ADMINISTRATION UNIX

GESTION DES PACKAGES

PACKAGE

- Les packages sont des paquets de logiciels et de métadonnées
  - Les métadonnées précisent :
    - Informations sur le package (telles que le nom complet du logiciel, la description de son but, le numéro de version, le fournisseur)
    - Liste des dépendances nécessaires au bon fonctionnement du logiciel
  - Le logiciel peut être à la fois sous forme source et binaire
- Un système de gestion de package est un ensemble d'outils permettant d'automatiser le processus d'installation, de mise à niveau, de configuration et de suppression des package logiciels d'un ordinateur.
- Les gestionnaires de packages offrent généralement aux utilisateurs la possibilité d'installer et de mettre à niveau des logiciels sur le réseau d'une manière intégrée et transparente.

PACKAGE

- Gérer les dépendances
  - Quelles sont les conditions préalables des packages ?
  - Est-ce qu'il remplace un autre package ?
- Gérer les conflits
  - Que se passe-t-il si deux programmes installent ou modifient le même fichier ?
- Gérer les mises à niveau
  - Que se passe-t-il si l'utilisateur a une configuration personnalisée ?
  - Que se passe-t-il si les propriétaires/permissions de fichiers ont changé ?
  - Que faire si l'utilisateur a besoin de l'ancienne et de la nouvelle version en même temps ?

PACKAGE

- Il n'y a pas de gestionnaire de package standard dans Linux
- Différentes distributions adoptent différents systèmes
  - Tarball-based (.tgz)
  - RPM (RedHat Package Manager) (.rpm)
  - DEB (Debian Package Manager) (.deb)
  - Solaris packages (.pkg)
  - …

TARBALL

- Les fichiers Tarball sont l'ancienne façon de distribuer les logiciels sous Linux/Unix.

- Compatible avec toutes les distributions

RPM

- RPM (RedHat Package Manager) (.rpm)
  - Initialement introduit par RedHat
  - Maintenant adopté par de nombreuses autres distributions (Fedora, …)
  - RPM est le format de package de base de la base standard Linux

- Normalement utilisé pour distribuer des logiciels binaires pré-packagés
- Il peut garder une trace des dépendances des packages
- Commande rpm

DEBIAN PACKAGE

- DEB (Debian Package Manager) (.deb)
  - Introduit par la distribution Debian (et maintenant utilisé par toutes les distributions de type Debian comme Ubuntu)
  - Debian a été la première distribution Linux à s'appuyer sur les interdépendances des packages pour effectuer des mises à niveau fiables du système
  - C'est maintenant la deuxième distribution la plus ancienne encore en vie (le slackware mène de quelques semaines)
- Commande deb

COMMANDES DE BASE

- **Installer un package**
  - > dpkg -i file.deb
  - > rpm -i file.rpm
- **Mettre à jour un package**
  - > dpkg -i file.deb
  - > rpm -U file.rpm
- **Supprimer un package**
  - > dpkg -r package_name

  remove (-r) ne supprime pas le fichier de configuration (pour une réinstallation ultérieure)
  - > dpkg -P package_name

  purge (-P) supprime tout
  - > rpm -e [--nodeps] package_name

  --nodepts force le système à supprimer un package même si d'autres logiciels en dépendent

QUERYING

- dpkg –l OU rpm –qa
  - Liste tous les packages installés dans le système
- dpkg -L pkg_name OU rpm -q -l pkg_name
  - Liste tous les fichiers installés par un package particulier
- dpkg -S filename OU rpm -qf filename
  - Affiche le package qui a installé un certain fichier dans le système
- dpkg -p pkg_name OU rpm -qi pkg_name
  - Imprime les détails d'un package particulier

RPM - APERÇU

- Extraire les dépendances des packages
  - rpm -qpR file.rpm
- Extraire les informations sur le package
  - rpm -qip file.rpm
- Extraire les scripts d'installation et de désinstallation des packages
  - rpm -qp --scripts file.rpm
- Extraire les fichiers de package
  - rpm2cpio package.rpm > package.cpio
  - cpio -idmv < package.cpio

RPM - VÉRIFICATION

- Vérifie les fichiers actuels avec le RPM original
  - taille
  - checksum MD5
  - permissions, type
  - propriétaire
  - groupe
- Syntaxe de base
  - rpm -V nom_package
- options
  - f <file> vérifie le fichier file
  - a vérifie tous les packages
  - p <package-file> vérifie par rapport au RPM original

DEB - APERCU

- Les fichiers .deb sont de pures archives *ar*
- Le contenu d'un fichier deb peut être extrait par:
  - ar vx file.deb
- L'archive contient trois fichiers:
  - debian-binary → version du format de fichier deb (2.0)
  - control.tar.gz → contient des fichiers spéciaux comme la description du package, les fichiers MD5, les scripts de pré et post-installation.
  - data.tar.gz → tous les fichiers à installer avec leurs chemins de destination

APT

- APT est un ensemble d'outils (*apt-get* et *apt-cache* sont les plus importants) pour récupérer, installer et gérer les relations (notamment les dépendances) entre les packages.

