

Sommaire

Gestion des services	1
Définitions.....	1
Fonctionnement.....	1
Commandes utiles.....	2
chkconfig.....	2
sysv-rc-conf.....	3
rc-update.....	4
Ajouter ses propres services.....	4
1 – Ecrire le script de configuration du démarrage.....	4
2 – Ajouter le service à la base de chkconfig.....	6
Ajouter ses propres services sur une Gentoo.....	6
Faire du ménage.....	7
Copyright	9

Gestion des services

Gestion des services

par Philippe (superjoker@club-internet.fr), complété par Anne, Fred et Jice

Comment ajouter et supprimer des services (daemons, etc.) au démarrage.

Définitions

Les daemons (ou démons) sont des programmes résidents chargés au démarrage. A chaque runlevel, correspond une liste de daemons à lancer (1 à 5) ou à arrêter (6 ou 0).

D'autres programmes que des démons peuvent également être lancés dès le démarrage de la machine, avec le même mécanisme.

Les runlevels ou "niveaux d'exécution", correspondent aux services qui vont être lancés au démarrage de la machine. En général (mais toutes les distributions n'utilisent pas la même numérotation), on peut avoir les niveaux d'exécution suivant :

- 0 : arrête la machine.
- 1 : mode simple utilisateur (ou single, ou encore failsafe). Ce mode est utile pour dépanner une machine plantée.
- 3 : mode console. Les services sont lancés, mais le serveur X n'est pas activé (ainsi que les services dont il dépend).
- 4 (Slackware) ou 5 (Mandrake, RedHat..) : mode graphique : le serveur X et les services dont ils dépend sont lancés.
- 6 : redémarre la machine.

Le niveau d'exécution est déterminé (dans l'ordre) soit :

- lors du boot : si vous précisez un niveau sur la ligne de commande du noyau (par exemple, au prompt LILO, taper "linux 1"),
- dans le fichier /etc/inittab, où le runlevel par défaut est défini,
- par la commande `init <runlevel>` qui permet de changer de runlevel en cours de fonctionnement.

Remarque : Le fonctionnement des services sur une gentoo, bien qu'assez similaire, est différent.

Fonctionnement

Les services sont lancés par des scripts situés dans /etc/init.d (ou dans /etc/rc.d/init.d, (qui sont souvent le même fichier, l'un étant un lien vers l'autre). Chaque script contient une description ce qui permet de savoir ce que fait chaque daemon en début de script.

Le répertoire /etc/rc.d/ contient aussi des répertoires nommés rcX.d (avec X numéro de runlevel). Chacun de ces répertoires contient un lien vers les scripts situés dans init.d.

Exemple:

```
$ ls -l /etc/rc.d/rc5.d
rwxrwxrwx 1 root root 13 Jun 16 23:23 K30usb -> /etc/rc.d/init.d/usb*
lrwxrwxrwx 1 root root 16 Jun 17 00:03 S30syslog -> /etc/rc.d/init.d/syslog*
```

Admin-admin_boot-daemons

```
lrwxrwxrwx 1 root root 18 Jun 17 00:05 S75keytable -> /etc/rc.d/init.d/keytable*
```

La 1^{ère} lettre détermine si le daemon est activé (S comme start) dans ce niveau d'exécution (runlevel) ou arrêté (K comme kill).

Les 2 chiffres permettent de trier l'ordre d'exécution des services (dans l'exemple, syslog est démarré avant keytable).

Remarque : sur une gentoo, les répertoires correspondant à un runlevel particulier sont stockés dans /etc/runlevels/. La gestion des runlevels par gentoo est complètement différente : il n'est pas nécessaire de se préoccuper de l'ordre de démarrage des services car chaque service précise quels sont les services qui lui sont nécessaires, le reste (dépendances...) est géré automatiquement. De plus les runlevels ont des noms plutôt que des numéros. Par default, une Gentoo arrive avec les runlevels : boot, default, nonetwork, single.

