CONNEXION ET PREMIERES COMMANDES

Introduction

Comment se connecter à un système GNU/Linux et de saisir ses premières commandes.

Pour cela, seront vus en détail les consoles disponibles sous Linux, le processus d’authentification et l’utilisation de la ligne de commande.

Consoles et terminaux Linux

Les termes "console" et "terminal" sont employés pour définir l’ensemble de périphériques permettant à l’utilisateur d’interagir avec le système ; cet ensemble est généralement composé d’un écran, d’un clavier et d’une souris.

1. Consoles virtuelles

Afin d’offrir plusieurs terminaux à l’utilisateur à partir du même ensemble écran/clavier/souris, Linux fournit un certain nombre de consoles virtuelles.

L’avantage de cette gestion est de disposer aussi bien de plusieurs terminaux "texte" pour les tâches devant être effectuées en ligne de commande, que d’une console "graphique" permettant de lancer des outils avec une interface plus évoluée comme un logiciel de retouche d’images ou un lecteur multimédia.

Ces consoles virtuelles, généralement au nombre de six ou sept sur les distributions comme Fedora, Ubuntu ou SUSE, sont composées de cinq à six terminaux texte et d’une console graphique. Elles sont représentées par les touches de fonction **[F1]** à **[F7]** du clavier (**[F1]** ou **[F7]** étant la console graphique) ; pour passer de l’une à l’autre, il faut appuyer simultanément sur les touches **[Ctrl]**+**[Alt]**+**[Fn]** où **n** est le numéro de la console virtuelle.

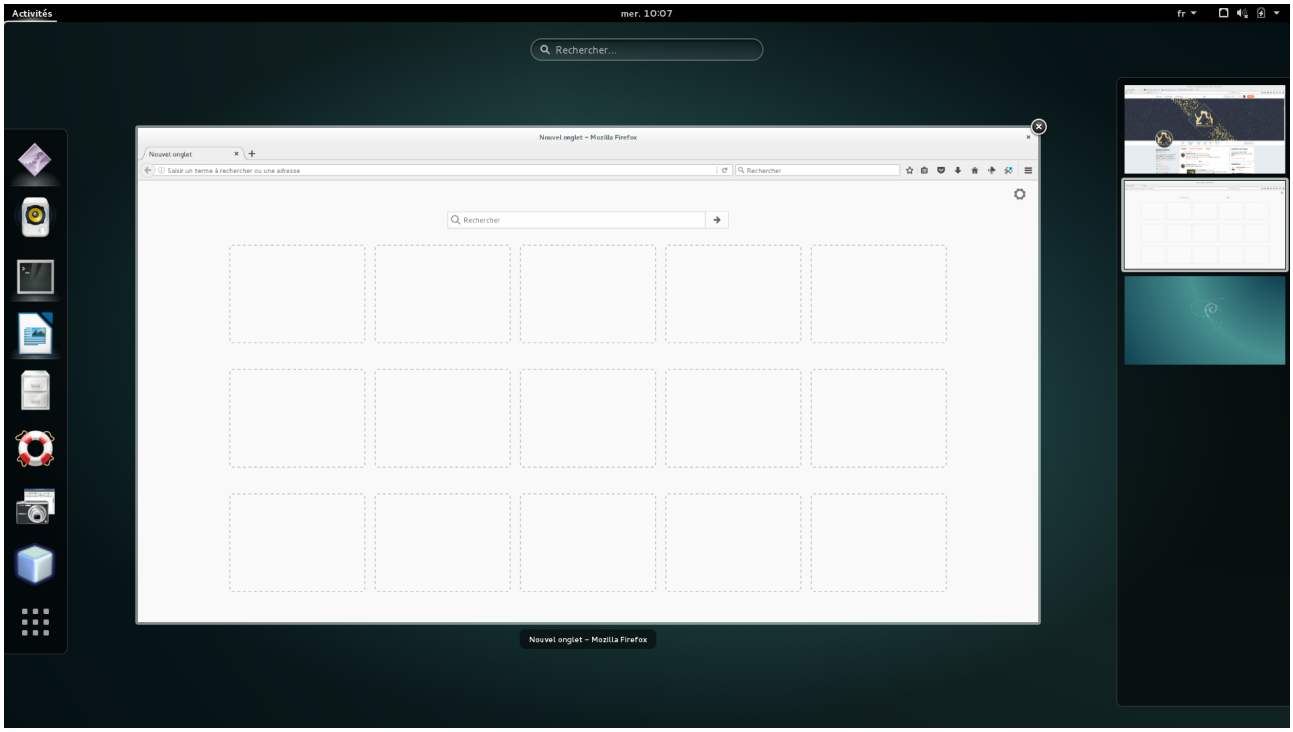
Il n’est pas obligatoire d’appuyer sur la touche **[Ctrl]** pour passer d’un terminal texte à l’autre.