- APT fait le tri de la liste des packages à installer ou à supprimer et appelle dpkg dans la meilleure séquence possible.

- Fonctionnalités de base :
  - > apt-cache search keywords
  - > apt-cache show package_name
  - > apt-get install package_name
  - > apt-get [--purge] remove package_name

APT

- **apt-get update**
  - Resynchroniser les fichiers d'index des packages locaux à partir de leurs sources. Les index des packages disponibles sont récupérés à partir des emplacements spécifiés dans /etc/apt/sources.list

- **apt-get upgrade**
  - Mettre à jour tous les packages installés avec la dernière version disponible. En aucun cas les packages actuellement installés ne sont supprimés ou de nouveaux packages installés dans le système.

- **apt-get dist-upgrade**
  - Mettre à jour tous les packages et en plus décider intelligemment quels nouveaux packages doivent être installés et lesquels doivent être supprimés.

GESTION DES PROCESSUS

GESTION DES PROCESSUS : INTÉRÊT

- L'administrateur se doit de surveiller l'utilisation des principales ressources du système : processeur, mémoire centrale, pagination ou swapping, et les entrées/ sorties

- Il doit être capable de remédier à leur mauvaise utilisation ou à leur insuffisance.

GESTION DES PROCESSUS

Définition d'un processus :

- Un processus est un programme en cours d'exécution. Un processus sous Linux est identifié par un numéro unique qui s'appelle le numéro d'identification du processus PID (Process IDentifier) et qui lui est attribué par le système à sa création.
- **Processus père et processus fils**
- Le processus fils est un processus qui a été créé par un autre qui prend le nom de père.

GESTION DES PROCESSUS ET CONTRÔLE DES JOBS

Il existe deux types de processus. Les *process*, et les *jobs*:

- Le *process* est un petit programme (ou parfois un gros) qui s'exécute sous les ordres de son *process* parent, qui lui-même s'exécute sous les ordres de son parent, etc.
- Le *job* est un *process* qui est le descendant uniquement d'un **shell** (bash pour GNU/Linux).

Par exemple

Vous exécutez la commande **date** dans un **shell** UNIX. La commande date engendre un *process*, ou plus précisément un *job.*

À son exécution, un processus se voit attribuer un numéro qui lui sera propre pour son exécution. Si le processus redémarre, il reçoit un nouveau numéro. Ce numéro est appelé ***PID***.

LA GESTION DES PROCESSUS

- Un processus devient tour à tour actif/inactif tout au long de sa vie; il passe donc par différents états, notamment:

  - *Élu*: le processus est en cours d'exécution.
  - *Prêt*: il dispose de toutes les ressources nécessaires à son fonctionnement.
  - *Bloqué*: attente d'un événement pour pouvoir continuer.
  - *Zombi*: le processus a terminé son exécution et il est prêt à mourir; à ce stade, l'Ordonnanceur a détecté l'arrêt de la tâche mais le père du processus n'a pas encore pris en compte sa mort; il existe donc toujours une entrée pour le processus dans la table des processus.

LES COMMANDES DE SURVEILLANCE

- Les commandes de surveillance du système sont nombreuses et nous mentionnons ici les plus utilisées sur Linux.

Les commandes

- **uptime**: La commande uptime affiche des statistiques sur la charge du système durant la dernière, les cinq et les quinze dernières minutes.
  - La charge du processeur est le nombre de processus en train d'utiliser le(s) processeur(s) ou en train d'attendre de pouvoir les utiliser

- **vmstat**: La commande vmstat affiche des statistiques sur les ressources du système, dont les processus, la mémoire, l'activité du CPU.

GESTION DES PROCESSUS ET CONTRÔLE DES JOBS

La commande ps

- La commande ps affiche les informations sur les *jobs*, dépendant uniquement du shell
- taper ps dans la console, voici un résultat possible:

```
PID    TTY          TIME  CMD
2571 ttys000    0:00.01 -bash
```

- *TTY* est l'abréviation de terminal dans le langage UNIX. Cette colonne indique de quel terminal le *job* est issu

Voici quelques options intéressantes de la commande `ps`.

