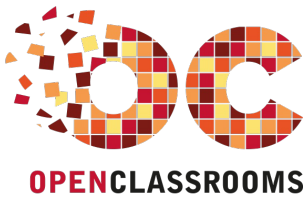
Faire un Demon sous Linux

Par Gabriel JUCHAULT (Extaze)



www.openclassrooms.com

Dernière mise à jour le 25/05/2011

Sommaire

Faire un Demon sous Linux

Par  Gabriel JUCHAULT (Extaze)

Mise à jour : 25/05/2011
Difficulté : Facile

Bonjour à tous et à toutes, chers linuxiens.
Vous est-il déjà arrivé de vouloir automatiser toutes vos tâches ? J'entends par là qu'au démarrage même de la machine, vous verriez tous vos scripts se lancer, automatiquement, sans avoir à lancer la session et se connecter en root pour enfin exécuter son script… ça ne vous tenterait pas ?
Eh bien c'est possible grâce aux ***demons*** (appelés « services » sous Windows).

Citation : Wikipédia

> [Un demon] désigne un type de programme informatique, un processus qui s'exécute en arrière-plan plutôt que sous le contrôle direct d'un utilisateur.

Demon signifie : Disk and Execution MONitor (on peut utiliser aussi l'écriture Daemon pour Disk And Execution MONitor).

Il est nécessaire d'avoir lu le tutoriel sur Linux de M@teo21.
Sommaire du tutoriel :

Pourquoi réaliser un demon ? Quels sont les pré-requis ?

Pourquoi faire un demon ?

Évidemment, on pourrait se demander pourquoi faire un demon.
Imaginez simplement que vous avez un tout petit serveur pour votre famille, pour vos amis, ou même un serveur de communication vocale. Ce serveur tourne sous Linux et à chaque fois que vous démarrez le serveur, il faut entrer login, mot de passe, puis lancer le terminal, s'identifier en tant que `root` et — enfin — lancer le script de démarrage.
Eh bien le demon permet de lancer un script avec toutes les autorisations nécessaires, avant d'ouvrir sa session.

Donc, j'aurai juste à démarrer l'ordinateur, et mon serveur se lancera tout seul ?

Oui.

Quels sont les pré-requis ?

Tout d'abord, il faut le script d'exécution. C'est un programme qui se charge de démarrer le serveur. Exemple :

Code : Bash

```
#!/bin/bash
/etc/init.d/apache start
```

Attention, le `/etc/init.d/apache start` est à titre informatif, apache s'enregistre à l'installation dans le init.d !

Il faut ensuite l'accès au compte `root`.

Et enfin, il faut deux-trois connaissances.
La plupart des serveurs demandent des accès au compte `root`, et le plus souvent vous n'avez pas besoin d'être derrière tout le temps ; vous le démarrez et le réduisez en tâche de fond.

Dans ce tutoriel, on abordera le dossier `init.d` qui contient tous les scripts demons.

On parlera aussi d'une variable `$PATH`. C'est une liste de répertoires qui permettent à l'utilisateur de faire directement des commandes sans se déplacer dans le dossier de l'exécutable.
Exemple :

Code : Console

```
man
```

Cette commande appelle en fait `/usr/bin/man`. Et c'est grâce aux variables d'environnement (`$PATH`) que vous pouvez utiliser *man* au lieu de `/usr/bin/man` : elle contient en fait une liste de dossiers (dont `/usr/bin/`).

Voilà, vous avez normalement les connaissances nécessaires pour créer un joli demon.

Au travail

Pour réaliser un demon, tout se passe dans ces dossiers : `/etc/init.d/` et `/usr/bin/`

Créez un tout petit programme en shell qui lance votre serveur, si ce n'est pas déjà fait. Sous certains programmes (TeamSpeak), un programme dit `startscript` est déjà présent.

Lancez un terminal.
Entrez-y la commande pour avoir accès aux droits root :

Code : Console

```
sudo -s
```

Entrez votre mot de passe (le mot de passe root).

Pour ce tutoriel, nous prendrons un exemple de serveur avec « launcher » (en shell, pas besoin d'extensions, bien que cela ne change rien).

Localisez votre script de démarrage et copiez-le dans /usr/bin :

Code : Console

```
cp /dir1/dir2/launcher /usr/bin/launcher
```

Ceci est valable pour tout type de fichiers (Python, shell, etc.).

La suite est un petit peu plus complexe : il faut créer un programme de lancement automatique situé dans /etc/init.d/

Mais je ne sais pas faire ça, moi !

