Les procédures de démarrage d'un système Unix

A Démarrage d'un système UNIX : /etc/inittab....2
Les étapes du démarrage d'un système Unix 2
Lancements réalisés au niveau de chaque session utilisateur 2
Le fichier /etc/inittab....3
man inittab....3
Exemple de fichiers inittab 4
Run level ou niveau d'exécution....6
Conditions d'activation des processus d'inittab....7
B Démarrage des services sous Linux 8
Mécanisme associé au démarrage automatique des services 9
C Démarrage des services réseau sous Linux....10
Le "super serveur" xinetd....10
D Démarrage d'une session utilisateur 11
Fichier "profile" selon "man bash"....12

A Démarrage d'un système UNIX : /etc/inittab

Les étapes du démarrage d'un système Unix

- Boot BIOS en ROM
- Chargeur du système d'exploitation MBR du disque d'amorçage (ou ROM de la carte réseau)
- Exécution du noyau Unix
- Lancement de l'ordonnanceur shed
- Lancement du processus **init** pour un niveau d"exécution (**run level**) donné et exécution des commandes de **/etc/inittab**

Lancements réalisés au niveau de chaque session utilisateur

Surveillance des terminaux:	getty	/etc/gettydefs /etc/issue (message)
Mot de passe .... :	login	/etc/passwd /etc/motd (message)
Lancement du Shell :	bash	/etc/profile $HOME/.profile (ou $HOME/.bash_profile)

Les premiers processus lancés sont donc l'ordonnanceur **shed**, puis le processus **init**.
init est l'ancêtre commun de tous les processus qui seront ultérieurement créés sur le système (voir par exemple le résultat de la commande **pstree –p**)

Le fichier /etc/inittab

Les actions du processus **init** sont déterminées par le contenu du fichier **/etc/inittab**, dont un exemple est donné ci-après :

man inittab

The inittab file describes which processes are started at bootup and during normal operation (e.g. /etc/init.d/boot, /etc/init.d/rc, gettys...). Init(8) distinguishes multi ple runlevels, each of which can have its own set of pro cesses that are started. Valid runlevels are 0-6 plus A, B, and C for ondemand entries. An entry in the inittab file has the following format:

id:runlevels:action:process

Lines beginning with `#' are ignored.

id is a unique sequence of 1-4 characters which iden tifies an entry in inittab (for versions of sysvinit compiled with libraries < 5.2.18 or a.out libraries the limit is 2 characters).

Note: For gettys or other login processes, the id field should be the tty suffix of the corresponding tty, e.g. 1 for tty1. Otherwise, the login accounting might not work correctly.

runlevels
lists the runlevels for which the specified action should be taken.

action describes which action should be taken.

process
specifies the process to be executed. If the pro cess field starts with a `+' character, init will not do utmp and wtmp accounting for that process. This is needed for gettys that insist on doing their own utmp/wtmp housekeeping. This is also a historic bug.

The runlevels field may contain multiple characters for different runlevels. For example, 123 specifies that the process should be started in runlevels 1, 2, and 3. The runlevels for ondemand entries may contain an A, B, or C. The runlevels field of sysinit, boot, and bootwait entries are ignored.

