Module n°3

ADMINISTRATION SYSTEME

Auteur : Labo-Linux
Version 0.1 – 17 novembre 2003
Nombre de pages : 126

Paris Academy of Computer Science
SUPINFO Paris
Ecole Supérieure d'Informatique

Ecole Supérieure d'Informatique de Paris
23. rue Château Landon 75010 – PARIS
www.supinfo.com

Table des matières

1. Configuration, Test et Utilisation du Réseau

1.1. Introduction

Rares sont, à l'heure actuelle, les systèmes isolés. Les besoins croissants de communication, tant entre utilisateurs qu'entre systèmes, expliquent le foisonnement des réseaux.

Dans ce domaine, on distingue deux types d'outils : ceux pour configurer et tester la connexion au réseau, utilisés principalement par les administrateurs, et ceux faisant usage du réseau dans diverses applications destinées à l'utilisateur.

Nous considèrerons, par la suite, que le réseau repose sur l'ensemble de protocoles TCP/IP (solution largement majoritaire, quel que soit le type de réseau) et que le lecteur maîtrise les notions de bases inhérentes à ce protocole.

1.2. Préliminaires

Avant toute configuration, il convient de vérifier :

- que l'ordinateur possède bien un adaptateur correspondant au réseau visé (carte token-ring, ethernet, wireless, ...)
- que le noyau supporte cet adaptateur, soit directement, soit à l'aide d'un module propre ou tiers
- que le noyau prend également en charge la couche TCP/IP.

Nous ne nous attarderons pas sur ces aspects, les distributions courantes étant déjà prêtes à l'emploi.

Pour communiquer via un réseau, il est nécessaire de configurer une interface en lui attribuant au moins une adresse IP et un masque de sous réseau. Eventuellement, on peut y ajouter les adresses de serveurs DNS, d'une passerelle ou des routes supplémentaires.

Tout cela peut bien sûr être automatisé l'aide d'un serveur DHCP. Pour plus de détails concernant ces notions, reportez-vous à vos cours d'architecture réseau.

1.3. Configuration automatique

S'il existe un serveur DHCP, la simple commande *dhcpcd* (ou *dhclient*) suffira pour configurer la première interface ethernet. Celle-ci *broadcast* sur le réseau à la recherche d'un vaillant serveur DHCP prêt à lui donner une identité.

Il n'y a en principe rien d'autre à ajouter. Pour configurer de cette manière une autre interface, il suffit de l'indiquer en argument de la commande.

En revanche, s'il n'existe pas de serveur DHCP (cas des petits réseaux) ou si la réponse de celui-ci n'est pas correcte, il faut effectuer la configuration à la main. Pas de panique, c'est à la portée de n'importe qui !

1.4. Configuration manuelle

1.4.1. Interfaces

La commande *ifconfig* affiche la configuration des interfaces réseau. Elle permet également de les configurer en leur attribuant une adresse IP et un masque de sous réseau. On peut éventuellement spécifier une adresse de broadcast.

```
$ /sbin/ifconfig -a
eth0  Link encap:Ethernet HWaddr 00:04:AC:45:63:7E
      inet addr:192.168.1.1 Bcast:192.168.1.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:164442 errors:0 dropped:0 overruns:0 frame:1
      TX packets:133671 errors:0 dropped:0 overruns:72 carrier:0
      collisions:0 txqueuelen:100
      RX bytes:91115012 (86.8 Mb) TX bytes:33953990 (32.3 Mb)
      Interrupt:10 Base address:0xc000

lo    Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      UP LOOPBACK RUNNING MTU:16436 Metric:1
      RX packets:3431 errors:0 dropped:0 overruns:0 frame:0
      TX packets:3431 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:197669 (193.0 Kb) TX bytes:197669 (193.0 Kb)

ppp0  Link encap:Point-to-Point Protocol
      inet addr:217.128.163.84 P-t-P:193.253.160.3 Mask:255.255.255.255
      UP POINTOPOINT RUNNING NOARP MULTICAST MTU:9178 Metric:1
      RX packets:11296 errors:0 dropped:0 overruns:0 frame:0
      TX packets:11283 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:3
      RX bytes:5061982 (4.8 Mb) TX bytes:1272910 (1.2 Mb)
```

Ici, on va configurer l'interface eth0 de telle sorte qu'elle ait comme ip 172.16.1.10, que son netmask soit 255.255.255.128 et que son adresse de broadcast soit 172.16.1.127.

```
$ /sbin/ifconfig eth0 172.16.1.10 netmask 255.255.255.128 broadcast
172.16.1.127
```

1.4.2. Routes

Une fois les interfaces réseaux configurées, il possible d'atteindre n'importe quelle machine située dans les réseaux locaux auxquels nous sommes attachés.

En revanche, pour atteindre des machines situées au delà de ces réseaux, comme un serveur web sur l'Internet, il est nécessaire d'utiliser une passerelle.

Les différents moyens d'accès, nommés routes, se situent dans la table de routage, que la commande route permet justement de consulter et de modifier :

```
$ /sbin/route
Kernel IP routing table
Destination   Gateway       Genmask         Flags Metric Ref Use Iface
193.253.160.3 *             255.255.255.255 UH    0      0   0   ppp0
192.168.1.0   *             255.255.255.0   U     0      0   0   eth0
default       193.253.160.3 0.0.0.0         UG    0      0   0   ppp0
```

Pour attacher un sous réseau local à une interface :

```
$ /sbin/route add -net 172.16.1.0/25 dev eth0
```

Pour atteindre un sous réseau distant, il est nécessaire de spécifier une passerelle (gateway, en anglais) :

```
$ /sbin/route add -net 172.16.1.128/25 gw 172.16.1.125
```

Les communications ne répondant à aucune des routes présentes utilisent la route par défaut (pour l'Internet notamment) :

```
$ /sbin/route add default gw 172.16.1.1
```

1.4.3. Résolution de nom

Enfin, pour profiter de la résolution de nom, il faut indiquer les adresses IP de serveurs DNS. Il faut garder en mémoire que, selon la nature du serveur, celui-ci pourra résoudre soit des noms internes à un réseau, soit des noms externes (l'Internet), soit les deux.

Cette liste d'adresses IP se situe dans le fichier */etc/resolv.conf* :

```
$ cat /etc/resolv.conf
domain popo.net
nameserver 216.223.224.7
nameserver 216.223.224.6
```

1.5.Outils du paquet IProute

Iproute2 regroupe une série d'outils qui permettent d'afficher ou de modifier un grand nombre d'informations sur la manière dont sont gérées les connexions réseau par le noyau linux. Les commandes route et ifconfig font des appels système à iproute2 avec les options par defaut.

1.5.1. Utilisation de l'outil ip

Affichage des liens :

Orion:/# ip link list

```
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:80:ad:73:09:b6 brd ff:ff:ff:ff:ff:ff

3: ppp0: <POINTOPOINT,MULTICAST,NOARP,UP> mtu 1432 qdisc pfifo_fast qlen 3
    link/ppp
```

Le premier lien représente le loopback, le second la connexion au reseau local et le dernier représente la connexion Internet. Il est à noter que cette commande ne nous montre pas les adresses IP mais uniquement les adresses physiques MAC.

Affichage des adresses IP :

Orion:/# ip address show

```
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo

2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:80:ad:73:09:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.1/24 brd 192.168.0.255 scope global eth0

3: ppp0: <POINTOPOINT,MULTICAST,NOARP,UP> mtu 1432 qdisc pfifo_fast qlen 3
    link/ppp
    inet 62.62.189.245 peer 62.62.189.1/32 scope global ppp0
```

Nous avons ici un affichage plus complet: nous voyons quelle adresse IP est liée à quelle carte.

mtu signifie Maximum Transfert Unit soit l'Unité Maximale de Transfert. Elle est exprimée en octets. qdisc designe Queueing Discipline (gestion de la mise en file d'attente).

