

Sommaire

L'initialisation de la distribution Slackware.....	1
Introduction.....	1
Les principes généraux.....	1
Les fichiers et scripts de configuration.....	2
Le fichier /etc/inittab.....	2
Les scripts contenus dans le répertoire /etc/rc.d et leur signification.....	5
L'ouverture d'une session.....	9
Le programmeagetty.....	9
Le programme login.....	10
Le programme Bash.....	10
Conclusion.....	11
Copyright.....	12

L'initialisation de la distribution Slackware

L'initialisation de la distribution Slackware

Par Michel Lalumiere

Révision : Pascal Cussy

Voici un résumé du système d'initialisation de la Slackware qui se veut un peu particulier de par sa conception et ses niveaux d'exécution par rapport aux autres distributions qui emploient le System V.

Introduction

Comprendre le mécanisme d'initialisation de la Slackware permet de personnaliser et de configurer correctement la distribution. C'est à ce stade par exemple que l'on pourra décider de charger certains modules ou de démarrer ou non certains services.

Ce mécanisme d'initialisation repose en grande partie sur le programme `init` qui, à partir de son fichier de configuration `/etc/inittab`, va lancer certains des scripts bash contenus dans le répertoire `/etc/rc.d`. Les développeurs de Slackware nomment son processus d'initialisation BSD-like parce qu'il s'appuie en partie sur le système développé par la Berkeley Software Development (BSD). Certains le décrivent ainsi "Faire du BSD avec du System V" (le "V" désigne le chiffre romain 5). Le System V constitue un autre mode d'initialisation, utilisé par exemple par la Red Hat et les distributions dérivées comme la Mandrake. Cet article vise à couvrir le processus d'initialisation, du démarrage de l'ordinateur à l'ouverture d'une session. Pour ce, quelques principes généraux sont exposés. La configuration de l'initialisation proprement dite est traitée dans un deuxième temps avec la présentation du fichier `/etc/inittab` et des scripts du répertoire `/etc/rc.d`. Pour finir, sont décrits les mécanismes conduisant à l'ouverture d'une session sur le système.

Les principes généraux

A l'allumage de l'ordinateur, le BIOS (Basic Input Output System) recherche les périphériques présents puis exécute le POST (Power On Self Test) qui effectue un certain nombre de vérifications du matériel. Il recherche ensuite un système d'exploitation dans la liste des supports précisé dans le Setup du BIOS.

Nous supposons ici que le système d'exploitation est situé sur le disque dur même si celui-ci n'est pas toujours le premier support recherché.

Sur la Slackware, comme sur toute distribution Linux, un chargeur de boot est installé. Il s'agit en l'occurrence de Lilo. Celui-ci place sur le disque un petit programme chargé de lancer le système. Ce programme peut être placé sur le premier secteur de la partition active ou, si l'on dispose de plusieurs système d'exploitation sur sa machine, sur le MBR (Master Boot Record).

Il s'agit en fait du premier secteur du disque dur et contient la table des partitions et un programme qui charge le secteur de début de la partition active si Lilo n'y est pas installé.

Lilo charge le noyau qui est exécuté en mémoire. Entre autres tâches, celui-ci détecte le matériel, monte le système de fichiers et se charge pour finir de lancer le programme `/sbin/init` qui démarre l'initialisation du système.

Seul et unique processus à être lancé directement par le kernel, `init` est le père de tous les autres processus (il s'agit du processus numéro 1 comme le montrent les commandes `ps -ax` ou `pstree -a`).

Il a pour tâche de lancer chacun des processus, démons, sessions de login et de gérer l'arrêt du système ainsi que d'autres fonctions pour lesquelles il a été conçu ou configuré.

Pour cela, différents états dits niveaux d'exécution (ou `runlevels`) sont définis. Plus précisément, un niveau d'exécution correspond à une configuration logicielle qui permet de lancer un certain nombre de processus.

Le programme `init` définit 8 niveaux d'exécution: les 7 niveaux numérotés 0 à 6 et le niveau S (ou s)

comme `Single`. Les niveaux 7 à 9 sont également reconnus mais ils ne sont pas documentés. Il existe également les pseudo-niveaux `a`, `b` et `c` qui sont utilisés pour lancer des commandes à partir du fichier `/etc/inittab` dans un mode particulier appelé `ondemand`.

