Chapitre 4
Préparer la carte micro SD

1. Introduction

Le Raspberry Pi est un ordinateur peu encombrant et peu onéreux. Ses concepteurs, dans un objectif de simplicité et de prix plafond du produit, ont choisi de faire démarrer leur ordinateur sur un des types de mémoire les plus répandus : la carte micro SD.

Après une explication détaillée du démarrage du Raspberry Pi, ce chapitre explique comment récupérer le système d'exploitation puis l'installer sur une carte micro SD, pour les débutants comme pour les plus experts.

Le téléchargement est réalisé sous Windows 8. Les utilisateurs de Linux en mode graphique transposeront aisément sur ce système. L'installation du système sur la carte micro SD est détaillée sous Windows 8 et sous Debian 7. Le système d'exploitation choisi est celui que préconise la Fondation Raspberry Pi pour les utilisateurs débutants : Raspbian.

La syntaxe des commandes présentées est limitée aux options utilisées dans ce chapitre. Pour plus d'informations, reportez-vous au chapitre Utiliser la ligne de commande ainsi que le `man` (manuel) de chaque commande.

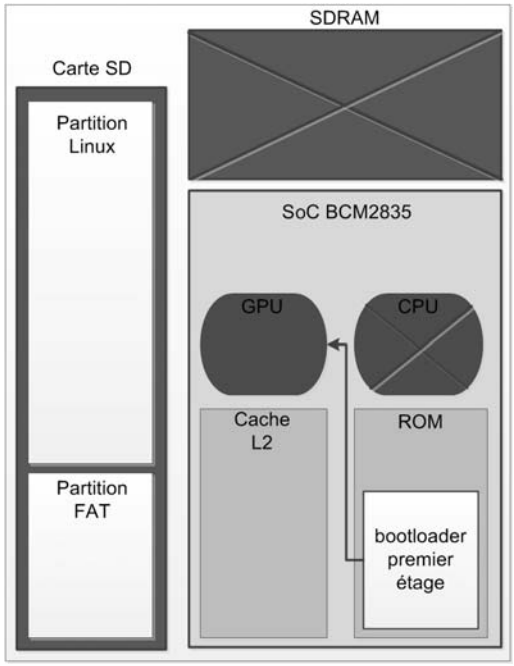
Les versions des systèmes d'exploitation et des utilitaires évoluent. Il vous appartient d'adapter les informations qui suivent en fonction des versions disponibles lorsque vous préparerez votre carte micro SD.

2. Séquence de boot du Raspberry Pi

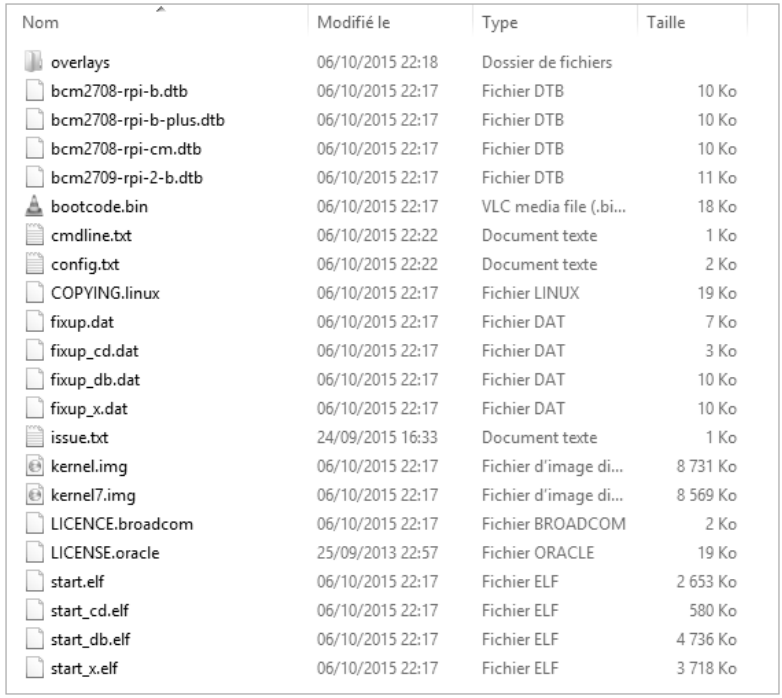
Le *boot* (démarrage) du Raspberry Pi est strictement identique sur les différents modèles Pi Zero et Raspberry Pi 3. Ce processus de démarrage implique un certain nombre d'opérations. La bonne compréhension de cette séquence est primordiale lorsqu'il s'agit de modifier volontairement le démarrage du système d'exploitation (démarrer sur une clé USB ou un disque dur). Mais lorsque le système ne démarre pas, l'utilisateur est en présence d'une énigme que seule la connaissance du déroulement exact de la séquence de boot permet de résoudre.