Affichage des routes :

```
Orion:/# ip route show

62.62.189.1 dev ppp0  proto kernel  scope link  src 62.62.189.245

192.168.0.0/24 dev eth0  proto kernel  scope link  src 192.168.0.1

default via 62.62.189.1 dev ppp0
```

On voit ici que le reste du monde (Internet) est disponnible par la connexion ppp0.

1.6. Tests

Tester la connexion d'un réseau, signifie, d'une part, vérifier que la configuration est bonne et, d'autre part, que les éléments constitutifs du réseau fonctionnent correctement (passerelle, dns, etc).

Pour vérifier que la configuration du système est bonne, il suffit d'utiliser les mêmes outils et fichiers qui ont permis cette configuration, à savoir ifconfig, route, /etc/resolv.conf.

D'autres outils permettent d'effectuer des tests à l'extérieur du système.

1.6.1. Ping

La commande ping est certainement la plus utilisée pour tester une connexion au réseau. D'une manière générale, elle vérifie qu'une machine distante est atteignable. Plus spécifiquement, elle permet de vérifier chacune des étapes de la configuration.

```
$ ping 127.0.0.1  # vérifie que TCP/IP est supporté par le noyau
$ ping <mon_ip>   # vérifie que l'interface est correctement configurée
$ ping <une_ip>   # vérifie que la table de routage est bonne
$ ping <un_nom>   # vérifie que la résolution de nom fonctionne
```

1.6.2. Traceroute

Si une communication échoue bien que le système soit correctement configuré, le problème vient certainement d'un intermédiaire.

La commande *traceroute* est ici très utile car son rôle est justement d'afficher les intermédiaires intervenant dans une communication.

On peut alors détecter les congestions. Attention cependant, cette commande fonctionne au niveau *réseau* uniquement (cf. modèle OSI). Cela signifie qu'elle ne détecte pas le matériel fonctionnant aux couches inférieures comme les switchs ou ponts filtrants :

```
$ /usr/bin/traceroute
traceroute to mail.supinfo.com (212.80.91.71), 30 hops max, 40 byte packets
1 gougniafier (192.168.1.1) 0.282 ms 0.147 ms 0.105 ms
2 193.253.160.3 (193.253.160.3) 55.493 ms 55.736 ms 58.933 ms
3 80.10.160.162 (80.10.160.162) 52.871 ms 50.823 ms 54.901 ms
4 P0-0.nrsta204.Paris.francetelecom.net (193.252.159.82) 54.891 ms 78.775
5 193.252.161.2 (193.252.161.2) 55.979 ms 52.761 ms 53.961 ms
6 193.252.161.57 (193.252.161.57) 58.317 ms 53.773 ms 52.900 ms
7 193.252.161.30 (193.252.161.30) 57.924 ms 54.768 ms 51.907 ms
8 P5-0.ntaub101.Aubervilliers.francetelecom.net (193.251.126.181) 55.824 ms
9 P7-0.noprx101.Paris.francetelecom.net (193.251.126.18) 56.951 ms 56.765
10 193.252.161.252 (193.252.161.252) 54.907 ms 52.759 ms 54.909 ms
```

```
11 ge0-0-0-0.br2.bbpar.fr.easynet.net (212.180.2.98) 51.945 ms 57.798 ms
12 easy-interlan.router.easynet.fr (212.180.0.135) 55.892 ms 53.760 ms
13 easy-interlan-10.easynet.fr (212.180.93.42) 59.920 ms 56.677 ms 59.887
14 supinfo-2.supinfo.com (212.180.91.71) 57.182 ms 51.742 ms 53.876 ms
```

1.6.3. netstat

La commande netstat, quant à elle, offre diverses informations et statistiques concernant les principales couches du modèle OSI.

Les plus intéressantes sont la liste des ports ouverts et celle des connexions actuelles.

```
$ netstat -tlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State    PID/Program name
tcp   0      0      *:36167       *:*             LISTEN   20091/artsd
tcp   0      0      *:6000        *:*             LISTEN   20032/X
tcp   0      0      *:www         *:*             LISTEN   17170/apache
tcp   0      0      *:ssh         *:*             LISTEN   8283/sshd
tcp   0      0      *:https       *:*             LISTEN   17170/apache
tcp   0      0      *:35515       *:*             LISTEN   7981/centericq
```

```
$ netstat -top
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address       State       PID/Program name
tcp   0      0      192.168.1.10:35514      205.188.9.63:5190     ESTABLISHED 7981/centericq
tcp   0      0      192.168.1.10:36240      nephtys.lip6.fr:45196 ESTABLISHED 28124/ncftp
tcp   0      0      192.168.1.10:33605      207.46.106.47:1863    ESTABLISHED 7981/centericq
tcp   0      0      192.168.1.10:35936      gougniafier:ssh       ESTABLISHED 18543/ssh
tcp   0      0      192.168.1.10:32771      gougniafier:ssh       ESTABLISHED 5947/ssh
tcp   0      0      192.168.1.10:36223      nephtys.lip6.fr:ftp   ESTABLISHED 28124/ncftp
```

1.6.4. Host

La commande *host* est un outil d'interrogation de serveur DNS.
Elle permet de résoudre un nom en adresse IP et inversement. Elle peut aussi consulter n'importe quel autre champ d'un serveur DNS (champ MX, par exemple) :

```
$ /usr/bin/host yaubi.dyndns.org
Received 148 bytes from 216.223.224.7#53 in 912 ms
yaubi.dyndns.org has address 80.13.155.128
```

```
$ /usr/bin/host 212.180.91.64
64.91.180.212.in-addr.arpa domain name pointer supinfo-net.supinfo.com.
```

```
$ /usr/bin/host -a www.redhat.fr
Trying "www.redhat.fr"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61734
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2
;; QUESTION SECTION:
;www.redhat.fr. IN ANY
;; ANSWER SECTION:
www.redhat.fr. 86368 IN CNAME www.fr.redhat.com.
;; AUTHORITY SECTION:
redhat.fr. 156228 IN NS auth04.ns.de.uu.net.
```

```
redhat.fr. 156228 IN NS auth54.ns.de.uu.net.
;; ADDITIONAL SECTION:
auth04.ns.de.uu.net. 29284 IN A 192.76.144.17
auth54.ns.de.uu.net. 29284 IN A 194.128.171.100
```

La commande dig est une alternative avancée de host. Ces deux outils sont utilisés pour dépanner un serveur de nom, ou tout au moins pour constater que le mauvais fonctionnement du réseau provient des DNS.

1.7. Utilisations

Les utilisations d'un réseau sont nombreuses et variées. Cependant certaines sont récurrentes : ssh, ftp, mail, web, irc. Voici un petit descriptif des applications les plus utilisées.

On peut les distinguer selon qu'elles possèdent une interface texte ou graphique. La première catégorie à l'avantage d'être légère, rapide, et l'affichage peut être facilement déporté sur une autre machine. La seconde est bien plus conviviale et intuitive.

1.7.1. Ssh

Avant toutes choses, il est un outil que chacun se doit de connaître : OpenSSH.

SSH signifie Secure SHell. C'est un ensemble de programmes qui fournissent une communication sécurisée entre deux ordinateurs. Par ce moyen, il est possible de transmettre des commandes à l'ordinateur distant (commande ssh), d'envoyer ou recevoir des fichiers (commandes scp et sftp), mais aussi et surtout d'ouvrir une session à distance (commande ssh).