Le niveau `S` n'est pas destiné à être utilisé directement mais pour les scripts exécutés au niveau 1. Il est utilisé pour passer en mode mono-utilisateur et ne requiert pas de fichier `/etc/inittab`.

La page de manuel d'`init` précise que les niveaux 0, 1 et 6 sont réservés. Le niveau 0 est utilisé pour arrêter le système, le niveau 6 pour relancer la machine et le niveau 1 pour passer en mode mono-utilisateur (par exemple, pour maintenance du système).

On peut passer à un autre niveau d'exécution à l'aide de la commande `telinit`.

Avec le System V, à chaque niveau d'initialisation correspond un répertoire `/etc/rc.d/rcn.d`, n désignant le niveau d'initialisation. En fait, les scripts sont situés dans le répertoire `/etc/rc.d/init.d` et des liens symboliques sont créés vers ceux-ci dans chaque répertoire `rcn.d`.

Avec la Slackware et son système d'initialisation de type BSD, il n'y a plus de répertoire particulier affecté à chaque niveau mais des scripts, organisés par fonction (par exemple `rc.M` pour le chargement des modules) et qui peuvent être communs à plusieurs niveaux d'exécution.

C'est en fait l'écriture du fichier de configuration `/etc/inittab` qui permet de lancer ces scripts. Ceci nous amène à parler maintenant des fichiers de configuration.

Les fichiers et scripts de configuration

Comme précisé auparavant, il s'agit du fichier `/etc/inittab` et des scripts situés dans le répertoire `/etc/rc.d`. De façon sommaire, on peut dire que le fichier `/etc/inittab` va dans tous les cas indiquer au programme `init` d'exécuter `rc.S` puis, selon le niveau d'exécution, `rc.6` ou `rc.4` ou `rc.K` ou `rc.M`, ce dernier exécutant la plupart des autres scripts du répertoire `/etc/rc.d` (avec un nom en général du type `rc.fonction`, l'extension `fonction` permettant de préciser l'objet du script).

Le fichier `/etc/inittab`

Il convient de dire quelques mots sur la façon de rédiger ce fichier avant de discuter de celui de la Slackware. La commande `man inittab` précise qu'une entrée dans le fichier `/etc/inittab` doit avoir la forme:

`code:niveau:action:commande`
avec

- `code` une séquence de 1 à 4 caractères qui identifient de manière unique une entrée dans le fichier (en général, deux caractères seulement sont utilisés pour des raisons historiques)
- `niveau` un ou plusieurs niveaux d'exécution (écrits les uns à la suite des autres sans espace)
- `action` l'action qui doit être entreprise (voir ci-dessous)
- `processus` la commande ou le script qui sera lancé pour le ou les niveaux précisés

Les actions peuvent être :

respawn	le processus est automatiquement relancé une fois terminé (par exemple, la console redémarre lorsque l'on ferme la session avec la commande <code>exit</code>).
wait	le processus n'est lancé qu'une seule fois et <code>init</code> attend qu'il se termine avant de poursuivre.
once	le processus n'est exécuté qu'une seule fois dans le ou les niveaux spécifiés.
boot	

Admin-admin boot-slack init

	le processus sera lancé au moment du lancement du système, le niveau d'exécution étant ignoré.
bootwait	comme boot mais <code>init</code> attend qu'il se termine pour continuer.
off	ne fait rien du tout.
ondemand	pour lancer les processus lors d'un pseudo passage en mode ondemand, ces niveaux étant nommés <code>a</code> , <code>b</code> et <code>c</code> (en fait, il n'y a pas de changement de niveau, les commandes processus sont simplement lancées).
initdefault	pour définir le niveau par défaut après le démarrage, le champ commande étant ignoré (si aucun niveau n'est spécifié, le niveau d'exécution sera demandé sur la console).
sysinit	définit un processus à lancer pendant le démarrage et qui sera exécuté avant toutes les lignes boot ou bootwait, le champ niveau étant ignoré.
powerwait	définit la commande à lancer quand <code>init</code> reçoit le signal SIGPWR qui indique un problème d'alimentation. <code>init</code> attendra que le processus soit terminé pour poursuivre.
powerfail	idem que powerwait mais <code>init</code> n'attend pas que le processus soit terminé.
powerokwait	définit le processus à lancer si un SIGPWR est reçu. Le fichier <code>/etc/powerstatus</code> s'il existe est contrôlé. Si celui-ci contient la chaîne OK, c'est que le problème d'alimentation est résolu.
powerfailnow	survient si la batterie d'un système d'alimentation externe est presque vide et que l'alimentation va cesser.
ctrlaltdel	définit le processus à lancer si la séquence de touches [Ctrl]+[Alt]+[Suppr] est initiée. Habituellement c'est la commande shutdown qui est appelée.
kbrequest	le processus sera lancé si <code>init</code> reçoit un signal indiquant qu'une combinaison spéciale de touches a été pressée.