Exemple de fichiers inittab

```
# The default runlevel is defined here
id:5:initdefault:

# First script to be executed, if not booting in emergency (-b) mode
si::bootwait:/etc/init.d/boot

# /etc/init.d/rc takes care of runlevel handling
#
# runlevel 0  is  System halt   (Do not use this for initdefault!)
# runlevel 1  is  Single user mode
# runlevel 2  is  Local multiuser without remote network (e.g. NFS)
# runlevel 3  is  Full multiuser with network
# runlevel 4  is  Not used
# runlevel 5  is  Full multiuser with network and xdm
# runlevel 6  is  System reboot (Do not use this for initdefault!)
#
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
#l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6

# what to do in single-user mode
ls:S:wait:/etc/init.d/rc S
~~:S:respawn:/sbin/sulogin

# what to do when CTRL-ALT-DEL is pressed
ca::ctrlaltdel:/sbin/shutdown -r -t 4 now

# special keyboard request (Alt-UpArrow)
# look into the kbd-0.90 docs for this
kb::kbrequest:/bin/echo "Keyboard Request -- edit /etc/inittab to let this work."

# what to do when power fails/returns
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
#pn::powerfail:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop

# for ARGO UPS
sh:12345:powerfail:/sbin/shutdown -h now THE POWER IS FAILING
```

```
# getty-programs for the normal runlevels
# <id>:<runlevels>:<action>:<process>
# The "id" field  MUST be the same as the last
# characters of the device (after "tty").
1:2345:respawn:/sbin/mingetty --noclear 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
#
#S0:12345:respawn:/sbin/agetty -L 9600 ttyS0 vt102

#
#  Note: Do not use tty7 in runlevel 3, this virtual line
#  is occupied by the programm xdm.
#

#  This is for the package xdmsc, after installing and
#  and configuration you should remove the comment character
#  from the following line:
#7:3:respawn:+/etc/init.d/rx tty7


# modem getty.
# mo:235:respawn:/usr/sbin/mgetty -s 38400 modem

# fax getty (hylafax)
# mo:35:respawn:/usr/lib/fax/faxgetty /dev/modem

# vbox (voice box) getty
# I6:35:respawn:/usr/sbin/vboxgetty -d /dev/ttyI6
# I7:35:respawn:/usr/sbin/vboxgetty -d /dev/ttyI7

# end of /etc/inittab
```

- L'identificateur de la ligne inittab composé de 1 ou 2 caractères doit être unique dans toute la table.
- La commande associée à l'entrée d'inittab peut être un programme exécutable ou un script shell; on pourra lui associer des paramètres.
- Des commentaires peuvent être insérés après le caractère #

Run level ou niveau d'exécution

Le numéro de **run level** , ou niveau d'exécution du système va permettre de **sélectionner** les processus activés par **init** .

Ce numéro peut prendre les valeurs suivantes :

0..6	niveau de configuration multiutilisateur
a,b,c	voir REMARQUES ci-après

- Au lancement du système, init demande à l'opérateur, sur la console système, quel est le numéro de runlevel choisi: seules les lignes d'inittab présentant le numéro choisi (ou ne comportant aucun numéro) donneront naissance à un processus.

- Si l'opérateur spécifie le niveau s, le système démarrera en monoutilisateur (single user)

- Le processus init ne demandera pas le niveau de configuration si le fichier inittab contient une ligne dont la condition d'activation est initdefault: c'est le niveau de configuration associé à cette ligne qui est dans ce cas automatiquement choisi.

- En mode superutilisateur, il sera ensuite possible de changer le niveau de configuration actif par la commande

  init <nouveau-numéro>

  Tous les processus qui se trouvent alors exclus seront tués.

REMARQUES:

1. Il est possible d'associer plusieurs numéros de **run level** à chaque ligne **d'inittab**.

2. Les niveaux a,b et c ne sont pas de vrai niveaux de configuration, mais ils permettent, par la commande **init** de lancer facilement les processus qui leur sont associés (un changement de niveau ne les détruira pas non plus)

3. La commande **init q** demande au processus **init** de relire **inittab** : elle sera utilisée à chaque fois que l'on aura modifié ce fichier, pour qu'**init** prenne immédiatement en compte les modifications.

init examine automatiquement la table dans les cas suivants :

⇨au chargement du système,
⇨à l'apparition d'un défaut secteur lorsque meurt l'un des processus lancés par init
⇨lorsque l'opérateur le demande explicitement par la commande init q

Conditions d'activation des processus **d'inittab**

Un processus qui correspond au niveau de configuration choisi ne sera effectivement lancé que s'il satisfait aussi la condition spécifiée au 3ème champ **d'inittab**.