L'option	L'utilisation
-a	Affiche tous les *jobs* exécutés par l'utilisateur et le système
-x	Affiche *jobs* et *process* confondus
-A	Affiche tous les *process* exécutés par l'utilisateur et le système (équivalent à `ps -ax`)
-j	Affiche plus de caractéristiques sur le processus, comme l'état (colonne *STAT*), ou l'utilisateur qui exécute le processus
-m	Affiche les processus par ordre d'utilisation de la mémoire, à la place du *PID*
-r	Affiche les processus par ordre d'utilisation du processeur
-u nom_utilisateur	Donne la liste des processus associés à un utilisateur.

LES COMMANDES DE SURVEILLANCE

- Les commandes

  - **free**: La commande affiche la quantité totale de mémoire vive et de swap ainsi que la quantité de mémoire libre. Par défaut, les résultats sont affichés en Ko.

  - **pstree**: La commande pstree affiche l'arbre des processus. L'arbre est construit d'après la filiation des processus. L'option -p est utile pour visualiser les PIDs des processus

  - **top**: La commande top est un logiciel libre qui affiche en permanence des informations générales sur les processus et les ressources. Par défaut les processus les plus gourmands en temps de calcul sont placés en tête.

LES COMMANDES DE SURVEILLANCE

- Les commandes
  - **w**: La commande w affiche les processus des utilisateurs connectés
  - **time**: La commande time affiche le temps CPU consommé par un processus :
    - Le temps réel écoulé entre sa création et sa mort.
    - Le temps consommé pour exécuter des instructions en mode « user », les instructions du processus, les calculs par exemple.
    - Le temps consommé pour exécuter des instructions en mode kernel, les primitives du noyau.

    La commande time de Linux affiche d'autres informations sur l'utilisation de la mémoire
  - **netstat**: La commande netstat affiche des informations sur l'utilisation du réseau.

LES COMMANDES DE GESTION DES PROCESSUS

- **kill:** envoie un signal à un (des) processus. Il est possible d'envoyer des signaux à un processus afin d'en modifier l'exécution.
  - **HUP** signal 1 Le signal Hang Up peut être interprété par certains processus comme un signal de fin d'exécution ou comme un signal indiquant que le processus doit relire son fichier de configuration.
  - **TERM** signal 15 Le signal Terminate indique à un processus qu'il doit s'arrêter.
  - **KILL** signal 9 Le signal Kill indique au système qu'il doit arrêter un processus qui ne répond plus.
- **killall:** La commande killall envoie un signal à tous les processus qui exécutent une commande spécifique.
- **lsof:** La commande lsof permet de lister les fichiers ouverts et de connaître les applications qui y accèdent

AUTOMATISATION DE TÂCHES

- Effectuer de tâches à des temps différés:
  - Ponctuellement (commande *at*)
  - De manière répétitive (commande *crontab*)

Ces tâches peuvent de toute nature:

(sauvegarde, ...) et sont souvent décrites dans des scripts shell.

LA COMMANDE AT

- La commande at exécute une commande à un moment donnée.
- Exemple:

La création du répertoire « docs » dans votre répertoire à 15h30

$ at 1530

at> mkdir /home/<moncompte>/docs

at> <Ctrl^D>

- Pour exécuter la commande demain à 14h17

```
$ at 14:17 tomorrow
```

- Pour exécuter la commande le 15 novembre à 14h17

```
$ at 14:17 11/15/08
```

EXÉCUTER UNE COMMANDE APRÈS UN DÉLAI

- pour exécuter la commande dans 5 minutes :

```
$ at now +5 minutes
```

ce qui signifie "Dans maintenant (now) + 5 minutes". Les mots-clés utilisables sont les suivants :

- minutes
- hours (heures)
- days (jours)
- weeks (semaines)
- months (mois)
- years (années)

exécutera les commandes dans 2 semaines

```
$ at now +2 weeks
```

ATQ ET ATRM : LISTER ET SUPPRIMER LES JOBS EN ATTENTE

- ***atq***

Visualisation de ses tâches en attente ou de toutes les tâches en attente pour l'administrateur.

```
$ atq
13          Mon Nov 10 14:44:00
12          Mon Nov 10 14:42:00
```

