

Linux – Initiation

Juillet 2023

Michel Blache

Votre Animateur

Michel Blache

Intervenant pour le Groupe IFPA

Chef de Projet Infrastructure Airbus, AFI, EFS

Technologies :

- Linux
- ITIL
- Merise, UML
- Database & SQL
- Office

Parlez-moi de vous

- Employeur
- Fonction
- Utilisation de l'outil informatique
- Besoins
- Attentes

Objectifs de formation

- Etre capable de travailler sur une station de travail ou un serveur linux / Unix
- Points abordés :
 - Le système d'exploitation Linux/Unix
 - Caractéristiques de Linux
 - Commandes de prise en main
 - Interpréteurs de commandes
 - Les répertoires et les chemins d'accès
 - Installation d'un poste sous Linux

Public & Prérequis

- Etre familiarisé avec les périphériques et le matériel informatique.

Programme

1. Présentation
2. Principes
3. Prise en main du système
4. Le Shell
5. L'éditeur nano
6. Le système de fichiers Linux
7. Gestion des utilisateurs
8. Les processus
9. Les services / daemons
10. Les Files System
11. Sauvegarde et Package
12. Les scripts SHELL

Présentation

Présentation

UNIX

- 1969 – Ken Thompson et Dennis Ritchie écrivent une première version du noyau pour les laboratoires Bell
- 1970 - Premières versions du système UNICS
- 1973 – Dennis Ritchie invente le langage C et réécrit le noyau en C
- Distribution libre dans les universités
- AT&T veut commercialiser le système
- L'université de Berkeley lance UNIX BSD (1977)
- Mise en réseau avec TCP/IP (BSD 4.2 et 4.3) (1983-1987)

Présentation

UNIX

- 1983 – le système AT&T s'appelle UNIX System V
- 1987 – AT&T Unix System V.3
- HP développe HP-UX, IBM : AIX
- 1990 – AT&T et SUN sortent UNIX System V.4
- Dès 1988, les autres grands constructeurs créent l'Open Software Foundation (OSF) : OSF/1 en 1991
- System V (SCO Xenix AIX Solaris BSD Réseau)
- Démarrage de nombreux projets libres (FreeBSD et Linux)

Présentation

LINUX

- Noyau développé par un étudiant finlandais : Linus Thorvald.
- Le noyau accompagné de nombreux composants : une DISTRIBUTION (OS complet)
- Disponible pour tout type d'architecture (x86, Power, Sparc, PA-RISC, Alpha, Mips, Itanium...)
- Gratuit

Présentation

LINUX

- Principales distributions
- RedHat (RHEL, Fedora, Centos)
- Mandriva (ex Mandrake)
- SUSE (appartient à Novell)
- Debian
- Ubuntu (basée sur Debian)

Présentation

LINUX

Avantages

- Système multitâche multi-utilisateurs conçu à l'origine pour des mainframes
- Le noyau est écrit en C (portabilité)
- Tout est fichier. Gestion des périphériques (clavier, souris écran, disque dur etc ...)
- Un programme est chargé d'assurer le dialogue avec les utilisateurs c'est le Shell
- Interface graphique X Windows.
- Toutes les sources sont fournies, il est libre, stable, réactif, gratuit, compatible Posix

Présentation

LINUX

Inconvénients

- Il est austère et touffu
- Pas ou peu de supports techniques
- Il n'a pas tous les drivers

Principes

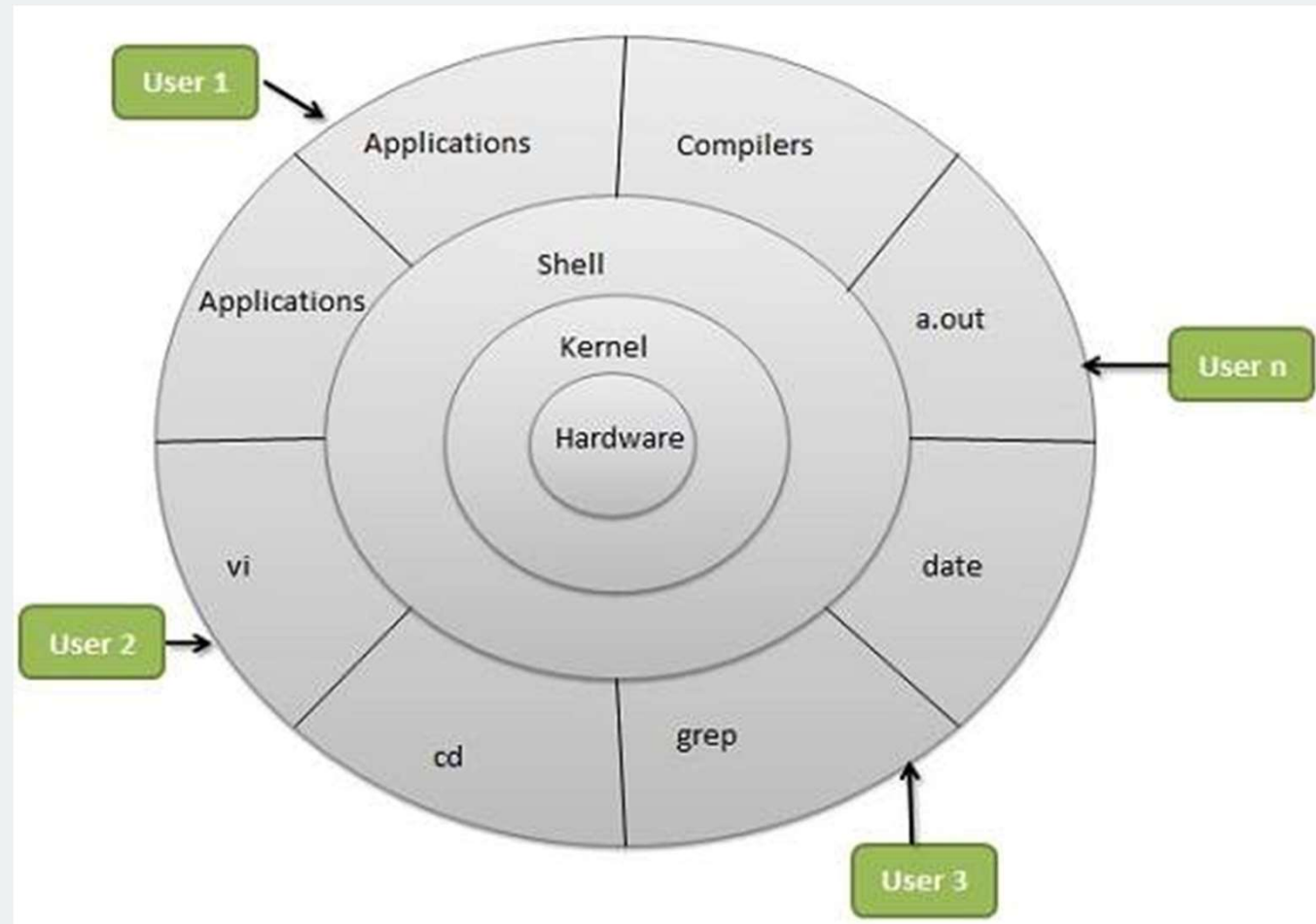
Principes

- Le noyau gère les tâches de base du système :
 - L'initialisation du système
 - La gestion des ressources
 - La gestion des processus
 - La gestion des fichiers
 - La gestion des entrées / sorties

L'utilisateur communique avec le noyau par l'intermédiaire d'un programme appelé le « shell ». Le(s) shell(s) sont aussi des langages de commande et de programmation

Principles

Structure:



Principes

Démarrage :



BIOS

Binary Input Output System

MBR

Master Boot Record

GRUB

GRand Unified Bootloader

LILO

Linux Loader

Principes

Démarrage :

- Lancement du système : boot et chargement du noyau
- Au boot le BIOS exécute le MBR (Master Boot Record) situé sur le premier secteur (512 octets) du support bootable choisi (disque, CD, clef USB, ...)
- Le MBR :
 - scanne le disque pour trouver la partition bootable (flag)
 - lance le boot loader du secteur de boot (premier secteur) de la partition bootable
- Le bootloader (chargeur de démarrage) :
 - charge le noyau en mémoire et l'exécute
 - charge le ramdisk initrd.img en mémoire
- 2 bootloader possibles:
 - LILO (LIinux LOader)
 - GRUB (GRand Unified Bootloader)

Prise en main du système

Prise en main du système

Démarrage :

Démarrez votre machine

Après un éventuel Lilo ou Grub (au boot de la machine) vous arrivez au Login

Entrez votre nom de login et votre password et vous êtes connecté au système.

Si vous tapez logout vous repassez en mode login

Prise en main du système

Arborescence :

L'arborescence a une racine ou un début; il est noté / on l'appelle aussi **root**.

On trouve sous root toujours les mêmes répertoires; ils ont un rôle bien défini.