Les conditions possibles sont les suivantes :

sysinit : le processus associé est exécuté avant **qu'init** n'essaie d'accéder à la console pour demander le niveau de **run**: normalement, l'action associée à cette entrée doit permettre d'initialiser correctement la console système.

initdefault: spécifie le niveau de **run** qui doit être choisi automatiquement par **inittab** (si cette entrée n'existe pas, **init** demandera à l'opérateur de spécifier un numéro)

boot: le processus est exécuté une seule fois au démarrage du système

bootwait : même cas que ci dessus, mais en plus **init** attendra la mort de ce processus avant de continuer l'examen **d'inittab**.

once: ce processus sera lancé chaque fois qu'il y aura un changement de niveau de configuration

wait : même cas que ci-dessus, mais **init** attendra la mort de ce processus avant de poursuivre la scrutation **d'inittab**.

respawn : (engendrer, générer) le processus est créé sans attente de la part **d'init**, à l'initialisation. lorsque le processus ainsi lancé meurt, il est automatiquement relancé, pourvu que le niveau de configuration le permette encore.

ondemand: est équivalent à **respawn** pour les pseudo-niveaux **a,b ou c**; les processus associés seront automatiquement relancés lorsqu'ils se termineront.

off : si le processus associé n'existe pas, il n'est pas créé; s'il existe (cas d'une modification **d'inittab**), il est tué dans un délai de 20 secondes (ce délai lui permettant de s'autodétruire en sauvegardant ses données)

powerfail : **init** lance ce processus lorsqu'il reçoit le signal **SIGPWR**, et poursuit sans attendre la scrutation **d'inittab** (cette possibilité peut demander des options matérielles particulières, qui permettront par exemple de sauvegarder les buffers par **sync** ou de recopier le contenu de la mémoire sur disque)

powerwait: même chose que ci-dessus, mais **init** attend la fin du processus avant de continuer l'examen **d'inittab**.

B Démarrage des services sous Linux

Les services sous Linux sont démarrés par l'intermédiaire de procédures situés dans le répertoire **/etc/rc.d/init.d** (ou **/etc/init.d** selon la distributions Linux)

Exemple de liste de procédures de démarrage et d'arrêt des services :

```
demo@mail ~ > ls /etc/rc.d/init.d

anacron  functions  irda      lpd          nfs       rusersd  snmpd
apmd     gpm        kdcrotate named        nfslock   rwalld   syslog
arpwatch halt       keytable  named.rpmsave pcmcia   rwhod    xfs
atd      identd     killall   named.run     portmap   sendmail ypbind
avgate   inet       kudzu     netfs         random    single   yppasswdd
crond    ipchains   linuxconf network       rstatd    smb      ypserv
```

Chacune de ces procédures peut être invoquée avec le paramètre **start** ou **stop** pour démarrer ou arrêter le service. D'autres paramètres sont éventuellement disponibles selon les procédures, et en général affichés lorsque l'on utilise la procédure sans paramètres.

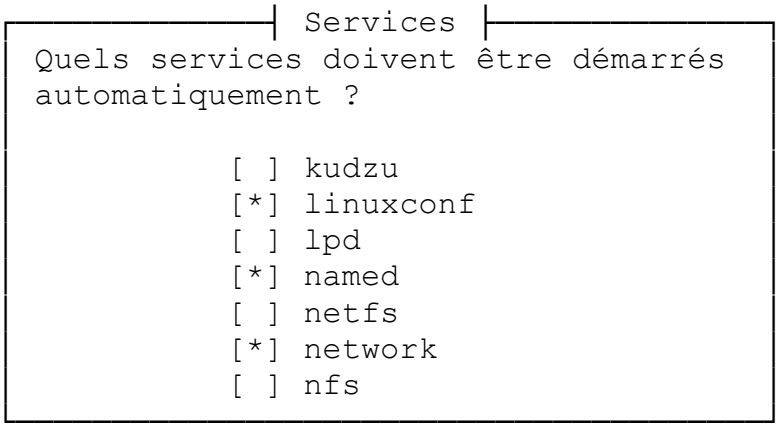
Sur certaines distributions, il est aussi possible d'utiliser la commande **service <nom du service>**

Exemples :

```
demo@mail ~ > /etc/rc.d/init.d/network
Usage: network {start|stop|restart|reload|status|probe}

demo@mail ~ > /sbin/service network
Usage: network {start|stop|restart|reload|status|probe}
```

Il est possible de prévoir un **démarrage automatique des services** grâce à la commande **ntsysv**[1]



[1] **ntsystv** considère automatiquement les services du niveau de configuration courant (**runlevel** courant) pour travailler sur un autre niveau il faut le préciser au lancement de la commande, par exemple **ntsysv --level 5 .** D'autres outils d'administration comme **linuxconf** ou **webmin** (**redhat-config**, **drakconf** ou **yast**…) permettent de paramétrer ainsi le démarrage automatique des services.

