Paris Embedded Meetup

L'actualité de l'embarqué libre

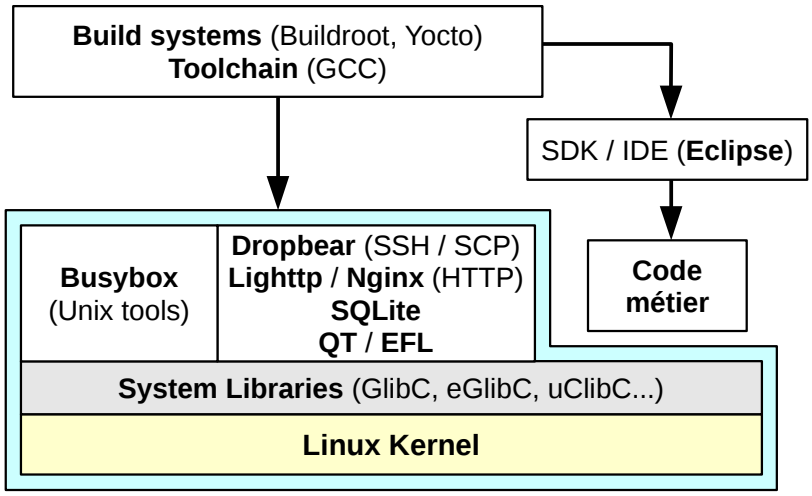
Mars 2015

Christophe Blaess

http://christophe.blaess.fr

www.logilin.fr

Composants d'un système Linux embarqué



Un système Linux embarqué est le fruit de nombreux outils, qui évoluent en permanence.

Cette édition de *l'actualité de l'embarqué libre* est consacrée aux évolutions du **noyau Linux** depuis un an, plus particulièrement ce qui concerne le domaine de l'embarqué.

Pour suivre l'actualité du noyau :

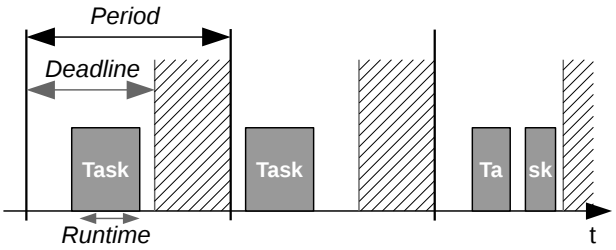
- http://lwn.net
- http://kernelnewbies.org/LinuxChanges
- http://linuxfr.org/wiki/depeches_noyau

Linux Kernel

Linux 3.14 – 2014/03/31

The Deadline Scheduling Policy

Une nouvelle classe d'ordonnancement `SCHED_DEADLINE` cumulant les algorithmes CBS (*Constant Bandwith Server*) et EDF (*Earliest Deadline First*) garantit la disponibilité régulière du CPU.



Deux nouveaux appels-système et une structure de configuration :

```
int sched_setattr  (pid_t task,  const struct sched_attr * attr,
                    unsigned int flags);
int sched_gettattr (pid_t task, struct sched_attr * attr,
                    unsigned int size,  unsigned int flags);
```

```
struct sched_attr {

        u32 size;

        u32 sched_policy;
        u64 sched_flags;

        /* SCHED_NORMAL, SCHED_BATCH */
        s32 sched_nice;

        /* SCHED_FIFO, SCHED_RR */
        u32 sched_priority;

        /* SCHED_DEADLINE */
        u64 sched_runtime;
        u64 sched_deadline;
        u64 sched_period;
};
```

La fiabilité de cet ordonnancement est identique à celle des ordonnancements temps réel classiques `SCHED_RR` et `SCHED_FIFO` : une tâche `SCHED_DEADLINE` ne sera pas perturbée par l'activité d'une tâche `SCHED_RR`, `SCHED_FIFO` ou `SCHED_OTHER` (temps partagé) mais sera soumise aux fluctuations dues à l'activité du noyau (interruptions, timers, pile réseau, etc.)

Zram block device

Le périphérique bloc *zram* (disque RAM compressé) est sorti du statut *staging*.

Option de compilation du noyau :

```
Device drivers -->
    [*] Block devices -->
        <M> Compressed RAM block device support
```

On peut l'utiliser en support de *swap* compressé pour augmenter la mémoire disponible :

```
# modprobe zram
# echo $((128*1024*1024)) > /sys/block/zram0/disksize
# mkswap /dev/zram0
Setting up swapspace version 1, size = 131068 KiB
no label, UUID=adebd276-09bc-457e-9972-5d925a873a25
# swapon /dev/zram0
# free
             total       used       free     shared    buffers     cached
Mem:       3863560    3480448     383112     345680     873508    1077972
-/+ buffers/cache:    1528968    2334592
Swap:       131068          0     131068
# cat /sys/block/zram0/mem_used_total
4096
#
```

On fixe la dimension maximale du périphérique bloc (128 Mo ci-dessus) et on peut l'utiliser comme support pour le swap. Le contenu est compressé, il utilise donc moins de mémoire que les 128 Mo configurés (4 Ko ci-dessus).