Une console virtuelle texte, ou plus simplement un terminal texte, ressemble à ceci :

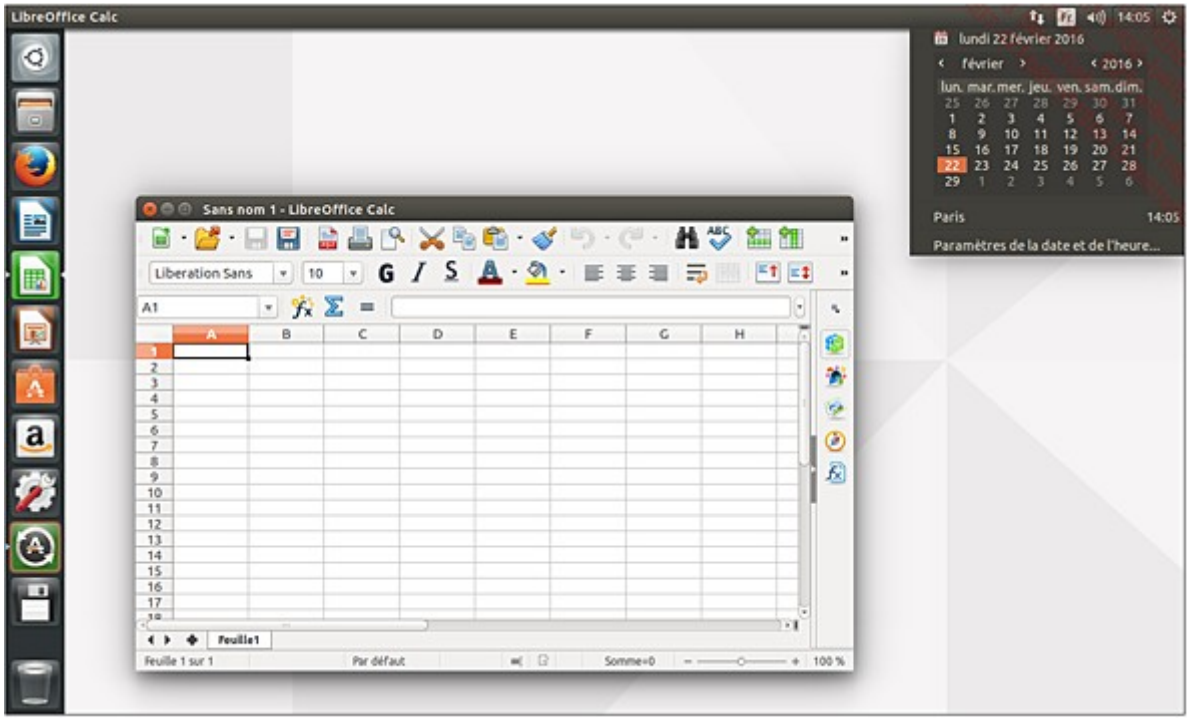
```
driss@mo:~$ cal
    Octobre 2017
di lu ma me je ve sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

driss@mo:~$ cd Documents/firefox
driss@mo:~/Documents/firefox$ ls
active-update.xml   dictionaries     icons                libmozsandbox.so
application.ini     firefox          icudt59l.dat         libmozsqlite3.so
browser             firefox-bin      libfreeblpriv3.chk   libnspr4.so
chrome.manifest     firefox-bin.sig  libfreeblpriv3.so    libnss3.so
crashreporter       firefox.sig      liblgpllibs.so       libnssckbi.so
crashreporter.ini   fonts            libmozavcodec.so     libnssdbm3.chk
defaults            gmp-clearkey     libmozavutil.so      libnssdbm3.so
dependentlibs.list  gtk2             libmozgtk.so         libnssutil3.so
driss@mo:~/Documents/firefox$
```

