MODULE 2

Gestion des services

Objectifs de ce module :

- ✓ ***Arrêter et démarrer des services***
- ✓ ***Ajouter ou enlever des services au démarrage***
- ✓ ***Connaître la structure des fichiers de service***
- ✓ ***Utiliser les principales commandes de gestion des services.***

Table des matières

Module 2 - Gestion des services

Sujets ***Page***

Introduction....3
La séquence de démarrage....3
1 Qu'est-ce que systemd ?....4
2 Qu'est-ce qu'un service ?....4
3 Historique....4
4 Quand sont lancés les services ?....5
6 Pourquoi me préoccuper des services ?....5
7 Niveaux d'exécution / cibles....6
8 Quelques informations sur systemd....7
8.1 Les grands principes à la base de systemd....7
8.2 Notion d'unité....7
8.3 Essai de typologie des services....7
Allons plus loin avec les services de type "oneshot"....8
8.4 Emplacement des fichiers de configuration....9
8.5 Analyse du temps de démarrage du système....10
10 Gestion des services....11
10.1 Changer le niveau / la cible d'exécution....11
10.1.1 Changer le niveau / la cible d'exécution immédiatement....11
10.1.2 Changer le niveau / la cible d'exécution par défaut....11
10.2 Lancer un service....12
10.3 Arrêter un service....12
10.4 Activer un service....12
10.5 Désactiver un service....13
10.6 Vérifier l'état d'un service....13
11 États d'un service....14
11.1 États d'un service avec systemd....14

Introduction

Les services permettent de répondre à des requêtes du systèmes face aux demandes des usagers. Ces services utilisent des ports du système et il est donc par conséquent extrêmement important de connaître le numéro des ports utilisés et les services associés à ces ports pour pouvoir gérer adéquatement la sécurité de votre système. Il se peut que vous vouliez rendre accessible un service en particulier (comme un serveur Web par exemple). Il faut donc connaître comment activer ou désactiver de tels services afin de minimiser les problèmes potentiels et aussi de diminuer la charge du serveur en activant des services dont on ne se servira pas.
Il exite plusieurs moyens de gérer l'accès aux services. Nous verrons dans ce module, une manière graphique de le réaliser, les commandes consoles ainsi qu'un utilitaire semi-graphique (ntsysv).

La façon la plus simple de refuser l'accès à un service est tout simplement de le désactiver. Il existe principalement une commande pour gérer les services sous un système, il s'agit de la commande “systemctl”.

La séquence de démarrage

Cette séquence comporte plusieurs étapes dont voici les principales:

Du BIOS jusqu'au Noyau :

- Le BIOS s'initialise.
- Le BIOS charge le MBR (Le Master Boot Record).
- (Assumant que GRUB est installé dans le MBR) Le chargeur initial localise et charge le chargeur secondaire.
- GRUB présente ensuite le menu de démarrage, c'est-à-dire, le choix du système d'exploitation à démarrer s'il y a plus d'un système d'installé.
- GRUB charge ensuite le noyau sélectionné et ensuite le ramdisk si celui-ci est présent.
- GRUB se termine et le noyau prend charge du reste de la séquence.
- Le noyau se décompresse, se charge et active les modules au démarrage.
- Le premier processus est alors lancé, il s'agit de “init”.
- C'est “init” qui continue ensuite le processus de démarrage.

1 Qu'est-ce que systemd ?

systemd est un gestionnaire de service. Il permet de les lancer, les arrêter, les relancer, d'en assurer un suivi, etc.
C'est aussi un gestionnaire de session. Il permet l'utilisation du même matériel en passant d'une session (~loging) à une autre sans que cela pose de problème.
Il permet encore de gérer les points de montage qu'ils soient automatiques ou non.

systemd intègre et va certainement intégrer d'autres aspects. Le présent tutoriel s’intéresse principalement à l'aspect service, qui est le cœur de systemd.

2 Qu'est-ce qu'un service ?

Un service est un programme qui est exécuté en tache de fond. C'est à dire qu'il effectue son travail sans interaction directe avec l'utilisateur.

Certains services sont des composants essentiels du système.

L'autre petit nom d'un service est *daemon*.

3 Historique

Différents système de gestion des services sont utilisés par les distributions GNU/Linux. Par exemple :

- SysVinit (init System V).
- UpStart.
- systemd.

SysVinit est le classique système de démarrage et de gestion des services qui a été utilisé jusqu'à Fedora 8.

UpStart a remplacé SysVinit dans Fedora 9, jusqu'à Fedora 14.

systemd prend maintenant place dans la plupart des distributions dont la distribution Linux Mint.