Linux 3.15 – 2015/06/08

Échange de deux fichiers

L'appel-système `renameat()` permet de renommer ou déplacer un fichier :

```
int renameat(int old_dir_fd, const char *old_path,
             int new_dir_fd, const char *new_path);
```

Le nouvel appel-système `renameat2()` ajoute un argument supplémentaire `flag`.

```
int renameat2(int old_dir_fd, const char *old_path,
              int new_dir_fd, const char *new_path,
              int flag);
```

Si `flag` vaut `RENAME_EXCHANGE`, les noms des deux fichiers sont échangés atomiquement.

Améliorations de F2FS (*Flash Friendly File-System*)

Pour les mémoires Flash NAND avec contrôleurs (eMMC, SD cards, disques SSD, etc.).

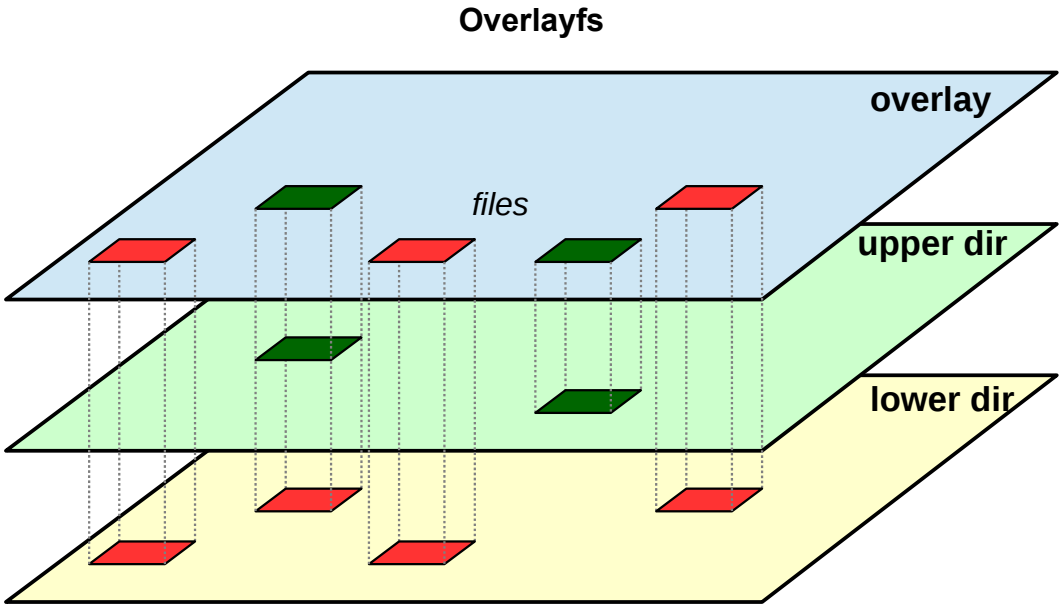
Système de fichiers inspiré de LFS (*Log File System*) assurant une bonne rapidité de lecture et de récupération en cas de crash.

F2FS a été régulièrement amélioré dans toutes les versions récentes de Linux.

L'échange atomique de deux fichiers peut servir par exemple à échanger les versions « N » et « N+1 » d'un code métier à la mise à jour ou à activer un nouveau fichier de configuration en gardant une sauvegarde de la version précédente, etc.

Les systèmes de fichiers utilisés typiquement pour les mémoires flash avec contrôleurs (VFAT, EXT2/3/4, etc.) ne sont pas spécifiquement conçus pour cet usage. **F2FS** est prévu pour assurer une répartition de l'usure (*Wear Leveling*) – en complément de celle proposée par le contrôleur de flash. L'organisation du système de fichiers permet une meilleure robustesse en cas de coupure d'alimentation électrique pendant une écriture.

Linux 3.18 – 2014/12/07



On peut superposer des répertoires (éventuellement sur différents systèmes de fichiers) afin que les modifications n'aient lieu que sur le répertoire supérieur.

```
mount none -t overlayfs -o lowerdir=/etc,upperdir=/data/etc,workdir=/data/work /etc
```

L'intérêt d'`overlayfs` pour les systèmes embarqués est de conserver le répertoire inférieur sur une partition montée en lecture seulement (avec la configuration sortie d'usine), et de faire toutes les modifications sur une partition indépendante que l'on peut toujours effacer ou reformater en cas de problème au démarrage.

