systemd

Une alternative au système d'initialisation pour Linux SysV init

Administration système de base Linux

Version 0.1 - Michel SYSKA

L'objectif premier de ce document est de fournir une synthèse en français des concepts et commandes utiles de `systemd` *aux étudiants de la LPMI IOTIA, aussi bien dans le cadre de la formation que lors des stages en entreprise. Les étudiants de la LPMI Admin promotion 2015 - 2016 ont testé les exemples sur CentOS 7.1, Ici on teste sous Ubuntu 18.04. Les tutoriels et documentations utilisés pour cette présentation sont cités dans la dernière section "Références".*

1 Amorçage

À la mise sous tension de la machine, c'est le BIOS (Basic Input Output System), ou son successeur UEFI (Unified Extensible Firmware Interface), qui prend en charge le processus de démarrage. L'ancien BIOS trouve la table de partitions et le chargeur de démarrage (on dit aussi chargeur d'amorçage) dans le mbr (Master Boot Record). Ce chargeur de démarrage (Microsoft boot loader, LILO, GRUB, ...) lit le noyau sur la partition indiquée dans sa configuration et le charge en mémoire pour l'exécuter. Dans le cas de l'UEFI, c'est une table de partitions GPT (GUID Partition Table) qui est utilisée. Ce nouveau standard permet en particulier de gérer plus de partitions (128 au lieu de 4 sous Windows) et de plus grande taille (256 To au lieu de 2,2 To). Les bootloader sont lus dans une partition FAT dédiée. En ce qui nous concerne, c'est Grub2 qui va charger le noyau Linux de notre installation Ubuntu.

Nous reviendrons plus en détail sur cette configuration, notamment dans le cas d'un système multiboot, mais ce qui nous intéresse ici c'est la suite, le démarrage et la gestion des services sous Linux.

Dans les distributions Linux, le boot était généralement contrôlé par init et on parle de init System V. De nombreuses distributions comme celles de la famille RedHat ou Ubuntu ont adopté un nouveau gestionnaire nommé systemd (System Management Daemon), mais les deux gestionnaires cohabitent et on va donc présenter certaines équivalences entre init et systemd.

2 Démarrage et gestion des services

Au démarrage du système, il faut exécuter un certain nombre de tâches avant d'arriver à un état fonctionnel satisfaisant : monter des partitions, activer des interfaces réseau, monter des partages réseau, se connecter aux annuaires d'authentification, passer éventuellement en mode graphique, … Parfois, on souhaite aussi lancer automatiquement des programmes comme un pare-feu, un gestionnaire de mises à jour, un serveur de bases` de données, un serveur Web, … Ces programmes sont communément appelés des daemon ou services et le nom de l'exécutable associé termine habituellement par la lettre d : firewalld, packagekitd, mysqld, httpd, … Le principe d'un daemon est de s'exécuter en arrière plan, avec une boucle infinie qui traite les requêtes jusqu'à la réception d'un signal de terminaison.

Le problème est alors de gérer l'ordre dans lequel les services sont démarrés ou terminés, la liste des services qui seront automatiquement lancés au boot, etc … On veut aussi pouvoir relancer un service après modification de sa configuration, le désactiver provisoirement, … Enfin, on veut aussi pouvoir arrêter le système "proprement".

