



Découverte de l'environnement LINUX embarqué





LINUX embarqué

Le cours de micro-informatique nous permettra de découvrir l'environnement informatique embarqué que sont les calculateurs sur les bords. L'approche adoptée nous permettra de développer sur des structures très proches des calculateurs que sont les nano-ordinateurs Raspberry PI 3 ou autres. Le système d'exploitation utilisé sera LINUX embarqué RASPIAN qui peut s'adapter à des contraintes temps réel mou. Les systèmes d'exploitation temps réel (RTOS) dures (type XENOMAI) ne seront pas abordés dans ce polycopié. Cette première partie va permettre de nous familiariser avec cet environnement de travail.

1) Présentation

Linux est un système d'exploitation, au même titre que Windows ou encore Mac OS (pour ceux qui ont un Mac). Linux est gratuit et vous possédez la maitrise totale du système d'exploitation car le code source est libre. Les logiciels sont gratuits et souvent mise à jour.
En 1984 **Richard Stallman** créa le projet GNU. Il voulait créer un nouveau système d'exploitation fonctionnant comme Unix (les commandes restant les mêmes). La collaboration de Richard Stallman et **Linus Torvalds** pour un projet commun (Le projet GNU (programmes libres) et Linux (noyau d'OS)) ont vu naitre **GNU/Linux**.

2) Les distributions

Afin de simplifier la vie des utilisateurs et leur permettre de faire un choix, différentes distributions de Linux ont été créées.
En fait, une distribution est en quelque sorte l'emballage de Linux. Le coeur ou le noyau de l'OS , lui, reste le même sur toutes les distributions. Il existe beaucoup de distributions Linux:

- **Mandriva** : éditée par une entreprise française, elle se veut simple d'utilisation ;
- **Red Hat** : éditée par une entreprise américaine, cette distribution est célèbre et très répandue, notamment sur les serveurs ;
- **SuSE** : éditée par l'entreprise Novell ;
- **Debian** : la seule distribution qui soit gérée par des développeurs indépendants. Debian a tellement de succès que de nombreuses distributions sont basées sur Debian la plus connu reste ubuntu.

3) Les modes consoles et le mode graphique

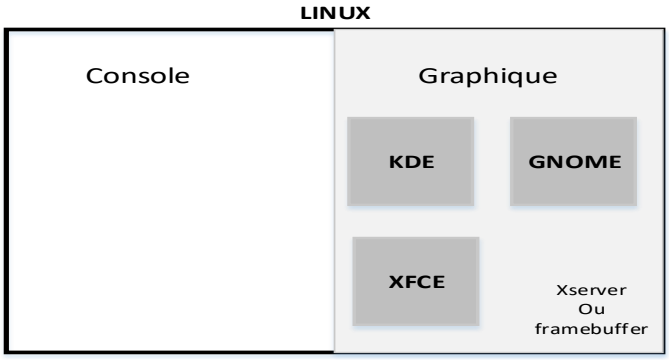
Il existe deux façons d'utiliser Linux :

- ✓ en mode console (équivalent à DOS) ;
- ✓ en mode graphique (équivalent à Windows).

En console, le fond est généralement noir (mais ce n'est pas une règle), la souris n'est pas utilisé. Tout se fait au clavier ce qui peut être déroutant pour un non initié
Le mode graphique semble beaucoup plus accueillant car très proche de celui de Windows. Tous les modes graphiques sont basés sur un programme **appelé X**. X est en fait la brique de base du mode graphique sous Linux. Il existe aussi un autre mode graphique **le frame buffer.**
Par-dessus X vient se greffer un programme appelé le gestionnaire de bureau. Le rôle du gestionnaire de bureau est de gérer les fenêtres, leur apparence, leurs options, on peut citer :

- ✓ Gnome ;
- ✓ KDE;
- ✓ XFCE.

Gnome et KDE sont de gros gestionnaires de bureau, très riches en fonctionnalités.
XFCE est une alternative plus légère qui peut fonctionner sur des ordinateurs plus anciens.



4) Les plateformes matérielles embarquées LINUX

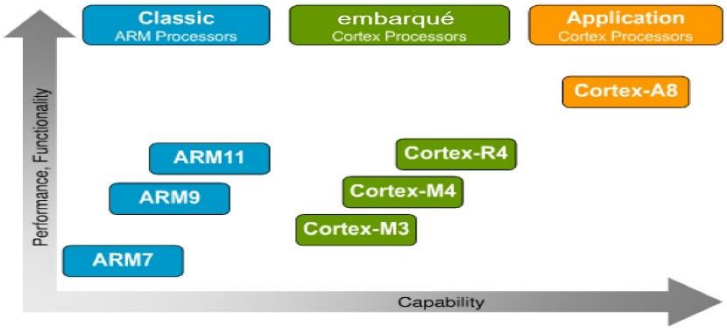
Il existe beaucoup de nano-ordinateurs qui peuvent accueillir une distribution LINUX. En TP on utilisera la plateforme :

- RAspberry PI 3

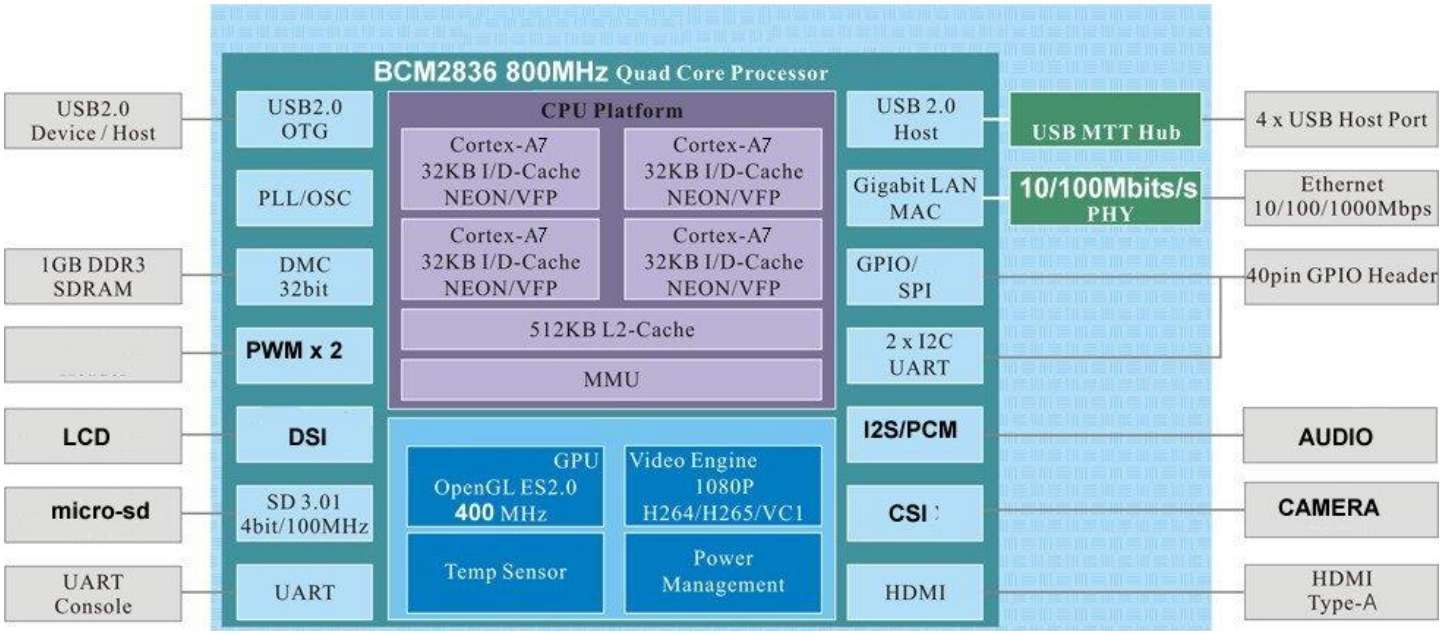
La RAspberry PI 3 est une carte CPU autonome à faible cout pour des systèmes embarqués. Elle fait partie d'un ensemble de nano-ordinateurs qui ont su évoluer au fil des années.

	Raspberry Pi 3 Model B	Raspberry Pi Zero	Raspberry Pi 2 Model B	Raspberry Pi Model B+
Introduction Date	2/29/2016	11/25/2015	2/2/2015	7/14/2014
SoC	BCM2837	BCM2835	BCM2836	BCM2835
CPU	Quad Cortex A53 @ 1.2GHz	ARM11 @ 1GHz	Quad Cortex A7 @ 900MHz	ARM11 @ 700MHz
Instruction set	ARMv8-A	ARMv6	ARMv7-A	ARMv6
GPU	400MHz VideoCore IV	250MHz VideoCore IV	250MHz VideoCore IV	250MHz VideoCore IV
RAM	1GB SDRAM	512 MB SDRAM	1GB SDRAM	512MB SDRAM
Storage	micro-SD	micro-SD	micro-SD	micro-SD
Ethernet	10/100	none	10/100	10/100
Wireless	802.11n / Bluetooth 4.0	none	none	none
Video Output	HDMI / Composite	HDMI / Composite	HDMI / Composite	HDMI / Composite
Audio Output	HDMI / Headphone	HDMI	HDMI / Headphone	HDMI / Headphone
GPIO	40	40	40	40

