

Table des matières

1	PROCESSUS DE DÉMARRAGE ET D'ARRÊT	2
1.1	PROGRAMME INIT.....	2
1.2	FICHER DE CONFIGURATION /etc/inittab	3
1.3	LES NIVEAUX DE FONCTIONNEMENT (RUNLEVELS).....	7
1.4	SCRIPTS DE DÉMARRAGE.....	7
1.4.1	/etc/rc.d/rc.sysinit.....	8
1.4.2	/etc/rc.d/init.d	8
1.4.3	/etc/rc.d.....	9
1.4.4	/etc/rc.d/rc[0-6].d	10
1.5	ARRÊT DU SYSTÈME	12
1.6	CONTRÔLE DU PROCESSUS INIT	15
1.7	AJOUT ET SUPPRESSION DE SERVICE AU DÉMARRAGE.....	17
1.8	MESSAGES.....	20

1 PROCESSUS DE DÉMARRAGE ET D'ARRÊT

La procédure d'initialisation suit celle des Unix "System V" et utilise le concept de niveaux d'exécution "runlevels" pour définir les services devant être lancés au démarrage.

Une fois son chargement terminé, le noyau Linux lance le processus Init (*/sbin/init*) qui prend en charge la fin de l'initialisation du système ; étant le premier processus lancé, il possède le PID 1 (Process ID).

Le processus Init a pour mission d'initialiser l'environnement logiciel du système et de lancer les services.

Les tâches d'initialisation comprennent :

- La mise à l'heure du système par rapport à l'heure matérielle
- La définition des consoles
- La définition du nom de la machine
- L'activation des quotas
- L'activation du swap
- La vérification des systèmes de fichiers et le montage de ceux-ci
- Remontage du système de fichiers racine qui était monté en lecture seule
- Le lancement du démon Syslog de journalisation des messages
- Le chargement des modules du noyau
- L'initialisation des variables d'environnement (PATH, HOME,...).

Les services démarrés dépendront du niveau d'exécution spécifié au processus Init

1.1 PROGRAMME INIT

Seul et unique processus lancé par le Kernel, Init a pour tâche de lancer chacun des processus, y compris les différents démons et les sessions de *login*. Il doit aussi collecter les zombies (processus orphelins) et gérer l'arrêt du système. Au démarrage, le Kernel charge les pilotes de périphériques pour lesquels il a été configuré puis il lance */sbin/init*, lequel prend en charge la suite des opérations. Par la suite le Kernel n'interviendra plus que pour répondre aux appels système. La plupart des distributions procurent un Init écrit par Miquel van Smoorenburg, lequel est similaire dans son approche au programme Init d'Unix System V.

Ainsi Init est le premier processus lancé, il est le père de tous les autres processus.

- Le programme Init est hautement configurable.
- Le programme Init de RedHat est compatible avec Unix système V.
- Unix système V utilise les niveaux d'exécution (runlevels) contrairement à Unix BSD (Berkeley Software Distribution).

- Le programme Init étant le premier à être lancé par le noyau, son process ID (PID) est 1.
- A son lancement le programme Init lit le fichier de configuration **/etc/inittab** qui détermine le comportement de Init.

1.2 FICHIER DE CONFIGURATION /etc/inittab

Pour définir le comportement du système et les services à démarrer en fonction du niveau d'exécution, Init se réfère au fichier **/etc/inittab**.

Le fichier **/etc/inittab** est un fichier texte composé de lignes de commentaires (commençant par #) et d'entrées constituées de 4 champs délimités par des deux-points, elles sont de la forme :

```
id:runlevels:action:process [arguments]
```

- **id**
Identifiant unique qui comprend de 1 à 4 caractères alphanumériques qui identifie l'entrée dans le fichier **/etc/inittab**.
- **runlevels**
Défini la liste des niveaux d'exécution pour lesquelles cette ligne s'applique. Si cette liste est vide, cela équivaut à indiquer tous les niveaux.
- **action**
Méthode ou manière d'exécuter la commande spécifiée dans le champ suivant.
- **process**
Chemin de la commande à lancer avec ses paramètres pour les niveaux d'exécution définis précédemment.

Les différentes directives pour le troisième champ "**action**" sont :

respawn

Le processus sera relancé s'il se termine. Utilisé principalement avec **mingetty** pour assurer la gestion des terminaux texte :

```
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

Indique qu'à tous les niveaux multi-utilisateurs (2 à 5), la commande `mingetty` doit être relancée sur les terminaux `tty1` à `tty6` lorsqu'elle se termine (afin de pouvoir se loguer, se déloguer et se reloguer).

once

Le processus n'est exécuté qu'une seule fois.

```
x:5:once:/etc/X11/prefdm nodaemon
```

wait

Identique à la directive précédente mais ici, Init attend que le processus soit terminé avant de passer à la ligne suivante.

Ce type d'entrée sert principalement à lancer les scripts d'initialisation pour chaque niveau. Exemples :

```
10:0:wait:/etc/rc.d/rc 0
11:1:wait:/etc/rc.d/rc 1
12:2:wait:/etc/rc.d/rc 2
13:3:wait:/etc/rc.d/rc 3
14:4:wait:/etc/rc.d/rc 4
15:5:wait:/etc/rc.d/rc 5
16:6:wait:/etc/rc.d/rc 6
```

Indique qu'il faut lancer le script `/etc/rc.d/rc n`.

boot

Le processus est exécuté au démarrage du système, le champ `runlevels` est ignoré.

bootwait

Identique à la directive précédente mais ici, Init attend que le processus soit terminé avant de passer à la ligne suivante.

initdefault

Définit le niveau de d'exécution par défaut au démarrage du système, le champ `runlevels` est ignoré. Si cette directive est absente, Init le demandera sur la console. Le champ `commande` est ignoré.

sysinit

Le processus doit s'effectuer au démarrage du système, avant toute entrée d'action **boot** ou **bootwait**. Le champ `runlevels` est ignoré.

Exemple :

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

off

Ne fait rien. Cela revient à commenter la ligne.

ondemand

Identique à **respawn** mais cette directive utilise les pseudos runlevels a, b et c. Cela permet de demander à Init d'entreprendre une action sans changer de niveau d'exécution.

powerfail

La commande est exécutée lorsque Init reçoit le signal SIGPWR signifiant que l'alimentation est sur le point d'être interrompue. Ce signal est envoyé par un dispositif de gestion d'énergie tel qu'un UPS.

Exemple :

