLEVEL = 20
EXTRAVERSION = -sunxi
NAME = "people's front"

# *DOCUMENTATION*
# To see a list of typical targets execute "make hel
```

C'est parti, on lance la compilation :

```
make modules_prepare
```

Puis on compile tous les drivers se trouvant dans drivers/input et qui ont été activé (M) dans le fichier de configuration .config

```
make M=drivers/input
```

```
root@bp11:/home/geii/linux-4.20.14# make M=drivers/input
AR      drivers/input/misc/built-in.a
CC [M]  drivers/input/misc/uinput.o
CC [M]  drivers/input/rmi4/rmi_spi.o
CC [M]  drivers/input/rmi4/rmi_smbus.o
CC      drivers/input/touchscreen/of_touchscreen.o
AR      drivers/input/touchscreen/built-in.a
```

Remarque : pour savoir où se trouve le fichier uinput.c de notre module, il suffit de lancer la recherche avec la commande : **find . -name uinput.c**

```
geii@bp11:~$ find linux-4.20.14/ -name uinput.c
linux-4.20.14/drivers/input/misc/uinput.c
```

On peut voir que le fichier uinput.c (source de notre module) se trouve dans **drivers/input**. Il n'est pas possible de compiler le module seul, nous devons compiler les modules se trouvant dans drivers/input et qui ont été activé dans le fichier

A la fin de la compilation , vérifiez que le module **uinput.ko** a bien été compilé

```
find . -name uinput.*  
root@bp11:/home/geii/linux-4.20.14/drivers/input# find . -name uinput*  
./tmp_versions/uinput.mod  
./misc/uinput.ko  
./misc/uinput.c  
./misc/uinput.mod.o  
./misc/uinput.mod.c  
./misc/uinput.o
```

Il ne reste plus qu'à le copier dans la zone des modules

```
cp ~/linux-4.20.14/drivers/input/misc/uinput.ko /lib/modules/4.19.20-sunxi/extramodules/
```

Premier test : avec l'outil **insmod** nous pouvons insérer le module et voir si celui-ci est accepté et chargé par le noyau

```
insmod /lib/modules/4.19.20-sunxi/extramodules/uinput.ko
```

Une fois le module chargé, vérifiez avec la commande dmesg que le module est chargé

```
[ 8099.898196] uinput: loading out-of-tree module taints kernel.  
[ 8346.267847] input: lircd-uinput as /devices/virtual/input/input2
```

Le module s'est chargé correctement, il ne reste plus qu'à activé le module pour le prochain démarrage et de tester les fonctionnalités de ce nouveau module

```
depmod -a
```

Puis on reboot

```
sudo reboot
```

On pourra après le reboot vérifier avec lsmod que le module uinput a bien été chargé