Il peut également être utilisé pour mettre en place un tunnel ou déporter des ports TCP comme le port 6000, pour déporter l'affichage d'une application sur un serveur graphique distant) le tout de manière sécurisée évidemment.

La sécurité d'OpenSSH repose sur le système d'authentification d'une part et le chiffrement des communications d'autre part. L'authentification peut fonctionner à l'aide d'un mot de passe ou par un système de clé privée / clé publique (recommandé) comme RSA ou DSA.

Le chiffrement des communications repose sur des protocoles forts comme 3DES, blowfish.

SSH est constitué de plusieurs parties :

sshd : Serveur ssh
scp : Copie distante sécurisée
ssh-keygen : génération de clefs d'authentification, management et conversion
sftp : Transfert sécurisé de fichiers
slogin/ssh : Client ssh
ssh-add : Ajoute les identités DSA ou RSA à l'agent d'authentification
ssh-agent : Agent d'authentification

ssh-keyscan : Recueille les clefs publiques ssh

Voici quelques exemples d'utilisation de ces programmes :

```
$ ssh user@host                                  # Ouvre une session distante
$ ssh user@host command                          # Envoie une commande
$ scp /path/fichiers user@host:/path/fichiers    # copie un fichier
$ sftp user@host:path/                           # Transfert de fichier
```

1.7.2. Applications mode texte

Lynx, Links et W3m sont les trois navigateurs web en mode texte les plus répandus.

Lynx est le père des deux autres. **Links** supporte les tableaux, les frames et un peu de javascript. Il utilise la couleur et peut déléguer l'affichage des images à une autre application.

W3m est sensiblement identique à Links, à la différence près qu'il support SSL et l'authentification de type "basic" (celle utilisée par mail.supinfo.com, par exemple).

Ncftp est un client ftp en mode texte. Il réagit comme un interpréteur de commande évolué (complément automatique, historique des commandes, etc…) et fournit toutes les fonctionnalités que l'on peut attendre d'un client ftp.

Wget est un petit utilitaire en ligne de commande permettant de télécharger des fichiers par le protocole HTTP. On peut l'utiliser pour réaliser un miroir local d'un site web.

IrcII est un client IRC historique dans le monde Unix, mais toujours utilisé. Irsii est son petit frère. Centericq est un client supportant conjointement les protocoles ICQ, MSN, AIM et IRC, dans une même interface.

Pine et Mutt, enfin, sont deux MUA (Mail User Agent). Pine peut se suffire à lui même pour récupérer, lire, composer et envoyer des mails. Il se veut aussi simple de mise en place et intuitif.

Mutt est d'approche moins évidente mais bien plus puissant. Il s'utilise conjointement à fetchmail et procmail.

1.7.3. Mozilla

Mozilla est la version open-source de Netscape Communicator. C'est évidemment un logiciel fonctionnant avec une interface graphique. Il intègre à son bord un navigateur web, un client FTP graphique, un client IRC, un logiciel complet de messagerie avec carnet d'adresse, un client de newsgroup, ainsi qu'un assistant de création de pages web.

Galeon est un clone de Mozilla dans l'environnement Gnome et Firebird est le petit frère de Mozilla : plus léger et exclusivement navigateur web.

2. L'éditeur Vi

Nous allons voir dans ce cours l'utilisation de l'éditeur de texte Vi.

Il existe sous Unix de nombreux autres éditeurs, partant de ed pour le plus simpliste à jed ou emacs.
Alors pourquoi vi ? La réponse est simple, c'est un éditeur que vous retrouverez sur 99,99% des systèmes Unix.

De nombreuses configurations de programmes Unix passent par l'édition de fichiers. Il est donc essentiel de maîtriser l'utilisation de vi pour être en mesure d'agir sur n'importe quel système Unix (même très ancien).

Voici un aide-mémoire pour son jeu de commandes de base. Des commandes supplémentaires peuvent toutefois exister sur votre propre éditeur vi ou compatible. Reportez-vous à votre documentation système.

Remarque : Vimtutor est un excellent moyen d'apprendre à utiliser vi. Je vous invite donc à passer quelque temps en compagnie de ce logiciel pour vous familiariser avec ce formidable éditeur !

2.1. Principes

2.1.1. Appel de l'éditeur

Editer le fichier s'il existe. Sinon, éditer un nouveau fichier vide :

```
vi fichier
```

Editer la dernière version du fichier en cas d'arrêt inopiné du système ou de l'éditeur :

```
vi -r fichier
```

Editer le fichier avec le curseur à la ligne nn (par défaut : à la dernière ligne) :

```
vi +[nn] fichier
```

Editer chacun des fichiers :

```
vi fichier_1 fichier_2
```

2.1.2. Modes d'édition

Contrairement aux éditeurs *wysiwyg* (what you see is what you get) vi fonctionne par mode : à l'ouverture vous vous trouvez en mode commande. Vous pouvez ainsi ouvrir un nouveau fichier, quitter, effacer une ligne, vous déplacer dans le texte etc...

Cependant vous ne pouvez pas écrire ! Pour ce faire, il faut passer en mode insertion
Une fois que vous avez terminé d'écrire, bien penser à appuyer sur la touche <ESC> pour revenir en mode commande.

Mode insertion

Les caractères tapés s'insèrent dans le texte du fichier édité. Un jeu réduit de commandes est disponible par l'intermédiaire de caractères de contrôle.
Ce mode se termine en tapant <ESC>.

Mode commande

Les caractères tapés sont considérés comme des commandes d'édition de texte.
Attention : Les commandes en majuscules et minuscules diffèrents. Il faut vérifier que les majuscules ne sont pas verrouillées si l'éditeur présente un comportement inhabituel.

2.1.3. Manipulation de fichiers

Commandes	Explications
:q	quitte vi
:q!	quitte vi sans sauver les changements
:ZZ :wq :x	sauve si possible et quitte vi
:[nn,nn] w[!] <Esp> [[>]fich]	sauve le texte dans fich
:w	sauve sous le nom de fichier courant
:w!	sauve sans vérifier les droits d'écriture
:w > fichier	ajoute le texte à la fin de fichier
:n,mw fichier	sauve les lignes de n à m dans fichier
:args	affiche la liste des fichiers à éditer
:n	édite le fichier suivant
:rew	édite le premier fichier dans la liste
:f fichier	change le nom de fichier courant
:e!+3 fichier	édite fichier en ligne 3, ignore changements
:e#	édite le dernier fichier
:[nn]r fichier	insère fichier dans le texte édité
:[nn]r !sh-cmd	exécute une commande shell et insère sa sortie dans le texte (Si nn, après la ligne nn, sinon après le curseur)
:cd [repertoire]	change de répertoire de travail

2.1.4. Déplacer le curseur

Dans les versions récentes de vi (ou vim) vous pouvez vous déplacer grâce aux touches fléchées de votre clavier. Cependant il est bon de connaître les touches qui ont été prévue à l'origine.

Pourquoi ? Deux choses justifient ce choix. Premièrement il existe de vieux terminaux Unix qui ne possèdent pas ces touches. Deuxièmement lorsque vous vous connectez à distance (ssh) sur une machine qui ne supporte pas le keymap approprié, le déplacement 'classique' de fonctionnera pas. En plus de cela, il faut savoir que les commandes vi (dont les touches de déplacement) ont été créées pour être combinées.