```
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"
```

powerwait

Identique à la directive précédente mais ici, Init attend que le processus soit terminé avant de passer à la ligne suivante.

powerokwait

Le processus est exécuté si Init est informé du rétablissement de l'alimentation.

powerfailnow

Le processus est exécuté si Init est informé que l'accumulateur de l'UPS externe est presque vide.

ctrlaltdel

La commande est lancée lorsque Init reçoit le signal **SIGINT** (généré par la combinaison de touches **CTRL-ALT-DEL**). Elle sert généralement à arrêter et/ou redémarrer le système.

Exemple :

```
ca::ctrlaltdel:/sbin/shutdown -t3 -h now
```

kbrequest

Permet de lancer un processus suivant certaines séquences de touches saisies au clavier.

L'analyse du fichier de configuration */etc/inittab* nous apprend que :

- Le niveau d'exécution par défaut est le niveau 5. Mettre cette valeur à 3 si on ne veut pas lancer systématiquement le système X Window à chaque démarrage; il faudra alors saisir la commande *startx* une fois connecté au système pour lancer l'interface graphique.
- La commande */etc/rc.d/rc.sysinit* est lancée en premier lieu.
- La commande */etc/rc.d/rc* est lancée pour chaque niveau d'exécution avec son numéro en argument.
- La commande */sbin/shutdown* est exécutée avec différents arguments lors de l'appui sur les touches **CTRL-ALT-DEL** et lors d'un événement concernant l'alimentation.
- Six terminaux virtuels "tty" sont initialisés avec la commande */sbin/mingetty* pour les niveaux d'exécution 2, 3, 4 et 5. De plus, ceux-ci seront relancés s'ils se terminent.
- Pour le niveau d'exécution 5, l'outil */etc/X11/prefdm* est exécuté ; cette commande lance le programme de connexion à la console graphique.

1.3 LES NIVEAUX DE FONCTIONNEMENT (RUNLEVELS)

Il existe huit runlevels définis sous Linux dont quatre réservés (0, 1, 6, S ou s) :

NIVEAU	DESCRIPTION
0	Halt - Arrêt de la machine.
1	Single user- Mode mono-utilisateur ou maintenance.
2	Multi-utilisateurs sans le support NFS.
3	Multi-utilisateurs.
4	Libre.
5	X11 (Multi-utilisateurs avec graphique).
6	Reboot - redémarrage de la machine.
S ou s	Mode mono-utilisateur dans lequel seule la partition racine est montée.

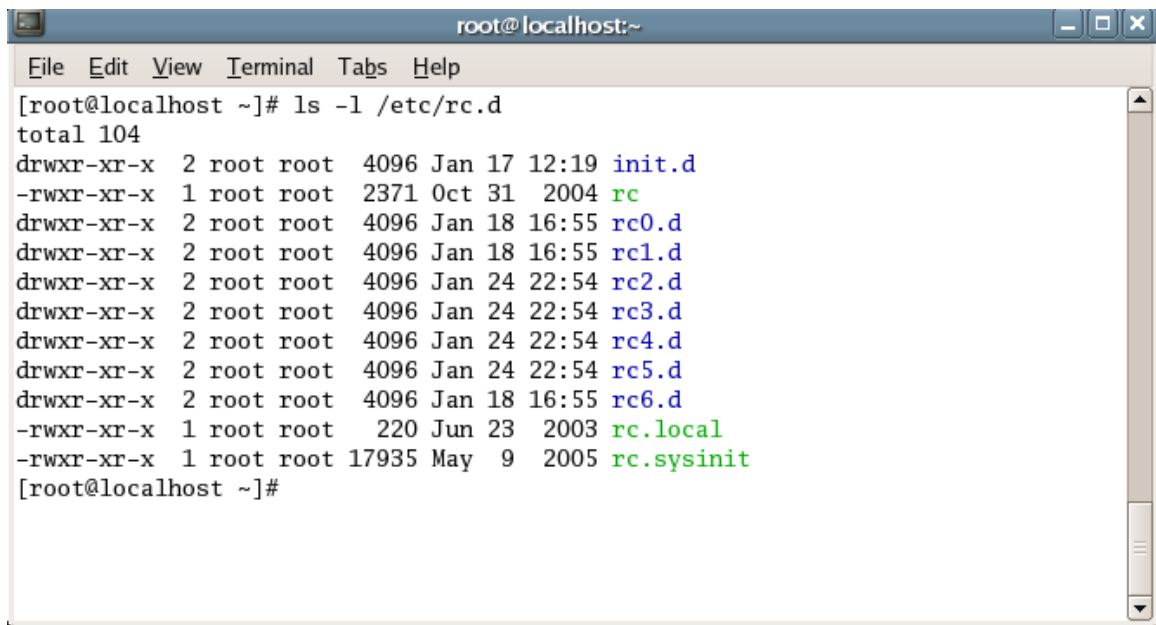
Les niveaux d'exécution non réservés sont configurables à souhait. L'administrateur pourra sélectionner les services associés à chaque runlevel.

Lors de la phase de démarrage, Init doit savoir à quel niveau placer le système. Pour cela, il recherche dans le fichier `/etc/inittab` la ligne qui configure ce niveau par défaut :

```
id:5:initdefault:
```

Signifie que le système démarrera en Mode multi-utilisateurs complet.

1.4 SCRIPTS DE DÉMARRAGE