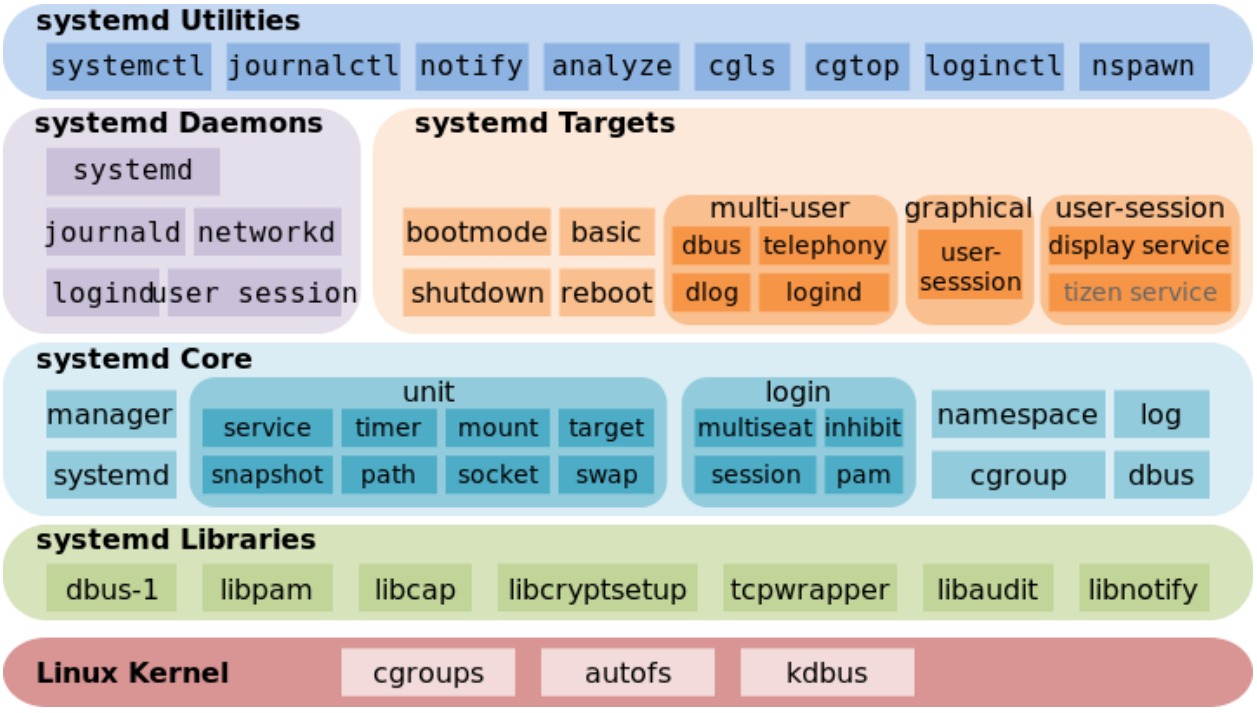

Image courtesy Wikimedia Commons, CC BY-SA 3.0

Systemd est censé être plus rapide que init pour gérer ces besoins car il tente de paralléliser les tâches et comme pour launchd (Mac OS X) on doit obtenir un meilleur temps de démarrage. Systemd propose aussi un système d'activation de sockets qui permet de lancer des services en arrière plan sans gérer les priorités entre ses services (une fois que les services de base sont actifs).

Systemd offre d'autres fonctionnalités comme la gestion des logs, des Control Groups, …

3 Units, Targets, CGroups

Toutes les tâches gérées par systemd sont organisées en units ou unités. À chaque unité correspond un type et un fichier de configuration. Par exemple, l'unité qui permet de gérer la tâche sshd est de type service et son fichier de configuration est le suivant.

```
# cat /etc/systemd/system/sshd.service

[Unit]
Description=OpenBSD Secure Shell server
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run

[Service]
EnvironmentFile=-/etc/default/ssh
ExecStartPre=/usr/sbin/sshd -t
```

```
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/usr/sbin/sshd -t
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartPreventExitStatus=255
Type=notify
RuntimeDirectory=sshd
RuntimeDirectoryMode=0755

[Install]
WantedBy=multi-user.target
Alias=sshd.service
```

Nous présenterons les autres types d'unités quand cela sera nécessaire.

Les unités peuvent être regroupées en targets ou cibles qui sont notamment utiles pour gérer les dépendances entre groupes d'unités. Par exemple, les unités de la cible graphical.target doivent être lancées après la cible multi-user.target qui doivent elles être lancées après basic.target.

Tous les processus associés à un service sont placés dans un cgroup, ou groupe de contrôle, pour en faciliter la gestion globale.