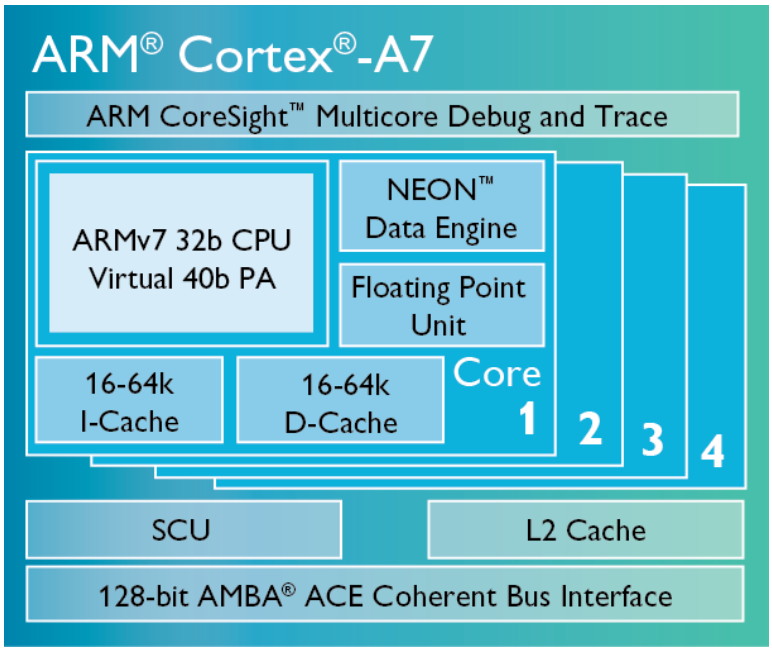
Le processeur utilisé sur la PI3 est compatible avec les processeurs ARM cortex A-7 A-8 (des processeurs d'application). La société ARM propose une gamme importante de microprocesseur répondant à une multitude d'application.



Le processeur BCM2836 sur la carte Raspberry PI 3 fait partie de la famille de systèmes sur puces **(SOC)** et basée sur une architecture ARM à faible consommation L'architecture du BCM2837 est donnée ci-dessous :



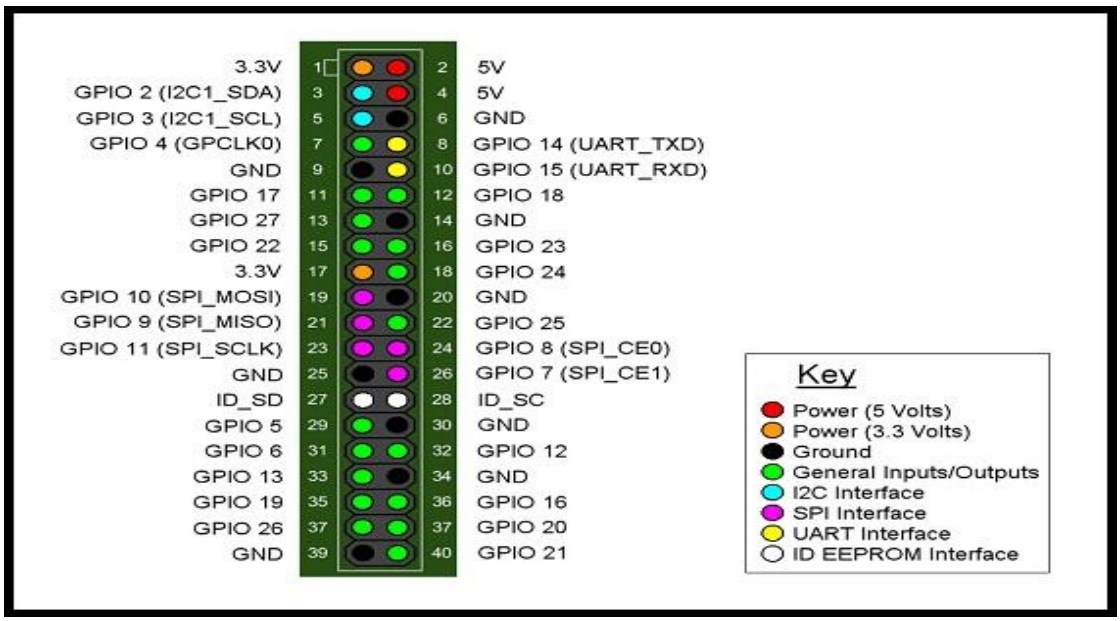
Le SOC **BCM2837** est doté de quatre processeurs ARM CORTEX A7 , avec une unité de données Neon est une unité de calcul à virgule flottante.



Tous ces éléments sont intégrés sur la même puce et conviennent pour une multitude d'application dont du traitement vidéo HD. Ces applications peuvent être :

- Systèmes de navigation
- Lecteur multimédia
- Systèmes de surveillance médicale
- Vision industriel
- Communication portable
- Tablette graphique

Attention l'ethernet, l'USB, le WIFI, le HDMI ne sont pas intégré directement dans le processeur mais le processeur est éventuellement capable de piloter ce type de contrôleur de communication qui sont externes à la puce. Son connecteur d'extension de 40 broches a le format suivant :

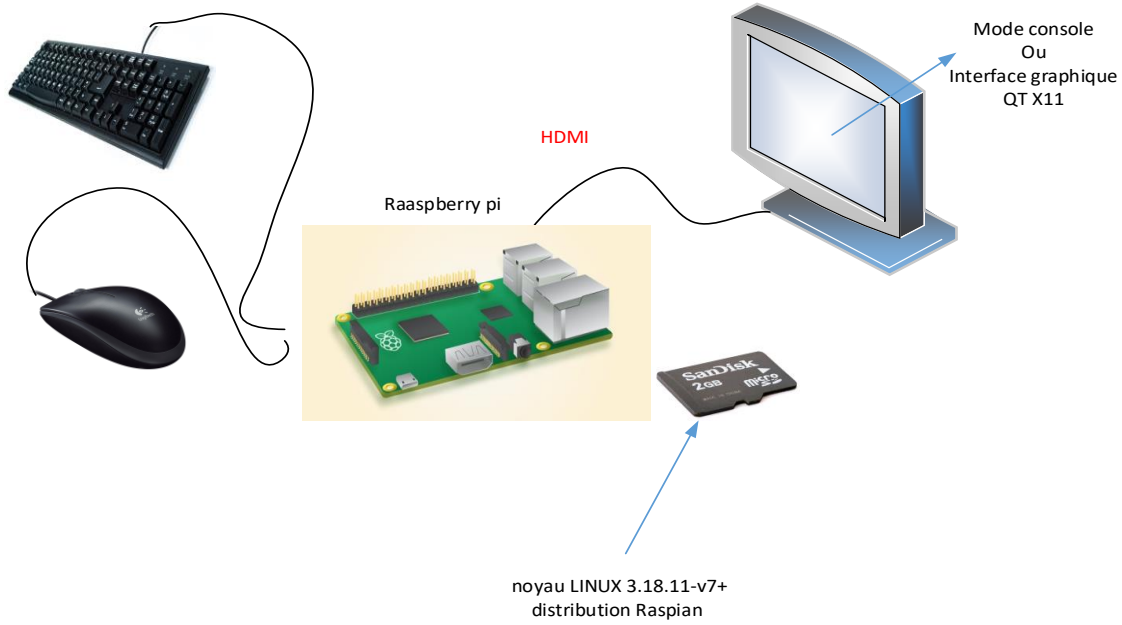


Cette carte peut accueillir un système d'exploitation type Linux embarqué Xenomai Android …

La partie suivante concerne la mise en place d'un noyau LINUX et son système de fichier ROOT afin de bénéficier d'un environnement LINUX embarqué sur la carte RASPBERRY PI3 . La distribution LINUX Raspian sera installée sur la SDCARD de la PI 3 (noyau LINUX et système de fichier ROOT).

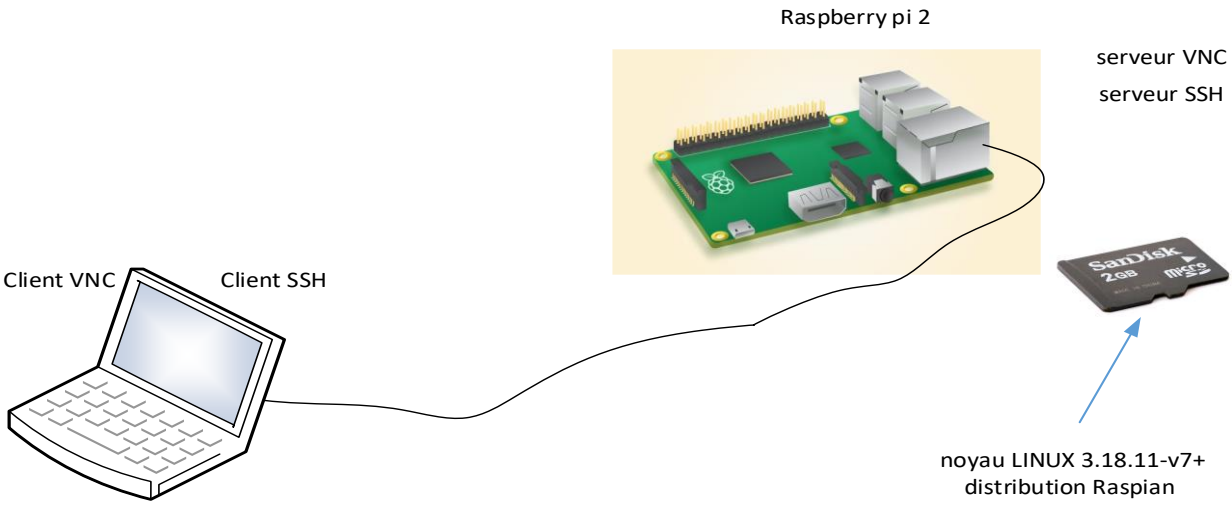
5) L'accès au système d'exploitation

On peut comme tout nano-ordinateur accédé au système d'exploitation en branchant une souris un clavier et un écran et en démarrant le système. L'OS (operating system) est implanté sur de la mémoire flash (SDCARD)



On peut aussi comme beaucoup d'environnement embarqué (calculateur) prendre la main à distance sur la carte grâce à des services réseaux préinstallés :

- En mode console client et serveur SSH
- En mode graphique (s'il existe) client et serveur VNC.