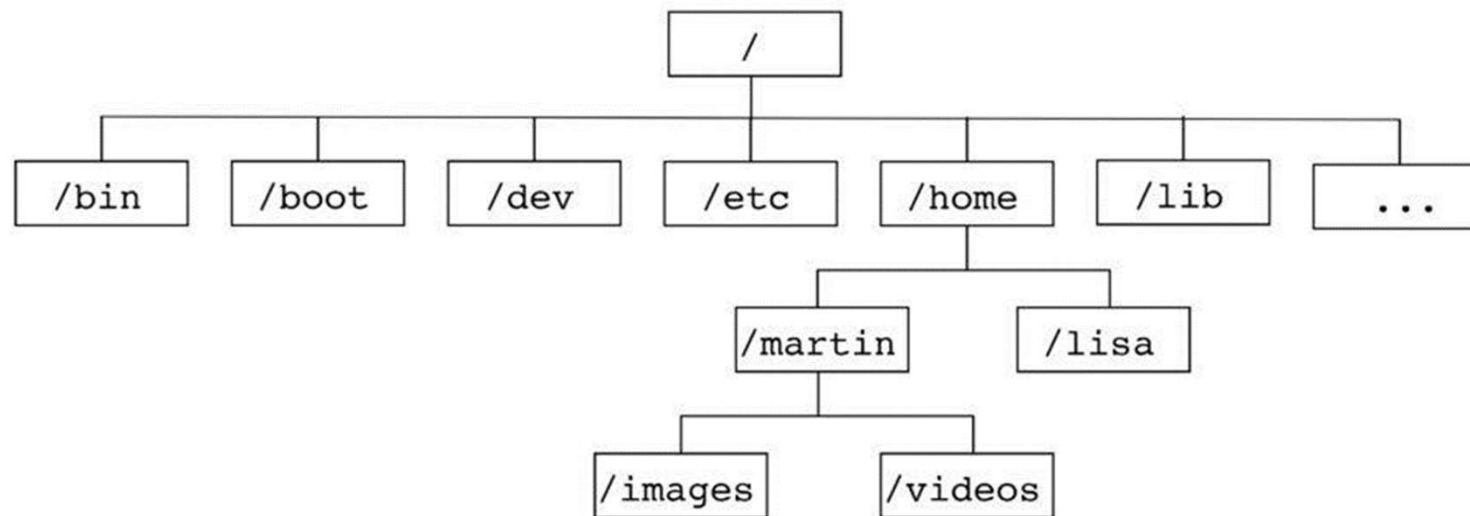
Nous reviendrons sur les commandes suivantes :

Tapez : `cd /` pour aller à la racine

Puis tapez : `ls -al` pour lister le contenu de la racine

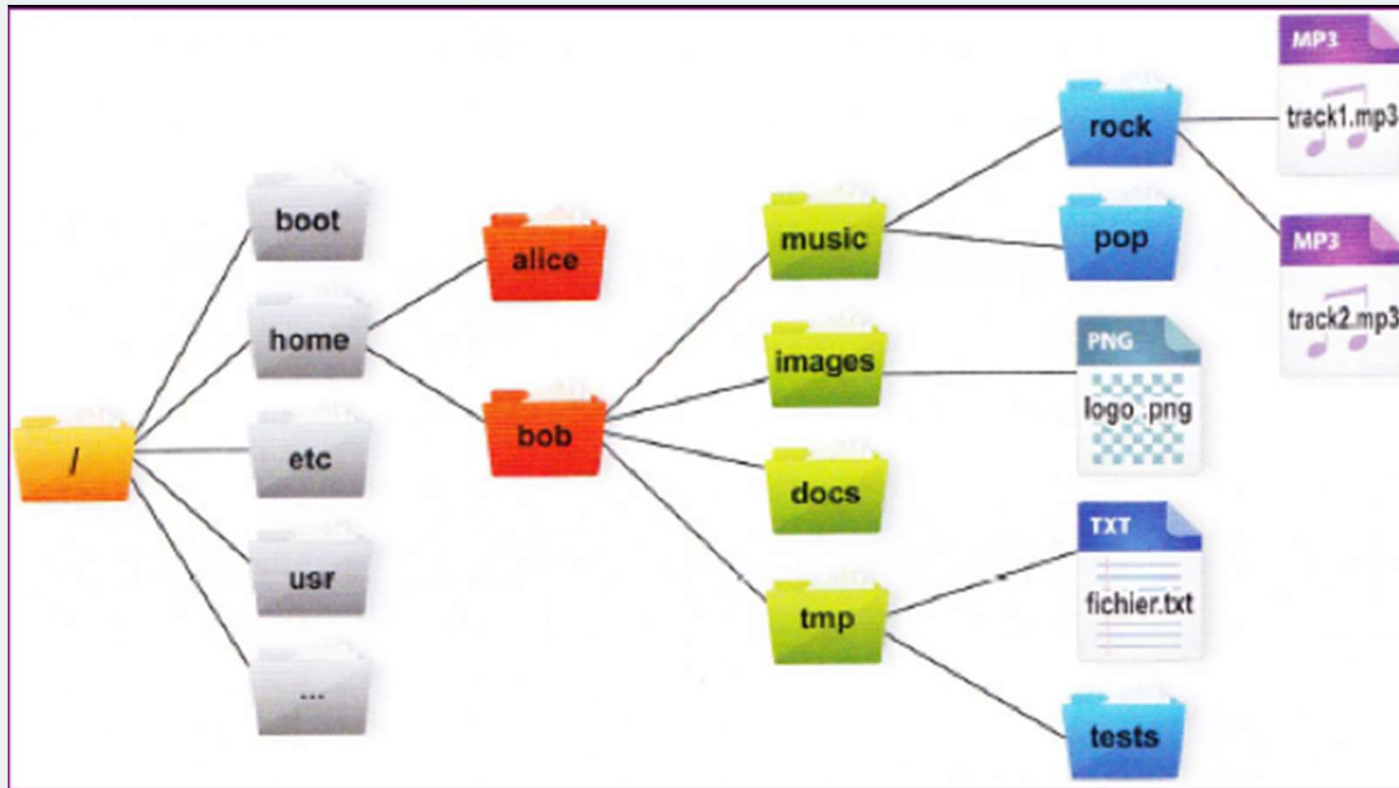
Prise en main du système

Arborescence :



Prise en main du système

Arborescence :



Prise en main du système

Arborescence :

/etc	Tous les fichiers de configuration sont rangés ici...
/bin	Les programmes importants.
/sbin	Les programmes nécessaires au démarrage de la machine ! <i>(Le S ici sous entend que les programmes sont statiques. (Compilés pour la machine))</i>
/boot	Le noyau est parfois rangé ici(noyau = cœur du système d'exploitation)
/usr	Ici, des programmes et des informations plus propre à cette machine...
/usr/bin	Programmes moins important que /bin...
/usr/local	Logiciels propres à cette machine (jeux, services, etc)
/home	Répertoires des utilisateurs.
/home/...	Chaque utilisateur a un répertoire pour stocker ses données. <i>(Il a les droits sur ce répertoire)</i>

Prise en main du système

Arborescence :

/tmp	Comme son nom l'indique... Accessible à tous
/var	Administration... Souvent les traces et suivi d'événements...
/var/log	L'historique des connexions, incidents, utilisation, requêtes, accès....
/var/spool	Le tampon d'impression ou de courrier par exemple
/mnt	Un répertoire de montage (pour l'accès aux périphériques amovibles...

Maintenant, c'est /media qui a ce rôle (à cause du programme udev, qui crée à la volée le répertoire correspondant à chaque ressource détectée)

Prise en main du système

Initiation aux droits :

Linux étant multi-utilisateurs, il offre la possibilité de sécuriser l'accès aux données.

Chaque utilisateur à une **home directory** qui lui est réservé au moment de la création de l'utilisateur. Ex : /home/tarzan

tarzan peut y enregistrer des données et gérer sa propre sécurité

L'administrateur système (**root**) est chargé de l'administration de la machine, il a lui tous les droits.

Un utilisateur appartient a un **groupe**

Vous disposerez donc d'un nom de login, d'un **UID** (User Identifier, numéro unique) et d'un **GID** (Group identifier).

Prise en main du système

Démarrage et arrêt de la machine:

L'ensemble des services est lancé une fois pour toute au démarrage, ce qui explique la multitude de message qui apparait au départ.

Au démarrage l'accès aux supports magnétiques sera lancé et mis en cache ce qui offrira des temps de réponse très courts.

Pour l'arrêt de la machine taper la commande "**shutdown**".

Prise en main du système

Ecrans virtuels:

Linux est un vrai système multi-utilisateurs. Il est donc possible de s'y connecter plusieurs fois.

En standard nous disposons de 6 écrans texte et 1 écran graphique.

La touche ALT-F1 vous donne accès au premier écran dit "tty1" (tty pour télétype)

La touche ALT-F2 vous donne accès au deuxième écran dit "tty2"

La touche ALT-F7 pointe sur X Windows

Pour ressortir de X Windows il faut faire ALT CTRL Fx

Prise en main du système

Démarrage et arrêt de session:

Login : Demande de votre nom de connexion (attention à la casse Maj/Min); tapez ensuite votre mot de passe. Login + mot de passe seront comparés avec les informations contenues dans le fichier `/etc/passwd` et `etc/shadow`.

Si tout est ok vous vous retrouverez sur votre home directory et vous pourrez alors taper des commande à l'aide du Shell qui aura été précisé dans la fichier `/etc/passwd`.

Commande **who** : Fournit des informations sur l'ensemble des utilisateurs qui sont actuellement connectés sur la station.

Commande **logout** : Fin de connexion et retour login

Commande **exit** : Sortie du programme en cours

Prise en main du système

Le manuel "man":

Consulter le manuel : `man [n]` commande

Visualisation à l'écran des informations concernant la commande spécifiée. L'affichage est réalisé par un `more` pour la pagination.

Le manuel est divisé en huit sections:

1 : Les commandes utilisateurs

2 : Les appels systèmes

3 : La librairie des sous-routines

4 : Les formats de fichiers

5 : Les fichiers spéciaux

7 : Les possibilités diverses

8 ou 1m : Les commandes d'administrations système

9 : glossaire

On peut spécifier la section dans laquelle on veut effectuer la recherche (grâce au paramètre `n`).

Puisqu'on est en mode pagination "`more`" taper "`q`" pour quitter et espace pour paginer.

Le shell

Le shell

- Le Shell est un programme (application), qui assure l'interface entre les différents programmes et la machine
- Interprète les commandes
 - Gère les I/O utilisateur sur le terminal
 - Mémore le setup de l'environnement de l'utilisateur dans le fichier .profile
- Les utilisateurs communiquent avec sh
- Le shell utilisé sur votre console est en général le bash (bourne again shell de Brian Foc). Publié la 1^{ère} fois en 1969.