Tout ceci peut-être illustré à l'aide du fichier `inittab` de la Slackware 9.1, dont voici un extrait. Les commentaires ont été traduits.

Les scripts qu'il appelle sont tous regroupés dans le répertoire `/etc/rc.d` et comme vous le constaterez plus loin, ce répertoire ne contient que des scripts `bash` reliés aux différents niveaux d'exécution et non des sous-répertoires comme pour le System V (qui est utilisé par exemple dans les distributions de type RedHat).

Il est précisé dans ce fichier que la Slackware utilise 6 niveaux d'exécution :

- 0 = arrêt du système
- 1 = mode mono-utilisateur
- 2 = inutilisé (mais configuré comme le niveau 3)
- 3 = mode multi-utilisateurs (niveau par défaut)
- 4 = mode graphique avec Kdm, Gdm ou Xdm
- 5 = inutilisé (mais configuré comme le niveau 3)
- 6 = redémarrage

Les niveaux sont alors définis de la façon suivante

```
# Niveau d'exécution par défaut défini par l'utilisateur
```

```
id:3:initdefault:
```

Il s'agit de la ligne qui précise le niveau d'initialisation par défaut. On remarque que bien évidemment aucune

Admin-admin boot-slack init

commande n'est spécifiée. Il est également précisé qu'il ne faut jamais mettre le niveau à 0 ou 6 pour éviter que le système ne s'arrête ou ne redémarre dès l'initialisation.

```
# Initialisation du système (lancé au démarrage du système).
```

```
si:S:sysinit:/etc/rc.d/rc.S
```

Le script `/etc/rc.d/rc.S` est lancé au démarrage du système quel que soit le niveau du fait que l'action `sysinit` est précisée (le choix du niveau `S` est en fait inutile ici).

Ce script, entre autres choses, active le swap, teste le système de fichiers, reboote si le système de fichiers a été réparé et a changé et lance le script `/etc/rc.d/rc.modules` qui charge les modules nécessaires.

```
# Script à lancer lorsque l'on passe en mode mono-utilisateur (niveau 1 ou S)
```

```
su:1S:wait:/etc/rc.d/rc.K
```

Ce script est chargé de terminer tous les démons et de passer ensuite au niveau 1 à l'aide de la commande `telinit -t 1 1`.

```
# Script à lancer lorsque l'on passe en mode multi-utilisateurs.
```

```
rc:2345:wait:/etc/rc.d/rc.M
```

Le script `rc.M` est lancé aux niveaux multi-utilisateurs 2, 3, 4 et 5. Comme on le verra plus loin, c'est l'un des scripts les plus importants du répertoire `/etc/rc.d` puisqu'il est chargé de lancer une grande partie des autres.

```
# Que faire des "trois doigts magiques"
```

```
ca::ctrlaltdel:/sbin/shutdown -t5 -r now
```

On peut remplacer l'option `-r` par `-h` pour que l'ordinateur soit arrêté et non relancé lorsque l'on appuie simultanément sur les trois touches **[Ctrl]+[Alt]+[Suppr]** dans une console

```
# Le niveau 0 arrête le système.
```

```
l0:0:wait:/etc/rc.d/rc.0
```

En fait, `rc.0` n'est qu'un lien vers `rc.6` qui teste néanmoins si c'est le niveau 0 qui a été appelé. Dans ce dernier cas, il exécute la commande `poweroff`.