Pour plus de détails, voir l'article « *Le système Overlayfs de Linux 3.18* » (14 décembre 2014) sur http://www.blaess.fr/christophe

Linux 3.19 – 2015/02/08

Device Tree Overlays

Le *Device Tree* permet de décrire des périphériques que le noyau ne peut pas découvrir seul.

Il est fourni au noyau par le *bootloader*.

Les ***overlays*** permettent une modification **dynamique** du *Device Tree*.

Extrait de l'overlay pour l'UART de la carte *Beagle Bone Black* :

```
[...]
fragment@0 {
    target = <&am33xx_pinmux>;
    __overlay__ {
        bb_uart1_pins: pinmux_bb_uart1_pins {
            pinctrl-single,pins = <
                0x184 0x20 /* P9.24 uart1_txd.uart1_txd MODE0 OUTPUT (TX) */
                0x180 0x20 /* P9.26 uart1_rxd.uart1_rxd MODE0 INPUT (RX) */
            >;
        };
    };
};
```

Pour plus d'information voir les liens suivants :

- https://learn.adafruit.com/introduction-to-the-beaglebone-black-device-tree/
- http://derekmolloy.ie/gpios-on-the-beaglebone-black-using-device-tree-overlays/

Nios II support

Nouvelle architecture `arch/nios2` pour les processeurs *softcore* des FPGA Altera :

```
[linux-3.19]$ make  ARCH=nios2  help
   [...]
  3c120_defconfig   - Build for 3c120
   [...]
```

Slave i²c framework

```
Device Drivers  --->
        I2C support  --->
                [...]
                [*] I2C slave support
                        <*> I2C eeprom slave driver
```

« Device Drivers -> Android »

Le code du *Binder* d'Android (communication entre processus) est officiellement intégré.

```
Device Drivers  --->
        Android  --->
                [*] Android Drivers
                        [*] Android Binder IPC Driver
```

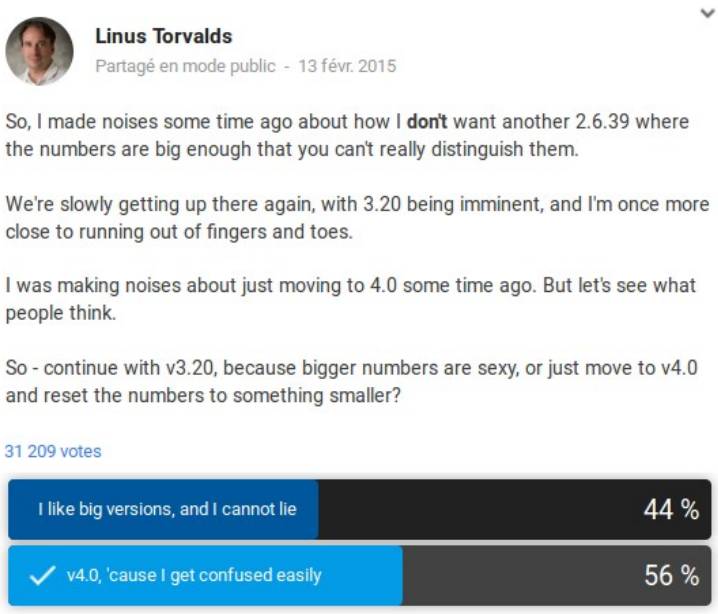
Les FPGA Altera peuvent intégrer des processeurs *Soft Core* (implémentés entièrement dans le FPGA) comme le **Nios II**. Celui-ci est maintenant supporté par le noyau standard. Notez qu'il existe également des processeurs *HardCore* intégrés dans les FPGA haut-de-gamme comme le Cyclone V.

Le noyau Linux supporte depuis longtemps le protocole **i²c** utilisé pour dialoguer avec de nombreux capteurs. Jusqu'à présent, seul le mode « *master* » était supporté, c'est-à-dire que le noyau imposait sa propre horloge pour la communication. Dorénavant, le mode « ***slave*** » est disponible, le système se comportant comme un périphérique i²c acceptant l'horloge d'un maître externe. Il s'agit pour le moment d'un *framework* général dans lequel des drivers viendront s'inscrire à l'avenir.

Le **Binder** est le module chargé de la communication entre les processus d'Android (à la manière des IPC Unix classiques). Il était présent depuis quelques temps dans le répertoire `staging/` du noyau Linux et vient d'être intégré dans l'arborescence `drivers/`.

Plus que l'aspect technique, c'est le côté symbolique de l'intégration de *patches* Android dans le noyau Linux qui est à noter.

Linux 4.0 !



Suite à un vote public sur Google+, Linus Torvalds a décidé que le prochain noyau ne serait pas un 3.20 mais un **4.0**.

Pour plus de détails, voir l'article « *Linux 4* » (24 février 2014) sur http://www.blaess.fr/christophe