Le shell

Commandes généralités

- **commande - option a b c – argument x y z**

Exemple : `ls -al /etc`

Ce qui veut dire que les option -a et -l seront appliquées sur l'argument /etc qui sera le répertoire à visualiser

Le shell

Commandes de répertoires

pwd	Print Working directory, répertoire actif
ls	affiche le contenu du répertoire (ls -al ls -Ral ls -lisa
cd	Change directory. cd .. Permet de remonter d'un niveau. cd tout seul vous repositionne en home directory
mkdir	création de répertoire
rmdir	suppression de répertoire (vide)

Le shell

Commandes de fichiers

ls	attention aux fichiers cachés qui commencent par un "." option -a et -l
file	pour connaître le type de fichier (Ascii text, executable, link ..)
cp	copie le ou les fichiers origine arrivée
mv	copie comme cp mais efface l'origine
rm	supprime le ou les fichiers
cat	affiche le contenu des fichiers (more & less)
touch	permet de changer la date de dernier accès à un fichier sans modifier son contenu. Utile pour les sauvegarde. Touch crée le fichier s'il n'existe pas, il sera vide (0 caractère)

Le shell

Les aides

man	permet d'avoir accès à l'aide sur cette commande. Toujours présenté selon une norme (Nom, Synopsys, Description, Option, Bugs)
apropos	propose une ligne d'information sur une commande
info	permet d'accéder à l'aide hypertextuelle plus complète que man.
--help	aide succincte de la commande. Ex "ls --help"

Le shell

L'entrée et la sortie standard

stdin ou 0 stdout ou 1 stderr ou 2

Les stdin pointeurs constants, stdout et stderr sont des flux standard pour l'entrée, la sortie et la sortie d'erreur.

En général c'est le clavier qui est considéré comme le stdin, l'écran est assigné à stdout et stderr.

- > pour rediriger le stdout en créant un fichier (s'il existe, il sera écrasé)
- >> pour rediriger le stdout vers un fichier existant. (s'il n'existe pas il sera créé)
Ex : ls >> /home/tarzan/liste.txt
- 2> redirige les erreurs vers un fichier
- < pour redirection d'entrée ex : ls < "nom d'un fichier" contenant le chemin d'un répertoire.

Le shell

Pipes et commandes filtres associées

Il est possible d'enchaîner des commandes afin qu'une commande travaille sur le résultat de la commande précédente avec le pipe |.

En fait on branche le stdout de la première sur le stdin de la suivante.

Exemple :

```
cd /
```

```
Ls -Ral | more
```

On évite ainsi de passer par des fichiers temporaires.

Le shell

Les filtres

more, less filtre de pagination (less permet de revenir en arrière.)
exemple : `ls -Ral | less`

grep recherche dans un flux les lignes qui correspondent au pattern demandé. Exemple : `ls-Ral | grep tarzan | more`

head et tail ne laisse passer que le début et la fin du flux
`tail .bashrc` ou `ls -al | head`

tr substitue un caractère par un autre dans le flux.
exemple : `ls -al | tr janv. févr.`

Le shell

Les filtres (suite)

sort tri les lignes du flux. Exemple : `cat /etc/passwd | sort`

uniq élimine les lignes en double dans le flux. *Attention il faut faire préalablement un sort.*

Exemple : `cat /etc/passwd | sort | uniq`

wc (word count) compte les caractères, les mots et les lignes du flux.

Exemple : `wc /etc/passwd`

Exemple : `ls -al | wc`

Le shell

su et sudo

su (substitute user ou switch user) est une commande Unix/Linux permettant d'exécuter un interpréteur de commandes en changeant d'identifiant de GID et de UID. Sans argument, la commande utilise les UID 0 et le GID 0, c'est-à-dire ceux du compte utilisateur **root**.

sudo (abréviation de superuser do, en anglais est une commande et un utilitaire informatique utilisé dans les systèmes d'exploitations de type Unix et Linux. Elle permet d'endosser l'identité d'un autre utilisateur sans se déconnecter. Cette commande utilisée sans login permet par défaut de prendre l'identité de **root**.

Le shell

Pour la route

w	lister les utilisateurs connectés
who	fournit des informations sur l'ensemble des utilisateurs qui sont actuellement connectés sur la station
who am i	renvoie uniquement les informations relatives à l'utilisateur courant
whoami	renvoie l'identificateur de l'utilisateur courant
id	renvoie l'UID (user identifier), le GID (Groupe identifier) de l'utilisateur courant
date	renvoie la date système
cal	imprime le calendrier ex: cal 7 2019 ou cal tout cours cal -y -2000

Il ne faut pas confondre who am i (cas particulier de la commande who) et whoami. Le premier donne des informations sur l'utilisateur connecté et le second l'identificateur de l'utilisateur courant

L'éditeur nano

L'éditeur nano

Comment installer l'éditeur de texte nano sous Debian

Pour vérifier la présence de nano, il suffit d'utiliser la commande suivante :

```
nano --version
```

Si vous voyez un résultat qui vous indique un numéro de version, vous pouvez sauter cette étape.

Installer nano sur Debian ou Debian/Ubuntu

```
sudo apt-get install nano
```

L'éditeur nano

Comment ouvrir et fermer l'éditeur nano

Pour ouvrir nano ou éditer un fichier : **nano nom_de_fichier**

Pour ouvrir un fichier nommé demo.txt : **nano demo.txt**

Vous pouvez ouvrir différents types de fichiers, comme .txt, .php, .html, etc. N'oubliez pas que si vous souhaitez ouvrir un fichier spécifique, vous devez être dans le répertoire où se trouve le fichier.

Si vous vous trouvez dans un autre dossier et que vous souhaitez ouvrir un fichier (demo.txt) dans /chemin, entrez la ligne suivante à la place :
nano /chemin/demo.txt

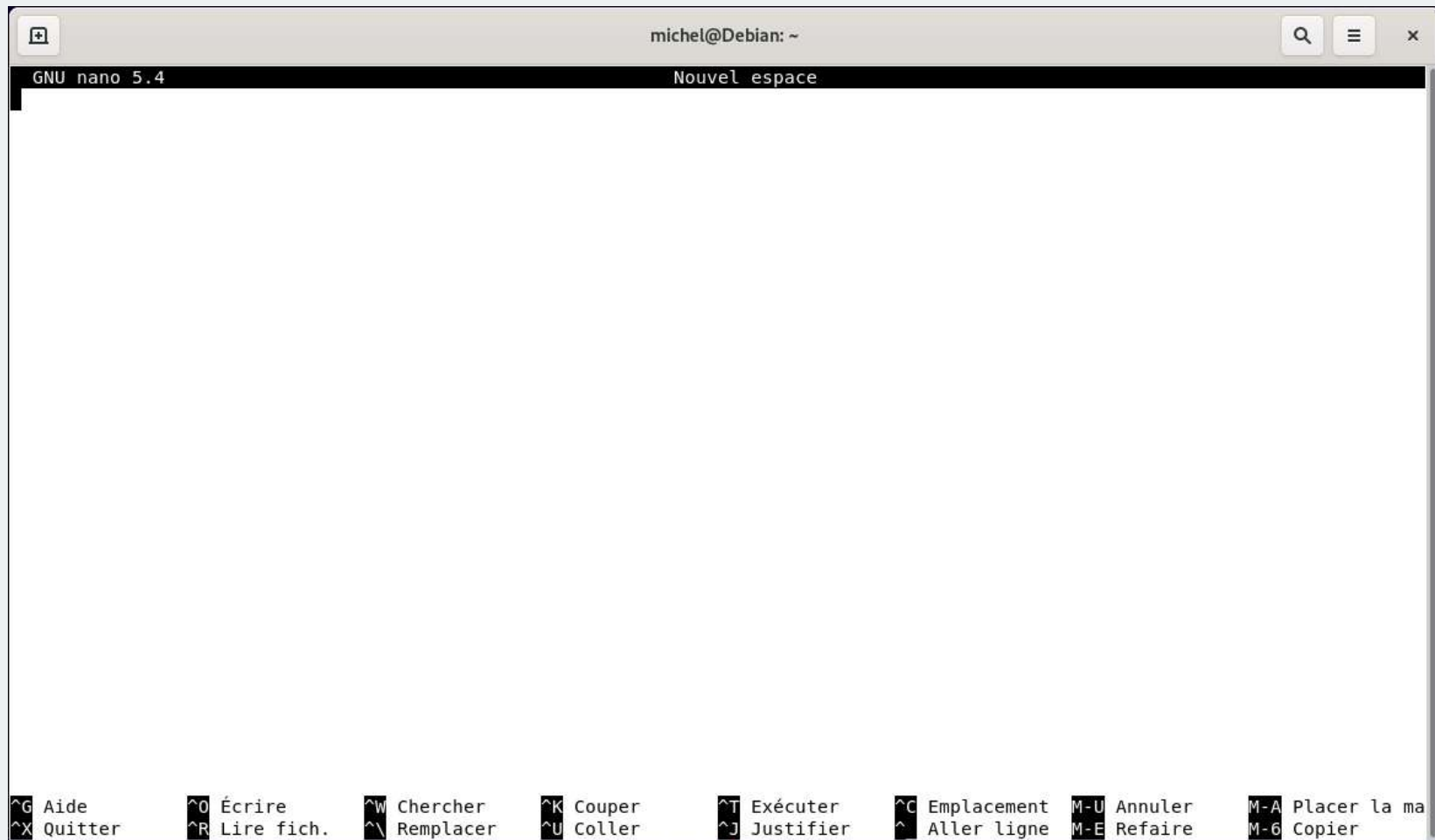
L'éditeur nano

Comment ouvrir et fermer l'éditeur nano (suite)

Si vous entrez un nom de fichier et que ce fichier n'est pas présent dans le répertoire, nano créera un nouveau fichier. En revanche, si vous exécutez uniquement la commande nano sans spécifier le nom du fichier, l'éditeur de texte Nano créera un fichier vide sans nom et vous demandera un nom lorsque vous quitterez l'éditeur.

Après l'exécution de la commande nano, une nouvelle fenêtre s'ouvrira dans laquelle vous pourrez librement éditer le fichier. Voici à quoi ressemble l'interface de l'éditeur de texte nano. Il suffit d'utiliser les touches fléchées de votre clavier pour déplacer le curseur sur le texte.

L'éditeur nano



L'éditeur nano

Présentation de nano

Au bas de cette fenêtre, vous verrez certains des raccourcis utilisables avec l'éditeur Nano.

Le symbole **^** signifie que vous devez appuyer sur **CTRL + [Touche]** pour lancer la commande choisie. Voici quelques exemples.

Appuyez sur **CTRL + O** pour enregistrer les modifications apportées au fichier et poursuivre l'édition.

Pour fermer l'éditeur, appuyez sur **CTRL + X**. S'il y a des modifications, il vous sera demandé si vous souhaitez les enregistrer ou non. Saisissez Y pour Oui, ou N pour Non, puis appuyez sur Entrée. Mais s'il n'y a pas de changements, vous quitterez l'éditeur immédiatement.