Ce script permet d'éteindre correctement la machine en arrêtant un certain nombre de démons comme `mysql` ou `apache` et en démontant le système de fichiers.

```
# Le niveau 6 relance le système.
```

```
l6:6:wait:/etc/rc.d/rc.6
```

Au niveau d'exécution 6, le script `rc.6` est appelé directement et lance dans ce cas la commande `reboot`.

```
# Que faire en cas de pb d'alimentation (shutdown en mode mono-utilisateur)
```

```
# (note de Martial: concerne les systèmes avec onduleurs et démon d'alim)
```

```
pf::powerfail:/sbin/genpowerfail start
```

```
# Si l'alimentation est rétablie avant le shutdown, on annule celui-ci
```

```
pg::powerokwait:/sbin/genpowerfail stop
```

```
# Les getties en mode multi-utilisateur
```

```
c1:1235:respawn:/sbin/agetty 38400 tty1 linux
```

```
c2:1235:respawn:/sbin/agetty 38400 tty2 linux
```

```
c3:1235:respawn:/sbin/agetty 38400 tty3 linux
```

```
c4:1235:respawn:/sbin/agetty 38400 tty4 linux
```

Admin-admin boot-slack init

```
c5:1235:respawn:/sbin/agetty 38400 tty5 linux
c6:12345:respawn:/sbin/agetty 38400 tty6 linux
```

Il s'agit des consoles texte. On passe de l'une à l'autre par **[Ctrl]+[ALT]+[Fn]**. On peut en commenter quelques unes si la machine n'est pas utilisée simultanément par plusieurs personnes afin d'économiser un peu d'espace mémoire.

L'action `respawn` entraîne que le programme `agetty` est automatiquement relancé dès que l'on quitte une session, permettant qu'une autre soit ouverte.

Elles sont activées pour les niveaux 1, 2, 3 et 5. On remarque que la console 6 est également activée au niveau 4. Une remarque dans le fichier explique pourquoi :

"Le niveau 4 n'était utilisé que pour les systèmes uniquement sous X, jusqu'à ce que l'on découvre qu'il mettait `init` dans une boucle qui maintenait la charge système en permanence à 1. En conséquence, il y a maintenant un `getty` ouvert sur `tty6`. Heureusement, personne ne s'en est rendu compte. ;^)

En tous cas, il n'est pas mauvais d'avoir une console texte au cas où quelque chose irait mal sous X."

```
# Lignes série
#s1:12345:respawn:/sbin/agetty 19200 ttyS0 vt100
#s2:12345:respawn:/sbin/agetty 19200 ttyS1 vt100

# Lignes Dialup
#d1:12345:respawn:/sbin/agetty -mt60 38400,19200,9600,2400,1200 ttyS0
vt100
#d2:12345:respawn:/sbin/agetty -mt60 38400,19200,9600,2400,1200 ttyS1
vt100
```

```
x1:4:wait:/etc/rc.d/rc.4
```

Cette dernière ligne définit ce qui doit être fait au niveau 4. Le script `rc.4` est exécuté ce qui permet de lancer un gestionnaire graphique (`gdm`, `kdm` ou `xdm`).

On voit donc que les niveaux 0 à 6 y sont référés ainsi que le niveau S, mais en fait ils sont regroupés pour que le système n'en voie que trois.

1. Le niveau d'arrêt et de redémarrage du système (0 et 6) le niveau 0 étant en fait un lien du niveau 6.
2. Un niveau pour le passage en mode mono-utilisateur (1 ou S)
3. Un niveau multi-utilisateur (2 et 3 et 4 pour le mode X)

On notera ainsi qu'il n'y a aucune différence entre les modes 2-3 et 5 et que les modes 0 et 6 partagent le même script puisqu'il n'y a que la dernière commande qui doit changer (`poweroff` ou `reboot` selon le cas). Les niveaux 2 et 5 sont néanmoins évoqués au cas où l'on souhaiterait définir de nouveaux niveaux multi-utilisateurs personnalisés.

Enfin, ce fichier peut être édité pour y insérer une commande quelconque que l'utilisateur aimerait avoir lors du démarrage. Voici à cet égard un exemple de l'insertion d'une telle commande pour avoir le contenu d'un fichier log sur une console quelconque.

```
log:2345:respawn:/usr/bin/tail -f --retry /var/log/sys.log >/dev/ttyXXX
```

Les scripts contenus dans le répertoire `/etc/rc.d` et leur signification

Nous avons vu auparavant que `init` utilisait fichier `/etc/inittab` pour exécuter les différentes tâches nécessaires à l'initialisation du système. C'est à partir de ce fichier que sont lancés les scripts contenus dans

le répertoire `/etc/rc.d`.