La console virtuelle graphique peut prendre plusieurs aspects, suivant l'environnement de bureau choisi et la version de la distribution Linux ; par exemple, sous Debian 8, avec l'environnement de bureau Gnome:



Et avec l'environnement GNOME sous Ubuntu 15.10 :

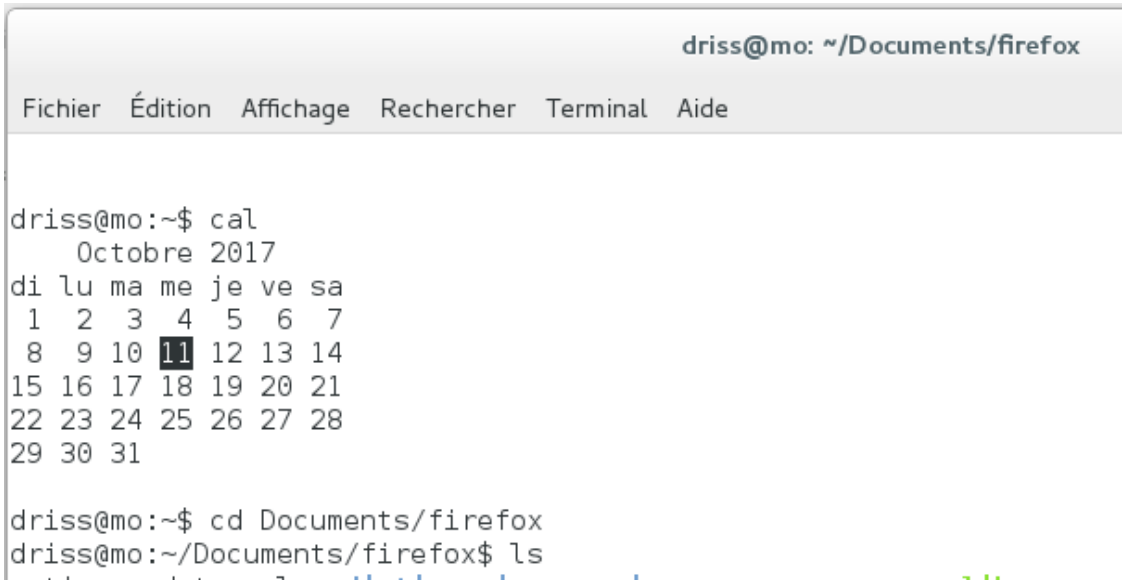


Il est possible d’avoir plus - ou moins - de six consoles virtuelles texte et plus d’une console virtuelle graphique sous Linux. Nous considérerons pour la suite que nous disposons de sept consoles virtuelles dont une graphique.

2. Émulateurs de terminaux

Sous l’interface graphique Linux, on peut saisir des commandes de la même manière que sur un terminal texte. Pour cela, on fait appel à un programme "émulant" la ligne de commande Linux à l’intérieur d’une fenêtre.

Il y a un grand nombre d’émulateurs de terminaux disponibles sous Linux, dont **xterm**, qui est un standard quel que soit le système UNIX utilisé, ou **konsole** qui est fourni avec l’environnement KDE :



L'interface graphique Linux propose généralement plusieurs icônes ou menus avec un pictogramme ressemblant à un écran ou à un coquillage qui permettent le lancement d'émulateurs de terminaux.

3. Terminaux distants

Une autre façon d'interagir avec la ligne de commande Linux est l'utilisation d'une connexion réseau et d'un logiciel de communication à distance tel que Telnet ou SSH, lancé à partir d'un terminal texte.

Il est alors possible de commander un système Linux à partir d'un autre Linux, ou même d'une machine Windows en utilisant un outil comme PuTTY sous licence GPL, donc librement téléchargeable.