4 Quand sont lancés les services ?

- Certains services sont lancés par défaut au démarrage du système.
  Par défaut, il s'agit d'un choix des concepteurs de la distribution. Ce choix, comprenant des services essentiels et d'autres qui le sont moins.
- Vous pouvez choisir de désactiver certains services qui ne sont pas essentiels.
  Pourquoi ? Parce que vous n'en avez pas besoin. Ou parce que vous n'en avez pas besoin souvent, voire très rarement ou même jamais.
- Vous pouvez aussi choisir d'arrêter momentanément des services. Par exemple, pour voir ce que ça fait (ou ne fait plus ...), avant de les désactiver complétement.
- Vous pouvez activer des services qui ne sont pas parti par défaut, mais dont vous avez besoin.
- Et enfin, vous pouvez lancer certain service uniquement au besoin.

6 Pourquoi me préoccuper des services ?

Les principales raisons sont :

- La sécurité.
- Optimiser le lancement du système.
- Optimiser le système.

La sécurité ?
Certain services sont à "l'écoute" de sollicitations externes. Toutes sollicitations externes peut venir d'une personne malveillante. Et donc, moins il y a de services de ce type en exécution, moins il y a de portes d'entrées potentielles pour ces personnes.

Optimiser le lancement du système ?
Plus il y a de services qui sont lancés au démarrage, plus cela prend du temps pour que le système deviennent fonctionnel.
Il s'agit d'ailleurs d'une des principales motivation de la naissance de Upstart et systemd.

Optimiser le système ?
Les concepteurs d'une distribution vont, généralement, essayer de faire en sorte que la distribution par défaut puisse convenir au plus grand nombre. Par exemple, pour cela, il faut être capable de gérer différents matériels. Ces matériels, peuvent être présents ou pas sur la machine de chaque utilisateur. Et le service associé, peut être utile ou pas.

7 Niveaux d'exécution / cibles

SysVinit et Upstart ont une notion de niveau d'exécution (RunLevel).
systemd a une notion de cible (target).

Sans enter plus dans les détails, à un niveau d'exécution ou une cible correspond un mode de fonctionnement du système où certains services sont arrêtés tandis que d'autres sont lancés.

Le tableau ci-dessous décrits les différents modes de fonctionnement standard.

Les modes de fonctionnements		
Niveaux d'exécution	**Cibles systemd**	**Commentaires**
0	runlevel0.target, poweroff.target	Arrêt du système
1,s, single	runlevel1.target, rescue.target	Mode utilisateur unique / mode maintenance (Single User Mode)
3	runlevel3.target, multi-user.target	Mode multi-utilisateur non graphique
2, 4	runlevel2.target, runlevel4.target multi-user.target	Ces modes sont identiques au mode multi-utilisateur non graphique. Ces modes peuvent être adaptés aux besoins
5	runlevel5.target, graphical.target	Mode multi-utilisateur graphique. Il s'agit du mode par défaut.
6	runlevel6.target, reboot.target	Redémarrage du système

8 Quelques informations sur systemd

8.1 Les grands principes à la base de systemd

- Le grand principe de systemd est de créer les canaux de communication entre les processus avant de lancer les processus des services.

- La synchronisation effectuée par le système permet de gérer une partie des dépendances.

Par exemple, si un service utilise une socket pour recevoir les requêtes à traiter, si la socket est créée par systemd les clients de ce services peuvent accéder à la socket avant que le service soit effectivement lancé et donc y écrirent leurs requêtes sans être en erreur par absence de la socket.

- L'autre mécanisme de communication inter-processus est DBUS. DBUS utilisant des sockets ...

8.2 Notion d'unité

- systemd utilise une notion d'unité ("Unit"). Il y a différentes type d'unités : automount, path, mount, service, socket, target, timer, wants.

Pour notre explication sur les services, nous retenons les unités de type socket et service.

- Les unités de type socket permettent à systemd de connaitre les socket à créer.

- Les unités de type service permettent à systemd de connaitre les services à lancer. Bien entendu, les socket sont traitées par systemd avant les services.

8.3 Essai de typologie des services

Type de service

- L'un des paramètres de configuration d'un service avec systemd est "Type=" avec comme valeur possible : simple, forking, oneshot, dbus, notify.

- Un service de type "simple" (type par défaut) lance un processus principal. Dans le cas où ce processus offre une fonctionnalité à d'autre processus sur le système, ses canaux de communication doivent être installés avant que le processus principal soit

démarré. Ce qui est rendu possible par l'activation des sockets, et autres canaux de communication, tel qu'indiqué ci-dessus. Ainsi, systemd peut traiter les autres unités sans se préoccuper de la fin du lancement d'une unité de type "simple".

