HACKABLE MAGAZINE

https://www.hackable.fr/?p=752

Raspbian Jessie, systemd et module RTC sur Raspberry Pi

5 janvier 2016 par Denis Bodor

Soyons honnêtes et directs : je déteste systemd ! C'est une horreur, une aberration, un monstre ignoble !

systemd est le système d'initialisation des nouvelles distributions GNU/Linux, qui remplace le bon vieux init sysV hérité d'UNIX et qui convenait parfaitement à tout le monde depuis des décennies jusqu'à récemment. Contrairement au bon vieux système d'*init* composé de scripts forts sympathiques et surtout intelligibles en quelques instants, *systemd* oublie totalement la philosophie UNIX de base : ne pas compliquer inutilement les choses (principe KISS).

systemd est une usine à gaz comparé à *init* et, de ce fait, lorsqu'on veut ajouter le support pour un simple module RTC sur une Raspberry Pi, histoire de la garder à l'heure sans reposer sur le réseau, ceci devient très vite énervant.

À la base, tout est simple. Grâce au *Device Tree*, il suffit de préciser qu'on veut utiliser le bus i2c et un module à base de DS3231 :

```
dtparam=i2c_arm=on
dtoverlay=i2c-rtc,ds3231
```

Partant de là, le noyau sait qu'il doit prendre en charge la chose et s'occupe de tout… ou presque. En effet, même si un /dev/rtc0 fait son apparition, ce n'est pas pour autant qu'il est utilisé.

Avec les versions précédentes de Raspbian il suffisait de préciser dans le fichier /etc/default/hwclock que nous avions un module RTC et le système restait simplement à l'heure… Mais ça c'était avant…

systemd a la bonne idée (sarcasmes) de faire le travail de hwclock… en partie. Il part, en effet, du principe que le noyau est configuré pour récupérer l'heure dans la RTC et va l'appliquer au système au démarrage. En revanche, à l'arrêt du système, systemd enregistre l'heure courante dans la RTC parce qu'elle a très certainement été mise à jour via le réseau avec NTP et qu'elle est bien plus juste (re-sarcasmes).

Seulement voilà, le noyau par défaut de Raspbian n'a pas cette option d'activée. Et même s'il vous prend l'envie de recompiler un noyau (cf *Hackable n°10* actuellement en kiosque), ça ne marchera pas :

```
rtc-ds1307 1-0068: rtc core: registered ds3231 as rtc0
hctosys: unable to open rtc device (rtc0)
```

Hé oui ! Le noyau, au démarrage ne peut pas accéder à la RTC, car le support est fourni par un module, chargé bien après cette tentative de mise à l'heure… Ceci ne fonctionne qu'avec les supports i2c et RTC compilés directement dans le noyau, pas en modules.

Il faut donc maintenant bidouiller pour que cela fonctionne (on appelle cela une « régression ») et comme on veut faire les choses proprement, il faut créer un service systemd.

On commence donc par créer un fichier /etc/systemd/system/rtc-init.service contenant :

```
[Unit]
Description=RTC Clock Setup and Time Sync
Before=cron.service

[Service]
Type=oneshot
ExecStart=/usr/lib/systemd/scripts/rtc-setup

[Install]
WantedBy=multi-user.target
```

Ceci défini un nouveau service qui ne s'exécute qu'une fois et est lié à un script qu'il faut également écrire. C'est le fichier /usr/lib/systemd/scripts/rtc-setup :

```
#!/bin/sh
hwclock -s --utc
echo "System Time synced with RTC Time"
```

On n'oubliera pas de le rendre exécutable avec sudo chmod +x /usr/lib/systemd/scripts/rtc-setup.

Et enfin, nous pouvons activer le service en question avec sudo systemctl enable rtc-init.

Voilà ! Au prochain démarrage, la Raspberry Pi se mettra à l'heure contenue dans le module…

Pourquoi ceci n'est pas fait par défaut ? Simplement parce que les Raspberry Pi n'ont pas de RTC et que c'est un ajout fait, ou non, par l'utilisateur.

À mon sens, systemd n'est clairement pas adapté à ce type d'utilisation sur nano-ordinateur pour bidouilleurs ou même aux systèmes embarqués en général. Il est utilisé avec Raspbian,

car ce système dérive de Debian GNU/Linux qui depuis 2015 et Jessie a basculé sur systemd comme bien d'autres distributions.

Cependant, systemd est maintenant utilisé sur Raspberry Pi et rend certaines tâches bien plus difficiles en offrant que peu d'avantages. Alors qu'il était relativement simple et rapide d'ajouter ou modifier un service avec le bon vieux init, il faut maintenant comprendre l'architecture complexe d'un système d'initialisation qui s'occupe aussi bien du démarrage que du plug'n'play (udev), des journaux systèmes, du montage/démontage des systèmes de fichiers, de la sécurité, de l'analyse des performances…

Souvenez-vous pourquoi les systèmes UNIX (GNU/Linux, Mac OS X, *BSD) sont toujours présents et utilisés après plus de 40 ans : « Ne faire qu'une seule chose, et la faire bien » ! Je pense que c'est une erreur de s'éloigner de cette philosophie, mais systemd est là et il faut maintenant faire avec…