Connexion et authentification

Le système Linux étant multi-utilisateur, il faut en premier lieu s'authentifier auprès de la machine.

L'authentification permet de vérifier que l'utilisateur présent devant la console est bien celui qu'il prétend être. Il doit fournir son identité (nom de connexion ou login) et une preuve de celle-ci (mot de passe) pour se connecter.

Qu'il s'agisse du mode texte ou du mode graphique, le mot de passe saisi par l'utilisateur n'est pas affiché.

```
driss@mo:~$ ssh [redacted] -p [redacted]
[redacted]'s password:
Linux [redacted] 3.14.32-xxxx-grs-ipv6-64 #7 SMP Wed Jan 27 18:05:09
 CET 2016 x86_64 GNU/Linux

server    : [redacted]
ip        : [redacted]
hostname  : [redacted]

Last login: Tue Oct 10 13:41:33 2017 from 193.50.75.114
[redacted]@sta1:~$
```

La saisie du nom de connexion et du mot de passe doit respecter scrupuleusement la casse des caractères (majuscules/minuscules). D'une manière générale, que ce soit dans un nom de fichier, un login, un nom de variable ou une commande, la lettre **a** n'est pas égale à la lettre **A**.

Exemple de connexion sur un terminal texte :

Invite shell (prompt)

Une fois connecté à un terminal texte, un programme - nommé shell - est lancé automatiquement. Il permet de saisir les commandes que nous évoquerons plus loin.

Le shell indique qu'il est en attente d'une instruction en présentant une invite (ou prompt) en début de ligne.

Suivant la configuration prédéfinie par l'éditeur de la distribution Linux, cette invite peut prendre plusieurs aspects, par exemple sur Red Hat, Fedora ou CentOS :

```
[nicolas@doe tmp]$
```

Sur Debian ou Ubuntu :

```
nicolas@doe:~$
```

Ou encore :

[root@doe bin]#

L'élément à retenir dans les invites que nous venons de présenter, est le dernier caractère de chacune : un **$** ou un **#** dans la plupart des cas. En effet, il indique que l'utilisateur actuel connecté est soit un utilisateur quelconque sans droits particuliers (**$**), soit l'administrateur qui possède tous les droits nécessaires à la configuration et à la maintenance du système (**#**).

Les autres informations présentées dans l'invite du shell sont le nom de l'utilisateur (avant le @), le nom de la machine (**doe**) et le nom du répertoire courant (**tmp**, **~** et **bin**).

Pour simplifier les exemples suivants, l'invite du shell sera réduite à ce dernier caractère (**$** ou **#**), éventuellement précédé du nom de l'utilisateur entre crochets :

$

ou :

[nicolas]$

Syntaxe des commandes

Il est important de connaître la syntaxe des commandes afin d'éviter un grand nombre d'erreurs de saisie.

Dans sa plus simple expression, sans option ou argument particulier, une commande est lancée en tapant son nom sur la ligne de commande :

$ commande

Rappel : dans les exemples suivants, le **$** situé en début de ligne correspond à l'invite du shell et ne doit pas être saisi par l'utilisateur.

Si des arguments doivent être spécifiés, ils sont ajoutés à la suite de la commande sur la ligne, séparés par un caractère d'espacement :

$ commande arg1 arg2

Dans la grande majorité des cas, les arguments sont les noms des fichiers sur lesquels agit la commande.

Les traitements ou la sortie (affichage) d'une commande GNU/Linux sont souvent modifiables à l'aide d'une option. Ces options, spécifiques à chaque commande, proviennent habituellement des implémentations originales et des apports des différents UNIX antérieurs à Linux. C'est pourquoi, pour une commande, il peut exister plusieurs dizaines d'options avec différentes syntaxes.

Il existe principalement deux types d'options : les options monocaractères et les options longues. Toutes deux doivent être saisies, séparées par un caractère d'espacement, à la suite de la commande et avant les éventuels arguments.

Le caractère d'espacement est un espace ou une tabulation et peut être répété plusieurs fois, que ce soit entre la commande, les options et les arguments, ou en début et fin de ligne.