```
root@localhost:~  
File Edit View Terminal Tabs Help  
[root@localhost ~]# ls -l /etc/rc.d  
total 104  
drwxr-xr-x  2 root root  4096 Jan 17 12:19 init.d  
-rwxr-xr-x  1 root root  2371 Oct 31  2004 rc  
drwxr-xr-x  2 root root  4096 Jan 18 16:55 rc0.d  
drwxr-xr-x  2 root root  4096 Jan 18 16:55 rc1.d  
drwxr-xr-x  2 root root  4096 Jan 24 22:54 rc2.d  
drwxr-xr-x  2 root root  4096 Jan 24 22:54 rc3.d  
drwxr-xr-x  2 root root  4096 Jan 24 22:54 rc4.d  
drwxr-xr-x  2 root root  4096 Jan 24 22:54 rc5.d  
drwxr-xr-x  2 root root  4096 Jan 18 16:55 rc6.d  
-rwxr-xr-x  1 root root   220 Jun 23  2003 rc.local  
-rwxr-xr-x  1 root root 17935 May  9  2005 rc.sysinit  
[root@localhost ~]#
```

Un certain nombre de commandes exécutées par Init se trouvent dans */etc/rc.d* :

1.4.1 */etc/rc.d/rc.sysinit*

Le programme Init exécute le script */etc/rc.d/rc.sysinit* avant tous les autres.

Dans le fichier */etc/initab* on retrouve la ligne suivante:

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

Dans cette ligne, l'information sur le niveau d'exécution n'est pas spécifiée car Init reconnait *sysinit* comme une action d'initialisation du système.

Ce script permet entre autre de:

- initialiser le hostname
- activer la partition swap
- vérifier l'intégrité du système de fichier
- charger les modules du noyau

1.4.2 */etc/rc.d/init.d*

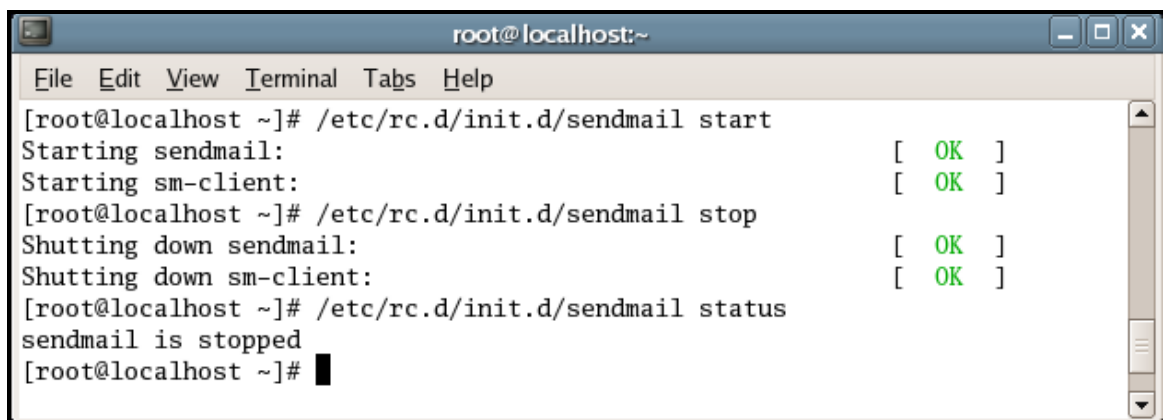
Ce répertoire contient les scripts Shell permettant de lancer tous les services sur le système.

Maintenir tous les scripts dans une seule location facilite grandement leur gestion.

Chaque script supporte l'argument **start** pour lancer le service et **stop** pour l'arrêter. Une grande majorité de ces scripts supportent aussi les arguments **restart**, **reload** et **status**.

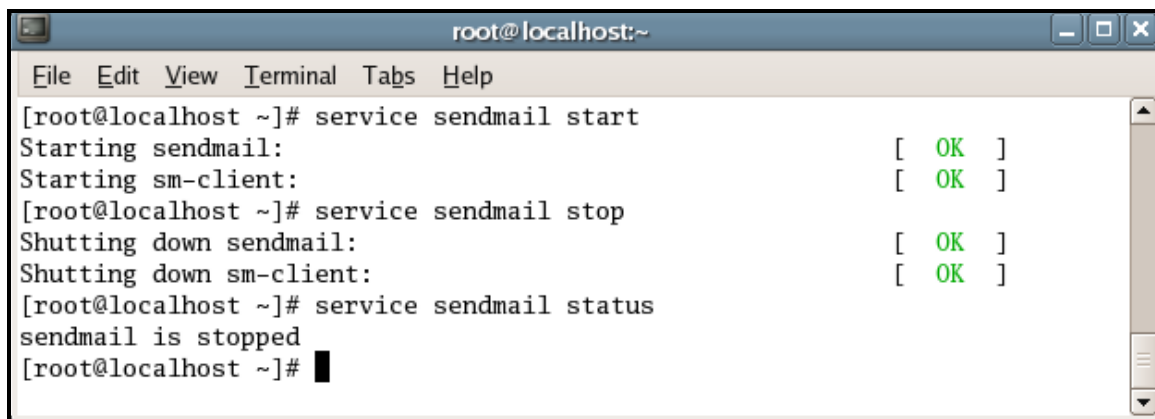
Chaque script est utilisé pour démarrer ou arrêter un service particulier.

Par exemple pour le service *sendmail*:



```
root@localhost:~  
File Edit View Terminal Tabs Help  
[root@localhost ~]# /etc/rc.d/init.d/sendmail start  
Starting sendmail: [ OK ]  
Starting sm-client: [ OK ]  
[root@localhost ~]# /etc/rc.d/init.d/sendmail stop  
Shutting down sendmail: [ OK ]  
Shutting down sm-client: [ OK ]  
[root@localhost ~]# /etc/rc.d/init.d/sendmail status  
sendmail is stopped  
[root@localhost ~]#
```

Sur RedHat, la commande *service* permet de s'abstraire du chemin désignant le script ; la commande suivante est donc équivalente à la précédente :

A terminal window titled 'root@localhost:~' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the following commands and outputs:

```
[root@localhost ~]# service sendmail start
Starting sendmail: [ OK ]
Starting sm-client: [ OK ]
[root@localhost ~]# service sendmail stop
Shutting down sendmail: [ OK ]
Shutting down sm-client: [ OK ]
[root@localhost ~]# service sendmail status
sendmail is stopped
[root@localhost ~]#
```

1.4.3 /etc/rc.d

Le répertoire */etc/rc.d* contient 3 fichiers exécutables : *rc*, *rc.local* et *rc.sysinit*.

rc, la commande */etc/rc.d/rc* est lancée pour chaque niveau d'exécution avec son numéro en argument.

Elle est appelée par Init avec un paramètre correspondant au niveau d'exécution dans lequel on entre. Ce paramètre est récupéré par le script **rc** pour former un nom de répertoire. Par exemple, l'appel */etc/rc.d/rc 2* fera que **rc** ira examiner le répertoire */etc/rc.d/rc2.d*. Il exécute d'abord les scripts de ce répertoire dont le nom commence par **K**, puis ceux dont le nom commence par **S**. Le script **rc** joue un rôle d'aiguillage et d'exécution d'autres scripts.

Par exemple, la ligne :