- Un service de type "forking", lance un processus père qui créera un processus fils dans le cadre de son démarrage. Le processus parent est prévu pour s'arrêter une fois le démarrage complet et que tous les canaux de communication sont mis en place. Le processus fils continue à fonctionner en tant que le processus principal du service. systemd poursuit le traitement des autres unités une fois que le processus père se termine.

- Un service de type "oneshot" est similaire à un service de type "simple". Cependant, systemd attend que le processus se termine avant de continuer ses traitements.

Allons plus loin avec les services de type "oneshot"

- Différents paramètres peuvent être utilisé dans la configuration des services. Nous allons en présenter quelques uns dans la perspective des services de type "oneshot".

- ExecStart permet d'indiquer la commande à exécuter au lancement du service. Ce paramètre est obligatoire pour tout les types de service.
- ExecStop permet d'indiquer une commande a exécuter pour arrêter le service. Ce paramètre est facultatif.

- RemainAfterExit à la valeur "yes" permet d'indiquer que quand la commande de lancement (ExecStart) est terminée, le service est considéré comme toujours lancé. Ce paramètre est très utile pour les services de type "oneshot" qui exécutent une commande à leur lancement (ExecStart) sans qu'il y ait un processus spécifique qui reste en exécution.

Un exemple est le service iptables. Ci-dessous, je donne un extrait de son fichier de configuration :

```
[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/libexec/iptables.init start
ExecStop=/usr/libexec/iptables.init stop
```

Nous pouvons voir que :

- A son lancement, la commande /usr/libexec/iptables.init start est exécutée. Cette commande va permettre de charger des modules iptables et les règles iptables que le modules netfilter du noyau linux va prendre en compte.

- A son arrêt, la commande /usr/libexec/iptables.init stop est exécutée. Cette commande va permettre de remettre en place la politique de traitements iptables par défaut et de décharger les modules iptables.

- Le service iptables continuera d'apparaitre comme actif après la fin de la commande /usr/libexec/iptables.init start et jusqu'à ce qu'il soit explicitement arrêté.

Remarquez qu'il y a des services de type oneshot, avec l'option ExecStart et RemainAfterExit sans option ExecStop. C'est juste qu'il n'y a pas de commande déclenchée à l'arrêt du service.

8.4 Emplacement des fichiers de configuration

- En plus des types de service, les fichiers de configuration des services permettent d'indiquer à systemd d'autres dépendances explicites à d'autres unités.

- La configuration des services est effectuées dans le répertoire /lib/systemd/system avec des fichiers d'extensions ".service". Il s'agit de l'emplacement standard.

- Il est possible d'utiliser le répertoire /etc/systemd/system pour effectuer des modifications sans risque quelles soient perdues suite à une mise à jour du système

Pour cela, il suffit de créer le fichier /etc/systemd/system/XXX.service avec le contenu du fichier /lib/systemd/system/xxx.service, puis de modifier à votre convenance ce nouveau fichier.

8.5 Analyse du temps de démarrage du système

- La commande systemd-analyze blame, permet de lister le temps de démarrage des différents services. Il est ainsi possible de repérer les services très ou trop long à se lancer.

Par exemple :

```
$ systemd-analyze blame
          6207ms udev-settle.service
          5228ms cryptsetup@luks\x2d9899b85d\x2df790\x2d4d2a.service
           735ms NetworkManager.service
           642ms avahi-daemon.service
           600ms abrtd.service
           517ms rtkit-daemon.service
           478ms fedora-storage-init.service
           396ms dbus.service
           390ms rpcidmapd.service
           346ms systemd-tmpfiles-setup.service
           322ms fedora-sysinit-unhack.service
           316ms cups.service
           310ms console-kit-log-system-start.service
           309ms libvirtd.service
           303ms rpcbind.service
           298ms ksmtuned.service
           288ms lvm2-monitor.service
           281ms rpcgssd.service
           277ms sshd.service
           276ms livesys.service
           267ms iscsid.service
           236ms mdmonitor.service
           234ms nfslock.service
           223ms ksm.service
           218ms mcelog.service
         ...
```

Et vous obtenez la liste des services lancés au démarrage classés de celui qui a pris le plus de temps à celui en a pris le moins.

10 Gestion des services

Dans ce chapitre, nous allons voir comment :

- Changer de niveau / cibles d'exécution.
- Lancer un service.
- Arrêter un service.
- Activer un service.
- Désactiver un service.
- Vérifier l'état d'un service.

tout ceci en ligne de commande.