Les options monocaractères

Les options monocaractères correspondent normalement aux options héritées de la famille UNIX et sont généralement introduites avec le caractère **-** :

```
$ commande -o -p -t
$ commande -o -p -t arg1 arg2
```

L'ordre des options n'est pas important, les deux lignes suivantes sont équivalentes :

```
$ commande -o -p -t
$ commande -t -o -p
```

Les options monocaractères peuvent être regroupées à condition de supprimer les tirets et les caractères d'espacements intermédiaires :

```
$ commande -o -p -t
$ commande -opt
```

Pour certaines commandes possédant un grand nombre d'options, les options monocaractères peuvent être introduites avec un + ou même directement saisies :

```
$ commande +o -pt
$ commande opt
```

Par conséquent, suivant la commande employée, ces trois lignes ne sont pas forcément équivalentes :

```
$ commande -o
$ commande +o
$ commande o
```

Enfin, certaines options acceptent une chaîne de caractères en paramètre ; dans ce cas, il n'y a pas forcément d'espace entre l'option et son paramètre :

```
$ commande -oparametre -pt
```

D'une manière générale, les options monocaractères introduites avec un **-** proviennent de commandes développées pour les systèmes répondant à la norme Unix98 tandis que celles ne nécessitant pas de tiret sont issues des UNIX de la famille BSD.

Les options longues

Les options longues sont généralement des options ajoutées dans le cadre de la réécriture de la commande GNU. Elles sont plus explicites que les précédentes car leur nom indique leur utilité et elles sont introduites par deux tirets **--** :

```
$ commande --help
$ commande --version
```

Dans le cas d'une option avec paramètre, ceux-ci sont séparés par un espace :

```
$ commande --variable parametre
```

Séparation des options et des arguments

Suivant leur syntaxe, faire la distinction entre options et arguments peut s'avérer difficile.

Si une option **a** est introduite par un tiret et la chaîne de caractères **-b** en argument, la syntaxe suivante peut porter à confusion :

```
$ commande -a -b
```

Dans cet exemple, la chaîne de caractères **-b** sera traitée comme l'option **b** introduite par un tiret et non comme un argument.

Pour enlever toute ambiguïté, on sépare les options des arguments sur la ligne de commande en intercalant deux tirets ; pour le problème précédent, cela donne alors :

```
$ commande -a -- -b
```

Dès que le shell rencontre **--** sur la ligne de commande, il interprète toutes les chaînes de caractères suivantes comme des arguments de la commande.

Applications avec interfaces graphiques

Il est naturel de lancer les outils graphiques en utilisant les raccourcis comme les icônes et les menus offerts par l'environnement de bureau.

Cependant, toutes ces applications peuvent être instanciées à partir d'une commande saisie en ligne dans un émulateur de terminal. Ainsi la ligne suivante - de la même manière que l'icône correspondante - permet de lancer le navigateur web Firefox :

```
$ firefox
```

Il est intéressant de connaître le nom des commandes graphiques et de les lancer de cette manière. En effet, il n'est pas évident que les mêmes raccourcis existent d'un environnement de bureau à un autre et d'une distribution à une autre.

Les options des commandes avec interface graphique présentent une syntaxe différente des autres, à savoir : une chaîne de caractères introduite par un seul tiret ; par exemple, pour afficher l'horloge avec un fond rouge :

```
$ xclock -bg red
```

Raccourcis-clavier

1. En mode texte

Les raccourcis présentés ici sont propres au shell Bash qui est utilisé par défaut sous Linux ; ils fonctionnent aussi bien sur une console virtuelle texte que sur un émulateur de terminal.

Contrôle de l'affichage

[Ctrl]+[l]	Efface l'affichage et repositionne l'invite du shell sur la première ligne du terminal.
[Maj]+[Page préc]	Remonte d'une demi-page dans l'affichage du terminal (de la même manière qu'un ascenseur) ; très pratique pour visualiser le résultat d'une commande un peu longue. Il est possible de remonter de cinq à six pages suivant la configuration du système.
[Maj]+[Page suiv]	À l'inverse du raccourci précédent, descend d'une demi-page dans l'affichage du terminal.