L'éditeur nano

Comment rechercher et remplacer du texte

Pour effectuer une recherche dans le texte, appuyez sur CTRL + W. Insérez votre texte et appuyez sur Entrée. Pour continuer à rechercher la même texte, utilisez ALT + W.

Si votre objectif est de trouver et de remplacer un texte, appuyez sur CTRL + W puis CTRL + R pour entrer le texte que vous voulez rechercher et le texte qui le remplacera. L'éditeur vous amènera alors à la première instance du texte. Vous pouvez appuyer sur Y pour remplacer un texte ou sur A pour remplacer toutes les instances.

Si vous souhaitez revenir en arrière après avoir tapé un raccourci, il vous suffit d'utiliser CTRL + C pour annuler le processus en cours.

L'éditeur nano

Raccourcis couramment utilisés pour éditer un texte dans nano

Pour **sélectionner** un texte, allez au début du texte souhaité et appuyez sur ALT + A. Cela permet de marquer la sélection. Ensuite, vous pouvez vous déplacer sur le texte à l'aide des touches fléchées.

Appuyez sur ALT + 6 pour **copier** le texte sélectionné dans le presse-papiers.

Pour **couper** le texte sélectionné, appuyez sur CTRL + K.

Si vous voulez **coller** le texte, naviguez à l'endroit voulu et appuyez sur CTRL + U.

L'éditeur nano

Commandes de base de l'éditeur de texte Nano

COMMANDE	EXPLICATION
----------	-------------

CTRL + A	Lancer une ligne
----------	------------------

CTRL + E	Terminer une ligne
----------	--------------------

CTRL + Y	Faire défiler la page vers le bas.
----------	------------------------------------

CTRL + V	Faire défiler la page vers le haut.
----------	-------------------------------------

CTRL + G	Cette commande ouvre une fenêtre d'aide qui contient des informations concernant toutes les commandes que vous pouvez utiliser
----------	--

CTRL + O	Pour sauvegarder votre fichier. Lorsque cette commande est utilisée, vous serez invité à modifier ou vérifier le nom de fichier souhaité et après avoir appuyé sur Entrée, elle enregistrera votre fichier.
----------	---

CTRL + W	Une des commandes les plus utiles. Elle est utilisée pour rechercher une expression spécifique dans votre texte. Elle fonctionne comme la commande CTRL + F sur d'autres applications. Pour rechercher la même expression, appuyez sur ALT + W autant de fois qu'il le faut.
----------	--

L'éditeur nano

Commandes de base de l'éditeur de texte Nano

COMMANDE	EXPLICATION
----------	-------------

CTRL + K	Coupe toute la ligne sélectionnée dans le “presse-papier”.
CTRL + U	Colle le texte du “presse-papier” dans la ligne sélectionnée.
CTRL + J	Justifie le paragraphe actuel.
CTRL + C	Affiche la position actuelle du curseur (ligne / colonne / caractère).
CTRL + X	Quitte l'éditeur de texte Nano. Dans le cas où vous avez apporté des modifications au fichier, elle vous invite à sauvegarder.
CTRL + R	Insérer un fichier dans le fichier actuel
CTRL + \	Se déplace vers le haut du presse-papier
CTRL + T	Exécute une commande
ALT + G	Va à une ligne spécifique
ALT + A	Activer/désactiver la marque. Vous pouvez combiner cette commande avec CTRL + K pour couper une partie spécifique du texte.

Le système de fichier Linux

Le système de fichier Linux

Généralités

Un système de fichier permet la gestion de données sur un périphérique regroupés dans des répertoires organisés de façon hiérarchique : c'est **l'arborescence des répertoires**.

C'est en fait une vue de l'esprit. Un disque dur est constitué de pistes et de secteurs.

Un fichier est stocké sur un ou plusieurs secteurs d'une ou plusieurs pistes.

L'astuce du "file system" est de présenter cette réalité d'une autre façon en introduisant un ensemble d'informations relatives à ce fichier (nom, position, sa date de création, ses attributs ...)

Le système de fichier Linux

Système de fichier (Ext3Fs)

Le file system de DOS (fat ou vfat) permet de donner un nom sur 8 caractères.

Sous DOS et sur les premières versions de Windows 95 la FAT16 est de rigueur. A partir de Windows 95 OSR2 vous avez le choix entre les systèmes de fichiers FAT16 et FAT32.

Le file system de Linux s'appelle ext3fs (extended 3 FS) qui est un pseudo système de fichier qui vient en surcouche du système réel de sorte qu'avec un seul type de commande fonctionnera sur l'ensemble des filesystem de votre machine.

Vous pourrez ainsi lire et écrire des fichiers linux, DOS car il est possible de travailler sur des partitions DOS à partir de Linux.

Le système de fichier Linux

Système de fichier (Ext3Fs) (suite)

Ext3Fs accepte les types de fichiers suivants :

- Les répertoires : ensemble de fichiers

- Les fichiers ordinaires : programmes, données ...

- Les fichiers spéciaux : les périphériques, les blocs

Ext3Fs accepte des noms longs (265 caractères composés des caractères suivants :

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

1234567890

On peut user et abuser du point ". Exemple .bashrc.20190715.sav.old.

Eviter le -?&<>*\

Le système de fichier Linux

Système de fichier (Ext3Fs) (suite)

```
total 4
  2 0 drwxr-xr-x 21 root root      4300 juin 28 19:01 .
  2 4 drwxr-xr-x 23 root root      4096 juin 21 15:25 ..
400 0 crw-r--r--  1 root root    10, 235 juin 28 19:00 autofs
217 0 drwxr-xr-x  2 root root      340 juin 28 19:00 block
343 0 drwxr-xr-x  2 root root       80 juin 28 18:59 bsg
417 0 crw-----  1 root root    10, 234 juin 28 19:00 btrfs-control
137 0 drwxr-xr-x  3 root root       60 juin 28 18:59 bus
433 0 lrwxrwxrwx  1 root root         3 juin 28 19:00 cdrom -> sr0
434 0 lrwxrwxrwx  1 root root         3 juin 28 19:00 cdrw -> sr0
171 0 drwxr-xr-x  2 root root     4000 juin 28 19:01 char
 14 0 crw-----  1 root root       5,  1 juin 28 19:01 console
166 0 lrwxrwxrwx  1 root root        11 juin 28 18:59 core -> /proc/kcore
159 0 drwxr-xr-x  2 root root       60 juin 28 18:59 cpu
161 0 crw-----  1 root root    10,  59 juin 28 19:00 cpu_dma_latency
416 0 crw-----  1 root root    10, 203 juin 28 19:00 cuse
348 0 drwxr-xr-x  6 root root      120 juin 28 18:59 disk
150 16 -rw-rw----  1 michel michel 14865 juin 28 19:00 liste.txt
 86 0 drwxr-xr-x  3 root root      100 juin 28 19:00 dri
435 0 lrwxrwxrwx  1 root root         3 juin 28 19:00 dvd -> sr0
```

Le système de fichier Linux

Système de fichier (Ext3Fs) (suite)

Sur l'écran précédent nous pouvons voir :

- l'inode (numéro qui désigne le fichier (il est unique)
- le type de fichier :
 - pour les fichiers ordinaires;
 - d pour les répertoires (directory);
 - b pour les fichiers spéciaux en mode bloc;
 - c pour les fichiers spéciaux en mode caractère;
 - l pour les liens symboliques;
 - p pour les fifo (pipes) (First In, First Out);
 - s pour les socket (permet la communication réseau entre deux processus locaux ou distants);
- les permissions d'accès;
- le nombre de liens physiques;
- le nom du propriétaire et du groupe;
- la taille en octets;
- l'horodatage (date de dernière modification).

Le système de fichier Linux

Droit et permission

Les droits sont gérés par les commandes suivantes :

chmod	change mode	ex : chmod 777 myfile.txt
chown	change owner	ex : chown tarzan myfile.txt
chgrp	change groupe	ex : chgrp jungle myfile.txt

En découle les permissions

```
15470 16 -rw-rw---- 1 michel michel 14865 juin 28 19:00 liste.txt
```

Concerne les droits d'accès du propriétaire. Ce dernier possède un accès en lecture (r pour read) et écriture (w pour write). Le fichier ne peut donc pas être exécuté sauf si la lettre x avait été présente.

Le système de fichier Linux

Droit et permission (suite)

```
15470 16 -rw-r----- 1 michel michel 14865 juin 28 19:00 liste.txt
```

Concerne les droits d'accès du groupe. Ici r indique que le fichier est accessible uniquement en lecture pour le groupe auquel appartient le fichier.

```
15470 16 -rw-r----- 1 michel michel 14865 juin 28 19:00 liste.txt
```

Concerne les droits d'accès des autres (n'appartenant pas au groupe). Ici aucun droit

Le système de fichier Linux

Droit et permission (suite)

Pour affecter une permission → `chmod` (octal)

0	---	Aucune permission
1	--x	Exécution uniquement
2	-w-	Ecriture uniquement
3	-wx	Ecriture Execution
4	r--	Lecture uniquement
5	r_x	Lecture Exécution
6	rw-	Lecture Ecriture
7	rwX	La totale Lecture Ecriture Exécution.

Le système de fichier Linux

Les liens

Il s'agit en fait d'une redirection d'un nom fictif vers un fichier réel. Toute l'astuce consiste à faire correspondre à un nom de fichier l'inode du fichier pointé.

En découle les permissions

`ln nom sosie` (2 fichiers)

`ln -s nom image`

Voir avec plus tard les répercussion avec la commande `mount`

Le système de fichier Linux

La commande find

Cette commande permet de rechercher un ou plusieurs fichiers, selon des critères. Il est même possible de lancer des actions sur le résultat (effacement, copie, etc).