Mécanisme associé au démarrage automatique des services

L'activation ou la désactivation du démarrage automatique d'un service, pour le niveau de configuration **3**, va créer dans **/etc/rc.d/rc3.d** un **lien** (ou raccourci) vers la procédure correspondante de **/etc/rc.d/init.d/** (respectivement pour les niveaux de configuration 2, 4 ou 5)…

- le nom du lien commencera par K si le service doit être arrêté (Kill)
- le nom du lien commencera par S si le service doit être démarré (Start).

Un **numéro d'ordre** permet en plus de déterminer dans quel ordre les service doivent être arrêtés ou démarrés.

Exemple :

```
root@mail > ls  /etc/rc.d/rc3.d

lrwxrwxrwx   1 root   root      K34yppasswdd -> ../init.d/yppasswdd
lrwxrwxrwx   1 root   root      K35smb -> ../init.d/smb
lrwxrwxrwx   1 root   root      K45arpwatch -> ../init.d/arpwatch
lrwxrwxrwx   1 root   root      K50snmpd -> ../init.d/snmpd
lrwxrwxrwx   1 root   root      K60lpd -> ../init.d/lpd
lrwxrwxrwx   1 root   root      K65identd -> ../init.d/identd
lrwxrwxrwx   1 root   root      K70nfslock -> ../init.d/nfslock
lrwxrwxrwx   1 root   root      K75netfs -> ../init.d/netfs
lrwxrwxrwx   1 root   root      S50inet -> ../init.d/inet
lrwxrwxrwx   1 root   root      S55named -> ../init.d/named
lrwxrwxrwx   1 root   root      S75keytable -> ../init.d/keytable
lrwxrwxrwx   1 root   root      S80avgate -> ../init.d/avgate
lrwxrwxrwx   1 root   root      S80sendmail -> ../init.d/sendmail
lrwxrwxrwx   1 root   root      S85gpm -> ../init.d/gpm
lrwxrwxrwx   1 root   root      S90xfs -> ../init.d/xfs
```

C Démarrage des services réseau sous Linux

Le "super serveur" xinetd

xinetd active les processus répondant aux demandes des clients en exploitant les directives de **/etc/xinetd.conf** et des différents fichiers de **/etc/xinetd.d**

La signification des différents champs d'un paragraphe de **/etc/xinetd.conf** est la suivante:

1. nom du service tel qu'il est défini dans **/etc/services**
2. type de **socket** utilisé (stream ou datagram)
3. nom du **protocole** utilisé et défini dans /etc/protocols
4. délai **wait** ou **nowait** : un serveur déclaré wait doit être capable de satisfaire les requêtes qui arrivent pour le même service pendant sa durée de vie, inetd ne doit pas dans ce cas lancer plus d'un serveur de ce type (c'est le cas de talk). Pour le cas nowait, inetd lancera autant de serveurs qu'il y a de requêtes (c'est le cas de telnet).
5. le nom de **login** donnant l'UID et les droits avec lesquels s'exécute le démon (souvent root)
6. le nom du fichier **exécutable** pour le démon (**internal** si **xinetd** assure lui-même ce service)
7. les **paramètres** (5 maximum) transmis au lancement du processus serveur (démon)