Édition de la ligne de commande

[Deb]	Déplace le curseur en début de ligne.
[Fin]	Déplace le curseur en fin de ligne.

[Gauche]	Déplace le curseur d’un caractère vers la gauche.
[Droite]	Déplace le curseur d’un caractère vers la droite.
[Suppr]	Supprime le caractère à droite du curseur.
[RetArr]	Supprime le caractère à gauche du curseur.
[Ctrl]+[w]	Efface le dernier mot.
[Ctrl]+[u]	Efface la ligne entière.

Historique de commandes

[Haut]	Remonte dans l’historique de commandes. Chaque pression sur cette touche permet d’afficher la commande précédente parmi celles qui ont déjà été lancées par le shell ; il est alors possible de modifier la commande avant de l’exécuter de nouveau.
[Bas]	Descend dans l’historique de commandes. Cette action est valable uniquement si l’utilisateur est préalablement remonté dans l’historique avec le raccourci précédent.
[Ctrl]+[r]	Recherche une chaîne de caractères dans l’historique de commandes. La recherche s’effectue à mesure que l’utilisateur saisit les caractères de la chaîne. Si la chaîne est contenue dans plusieurs commandes de l’historique, une pression supplémentaire sur **[Ctrl]+[r]** permet de poursuivre la recherche en remontant encore dans l’historique.
[Ctrl]+[j] ou **[Echap]**	Termine une recherche initiée avec **[Ctrl]+[r]** et permet à l’utilisateur de modifier la ligne avant de l’exécuter.
[Ctrl]+[g]	Annule une recherche initiée avec **[Ctrl]+[r]**.

Divers

[Ctrl]+[c]	Interrompt la commande en cours sans attendre la fin de son exécution normale.
[Ctrl]+[d]	Envoie le caractère de fin de fichier (EOF ou End Of File). Signifie à la commande en cours que la saisie au clavier est terminée.
[Ctrl]+[s]	Marque une pause dans l'affichage du terminal (équivalent à une pression sur la touche **[Arrêt défil]**).
[Ctrl]+[q]	Relance l'affichage d'un terminal mis en pause précédemment (équivalent à une seconde pression sur la touche **[Arrêt défil]**).

2. En mode graphique

Les raccourcis-clavier disponibles sous l'interface graphique dépendent du gestionnaire de fenêtres et de l'environnement de bureau utilisé.

Les environnements les plus connus à ce jour sont KDE (K Desktop Environment) et GNOME (GNU Network Object Model Environment) ; leur évolution rapide et la diversité des applications disponibles sous GNU/Linux ne permettent pas d'énumérer les nombreux raccourcis-clavier mis en œuvre.

Premières commandes

Ces premières applications fournissent un bon exemple d'utilisation de la ligne de commande Linux tout en permettant d'effectuer des opérations essentielles sur le système ; certaines d'entre elles seront d'ailleurs réutilisées ultérieurement dans ce livre.

1. Identité des utilisateurs : who, whoami, finger

La commande **who** liste tous les utilisateurs connectés actuellement sur le système :

```
[nicolas]$ who
nicolas  vc/1        Apr  3 01:04
root     pts/0       Apr  2 22:42 (192.168.200.1)
```

Sur cet exemple, l'utilisateur **nicolas** est connecté sur la première console virtuelle texte (**vc/1** ou **tty1**) depuis le 3 avril à 1h04 et l'administrateur (**root**)

est connecté, à partir d'un émulateur de terminal (**pts/0**) distant (**192.168.200.1**), depuis le 2 avril à 22h42.