La syntaxe est la suivante : `find [chemin] [critères]`

Exemple `find /home -name liste.txt`

Gestion des utilisateurs

Gestion des utilisateurs

/etc/passwd

Un utilisateur est défini par son ID (un numéro). Root a le numéro 0. Souvent, les simples utilisateurs ont un numéro supérieur à 100. L'utilisateur a un groupe par défaut. Un groupe est défini par son numéro (le GID). Il peut regrouper plusieurs users. Le système s'arrête là. Vous retrouvez ces notions dans la gestion des fichiers (user, group et other)

Les champs de ce fichier sont les suivant :

login

mot de passe crypté (ou x si shadow actifs):

IDentifiant

Group IDentifiant:

Nom en clair (Gecos)

Home directory: (programme lancé à la connexion)

```
root:x:0:0:root:/root:/bin/bash bin:x:1:1:bin:/bin:
```

```
lists:x:500:500:BeroList:/dev/null:/dev/null
```

```
gdm:x:42:42:./home/gdm:/bin/bash
```

```
michel:x:42:42:/home/michel:/bin/bash
```

Gestion des utilisateurs

/etc/group

Ce fichier contient la liste des groupes. On y trouve son nom, suivi du mot de passe du groupe (très peu utilisé), puis le Groupe Identificateur (qui est repris dans /etc/passwd), et enfin la liste des users appartenant au groupe...

```
root::0:root bin::1:root,bin,daemon
bin::1:root,bin,daemon
daemon::2:root,bin,daemon sys::3:root,bin,adm adm
disk::6:root lp::7:daemon,lp mem::8:
news::13:news uucp::14:uucp man::15: games::20: gopher::30: dip::40:
ftp::50: nobody::99: users::100: utmp:x:101: lists:x:500:
floppy:x:19: console:x:11: popusers:x:231: slipusers:x:232:
slocate:x:21: sylvain::501: audio:x:60: jeux:x:1002:
```

Gestion des utilisateurs

/etc/shadow

Fichier privilégié, qui contient les mots de passe cryptés, et qui n'est accessible qu'à root

```
root:$1$npqvHBzG$okPjWk/:10853:0:99999:7:::134542424 bin:*:10853:0:99999:7:::
daemon:*:10853:0:99999:7::: adm:*:10853:0:99999:7:::
lp:*:10853:0:99999:7:::
sync:*:10853:0:99999:7:::
shutdown:*:10853:0:99999:7::: halt:*:10853:0:99999:7:::
mail:*:10853:0:99999:7:::
news:*:10853:0:99999:7:::
uucp:*:10853:0:99999:7:::
```

Gestion des utilisateurs

Commande

adduser & addgroup : les commandes adduser et addgroup permettent d'ajouter des utilisateurs ou des groupes au système en fonction des options fournies en ligne de commande et des informations contenues dans le fichier de configuration /etc/adduser.conf

useradd : commande pour créer un nouvel utilisateur ou modifier les informations par défaut appliquées aux nouveaux utilisateurs

passwd : pour changer le password d'un utilisateur (doit être en root) ex : passwd michel. Si passwd seul, change le passwd courant sur la console.

Les processus

Les processus

Linux est un système d'exploitation multitâche multiutilisateur. Linux utilise plusieurs processus, qui s'enchaînent les uns les autres, ou sont en concurrence. Le démarrage de la machine utilise ce système de processus qui lance d'autres processus (principe du bootstrap).

Un processus est l'image d'un programme en mémoire. Le programme est un fichier binaire, c'est à dire des commandes compréhensibles par le processeur

Chaque processus a un numéro unique, un lien vers son père (celui qui l'a crée...), un numéro utilisateur et de groupe, la durée de traitement, sa priorité d'exécution, la référence à son répertoire de travail....

Les processus

Le premier processus qui est lancé est init, qui lance ensuite les suivants.... Votre bash est le fils de votre login, lui même fils d'un getty sur votre terminal (attente d'une connexion), etc....

Lorsque vous tapez une commande, celle-ci lance un nouveau processus, dont votre bash est le père.

La création des processus est régie par le principe suivant. un processus crée un processus fils, en forkant (appel système de fork (fourche), qui crée un double du processus père, en lui donnant un nouveau PID).

Les processus

Etat d'un processus

- R pour runnable
- S pour Sleeping
- D attente Entrée Sortie
- Z pour Zombie (plus de père)

Les processus

Tâche de fond

De par ce principe de multi tâches, il est possible de lancer une tâche en arrière plan, et de continuer à travailler sur le terminal. Il suffit de taper un & en fin de commande. Le numéro du processus fils s'affichera, et l'appel système est supprimé. On peut donc continuer à travailler. Attention cependant, les messages de sortie de la commande en arrière plan arriveront sur votre terminal...

On appelle parfois ce mode l'exécution de processus asynchrone (plus de dépendance de temps entre le père et le fils).

Attention : Le fait de vous déloguer arrêtera tous vos processus, dont ceux en tâches de fond...

Voir fg et bg foreground et background

Les processus

Les Processus différés

- at** permet le lancement de tâches à une heure précise. il accepte de multiples façon d'indiquer l'heure (par rapport à maintenant, à midi, à minuit, aujourd'hui, demain, ou une date et une heure précise) : now, noon, midnight, today et tomorrow suivi de + ou - quelque chose comme hour, minute, ou bien directement 18:00 May 15, 2000)
- cron** permet quand à lui la programmation de taches à intervalles réguliers. La tache demandée sera lancée automatiquement, sous l'identité d'un utilisateur, même si celui ci n'est pas connecté...

Le fichier est constitué de la façon suivante :

Min, heure, jour, mois, jourSemaine, commande

Les processus

Les Processus différés (suite)

la création et la gestion de ce fichier se fait par la commande suivante :
crontab -e (pour éditer) ou -l pour l'afficher...

Exemple de fichier crontab (en fait, il s'agit d'un fichier portant le nom du user, rangé dans /var/spool/cron)

00	08	*	*	6	updatedb
30	14	01	01	*	shutdown -r now

à 8 heures, tous les samedi (6 eme jour de la semaine), exécution de updatedb (base de données des noms de fichiers, consultable ensuite par locate.

le 1er janvier de chaque année, à 14H30, redémarrage de la machine.

Les processus

Commandes

ps	affiche les informations sur les processus (option axl)
kill	envoie un signal au processus (permet aussi d'arrêter la tâche, selon le signal)
top	affiche à intervalle régulier l'ensemble des tâches
pstree	affiche une arborescence des tâches time permet de mesurer les temps d'exécution jobs affiche les tâches en fond.
nice	change la priorité d'une tâche
renice	permet la gestion de la priorité (même après son lancement)
sleep	'endort' une tâche (la suspend)
nohup	évite à une tâche de fond d'être tuée par l'arrêt de son père
bg et fg	permet d'envoyer en fond ou de récupérer une tâche de fond en premier plan
wait	met le processus courant en état d'attente de ses fils asynchrones

Les services / daemons

Les services / daemons

Un service (parfois appelé daemon) est un programme qui s'exécute en arrière-plan, plutôt contrôlé par le système d'exploitation que par l'utilisateur directement.

Chaque programme qui s'exécute prend la forme d'un processus. Vous pouvez utiliser la commande `ps` pour voir les processus qui tournent sur votre système.

L'option `-o` précise les champs que vous souhaitez voir apparaître :

```
michel@Debian:~$ ps -o ppid,pid,tt,cmd
  PPID      PID TT      CMD
  1890      1895 pts/0    bash
  1895      2101 pts/0    ps -o ppid,pid,tt,cmd
michel@Debian:~$
```

Vous voyez alors que le processus `ps` que vous lancez manuellement :

- est attaché à votre terminal `pts/0` ;
- a pour parent votre shell de connexion `bash` (PID 1895 = PPID de `ps`) ;
- ne mourra que s'il a terminé son exécution ou si son parent meurt.

Les services / daemons

Parmi tous les processus de votre système, certains :

- ne sont associés à aucun terminal et tournent en “tâche de fond” indépendamment des utilisateurs qui se connectent ;
- ont pour parent le processus de PID 0 qui ne meurt qu'à l'arrêt du système
- s'exécutent “à durée indéterminée” et attendent qu'on leur donne du travail.

Dans le monde Unix/Linux, on appelle ces processus des **Daemons**, qu'on traduit parfois par démons, en français. Dans le monde Windows, on parle de **Services**.

La plupart de ces daemons sont lancés au démarrage du système.

Les services / daemons

Liste des services avec systemctl

```
systemctl list-unit-files --type=service
```

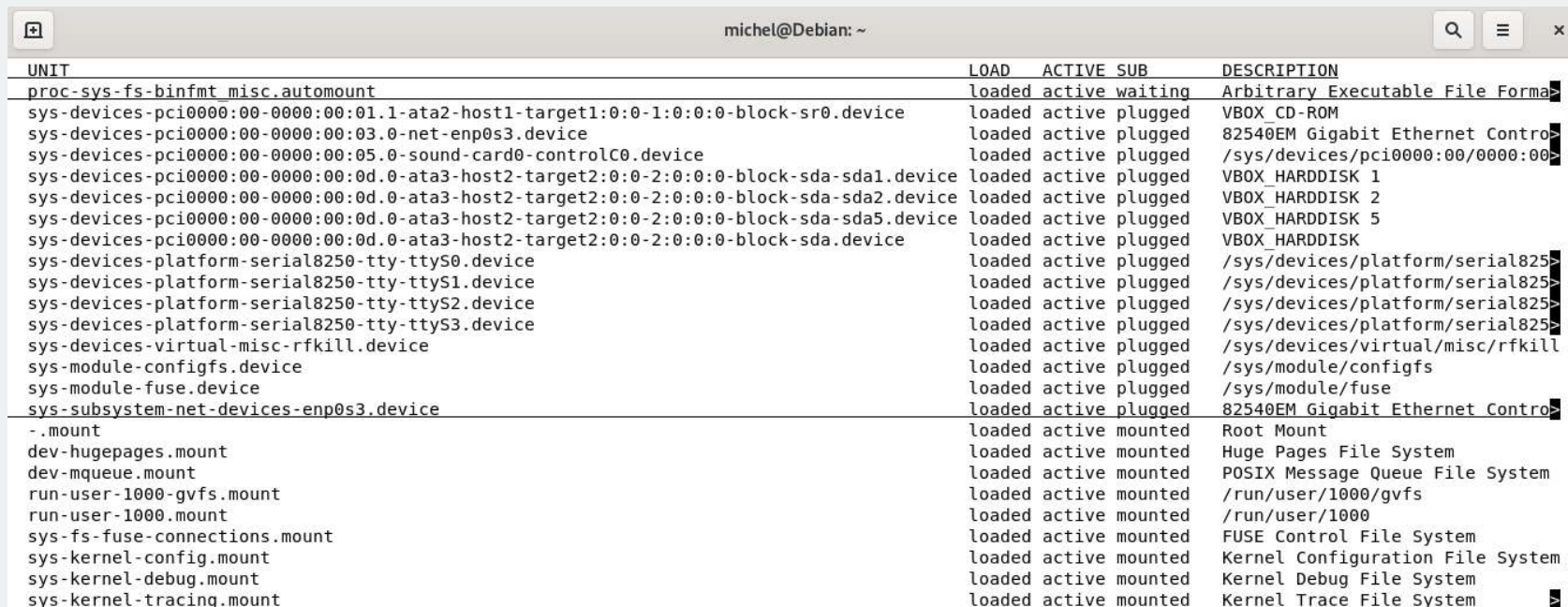


UNIT FILE	STATE	VENDOR PRESET
accounts-daemon.service	enabled	enabled
alsa-restore.service	static	-
alsa-state.service	static	-
alsa-utils.service	masked	enabled
anacron.service	enabled	enabled
apache-htcacheclean.service	disabled	enabled
apache-htcacheclean@.service	disabled	enabled
apache2.service	enabled	enabled
apache2@.service	disabled	enabled
apparmor.service	enabled	enabled
apt-daily-upgrade.service	static	-
apt-daily.service	static	-
autovt@.service	alias	-
avahi-daemon.service	enabled	enabled
bluetooth.service	enabled	enabled
bolt.service	static	-