Commandes utiles

Plutôt que de modifier directement les liens, on va utiliser la commande suivante :

```
/etc/init.d/nom_service {start|stop|restart|reload|status} ou
```

```
service nom_service {start|stop|restart|reload|status}. À ce propos, je recommande d'utiliser "bash-completion" qui permet d'améliorer la complétion automatique de la ligne de commande, et par exemple de lister tous les services disponibles, en tapant "service" puis la touche [Tab].
```

Exemple:

```
/etc/rc.d/init.d/linuxconf start
```

Note: les options start/stop/restart lancent/arrêtent/redémarrent le service spécifié pour tous les niveaux.

Note (Mandrake, RedHat...) : on peut utiliser la commande service pour démarrer un service particulier,

```
$ service nomduservice start
```

ou :

```
$ service nomduservice stop
```

pour l'arrêter.

chkconfig

pour Mandrake et RedHat

La commande chkconfig est un peu plus puissante.

Pour ajouter un daemon dans tous les niveaux de démarrage:

```
/sbin/chkconfig --add le_service
```

Note: le daemon doit obligatoirement se trouver dans /etc/rc.d/init.d ou /etc/init.d.

Pour lister tous les daemons avec leurs status:

```
/sbin/chkconfig --list
```

```
atd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
```

```
xfst 0:on 1:on 2:on 3:on 4:on 5:on 6:on
```

```
keytable 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

```
gpm 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

Autre option:

```
/sbin/chkconfig --list le_service
```

pour ne lister que celui souhaité.

Pour activer ou désactiver un service au démarrage :

```
/sbin/chkconfig --level 123456 mon_service on/off
```

(avec 123456 le(s) runlevel(s) pour le(s)quel(s) le service doit être ou non activé).

Pour plus de détails, voir la man page.

C'est bien beau, mais si je dois me taper tout ça à la mimine ?! Stop ! Linux a pensé à nous, et pour se simplifier la tâche, on a plusieurs outils : Linuxconf via Panneau de configuration/gestion des services (qui stoppe ou arrête un daemon pour tous les runlevels), Runleveditor (qui permet de choisir pour chaque runlevel les daemons à activer ou non), ksysv, etc. Fais ton choix camarade ;-)

sysv-rc-conf

pour (K)(X)(Ed)Ubuntu

La commande `sysv-rc-conf` remplit la même fonction que la commande `chkconfig` ci-dessus mentionnée pour les distributions basées sur ubuntu (et sauf erreur debian, à vérifier).

Pour ajouter ou enlever un daemon dans tous les niveaux de demarrage:

```
/usr/sbin/sysv-rc-conf le_service on | off
```

Note: le daemon doit obligatoirement se trouver dans `/etc/rc.d/init.d` ou `/etc/init.d`.

Pour lister tous les daemons avec leurs status:

```
/usr/sbin/sysv-rc-conf --list
atd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
xfs 0:on 1:on 2:on 3:on 4:on 5:on 6:on
keytable 0:off 1:off 2:on 3:on 4:on 5:on 6:off
gpm 0:off 1:off 2:on 3:on 4:on 5:on 6:off
Pour lister un daemon déterminé avec son status:
```

```
/usr/sbin/sysv-rc-conf --list le_service
```

Pour activer ou désactiver un service au démarrage :

```
/usr/sbin/sysv-rc-conf --level 123456 mon_service on/off (avec 123456 le(s) runlevel(s) pour le(s)quel(s) le service doit être ou non activé).
```

Enfin il existe un mode interactif:

```
/usr/sbin/sysv-rc-conf sans arguments ouvre un utilitaire graphique en mode texte listant tous les services et permettant d'éditer leur comportement dans chacun des runlevels
```

Pour plus de détails, voir la man page.

rc-update

pour Gentoo

La configuration des services passe par la commande rc-update. Son fonctionnement est des plus simples.

```
$ rc-update add nomduservice nomdurunlevel
```

Par exemple, supposons que vous ayez un script speedtouch qui démarre votre connexion internet, pour qu'elle démarre automatiquement, il suffit de faire :

```
$ rc-update add speedtouch default
```

Pour arrêter un service c'est exactement pareil :

```
$ rc-update del speedtouch default
```

À partir de maintenant, votre speedtouch ne démarrera plus automatiquement.