C'est là que Linux est bien gentil, il nous fournit un squelette, un cadre : skeleton.
Donc copiez le skeleton dans votre script avec :

Code : Console

```
cp /etc/init.d/skeleton /etc/init.d/launcher
```

Puis on édite ce startscript :

Code : Console

```
gedit /etc/init.d/launcher
```

Ou sous KDE :

Code : Console

```
kate /etc/init.d/launcher
```

Là, plein de choses s'affichent, mais seulement quelques-unes sont importantes :

Code : Autre

```
PATH=/usr/sbin:/usr/bin:/sbin:/bin
DESC="Description du service"
NAME=nomdudemon
DAEMON=/usr/bin/$NAME
DAEMON_ARGS="--options args"
PIDFILE=/var/run/$NAME.pid
SCRIPTNAME=/etc/init.d/$NAME
```

- PATH : il ne faut pas toucher à cette ligne, c'est la liste des PATH
- DESC : mettez une courte description de votre launcher.
- NAME : mettez le nom de votre exécutable (ici, launcher).
- DAEMON : on n'y touche pas (c'est là que se situe votre script).
- DAEMON_ARGS : les options de lancement (quand vous lancez la commande, il est possible que vous ayez à mettre des paramètres).
- PIDFILE : on laisse.
- SCRIPTNAME : on laisse.

La variable PATH est :

Citation : CCM

> Ce sont les répertoires dans lesquels le shell cherche la commande qu'on écrit au clavier.

Elle permet de faire la commande `ifconfig` et non `/sbin/ifconfig`

Vous remplissez avec vos paramètres comme ci-dessus.
Un exemple avec un script « launcher » dans /usr/bin :

Code : Autre

```
PATH=/usr/sbin:/usr/bin:/sbin:/bin
DESC="Un launcher de mon serveur"
NAME=launcher
DAEMON=/usr/bin/$NAME
DAEMON_ARGS="-option valeur"
PIDFILE=/var/run/$NAME.pid
SCRIPTNAME=/etc/init.d/$NAME
```

Nous allons maintenant passer à « l'enregistrement » de votre demon.

Hé, mais tu oublies de rendre le script exécutable ! Ce n'est qu'un fichier texte pour le moment !

Oooups ! 

Code : Console

```
chmod +x /etc/init.d/launcher
```

Update-rc.d

Le programme qui va gérer tous les demons (et bien plus) est « update-rc.d ».

Mais que fait-il exactement ?

Il crée plusieurs liens depuis /etc/rc0.d/launcher vers /etc/init.d/launcher.

Citation : Man update-rc.d traduit

update-rc.d met à jour automatiquement les liens vers les scripts d'initialisation de type System-V dont le nom est /etc/rc***runlevel***.d/***NN***nom vers les scripts /etc/init.d/name. Ils sont lancés par *init* quand on change de niveau de fonctionnement et sont généralement utilisés pour démarrer ou arrêter des services tels que les demons. « ***runlevel*** » est l'un des niveaux de fonctionnement autorisés par *init*, 0123456789S, et « ***NN*** » est le code à deux chiffres utilisé par *init* pour décider de l'ordre d'exécution des scripts.

Je juge cette citation plutôt claire, et ne pouvant faire mieux, je vous la laisse.

Si vous êtes à la recherche d'une documentation complète (hé, on est sur un site pour Zéros, quand même), je vous redirige sur la documentation officielle, dont j'ai tiré la citation ci-dessus : Update-rc.d sur manpage.ubuntu.com.

Indexation et autres

Vous êtes presque arrivés à la fin. Maintenant, il faut « enregistrer » votre script pour qu'il soit pris en compte.
Tapez

Code : Console

```
update-rc.d launcher defaults
```

update-rc.d pour mettre à jour,
launcher pour le nom de votre script,
defaults options par défaut : placement en bout de file d'attente, pour éviter les conflits

Normalement update-rc.d vous répond (il se peut que le message diffère) :

Code : Console

```
Adding system startup for /etc/init.d/launcher ...
   /etc/rc0.d/K20launcher -> ../init.d/launcher
```

```
/etc/rc1.d/K20launcher -> ../init.d/launcher
/etc/rc6.d/K20launcher -> ../init.d/launcher
/etc/rc2.d/S20launcher -> ../init.d/launcher
/etc/rc3.d/S20launcher -> ../init.d/launcher
/etc/rc4.d/S20launcher -> ../init.d/launcher
/etc/rc5.d/S20launcher -> ../init.d/launcher
```

Vous pouvez maintenant exécuter votre script avec `/etc/init.d/launcher start` ou `/etc/init.d/launcher stop`.