- ***atrm num***

Destruction d'une tâche en attente (num est le numéro de la tâche à supprimer obtenu grâce à la commande atq).

```
$ atrm 13
```

CRONTAB

- crontab est une commande qui permet de lire et de modifier un fichier appelé la « crontab ».
- crontab permet donc de changer la liste des programmes régulièrement exécutés.
- Les fichiers crontabs des utilisateurs sont enregistrés dans:
  - /var/spool/cron/username
  - le fichier crontab système est /etc/crontab

- le programme cron qui se charge d'exécuter ces programmes aux heures demandées.
- Il y a trois paramètres différents à connaître :
  - -e : modifier la crontab ;
  - -l : afficher la crontab actuelle ;
  - -r : supprimer votre crontab. Attention, la suppression est immédiate et sans confirmation !

CRONTAB

- Pour configurer une nouvelle tache ou en supprimer une, il faut ouvrir cron en édition crontab -e puis ajouter une ligne pour sa tache qui se présente sous cette forme :
- "minute" "heure" "jour du mois" "mois" "jour de la semaine" "utilisateur" "action« =

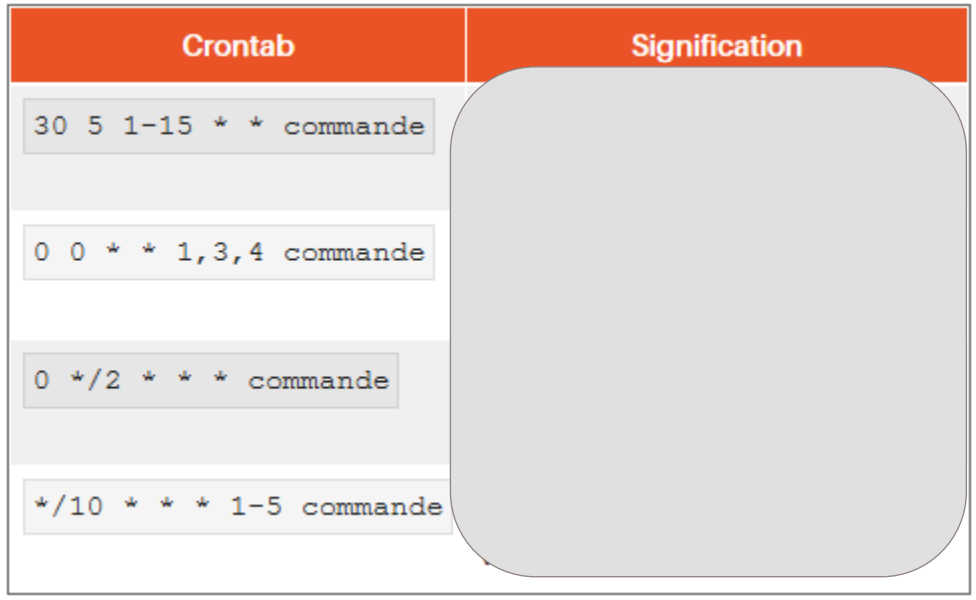
  mm hh jj MMM JJJ tâche

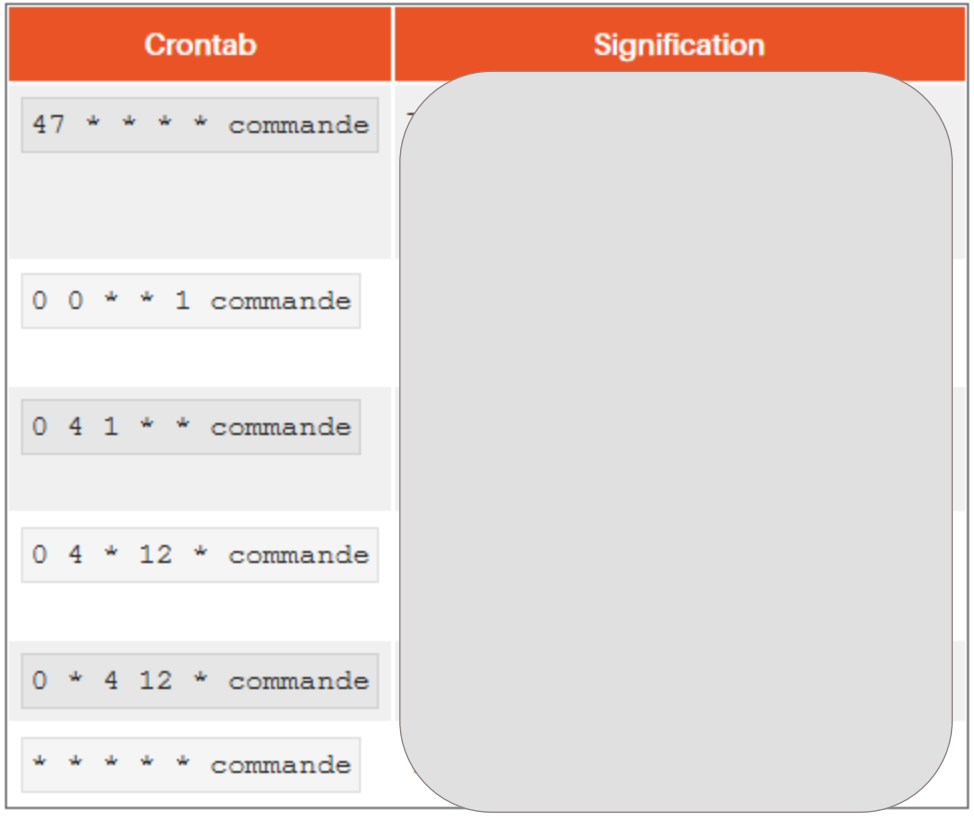
- mm représente les minutes (de 0 à 59)
- hh représente l'heure (de 0 à 23)
- jj représente le numéro du jour du mois (de 1 à 31)
- MMM représente l'abréviation du nom du mois (jan, feb, ...) ou bien le numéro du mois (de 1 à 12)
- JJJ représente l'abréviation du nom du jour ou bien le numéro du jour dans la semaine :

  0 = dimanche
  1 = lundi
  2 = mardi
  …
  6 = samedi
  7 = dimanche

CRONTAB

- **Les différentes notations possibles**
- Pour chaque champ, on a le droit à différentes notations :
- 5 (un nombre) : exécuté lorsque le champ prend la valeur 5 ;
- * : exécuté tout le temps (toutes les valeurs sont bonnes) ;
- 3,5,10 : exécuté lorsque le champ prend la valeur 3, 5 ou 10. Ne pas mettre d'espace après la virgule ;
- 3-7 : exécuté pour les valeurs 3 à 7 ;
- */3 : exécuté tous les multiples de 3 (par exemple à 0 h, 3 h, 6 h, 9 h…).





TÂCHES EN ARRIÈRE-PLAN :

- Linux attend la fin de l'exécution de la commande en cours d'exécution avant de permettre à l'utilisateur de relancer une nouvelle commande.