Pour lister les services d'un runlevel particulier :

```
$ ls /etc/runlevels/nomdurunlevel
```

Ajouter ses propres services

Nous allons détailler les étapes nécessaires pour configurer un nouveau service et l'ajouter à la base de chkconfig. Nous allons prendre l'exemple d'un service nommé **bidule**, qui devra être démarré aux runlevels 3, 4 et 5.

1 – Ecrire le script de configuration du démarrage

Pour écrire le script /etc/rc.d/init.d/biduled, on peut s'inspirer d'un script existant dans ce répertoire.

Ci-dessous un exemple de ce script que nous allons détailler

```
root@pingu# cat /etc/rc.d/init.d/biduled
```

```
#!/bin/sh
```

```
# description: exemple de script pour Léa
```

```
# chkconfig: 345 99 0
```

Attention la syntaxe des 2 dernières lignes est à respecter à la lettre pour que le service puisse être ajouté correctement par la commande chkconfig. Nous devons spécifier 2 mots clé :

- **description** : décrire le service en quelques mots. Si votre description utilise plus d'une ligne, utiliser un "\n" pour respecter la continuité.
- **chkconfig** : cette ligne contient 3 informations. Ici 345 indique les runlevels auxquels sera démarré le service, 99 indique le numéro de séquence des liens S, indique le numéro de séquence des liens K.

```
#source function library
```

```
./etc/init.d/functions
```

On charge ici en mémoire les fonctions définies dans le fichier /etc/init.d/functions. Nous allons en effet utiliser certaines d'entre elles comme daemon, killproc, status.

```
case $1 in
```

```
'start')
```

```
rc-update
```

Admin-admin_boot-daemons

```
[ -f /var/lock/subsys/biduled ] &&  
exit 0  
echo -n "exécute bidule"  
daemon /usr/bin/biduled  
echo  
touch /var/lock/subsys/biduled  
;;
```

On démarre ici le traitement des arguments de gestion du service. Pour le **démarrage**, on vérifie d'abord qu'il n'existe pas de fichier de lock, ce qui permet d'éviter de démarrer 2 fois le même service. S'il existe alors on sort du script. Dans le cas contraire, on affiche un message sur la console indiquant le démarrage de bidule et c'est la fonction daemon qui se charge de lancer le service. Enfin au moment du lancement, on crée le fichier de lock.

```
'stop')  
echo -n "arrête bidule"  
killproc biduled  
echo  
rm -f /var/lock/subsys/biduled  
;;
```

En ce qui concerne l'**arrêt** du service, on utilise la fonction killproc qui récupère le numéro de processus de notre service et le tue avec la commande kill. On efface également le fichier de lock afin de permettre éventuellement le redémarrage du service.

```
'restart')  
$0 stop  
$0 start  
;;
```

Le **redémarrage** du service consiste simplement à utiliser ce même script pour l'arrêter puis le démarrer

```
'status')  
status biduled  
;;
```

Cet argument permet de vérifier si le service est lancé, au moyen de la fonction status.

```
*)  
echo "Usage : biduled \  
{start|stop|restart|status}"  
exit 1  
;;  
esac  
exit 0
```

Ce cas de figure traite le cas où l'utilisateur ne rentre pas le bon argument. On lui renvoie donc un message d'erreur lui indiquant ceux qui doivent être utilisés.