Quelques options

Vous pouvez faire en sorte que votre programme soit en tâche de fond. Pour cela, dans votre /etc/init.d/launcher, regardez au niveau de la fonction `do_start()` :

Code : Bash

```
do_start()
{
        # Return
        #   0 if daemon has been started
        #   1 if daemon was already running
        #   2 if daemon could not be started
        start-stop-daemon --start --quiet --background --make-pidfile --
pidfile $PIDFILE --exec $DAEMON --test > /dev/null \
                || return 1
        start-stop-daemon --start --quiet --background --make-pidfile --
pidfile $PIDFILE --exec $DAEMON -- \
                $DAEMON_ARGS \
                || return 2
        # Add code here, if necessary, that waits for the process to be ready
        # to handle requests from services started subsequently which depend
        # on this one.  As a last resort, sleep for some time.
}
```

`start-stop-daemon` est la commande principale. Analysons ses options :

- `--start` pour démarrer le script ;
- `--quiet` pour rendre le script silencieux (CQFD) ;
- `--background` pour mettre le script en arrière-plan (CQFD) ;
- `--make-pidfile` pour faire un fichier de processus (pour pouvoir forcer l'arrêt plus tard) ;
- `--pidfile $PIDFILE` pour localiser le pidfile créé ci-dessus ;
- `--exec $DAEMON` pour lancer le programme ;
- --test.

Le reste, ce n'est pas la peine d'expliquer.
Vous pouvez donc modifier ce code à votre guise pour faire votre demon.

Je veux supprimer mon demon

Si jamais votre script ne fonctionne pas, ou que vous voulez tout simplement enlever votre serveur, il faut exécuter une suite de commandes :

Code : Console

```
/etc/init.d/launcher stop
update-rc.d -f launcher remove
rm /etc/init.d/launcher
rm /usr/bin/launcher
```

- /etc/init.d/launcher stop → Termine le programme.
- update-rc.d -f launcher remove → Supprime l'enregistrement du script.
- rm /etc/init.d/launcher pour supprimer le skeleton modifié.
- rm /usr/bin/launcher pour supprimer la copie de votre script.

Votre init.d est maintenant nettoyé.

Sous d'autres distributions

Mais je fais quoi si je suis sous Kubuntu, sous Debian, … ?

Je vais vous aider pour quelques autres OS bien connus.
J'ai testé sous toutes les versions ci-dessous.

Je vous parle d'éditeurs de texte, mais le meilleur reste votre préféré, il n'y a pas besoin d'autre chose que la lecture / écriture de texte, même si la coloration rend plus simple la tâche, etc.

Sous Kubuntu

La technique est la même : on copie notre exécutable sur /usr/bin/ puis on copie skeleton, on le modifie pour qu'il lance notre programme, et on fait un `update-rc.d`.
La seule chose qui change est l'éditeur de texte : utilisez ***Kate*** (ou nano, ou Vim, etc.).

Sous Debian

De la même façon que sous Ubuntu ou Kubuntu, Debian possède un skeleton, un dossier `init.d/` un `update-rc.d`, etc.
Comme sous Ubuntu et Kubuntu, vous pouvez utiliser **nano**, **Vim**, etc.

Sous Mandriva

Là ça se complique : le skeleton est un peu différent, il y a des chmods pour les priorités d'exécution, etc.
Je vous redirige donc sur la documentation officielle : Wiki de Mandriva.

Sous Fedora

Fedora gère ses demons de la même manière que Debian, donc pas de problèmes

Sous ArchLinux

Ici, autre fonctionnement, il n'y a pas de `/etc/init.d`. Il faut en fait ajouter son script au `/etc/rc.d/` en respectant le skeleton suivant (attention à bien remplacer tous les noms de programmes, et autres chaines contenant "skeleton" :

Secret (cliquez pour afficher)

Code : Bash

```
#!/bin/bash

. /etc/rc.conf
. /etc/rc.d/functions

PID=$(pidof -o %PPID -x /usr/sbin/skeleton)
case "$1" in
  start)
    stat_busy "Starting skeleton"
    [ -z "$PID" ] && /usr/sbin/skeleton > /dev/null 2 > &1 &
    if [ $? -gt 0 ]; then
stat_fail
    else
```

```
        add_daemon skeleton
              stat_done
            fi
            ;;
          stop)
            stat_busy "Stopping skeleton"
            [ ! -z "$PID" ]  && kill $PID &> /dev/null
            if [ $? -gt 0 ]; then
        stat_fail
            else
        rm_daemon skeleton
              stat_done
            fi
            ;;
          restart)
            $0 stop
            $0 start
            ;;
          *)
            echo "usage: $0 {start|stop|restart}"
        esac
        exit 0
```

Puis il faut modifier le fichier /etc/rc.conf en ajoutant à la liste des demons (bas du fichier) « @skeleton » (ou skeleton = nom du script, et @ pour mettre le script en tâche d'arrière-plan).
Et voilà, vous savez faire des demons. J'espère que ce tutoriel vous a plu !

Je tiens à remercier Norrin pour ses commentaires, nim65s qui m'a aidé pour ArchLinux, et enfin merci aux commentaires qui m'ont aidé.
Je remercie aussi Poulpette et ptipilou pour leur correction plus qu'utile.

Ce tutoriel a été corrigé par les zCorrecteurs.