10.1 Changer le niveau / la cible d'exécution

10.1.1 Changer le niveau / la cible d'exécution immédiatement

systemctl isolate CIBLE.target

Par exemple :

- systemctl isolate multi-user.target ou systemctl isolate runlevel3.target, pour passer en mode multi-utilisateur non graphique.
- systemctl isolate graphical.target ou systemctl isolate runlevel5.target, pour passer en mode multi-utilisateur graphique.

10.1.2 Changer le niveau / la cible d'exécution par défaut

systemctl set-default CIBLE.target

Par exemple :

- ln -sf /lib/systemd/system/multi-user.target /etc/systemd/system/default.target, pour passer en mode multi-utilisateur non graphique par défaut.
- ln -sf /lib/systemd/system/graphical.target /etc/systemd/system/default.target, pour passer en mode multi-utilisateur graphique par défaut.

10.2 Lancer un service

systemctl start NOM.service

Par exemple :

- systemctl start network.service, pour lancer le service network.

Lancement versus activation
Le lancement d'un service n'est pas suffisant pour que celui-ci soit redémarré au prochaine lancement du système. Il est nécessaire que celui-ci soit activé, cf. ci-dessous.

10.3 Arrêter un service

systemctl stop NOM.service

Par exemple :

- systemctl stop network.service, pour arrêter le service network.

Arrêt versus désactivation
L'arrêt d'un service n'est pas suffisant pour que celui-ci soit pas redémarré au prochaine lancement du système. Il est nécessaire que le service soit désactivé, cf. ci-dessous.

10.4 Activer un service

systemctl enable NOM.service

Par exemple :

- systemctl enable network.service, pour activer le service network.

Activation versus lancement
L'activation d'un service permet uniquement de le lancer, par défaut, au démarrage du système. Pour le lancer immédiatement, cf. ci-dessus.

10.5 Désactiver un service

systemctl disable NOM.service

Par exemple :

- systemctl disable network.service, pour désactiver le service network.

Désactivation versus arrêt
La désactivation d'un service permet uniquement de ne pas le lancer, par défaut, au démarrage du système. Pour l'arrêter immédiatement, cf. ci-dessus.

10.6 Vérifier l'état d'un service

systemctl is-enabled NOM.service

et

systemctl status NOM.service

Par exemple :

- systemctl is-enabled network.service, pour vérifier l'état d'activation du service network.
- systemctl status network.service, pour vérifier l'état de lancement du service network.

11 États d'un service

11.1 États d'un service avec systemd

- La commande systemctl status NOM.service permet d'afficher plusieurs lignes d'informations sur un service. Ces informations concernent l'état courant du service. Nous nous intéresserons aux lignes préfixées "Loaded" et "Active".

Quelques exemples :

systemctl status iptables.service

iptables.service - IPv4 firewall with iptables
Loaded: loaded (/lib/systemd/system/iptables.service; enabled)
Active: active (exited) since Sat, 10 Mar 2012 20:38:28 +0100; 1h 5min ago
Process: 1060 ExecStart=/usr/libexec/iptables.init start (code=exited, status=0/SUCCESS)
CGroup: name=systemd:/system/iptables.service

systemctl status NetworkManager.service
NetworkManager.service - Network Manager
Loaded: loaded (/lib/systemd/system/NetworkManager.service; disabled)
Active: inactive (dead)
CGroup: name=systemd:/system/NetworkManager.service

network.service - LSB: Bring up/down networking
Loaded: loaded (/etc/rc.d/init.d/network)
Active: active (running) since Sat, 10 Mar 2012 20:38:32 +0100; 1h 8min ago
Process: 1170 ExecStart=/etc/rc.d/init.d/network start (code=exited, status=0/SUCCESS)
CGroup: name=systemd:/system/network.service
└ 1689 /sbin/dhclient -H hostname -1 -q -lf /var/lib/dhclient/dhclient-em1.leases -pf /var/run/dhclient-em1.pid em1

La ligne préfixée "Loaded" permet d'indiquer l'état de chargement du service:

Si le service est configuré, avec le mot clé "loaded", suivi du chemin du fichier de configuration.
Si le service est activé, avec le mot clé "enabled" en fin de ligne.
Si le service est déactivé, avec le mot clé "disabled" en fin de ligne.
Si le service n'est pas configuré ou s'il est en erreur, le mot clé "error" est présent,

suivi de la raison.

La ligne préfixée "Active" permet d'indiquer l'état d'exécution du service :

Si le service a été lancé avec la mot clé "active".
Si le service n'a pas été lancé avec le mot clé "inactive".
Une deuxième indication, donnée entre parenthèse, permet de préciser l'état du processus en cours pour le service. Avec "runnning" si ce processus est en cours d'exécution, "exited" si ce processus est terminé, et "dead" si ce processus est en erreur ou s'il n'a pas été lancé.