Ainsi :12w avancera de 12 mots.
Voici donc les différents types de déplacement :

Commandes	**Explications**
h ou flèche gauche ou <RET>	un caractère à gauche
l ou flèche droite ou <ESP>	un caractère à droite
j ou flèche bas ou Ctrl+n	un caractère vers le bas
k ou flèche haut ou Ctrl+p	un caractère vers le haut
<ENT> ou +, -	début de ligne suivante (pour +), précédente (pour -)
w,b,e	mot suivant, mot précédent, fin du mot
W, B, E	pareil sans tenir compte de la ponctuation
),(	phrase suivante, précédente
},{	paragraphe suivant, précédent
]],[[	section suivante, précédente
_ ou ^	avance jusqu'au prochain non-blanc
fcar	avance jusqu'au prochain caractère "car"
Fcar	revient au dernier caractère "car"
tcar	avance jusqu'au caractère précédent "car"
Tcar	revient au caractère suivant "car"
'car	premier non-blanc de la ligne, marqué "car"
`car	va au caractère marqué car

Les commandes suivantes peuvent aussi être précédées par n, mais le résultat dépend de la commande :

Commandes	**Explications**
G,nG ou :n	fin du texte, ligne n du texte
H, nH	debut de l'écran, ligne n de l'écran
L, nL	fin de l'écran, n lignes avant la fin de l'écran
O,$	début de la ligne, fin de la ligne
n $	fin de la (n-1)eme ligne après la ligne courante

|,n|	première colonne, n-ième colonne
/chn[/]<Ret>	Recherche vers l'avant de l'expression chn
?chn [?]<Ret>	Recherche vers l'arrière de l'expression chn

2.1.5. Ajuster l'écran

- Ctrl+l : effacer et redessiner l'écran
- Ctrl+e, Ctrl+y : déplacer l'écran d'une ligne vers le haut, vers le bas
- Ctrl+u,Ctrl+d : déplacer l'écran d'une demi-page vers le haut, vers le bas
- Ctrl+b,Ctrl+f : déplacer l'écran d'une page vers le haut, vers le bas
- Ces commandes précédées par n s'exécutent n fois ex : 3C-b remonte de trois pages dans le texte
- z<Ent> : mettre la ligne courante en haut de l'écran
- nnz <Ent> : mettre la ligne nn en haut de l'écran
- nnz. : mettre la ligne courante (ou nn) au milieu de l'écran
- nnz- : mettre la ligne courante (ou nn) en bas de l'écran

2.1.6. Modifier le texte

Ces commandes sont bien évidement essentielles, c'est grâces à elles que vous allez pouvoir écrire du texte. Celles-ci demandent un petit temps d'adaptation mais une fois maitrisées elles vous sembleront tout à fait logiques :

Passage en mode insertion :

- Si n est tapé avant la commande, le texte inséré est répété n fois.
- "a" : insère du texte après le curseur, fin par <Esc>
- "A" : insère du texte après la fin de la ligne
- "i" : insère du texte avant le premier non-blanc de la ligne
- "o","O" : ouvre une ligne avant, après la ligne du curseur

Commandes accessibles en mode insertion :

- <Ech> : retour en mode commande
- <Ent> : passe a la ligne
- <Ret> ou Ctrl+h : efface le dernier caractère

- Ctrl+vcar : permet d'insèrer un caractère de commande (<Esc>, Ctrl+h, Ctrl+@, etc...)
- Ctrl+@ : répète la dernière insertion si le 1er caractère du mode
- Ctrl+w : efface le mot précédent
- Ctrl+t : indente d'une position à droite (v. option sw)
- "d" : coupe jusqu'au curseur déplace dd : la ligne
- "y" : copie jusqu'qu curseur déplace Y ou yy : la ligne
- "c" : change jusqu'au curseur déplace cc : la ligne

Exemple :
d3w coupe les 3 mots suivants, 4d3w coupe les 12 mots suivants (4 fois 3 mots),
4yy copie les 4 lignes suivantes.

2.1.7. Autres opérateurs

- p,P : colle après, avant le curseur
- [n]x,[n]X : coupe n caractères après, avant le curseur
- D,C : coupe ou change jusqu'en fin de ligne (v. d$ ou c$)
- ^ : change le caractère sous le curseur maj. <-> min
- [n]. : répète le dernier changement (n fois)
- [n]rcar : remplace le caractère sous le curseur par car (n fois)
- [n]Rchn <Ech> : remplace le texte a partir du curseur par chn (n fois)
- [n]schn <Ech> : substitue chn au caractère sous le curseur (n car)
- [n]Schn <Ech> : substitue chn a la ligne (n lignes a partir du curseur)
- [n]J : concatène deux lignes (n lignes)
- [n]<,> : indente de n car. a gauche, a droite (v. opt. sw)
- :[nn,nn]m n : déplace l'intervalle ou la ligne après la n-ième ligne
- :[nn,nn]t n : copie l'intervalle ou la ligne après la n-ième ligne
- "u" : Annule la dernière modification
- "U" : Annule les dernières modifications de la ligne courante
- "." : répète le dernier changement

L'utilisateur dispose de :
- 26 tampons de texte nommés de « a » à « z »
- 9 tampons des suppressions successives nommés de 1 à 9

2.2. Commandes avancées

2.2.1. Rechercher et remplacer

Voici les commandes qui vous permettront d'effectuer des recherches au sein de votre texte. Celles-ci peuvent s'effectuer grâce aux expressions régulières. Attention elles ne sont pas toutes équivalentes à celles utilisées avec le shell ou grep.

Recherche de délimiteur associé :

% avec curseur sur (,), [,], { ou }, va au second délimiteur

Recherche d'une lettre :
(voir f,F,t,T déplacer le curseur) ; : répète la dernière recherche par f,F,t,T , : comme ; mais en sens inverse

Recherche et remplacement de chaines :

On peut utiliser des expressions régulières qui acceptent les caractères spéciaux suivants :

- ^ : début de ligne (si 1er caractère de l'expression)
- $: fin de ligne (si dernier caractère de l'expression) et si l'option magic magic a été choisie
- . : joker, représente un caractère quelconque
- [chn] : représente un caractère contenu dans chn
- [^ chn] : représente un caractère non contenu dans chn
- [c1-c2] : représente un caractère entre c1 et c2 inclus
- \< , \> : représente le début, la fin d'un mot
- : cherche 0, 1 ou plusieurs occurrences du dernier car
- \(,\) : délimitent une sous-expression pour les remplacements
- & : remplace par la chaine trouvée
- \n : remplace par la n-ième sous-expression ...
- \car : remplace par car (permet d'obtenir /$%.*[&|~ et \ )

Recherche de chaine :

- /chn, ?chn : recherche l'expression chn en avant, en arrière
- / <Ent>, ? : répète la dernière recherche avant, arrière
- // : dernière expression reguliere entrée
- ~ : dernière expression de remplacement entrée
- /chn / ; //; // : recherche la troisieme occurence de chn
- /chn /+-n : va à la n-ième ligne après (+) ou avant (-) chn
- n : répète la dernière recherche dans le meme sens
- N : répète la dernière recherche en sens inverse

Remplacement de chaine :

- :s/chn1/chn2/ : substitue chn2 à chn1
- c : confirmation des substitutions
- g : substitue toutes les occurences de chn1 dans la ligne
- p : affiche les changements
- :[nn,nn]g/chn : va à la dernière ligne contenant chn [dnas nn,nn]
- :[nn,nn]g/chn /cmd : exécute cmd sur les lignes contenant chn [dans nn,nn]
- :[...]g !/chn /cmd : exécute cmd sur les lignes ne contenant pas chn

Voici une application typique de :g :
:g/chn1/chn2/g change chn1 en chn2 partout dans le texte

Les separateur après : g ou :s n'est pas obligatoirement /:s&/&\\&g change tous les / de la ligne en \

2.2.2. Macros et abréviations

Macros :

- :map chn cmd : exécute cmd lorsque chn est pressé
- :unmap chn : supprime la définition de la macro chn
- :map : affiche toutes les macros définies

Ces directives s'appliquent en mode commande seulement. Pour créer, supprimer ou afficher des macros valides aussi en mode insertion, il faut utiliser :map ! ou :unmap !.