- Lorsqu'une commande ne nécessite pas de dialoguer avec l'utilisateur et que sa durée d'exécution est importante, il est possible de l'exécuter en **arrière-plan** en ajoutant le caractère spécial **&** à la fin de la commande.
- Le système Linux lance alors la commande et redonne immédiatement la main à l'utilisateur pour d'autres travaux.
- Le lancement de commandes en arrière-plan permet à un seul utilisateur de lancer plusieurs tâches à partir d'un même terminal. Cela correspond à l'aspect multitâche du système d'exploitation Linux.
- Ces processus en arrière-plan ne peuvent plus être interrompus directement à partir de la console à l'aide de la touche <ctrl c>. Pour les interrompre il faut :

    - soit sortir de sa session, dans ce cas tous les processus attachés à la session seront interrompus,
    - soit rechercher le numéro du processus et lui envoyer un signal à l'aide de la commande ***kill***.

TÂCHES EN ARRIÈRE-PLAN :

- Lors du lancement d'une commande en arrière-plan, l'interpréteur de commandes Bash attribue un numéro croissant pour chacune des commandes lancées : le *numéro job*.
- La liste des travaux en arrière-plan, avec leur numéro, est obtenue à l'aide de la commande *jobs*.
- Cette commande indique si le processus en arrièreplan est en cours d'exécution (running) ou suspendu (stopped).

Action	État initial du processus	État final du processus
| xstra> **commande**		processus en avant-plan
| xstra> **commande &**		processus en arrière-plan
| xstra> **fg %numéro_job**	processus en arrière-plan	processus en avant-plan
| **<ctrl z>**	processus en avant-plan	processus stoppé (suspendu)
| xstra> **fg %numéro_job**	processus stoppé	processus en avant-plan
| xstra> **bg %numéro_job**	processus stoppé	processus en arrière-plan
• **kill –STOP %numéro_job** • soit tentative lecture sur le terminal • soit tentative d'écriture sur le terminal (il faut avoir positionné stty tostop)	processus en arrière-plan	processus stoppé

GESTION DES SERVICES

GESTION DES SERVICES

Le répertoire /etc/rc.d

- Ce répertoire contient les scripts utilisés pour l'initialisation du système. Ils sont prévus pour démarrer les différents services et processus et effectuer quelques vérifications de configuration.