```
lsmod | grep uinput
```

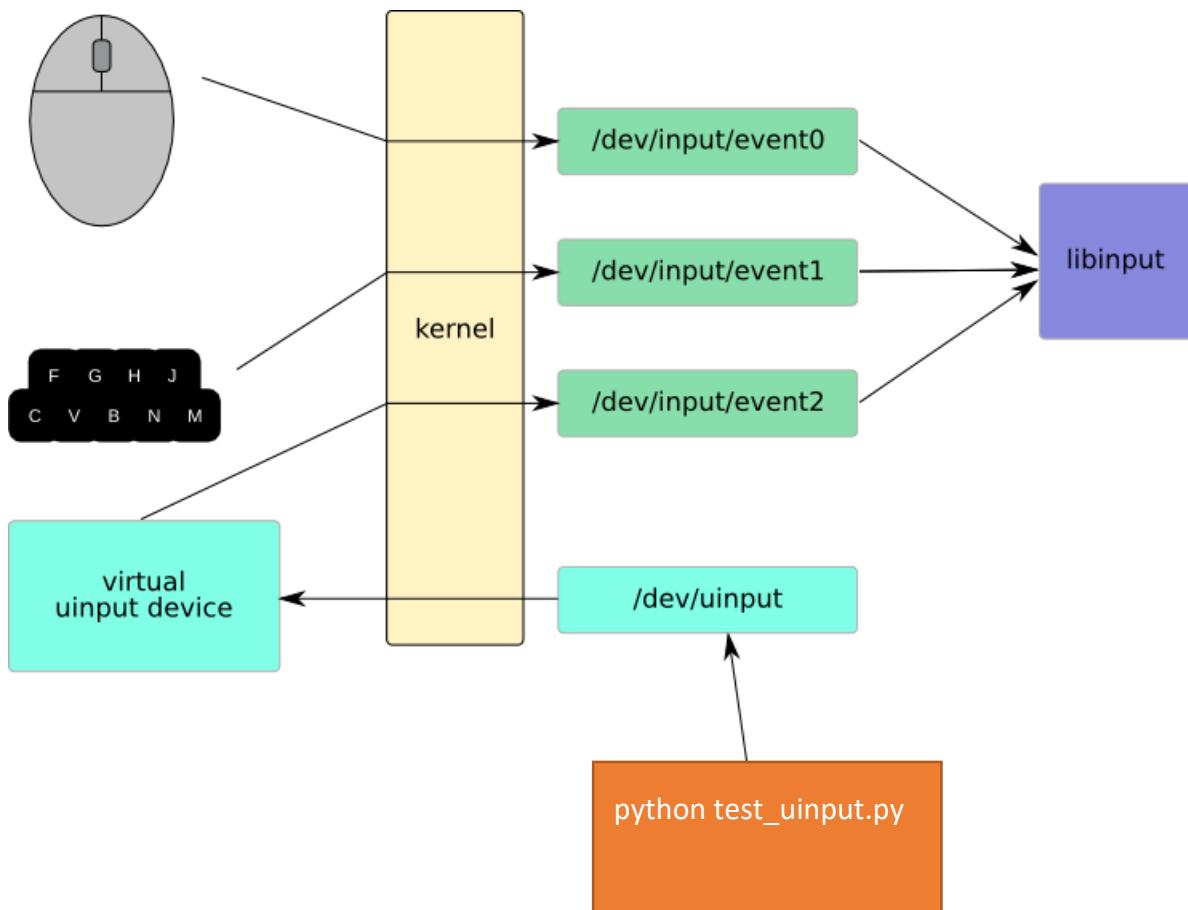
Remarque : si le module n'est pas chargé automatiquement par le noyau au démarrage, on pourra forcer le chargement de ce module avec le fichier /etc/modules

```
sudo nano /etc/modules
```

```
brcmfmac  
g_serial  
uinput
```

Mais que fait le module uinput ?

Uinput permet de simuler des événements claviers, souris ou autres. Un événement est utilisé en général par votre interface graphique Xorg pour gérer les événements venant de la souris, du clavier, ou de tout autre entrée. Le driver qui gère ces entrées physiques s'appelle input et génère dans /dev/input des fichiers associés à chaque matériel. Notre driver uinput permet quant à lui de simuler des événements venant de programmes. Ces événements pourront ensuite déclencher des actions de façon automatique.



Pour notre test, nous allons écrire un programme en python qui envoie toutes les secondes des événements fictifs et nous vérifierons que ces événements sont visibles dans /dev/input/

Mais avant de commencer notre test, nous allons modifier le groupe et les droits du fichier /dev/uinput (qui a du être créé lors du chargement de votre module uinput). Pour cela il faut ajouter un fichier dans /etc/udev/rules.d/ fichier qui sera lu par le deamon udev qui a la charge de monter les fichiers spéciaux dans /dev lors du chargement des modules par le noyau. Par défaut ces fichiers appartiennent à root. Il est donc important de modifier les droits par défaut pour pouvoir accéder à ce fichier sans avoir à lancer la commande sudo.

```
echo KERNEL=="uinput", GROUP="$USER", MODE=="0660" | sudo tee /etc/udev/rules.d/99-$USER.rules
sudo udevadm trigger
```

Vérifier que votre fichier spécial a été modifié et que le groupe est geii.

```
geii@bp11:/etc/udev/rules.d$ ls -l /dev/uinput
crw-rw---- 1 root geii 10, 223 Mar 17 15:11 /dev/uinput
```

Test du module uinput :

Pour tester le module uinput, nous avons besoin de télécharger la bibliothèque python-uinput et d' l'installer sur notre cible.

```
wget http://tjjr.fi/sw/python-uinput/releases/python-uinput-0.11.2.tar.gz
tar xzvf python-uinput-0.11.2.tar.gz
cd python-uinput-0.11.2/
sudo python setup.py install
```

Optionnel : vous pouvez installer le paquet input-utils (pas obligatoire, permet d'avoir la commande lsinput)

```
sudo apt install input-utils
```

Copier le programme test_uinput.py qui envoie toutes les secondes des événements liés au clavier

```
nano test_uinput.py
```

```
import uinput
import time

device = uinput.Device([
    uinput.KEY_E,
    uinput.KEY_H,
    uinput.KEY_L,
    uinput.KEY_O,
])
while(True):
    device.emit_click(uinput.KEY_H)
    device.emit_click(uinput.KEY_E)
    device.emit_click(uinput.KEY_L)
    device.emit_click(uinput.KEY_L)
    device.emit_click(uinput.KEY_O)
    time.sleep(1)
```

Puis lancer ce programme en tâche de fond

```
python test_uinput.py &
```

Vérifier avec la commande lsinput qu'un nouvel évènement a été créé :

```
ls /dev/input
```

```
^Cgeii@bp11:~$ ls /dev/input/
by-path  event0  event1  event2  event3
geii@bp11:~$
```

La commande evtest permet de visualiser les évènements en cours.