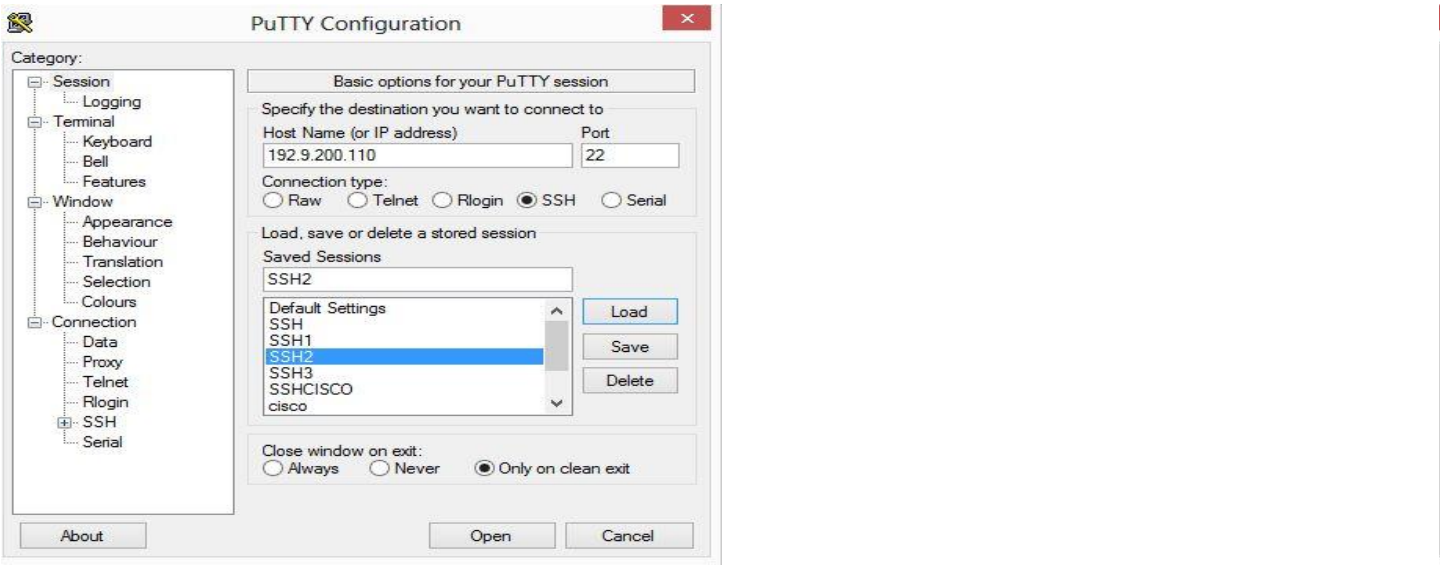


Ouvrir Putty sur l'environnement windows :



Configurez putty avec la configuration de la liaison réseau Ethernet IP Raspberry pi 3 distribution LINUX

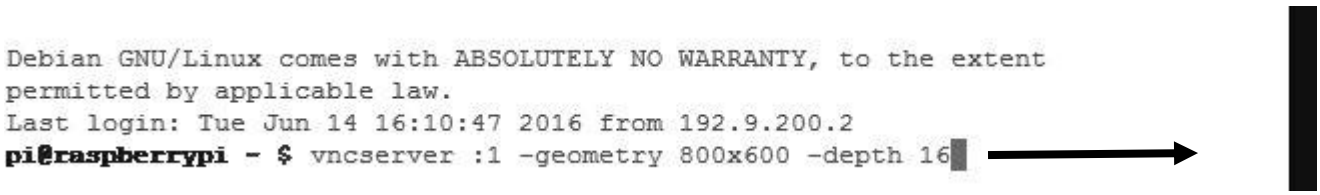
Réseau **Série port**



Tapez pour la raspberry login pi password raspberry

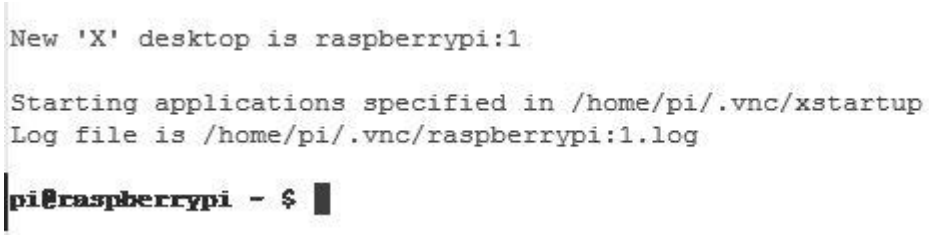


Pour accéder au gestionnaire de bureau lancez le programme serveur VNC x11vnc sur la Raspberry



Lancement de X11VNC

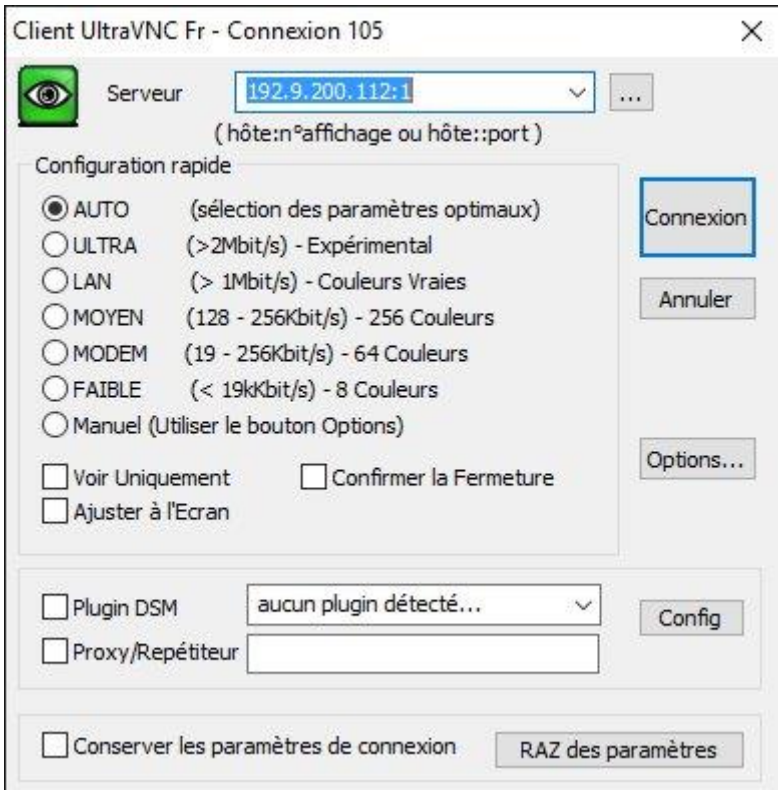
Le menu suivant apparait :



Puis sur le bureau de Windows le client VNC appelé client ultra vnc :



Tapez l'adresse IP de la carte Raspberry puis cliquez sur connexion :