Les services / daemons

Liste des services avec systemctl (suite)

systemctl list-units



UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
proc-sys-fs-binfmt_misc.automount	loaded	active	waiting	Arbitrary Executable File Forma
sys-devices-pci0000:00-0000:00:01.1-ata2-host1-target1:0:0-1:0:0:0-block-sr0.device	loaded	active	plugged	VBOX_CD-ROM
sys-devices-pci0000:00-0000:00:03.0-net-enp0s3.device	loaded	active	plugged	82540EM Gigabit Ethernet Contro
sys-devices-pci0000:00-0000:00:05.0-sound-card0-controlC0.device	loaded	active	plugged	/sys/devices/pci0000:00/0000:00
sys-devices-pci0000:00-0000:00:0d.0-ata3-host2-target2:0:0-2:0:0:0-block-sda-sda1.device	loaded	active	plugged	VBOX_HARDDISK 1
sys-devices-pci0000:00-0000:00:0d.0-ata3-host2-target2:0:0-2:0:0:0-block-sda-sda2.device	loaded	active	plugged	VBOX_HARDDISK 2
sys-devices-pci0000:00-0000:00:0d.0-ata3-host2-target2:0:0-2:0:0:0-block-sda-sda5.device	loaded	active	plugged	VBOX_HARDDISK 5
sys-devices-pci0000:00-0000:00:0d.0-ata3-host2-target2:0:0-2:0:0:0-block-sda.device	loaded	active	plugged	VBOX_HARDDISK
sys-devices-platform-serial8250-tty-ttyS0.device	loaded	active	plugged	/sys/devices/platform/serial825
sys-devices-platform-serial8250-tty-ttyS1.device	loaded	active	plugged	/sys/devices/platform/serial825
sys-devices-platform-serial8250-tty-ttyS2.device	loaded	active	plugged	/sys/devices/platform/serial825
sys-devices-platform-serial8250-tty-ttyS3.device	loaded	active	plugged	/sys/devices/platform/serial825
sys-devices-virtual-misc-rfkill.device	loaded	active	plugged	/sys/devices/virtual/misc/rfkill
sys-module-configfs.device	loaded	active	plugged	/sys/module/configfs
sys-module-fuse.device	loaded	active	plugged	/sys/module/fuse
sys-subsystem-net-devices-enp0s3.device	loaded	active	plugged	82540EM Gigabit Ethernet Contro
-.mount	loaded	active	mounted	Root Mount
dev-hugepages.mount	loaded	active	mounted	Huge Pages File System
dev-mqueue.mount	loaded	active	mounted	POSIX Message Queue File System
run-user-1000-gvfs.mount	loaded	active	mounted	/run/user/1000/gvfs
run-user-1000.mount	loaded	active	mounted	/run/user/1000
sys-fs-fuse-connections.mount	loaded	active	mounted	FUSE Control File System
sys-kernel-config.mount	loaded	active	mounted	Kernel Configuration File System
sys-kernel-debug.mount	loaded	active	mounted	Kernel Debug File System
sys-kernel-tracing.mount	loaded	active	mounted	Kernel Trace File System

Les services / daemons

systemctl

On peut chercher un service avec un wildcard.

Par exemple pour lister l'état des services qui commencent par ss, on peut taper

```
systemctl status ss*
```

Vous pouvez afficher l'aide de systemctl avec la commande

```
man systemctl
```

Les services / daemons

systemctl (suite)

Commande

`systemctl list-units`

`systemctl status [nom du service].service`

`systemctl enable [nom du service].service`

`systemctl disable [nom du service].service`

`systemctl stop [nom du service].service`

`systemctl start [nom du service].service`

`systemctl restart [nom du service].service`

`systemctl reload [nom du service].service`

`systemctl daemon-reload`

Fonction

Lister les services

Connaitre l'état d'un service

Activer un service

Désactiver un service

Arrêter un service

Démarrer un service

Redémarrer un service

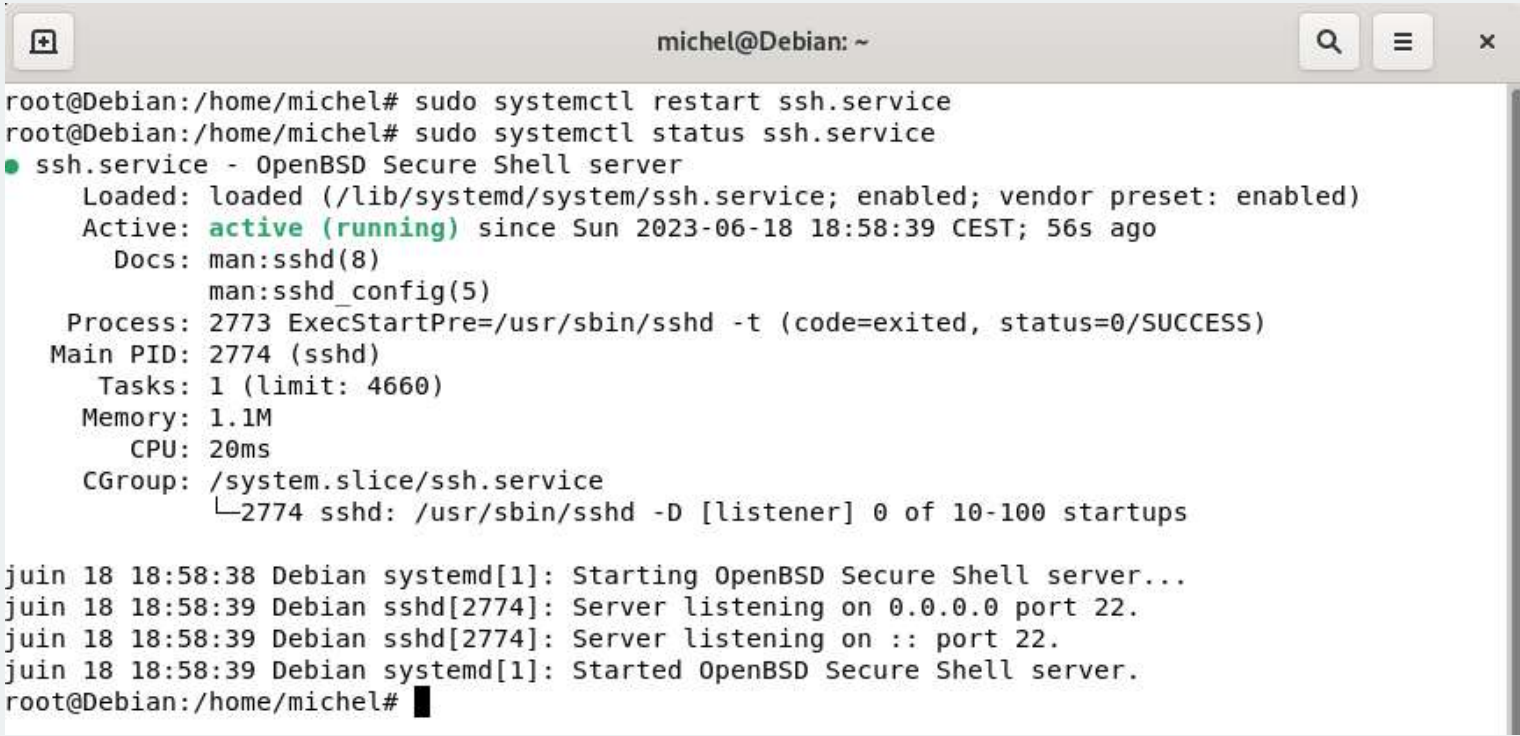
Recharge du service avec les nouveaux paramètres (évite les coupures)

Pour prendre en compte la modification d'un service

Les services / daemons

Systemctl (suite)

Exemple de redémarrage d'un service avec systemctl :



```
michel@Debian: ~
root@Debian:/home/michel# sudo systemctl restart ssh.service
root@Debian:/home/michel# sudo systemctl status ssh.service
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2023-06-18 18:58:39 CEST; 56s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 2773 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 2774 (sshd)
    Tasks: 1 (limit: 4660)
  Memory: 1.1M
     CPU: 20ms
   CGroup: /system.slice/ssh.service
           └─2774 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

juin 18 18:58:38 Debian systemd[1]: Starting OpenBSD Secure Shell server...
juin 18 18:58:39 Debian sshd[2774]: Server listening on 0.0.0.0 port 22.
juin 18 18:58:39 Debian sshd[2774]: Server listening on :: port 22.
juin 18 18:58:39 Debian systemd[1]: Started OpenBSD Secure Shell server.
root@Debian:/home/michel#
```

Les services / daemons

Systemctl (suite)

les fichiers de configuration des services sont dans :

`/lib/systemd/system` ou

`/etc/systemd/system/`

nous pouvons donc les voir en effectuant :