```
evtest
```

```
^Cgeii@bp11:~$ evtest
No device specified, trying to scan all of /dev/input/event*
Not running as root, no devices may be available.
Available devices:
/dev/input/event0:      gpio_keys
/dev/input/event1:      sunxi-ir
/dev/input/event2:      lircd-uinput
/dev/input/event3:      python-uinput
Select the device event number [0-3]: 3
```

Vous devriez voir apparaître sur l'interface associée à votre programme (interface virtuelle) les évènements claviers toutes les secondes.

```
Event: time 1552833427.911014, type 1 (EV_KEY), code 24 (KEY_O), value 1
Event: time 1552833427.911014, type 1 (EV_KEY), code 24 (KEY_O), value 0
Event: time 1552833427.911014, ----- SYN_REPORT -----
Event: time 1552833428.912777, type 1 (EV_KEY), code 35 (KEY_H), value 1
Event: time 1552833428.912777, type 1 (EV_KEY), code 35 (KEY_H), value 0
Event: time 1552833428.912777, ----- SYN_REPORT -----
Event: time 1552833428.913040, type 1 (EV_KEY), code 18 (KEY_E), value 1
Event: time 1552833428.913040, type 1 (EV_KEY), code 18 (KEY_E), value 0
Event: time 1552833428.913040, ----- SYN_REPORT -----
Event: time 1552833428.913250, type 1 (EV_KEY), code 38 (KEY_L), value 1
Event: time 1552833428.913250, type 1 (EV_KEY), code 38 (KEY_L), value 0
Event: time 1552833428.913250, ----- SYN_REPORT -----
Event: time 1552833428.913450, type 1 (EV_KEY), code 38 (KEY_L), value 1
Event: time 1552833428.913450, type 1 (EV_KEY), code 38 (KEY_L), value 0
Event: time 1552833428.913450, ----- SYN_REPORT -----
Event: time 1552833428.913634, type 1 (EV_KEY), code 24 (KEY_O), value 1
Event: time 1552833428.913634, type 1 (EV_KEY), code 24 (KEY_O), value 0
Event: time 1552833428.913634, ----- SYN_REPORT -----
```

Ressources

https://wiki.archlinux.org/index.php/Compile_kernel_module

11.5 Uboot

Uboot possède ses propres variables d'environnement (variables par défaut au moment de la compilation). Au lancement uboot vient chercher le fichier boot.scr dans lequel se trouve le script et les variables d'environnement.

Il est possible de visualiser ces variables d'environnement sous linux :

```
sudo fw_printenv
```

Ces variables sont utilisées dans le script de démarrage boot.scr

Ainsi bootdelay=2 (2s avant de lancer le script)

boot_scripts=boot.scr.uimg boot.scr (script que va chercher uboot au démarrage)

Dans boot.scr ensuite si le fichier armbianEnv.txt existe celui-ci est chargé (mise à jour des variables utilisées lors du chargement du noyau)

```
if test -e ${devtype} ${devnum} ${prefix}armbianEnv.txt; then
    load ${devtype} ${devnum} ${load_addr} ${prefix}armbianEnv.txt
```

attention boot.scr a été créé a partir de boot.cmd

```
mkimage -C none -A arm -T script -d boot.cmd boot.scr
```

Ressources :

<https://www.vermasachin.com/posts/3-u-boot-environment-variables/>

11.6 uInitrd (ramdisk)

Pour visualiser le système de fichier se trouvant dans /boot

```
dd if=/boot/uInitrd of=~/initrd.img.gz bs=64 skip=1
cd
gunzip initrd.img.gz
mkdir init
cd init
cpio -i < ../initrd.img
```

11.7 Boot du système

Quels sont les process lancés et par qui ?

```
ps axjf
```

<https://www.digitalocean.com/community/tutorials/how-to-configure-a-linux-service-to-start-automatically-after-a-crash-or-reboot-part-2-reference#init-and-pid-1>

<https://www.digitalocean.com/community/tutorials/understanding-systemd-units-and-unit-files>

<https://www.lions-wing.net/maker/raspberry-1/boot.html>