On va maintenant passer en revue ces scripts pour voir à quoi ils servent et leur interdépendance.

La configuration du fichier `/etc/inittab` est telle que:

- `rc.S` est lancé au démarrage quel que soit le niveau d'initialisation
- `rc.K` est lancé en mode mono-utilisateur (niveaux 1 et S)
- `rc.0` est lancé à l'arrêt de l'ordinateur (niveau 0)
- `rc.6` est lancé lors du redémarrage de l'ordinateur (niveau 6)
- `rc.4` est lancé seulement pour le niveau 4
- `rc.M` est lancé aux niveaux multi-utilisateurs 2, 3, 4 et 5

De façon générale, le script `rc.S` lance `rc.modules` pour charger les modules au démarrage et peut lancer le script `rc.serial` qui initialise les ports série (le chargement de `rc.serial` est toutefois commenté par défaut dans `rc.S`).

Tous les autres scripts du répertoire `/etc/rc.d` sont lancés directement ou indirectement par `rc.M` (indirectement parce que certains sont lancés par `rc.inet1` ou `rc.inet2`).

On peut maintenant donner une description plus complète de l'architecture du répertoire.

rc.S

C'est le premier script à être appelé. Il sert à préparer le système avant qu'il n'entre dans le niveau de démarrage désiré. Il monte le système fichier utilisé sur votre système (`ext2`, `ext3`, `reiserfs`, ?), initialise les périphériques Plug&Play, charge les modules du kernel (via le script `rc.modules`), configure les ports série etc ?

Le fichier `rc.S` est le plus imposant et contient aussi deux appels de scripts pour compléter son travail qui sont :

rc.modules

Ce script contient presque tous les modules qui ont été compilés avec le noyau de base, qui est installé avec la distribution. Il utilise la commande `modprobe` pour charger ces modules et l'on pourra donc, en décommentant l'entrée devant celle-ci, charger un module qui nous est nécessaire. Le fichier `/etc/modules.conf` reste malgré tout nécessaire du fait que certaines applications y insèrent des commandes pour charger leurs modules lorsqu'on les installe.

rc.serial

Configure les ports série en activant la commande `setserial` appropriée. Il est lancé par `rc.S` mais il est commenté par défaut

Il y a d'autres fonctions que ce script appelle ainsi un rapide coup d'oeil sur ledit fichier vous en dira plus.

Après que `init` ait fini de compléter son travail avec `rc.S`, il entre dans le mode défini par défaut dans le fichier `/etc/inittab`, soit en mode multi-utilisateur ou mono-utilisateur ; de là il activera les différents scripts dont il a besoin pour le mode spécifié, scripts dont voici un bref aperçu.

rc.0

N'est qu'un lien du script `rc.6`, il a donc la même fonction.

rc.6

Ce script termine tous les processus par les commandes `killall15` puis `killall9`, stoppe les services de quotas et de comptabilité si ceux-ci sont utilisés, démonte les différents systèmes fichiers NFS et autres, désactive le swap, démonte les systèmes fichiers locaux, sauvegarde quelques informations telles que l'heure système.

Enfin, il exécute la commande `reboot` si c'est `rc.6` qui a été appelé directement ou `poweroff` si c'est `rc.0` qui a été demandé (le script effectue un test au départ pour savoir lequel des deux a été

appelé).

On peut modifier ce script par exemple pour qu'il vide le répertoire /tmp en insérant une ligne du type `rm -r -f /tmp/*` juste avant qu'il ne démonte le système de fichiers.

rc.K

Ce script est appelé lorsque `init` passe en mode mono-utilisateur (niveau 1 ou S) et consiste à terminer tous les processus en leur envoyant le signal TERM, à stopper les services quotas et les serveurs et à entrer en mode mono-utilisateur par la commande :

```
telinit - t 1 1
```

rc.4

C'est le script qui est lancé en mode graphique (X Window) et sert uniquement à définir le gestionnaire de connexion (`kdm`, `gdm`, `xdm`). On peut éventuellement le modifier à son goût et y ajouter d'autres fonctions selon les goûts personnels de l'utilisateur.

rc.M

C'est le deuxième script à être lu par `init` puisqu'il définit le mode "niveau démarrage" et initialise les différents scripts se rapportant à ce mode. En fait, comme on l'a déjà dit, il lance tous les autres scripts du répertoire `/etc/rc.d`.