2.1 Étape 1 : mise sous tension

Le schéma ci-dessous représente les différentes parties impliquées dans le démarrage du Raspberry Pi. Sur la partie gauche figure la carte micro SD divisée en deux partitions. Le SoC intègre le microprocesseur ARM (CPU) et le processeur graphique (GPU). Il est situé à droite, sous la mémoire (SDRAM).



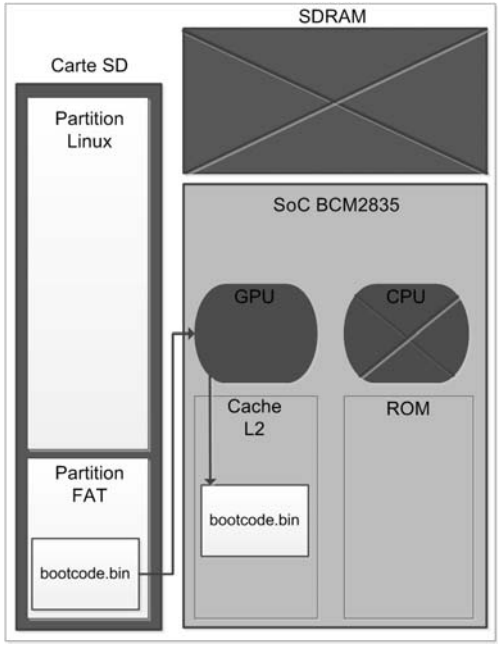
Lors de la mise sous tension, seul le GPU est activé. Il maintient le CPU à l'arrêt. La mémoire SDRAM est elle aussi désactivée. Le SoC contient une ROM (*Read Only Memory*) dans laquelle le fabricant a enregistré un programme faisant partie de la chaîne de boot. Ce programme est le premier de la chaîne de boot (*bootloader* = chargeur de démarrage). Il est inaccessible par l'utilisateur et n'est donc pas modifiable. Le GPU exécute ce premier programme dont le seul rôle est d'accéder à la partition FAT de la carte micro SD pour charger le fichier *bootcode.bin* en mémoire. Cette partition contient les fichiers suivants :



Nom	Modifié le	Type	Taille
overlays	06/10/2015 22:18	Dossier de fichiers	
bcm2708-rpi-b.dtb	06/10/2015 22:17	Fichier DTB	10 Ko
bcm2708-rpi-b-plus.dtb	06/10/2015 22:17	Fichier DTB	10 Ko
bcm2708-rpi-cm.dtb	06/10/2015 22:17	Fichier DTB	10 Ko
bcm2709-rpi-2-b.dtb	06/10/2015 22:17	Fichier DTB	11 Ko
bootcode.bin	06/10/2015 22:17	VLC media file (.bi...	18 Ko
cmdline.txt	06/10/2015 22:22	Document texte	1 Ko
config.txt	06/10/2015 22:22	Document texte	2 Ko
COPYING.linux	06/10/2015 22:17	Fichier LINUX	19 Ko
fixup.dat	06/10/2015 22:17	Fichier DAT	7 Ko
fixup_cd.dat	06/10/2015 22:17	Fichier DAT	3 Ko
fixup_db.dat	06/10/2015 22:17	Fichier DAT	10 Ko
fixup_x.dat	06/10/2015 22:17	Fichier DAT	10 Ko
issue.txt	24/09/2015 16:33	Document texte	1 Ko
kernel.img	06/10/2015 22:17	Fichier d'image di...	8 731 Ko
kernel7.img	06/10/2015 22:17	Fichier d'image di...	8 569 Ko
LICENCE.broadcom	06/10/2015 22:17	Fichier BROADCOM	2 Ko
LICENSE.oracle	25/09/2013 22:57	Fichier ORACLE	19 Ko
start.elf	06/10/2015 22:17	Fichier ELF	2 653 Ko
start_cd.elf	06/10/2015 22:17	Fichier ELF	580 Ko
start_db.elf	06/10/2015 22:17	Fichier ELF	4 736 Ko
start_x.elf	06/10/2015 22:17	Fichier ELF	3 718 Ko

2.2 Étape 2 : chargement de bootcode.bin

Le schéma ci-dessous explique la première phase de démarrage du Raspberry Pi au cours de laquelle le GPU charge le code de démarrage en mémoire.