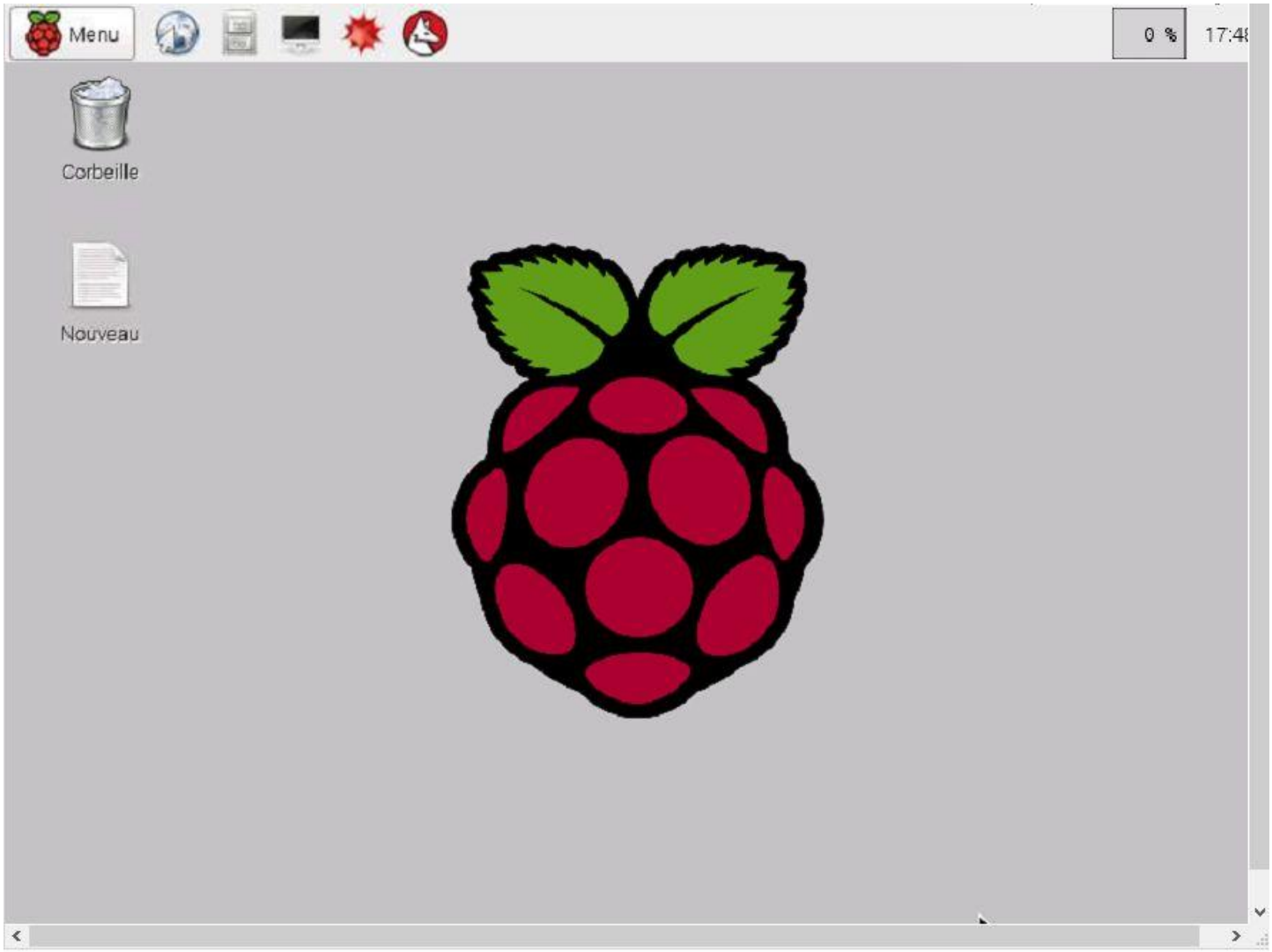


Le login apparait :



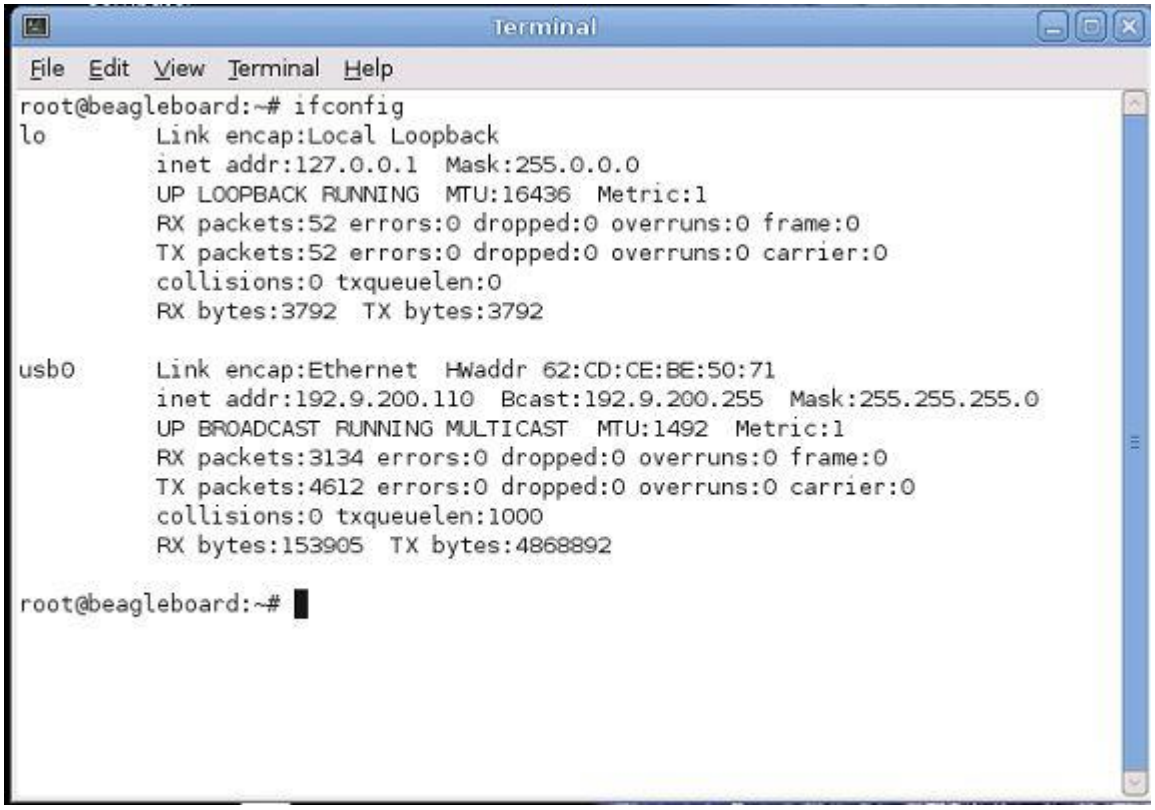
Tapez comme password raspberry

Le gestionnaire de bureau apparait enfin :


Menu
0 %
Corbeille
Nouveau

6) La configuration réseau sous LINUX

Pour connaitre la configuration réseau sous LINUX ouvrir une fenêtre terminale pour la Raspberry pi 3 (menu -→ accessoire -→ Lxterminal)
Tapez la commande ifconfig ou sudo ifconfig pour la raspberry



Sous Linux la configuration réseau se trouve dans un fichier nommé interface qui se trouve dans le répertoire /etc/interfaces/network.
Ouvrant ce fichier :

```
iface atml0 inet dhcp

# Wired or wireless interfaces
auto eth1
#iface eth0 inet dhcp
iface eth1 inet dhcp

# Ethernet/RNDIS gadget (g_ether) or LAN9514
auto usb0
iface usb0 inet static
address 192.9.200.110
netmask 255.255.255.0
gateway 192.9.200.1

# Bluetooth networking
#iface bnep0 inet dhcp
#auto usb0
#iface usb0 inet static
#address 192.168.2.11
#netmask 255.255.255.0
#gateway 192.168.2.1
```

Notre interface ethernet sur la Raspberry a comme nom eth0 et non usb0 :

Deux configurations sont possibles :

- Adressage statique :

```
auto eth0
iface eth0 inet static
address 192.9.200.1
netmask 255.255.255.0
gateway 192.168.1.1
```

- Adressage dynamique par serveur DHCP

```
auto eth0
iface eth0 inet dhcp
```

Les modifications sont à réaliser dans le fichier network pour l'adressage statique ou dynamique.

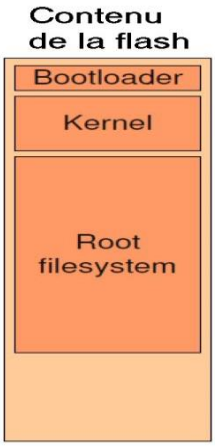
Chargement de la distribution Linux

L'objectif de ce tutorial est de comprendre les mécanismes de chargement d'une distribution LINUX Raspian sous une carte Raspberry pi 3.
Cette distribution LINUX est compatible avec les architectures de processeur ARM7A dont fait partie le SOC de la carte Raspberry pi 3 le BCM2837.

1) Présentation

Une mémoire Flash du type SDCARD de dimension égale ou supérieure à 16Go est nécessaire:
Elle dispose :

- D'une partition principale FAT32 d'une centaine de mégaoctet contenant un bootloader et une copie de l'image du noyau
- D'une partition étendue LINUX EXT3 ou EXT4 d'une capacité correspondant au reste de l'espace de la carte SDCARD. Cette partition contient le système de fichier ROOT de LINUX



2) Description du contenu de la flash

Rôle du Bootloader

Démarré par le matériel directement, responsable des initialisations de base, du chargement de l'exécution du noyau et de lui spécifier son root file system installé généralement en flash.

Rôle du Noyau Linux (Kernel)

- ✓ Responsable de la gestion des différents services pour les applications (processus, mémoire virtuelle, pile réseau, pilote de périphérique, ...)

Dans un système Linux, plusieurs systèmes de fichiers peuvent être montés et créer une hiérarchie globale de fichiers et répertoires_ Un système de fichier particulier, le système de fichier root est monté en "/ ".

- ✓ Construire ce système de fichier est l'une des tâches essentielles de l'intégration de Linux sur la plate-forme embarquée.
- ✓ En général le noyau est séparé du système de fichier root

Sur les architectures embarquées, le boot de bas niveau est très dépendant du CPU et de la plate-forme.