```
12:2:wait:/etc/rc.d/rc 2
```

Signifie que Init doit lancer le script */etc/rc.d/rc* en lui passant 2 en paramètre à chaque fois qu'on entre dans le niveau 2 et qu'il doit attendre la terminaison de ce script avant de poursuivre.

rc.local, comme son nom l'indique est "local" à votre machine : c'est dans ce fichier que vous ajouterez les commandes d'initialisation propres à votre système. **rc.local** est appelé en dernier, c'est-à-dire après tous les autres scripts.

Dans les niveaux 2,3 et 5 il porte le nom: **S99local** qui est un lien symbolique vers */etc/rc.d/rc.local*

Ce script peut être utilisé par l'administrateur pour exécuter certaines tâches une seule fois à la fin du niveau d'exécution correspondant. Par exemple après chaque démarrage du serveur on voudra démarrer la base de données.

rc.sysinit, est le script qui est lancé lors de l'initialisation du système, il sert à configurer la variable d'environnement PATH au niveau du système, à activer le swap, à configurer le nom de la machine avec la commande hostname, à lancer les commandes de vérification des systèmes de fichiers, à les monter, à activer les modules si ceux-ci sont utilisés, etc. Parcourez-le, cela vous donnera envie de connaître la syntaxe du Shell.

1.4.4 /etc/rc.d/rc[0-6].d

Ces répertoires contiennent des liens vers les scripts du répertoire */etc/rc.d/init.d* à lancer lorsque le processus Init entre dans le niveau d'exécution correspondant au numéro du répertoire. Le nom des liens est important car il détermine la façon dont ils seront appelés.

Le script **rc** lancera les scripts contenus dans le répertoire correspondant au niveau d'exécution passé en argument et exécute chaque script qui y figure en lançant en premier ceux commençant par la lettre **K (Kill)** avec l'argument **stop**, puis ceux débutant par lettre **S (Start)** avec l'argument **start**. Le numéro suivant la lettre indique l'ordre d'exécution. Si deux fichiers ont le même numéro, l'ordre alphabétique primera.

Cette gestion des scripts de démarrage peut paraître complexe, mais elle permet de centraliser dans un même script le contrôle complet d'un service, tout en autorisant plusieurs niveaux d'exécution.

Les répertoires */etc/rc.d/rc0.d* et */etc/rc.d/rc6.d* sont particuliers puisqu'ils correspondent respectivement à l'entrée dans le niveau de mise hors service et dans le niveau de redémarrage. L'examen du premier vous montrera qu'il ne comporte que de liens **K** et un seul lien **S** : **S00halt**. La mise hors service doit commencer par arrêter tous les services en cours, puis lancer la routine d'arrêt du système. De même, rc6.d contient des liens **K** et un lien vers le script de redémarrage (**S00reboot**).

En fait, cette méthode peut sembler lourde alors qu'elle est finalement d'une simplicité à toute épreuve : à part rc.local, tous les scripts sont regroupés dans le répertoire */etc/rc.d/init.d* et on décide dans le répertoire associé au niveau n (*/etc/rc.d/rcn.d*) des scripts qu'il faut lancer et de ceux qu'il faut arrêter...

Résumé

Les répertoires */etc/rc.d/rc0.d* jusqu'à */etc/rc.d/rc6.d* sont utilisés pour les niveaux d'exécution **0** à **6** respectivement.

La syntaxe des scripts est la suivante:

K[numéro] [nom du script] ou bien S[numéro] [nom du script]

avec 'numéro': 2 chiffres.

Tous les scripts préfixés par la lettre '**K**' sont exécutés avec l'argument '**stop**'

Tous les scripts préfixés par la lettre '**S**' sont exécutés avec l'argument '**start**'

'numéro' détermine l'ordre d'exécution. Par exemple le script '**K20nfs**' est exécuté avant '**K30sendmail**'

Bien entendu les scripts '**K**' sont exécutés avant les scripts '**S**'

Pour chaque niveau d'exécution, Init exécute un script spécifique comme indiqué dans lignes suivantes du fichier */etc/inittab*:

```
10:0:wait:/etc/rc.d/rc 0
11:1:wait:/etc/rc.d/rc 1
12:2:wait:/etc/rc.d/rc 2
13:3:wait:/etc/rc.d/rc 3
14:4:wait:/etc/rc.d/rc 4
15:5:wait:/etc/rc.d/rc 5
16:6:wait:/etc/rc.d/rc 6
```

Pour chaque niveau d'exécution le script */etc/rc.d/rc* est exécuté avec le niveau d'exécution comme argument (0-6). Ce script est responsable du démarrage et de l'arrêt de tous les services du niveau spécifié.

Par exemple au niveau d'exécution 3. Init lance le script:

```
/etc/rc.d/rc 3
```

Le script */etc/rc.d/rc* permet principalement de faire les tâches suivantes:

Vérifie si le répertoire du niveau d'exécution spécifié comme argument existe. En d'autres termes si le script est exécuté avec l'argument 3, il vérifie si le répertoire */etc/rc.d/rc3.d* existe.

Le script détermine si des services sont actifs. Dans ce cas il les arrête en exécutant les scripts "**K**" du niveau d'exécution spécifié avec l'argument "**stop**".

Le script exécute tous les scripts "**S**" du niveau d'exécution spécifié avec l'argument "**start**".

1.5 ARRÊT DU SYSTÈME

Pour éviter toute mauvaise surprise comme la perte de données. Il faut exécuter un certain nombre de tâches avant de couper le courant :

- Prévenir les utilisateurs connectés au système de l'arrêt imminent de la machine.
- Arrêter tous les services.
- Vider les buffers sur le disque et démonter les systèmes de fichiers. Sans cela, les données se trouvant dans le cache et non encore écrites sur les unités de stockage seront perdues.

shutdown

La commande **shutdown** permet d'arrêter, de redémarrer et de passer le système en mode maintenance. Elle offre la possibilité de programmer cette opération à une date précise et d'en informer les utilisateurs. C'est l'arrêt du système est prévu dans moins de cinq minutes, la commande **shutdown** empêchera tout utilisateur, autre que **root** de se connecter.

Sa syntaxe est la suivante :