Ouf ! Nous avons terminé le script. Il ne reste plus qu'à **le rendre exécutable** :

```
root@pingu# chmod +x /etc/rc.d/init.d/biduled
```

2 – Ajouter le service à la base de chkconfig

Le plus gros du travail est fait ! Il ne reste plus qu'à ajouter le service biduled à la base des services gérée par chkconfig. Pour ce faire, rien de plus simple :

```
root@pingu# chkconfig --add biduled
```

Le service fait maintenant partie de la base, les liens symboliques de démarrage ont été créés automatiquement de la manière suivante :

```
# ll /etc/rc.d/rc?.d/*biduled lrwxrwxrwx 1 root root 17 oct 11 09:37 rc0.d/K00biduled -> /etc/rc.d/init.d/biduled
```

```
lrwxrwxrwx 1 root root 17 oct 11 09:37 rc1.d/K00biduled -> /etc/rc.d/init.d/biduled
```

```
lrwxrwxrwx 1 root root 17 oct 11 09:37 rc2.d/K00biduled -> /etc/rc.d/init.d/biduled
```

```
lrwxrwxrwx 1 root root 17 oct 11 09:37 rc3.d/S99biduled -> /etc/rc.d/init.d/biduled
```

```
lrwxrwxrwx 1 root root 17 oct 11 09:37 rc4.d/S99biduled -> /etc/rc.d/init.d/biduled
```

```
lrwxrwxrwx 1 root root 17 oct 11 09:32 rc5.d/S99biduled -> /etc/rc.d/init.d/biduled
```

```
lrwxrwxrwx 1 root root 17 oct 11 09:37 rc6.d/K00biduled -> /etc/rc.d/init.d/biduled
```

Si vous n'avez pas chkconfig ou désirez faire cette opération manuellement, il vous suffit de créer les liens symboliques listés ci-dessus.

Nous pouvons également vérifier de la manière suivante :

```
root@pingu# chkconfig --list | grep biduled
```

```
biduled 0:Arrêt 1:Arrêt 2:Arrêt 3:Marche 4:Marche 5:Marche 6:Arrêt
```

Ajouter ses propres services sur une Gentoo

Le principe est le même que dans la section précédente, mais les fonctions disponibles sont différentes.

Un script de démarrage de Gentoo doit forcément être stocké dans /etc/init.d et commencer par :

```
#!/sbin/runscript
```

```
# Copyright Léa Linux 2003.
```

```
# Distributed under the terms of the GNU General Public License v2
```

Vous pouvez remarquer qu'il ne commence pas par #!/bin/sh comme le fait habituellement un script. En fait c'est un habillage de /bin/sh qui met en place tout ce qui est nécessaire aux services de démarrage.

Ensuite, on précise ce qui est nécessaire à notre script ainsi que ce qu'il fournit comme service (si le nom du service est différent de celui du script) :

```
depend() {
use cups
need net my-firewall
provide monbeauservice
}
```

Ce qui signifie :

- use cups : si le service cups doit être démarré lui aussi, alors notre service est capable de l'utiliser, mais pour cela il faut que cups soit démarré avant le notre.
- need net my-firewall : ces deux services (net et my-firewall) sont nécessaires à notre service et ils doivent absolument être démarrés avant le nôtre.

Admin-admin_boot-daemons

- provide monbeauservice : notre service à un nom différent de celui du script (qu'on appellera par exemple, /etc/init.d/monservice). Ceci n'est nécessaire que si notre service peut être utilisé par un autre.

La fonction depend() est appelée en interne par /sbin/runscript pour déterminer l'ordre de démarrage des services ainsi que ceux qu'il faut démarrer.

Ensuite, il nous faut écrire ce qui doit se passer quand on démarre un service.

```
start() {
ebegin "Démarrage de MonService à Moi"
start-stop-daemon --start --quiet --exec /usr/bin/monservice
end $?
}
```

Ici, nous avons utilisé trois fonctions qui simplifient la rédaction d'un script de démarrage pour Gentoo :

- ebegin qui affiche un message sans aller à la ligne en utilisant un peu de couleur.
- start-stop-daemon qui gère le démarrage de services (en évitant par exemple de démarrer deux fois le même, et autres astuces).
- end : qui teste la valeur de retour du programme précédent et affiche '[OK]' ou '[!]' suivant la valeur.

L'utilisation de ces fonctions n'est absolument pas nécessaires. Mais elle est bien pratique.