- ✓ Certaines cartes ont une Flash à partir de laquelle le CPU boote après un reset

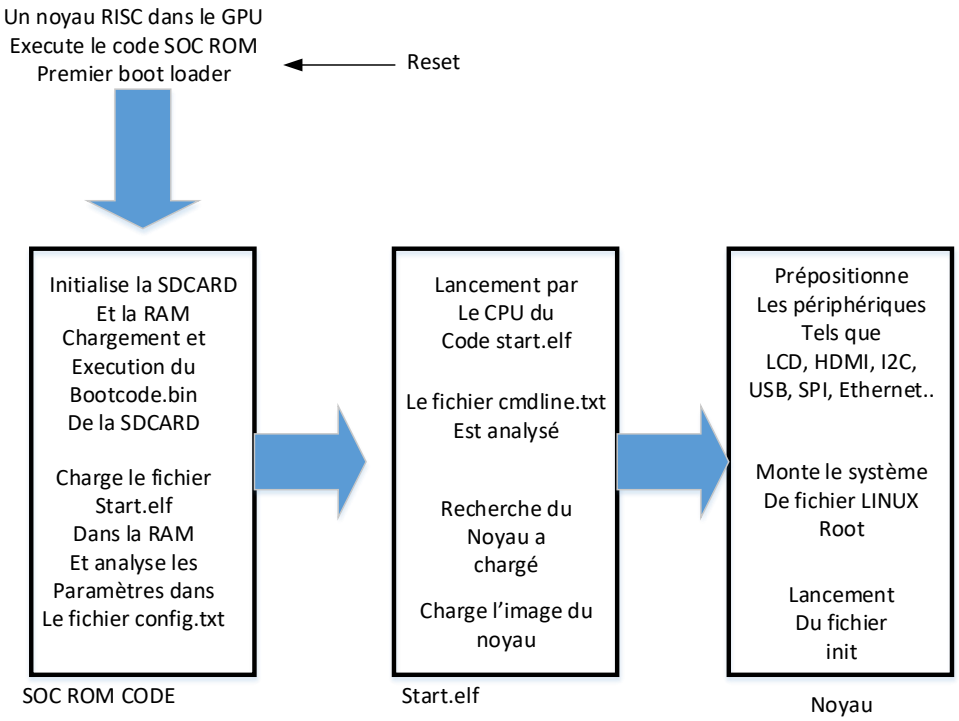
- ✓ Certains CPU ont un morceau de code intégré dans une ROM qui charge automatiquement une petite portion de Flash dans la RAM

3) La séquence de boot traditionnelle

3.1) Chargement du Bootloader

- ✓ Exécute par le CPU à une adresse fixe en ROM/Flash
- ✓ Initialise le support du matériel ou se trouve le noyau (Flash, réseau, SD-card...)
- ✓ Charge l'image du noyau dans la RAM
- ✓ Lance l'exécution du noyau
- ✓ Le système de boot actuel contient donc le mécanisme initramfs (ex-initrd)

Un petit système de fichier contenu dans le noyau lui-même.
Il peut détecter le matériel, charger les modules du noyau nécessaires
Démarré par le matériel directement, responsable des initialisations de base, du chargement et de l'exécution du noyau



Le fichier Bootcode.bin est le premier des plus petits fichiers qui chargera le fichier binaire Start.elf.
Le chargeur de boot start.elf sera utilisé pour copier le noyau et le contenu des applicatifs ROOT dans la mémoire.

3.2) Noyau (fichier bin ou img)

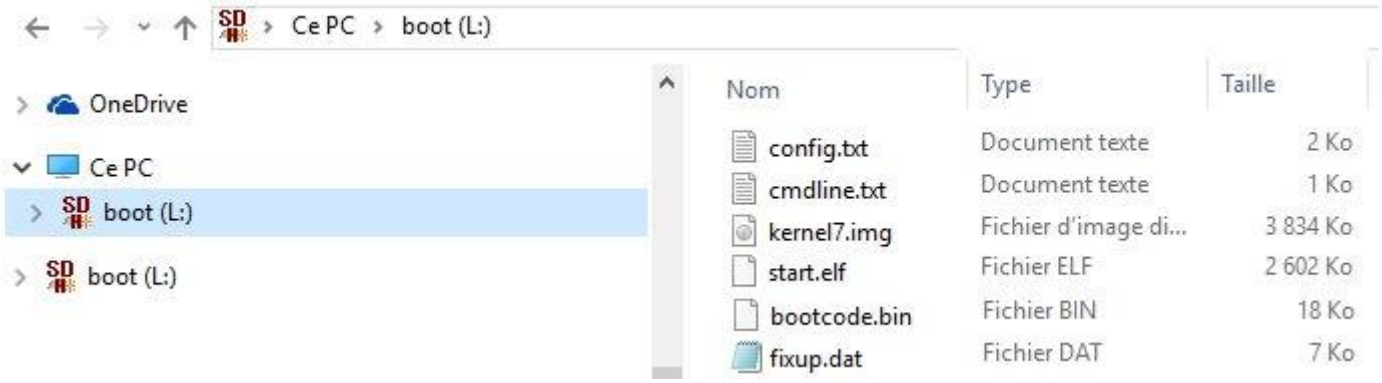
- ✓ En général le noyau est séparé du système de fichier root
- ✓ Il se décompresse lui-même
- ✓ Sur les Linux embarqués, le système de fichier root contient toutes les librairies, applications, et données du système.
- ✓ Il est responsable de la gestion des différents services pour les applications (processus, mémoire virtuelle, pile réseau, pilote de périphérique, ...)
- ✓ Il Initialise le noyau et les pilotes compilés statiquement utilisés pour accéder au système de fichier root.

La plupart des pilotes de périphériques sont compilés statiquement dans le noyau.

Cette hypothèse est valide dans le monde embarqué ou le noyau est paramétré en fonction du matériel, elle l'est beaucoup moins pour les PC de bureau

- ✓ Il monte le système de fichier root
- ✓ Il exécute le premier programme utilisateur (spécifié par le parametre init du noyau).

Le contenu de la partition boot (peut varier suivant les versions de LINUX) :

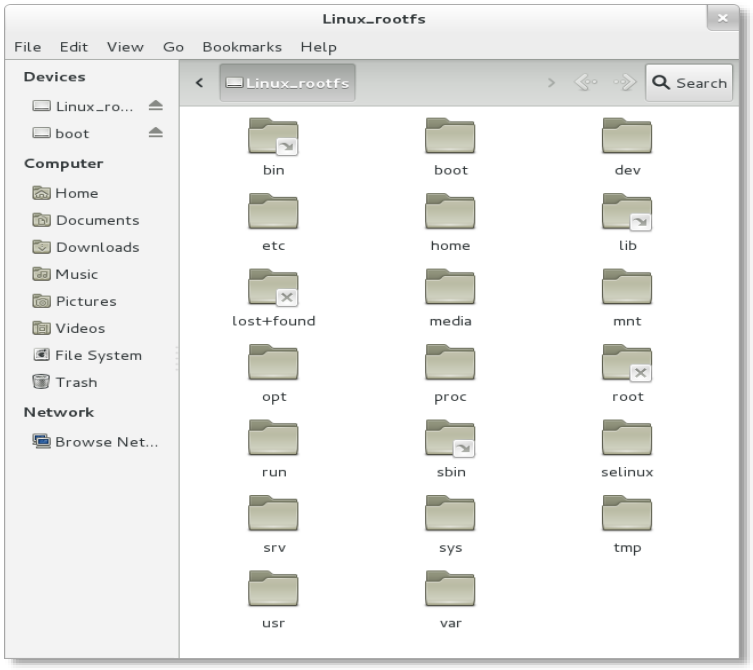


Nom	Rôle	format
Bootcode.bin	bootloader	Binaire
Start.elf	Code ARM exécuté par le CPU	Binaire
Fixup.dat	Configure la SDRAM GPU/CPU	Binaire
Config.txt	Paramètre de configuration	texte
Cmdline.txt	Paramètres en ligne de commande pour le noyau	texte
Kernel7.img	Noyau	Binaire

3.3) Premier programme utilisateur

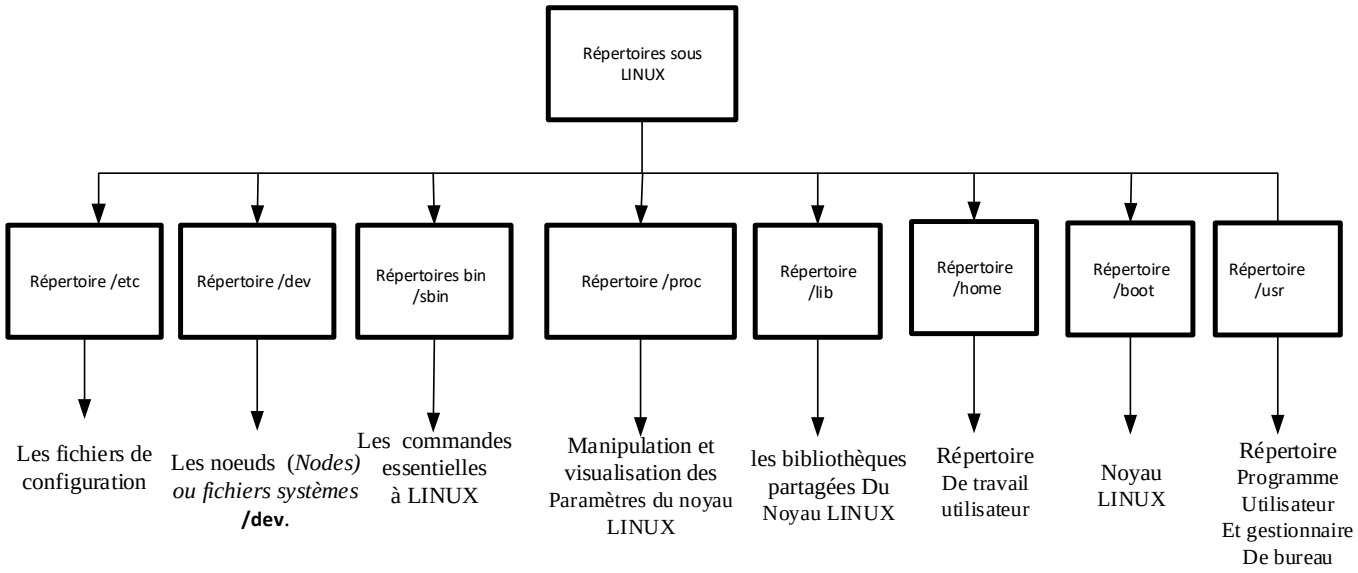
- ✓ Il Configure l'espace utilisateur et démarre les services systèmes