Abréviations :

- :ab chn texte : dès que chn est insèré, remplace par texte
- :una chn : supprime la définition de l'abréviation chn
- :ab : affiche toutes les abréviations définies

La liste des macros et abréviations d'un utilisateur est souvent regroupée dans le fichier .exrc (voir Configuration). Cela permet par exemple de gérer les touches de fonction, les flèches et autres touches spéciales sur des terminaux non standard. Les touches inutilisées dans vi sont les touches de fonction et K V g q v * =.

Voici des commandes vous permettant d'obtenir l'état de votre fichier texte, le résultat s'affiche sur la ligne d'état, en bas de l'écran :

- :f ou C-g : affiche le nom de fichier courant et la position
- :.= : affiche le numéro de ligne courant (ligne du curseur)
- := : affiche le nombre de lignes du fichier
- :l : affiche les caractères spéciaux de la ligne courante

2.2.3. Commandes 'SHELL'

On peut lancer un interpréteur de commandes tout simplement en mettant en sommeil vi avec :stop ou C-z, retour par fg.

On nous propose également les commandes suivantes :

- :sh : lance un shell depuis vi, retour par C-d
- :so fich : exécute le script shell fich
- :! cmd : exécute la commande cmd
- % : remplace par le nom du fichier courant
- # : remplace par le nom du dernier fichier édité
- ! : remplace par la dernière commande lancee par : !

- :[nn] r! cmd : insère la sortie de cmd
- `:[nn,nn]w! cmd` : exécute cmd avec le texte édité en entrée

Déplacement depuis le curseur courant, voir Déplacer le curseur sur l'entrée de la commande cmd, puis remplace cette région par la sortie de cmd.

Exemple : 4 !+sort -n
Trie numériquement cinq lignes à partir du curseur, !Gexpand réécrit les tabulations avec des espaces depuis la position du curseur jusqu'à la fin du texte.

2.3. Configurer l'éditeur

Modifier les options de vi :

- :set opt : modifie l'otion opt. Pour une option de nom xxx,
- :set xxx : positionne l'option xxx à vrai
- :set noxxx : positionne l'option xxx à faux
- :set xxx=val : donne une valeur à l'option xxx
- :set xxx? : affiche la valeur de l'option xxx
- :set all : affiche toutes les options
- :set : affiche les options au demarrage et les changements

Voici une liste non exhaustive des options de vi. Une implémentation particulière peut toujours comporter des variantes sur les valeurs par défaut, ou des options non listées ici.

modifie ()[]{}laisse passer les messages write pas de vérification de droits d'accès
Nom abrev. defaut description
autoindent ai noai indentation automatique
autoprint ap ap affiche la ligne après c,d,j,m,:s,t,u
autowrite aw noaw sauve automatiquement le texte
directory dir dir=/tmp emplacement du tampon disque
errorbells eb noeb sonne lors des messages d'erreur
hardtabs ht ht=8 position des tabulateurs du terminal
ignorecase ic noic maj/min indifferenciees pour recherche
lisp - nolisp pour codage en lisp
list - nolist affiche tabulations et fins de lignes
magic - magic etend car. spéciaux en recherche
mesg - mesg laisse passer les messages write
number nu nonu numérote les lignes
open - open autorise open avec vi
optimize opt opt supprime retour chariot automatique
prompt - prompt invite : en mode ligne de cmd
readonly ro noro interdit la modification du fichier
remap - remap autorise macros dans des macros
report - report=10 taille maxi. modifications notifiees
scroll - scroll=11 nombre de lignes pour C-d et z
shell sh sh=$SHELL nom du shell pour ! et :sh
shiftwidth sw sw=8 nombre d'espaces pour indenter

showmatch sm nosm
ignorecase ic noic maj/min indifferenciees pour recherche
lisp - nolisp modifie ()[]{} pour codage en lisp
list - nolist affiche tabul
magic - magic etend car. spéciaux en recherche
mesg - mesg
number nu nonu numérote les lignes
open - open autorise open avec vi
optimize opt opt supprime retour chariot automatique
prompt - prompt invite : en mode ligne de cmd
readonly ro noro interdit la modification du fichier
remap - remap autorise macros dans des macros
report - report=10 taille maxi. modifications notifiees
scroll - scroll=11 nombre de lignes pour C-d et z
shell sh sh=$SHELL nom du shell pour ! et :sh
shiftwidth sw sw=8 nombre d'espaces pour indentershowmatch sm nosm indique le
separateur associe slowopen slow slow retarde l'affichage lors des insertions
tabstop ts ts=8 taille des tabulations C-i
taglength tl tl=0 taille minimum pour generation de tags (0 = pas de limites)
tags tags /usr/lib/tags emplacement du fichier de tags
term - term=$TERM nom du terminal utilise
terse - noterse messages d'erreurs raccourcis
timeout to noto 1s maxi. pour taper une macro
warn - warn <changements non enregistres>
window - window=n nombre de lignes de la fenetre texte
wrapmargin wm wm=0
retour automatique a n caractères de la margede droite
wrapscan ws ws les recherches a la fin du texte recommences au debut
writeany wa nowa

Si vous voulez activer une option automatiquement au démarrage, c'est dans /.vimrc qu'il faut la placer.

Voici un exemple de .vimrc :

```
syntax on
ab incl #include<>
ab main int main () {
ab for for(;;){ }
```

3. Recompiler son noyau

3.1. Préambule

Dans un premier temps, revenons sur ce qu'est un noyau (ou kernel).

Il s'agit, comme son nom l'indique, du coeur du système d'exploitation. C'est lui qui gère toutes les ressources matériel de l'ordinateur et leur accès : droits d'accès aux périphériques, à la mémoire, au CPU etc...

C'est donc le noyau qui détermine une partie importante des fonctionnalités du système via des pilotes (ou drivers).

Notez bien que la notion de pilotes inclus, mais ne se limite pas, à la gestion des périphériques matériels. Ainsi il y a un driver nommé "fat" qui permet au système de lire et d'écrire sur de partitions fat32 (Win9X).

On distingue deux "types" de noyaux selon la façon dont vous allez le compiler et le configurer, les noyaux monolithiques et les noyaux modulaires.

La distinction entre les deux vient du fait que certains drivers peuvent êtres soit en "build-in", c'est à dire dans la partie "statique" du noyau qui comporte tout ce qui est indispensable quelque soit le matériel, soit en module, c'est à dire sous forme de "bout de code" (objet : .o) chargeable dynamiquement.

Un noyau monolithique est donc un noyau où tous les drivers sont en "build-in" et un noyau modulaire est un noyau comportant un maximum de modules.

La bonne solution est bien entendu un mixe des deux.

Nous allons voir dans les chapitres suivants que ce choix vous appartient et quelles en sont les conséquences.

3.2. Pourquoi recompiler son noyau

Lorsque vous installez une distribution, Red Hat par exemple, faites un "lsmod", vous verrez la liste des modules chargés elle sera sans doute importante, la raison en est simple : il s'agit d'un noyau générique, c'est à dire qu'il est fait pour fonctionner avec un large panel de configurations hardware.