Dans la suite, nous allons brièvement présenter le système de contrôle de base des services, puis nous reviendrons sur les cibles et le boot. Enfin nous présenterons des outils d'analyse du boot et de la gestion des logs.

4 Contrôle des services : systemctl

La commande `systemctl` remplace notamment les commandes `/sbin/service` et `chkconfig` de System V init. En plus de l'aide en ligne et des documents cités en référence, n'oubliez pas de lire le manuel :

```
$ man systemctl
```

```
# systemctl
```

sans argument affiche la liste des unités chargées ainsi que leur état.

```
# systemctl list-unit-files
```

affiche la liste des unités installées ainsi que leur état. Avec un petit peu de bash (pour réviser) on peut en extraire la liste des différents types d'unité existant.

```
# systemctl list-unit-files | head -n -2 | tail -n +2 | grep -o '^[^
]*' | grep -o '[^.]*$' | sort | uniq

automount
mount
path
scope
service
slice
socket
swap
target
timer
```

Dans cette présentation, nous allons essentiellement travailler avec les types service et target. Pour ce qui est des services, nous illustrons dans la suite les principales commandes avec l'exemple du service www et de son daemon apache2 (httpd sous RedHat).

Toutes les informations sur l'état d'un service sont affichées par la commande :

```
# systemctl status apache2
```

Pour activer apache2 au boot :

```
# systemctl enable apache2
```

ou

```
# systemctl enable apache2.service
```

Le type est déduit automatiquement.

Pour savoir si apache2 est activé au boot :

```
# systemctl is-enabled apache2
```

Pour traiter le résultat dans un script, voici deux exemples utiles:

```
# systemctl is-enabled apache2.service
disabled
# echo $?
1
# systemctl is-enabled sshd.service
enabled
# echo $?
0
```

Pour annuler le lancement d'apache 2 au boot :
```
# systemctl disable apache2
```

Pour savoir si apache2 est actif (même traitement possible que pour is-enabled) :
```
# systemctl is-active apache2
```

Pour démarrer temporairement apache2 s'il n'est pas actif:
```
# systemctl start apache2
```

Pour arrêter apache2 :
```
# systemctl stop apache2
```

Pour redémarrer apache2 :
```
# systemctl restart apache2
```

Pour que apache2 relise sa configuration (ici apache2.conf) et non celle de l'unité sans interrompre le service (cette option n'est pas disponible pour tous les services) :
```
# systemctl reload apache2
```
Il ne faut pas la confondre avec la commande:
```
# systemctl daemon-reload
```
qui doit être utilisée après modification du fichier de configuration d'une unité pour recharger le daemon de systemd. Dans ce cas il faudra aussi faire un restart de l'unité en question.

5 Runlevels et targets

Avec init SysV les services sont gérés par des scripts écrits dans /etc/init.d. Les scripts acceptent des arguments comme start, stop, status, ... SysV définit des niveaux d'exécution, ou runlevels, qui correspondent à des répertoires de /etc/rc[0-6].d/ Chacun de ces répertoires contient des liens vers les scripts de /etc/init.d et le nom du lien indique avec quelle priorité le service correspondant doit être démarré (lien dont le nom commence par s) ou stoppé (lien dont le nom commence par k) dans le runlevel correspondant. La commande chkconfig permet de gérer ces liens.

Les runlevels étaient listés dans le fichier /etc/inittab (avant l'usage de systemd). Ici l'exemple de RedHat :

```
# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#  id:3:initdefault:
```

- ❏ Runlevel 0 - halt ou arrêt du système.
- ❏ Runlevel 1 – single user mode. Seul root peut se connecter, sans réseau ou environnement graphique. Ce runlevel est utilisé pour des opérations de maintenance ou de récupération après un incident.
- ❏ Runlevel 2 - multi-user mode. Connexion en mode console texte seulement et sans réseau.
- ❏ Runlevel 3 - comme le runlevel 2 mais avec les services réseau démarrés. Mode utilisé pour les serveurs qui ne nécessitent pas de connexion graphique.
- ❏ Runlevel 4 - indéfini : peut être customisé pour des besoins particuliers
- ❏ Runlevel 5 - comme le runlevel 3 mais avec l'environnement graphique. Mode utilisé pour les stations de travail.
- ❏ Runlevel 6 - reboot du système.