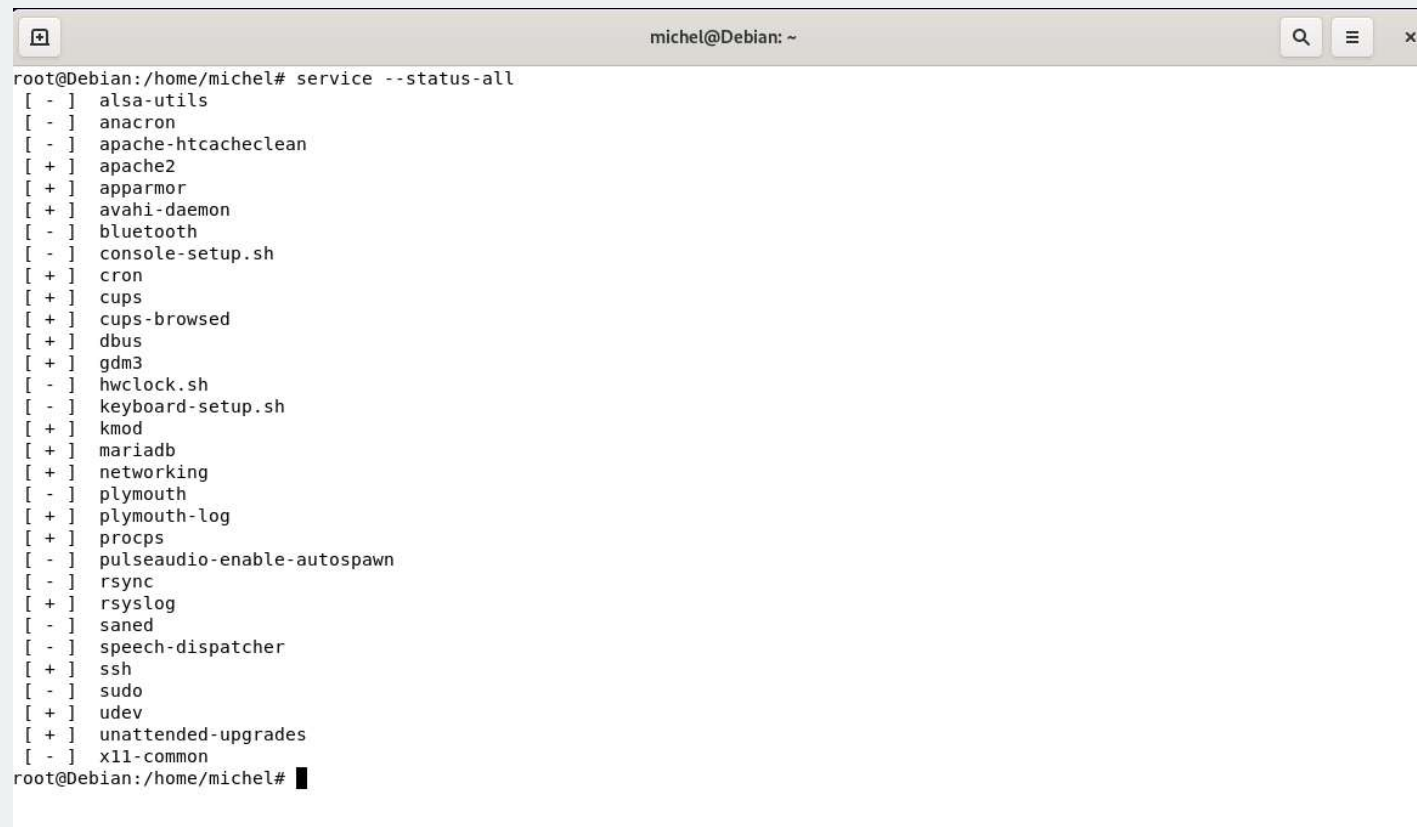
`ls -ltr /etc/systemd/system/`

`ls -ltr /usr/lib/systemd/system/`

`ls -ltr /lib/systemd/system`

Les services / daemons

Systemctl (suite)



```
michel@Debian: ~  
root@Debian:/home/michel# service --status-all  
[ - ] alsa-utils  
[ - ] anacron  
[ - ] apache-htcacheclean  
[ + ] apache2  
[ + ] apparmor  
[ + ] avahi-daemon  
[ - ] bluetooth  
[ - ] console-setup.sh  
[ + ] cron  
[ + ] cups  
[ + ] cups-browsed  
[ + ] dbus  
[ + ] gdm3  
[ - ] hwclock.sh  
[ - ] keyboard-setup.sh  
[ + ] kmod  
[ + ] mariadb  
[ + ] networking  
[ - ] plymouth  
[ + ] plymouth-log  
[ + ] procps  
[ - ] pulseaudio-enable-autospawn  
[ - ] rsync  
[ + ] rsyslog  
[ - ] saned  
[ - ] speech-dispatcher  
[ + ] ssh  
[ - ] sudo  
[ + ] udev  
[ + ] unattended-upgrades  
[ - ] x11-common  
root@Debian:/home/michel#
```

Les Files System

Les Files System

Le formatage de bas niveau

fdformat effectue un formatage de bas niveau sur une disquette.

Le formatage de haut niveau

mkfs (Makes File System) est la commande à utiliser en ligne de commandes depuis un terminal pour créer et formater un système de fichiers.

Une fois le système formaté, on y dépose un file system

Le montage

mount La commande mount permet de demander au système d'exploitation de rendre un système de fichiers accessible, à un emplacement spécifié (le point de montage).

mount [-t FS] device répertoire

mount -t msdos /dev/fd0 /mnt/dos

umount /dev/fd0

/etc/fstab (stocke tous les points de montage qui sont exécutés au démarrage)

La commande mount permet de demander au système d'exploitation de rendre un système de fichiers accessible, à un emplacement spécifié (le point de montage)

Les Files System

swap

La partition de swap est utilisée par Linux pour stocker temporairement des données lorsque la mémoire RAM est saturée. On peut utiliser soit un fichier de swap (comme sous Windows), ou plutôt ce système qui est plus propre.

La partition de swap peut être une partition étendue. Sa taille ne peut pas dépasser 128 Mo (mais il est possible d'avoir autant de partitions swap que l'on veut), et est comprise habituellement entre 1 à 2 fois la taille de la RAM.

Sauvegarde et Package

Sauvegarde et Package

Commandes

dd permet la création de copies physiques. (travail de bas niveau)

tar est une commande plus courante, raccourci de Tape Archiver.

tar permet la création de sauvegarde d'une liste de fichier soit dans un fichier archive, soit sur un support (/dev/fd0). Un de ses avantages est sa récursivité. (sous répertoires).

Options de tar.

c	Créer une archive
t	Test une archive
x	Extrait le contenu de l'archive
v	Mode verbeux
f	l'argument suivant sera le nom du fichier archive (régulier, ou spécial...)
z	l'ensemble est compressé (par gzip)
M	Sauvegarde en multi volume (pratique si l'archive est plus grande que le support amovible)

tar cvf /dev/fd0 /home.

Sauvegarde et Package

Commandes (suite)

cpio cette commande sert aussi pour la gestion des données. L'option -o crée des sauvegardes, tandis que -i restaure les archives...

cpio est plus efficace que tar en cas de pépin sur la sauvegarde (disquette problématique, pb de bandes...).

En mode création de sauvegarde, cpio attend une liste de fichiers sur son entrée standard. On utilise par exemple des pipes pour l'alimenter. Le fichier archive sort sur sa sortie standard... A vous de le rediriger, par exemple dans un fichier.

En mode restauration, cpio attend un fichier archive sur son entrée standard, et restaure dans le répertoire courant.

Exemple :

création d'une archive.tgz

```
ls | cpio -o > /tmp/toto.cpio
```

la liste des fichiers du répertoire courant est donnée à cpio, qui crée alors une archive /tmp/toto.cpio

Sauvegarde et Package

Commandes (suite)

dpkg (debian)

`dpkg -i package.deb` lance l'installation de ce package, en rangeant tout au bon endroit et en mettant à jour la base de données de votre machine...

`dpkg -r package` le désinstalle

`dpkg -i -R répertoire` lance l'installation de tous les packages contenus dans le répertoire

Sauvegarde et Package

Commandes (suite)

apt-get est un frontal très simple qui permet la maintenance et la mise à jour de votre système. Ce programme interroge un fichier qui s'appelle `sources.list`, qui lui indique où il trouvera les données d'installation standard (en général, les CD, ou un serveur dans votre réseau, voire même les serveurs Debian !!!). Il interrogera ensuite la base de données des installations et vous demandera d'introduire les Cds correspondant, ou se connectera au serveur voulu.

apt-get install package	lance l'installation de ce package, après vérification des dépendances, et de la pertinence de cet installation (numéro de version).
apt-get remove package	l'enlève.

Sauvegarde et Package

Commandes (suite)

apt-get update met à jour l'installation

apt-get dist-upgrade se connecte sur les serveurs, et met à jour tous les packages à partir des serveurs Debian...

apt-get source « package » récupère les sources du package voulu...
Attention, apt-get "rouspète" si vous avez fait des installations "sauvages"...

Aptitude est peut être plus intéressant, car il regroupe toutes les commandes (contrairement à apt-get, search fait partie des commandes d'aptitude). Il gère aussi mieux les problèmes de dépendance. Enfin, en mode graphique, de nombreux outils permettent la gestion des paquets (synaptic sous gnome, kadept sous kde, etc...)

Les scripts SHELL

Les scripts SHELL

Le Shell est un interpréteur de commandes. Lorsque vous tapez une commande, il effectue différentes opérations et transfère la commande interprétée au noyau du système d'exploitation pour exécution.

Dans le cas où la commande est simple, il suffit de la taper directement après le prompt (ie : en ligne de commande). Mais parfois une commande nécessite plusieurs arguments, des tests et des redirections, auquel cas il est préférable d'écrire un programme.

Ce programme est en fait un simple fichier texte qui contient une suite de commande Linux. Il doit être exécutable (bit x activé). Utilisez nano pour parvenir à vos fins.