Cet aspect générique a deux inconvénients majeurs :

- Le noyau est compilé avec les "optimisations" i386, c'est à dire qu'il fonctionnera avec tout les processeurs compatibles i386 et supérieurs mais qu'il ne pourra pas

profiter des améliorations apportées pas les versions supérieures (registres et instructions supplémentaires).

- Un module nécessite une “interface” dans la partie statique du noyau, cette interface utilise de la mémoire et rend le noyau moins rapide.

Par conséquent recompiler votre noyau vous permettra de :

- Optimiser celui-ci pour votre processeur (athlon par exemple)
- Réduire son occupation mémoire en ne sélectionnant que les drivers correspondants à votre matériel actuel et éventuellement futur et ainsi en augmenter les performances.

D’autre part, il existe de nombreux patchs plus ou moins officiels permettant d’étendre les fonctionnalités du noyau linux et pouvant êtres appliqués aux sources de celui-ci.

Citons par exemple celui de freeswan qui permet le support du protocole IPSEC ou celui de poptop qui permet le support de la version “crypté” du protocole pptp (VPN de Microsoft(c)).

3.3. Note sur les modules

Les modules sont en théorie spécifiques à un noyau au sens d’une compilation et d’une version du kernel. Il est toutefois possible que certains modules une fois recompilés s’adaptent à une version d’un noyau, c’est le cas des drivers NVIDIA (closed source) mais ce n’est PAS le cas général.

Ces modules sont stockés sur le disque dans /lib/modules/KERNEL_VERSION/ par conséquent le fait de compiler sous forme de module le driver de disque sur lequel se trouve les modules peut poser problème.

D’une manière générale il vaut mieux compiler les drivers nécessaire à la lecture de vos disques en ”build-in” (driver du contrôleur, du type de file system et de disque). Si pour une raison ou une autre, cela n’est pas possible, il existe une méthode nommée initrd pour ”initial ramdisk” qui n’est pas traitée par ce document mais qui est couverte par de nombreux HOW-TO.

Un avantage des modules est aussi la possibilité de les charger/décharger en leur passant des paramètres. Par exemple pour le driver de carte contrôleur scsi adaptec 2940 (chip 7700) le module se nomme aic7xxx et peut prendre certaines options comme verbose, no probe etc...

C’est surtout pratique dans le cas de carte ISA dont on ne connaît pas forcément toutes les caractéristiques (DMA, IO, IRQ).

Depuis les versions 2.4.x du noyau, la commande *modinfo* permet d’obtenir des informations sur les modules de votre noyau en cours d’utilisation, pour cela il faut lui passer en paramètre le nom du module (donc sans le .o). Exemple :

```
modinfo aic7xxx
```

3.4. Configuration et compilation

Pour compiler votre propre noyau il vous faut, dans un premier temps, le télécharger sur l'un des nombreux serveurs ftp miroirs de ftp.kernel.org. Décompressez l'archive téléchargée dans /usr/src/ et créez un lien symbolique nommé "/usr/src/linux" pointant dessus.

Si vous avez déjà compilé un noyau et que vous voulez utiliser les mêmes sources vous pouvez taper les commandes "make clean" qui efface toute trace de compilation précédente hormis la configuration et "make mrproper" qui efface toute trace de compilation précédente.

Une fois ceci fait, placez-vous dans le répertoire contenant les sources et vous pouvez taper les commandes suivantes :

- **make config** : C'est l'interface de configuration en mode texte
- **make menuconfig** : C'est toujours du mode texte, mais avec des menus et des couleurs
- **make xconfig** : C'est l'interface de configuration sous X.

Remarque : Avant de compiler votre noyau tapez la commande :

```
export LANG=C
```

Ceci évitera des problèmes liés aux terminaux unicode (en particulier sous redhat 8.0 fr).

Une fois ceci fait, il ne vous reste qu'à vous promener dans les divers répertoires proposés en indiquant si vous souhaitez intégrer la fonctionnalité sur laquelle vous vous trouvez en statique dans le noyau ([X]), en module ([M]) ou pas du tout ([ ]).

Si vous n'êtes pas sûr des choix à faire vous pouvez consulter à la fois l'aide proposée, ou en trouver d'autres dans le répertoire "/usr/src/linux/Documentations/".

Si vous ne savez pas quel matériel est présent dans votre machine, il y a plusieurs moyens de le savoir. Tout d'abord, vous pouvez consulter les messages de démarrage avec la commande :

```
dmesg
```

Ou encore, vous pouvez consulter le contenu des divers fichiers présents dans le "/proc" (intéressez vous surtout aux fichiers ayant un nom explicite).

Pour le reste il y a quelques utilitaires comme "lspci" ou "lsisa" qui permettent de connaître le matériel présent sur la machine (en pci et isa).

Une fois que vous avez terminé votre sélection lorsque vous quittez l'interface, on vous demande de sauvegarder, ce que vous faites. Le fichier dans lequel seront enregistrées les diverses informations de votre futur noyau est "/usr/src/linux/.config".

Il vous faut maintenant créer diverses dépendances utiles pour la compilation avec :

```
make dep
```

Ensuite vous allez compiler la partie statique du noyau par la commande :

```
make bzImage
```

Cela va prendre un certain temps et placer le résultat de la compilation dans « /usr/src/linux/arch/*votre archi*/boot/bzImage".

Lorsque la compilation est terminée il faut compiler les modules et les installer :

```
make modules && make modules_install
```

Cette dernière phase va installer les modules dans "/lib/modules/*version du noyau*".

Si vous compilez plusieurs fois la même version du noyau, 2.4.19 par exemple, le "make module install" aura pour effet d'écraser /lib/modules/2.4.19/ ce qui peut ne pas être voulu. (pour les version 2.2 c'était pire : il copiais simplement en écrasant les version précédentes ce qui pouvais mélanger les anciens modules avec les nouveaux).

Pour éviter cela il suffit de créer une "sous version" du noyau en éditant le fichier "Makefile" de la racine des sources et en ajoutant un label dans l'option EXTRAVERSION, par exemple "EXTRAVERSION =-usb" puis de procéder comme précédemment :

```
make dep && make bzImage && make modules && make modules_install
```

Ainsi la version du noyau ne sera plus 2.4.19 mais 2.4.19-usb et ses modules seront placés dans le répertoire /lib/modules/2.4.19-usb/.

Voici le résumé de ce qu'il faut faire :

1. make clean ou make mrproper si besoin
2. make config, make menuconfig ou make xconfig
3. make dep
4. make bzImage
5. make modules
6. make modules_install

Une fois ces diverses étapes terminées, il va falloir indiquer à notre "bootloader" quel noyau utiliser pour démarrer le système.

4. Le “boot loader”

Le “boot loader” est une application lancée avant n’importe quel système d’exploitation qui va pouvoir gèrer comment tel ou tel système doit démarrer, avoir une liste avec un choix (typiquement, votre Unix et un Windows, ou encore plusieurs versions d’un noyau).

Il en existe de nombreux dont GRUB, Lilo, Silo...

Nous allons voir comment modifier les fichiers de configuration de GRUB et Lilo pour prendre en compte un nouveau noyau GNU/Linux (au hasard).