```
/sbin/shutdown [-t sec] [-arkhncfF] heure [messages]
```

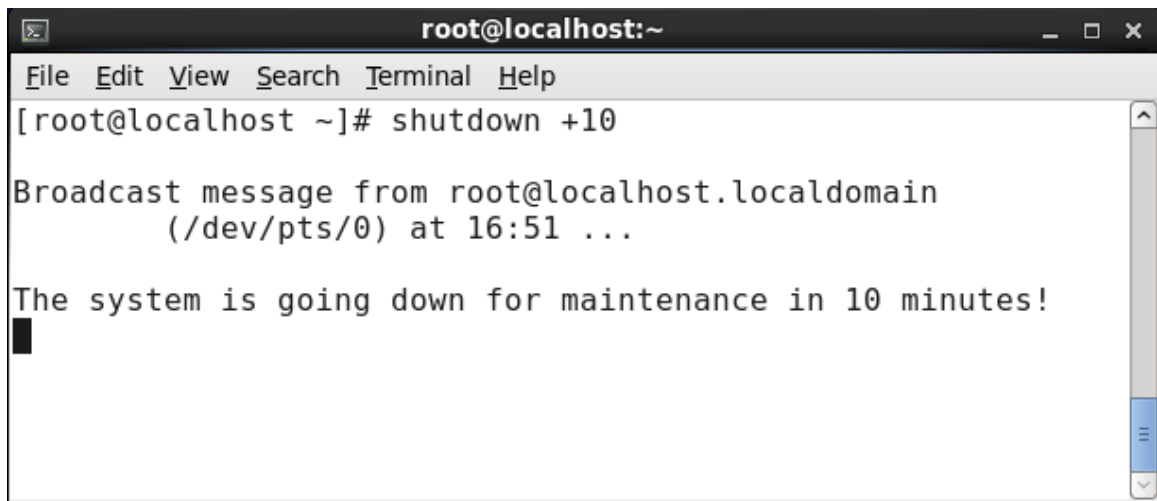
L'heure peut être spécifiée de plusieurs manières :

hh:mm heure à laquelle l'opération est programmé
[+]m nombre de minutes avant que l'opération soit effectuée
now l'opération doit être immédiate (alias de **+0**)

Les options à retenir sont :

-h (halt) arrêter le système.
-r (reboot) redémarrer le système.
-c (cancel) annuler l'opération d'arrêt ou de redémarrage programmée.
-f effectuer un redémarrage rapide sans vérification des systèmes de fichiers.
-F forcer la vérification des systèmes de fichiers au prochain démarrage.

Par exemple la commande suivante permet de mettre le système en mode maintenance dans exactement 10 minutes.

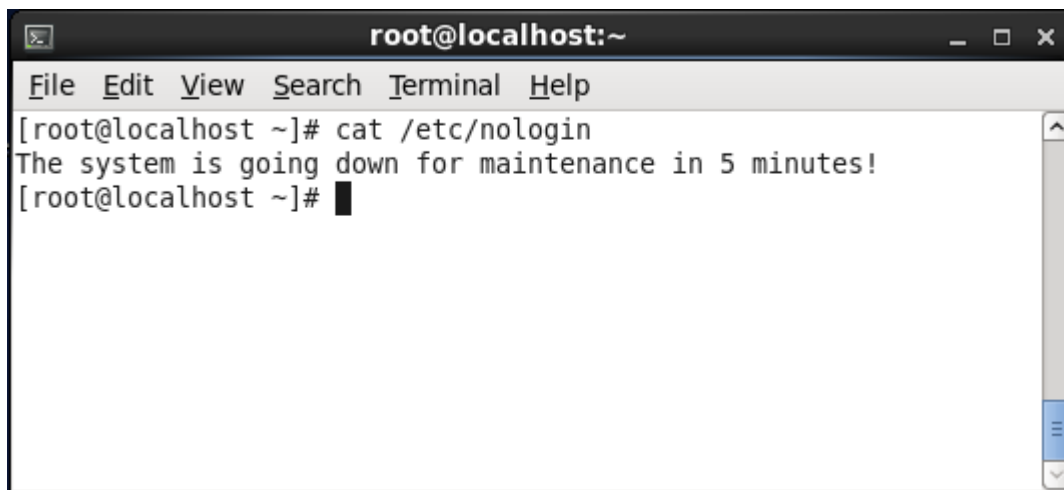


```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]# shutdown +10  
  
Broadcast message from root@localhost.localdomain  
(/dev/pts/0) at 16:51 ...  
  
The system is going down for maintenance in 10 minutes!  
█
```

Pour annuler un shutdown l'option `-c` (cancel) est utilisée.

```
shutdown -c
```

Lors d'un shutdown différé, le fichier `/etc/nologin` est créé 5 minutes avant le shutdown. Ceci a pour effet d'interdire toute nouvelle connexion non root.



```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]# cat /etc/nologin  
The system is going down for maintenance in 5 minutes!  
[root@localhost ~]# █
```

```
CentOS Linux release 6.0 (Final)
Kernel 2.6.32-71.29.1.el6.i686 on an i686

localhost login: root
Password:
Last login: Sat Oct 22 16:54:10 on tty3
[root@localhost ~]# shutdown -c "Arret du systeme annule"
[root@localhost ~]#
Broadcast message from root@localhost.localdomain
(/dev/tty3) at 16:55 ...

Arret du systeme annule

[root@localhost ~]# _
```

halt, reboot et poweroff

Les commandes d'arrêt **halt**, **reboot** et **poweroff** sont utilisées dans le dernier script lancé des niveaux d'exécution 0 et 6.

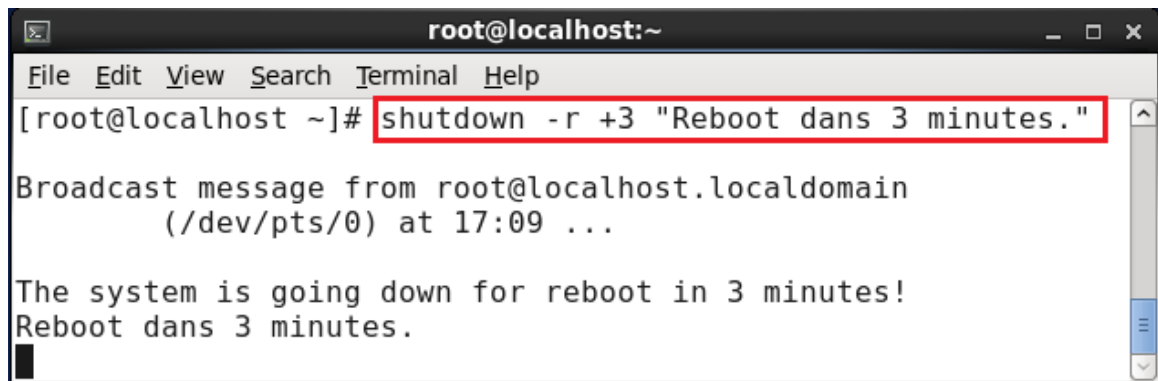
halt synchronise les disques avec les buffers, met à jour le fichier `/var/log/wtmp` (qui contient un historique des utilisateurs connectés, visible à l'aide de la commande **last**) et arrête le système.

reboot est identique à **halt** sauf pour le dernier point où la commande redémarre le système au lieu de l'arrêter.

poweroff est identique à **halt** mais tente en plus d'éteindre l'alimentation électrique de la machine si la carte mère de celle-ci est dotée de cette fonctionnalité.

Les deux dernières commandes sont en fait un lien vers la première.

La commande **shutdown** permet de faire un reboot en utilisant l'option `-r` (reboot)



```
root@localhost:~
File Edit View Search Terminal Help
[root@localhost ~]# shutdown -r +3 "Reboot dans 3 minutes."
Broadcast message from root@localhost.localdomain
(/dev/pts/0) at 17:09 ...