Les scripts SHELL

Les jokers

* remplacement d'une chaîne de 0 à n caractères

`rm *.deb` supprime tous les fichiers avec extension `.deb`

? remplacement strict d'un caractère

`rm *.d?b` supprime tous les fichiers avec extension `.d?b` (dib, dab, dob, dcb, d1b)

[xyz] Définit un ensemble

`ls ab[123]` → ab1 ab2 ab3

[x-y] Ensemble de x à y

`ls gr[5-7]` → gr5 gr6 gr7

[!x-y] Tout sauf l'ensemble de x à y

Les scripts SHELL

Le quoting

Protection face à l'interpréteur du shell

```
ls *      puis echo *      puis echo \*
```

Les scripts SHELL

Procédures à suivre

Il est bon (mais pas indispensable) de mettre l'instruction suivante en première ligne du programme :

```
#!/bin/bash
```

Cela indique à l'interpréteur que ce qui va suivre est du Shell (en l'occurrence du Bourne Again Shell - bash).

Le fichier contenant le code doit être rendu exécutable avant de lancer le programme. Ou alors, il faut le lancer en argument d'un shell (\$ bash ./prog1)

Pour lancer le programme, il faut taper la ligne suivante :

\$ prog si votre variable PATH intègre le chemin du répertoire dans lequel vous lancez le programme

\$./ prog Sinon...

\$ sh ./prog Au cas où le fichier ne soit pas exécutable....

Les scripts SHELL

Les arguments

Comme tous programmes, les scripts acceptent des arguments. Pour récupérer et traiter le contenu de ces arguments dans vos scripts, vous utiliserez le signe \$

\$1	contient le premier argument,
\$2	le second, et ainsi de suite, jusqu'à 9.
\$* ou \$@	indique tous les arguments...
\$0	indique le programme lui même...
\$#	renvoie le nombre d'arguments passés à la commande.

Les scripts SHELL

Les variables

Dans vos programmes, vous aurez parfois besoin de stocker des valeurs. Vous pourrez alors créer des variables (non typés). Une variable a un nom. L'affectation se fait en utilisant l'opérateur = sans espace autour. L'utilisation du contenu de la variable se fait en ajoutant \$ devant.

```
MOT=chat
```

```
echo le pluriel de $MOT est ${MOT}s
```

```
réponse : le pluriel de chat est chats
```

Si vous voulez affecter une chaîne de plusieurs mots à une variable, protégez la avec des guillemets.

```
NOM="Cherrier Sylvain Claude Philippe"
```

```
echo $NOM
```

```
réponse : Cherrier Sylvain Claude Philippe
```

Les scripts SHELL

Les commandes

export : La commande export est une commande bash intégrée utilisée pour rendre les variables disponibles pour les processus enfants du shell actuel. Une fois que vous avez exporté une variable dans un shell, tout processus exécuté à partir de ce shell pourra accéder à cette variable.

La commande export utilise la syntaxe suivante.

```
export variable=value
```

La commande ci-dessus signifie que les processus lancés dans le shell accéderont à la variable.

Déclarer une variable sans la commande export signifie que la variable ne sera disponible que pour le shell et non pour les autres processus du shell.

Les scripts SHELL

Les commandes

echo : La commande echo est l'une des commandes les plus basiques et les plus fréquemment utilisée sous Linux. Les arguments passés à echo sont affichés sur la sortie standard.

echo [OPTIONS] [TEXTE]

OPTIONS	DESCRIPTION
-n	n'affiche pas la nouvelle ligne de fin
-e	permet l'interprétation des échappements antislash
-E	désactiver l'interprétation des échappements antislash (par défaut)

Les scripts SHELL

Les commandes

read : La commande read lit son entrée standard et affecte les valeurs saisies dans la ou les variables passées en argument.

```
read var1
```

```
coucou
```

```
echo $var1
```

```
coucou
```

Les scripts SHELL

Les commandes

test : La commande test permet de vérifier les attributs d'un fichier ou le contenu d'une variable. Elle est souvent utilisée a l'intérieur de scripts Shell (avec la structure de contrôle if).

Expression	Code de retour
------------	----------------

-b FILE	Vrai si le fichier existe et est du type spécial bloc
---------	---

-c FILE	Vrai si le fichier existe et est du type spécial caractère
---------	--

-d FILE	Vrai si le fichier existe et est du type répertoire
---------	---

-e FILE	Vrai si le fichier existe
---------	---------------------------

-f FILE	Vrai si le fichier existe et est du type ordinaire
---------	--

-G FILE	Vrai si le fichier existe et si l'utilisateur appartient au groupe propriétaire du fichier
---------	--

-h FILE	Vrai si le fichier existe et est du type lien symbolique
---------	--

-L FILE	Vrai si le fichier existe et est du type lien symbolique (idem -h)
---------	--

Les scripts SHELL

Les commandes

test : (suite)

Expression	Code de retour
-O FILE	Vrai si le fichier existe et si l'utilisateur est le propriétaire du fichier
-r FILE	Vrai si le fichier existe et est accessible en lecture
-s FILE	Vrai si le fichier existe et n'est pas vide
-S FILE	Vrai si le fichier existe et est du type socket
-w FILE	Vrai si le fichier existe et est accessible en écriture
-x FILE	Vrai si le fichier existe et est exécutable
FILE1 -ef FILE2	Vrai si les fichiers ont le même lien physique
FILE1 -nt FILE2	Vrai si FILE1 est plus récent que FILE2
FILE1 -ot FILE2	Vrai si FILE1 est plus ancien que FILE2

Les scripts SHELL

Les commandes

test : (suite)

Le fichier /etc/group est un fichier ordinaire

```
test -f /etc/group
```

```
echo $?
```

0

Le fichier /etc/groupe n'existe pas (test avec l'autre syntaxe)

```
[ -f /etc/groupe ]
```

```
echo $?
```

1

Les scripts SHELL

Les structures de contrôle

boucle for

Syntaxe : La boucle for permet de traiter une liste de valeurs indiquée à droite du mot clé in. A chaque tour de boucle, la variable var est initialisée avec une des valeurs de la liste. Elles sont traitées dans l'ordre de leur énumération.

Liste de valeur citée directement

```
for var in valeur1 valeur2 valeur3 ... valeur n
do
    commande
done
```

Les scripts SHELL

Les structures de contrôle

boucle for (suite)

Avec incrémentation d'une variable

```
for (( var=valeurMin; var<=valeurMax; var++ ))  
do  
    commande  
done
```

Voir le manuel pour toutes les options de la boucle for

Les scripts SHELL

Les structures de contrôle

If

La structure de contrôle if permet de réaliser des tests. La commande située à droite du if est exécutée.

Si le code retour de la commande (\$?) est égal à 0 (vrai), les commandes situées dans le bloc then sont exécutées.

Si le code de retour est supérieur à 0 (faux), ce sont les commandes situées dans le bloc else (optionnel) qui sont exécutées.

Dans le cas où le bloc else n'est pas spécifié, le shell continue à la première commande située sous le fi.

Les différentes syntaxes :

if, then, else, fi

Les scripts SHELL

Les structures de contrôle

If (suite)

```
if commande1
then
    commande2
    commande3
    ...
else
    commande4
    ...
fi
```


Les scripts SHELL

Les structures de contrôle

If (suite)

```
$ nl user_passwd.sh
```

```
1  #!/bin/bash
2  if [[ $# -ne 1 ]]
3  then
4      echo -e "Saisir le nom d'un user : \c"
5      read user
6  else
7      user=$1
8  fi
9  if grep -q "^$user:" /etc/passwd
10 then
11     echo "Le user $user existe"
12     echo "Son UID est le : $(grep "^$user:" /etc/passwd | cut -d":" -f3)"
13     echo "Son GID est le : $(grep "^$user:" /etc/passwd | cut -d":" -f4)"
14 else
15     echo "Le user $user n'existe pas !!!"
16 fi
17 exit 0
```

Les scripts SHELL

Les structures de contrôle

case

La structure de contrôle case permet elle aussi d'effectuer des tests. Elle permet d'orienter la suite du programme en fonction d'un choix de différentes valeurs. Quand il y a un nombre important de choix, la commande case est plus appropriée que la commande if.

```
case $variable in
  modele1) commande1
    ...
    ;;
  modele2) commande2
    ...
    ;;
  modele3 | modele4 | modele5 ) commande3
    ...
    ;;
esac
```

Les scripts SHELL

Les structures de contrôle

case (suite)

Le shell compare la valeur de la variable aux différents modèle renseignés.

Lorsque la valeur correspond au modèle, les commandes faisant partie du bloc sont exécutées.

Les caractères ;; permettent de fermer le bloc et de mettre fin au case.

Le shell continue à la première commande située sous esac.

Il ne faut surtout pas oublier les caractères ;; car cela engendrera une erreur.

Les scripts SHELL

Les structures de contrôle

boucle while

La boucle while permet d'exécuter les commandes présentes entre le do et le done tant que la commande1 placée à droite du while retourne un code vrai.

```
while commande1
do
    commande2
    ...
done
```

Les scripts SHELL

Les structures de contrôle

boucle until

A l'inverse de while, la commande until exécute les commandes situées entre le do et le done tant que la commande située à droite du until retourne un code faux.

```
until commande1
```

```
do
```

```
    commande2
```

```
    ...
```

```
done
```

Les scripts SHELL

Les commandes

exit

La commande exit est utilisée pour terminer un script, comme dans un programme C. Elle peut également renvoyer une valeur, qui sera disponible pour le processus parent du script.

Par convention le code retour par défaut est 0 et indique un succès alors qu'un code de sortie différent de zéro indique une erreur ou une condition anormale.

```
#!/bin/bash
```

```
echo bonjour
```

```
exit 113 # Retournera 113 au shell.
```

Pour vérifier ceci, tapez "echo \$?" une fois le script terminé.

Les scripts SHELL

Les commandes

break et continue

Les commandes break et continue peuvent s'utiliser à l'intérieur des boucles for, while, until et select.

La commande break permet de sortir d'une boucle.

La commande continue permet de remonter à la condition d'une boucle.

exec

La commande exec sous Linux est utilisée pour exécuter une commande à partir du bash lui-même. Cette commande ne crée pas de nouveau processus, elle remplace simplement le bash par la commande à exécuter. Si la commande exec réussit, elle ne revient pas au processus appelant.

Merci

Michel Blache
+33679674276
michel@blache.eu