Avec l'option **-q**, la même commande **who** liste uniquement les noms de connexion et fait le total du nombre d'utilisateurs connectés actuellement :

```
[nicolas]$ who -q
nicolas root
# usager=2
```

Par contre, la commande **who am i** qui possède une syntaxe bien particulière - au demeurant très appréciée des producteurs cinématographiques en mal de systèmes informatiques comprenant le langage naturel - affiche uniquement la ligne concernant l'utilisateur connecté :

```
[nicolas]$ who am i
nicolas  vc/1        Apr  3 01:04
```

La commande **whoami**, quant à elle, indique l'identité sous laquelle se trouve actuellement l'utilisateur :

```
[nicolas]$ whoami
nicolas
```

Concernant toujours les utilisateurs du système, la commande **finger** affiche une description plus précise d'un compte :

```
$ finger
Login    Name          Tty    Idle  Login Time   Office   Office Phone
nicolas  Nicolas dupond   vc/1   1  Apr  3 01:04
root     root          pts/0  Apr  2 22:42 (192.168.200.1)
```

La colonne **Idle** affiche notamment le temps d'inactivité des utilisateurs (en minutes, si aucune unité n'est précisée).

Pour obtenir encore plus de précisions sur un utilisateur, on indique son nom de connexion en argument de la commande **finger** :

```
$ finger nicolas
Login: nicolas                  Name: Nicolas dupond
Directory: /home/nicolas              Shell: /bin/bash
On since Mon Apr  3 01:04 (CEST) on tty1   1 minute 12 seconds idle
No mail.
No Plan.
```

En plus des informations de connexion de l'utilisateur **nicolas**, on constate ici que :

- Son nom réel est "Nicolas dupond".
- Son répertoire personnel est /home/nicolas.
- Son shell par défaut est /bin/bash.

- Cet utilisateur n’a pas de message électronique dans sa boîte aux lettres.
- Rien n’est indiqué dans son fichier personnel .plan.

Les utilisateurs inscrivent dans leur fichier .plan ("planning") leurs rendez-vous, congés et autres informations figurant dans leur agenda.

2. Changement de mot de passe : passwd

La commande **passwd** permet à l’utilisateur de modifier son mot de passe. À la différence de la commande précédente, celle-ci est interactive et demande à l’utilisateur de saisir son ancien mot de passe avant de taper deux fois le nouveau :

```
[nicolas]$ passwd
Changing password for nicolas
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Les mots de passe n’apparaissent pas lors de la saisie et le système exige la saisie d’un mot de passe valide ; suivant les règles établies par l’administrateur, le mot de passe devra comprendre un minimum de six caractères tout en étant composé de lettres (minuscules et/ou majuscules) et de chiffres.

Seul l’administrateur du système est autorisé à modifier le mot de passe d’un autre utilisateur en spécifiant le login du compte utilisateur en argument sur la ligne de commande.

3. Comptage : wc

De l’anglais "Word Count", la commande **wc** permet de compter le nombre de lignes, de mots et de caractères contenus dans un fichier :

```
[nicolas]$ wc /etc/issue
 2 14 82 /etc/issue
```

Les options **-l**, **-w** et **-c** permettent respectivement de ne compter que le nombre de lignes, de mots et de caractères contenus dans le fichier :

```
[nicolas]$ wc -l /etc/issue
2 /etc/issue
[nicolas]$ wc -w /etc/issue
14 /etc/issue
[nicolas]$ wc -c /etc/issue
```

82 /etc/issue

4. Affichage : clear, echo

La commande **clear** nettoie l'écran (la fenêtre virtuelle) :

```
[nicolas]$ clear
```

La commande **echo** retourne la chaîne de caractères reçue en arguments :

```
[nicolas]$ echo petit bonjour de Montpellier
petit bonjour de Montpellier
```

Pour être plus exact, elle retourne tous ses arguments séparés par un seul espace :

```
[nicolas]$ echo cette   fois il     y a   plusieurs espaces   entre
les  arguments
cette fois il y a plusieurs espaces entre les arguments
```

Bien que très sommaires, ces deux dernières commandes trouveront leur utilité lors de l'écriture de scripts shell.

5. Temps : date, cal

Comme son nom l'indique, la commande **date** indique l'heure du système :

```
[nicolas@localhost ~]$ date
lun. févr. 22 15:06:14 CET 2016
```

Il est néanmoins possible de formater l'affichage de celle-ci, par exemple :

```
[nicolas@localhost ~]$ date +"nous sommes le %x"
nous sommes le 22/02/2016
```

La commande **cal** affiche un calendrier. Si elle est appelée seule, c'est le calendrier du mois courant qui est affiché :