Le contenu du système de fichier Root



Sous la distribution Raspian, start.elf va charger le noyau dans la partition ROOT, répertoire boot et fichier kernel7.img.
(Attention au SDCARD de mauvaise qualitée, source d'erreur d'entrée sortie lors des accès du noyau sur la partition ROOT)

Les répertoires du système de fichier ont un rôle bien particulier :



```
pi@raspberrypi ~ $ cat /proc/interrupts
           CPU0       CPU1       CPU2       CPU3
 16:          0          0          0          0   ARMCTRL  bcm2708_fb dma
 24:        251          0          0          0   ARMCTRL  DMA IRQ
 25:       3129          0          0          0   ARMCTRL  DMA IRQ
 32:     875071          0          0          0   ARMCTRL  dwc_otg, dwc_otg_pcd
, dwc_otg_hcd:usb1
 52:          0          0          0          0   ARMCTRL  BCM2708 GPIO catchal
l handler
 65:         55          0          0          0   ARMCTRL  ARM Mailbox IRQ
 66:          9          0          0          0   ARMCTRL  VCHIQ doorbell
 75:          1          0          0          0   ARMCTRL
 79:          4          0          0          0   ARMCTRL  bcm2708_i2c.1
 80:          0          0          0          0   ARMCTRL  bcm2708_spi.0
 84:      13514          0          0          0   ARMCTRL  mmc0
 99:      62635      30422       8750       6341   ARMCTRL  arch_timer
FIQ:              usb_fiq
IPI0:          0          0          0          0  CPU wakeup interrupts
IPI1:          0          0          0          0  Timer broadcast interrupts
IPI2:      10116      22496      25658      26519  Rescheduling interrupts
IPI3:          6          8          7          7  Function call interrupts
IPI4:          1          4          1          1  Single function call interrup
```

```
pi@raspberrypi ~ $ cat /proc/cpuinfo
processor       : 0
model name      : ARMv7 Processor rev 5 (v7l)
BogoMIPS        : 38.40
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xc07
CPU revision    : 5
```

Le noyau chargé et exécuté par le bootloader qui initialise un grand nombre d'éléments matériels et logiciels. II prend ainsi en charge tout le matériel directement intégré à la configuration comme les contrôleurs IDE, les ports série, etc. Ceci fait, il montera le système de fichier spécifié via le paramètre root et tentera d'exécuter le premier **programme** utilisateur du système init, le processus numéro 1.
Le Passage de l'espace noyau à l'espace utilisateur se fait donc par le lancement **du premier processus init** (/sbin/init ou le programme pointé par le paramètre init =).

```
root@beagleboard:~# ps -aux
Warning: bad ps syntax, perhaps a bogus '-'? See http://procps.sf.net/faq.html
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1   1560   564 ?        Ss   04:11   0:01 init [5]
root         2  0.0  0.0      0     0 ?        S    04:11   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    04:11   0:00 [ksoftirqd/0]
root         4  0.0  0.0      0     0 ?        S    04:11   0:00 [watchdog/0]
root         5  0.0  0.0      0     0 ?        S    04:11   0:00 [events/0]
root         6  0.0  0.0      0     0 ?        S    04:11   0:00 [khelper]
root         7  0.0  0.0      0     0 ?        S    04:11   0:00 [async/mgr]
```

Le processus de démarrage d'un système Linux est très simple, du moins durant les premières secondes.

Le fichier init cherche sa configuration dans /etc/inittab.
Nous commençons donc par éditer le fichier de configuration principal d'init (/etc/inittab) :

```
id:5:initdefault: #niveau par défaut

# Boot-time system configuration/initialization script.
# This is run first except when booting in emergency (-b) mode.
si::sysinit:/etc/init.d/rcS

# What to do in single-user mode.
~~:S:wait:/sbin/sulogin

# /etc/init.d executes the S and K scripts upon change
# of runlevel.
#
# Runlevel 0 est lié à l'événement halt.
# Runlevel 1 est lié à un niveau simple utilisateur
# Runlevels 2-5 sont des niveaux multi-utilisateurs
# Runlevel 6 est lié à l'événement halt.

l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
```

Dans l'ordre nous :

- spécifions le niveau d'exécution par défaut 5 ici ;
- précisons le script systématiquement utilisé au moment du **boot,** le script d'initialisation du système rcS
- pour chaque niveau d'exécution, précisons un argument différent pour un script unique.

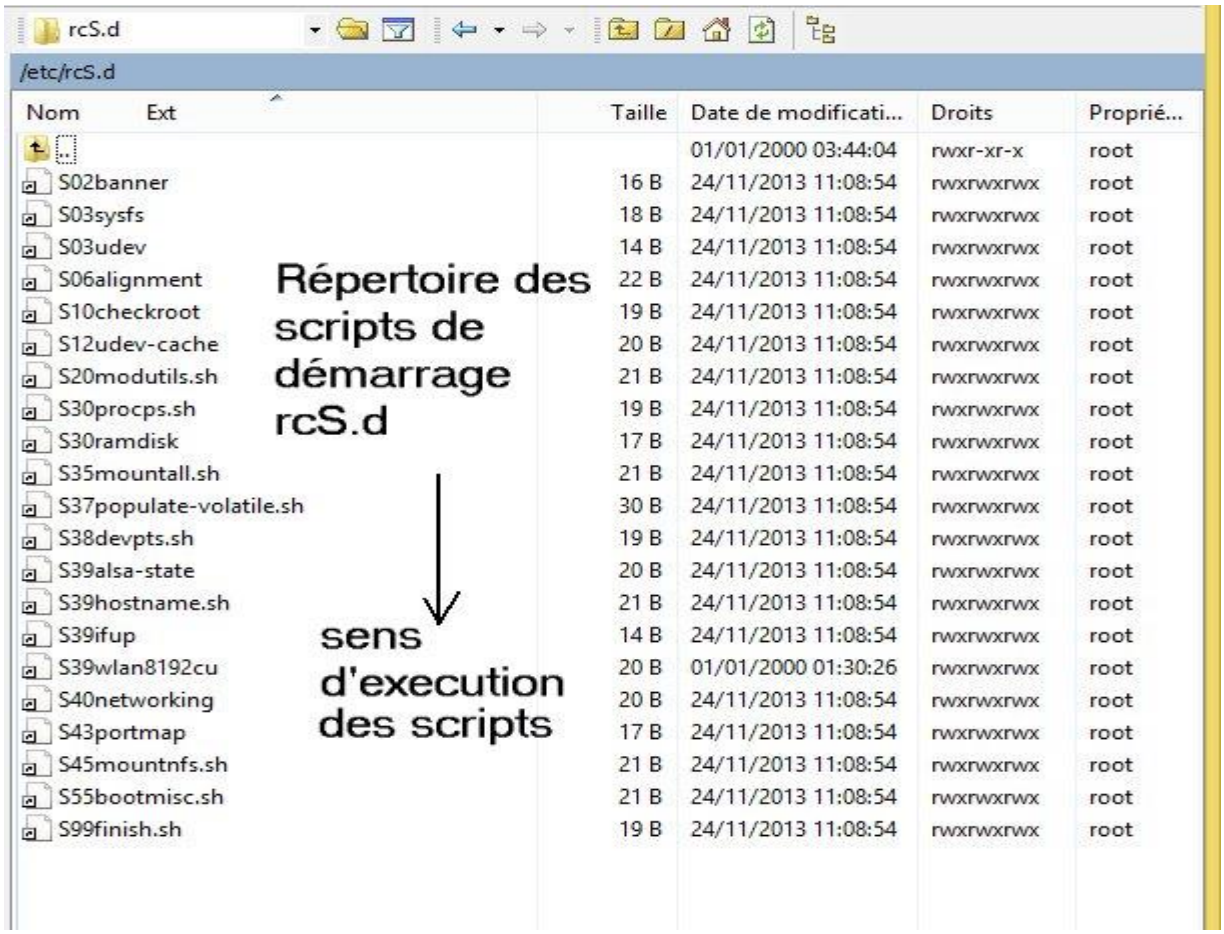
Nous retrouvons le fameux script rc avec l'argument S en lieu et place d'une valeur numérique. Tout le fonctionnement de la configuration d' init repose donc sur le script rc
A **ce stade,** init utilisera son inittab **pour exécuter le script** rc **avec l'argument** S **puis le même script avec l'argument** 2. **Ce script s'occupera de lancer d'autres scripts en cas de changement de niveau d'exécution.** Pour **déterminer quel script lancer et avec quel argument,** il **utilisera l'architecture suivante :**

- Le répertoire /etc/init.d contient tous **les scripts.**
- **Des répertoires** correspondant **aux niveaux d'exécution** (/etc/rc0.d /etc/rcl.d... /etc/rcS.d) contiennent **des liens symboliques vers les scripts correspondants dans** /etc/init.d.

rc0.d	01/01/1970 01:31:52	rwxr-xr-x	root
rc1.d	01/01/1970 01:31:52	rwxr-xr-x	root
rc2.d	01/01/1970 01:31:52	rwxr-xr-x	root
rc3.d	01/01/1970 01:31:52	rwxr-xr-x	root
rc4.d	01/01/1970 01:31:52	rwxr-xr-x	root
rc5.d	01/01/1970 01:31:52	rwxr-xr-x	root
rc6.d	01/01/1970 01:31:52	rwxr-xr-x	root
rcS.d	01/01/2000 01:10:28	rwxr-xr-x	root