4.1. Lilo

Lilo (Linux Loader) est probablement le “boot loader” le plus répendu sous les distributions à base GNU/Linux. Son fichier de configuration (“/etc/lilo.conf”) ressemble à ceci :

```
$ cat /etc/lilo.conf
# Emplacement du secteur de boot
boot = /dev/hda

# Temps d’attente avant de démarrer sur l’image par defaut
delay = 10

# Quel système sur lequel il doit démarrer par defaut
default = Nouveau_noyau

# Emplacement de l’image du noyau sur le disque
image = /boot/vmlinuz-2.4.14

# Nom qui sera affiche dans le menu au moment du boot
      label = Vieux_noyau

# Quelle partition sur laquelle le noyau doit démarrer
      root = /dev/hda1
      read-only

# Ce qu’il faudrait rajouter avec un nouveau noyau
image = /boot/vmlinuz-2.4.16
      label=Nouveau_noyau
      root= /dev/hda1
      read-only

# Exemple d’entrée pour démarrer sur Windows
other = /dev/hda3
      label = Windows
      table = /dev/hda
```

```
$ lilo
Vieux_noyau
Nouveau_noyau *
```

Il est important, une fois qu'on a modifié la configuration de "lilo" de taper la commande du même nom pour que les données soient écrites où de droit.

4.2. GRUB

GRUB pour GRand Unified Bootloader utilise le fichier de configuration qui se trouve normalement dans "/boot/grub/menu.lst". Voici comment faire la même chose que lilo dans l'exemple précédent :

```
$ cat /boot/grub/menu.lst
# Temps d'attente avant de démarrer sur l'image par defaut
timeout 30

# Quel système sur lequel il doit démarrer par defaut
default 1

# Nom qui sera affiche dans le menu au moment du boot
title Vieux_noyau

# Quelle partition sur laquelle le noyau doit demarrer.
# Le comptage commence a 0, donc pour la premiere partition
# du premier disque dur c'est (hd0,0)
root (hd0,0)

# Emplacement de l'image du noyau sur le disque
kernel /boot/vmlinuz-2.4.14 root=/dev/hda1 ro

# Ce qu'il faudrait rajouter avec un nouveau noyau
title Nouveau_noyau
root (hd0,0)
kernel /boot/vmlinuz-2.4.16 root=/dev/hda1 ro

# Exemple d'entree pour un Windows
title Windows
rootnoverify (hd0,0)
makeactive
chainloader +1
```

Ici il n'y a pas besoin de taper une commande pour enregistrer les données. GRUB étant capable de lire les partitions sur lesquelles se trouve la configuration, ce n'est pas nécessaire.

Une fois les modifications des fichiers de configuration des "boot loader" terminées, il ne reste qu'à rebooter et selectionner ce sur quoi nous souhaitons démarrer.

5. Phases du “boot”

Lorsque le “boot loader” nous demande sur quel système démarrer, et que vous choisissez un Unix que se passe-t’il (nous n’allons pas descendre au niveau CPU et memoire) ?

La première étape consiste à charger le noyau en mémoire et d’en exécuter le code.

Lors de son exécution, il met en place un certain nombre de mécanismes de protection de la mémoire, et cherche ensuite sur la partition qu’on lui a indiquée comme contenant le répertoire racine “/”, un utilitaire qui se nomme “init”, qui lui-même va consulter son fichier de configuration (“/etc/inittab”) et en exécuter le contenu.

C’est à ce niveau que sont éxécutés les scripts de démarage pour la gestion du réseau, du montage des disques, des mises en place de terminaux virtuels...

Ce qui explique pourquoi, lorsqu’on liste les processus, il dipose d’un PID de 1 et qu’il est le père, grand-père, arrière-grand-père... de tous les autres processus.

En général il y a plusieurs modes d’initialisation symbolisée par des chiffres de 0 à 6. Ils ont tous une vocation particulière.

Certain vont nous permettre de démarrer en mode administration (“single user”) ou il n’y aura pas d’autres utilisateurs de connectés sur la machine, et qui ne pourront pas le faire dans ce mode.

Il peut permettre également de démarrer avec une authentification graphique, d’exécuter des scripts lorsqu’on souhaite redémarrer ou stopper notre machine, ou encore lorsque la machine reçoit un signal du système de sauvegarde qui arrive en fin de batterie...

Pour passer d’un mode d’init à un autre, il suffit de taper :

```
init id
```

Où l’id est le numéro d’initialisation souhaitée tel qu’il est defini dans “/etc/inittab”.

Chaque ligne du fichier “inittab” se compose comme suit :

```
id:niveaux_init:action:commande_ou_script
```

id
: Permet d’identifier la tâche à exécuter (nombre de caractères compris entre 1 et 4).

niveaux init
: Ce sont les numéros d’init dans lesquels le script ou la commande s’exécutera.

action
: Ici on spécifie comment doit réagir “init” : s’il doit attendre que la tâche du processus lancé soit terminée ou non, s’il doit “ressusciter” le processus lorsqu’il meurt.

commande ou script

C'est comme le "Port-Salut"

Voici un exemple d'inittab issue d'une Slackware :

```
# inittab This file describes how the INIT process should set up
# the system in a certain run-level.
#
# Version: @(#)inittab 2.04 17/05/93 MvS
# 2.10 02/10/95 PV
# 3.00 02/06/1999 PV
#
# Author: Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
# Modified by: Patrick J. Volkerding, <volkerdi@ftp.cdrom.com>
#
# These are the default runlevels in Slackware:
# 0 = halt
# 1 = single user mode
# 2 = unused (but configured the same as runlevel 3)
# 3 = multiuser mode (default Slackware runlevel)
# 4 = X11 with KDM/GDM/XDM (session managers)
# 5 = unused (but configured the same as runlevel 3)
# 6 = reboot

# Default runlevel. (Do not set to 0 or 6)
id:3:initdefault:

# System initialization (runs when system boots).
si:S:sysinit:/etc/rc.d/rc.S

# Script to run when going single user (runlevel 1).
su:1S:wait:/etc/rc.d/rc.K

# Script to run when going multi user.
rc:2345:wait:/etc/rc.d/rc.M

# What to do at the "Three Finger Salute".
ca::ctrlaltdel:/sbin/shutdown -t5 -rf now

# Runlevel 0 halts the system.
l0:0:wait:/etc/rc.d/rc.0

# Runlevel 6 reboots the system.
l6:6:wait:/etc/rc.d/rc.6

# What to do when power fails (shutdown to single user).
pf::powerfail:/sbin/shutdown -f +5 "THE POWER IS FAILING"

# If power is back before shutdown, cancel the running shutdown.
pg:0123456:powerokwait:/sbin/shutdown -c "THE POWER IS BACK"

# If power comes back in single user mode, return to multi user mode.
ps:S:powerokwait:/sbin/init 3

# The getties in multi user mode on consoles an serial lines.
#
# NOTE NOTE NOTE adjust this to your getty or you will not be
# able to login !!
#
```

```
# Note: for 'agetty' you use linespeed, line.
# for 'getty_ps' you use line, linespeed and also use 'gettydefs'
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
c5:1235:respawn:/sbin/agetty 38400 tty5 linux
c6:12345:respawn:/sbin/agetty 38400 tty6 linux

# Runlevel 4 used to be for an X window only system, until we discovered
# that it throws init into a loop that keeps your load avg at least 1 all
# the time. Thus, there is now one getty opened on tty6. Hopefully no one
# will notice. ;^)

# It might not be bad to have one text console anyway, in case something
# happens to X.
x1:4:wait:/etc/rc.d/rc.4
# End of /etc/inittab
```

La valeur de l'init sur laquelle démarrer par defaut est définie au début du fichier avec "initdefault".

Nous pouvons constater qu'à la fin du fichier, un utilitaire est lancé du nom de "agetty". C'est lui qui va gèrer les sessions des terminaux virtuels au niveau physique.