Puis, nous devons écrire la fonction appelé quand on veut stopper le service :

```
stop() {
ebegin "J'arrête Mon Service à Moi"
start-stop-daemon --stop --quiet --exec /usr/bin/monservice
end $?
}
```

Je pense que c'est assez clair.

Voilà c'est tout ! Le reste est géré par le programme /sbin/runscript. Si vous sauvez ce script dans /etc/init.d/monservice, vous l'activez par :

```
# chmod +x /etc/init.d/monservice
# rc-update add monservice default
La classe quoi ! Qui a dit que Gentoo était complexe ?
```

Faire du ménage

Dans mon cas et à titre d'exemple (internet, pas d'imprimante, travail sous X, son) je ne garde en activité que : syslog, xfs, keytable.

Pour information, voici une liste (non exhaustive) de quelques daemons et de leur fonctions.

apmd: nécessaire uniquement pour les ordinateurs portables

xntpd: Network time protocol

portmap: nécessaire si vous utilisez un service rpc, comme NIS ou NFS

sound: configuration des sons (ma carte fonctionne très bien sans ??? ndlr:normal si le fichier

Admin-admin_boot-daemons

/etc/modules.conf est bien conçu)

netfs: c'est le client nfs, utilisé pour monter des filesystems depuis un serveur nfs

rstatd, rusersd, rwhod, rwall: ne pas exécuter tous les services car ils fournissent trop d'informations aux utilisateurs à distance

bootparamd: Utilisé par les clients sans lecteur de disquette (vulnérable)

squid: serveur proxy

yppasswdd: nécessaire si vous êtes un serveur NIS (extrêmement vulnérable)

ypserv: nécessaire si vous êtes un serveur NIS (extrêmement vulnérable)

dhcpcd: démarre le daemon du serveur dhcp

atd: utilisé pour le service at, similaire à cron, mais n'est pas nécessaire

pcmcia: parle de lui-même

snmpd: daemon SNMP, peut donner à des utilisateurs distants des informations détaillées sur votre système

named: serveur DNS

routed: RIP, n'exécutez cela que si vous en avez vraiment besoin

lpd : services d'impression

mars-nwe: fichier Netware et serveur d'impression

nfs: Utilisé pour le serveur NFS, lancez le que si vous en avez absolument besoin

amd: daemon AutoMount, sert à monter les filesystems distants

gated: sert à lancer d'autres protocoles de routage comme OSPF

sendmail: Vous pourrez toujours envoyer/recevoir des emails par Netscape (ou autre) sans lui.

httpd: serveur web Apache

yplibind: nécessaire si vous êtes un client NIS

xfs: xfont server (indispensable si vous êtes sous X).

innd: serveur de news

arpwatch: off par défaut. Rapport d'activité de datagrammes IP via mail

kudzu: détection des périphériques. A réactiver à l'occasion

anacron: reprise de jobs de la crontab après un crash

crond: si vous ne savez pas ce qu'est une [./admin/automate.php3 crontab], désactivez-le.

rawdevices: partitions spécifique sous ORACLE ou autre SGBD

random: améliore la génération aléatoire de nombres (peut être utile pour les joueurs)

rhnd: redhat network

linuxconf: j'utilise linuxconf sans ce daemon (peut être est-ce utile pour l'administration à distance ?)

nfslock: si vous n'êtes pas serveur NFS, désactivez-le

usb: parle de lui-même

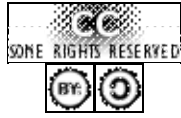
gpm: fournit des fonctions pour le support de la souris en mode texte (utile pour midnight commander en particulier)

Ouf, c'est fini...

Cette page est issue de la documentation 'pré-wiki' de Léa a été convertie avec HTML::WikiConverter. Elle fut créée par Philippe le 08/07/2001.

Copyright

Copyright © 08/07/2001, Philippe



*Ce document est publié sous licence Creative Commons
Attribution, Partage à l'identique 2.0 :*
<http://creativecommons.org/licenses/by-sa/2.0/>