Ainsi, quand le système démarre, on passe de 0 à 1, puis 2, puis 3 (car défini comme mode par défaut dans le fichier inittab de notre exemple). L'utilisateur peut éventuellement se connecter et passer en runlevel 5 avec la commande init si besoin. Root peut aussi décider d'arrêter le système avec la commande init 0. Dans ce cas, on va passer de 5 à 3, puis de 3 à 2, de 2 à 1 et finir en runlevel 0. À chaque descente de runlevel, on arrête dans l'ordre des priorités les services définis dans ce niveau.

Sous Ubuntu, on peut lire les informations dans le manuel :

```
# man runlevel
```

Dans systemd, les targets offrent les mêmes fonctionnalités que les runlevels, mais de façon plus générale. Une cible est un groupe de services, et on peut définir les dépendances entre groupes pour d'autres besoins que le boot ou l'arrêt du système.

La description d'une cible est contenue dans son fichier d'unité. Par exemple, pour basic.target on a :

```
# cat /lib/systemd/system/basic.target

#  SPDX-License-Identifier: LGPL-2.1+
```

```
#
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#   under the terms of the GNU Lesser General Public License as published
by
#  the Free Software Foundation; either version 2.1 of the License, or
#  (at your option) any later version.

[Unit]
Description=Basic System
Documentation=man:systemd.special(7)
Requires=sysinit.target
Wants=sockets.target timers.target paths.target slices.target
After=sysinit.target sockets.target paths.target slices.target tmp.mount

# We support /var, /tmp, /var/tmp, being on NFS, but we don't pull in
# remote-fs.target by default, hence pull them in explicitly here. Note
that we
# require /var and /var/tmp, but only add a Wants= type dependency on
/tmp, as
# we support that unit being masked, and this should not be considered an
error.
RequiresMountsFor=/var /var/tmp
Wants=tmp.mount
```

D'autres informations sont obtenues avec :

```
# systemctl show basic.target
```

Pour changer de cible, à l'instar des runlevels décrits précédemment, on utilise l'option isolate de systemctl. Par exemple :

```
# systemctl isolate runlevel3.target
```

ou de manière équivalente

```
# systemctl isolate multi-user.target
```

permettent de passer en cible équivalente à runlevel 3 de init Sys V (connexion en mode console non graphique). Pour repasser en mode graphique (runlevel 5 dans init) :

```
# systemctl isolate graphical.target
```

Pour connaître la cible par défaut au boot:

```
# systemctl get-default
graphical.target
```

Pour changer cette cible par défaut en mode non graphique :
```
# systemctl set-default multi-user.target
```

D'autres commandes courantes utilisant les cibles sont:

```
# systemctl poweroff
```
pour arrêter la machine

```
# systemctl reboot
```
pour redémarrer

Si la gestion de l'énergie est bien prise en charge, on a aussi :
```
# systemctl suspend
# systemctl hibernate
```

Notons que par soucis de compatibilité avec init, les raccourcis poweroff, reboot … existent.

Enfin, en cas de problème, on peut passer en mode maintenance avec :
```
# systemctl rescue
```

6 Démarrage

Systemd gère le démarrage et fournit des informations à ce sujet.
Informations globales sur la durée du boot :
```
# systemd-analyze
Startup finished in 441ms (kernel) + 3.144s (initrd) + 23.137s
(userspace) = 26.723s
```

Informations détaillées pour chaque tâche :
```
# systemd-analyze blame
          5.262s initial-setup-text.service
          5.163s mariadb.service
...
          2.289s firewalld.service
          1.977s firstboot-graphical.service
```