```
[nicolas@localhost ~]$ cal
    février 2016
lu ma me je ve sa di
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29
```

Suivie d'un seul argument, la commande affiche le calendrier complet de l'année mentionnée.

Enfin, si la commande **cal** est invoquée avec deux arguments, le premier est le

mois à afficher et le second, l’année concernée :

```
[nicolas]$ cal 6 2008
    juin 2008
di lu ma me je ve sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
```

L’année mentionnée en argument doit être totalement définie. Aussi, l’année 08 correspond bien à l’an 8 après JC et non à 2008, comme on pourrait le supposer !

Déconnexion

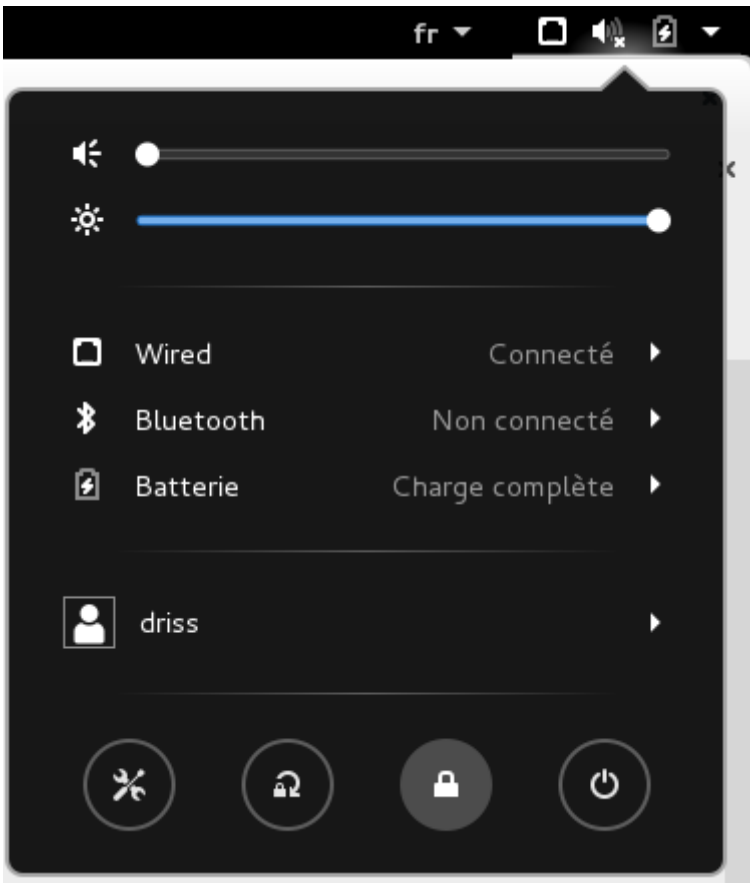
Une fois que l’utilisateur a terminé le travail pour lequel il s’est connecté au système Linux, il doit se déconnecter (ou fermer sa session) afin de libérer la console qu’il utilise.

La déconnexion d’un terminal texte peut s’effectuer de trois manières :

- avec la commande **exit** ;
- avec la commande **logout** (ne fonctionne qu’à partir du shell lancé automatiquement à la connexion) ;
- en tapant simultanément sur les touches **[Ctrl]+[d]**.

La séquence de touches **[Ctrl]+[d]** correspond au caractère de fin de fichier. Lorsque le terminal reçoit ce caractère, il considère que son "fichier" d’entrée (le clavier) est fini et donc, termine son exécution. Cette notion de fichier d’entrée est détaillée dans la section Redirections du chapitre traitant du shell Bash.

Les interfaces graphiques Linux permettent, quant à elles, de se déconnecter via un menu :



Exercices

Exercice 1

Connectez-vous à un terminal virtuel texte, une console graphique ou un terminal distant et lancez un shell si ce n’est pas déjà fait.

Exercice 2

Modifiez votre mot passe.

Exercice 3

Listez les utilisateurs connectés au système.

Exercice 4

Affichez le calendrier de l’année en cours puis effacez l’écran.

Exercice 5

Déconnectez-vous.