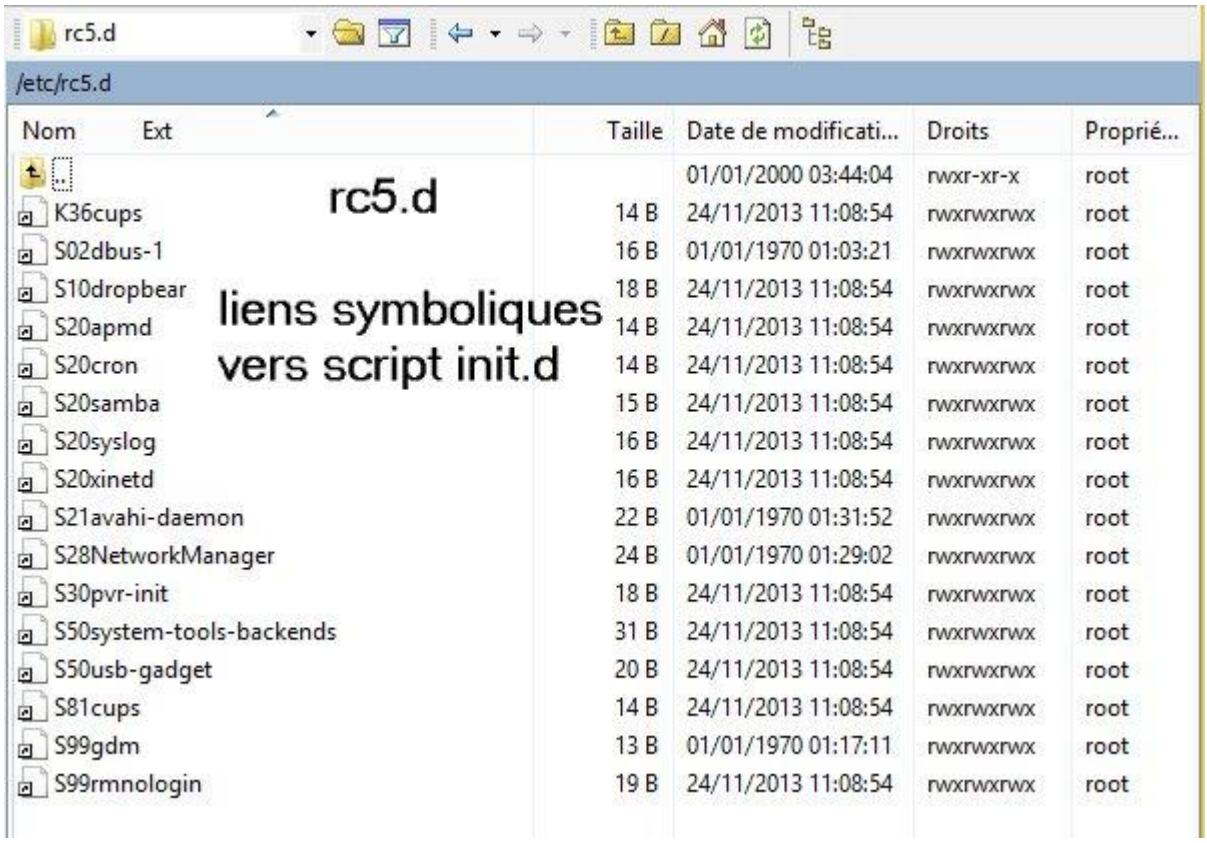
- **Le nommage des liens symboliques détermine ce que le** script rc doit en faire. Le premier caractère du nom définit l'action. Un **S demandera le lancement du** script dans /etc/init.d avec start **en argument. Un** K **fera de même avec** stop en argument. Le numéro spécifié entre la première lettre du nom et le nom du script détermine l'ordre d'exécution. Nous pourrons avoir **par exemple,** /etc/rc.5/S99gdm correspondant à /etc/init.d/gdm **start tel que lancé par** rc.

Les scripts de démarrage ou d'initialisation du système sont donnés ci-dessous.



Nom Ext	Taille	Date de modificati...	Droits	Proprié...
		01/01/2000 03:44:04	rwxr-xr-x	root
S02banner	16 B	24/11/2013 11:08:54	rwxrwxrwx	root
S03sysfs	18 B	24/11/2013 11:08:54	rwxrwxrwx	root
S03udev	14 B	24/11/2013 11:08:54	rwxrwxrwx	root
S06alignment	22 B	24/11/2013 11:08:54	rwxrwxrwx	root
S10checkroot	19 B	24/11/2013 11:08:54	rwxrwxrwx	root
S12udev-cache	20 B	24/11/2013 11:08:54	rwxrwxrwx	root
S20modutils.sh	21 B	24/11/2013 11:08:54	rwxrwxrwx	root
S30procps.sh	19 B	24/11/2013 11:08:54	rwxrwxrwx	root
S30ramdisk	17 B	24/11/2013 11:08:54	rwxrwxrwx	root
S35mountall.sh	21 B	24/11/2013 11:08:54	rwxrwxrwx	root
S37populate-volatile.sh	30 B	24/11/2013 11:08:54	rwxrwxrwx	root
S38devpts.sh	19 B	24/11/2013 11:08:54	rwxrwxrwx	root
S39alsa-state	20 B	24/11/2013 11:08:54	rwxrwxrwx	root
S39hostname.sh	21 B	24/11/2013 11:08:54	rwxrwxrwx	root
S39ifup	14 B	24/11/2013 11:08:54	rwxrwxrwx	root
S39wlan8192cu	20 B	01/01/2000 01:30:26	rwxrwxrwx	root
S40networking	20 B	24/11/2013 11:08:54	rwxrwxrwx	root
S43portmap	17 B	24/11/2013 11:08:54	rwxrwxrwx	root
S45mountnfs.sh	21 B	24/11/2013 11:08:54	rwxrwxrwx	root
S55bootmisc.sh	21 B	24/11/2013 11:08:54	rwxrwxrwx	root
S99finish.sh	19 B	24/11/2013 11:08:54	rwxrwxrwx	root

Les scripts du niveau 5 (niveau par défaut) sont lancés par la suite. Le niveau 5 est Multi-utilisateur, avec réseau et environnement graphique X Window.
Les services sont analysés à chaque niveau d'exécution. Lors du passage d'un niveau à un autre, et quel que soit l'ordre (du 2 au 5, du 5 au 3, etc.) le script rc compare les services qui doivent être arrêtés ou démarrés entre l'ancien et le nouveau niveau. Si un service est commun aux deux niveaux, il est maintenu. Si un nouveau service doit être lancé dans le nouveau niveau, il le lance. Si un service doit être arrêté car il est absent du nouveau niveau, il l'arrête.



Nom	Ext	Taille	Date de modificati...	Droits	Proprié...
..			01/01/2000 03:44:04	rwxr-xr-x	root
K36cups		14 B	24/11/2013 11:08:54	rwxrwxrwx	root
S02dbus-1		16 B	01/01/1970 01:03:21	rwxrwxrwx	root
S10dropbear		18 B	24/11/2013 11:08:54	rwxrwxrwx	root
S20apmd		14 B	24/11/2013 11:08:54	rwxrwxrwx	root
S20cron		14 B	24/11/2013 11:08:54	rwxrwxrwx	root
S20samba		15 B	24/11/2013 11:08:54	rwxrwxrwx	root
S20syslog		16 B	24/11/2013 11:08:54	rwxrwxrwx	root
S20xinetd		16 B	24/11/2013 11:08:54	rwxrwxrwx	root
S21avahi-daemon		22 B	01/01/1970 01:31:52	rwxrwxrwx	root
S28NetworkManager		24 B	01/01/1970 01:29:02	rwxrwxrwx	root
S30pvr-init		18 B	24/11/2013 11:08:54	rwxrwxrwx	root
S50system-tools-backends		31 B	24/11/2013 11:08:54	rwxrwxrwx	root
S50usb-gadget		20 B	24/11/2013 11:08:54	rwxrwxrwx	root
S81cups		14 B	24/11/2013 11:08:54	rwxrwxrwx	root
S99gdm		13 B	01/01/1970 01:17:11	rwxrwxrwx	root
S99rmnologin		19 B	24/11/2013 11:08:54	rwxrwxrwx	root

Gdm permet de lancer une interface graphique pour vous connecter afin de ne plus passer par une console.
Gdm est donc le gestionnaire de connexion à gnome
Rmnologin est un service qui permet de se loguer au système

Les consoles virtuelles permettent d'obtenir des terminaux virtuels sur une machine. Elles sont définies dans **/etc/inittab**. Elles sont disponibles via les périphériques **/dev/ttyn** où n est le numéro de console. /dev/ttyn représente la console virtuelle n.

● /dev/tty0 représente la console courante.

Vous pouvez changer de niveau à la volée après le démarrage de la machine avec la commande **/sbin/init** ou **/sbin/telinit**, cette dernière étant un simple lien symbolique vers init. La commande suivante passe au niveau 5.
telinit 5

Compilation d'un applicatif dans le domaine utilisateur

Pour développer sur la plateforme LINUX embarqué en langage C ou C++, deux possibilités s'offrent à nous :

- L'utilisation d'un Cross compilateur qui nous permet d'avoir toute la chaine d'outil de compilation pour un processeur à architecture ARM mais sur une machine du type à architecture INTEL X86 ou X64 (Windows ou LINUX). On choisira la suite eclipse sous windows.