The system is going down for reboot in 3 minutes!
Reboot dans 3 minutes.
```

Ctrl+Alt+Delete permet également de rebooter un système à partir d'une console.

Ceci est possible grâce à l'entrée suivante dans le fichier */etc/inittab*:

```
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

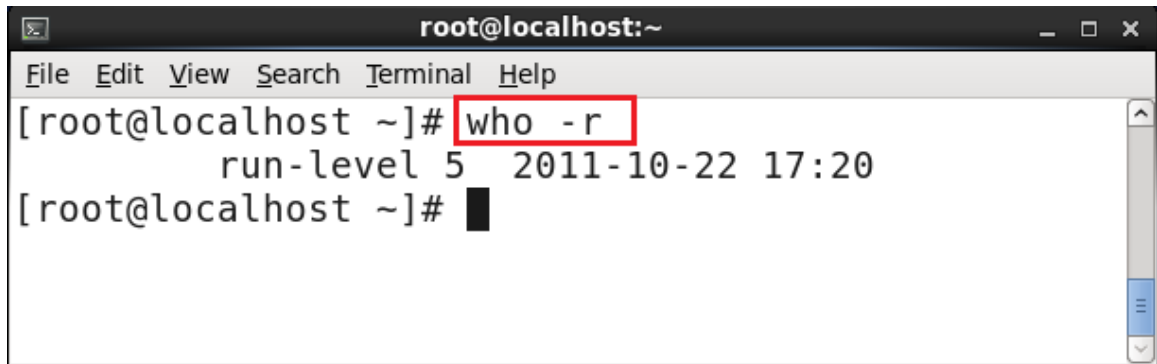
N'importe quel utilisateur qui a un accès physique à une console peut rebooter le système en actionnant les touches Ctrl+Alt+Delete.

Si on veut désactiver Ctrl+Alt+Delete pour tous les utilisateurs incluant root, il faut mettre en commentaire l'entrée correspondante dans */etc/inittab*

1.6 CONTRÔLE DU PROCESSUS INIT

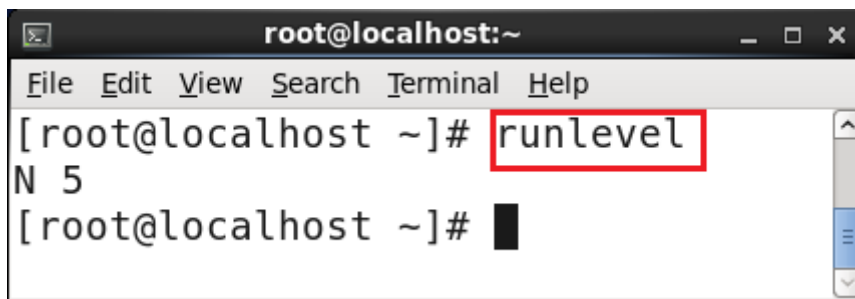
Connaître le niveau d'exécution courant

La plupart des systèmes Linux utilisent une version de la commande **who** qui permet d'indiquer le niveau d'exécution courant :



```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]# who -r  
run-level 5 2011-10-22 17:20  
[root@localhost ~]#
```

On peut aussi utiliser une autre commande pour avoir cette information :



```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]# runlevel  
N 5  
[root@localhost ~]#
```

Indique que le système est au niveau 5 et qu'il n'y a pas de niveau précédent.

Changement de niveau d'exécution

Pour contrôler le processus init et changer de runlevel, il suffit d'appeler la commande **init** ou **telinit** avec le niveau d'exécution (0, 1, 2, 3, 4, 5, 6, S, ou s) ou le pseudo-niveau d'exécution (a, b ou c) en argument :

```
telinit 5
```

```
runlevel  
5 3
```

Ici, on est passé au niveau 5, le niveau précédent était le niveau 3.

Le délai par défaut que donne Init aux processus pour s'arrêter lors du changement de runlevel est de cinq secondes ; c'est le temps entre l'envoi du signal **SIGTERM** et **SIGKILL**.

Ce temps peut être modifié en ajoutant l'option **-t** sur la ligne de commande de telinit. Par exemple pour le passer à 30 secondes :

```
telinit -t 30 5
```

Relire /etc/inittab

Lorsqu'il démarre, Init lit le fichier `/etc/inittab` ligne par ligne (si on modifie ce dernier, il n'est pas nécessaire de rebooter la machine. Il suffit d'envoyer un signal HUP à Init pour le forcer à le relire avec la commande :

```
kill -HUP 1
```

On peut aussi forcer Init à effectuer une relecture de `/etc/inittab` en utilisant la commande **telinit** avec l'option **q** ou **Q** :

```
telinit q
```

Démarrage en mode monutilisateur

Il est possible de forcer le niveau d'exécution du système au démarrage de la machine en spécifiant son numéro au noyau à charger (ou le mot **single** pour le runlevel S).

Sous GRUB, il faudra éditer les arguments de l'une des entrées en appuyant sur la touche **[e]**, puis valider après l'ajout de **single** en fin de ligne.

Le mode mono-utilisateur permet de se connecter avec les droits **root** sur la console de la machine sans entrer le mot de passe de **root**. Ceci est intéressant pour les administrateurs ayant oublié leur mot de passe.