Exemple de configuration pour xinetd et telnetd

```
demo@Forum:/etc> ls -ld xinet*
-rw-r--r--    1 root     root          623 2003-10-02 22:17 xinetd.conf
drwxr-xr-x    2 root     root          496 2004-04-28 15:22 xinetd.d

demo@Forum:/etc> ls xinetd.d
     chargen      cvs          echo      netstat   systat  time-udp
     chargen-udp  daytime      echo-udp  servers   telnet  vnc
     cups-lpd     daytime-udp  ftpd      services  time    vsftpd
```

demo@Forum:/**etc**> c**at xinetd.d/telnet**

```
# default: off
# description: Telnet is the old login server which is INSECURE and should
#       therefore not be used. Use secure shell (openssh).
#       If you need telnetd not to "keep-alives" (e.g. if it runs over a ISDN \
#       uplink), add "-n".  See 'man telnetd' for more details.

service telnet
{
      socket_type           = stream
      protocol        = tcp
      wait            = no
      user            = root
      server          = /usr/sbin/in.telnetd
      disable         = no
}
```

D Démarrage d'une session utilisateur

L'examen de la table **/etc/inittab** permet de se rendre compte que chaque ligne de liaison série, est surveillée par un processus **getty** (ou mingetty) créé par **inittab** .

```
# getty-programs for the normal runlevels
# <id>:<runlevels>:<action>:<process>
# The "id" field  MUST be the same as the last
# characters of the device (after "tty").

1:2345:respawn:/sbin/mingetty --noclear 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
```

getty **getty** utilise un fichier de description des caractéristiques des terninaux : **/etc/gettydefs;** on trouvera en particulier dans ce fichier des indications sur la vitesse de transmission (en bauds) à utiliser, ainsi que le texte du message de login.

Enfin le message d'accueil envoyé sur les terminaux est le contenu du fichier **/etc/issue.**

getty, par une fonction **exec** (2) , se trouve ensuite remplacé par le programme **login** .

login Ce programme va interpréter le contenu de /etc/passwd et autoriser ou non la connexion, en fonction du mot de passe.

Si la connexion est réalisée le contenu du fichier **/etc/motd** (message of the day) est affiché sur l'écran.

Puis une nouvelle fonction **exec**(2) est exécutée, et la commande (**shell**) associée à **l'utilisateur** dans le fichier **/etc/passwd**, est lancée; il s'agit en général de l'interpréteur de commandes /bin/sh ou /bin/bash

bash Ce shell va exécuter les procédures /etc/profile et $HOME/.profile, puis attendre les commandes de l'opérateur.

Remarquons que le **shell**, grâce aux différentes fonctions **exec()** réalisées correspond au même processus que le processus **getty** qui fut créé par **init**. La mort de ce **shell**, sera donc bien la mort d'un des fils **d'init**, ce qui va réactiver **init**, relancer l'examen **d'inittab**, et recréer un processus **getty** car la condition d'activation était **respawn**.

Fichier "profile" selon "man bash"

Lorsque **bash** est lancé comme shell de login interactif, ou comme shell non-interactif avec l'option --login, il lit et exécute tout d'abord les commandes se trouvant dans le fichier **/etc/profile** s'il existe. Après lecture de ce fichier, il recherche **~/.bash_profile, ~/.bash_login, et ~/.profile, dans cet ordre,** et exécute les com mandes se trouvant dans le premier fichier existant et accessible en lecture. L'option --noprofile peut être utilisée au démarrage du shell pour empêcher ce comportement.

Lorsqu'un shell de login se termine, bash lit et exécute les comman des du fichier **~/.bash_logout,** s'il existe.

Quand un shell interactif démarre sans être un shell de login, bash lit et exécute les commandes se trouvant dans **~/.bashrc** s'il existe. Ce comportement peut être inhibé à l'aide de l'option --norc. L'option --rcfile fichier forcera bash à exécuter les commandes dans fichier plutôt que dans ~/.bashrc.

Quand bash est démarré de manière non-interactive, pour lancer un script shell par exemple, il recherche la variable BASH_ENV dans l'environnement, développe son contenu si elle existe, et considère cette valeur comme le nom d'un fichier à lire et exécuter. Bash se comporte comme si la commande suivante se trouvait en début de script :

```
if [ -n "$BASH_ENV" ]; then . "$BASH_ENV"; fi
```

mais la valeur de la variable PATH n'est pas utilisée pour rechercher le fichier.