Analyse de la chaîne critique du boot :
```
# systemd-analyze critical-chain
The time after the unit is active or started is printed after the
"@" character.
```

```
The time the unit takes to start is printed after the "+" character.

graphical.target @20.490s
└─multi-user.target @20.490s
  └─mariadb.service @12.201s +8.287s
    └─network.target @11.904s
      └─NetworkManager.service @11.625s +278ms
        └─firewalld.service @10.538s +1.086s
          └─basic.target @10.537s
            └─sockets.target @10.537s
              └─dbus.socket @10.537s
                └─sysinit.target @10.531s
...
```

Analyse de la chaîne critique d'un service particulier :

```
# systemd-analyze critical-chain sshd.service
The time after the unit is active or started is printed after the
"@" character.
The time the unit takes to start is printed after the "+" character.

sshd.service @17.932s
└─network.target @17.923s
  └─NetworkManager.service @17.791s +129ms
    └─firewalld.service @15.502s +2.289s
      └─basic.target @15.501s
        └─sockets.target @15.501s
          └─dbus.socket @15.501s
            └─sysinit.target @15.494s
...
```

Analyse de la chaîne critique d'une cible particulière :

```
# systemd-analyze critical-chain multi-user.target | grep target
multi-user.target @23.126s
  └─network.target @17.923s
        └─basic.target @15.501s
          └─sockets.target @15.501s
            └─sysinit.target @15.494s
                      └─local-fs.target @13.482s
```

Voir la liste des dépendances :

```
# systemctl list-dependencies
default.target
 ├──accounts-daemon.service
 ├──firstboot-graphical.service
 ├──gdm.service
 ├──initial-setup-graphical.service
 ├──livesys-late.service
 ├──livesys.service
 ├──rhnsd.service
 ├──rtkit-daemon.service
 ├──systemd-readahead-collect.service
 ├──systemd-readahead-replay.service
 ├──systemd-update-utmp-runlevel.service
 └──multi-user.target
   ├──atd.service
   ├──auditd.service
   ├──avahi-daemon.service
   ├──brandbot.path
   ├──chronyd.service

...

   │  └──timers.target
   │    └──systemd-tmpfiles-clean.timer
   ├──getty.target
   │  └──getty@tty1.service
   └──remote-fs.target
```

Voir la liste des dépendances pour un service donné :

```
# systemctl list-dependencies httpd.service
httpd.service
 ├──system.slice
 └──basic.target
   ├──firewalld.service
   ├──microcode.service

...
```

Voir la liste des dépendances pour une cible donnée :

```
# systemctl list-dependencies multi-user.target | grep target
multi-user.target
 ├─basic.target
 │ ├─paths.target
 │ ├─slices.target
 │ ├─sockets.target
 │ ├─sysinit.target
 │ │ ├─cryptsetup.target
 │ │ ├─local-fs.target
 │ │ └─swap.target
 │ └─timers.target
 ├─getty.target
 └─remote-fs.target
```

On peut aussi obtenir la liste des services qui ont échoué au boot avec (non documenté dans le manuel) :
`# systemctl --failed`

7 Analyse des journaux - logs

La gestion des log du système est habituellement confiée au daemon rsyslogd (voir http://www.rsyslog.com/doc et man 8 rsyslogd). Systemd propose un remplaçant, journalctl.

Les messages sont habituellement enregistrés dans `/var/log/messages` (voir commande `dmesg`), `journalctl` utilise `/var/log/journal`

Des exemples des commandes de base sont donnés dans la suite.