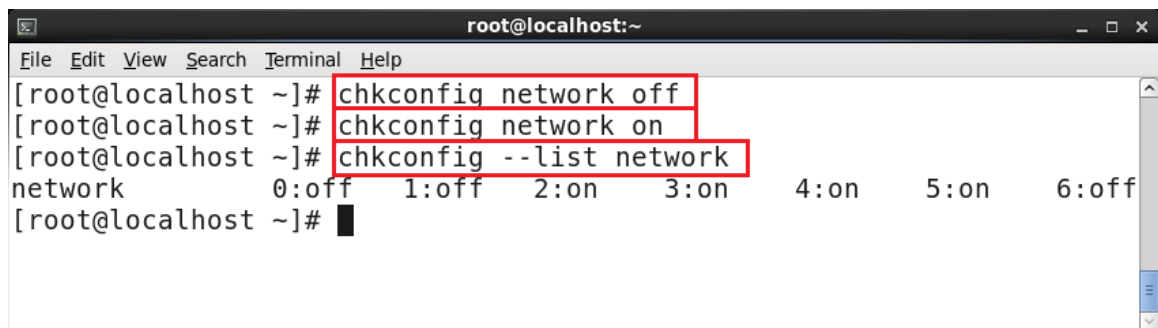
1.7 AJOUT ET SUPPRESSION DE SERVICE AU DÉMARRAGE

Le programme `/etc/rc.d/rc` exécutant tous les scripts présents dans le répertoire du runlevel, il suffira d'ajouter un lien dans le répertoire vers le script adéquat pour lancer ou arrêter un service lorsque le système entre dans ce niveau d'exécution.

Il existe des outils en ligne de commandes ou avec une interface graphique qui permettent l'ajout et la suppression de services au démarrage :

chkconfig

La commande **chkconfig** permet de lister, d'ajouter et de supprimer les différents services des runlevels.

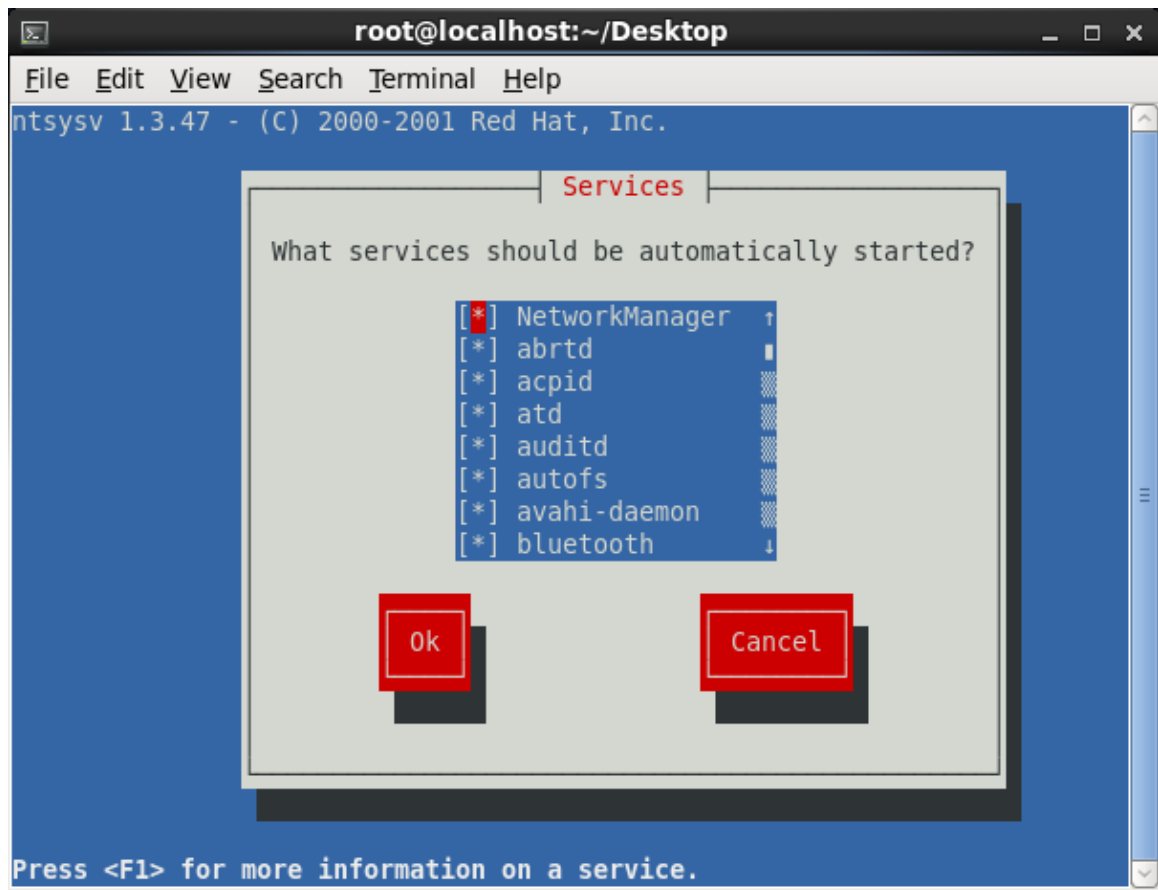


```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]# chkconfig network off  
[root@localhost ~]# chkconfig network on  
[root@localhost ~]# chkconfig --list network  
network          0:off  1:off  2:on   3:on   4:on   5:on   6:off  
[root@localhost ~]#
```