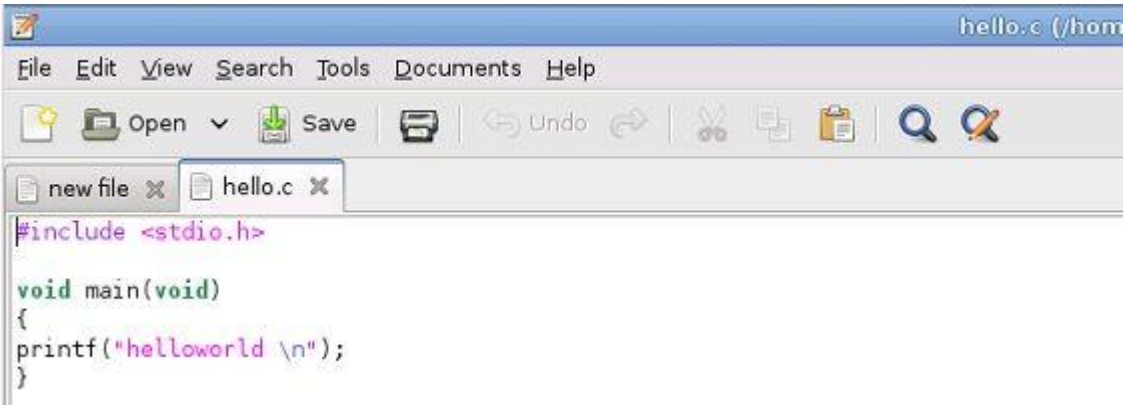
- De mettre directement les outils de compilation sur la carte cible Raspberry pi3 en bénéficiant d'une connexion internet sur celle-ci . Le compilateur gcc intégré à la suite QT est installé.

Compilons notre premier programme en langage C directement sur la carte Raspberry. Un simple éditeur de texte nous permet de saisir notre ficher source.
Avant tout chose, créons un répertoire :

```
root@beagleboard:/home# mkdir hello
root@beagleboard:/home# cd hello
```

sudo leafpad hello.c sur la Raspberry

Complétez le code source suivant :



Sous l'environnement Raspberry PI3 les commandes QT sont:

```
pi@raspberrypi ~/hello $ qmake -project
pi@raspberrypi ~/hello $ ls
hello.c  hello.pro
pi@raspberrypi ~/hello $ qmake
pi@raspberrypi ~/hello $ ls
hello.c  hello.pro  Makefile
pi@raspberrypi ~/hello $ make clean
/usr/local/qt5/bin/qmake -o Makefile hello.pro
rm -f hello.o
rm -f *~ core *.core
pi@raspberrypi ~/hello $ make
pi@raspberrypi ~/hello $ ls
hello  hello.c  hello.o  hello.pro  Makefile
pi@raspberrypi ~/hello $ ./hello
helloworld
```

Création automatique du fichier .pro

Création automatique du Makefile

compilation

Execution du programme

Les modules sous LINUX

1) Présentation

Dans les cas d'applications classiques de Linux, **le noyau utilise le plus souvent des modules** qui peuvent être dynamiquement chargés et déchargés en fonction des besoins du système. **Ces modules peuvent être des pilotes de périphériques matériels**, comme des cartes d'extension, ou bien liés à un support générique. Au démarrage, un noyau minimal est chargé puis les modules nécessaires sont chargés/déchargés au fur et à mesure des besoins du noyau.
Un pilote de périphérique peut être implémenté directement dans le noyau de LINUX sans être obligé de passer par la notion de module.
Un module se charge dans l'espace noyau de LINUX

Les commandes de base pour manipuler les modules sont les suivantes :

Afficher la liste des modules charges
lsmod

```
root@beagleboard:/home/hello# lsmod
Module                  Size  Used by
gpiokernel1mknod1       3185  0
bufferclass_ti          4768  0
omaplfb                 8733  0
pvrsrvkm              154248  2 bufferclass_ti,omaplfb
sdmak                   3743  0
lpm_omap3530            6605  0
dsplinkk              125793  1 lpm_omap3530
cmemk                  21482  0
8192cu                522320  0
ipv6                  249063  22
minix                  25759  0
rtc_twl                 4451  0
mt9t112                 9246  0
rtc_core               12535  1 rtc_twl
root@beagleboard:/home/hello#
```

Charger un module
insmod nom module.ko

Génère le fichier des dépendances entre les modules
depmod

Charge le module demande et ceux qui sont nécessaires
modprobe

Décharger un module
rmmod nom module

Obtenir des informations sur un module
modinfo nom module

Architecture d'un module hellok.c

Cette architecture est la suivante :

```
#include <linux/init.h>
#include <linux/module.h>


MODULE_LICENSE("GPL v2");

static int hello_init(void)
{
        printk(KERN_ALERT "Hello World!\n");
        return 0;
}

static void hello_exit(void)
{
        printk(KERN_ALERT "Goodbye Cruel World!\n");
}

module_init(hello_init); // lancez au chargement du module
module_exit(hello_exit);// lancez au déchargement du module
```

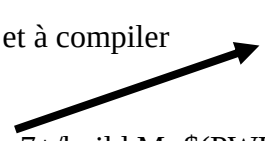
2) Création d'un module sous LINUX

2.1) Création du Makefile et du fichier source sous Raspberry PI 3

On va utiliser l'utilitaire make pour créer notre environnement de compilation, il nous faut donc déclarer un fichier Makefile définissant notre environnement de compilation.

Sous raspberry PI3 la création du Makefile est la suivante :

```
pi@raspberrypi ~ $mkdir hellok
pi@raspberrypi ~ $cd hellok
pi@raspberrypi ~ $nano Makefile
```




```
obj-m += hello.o # nom du fichier objet à compiler
PWD := $(shell pwd)
all:
        make -C /lib/modules/3.18.11-v7+/build M=$(PWD) modules # librairie pour la création
clean:
        make -C /lib/modules/3.18.11-v7+/build M=$(PWD) clean
```

attention au backslah pour la directive clean et all ou le format du fichier n'est pas reconnu

Le chemin /lib/modules/3.18.11-v7+/build correspond aux librairies du noyau qui ont été installées lors de la phase de compilation du noyau sur la Raspberry pi3 (installation du noyau dans home/pi/rpi-kernel)

Créez le fichier source hellok.c avec un éditeur de texte

```
pi@raspberrypi:/home/pi/hellok#nano hellok.c
pi@raspberrypi:/home/pi/hellok# make
make -C /lib/modules/3.18.11-v7+/build M=/home/pi/hellok modules # librairie pour
la crÃ©ation
in the future
  MODPOST 1 modules
CC      /home/pi/hellok/hello.mod.o
  LD [M]  /home/pi/hellok/hello.ko
pi@raspberrypi:/home/pi/hellok# insmod hello.ko
pi@raspberrypi:/home/pi/hellok# dmesg
[  986.203490] helloworld
pi@raspberrypi:/home/pi/hellok# modinfo hellok.ko
pi@raspberrypi:/home/pi/hellok# rmmod hellok
pi@raspberrypi:/home/pi/hellok# dmesg
[  986.203490] Goodbye Cruel World
```

Insertion du module

Message noyau

Information du module

Déchargement du module

Pour insérer notre module

insmod hellok.ko pour charger le module

lsmod pour visualiser les modules qui sont chargés

rmmod hellok.ko pour décharger le module

dmesg pour voir les messages du noyau
Le module doit être compatible avec le noyau qui tourne sur la raspberry pi 3 d'où la nécessité d'avoir compilé le noyau et d'utiliser ces librairies par la suite.

Résumé des principales lignes de commande sous LINUX

cd <nomdurépertoire>	pour changer de répertoire
cd <repertoire1>/<srepertoire1>/<ssrepertoire1>	pour naviguer vers le sous sous repertoire1 du répertoire1
cd ..	pour remonter d'un cran dans la hiérarchie des répertoires
cd /	pour remonter à la racine
pwd	pour visualiser les répertoires sous répertoires
ls	pour lister la liste des soud-répertoires et fichiers
ls -all	pour lister la liste des soud-répertoires et fichiers avec toutes les informations
uname -a	pour voir la version du noyau
sudo ./<nom1>	pour lancer l'excutable baptisé nom1 en mode administrateur
sudo chmod 777 <nomdufichier>	pour donner tous les droits à un fichier
lsmod	pour voir les modules chargés
sudo insmod <nomdumodule>	pour insérer un module
lsusb	pour voir les périphériques USB
cat /proc/interrupts	pour voir la gestion des interruptions
dmesg	pour voir les messages noyaux
sudo ifconfig	pour voir la config réseau
sudo leafpad <hello.c>	pour ouvrir un fichier texte hello.c
ps -aux	pour voir les processus en cours
sudo kill <numeropid>	pour tuer un processus en cours de numéro de pid numeropid
top	pour voir les processus en cours d'execution
df -g	pour visualiser le format des partitions
rmdir -Rf <nomrep>	pour effacer un répertoire et tous son contenu avec son arborescence
rm <nomfichier>	pour effacer un fichier
sudo reboot	pour redémarrer
sudo halt	pour arreter
sudo date --set=01/18/2016	pour changer la date en cours
su <nomdelutilisateur>	pour changer d'utilisateur nomdelutilisateur
sudo vncserver :1 -geometry 1280x768 -depth 16	pour lancer le serveur VNC
sudo service networking restart	pour relancer les services réseau sans redémarrer
sudo route add default gw 192.168.1.1	pour fixer une adresse passerelle
sudo netstat -paunt	pour lister les serveurs et les clients connectés
sudo service --status-all	pour lister les services ouverts
`cat /etc/*-release`	`pour connaitre la version de la distribution`

Les directives de compilation d'un programme en C ou C++

qmake -project

qmake

make clean

make

gcc -o mp9c mp9c.c -*****