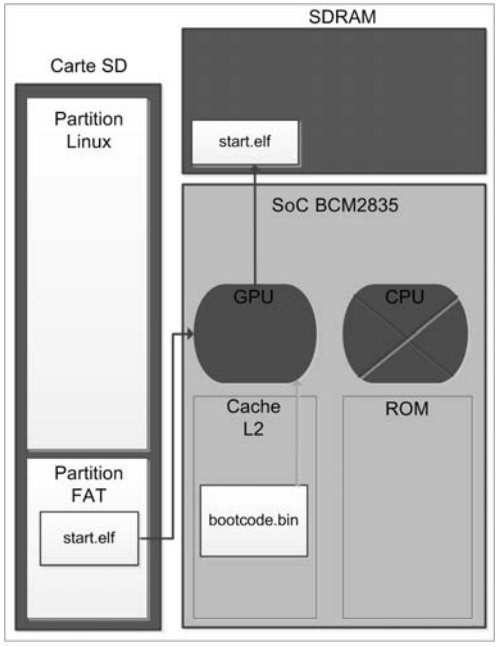
Le GPU exécute le programme contenu dans la ROM du SoC. Il accède à une partition FAT de la carte micro SD, trouve le fichier *bootcode.bin* qu'il charge dans la mémoire cache L2 (*Level 2* = mémoire cache de niveau 2).

Le programme dans la ROM ainsi que le programme *bootcode.bin* sont écrits en code spécifique pour le GPU. Les spécifications du GPU ne sont pas publiées, et *bootcode.bin* ne peut être redistribué qu'en format binaire. Ce n'est pas un programme libre (licence Broadcom).

Comme la mémoire SDRAM montée sur le SoC n'est pas encore activée, c'est la mémoire cache du GPU, la mémoire L2, qui est utilisée pour le chargement de *bootcode.bin*.

2.3 Étape 3 : exécution de bootcode.bin par le GPU

Le schéma ci-dessous explique la deuxième phase de démarrage du Raspberry Pi au cours de laquelle le GPU charge le firmware (*start.elf*) en mémoire.

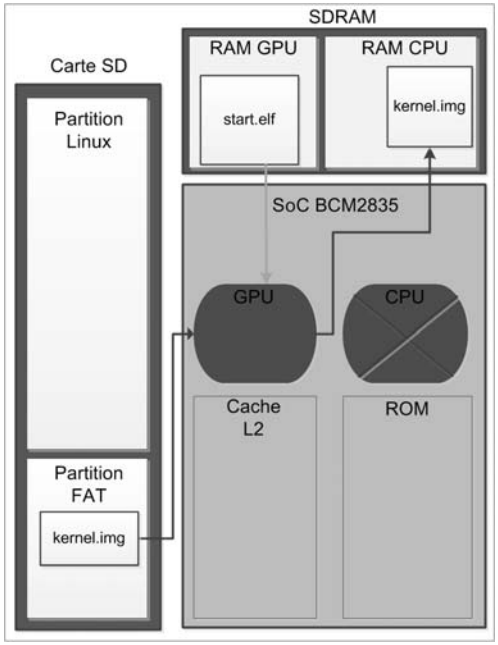


Une fois *bootcode.bin* chargé dans la mémoire cache L2, le GPU exécute ce programme (flèche grise) qui est le second étage du bootloader. Le but final est de récupérer le programme *start.elf*, qui se situe également sur la carte micro SD, dans la partition FAT. *start.elf* est le firmware du GPU. Comme *bootcode.bin*, ce programme n'est pas libre et est distribué sous forme binaire.

Le GPU, sous les ordres de *bootcode.bin*, active la mémoire SDRAM du Raspberry Pi et transfère une copie de *start.elf* dans le haut de la mémoire. Une fois *start.elf* chargé en mémoire, *bootcode.bin* lui passe le relais.

2.4 Étape 4 : exécution de start.elf par le GPU

Le schéma ci-dessous indique comment le GPU partage la mémoire avec le CPU en fonction des paramètres puis charge le noyau Linux en mémoire.



Le GPU exécute maintenant son firmware : *start.elf* (flèche grise).

start.elf répartit la mémoire entre le GPU (partie haute) et le CPU ARM (partie basse) en fonction du paramètre présent dans le fichier de configuration (*config.txt*). Le paramètre *gpu_mem* indique en Mo la quantité de mémoire qui doit être allouée au GPU. Si ce paramètre est absent du fichier *config.txt*, la valeur est fixée par défaut à 64 Mo. La valeur de *gpu_mem* est au minimum 16 Mo. Elle doit être un multiple de 16 Mo.

Après la mise en place des zones mémoire, le programme *start.elf* transfère le noyau Linux *kernel.img* dans la partie basse de la mémoire, zone réservée au CPU ARM. Il lit ensuite le fichier *cmdline.txt* qui contient les arguments à passer au noyau lors de son exécution.