Afficher tout le journal:

```
journalctl
```

Afficher le journal d'une commande en particulier:

```
journalctl /usr/sbin/sshd
```

```
journalctl $(which sshd)
```

Afficher le journal depuis le dernier boot

```
journalctl -b
```

Afficher le journal depuis une date donnée

```
journalctl --since=today
```

Afficher les 10 derniers événements et attendre les nouveaux
(comme avant avec `tail -f /var/log/messages`)

```
journalctl -f
```

Par défaut, les logs sont enregistrés dans le répertoire `/var/log/journa`l et sont effacés au reboot. Pour les garder en permanence, on peut éditer la configuration ainsi:

```
# mkdir /var/log/journal
# echo "SystemMaxUse=50M" >> /etc/systemd/journald.conf
# systemctl restart systemd-journald
```

Notons qu'il est important de contrôler l'espace disque utilisé par le journal. On peut consulter sa taille avec :

```
journalctl --disk-usage
```

8 Contrôle de groupes de processus

Les "control groups" permettent de gérer ensemble tous les processus associés à un même service.

```
# systemd-cgls
 ...
└─system.slice
  ├─bolt.service
  │ └─1359 /usr/lib/x86_64-linux-gnu/boltd
  ├─apache2.service
  │ ├─647 /usr/sbin/apache2 -k start
  │ ├─649 /usr/sbin/apache2 -k start
  │ └─650 /usr/sbin/apache2 -k start
  ├─vboxadd-service.service
  │ └─1066 /usr/sbin/VBoxService --pidfile
 ...
```

```
# systemd-cgtop
```
À la manière de la commande top

Si on considère le service apache2, l'option status nous indique aussi le CGroup correspondant, attaché au slice system (unité représentant un noeud de l'arbre des CGroups).

```
# systemctl status apache2.service
● apache2.service - The Apache HTTP Server
    Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor
preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since Mon 2019-09-16 12:31:32 CEST; 5h 53min
ago
     Process: 619 ExecStart=/usr/sbin/apachectl start (code=exited,
status=0/SUCCESS)
 Main PID: 647 (apache2)
    Tasks: 55 (limit: 4675)
   CGroup: /system.slice/apache2.service
           ├─647 /usr/sbin/apache2 -k start
           ├─649 /usr/sbin/apache2 -k start
           └─650 /usr/sbin/apache2 -k start

sept. 16 12:31:31 linserv systemd[1]: Starting The Apache HTTP Server...
sept. 16 12:31:32 linserv apachectl[619]: AH00558: apache2: Could not
reliably determine the server's fully qualified domain name, using
127.0.1.1. Set the 'ServerName' directive globally
sept. 16 12:31:32 linserv systemd[1]: Started The Apache HTTP Server.
```

Grâce à cette organisation en CGroup, la commande
```
# systemctl kill apache2
```
termine tous les processus associés à httpd

```
# systemctl show apache2.service
```

Chaque service possède par défaut 1024 CPUShares, on peut augmenter ou diminuer cette quantité.

```
# systemctl show -p CPUShares apache2.service
CPUShares=1024
```

ou

```
systemctl show apache2.service | grep CPUShares
```

```
# systemctl set-property apache2.service CPUShares=500
```

Si on ne veut pas rendre ce changement définitif, il faut ajouter l'option -runtime.

9 Références

Base essentielle de ce document :
RHEL7: How to get started with Systemd - CertDepot
RHEL7: How to get started with CGroups - CertDepot

Control Centre: The systemd Linux init system - by Lennart Poettering, Kay Sievers, Thorsten Leemhuis - The H Open

Support > Product Documentation > Red Hat Enterprise Linux > 7 > System Administrator's Guide - chapter 8. Managing services with systemd - Red Hat, Inc.

How To Use Systemctl to Manage Systemd Services and Units - by Justin Ellingwood - DigitalOcean, Inc.

Systemd Essentials: Working with Services, Units, and the Journal - by Justin Ellingwood - DigitalOcean, Inc.

https://docs.fedoraproject.org/en-US/quick-docs/understanding-and-administering-systemd/index.html

Référence contenant de nombreuses explications utiles mais pas toujours à jour et parfois spécifiques à Fedora :
Fedora-fr - Systemd - par PhilippeMarcovici

Pour les nombreuses références :
systemd System and Service Manager - freedesktop.org

10 Compléments

Wiki ubuntu-fr https://doc.ubuntu-fr.org/systemd

Overview of systemd for RHEL 7
https://access.redhat.com/articles/754933