Outre le lancement des scripts, il exécute les commandes suivantes (si celles-ci sont disponibles, c'est-à-dire si les paquets correspondants ont été installés) :

- ◇ efface la console à l'aide de la commande `setterm`
- ◇ attribue le nom de la machine contenu dans le fichier `/etc/HOSTNAME` si celui-ci n'est pas vide ou, dans le cas contraire, copie le nom par défaut (`darkstar.example.net`) dans le fichier et l'affecte à la machine (ainsi, lors de l'installation, si aucun nom n'est attribué, c'est le nom par défaut qui devient actif)
- ◇ efface un certain nombre de fichiers dans `/var/lock`, `/var/spool/uucp` et `/tmp`
- ◇ actualise les bibliothèques avec `ldconfig`
- ◇ actualise l'index des fontes Freetype avec `fc-cache`
- ◇ lance la commande `accton`
- ◇ lance le démon `crond` pour exécuter des tâches périodiques
- ◇ lance le démon `atd` pour exécuter des tâches à des moments donnés
- ◇ active le système de quotas
- ◇ lance le démon `apmd`

D'autres fonctions sont activées par `rc.M` à partir des scripts du répertoire `/etc/rc.d`. Il convient de noter que ces scripts ne seront lancés que s'ils sont exécutables.

Ces scripts ont été classés ici en trois rubriques : réseau, matériel et divers.

Initialisation réseau

rc.inet1

Il est chargé de mettre en place les interfaces réseau notamment en chargeant le module de la ou des cartes réseau, en configurant les interfaces à l'aide de la commande `ifconfig` et en définissant un routage avec la commande `route`.

Même si vous n'êtes pas connecté par une carte réseau, il faut que ce script soit lancé si vous souhaitez activer la fonction de rebouclage (`loopback`) utile si vous voulez faire tourner Apache pour tester vos pages Web.

Ce script exécute le script `rc.inet1.conf` qui contient les paramètres de configuration du réseau, par exemple l'adresse IP ou le masque de sous-réseau (il faut donc rendre ce script exécutable si l'on veut que les paramètres réseau soient pris en compte).

rc.inet2

Admin-admin boot-slack init

Il est exécuté après `rc.inittl`, monte les systèmes de fichiers NFS et SMBFS et démarre les différents services de base qui sont listés ci-après :

- ◇ **rc.portmap** : nécessaire pour monter les partitions NFS
- ◇ **rc.syslog** : il lance les démons `syslogd` et `klogd` pour les logs du système
- ◇ **rc.firewall** : il met en place un pare-feu
- ◇ **rc.ip_forward** : pour la redirection des paquets si la machine est configurée comme routeur
- ◇ **rc.inetd** : il lance le super démon `inetd` qui permet de lancer d'autres démons (comme le serveur ftp)
- ◇ **rc.sshd** : pour démarrer le serveur SSH
- ◇ **rc.bind** : démarre le service de noms BIND
- ◇ **rc.ypp** : pour le service NIS
- ◇ **rc.nfsd** : démarre le serveur NFS

rc.httpd

Il démarre le serveur Apache

rc.sendmail

Il démarre le serveur de mail

rc.mysld

Il démarre le démon serveur de la base de données MySQL (ce n'est pas à proprement parler un élément réseau mais il est souvent associé à Apache)

rc.atalk

Il démarre le serveur AppleTalk

rc.samba

Il démarre le serveur Samba

Configuration matérielle

rc.pcmcia

Recherche et configure tous les périphériques PCMCIA qui se trouvent sur la machine. Ceci est plus intéressant pour les utilisateurs de portables qui ont un modem ou une carte réseau de ce type.

rc.hotplug

Il initialise les périphériques "hotplug".

rc.cups

Il permet de lancer le démon d'impression Cups

rc.lprng

Il permet de lancer le démon d'impression Lpd. Attention de ne rendre exécutable qu'un seul des deux scripts pour l'imprimante.

rc.acpid

Il démarre le démon de gestion de l'alimentation ACPI.

rc.alsa

Il charge les modules de son ALSA et initialise le mixer avec les anciens paramètres

rc.keymap

Il charge la configuration du clavier pour la console.

rc.gpm

Il charge le serveur de la souris pour la console, GPM

Divers

rc.font

Charge les fontes personnalisées en mode console. Au départ, il est présent sous le nom `rc.font.sample` et il faut donc le renommer en `rc.font` pour qu'il soit lancé.

rc.sysvinit

Il a été inséré à partir de la version 7.0 de la Slackware pour palier à un problème : certains logiciels avaient besoin d'installer des scripts de démarrage nécessaires à leur fonctionnement et ce dans le mode System V.