DÉMARRER ET ARRÊTER LES SERVICES MANUELLEMENT

- Les scripts dans init.d peuvent être utiliser pour démarrer et arrêter les services manuellement
- la commande service fait appel à ce script
- Syntaxe : **service nom_service option**
- Options:
  - Status : statut du service
  - Start : démarre le service
  - Stop : arrête le service
  - Restart : arrête et redémarre le service

L'ARRÊT DU SYSTÈME

- Ne pas arrêter brutalement le système.
  - La procédure d'arrêt permet :
  - d'avertir les utilisateurs que le système doit être arrêté
  - de demander aux applications de s'arrêter et de fermer les connexions et les fichiers ouverts
  - de passer le système en mode mono-utilisateur
  - de vider les tampons mémoire du cache disque
- Le système garde une trace du fait qu'il est démarré pour permettre une vérification d'intégrité dans le cas d'un arrêt brutal.
- La commande d'arrêt du système est la commande **shutdown** qui permet, selon les options utilisées :
  - de donner l'heure de l'arrêt (now, hh:mm, +minutes)
  - de donner le mode arrêt (arrêt ou redémarrage)

Exemple : pour redémarrer shutdown -r now ou reboot

- pour arrêter shutdown -h now ou halt

PROGRAMMATION PÉRIODIQUE AVEC SYSTEMD

Systemd (pour « system daemon » : le démon du système) est le processus qui gère tous les services (on parle de processus init)

Le processus init a un status particulier.

- Il est le tout premier processus appelé par le kernel (PID = 1).
- Il est le parent de tous les processus qui deviennent orphelins.
- Il reste vivant jusqu'à l'arrêt du système.
- A l'arrêt du système, il est chargé de fermer tous les services et de quitter proprement.

Les différents processus init

- Il existe différents processus init, dont les plus connus sont:
  - SysVinit: processus init historique, hérité de UNIX System V.
  - Upstart: sorti en 2006, à l'origine pour Ubuntu.
  - Systemd: sorti en 2010.
- systemd se manipule grâce à la commande **systemctl**.

PROGRAMMATION PÉRIODIQUE AVEC SYSTEMD

Démarrer, arrêter un service

- Dans les exemples ci-dessous, remplacez application par le nom de votre service : par exemple, mariadb, httpd, firewalld, nfsd etc.

Pour démarrer le service :

```
systemctl start application.service
```

Pour arrêter le service :

```
systemctl stop application.service
```

Pour redémarrer un service qui est lancé, faites :

```
sudo systemctl restart application.service
```

Pour recharger les fichiers de configuration d'un service dans le redémarrer (typiquement, le serveur web Apache) , faites :

```
sudo systemctl reload application.service
```

Pour que le service soit lancé au démarrage du système :

```
systemctl enable application.service
```

Pour que le service ne soit pas lancé au démarrage du système :

```
systemctl disable application.service
```

Pour empêcher l'activation d'un service (par exemple on masquera httpd car on veut utiliser nginx et donc éviter que l'admin système démarre httpd par inadvertance) :

```
systemctl mask application.service
```

Pour envoyer un signal d'arrêt (SIGTERM) à tous les processus du service (plus élégant qu'un `killall` qui tue en fonction d'une chaine de caractère):

```
systemctl kill application.service
```

PROGRAMMATION PÉRIODIQUE AVEC SYSTEMD

Information sur un service

Pour voir tous les services disponibles et leur statut, y compris les services de statut inactif :

```
systemctl list-unit-files --type=service --all
```

Pour vérifier si le service est démarré, arrêté, afficher des informations pour le niveau d'exécution en cours :

```
systemctl status application.service
```

Pour voir si le service est actuellement démarré :

```
systemctl is-active application.service
```

Pour vérifier si le service sera démarré au démarrage du système :

```
systemctl is-enabled application.service
```

Pou vérifier qu'il y a eu un problème lors du démarrage d'un service :

```
systemctl is-failed application.service
```

Pour voir tous les services qui ont un problème :

```
systemctl --failed --type=service
```

Évidemment les commandes Unix classiques vous permettent aussi de comprendre ce qui se passe, vous pouvez par exemple lister les processus avec :

```
ps aux
```