6. Outils système

6.1. Commandes d'informations sur la machine

uname : Affiche le nom du système, et d'autres informations utiles :

```
[mathieu@mgtnode /]$ uname -a
Linux mgtnode 2.4.9-31 #1 Tue Feb 26 06:07:30 EST 2002 alpha unknown
```

hostname : Affiche le nom de la machine :

```
[mathieu@math mathieu]$ hostname
math.labo-unix.org
```

ifconfig : Affiche les informations sur les interfaces réseaux :

```
[mathieu@battousai mathieu]$ /sbin/ifconfig -a
eth0  Link encap:Ethernet HWaddr 00:40:05:A5:F7:8C
      inet addr:192.168.1.2 Bcast:192.168.1.255 Mask:255.255.255.0
      UP BROADCAST NOTRAILERS RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:3879 errors:0 dropped:0 overruns:0 frame:0
      TX packets:3695 errors:0 dropped:0 overruns:0 carrier:0
      collisions:24 txqueuelen:100
      RX bytes:826709 (807.3 Kb) TX bytes:601079 (586.9 Kb)
      Interrupt:5 Base address:0xd000

lo    Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      UP LOOPBACK RUNNING MTU:16436 Metric:1
      RX packets:1727 errors:0 dropped:0 overruns:0 frame:0
      TX packets:1727 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:357651 (349.2 Kb) TX bytes:357651 (349.2 Kb)
```

w : Affiche qui est en ligne, et ce qu'ils font.

```
[mathieu@battousai mathieu]$ w
3:56pm up 3:45, 5 users, load average: 0.12, 0.05, 0.01
USER     TTY   FROM              LOGIN@   IDLE  JCPU  PCPU  WHAT
mathieu tty1  -                  12:26pm  3:29m 2:03  0.00s /bin/sh
mathieu pts/0 -                  12:27pm  2:56m 0.07s 0.07s -bash
mathieu pts/1 -                  12:27pm  2:27m 0.00s 0.00s -bash
mathieu pts/2 math.labo-unix.org 3:42pm   1:08  0.03s 0.03s -bash
mathieu pts/3 math.labo-unix.org 3:55pm   0.00s 0.05s 0.01s w
```

free : Affiche la mémoire libre, et utilisée.

```
[mathieu@battousai mathieu]$ free
      total         used      free     shared   buffers     cached
Mem:  516436        485640    30796    0        51864       341028
-/+ buffers/cache:  92748     423688
Swap: 205800        0         205800
```

6.2. Plannification des tâches

On peut utiliser plusieurs commandes, suivant les besoins :

sleep :

```
[mathieu@battousai mathieu]$ date
Fri Oct 18 16:00:44 CEST 2002

[mathieu@battousai mathieu]$ date && sleep 3m && date
Fri Oct 18 16:00:56 CEST 2002
Fri Oct 18 16:03:56 CEST 2002
```

at : qui permet de lancer une ou plusieurs commandes à une heure précise.

6.3. Le programme cron de Paul Vixie

Ce programme permet d'exécuter périodiquement des tâches, même si l'utilisateur n'est pas connecté.

Chaque utilisateur possède un fichier "crontab", où sont recensés les tâches à exécuter. Un daemon (processus tournant en tâche de fond) "crond" doit être lancé pour que les tâches puissent être exécutés.

6.3.1. Format du fichier crontab

Pour plus d'informations, referrez vous aux pages de man (man cron, man crontab, man crond).

Une entrée dans la crontab doit être sur une ligne, et est composé de 5 champs délimités par des espaces. Après ces 5 champs, la ligne de commande doit etre placée.

Les différents champs sont :
– minute : 0-59
– heure : 0-23
– jour du mois : 1-31
– mois : 1-12
– jour de la semaine : 0-6 (sachant que 0 signifie dimanche)

Les valeurs peuvent être séparées par une virgule, et le caractère "*" signifie toutes les valeurs possibles.

6.3.2. Utilisation

Vous êtes l'administrateur d'une machine. Vous voulez que le script /usr/local/bin/save.sh soit exécuté tous les jours à 23H. De plus, vous voulez que le script gestapo.sh soit lancé

toutes les 2 minutes, du lundi au vendredi, afin de vérifier si vos utilisateurs possédent des fichiers prohibés (mp3/divx/...) sur leur compte. Voici la procédure à suivre :

On veut utiliser vim pour éditer la crontab :

```
[root@escaflowne mathieu]# export EDITOR=vim
```

On liste les tâches déjà précisées dans la crontab :

```
[root@escaflowne mathieu]# crontab -l
crontab: no crontab for root
```

Editons la crontab :

```
[root@escaflowne mathieu]# crontab -e

# ceci est un commentaire nous sommes dans le fichier temporaire du crontab
# lancement du script save.sh

#Min  heure Datedumois  Mois  JourDelaSemaine   commande
*     23    *           *     *                 /usr/local/bin/save.sh

# lancement du script gestapo.sh
*/2   *     *           *     1,2,3,4,5         /usr/local/bin/gestapo.sh
```

Après avoir éditer le fichier temporaire, sauvegardez le et quittez. Crontab vous annoncera que les changements ont été bien pris en compte.

6.4. RPM

Les RPMS sont des paquetages contenant des logiciels (à savoir des fichiers exécutables, des librairies, des fichiers de conf, etc). La commande **rpm** permet d'installer, d'effacer, de mettre à jour et de construire des paquetages, mais aussi de consulters les différents paquetages installés sur la machine.

Lorsque l'on veut installer ou mettre à jour un paquetage, le programme rpm va vérifier si le système dispose de suffisament d'espace disque, et si tous les paquets dont dépend le programme à installer sont présents.

Le programme rpm gére une base de données où sont stockés toutes les informations sur les paquets.

6.4.1. Utilisation de la commande rpm

Pour plus d'informations, referrez vous aux pages de man (man rpm).

Installation d'un paquet (vous devez être root) :

```
[root@bleu mathieu]$ rpm -ivh WineX-2.1-1.i386.rpm
```

Notez que la convention de nomenclature est construite sur le modele suivant : Nomduprogramme-version-architecture.rpm

Mise à jour d'un paquet (vous devez être root) :

```
[root@bleu mathieu]$ rpm -Uvh WineX-2.3-1.i386.rpm
```

La "mise à jour" peut donner lieu à une installation pure et simple même si une ancienne version de WineX n'était pas présente.

Effacer un rpm (vous devez être root) :

```
[root@bleu mathieu]$ rpm -e WineX
```

6.4.2. Obtenir des informations avec la commande rpm

Connaitre la version d'un paquet installé :

```
[mathieu@battousai mathieu]$ rpm -q WineX
WineX-2.1-1
```

Connaitre tous les paquets ayant le mot "mot" dans leur nom (en utilisant les pipes) :

```
 [mathieu@battousai mathieu]$ rpm -qa | grep glib
glibc-devel-2.2.5-34
glib10-1.0.6-10
glib2-devel-2.0.1-2
glibc-common-2.2.5-34
glib-devel-1.2.10-5
glibc-2.2.5-34
glibc-kernheaders-2.4-7.14
glib-1.2.10-5
compat-glibc-6.2-2.1.3.2
glib2-2.0.1-2
```

Connaitre à quel paquet appartient le fichier "fichier" :

```
[mathieu@battousai mathieu]$ rpm -qf /lib/libdb.so
db3-devel-3.3.11-6
```

Connaitre le changelog (liste de changements apportés depuis la création du paquet) d'un paquet (ici db3) :

```
[mathieu@battousai mathieu]$ rpm -q --changelog db3
```