Ce script peut aussi être utilisé en créant des sous-répertoires de niveau de démarrage (`rcn.d`) et en y installant les scripts que l'on désire ; ceci pour ceux qui aimeraient mieux le System V pour insérer de futurs scripts s'ils sont plus à l'aise avec ce genre d'initialisation.

On peut même se servir de scripts pris sur une autre distribution en prenant soin de bien vérifier les chemins des différentes commandes utilisées par ces scripts en fonction de ceux utilisés par le système.

rc.local

Il est destiné à contenir les différentes commandes personnelles à l'utilisateur qu'il veut initialiser lors du démarrage du système. Ce script est le dernier appelé après que tous les autres aient été lancés.

Notes

La plupart de ces scripts peuvent être démarrés avec l'un des trois paramètres `start`, `stop` ou `restart`. Par exemple, pour arrêter un service lancé par un script `rc.fonction`, il suffit de faire :

```
/etc/rc.d/rc.fonction stop.
```

Par ailleurs, si l'on souhaite qu'un script ne soit pas exécuté au démarrage, plusieurs méthodes sont possibles. Une première consiste à le détruire mais ce n'est pas conseillé dans la mesure où vous pouvez en avoir besoin par la suite (et qu'il prend peu de place).

Une deuxième méthode consiste à le rendre non exécutable par la commande `chmod -x rc.fonction` (`chmod +x rc.fonction` pour le rendre exécutable).

Une troisième méthode consiste à en commenter l'entrée dans un des scripts qui le lance (il s'agit souvent de `rc.M` comme on l'a vu).

Par exemple, on aura généralement des lignes du type :

```
if [ -x /etc/rc.d/rc.fonction ]; then
    /etc/rc.d/rc.fonction
fi
```

Il suffit alors d'ajouter le signe `#` devant la ligne `/etc/rc.d/rc.fonction` pour qu'elle soit commentée. Cette méthode se révèle très pratique si l'on souhaite utiliser un programme de temps en temps sans qu'il soit lancé systématiquement (par exemple, si l'on utilise `mysql` périodiquement). En effet, puisqu'il reste exécutable, il suffit de le lancer comme indiqué précédemment.

L'ouverture d'une session

Si le système est lancé au niveau 4, un des programmes `gdm`, `kdm` ou `xdm` est lancé par le script `rc.4`. Le nom de login de l'utilisateur et son mot de passe sont ensuite vérifiés et une session X est ouverte.

Mais c'est surtout le niveau 3 qui nous intéresse ici.

Dans ce cas, c'est le programme `/sbin/agetty` qui est lancé. Il appelle ensuite le programme `/bin/login` qui lance enfin l'interpréteur de commandes `Bash`. Nous allons entrer un peu dans le détail.

Le programme agetty

Il est lancé pour les niveaux d'initialisation 1 à 6 comme indiqué dans le fichier `/etc/inittab`. Il effectue les opérations suivantes:

- il affiche le contenu du fichier `/etc/issue` qui correspond à une ligne du type "Welcome to Linux 2.4.22 (tty1)" (voir man `agetty` pour une description des codes d'échappement utilisés dans ce fichier)
- il demande le nom de connexion en affichant une ligne du type "nom_hôte login:"
- il appelle ensuite le programme `/bin/login`

Le programme login

Les paramètres de connexion sont définis dans le fichier `/etc/login.defs`. Par exemple, si l'on souhaite que des caractères '*' apparaissent lorsque l'on saisit le mot de passe, il faut décommenter la ligne `GETPASS_ASTERISKS 1`. Les différents paramètres possibles sont commentés dans ce fichier et il peut être utile de le consulter si l'on souhaite les modifier. On peut également faire man `login.defs` pour obtenir quelques précisions sur sa rédaction. Le programme `login` présente les caractéristiques suivantes:

- il demande le mot de passe en affichant la ligne "Password:" puis vérifie sa validité en le comparant à celui qui est contenu, de manière codée, dans le fichier `/etc/shadow`
- si le mot de passe n'est pas valide, il réaffiche une ligne du type "nom_hôte login:" pour demander un nom de connexion
- comme précisé dans le fichier `login.defs`, le nombre de tentatives de connexion est fixé à 5 par défaut mais, comme l'action dans le fichier est `respawn`, au bout du cinquième échec, le système ferme le port `tty` et le réouvre en relançant le programme `agetty`
- si tout est correct, il lit le fichier `/etc/passwd` qui contient les informations sur les utilisateurs (par exemple, leur Id, leur répertoire etc) et ouvre une session
- il affiche alors le contenu du fichier `/etc/motd`, en l'occurrence la chaîne "Linux 2.4.22."
- il affiche ensuite des lignes indiquant le nombre d'échecs de connexion, des informations sur la dernière connexion
- il informe enfin l'utilisateur s'il a reçu du courrier en regardant le contenu du fichier `/var/spool/mail/nom_connexion` (le répertoire est défini dans le fichier `login.defs`)
- pour qu'il n'affiche aucun message après le mot de passe, il faut créer un fichier vide nommé `.hushlogin` dans le répertoire utilisateur, le nom étant encore défini dans le fichier `login.defs` (il est inutile de placer quelque chose dans ce fichier car il n'est pas lu)
- pour finir, le programme `login` lance le shell défini dans le fichier `/etc/passwd`, par défaut il s'agit du programme `/bin/bash`

Le programme Bash

Comme l'indique la commande `man bash`, celui-ci, lors d'une connexion, lit le fichier `/etc/profile` puis, s'ils existent, le premier des trois fichiers du répertoire utilisateur `.bash_profile`, `.bash_login` et `.profile` (il utilise également le fichier `.bashrc` mais celui-ci n'est lu que s'il ne s'agit pas d'un shell de connexion, par exemple, l'ouverture d'un terminal dans une session X).

Par défaut, seul le fichier `/etc/profile` est présent et c'est le plus important car il définit les variables d'environnement. C'est donc son contenu que nous allons en partie décrire.

Il définit entre autres les variables suivantes :

- `MANPATH` pour les pages de manuel
- `HOSTNAME`, initialisée au contenu du fichier `/etc/hostname` pour le nom de la machine
- `INPUTRC`, égale par défaut à `/etc/inputrc` pour le comportement du clavier, notamment dans un terminal
- `PATH`, c'est-à-dire le chemin par défaut pour les programmes et commandes (si on lance une commande ou un programme, il est recherché dans tous les répertoires du `PATH` et, s'il n'est pas

trouvé, une erreur est générée)

- LC_ALL pour les paramètres régionaux et qui remplace toutes les autres variables du type LC_*
- PS1 pour le prompt du shell qui, par défaut, est un message du genre
nom_utilisateur@nom_machine:~\$ (faire man bash pour savoir comment le redéfinir)
- umask pour les droits par défaut des fichiers et répertoires nouvellement créés (il est fixé par défaut à 022)

Il effectue également les opérations suivantes:

- il lance la commande eval 'dircolors -b' qui lit le fichier /etc/DIR_COLORS pour les options de la commande ls
- il exécute tous les scripts d'extension .sh du répertoire /etc/profile.d ; il s'agit en général de définition de variables propres à certains programmes comme Java ou Latex

Conclusion

Voici donc résumés les principes essentiels de l'initialisation de la Slackware qui vous permettront, nous l'espérons, d'approfondir vos connaissances de cette distribution et de la personnaliser selon vos besoins. Vous pouvez obtenir des informations supplémentaires en parcourant le Net mais la plupart de celles-ci sont probablement présentes sur votre machine, que ce soit avec les pages de manuel ou dans les Howto situés dans le répertoire /usr/doc qui sont trop souvent méconnus.

Cette page est issue de la documentation 'pré-wiki' de Léa a été convertie avec HTML::WikiConverter. Elle fut créée par Michel Lalumière le 25/09/2001.

Copyright

Copyright © 25/09/2001, Michel Lalumiere



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