ntsysv

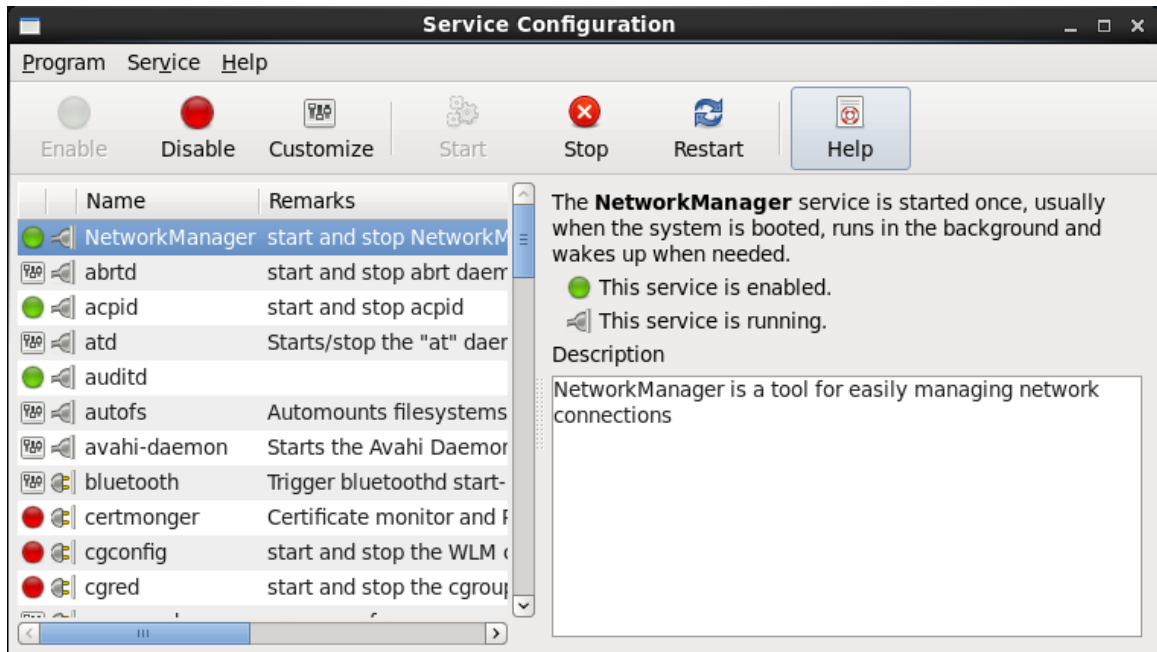
La commande **ntsysv** affiche dans un menu tous les scripts présents dans */etc/rc.d/init.d* et propose de les ajouter ou supprimer du niveau d'exécution courant ou de ceux spécifiés sur la ligne de commandes avec l'option `--level`.

```
ntsysv --level 345
```



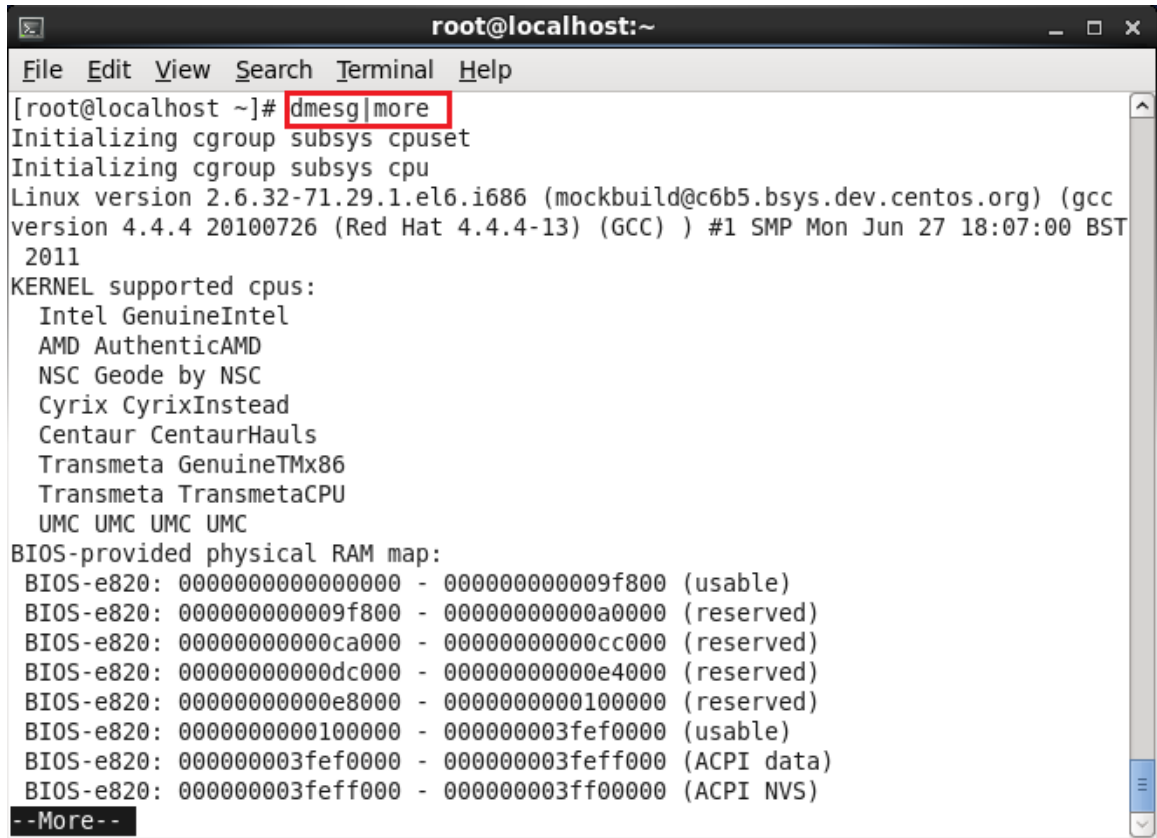
system-config-services

Cet outil est accessible via le menu GNOME.



1.8 MESSAGES

La commande `dmesg` permet d'examiner ou de contrôler le tampon des messages du noyau (kernel). Cette commande permet d'afficher les messages du démarrage de la machine.



```
root@localhost:~
File Edit View Search Terminal Help
[root@localhost ~]# dmesg|more
Initializing cgroup subsys cpuset
Initializing cgroup subsys cpu
Linux version 2.6.32-71.29.1.el6.i686 (mockbuild@c6b5.bsys.dev.centos.org) (gcc
version 4.4.4 20100726 (Red Hat 4.4.4-13) (GCC) ) #1 SMP Mon Jun 27 18:07:00 BST
2011
KERNEL supported cpus:
  Intel GenuineIntel
  AMD AuthenticAMD
  NSC Geode by NSC
  Cyrix CyrixInstead
  Centaur CentaurHauls
  Transmeta GenuineTMx86
  Transmeta TransmetaCPU
  UMC UMC UMC UMC
BIOS-provided physical RAM map:
BIOS-e820: 0000000000000000 - 0000000000009f800 (usable)
BIOS-e820: 0000000000009f800 - 000000000000a0000 (reserved)
BIOS-e820: 000000000000ca000 - 000000000000cc000 (reserved)
BIOS-e820: 000000000000dc000 - 000000000000e4000 (reserved)
BIOS-e820: 000000000000e8000 - 00000000000100000 (reserved)
BIOS-e820: 00000000000100000 - 000000000003fef0000 (usable)
BIOS-e820: 000000000003fef0000 - 000000000003feff000 (ACPI data)
BIOS-e820: 000000000003feff000 - 000000000003ff00000 (ACPI NVS)
--More--
```